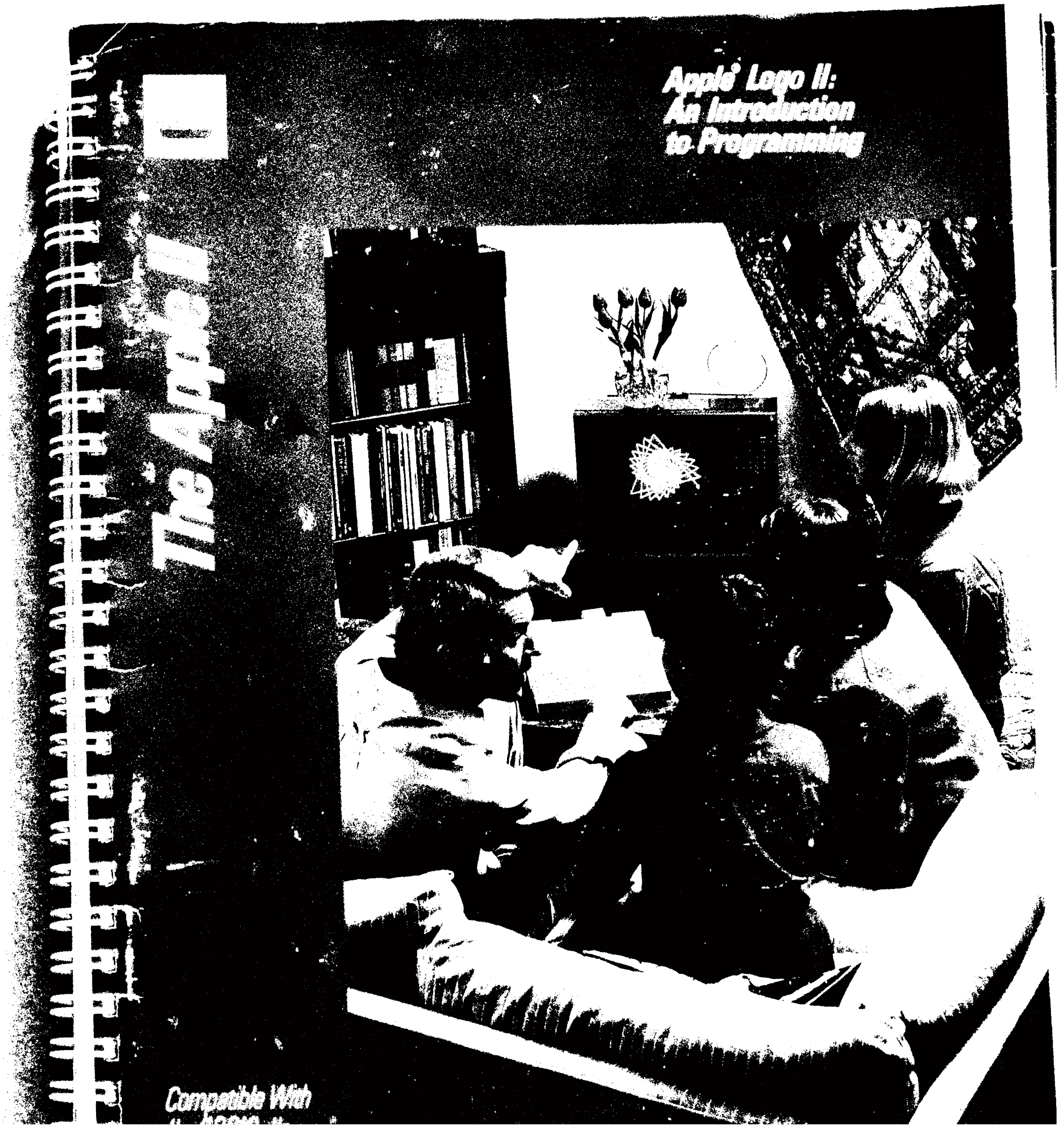


Apple® Logo II: An Introduction to Programming

The Apple II

Compatible With

The Apple II

Compatible with
IIe (128K), IIc.

Apple Logo II: An Introduction to Programming

Table of Contents

Read Me First

You are about to start a journey into the world of Apple Logo II—a computer language that is much more than a computer language. Before you begin, make sure you have everything you require.

To use Apple Logo II, you need

- an Apple II computer with at least 128K of memory
- a video monitor or television set (color not required)
- at least one disk drive (may be built in to your Apple)
- the Apple Logo II program disk
- a blank disk to save copies of your work.

Your first steps into Logo can be guided by an interactive training program. It's on the disk labeled *Apple Presents APPLE LOGO II*. You can use the training disk with the equipment listed here. You'll find instructions for starting it on the first two pages of Chapter 1. If you wish, you can also start with the section of Chapter 1 of this manual called "The Apple Logo II Disk."

After you go through the training disk, use this manual to review and add to your Logo skills. You will understand Logo better if you plan to do Chapters 1 through 4 in one sitting. Chapter 4 teaches you how to save on a disk the work you have done up to that point.

Follow the explanations and directions carefully. Occasionally, you'll find a **bug box** to assist with unexpected difficulties. If you ever feel like leaving the book and striking out on your own, don't hesitate to do so. That's the true spirit of Logo—play, exploration, and learning.

This manual introduces you to only some of Logo's features, namely turtle graphics and sound. For a complete description of Apple Logo II, see the *Apple Logo II Reference Manual*.

Bon voyage!

Getting Started

Getting Started

chapter 1

Now you begin your adventure with Logo.

What Is Logo?

Logo is a language for learning—for learning about your computer and for learning about the world outside your computer. Using Logo, you learn important programming concepts. Exploring Logo's abilities to create pictures and produce sounds, you encounter powerful ideas from mathematics, science, and the arts.

If you are a beginner, Logo makes it easy to get dramatic and interesting results quickly. If you are an advanced programmer, Logo offers the power of a complete computer language.

The Logo Training Disk

The *Apple Presents APPLE LOGO II* disk introduces you to the philosophy of Logo and some of its basic features.

The disk has two sides. Start with Side 1 and, if you wish, go on to Side 2.

With your Apple turned off, follow these steps:

1. Hold the disk with your thumb on the label of the side you want to use. (Start with Side 1.)

2. Put the disk in drive 1 and close the door.

3. Turn on your monitor or TV.

4. Turn on your Apple. In a few moments, a menu appears on the screen.

5. Select the Introduction and follow the directions.

Bug Box
If you do not see a screen display of any sort, it may be that

- The video monitor or TV set is still off;
- The contrast and brightness of the monitor or TV are turned down;
- The monitor is not properly connected to the computer;
- The disk drive door is not closed all the way.

If you see the message `I'M HAVING TROUBLE WITH THE DISK`, it may be that

- The disk is not in the drive properly;
- The door to the disk drive is not closed all the way;
- The disk is damaged;
- The computer itself is not functioning properly.

The Apple Logo II Disk

Now you're going to start up Logo itself and then follow the hands-on steps in this tutorial manual.

To start up Logo:

1. If you haven't already, take the Logo training disk out of the disk drive and put it back in its protective envelope.

2. Put the Apple Logo II program disk in drive 1.

3. Turn on the monitor or TV, if it isn't already on.

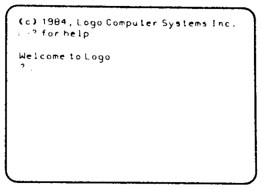
4. If your Apple is on, hold down the (ó) key (the one with an outline of the Apple) and then press the (CONTROL) and (RESET) keys. If your Apple is off, turn it on.

   In a few moments, you should see on the screen the name `APPLE LOGO II` and the message

```
Press RETURN to start Logo
Press (ó)-S to see what Logo can do
```

5. Press (RETURN).

Sneak Preview: If you'd like to see how Logo works by running some sample programs, hold down the (ó) key and type S (for sample) at the same time. Logo now gives you a demonstration.



```
(c) 1984, Logo Computer Systems Inc.
(ó)-? for help

Welcome to Logo
?
```

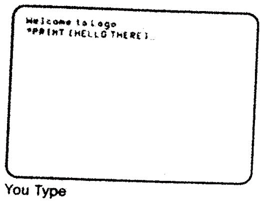
The Welcome Screen

Now you see the Logo welcome message and the Logo ? (**prompt**) and blinking _ (**cursor**). The ? (prompt) means that Logo is waiting for you to type a command, and the _ (cursor) shows you where your first typed character will appear.

A Note for IIe Owners: If you are using an Apple IIe, the (ó) character shown on the welcome screen may appear on your screen as a black letter A on a light-colored rectangle. Whenever you see this on your screen, it stands for the (ó) key.

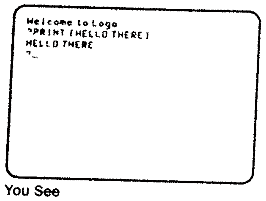
Bug Box
If Logo does not start up, check the list of possible bugs in the previous bug box.

Your First Logo Command



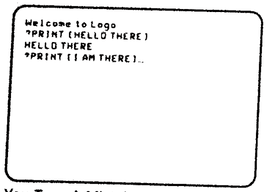
```
Welcome to Logo
?PRINT [HELLO THERE]_
```

You Type



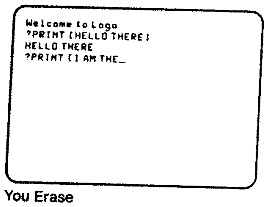
```
Welcome to Logo
?PRINT [HELLO THERE]
HELLO THERE
?_
```

You See



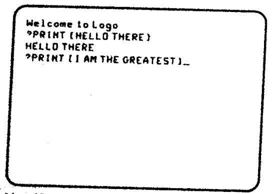
```
Welcome to Logo
?PRINT [HELLO THERE]
HELLO THERE
?PRINT [I AM THERE]_
```

You Type A Mistake



```
Welcome to Logo
?PRINT [HELLO THERE]
HELLO THERE
?PRINT [I AM THE_
```

You Erase



```
Welcome to Logo
?PRINT [HELLO THERE]
HELLO THERE
?PRINT [I AM THE GREATEST]_
```

You Type It Right

You're all set! A good way to start is to jump in and swim. Try your first Logo command by typing these next words on the keyboard just as you see them here:

```
PRINT [HELLO THERE]
```

The instruction appears on the screen but nothing happens until you press (RETURN). Press (RETURN) now. Logo responds

```
HELLO THERE
```

Suppose you wanted Logo to print another message, like

```
I AM THE GREATEST
```

but you made a mistake and typed

```
PRINT [I AM THERE]
```

OK, hold on. Don't press (RETURN) yet.

Press (DELETE) for each character you want to erase until the screen shows

```
PRINT [I AM THE
```

Now type the rest of the line—but be sure to type a space (using the (SPACE) key) after THE.

```
PRINT [I AM THE GREATEST]
```

Press (RETURN) and Logo obeys your instruction.

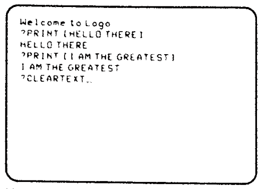
```
I AM THE GREATEST
```

Editing Keys: (DELETE) is one of Logo's editing keys that lets you change what you have typed without having to type the entire instruction over again. You will be introduced to more editing keys in later chapters of this manual.

Ask Logo to print some other messages. Type PRINT and then put in square brackets whatever you want to see on the screen. You can also type the short form of this

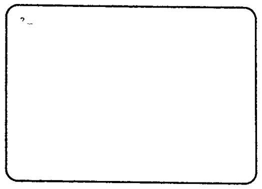
command, PR. (Most Logo commands have a short name that saves you time and accomplishes the same thing as the long name.)

Clearing the Screen



```
Welcome to Logo
?PRINT [HELLO THERE]
HELLO THERE
?PRINT [I AM THE GREATEST]
I AM THE GREATEST
?CLEARTEXT_
```

You Type



```
?_
```

You See

A useful command is CLEARTEXT (or CT). It clears the screen of text and lets you start new text at the top of the screen. Type

```
CLEARTEXT                Or type CT
```

and press (RETURN).

The screen clears and the ? (prompt) and _ (cursor) appear at the top of a fresh screen.

Writing a Procedure

A Logo word like PRINT or CLEARTEXT is called a Logo **primitive**. It is a **procedure**, or little computer program, built into Logo for your use. You can add to Logo's vocabulary by **defining** new procedures. To define a procedure you type in a set of Logo commands preceded by the primitive TO and END.

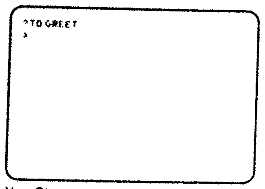
Now you'll write your first Logo procedure. You use the command TO to signal Logo you intend to define a new procedure. On the same line, you tell Logo what name you want the procedure to go by. Then you tell Logo what you want the procedure to do. For example, you can define a procedure, GREET, so that whenever you type GREET, Logo responds

```
HI THERE
BYE NOW
```

To begin writing the procedure, type `TO` followed by the name of the procedure.

`TO GREET` — Type a space to separate the words TO and GREET.

Press RETURN.

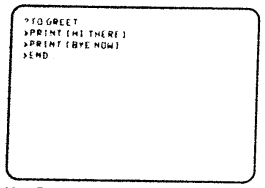


```
?TO GREET
>
```

You Start

Logo now uses > instead of ? as the prompt symbol. This reminds you that you are defining a procedure and not typing instructions to be carried out right away.

Bug Box
If you make a typing goof while entering the lines of the procedure, simply press -ESCAPE. Then start again by typing the `TO GREET` line and pressing RETURN.



```
?TO GREET
>PRINT [HI THERE]
>PRINT [BYE NOW]
>END
```

You Finish

Now type the rest of the procedure.

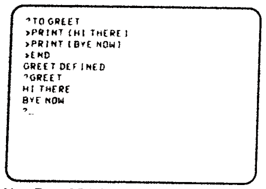
```
PRINT [HI THERE]
PRINT [BYE NOW]
END
```

Press RETURN each time.

The word END signals Logo that you have finished defining the procedure. Logo now prints

```
GREET DEFINED
```

and returns to ? as the prompt symbol.



```
?TO GREET
>PRINT [HI THERE]
>PRINT [BYE NOW]
>END
GREET DEFINED
?GREET
HI THERE
BYE NOW
?_
```

You Run GREET

Any time you want to try a procedure after defining it, you **run** or **execute** it. Run your first procedure by typing

```
GREET
```

and pressing RETURN. Logo greets you:

```
HI THERE
BYE NOW
```

Before you defined the procedure, Logo didn't know how to GREET. Now it does—because you added a primitive to Logo's vocabulary.

Bug Box
Suppose GREET does not work. Perhaps you typed something strange. Soon you will learn how to **edit** your procedure—that is, to change the parts you don't like. In the meantime, if GREET doesn't work, start over. Define the procedure again, but give it a different name—NEWGREET.

```
TO NEWGREET
PR [HI THERE]
PR [BYE NOW]
END
```

Remember the space.
Remember RETURN.

Run this procedure by typing

```
NEWGREET
```

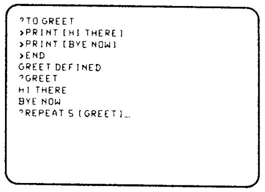
and pressing RETURN. Logo should respond

```
HI THERE
BYE NOW
```

You can use the command REPEAT to run GREET over and over. For example, you can have REPEAT run GREET five times.

`REPEAT 5 [GREET]` — Type a space between REPEAT and 5.

Press RETURN.

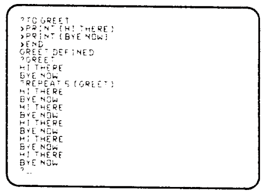


```
?TO GREET
>PRINT [HI THERE]
>PRINT [BYE NOW]
>END
GREET DEFINED
?GREET
HI THERE
BYE NOW
?REPEAT 5 [GREET]_
```

Your Instruction

(If you have a NEWGREET procedure, you can use it instead of GREET.)

Notice what happens.



```
?TO GREET
>PRINT [HI THERE]
>PRINT [BYE NOW]
>END
GREET DEFINED
?GREET
HI THERE
BYE NOW
?REPEAT 5 [GREET]
HI THERE
BYE NOW
HI THERE
BYE NOW
HI THERE
BYE NOW
HI THERE
BYE NOW
HI THERE
BYE NOW
?_
```

Logo's Response!

You could have REPEAT run GREET a thousand times. Go ahead, try it.

```
REPEAT 1000 [GREET]
```

If you want to stop this overdone greeting, press -ESCAPE. That is, hold down the key and press ESCAPE at the same time.

Bug Box
If (⌘)-(ESCAPE) doesn't stop Logo from printing on the screen, make sure you are typing the key combination correctly. While holding down (⌘), press (ESCAPE). Of course, you can wait until REPEAT is finished.

In the next chapters of this manual, we will not always remind you to press (RETURN) or to type a space between words.

Logo Vocabulary

These are the Logo primitives you learned in this chapter:

Full Name	Short Name
PRINT	PR
CLEARTEXT	CT
TO	
REPEAT	

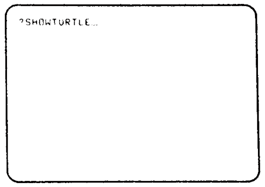
Special Keys

You learned to use a few special keys in this chapter:

(RETURN)	(SPACE)
(DELETE)	(⌘)-(S)
[	(⌘)-(ESCAPE)
]	

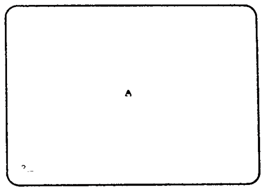
Meet the Turtle

Meet the Turtle



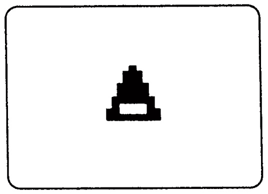
```
?SHOWTURTLE_
```

You Type



```
?_
```

You See



Turtle Up Close

This chapter introduces you to programming by helping you learn to control on the screen a creature known as a **turtle**. Why is it called a turtle? It has to do with Logo's history.

The first device that carried out Logo graphics instructions was a robot that resembled a turtle. It was a big cannister that moved about on wheels and was attached to the computer by a long cable. It drew lines on the floor—which normally was covered with paper—in response to Logo commands.

Now the Logo turtle lives on the computer screen. It too has a pen of sorts. It can write and erase lines on the screen. The pictures you create with the turtle are called **turtle graphics**. Logo has many commands that you can use to control the turtle. This chapter introduces you to some of the most important turtle commands.

To see the turtle, type the command

`SHOWTURTLE` Or the short form, `ST`.

Press (RETURN).

The turtle appears on the screen as a triangle. At any given time, the graphics turtle is at a specific place (its **position**) and is pointing with its shaded point in a specific direction (its **heading**). The position and heading are called the turtle's **state**. The most important turtle commands are those that change its state.

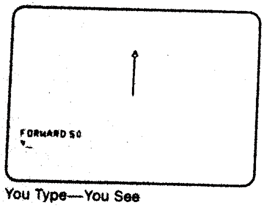
At the start, the turtle is in the center of the screen heading straight up. You can think of this as its "home" state.

Prompt and Cursor Location: Notice also that the ? (prompt) and the _ (cursor) are now near the bottom of the screen. The commands you type appear on the lowest four lines of the screen.

Changing the Turtle's State

There are several important commands that affect the turtle's state.

FORWARD



```
FORWARD 50
?_
```

You Type—You See

You move the turtle straight forward in the direction it is pointing by using the FORWARD command. FORWARD takes an **input**. An input is a value some commands need to complete their work. Here the input is a number telling how many steps the turtle should move.

Type the following command, remembering to press RETURN when you want Logo to "do it!"

`FORWARD 50` Or the short form, `FD 50`.

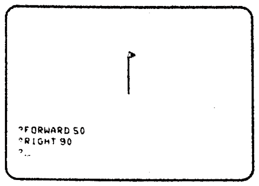
Although you used 50 as the input, you can choose any number.

Bug Box

The space between FORWARD and 50 is very important. Without the space, Logo does not recognize the word. You must always put a space between a command and its input. If by accident you put in an extra space or two, that's all right. Logo ignores the extra spaces.

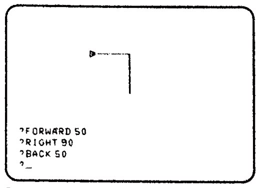
Also, Logo does not carry out your command if the command word is misspelled or if you forget to type in an input value. Always check your typing!

Notice that when you use FORWARD, the turtle changes its position but not its heading.



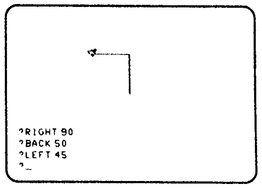
```
?FORWARD 50
?RIGHT 90
?_
```

RIGHT 90



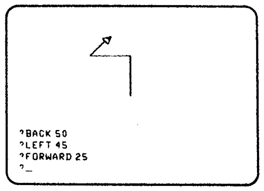
```
?FORWARD 50
?RIGHT 90
?BACK 50
?_
```

BACK 50



```
?RIGHT 90
?BACK 50
?LEFT 45
?_
```

LEFT 45



```
?BACK 50
?LEFT 45
?FORWARD 25
?_
```

FORWARD 25

RIGHT

To change the turtle's heading, you tell it to turn right or left a specified number of degrees—any number of degrees.

In the following example, you tell the turtle to turn right 90 degrees:

`RIGHT 90` Or `RT 90`

The turtle turns 90 degrees to the right of where it was headed previously. Notice the turtle changes its heading (its direction), not its position on the screen.

BACK

BACK is similar to FORWARD except that the turtle backs away from its current position. BACK changes the turtle's position only. Type

`BACK 50` Or `BK 50`

The number 50 is again a sample input—you can choose any number.

LEFT

LEFT is similar to RIGHT except that the turtle turns in the opposite direction. Type the command

`LEFT 45` Or `LT 45`

The turtle turns 45 degrees to the left of where it was heading. It changes only its heading, not its position. You can see the effect of the turn more clearly if you now tell the turtle to go

`FORWARD 25` Or `FD 25`

HIDETURTLE and SHOWTURTLE

Sometimes you don't want the turtle showing in the middle of your drawing. Type

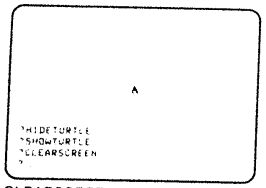
```
HIDETURTLE                Or HT
```

HIDETURTLE makes the turtle invisible. Also, it speeds up the turtle as it draws.

You can make the turtle visible again. Type

```
SHOWTURTLE                Or ST
```

Clearing the Graphics Screen



```
?HIDETURTLE
?SHOWTURTLE
?CLEARSCREEN
?
```

CLEARSCREEN

You might want to clear the screen and start again. The command CLEARSCREEN does that. It erases current turtle tracks from the screen and puts the turtle at its startup or home state in the center of the screen heading straight up.

```
CLEARSCREEN               Or CS
```

Try experimenting with these state change commands on your own.

Bug Box
If you get lost, you can always use CLEARSCREEN to start over. Or you can press ()-(?) to get help. Actually, this is a good time to try this particular help function. Hold down both the () key and the (SHIFT) key and at the same time type ?. Now Logo displays one of the available help screens. To move the text up or down on the screen, press the (↑) or (↓) key. To leave the help screen and return to the Logo ? (prompt) and _ (cursor), press ()-(ESCAPE).

Logo Vocabulary

These are the Logo primitives you learned in this chapter:

Full Name	**Short Name**
FORWARD	FD
RIGHT	RT
BACK	BK
LEFT	LT
HIDETURTLE	HT
SHOWTURTLE	ST
CLEARSCREEN	CS

Special Key

You learned to use a special key in this chapter:

()-(?)

A Procedure to Draw a Square

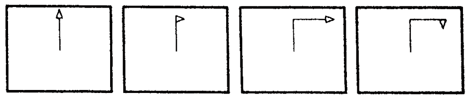
A Procedure to Draw a Square

Using the commands FORWARD and RIGHT or LEFT, you can make the turtle draw a square. This chapter tells you how to define a square and use it in other procedures.

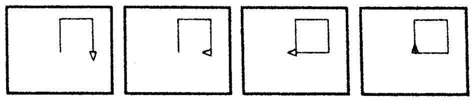
Defining SQUARE

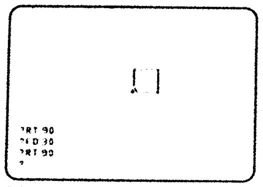
For convenience, now use the short names for commands. Type these instructions, pressing (RETURN) at the end of each line:

```
FD 30
RT 90
FD 30
RT 90
```



```
FD 30
RT 90
FD 30
RT 90
```





```
?RT 90
?FD 30
?RT 90
?
```

SQUARE

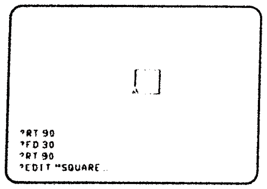
You can change the size of the square by changing 30 to any other number. For example, if you use 50 instead of 30, the turtle draws a larger square.

Now, define a new Logo procedure to get the turtle to draw a square. Then each time you want to draw a square, you can use your new procedure rather than retype the individual instructions.

To define a new procedure, you first choose a name, as you did for GREET in Chapter 1. Use SQUARE for this procedure because that seems practical. But any name will do.

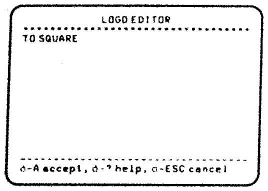
At this point, you could use the Logo command TO and define SQUARE in the same way as you did GREET. That is, you could type `TO SQUARE`, then the instructions, then `END`. This method is good if you are careful about typing and you know exactly what you want to type in advance. But there is also another way.

Introducing the Logo Editor



```
?RT 90
?FD 30
?RT 90
?EDIT "SQUARE
```

You Type



```
                LOGO EDITOR
..................................
TO SQUARE




----------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

You See

Another way to define a procedure is to use the Logo **editor**. This way, if you make typing mistakes, you can remove them more easily than you can when using the TO command. When you are using the editor, Logo carries out only editing actions; it does not run a procedure.

EDIT or ED signals Logo that you want to edit. Type this command and follow it with the name of the procedure you want to edit. You must prefix the name with " (quotation mark). Do not type a space between " and the name of the procedure. Type

```
EDIT "SQUARE
```

Remember the " (quotation mark).

After you press (RETURN), you are using the Logo editor and only editing actions are carried out.

Bug Box

If you don't remember to prefix SQUARE with " (quotation mark) and instead type

```
EDIT SQUARE
```

Logo responds

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

If you have turtle drawings on the screen, they disappear when you start editing. But they are there when you exit the editor.

When the editor starts up, you see the words `LOGO EDITOR` and right below that the title line of the procedure:

`TO SQUARE` This is the title line.

`TO` informs Logo that the text that follows is part of a procedure.

`SQUARE` is the name of the procedure.

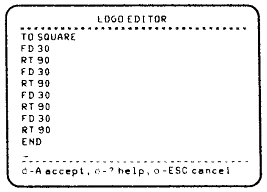
The _ (cursor) is at the beginning of the title line.

The message at the bottom of the screen reminds you of three useful keyboard commands.

Notice that Logo does not print ? or any other prompt symbol while you are using the editor.

You could change the title line now, but it is not a good idea. The title SQUARE is referred to in several places throughout this manual.

Now press (↓).



```
                LOGO EDITOR
..................................
TO SQUARE
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
_
----------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

SQUARE Defined

Type in the commands that make up SQUARE. They are the commands you used previously.

```
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
```

Making Changes While in the Editor: Think of everything you type in the editor as a stream of characters. If you want to add something to the stream, you move the _ (cursor) to that place.

If you want to move the _ (cursor) right in the stream, press (→). You can move all the way to the end of the stream, to the position after the E in the title line, by pressing (→) several times.

If later you want to move the _ (cursor) left, press (←).

When the cursor is where you want it, you type the characters you wish to add. They become part of the stream.

So if you made a typing error in a previous line, use an arrow key to move the cursor to where the error is. As the cursor passes under characters already typed, they remain unchanged.

If you want to erase a character, use (DELETE). (DELETE) erases the character to the left of the cursor and moves the cursor to the position that was occupied by that character.

Notice that if the cursor is at the beginning of a text line, pressing (DELETE) moves it up with the entire line of text to the end of the previous text line. So

```
FORWARD 30
RIGHT 90
```

becomes

```
FORWARD 30RIGHT 90
```

if you press (DELETE).

Then you can press (RETURN) to separate the lines again.

```
FORWARD 30
RIGHT 90
```

If you press (SPACE) instead of (RETURN), then you see

```
FORWARD 30 RIGHT 90
```

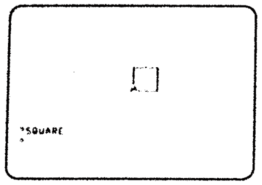
Pressing (RETURN) places the two commands on separate lines.

Press (ɑ)-(A) when you have completed typing in your commands. You do not need to type END. Logo inserts the word when you press (ɑ)-(A). As soon as you press (ɑ)-(A), Logo accepts what you have written, announces

```
SQUARE DEFINED
```

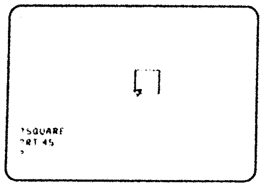
and returns you to the ? (prompt) and _ (cursor).

Using the New Command



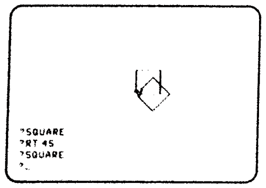
```
?SQUARE
?
```

SQUARE



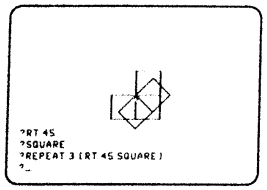
```
?SQUARE
?RT 45
?
```

RT 45



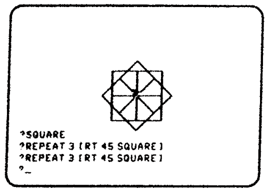
```
?SQUARE
?RT 45
?SQUARE
?_
```

Two Squares



```
?RT 45
?SQUARE
?REPEAT 3 [RT 45 SQUARE]
?_
```

Three More Squares



```
?SQUARE
?REPEAT 3 [RT 45 SQUARE]
?REPEAT 3 [RT 45 SQUARE]
?_
```

Final Three Squares

Try your new command. Type

```
SQUARE
```

Whenever you run SQUARE, Logo carries out the orders given in each line as though you were just then typing each line.

Again, type

```
SQUARE
```

This time the turtle simply retraces its path.

If you turn the turtle left or right and then type SQUARE, a new drawing appears. For example, tell the turtle

```
RIGHT 45
```

and then type

```
SQUARE
```

Continue to repeat these two commands. To do this you can use the Logo command REPEAT. REPEAT requires two inputs. For example,

```
REPEAT 3 [RT 45 SQUARE]
```

The first input (3) indicates how many times to repeat the listed instructions. The second input ([RT 45 SQUARE]) is a list of instructions. The instructions must be enclosed in square brackets, which you can think of as making an envelope.

Type again

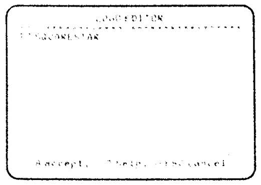
```
REPEAT 3 [RT 45 SQUARE]
```

Now, define a procedure for this design and call it SQUARESTAR. Type

```
EDIT "SQUARESTAR
```

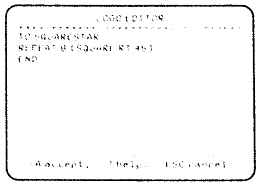
You are now using the editor and the title line should be displayed on the screen:

```
TO SQUARESTAR
```



```
LOGO EDITOR
TO SQUARESTAR

ⓐ-A accept,  ⓐ-? help,  ESC cancel
```

Ready to Define



```
LOGO EDITOR
TO SQUARESTAR
REPEAT 8 [SQUARE RT 45]
END

ⓐ-A accept,  ⓐ-? help,  ESC cancel
```

Definition Complete

Notice that the _ (cursor) is at the beginning of the title line. There is no need to change the title line, so press (→) several times to move the cursor to the end of the line. When the cursor is in the position right after the R, press (RETURN) to go to the next line. Now type

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

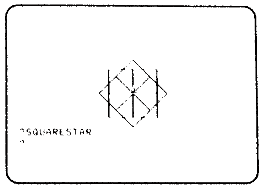
Press (RETURN) after typing END. Don't forget, press (ⓐ)-(A) when you've finished defining your new procedure.

It's always a good idea to try out a new procedure. Show the turtle screen and put the turtle in its home state in the center of the screen pointing straight up. To do this, type

```
CS
```

followed by (RETURN). Then to run the new procedure, type

```
SQUARESTAR
```



```
?SQUARESTAR
?
```

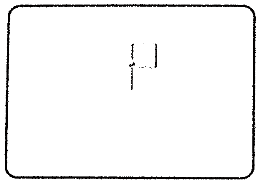
SQUARESTAR

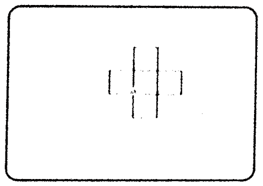
▲ **Warning**

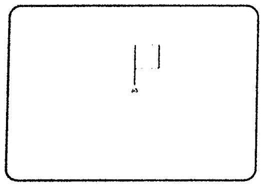
Don't turn off your Apple now or at the end of this chapter. If you do, all your procedures in the ***workspace****, some of which you will need later, will be permanently lost. (The Logo workspace is that part of your Apple's memory where procedures are stored. It is a temporary storage place; every time you turn off the Apple or use (CONTROL)-(RESET), the workspace is emptied.) In the next chapter, you'll learn how to save your procedures on a disk.*

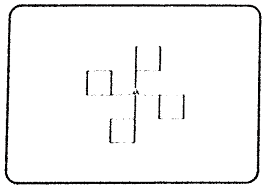
Other Uses of SQUARE

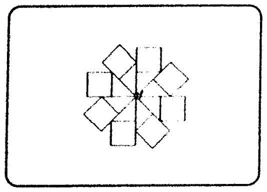
Once you have defined a procedure, you can use it as you would any Logo primitive, such as PRINT, FORWARD, CLEARSCREEN, and so on. This means that a procedure you define can be used as part of the definition of other procedures. This is one of the powerful features of Logo.


FLAG

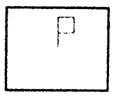

CROSS

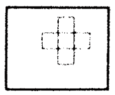

FLAGBACK

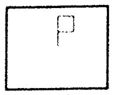

FLAGS

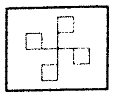

MANYFLAGS

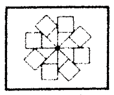
For example, there are many designs that use SQUARE. Here are some of them:


FLAG


CROSS


FLAGBACK


FLAGS


MANYFLAGS

```
TO FLAG
FD 30
SQUARE
END
```

```
TO CROSS
REPEAT 4 [FLAG RT 90]
END
```

```
TO FLAGBACK
FLAG
BK 30
END
```

```
TO FLAGS
REPEAT 4 [FLAGBACK RT 90]
END
```

Both FLAG and FLAGBACK make the turtle draw the same design, but they end the drawing with the turtle in different states. Both procedures leave the turtle with the same heading as they found it, but FLAG leaves the turtle in a different position from where it started.

FLAGBACK, on the other hand, leaves the turtle in the same position on the screen as it was found. You can see the result of these differences in CROSS and FLAGS. CROSS runs FLAG four times and FLAGS runs FLAGBACK four times, but the results are different.

Another procedure, MANYFLAGS, also uses FLAGS.

```
TO MANYFLAGS
FLAGS
RT 45
FLAGS
END
```

Full Graphics Screens: The screen pictures you see in the margin do not show any text. Your screen, however, may have text on it. You'll learn in Chapter 5 how to use the full screen for graphics without any text appearing.

Logo Vocabulary

This is a Logo primitive you learned in this chapter:

Full Name	**Short Name**
EDIT	ED

Special Keys

You learned to use a few special keys in this chapter:

[←]
[→]
[ó]-[A]
[↓]

Special Character

You also learned this special character:

" (quotation mark)	Tells Logo to interpret the next word literally and not run it as a procedure. Use with EDIT.

Saving and Printing

Saving and Printing

You can save your Logo procedures on a disk. Once they're saved on disk, you have them to use any time. Later, you can load them back into your computer.

Besides your procedures, you can also save turtle pictures or other work on a disk. Everything on a disk is kept in **files**. In this chapter, you'll learn a few Logo commands to handle disks and files. You'll also learn how to print on paper—if you have a compatible graphics printer attached to your Apple computer—an image of a turtle picture that is on the screen.

Formatting a Disk

You're going to save the procedures you defined so far. But before you can do this, you need to **format** a new blank disk. Formatting prepares the surface of a new disk so that Logo can put files on it. Follow these steps:

1. Make sure your Apple Logo II disk is in drive 1. If Logo is not started up already, do that now.

2. When the Logo ? (prompt) and _ (cursor) are displayed on the screen, type the command that transfers the FORMAT program from the Logo disk into the computer:

```
LOAD "FORMAT
```

Then check your typing, and after that press RETURN. In a moment, you see a special FORMAT screen.

3. Take out of the drive your Apple Logo II disk and replace it with a new blank disk.

Warning
If you don't do this, the FORMAT program may delete everything on the Apple Logo disk!

4. On the screen, you see the cursor _ after the words `Slot Number:`. Type 6 and press RETURN to tell the program that your disk drive is connected to slot or port 6. Then the cursor moves down to the right of `Drive:`. Type 1 and press RETURN again to tell the program that your blank disk is in drive 1.

About the Format Screen: Most formatting is done in drive 1 connected to slot or port 6. You can format in a different drive connected to a different slot or port. Just be sure you type the appropriate numbers for the drive and port or slot you are using.

Notice the messages at the bottom of the FORMAT screen. You can press -ESCAPE if you goofed and want to start over. By pressing -ESCAPE, you leave the FORMAT program and return to the Logo ? (prompt). Then to start over, you again load the FORMAT program. If you need help, press -?. That is, hold down both the key and the SHIFT key and at the same time type ?. The screen then displays a few simple formatting directions.

5. You need to pick a name, called a **volume name**, for your new program disk. Type

```
MYLOGO
```

and press RETURN. When the drive light goes out, you see the message

```
FORMAT COMPLETE
```

Bug Box
If you place an already formatted disk instead of a blank one in drive 1, the program asks you `IS IT ALL RIGHT TO DESTROY THE CONTENTS OF DISK?` and gives that disk's name. If it gives the name of the Apple Logo disk, you forgot to take out the disk with the Logo system on it. So type `N` for no, and immediately take the Apple Logo disk out of the drive. Put the blank disk in, close the drive door, and follow the directions on the screen.

6. With a pen, print the new disk name, exactly as you typed it, on a pregummed paper label. Take the newly formatted disk out of the drive, and stick the label on it.

7. If you are done formatting, press -ESCAPE to return to the Logo ? (prompt) and _ (cursor).

Anytime you wish to format another blank disk, you can return to the FORMAT program by loading it again.

Setting the Volume Prefix

Now tell Logo the volume name of the disk you will be using. Use the SETPREFIX command. This command saves you the trouble of typing the **prefix**—the disk or volume name—each time you give a disk command. Type

`SETPREFIX "/MYLOGO`   Press RETURN.

Saving Procedures on Disk

To save all the procedures you've typed into the computer so far, use the SAVE command and a filename. Be sure to use the " character exactly as shown. If you don't, Logo will not know you are referring to a filename. Type

```
SAVE "MYSQUARES
```

Press RETURN.

In a moment, Logo announces

```
MYSQUARES SAVED
```

Selective Saving: Sometimes you may want to save only one or a few, but not all, the procedures you typed into the computer. In that case, you can use the command SAVEL. See the *Reference Manual* for details.

To verify that the new file MYSQUARES was saved on your disk, type

```
CATALOG
```

Logo now lists the files that have been saved on MYLOGO.

Loading Procedures From Disk to Computer

Often you will want to transfer procedures stored in files on a disk back into the computer. To do so, you use the LOAD command. You already used it for the FORMAT program. The preliminary steps are the same: starting up the computer and the Logo disk, making sure the disk with the file you want to load is in a disk drive, and setting the prefix to the right volume name. Then you tell Logo to

```
LOAD "MYSQUARES
```

Or whatever filename is appropriate.

Other Useful Disk and File Commands: You can learn about other useful disk and file commands from the *Reference Manual*. Here are three very useful ones:

PREFIX	Tells you what volume name the prefix is set to.
ONLINE	Lists the volume names of all the disks in drives connected to your Apple.
ERASEFILE	Erases a file you name. Use with caution.

Printing Pictures

To print a picture, you must have an Apple Imagewriter printer attached to your Apple and turned on.

The picture you want to print must be on the screen before you try to print it. For example, to print the SQUARESTAR picture, first run the procedure to get the picture on the screen. Clear the screen with

```
CS
```

and type

```
SQUARESTAR
```

Now give the print picture command. Type

```
PRINTPIC 1
```

The 1 indicates which slot or port the printer is attached to.

Printer Slot or Port: The PRINTPIC command assumes the printer is connected to slot or port 1 of your Apple. If you know that the printer is connected to a different slot or port, direct Logo to that slot or port by typing the correct number in place of 1 when you give the command.

Bug Box
If the picture does not print, check all the power cord connections and also the cable connections between the printer and your Apple.

Using PRINTPIC: You can use the PRINTPIC command in two other ways:

- You can place the command inside a procedure. It must come after the instructions that draw the picture on the screen.
- You can draw with the turtle directly from the keyboard, and then print when you have a design you like on the screen.

If you're wondering how to save turtle pictures on a disk and load them from a disk, see the SAVEPIC and LOADPIC commands in the *Reference Manual.*

Logo Vocabulary

These are the Logo primitives you learned in this chapter:

SAVE
LOAD
SETPREFIX
CATALOG
PRINTPIC

Special Logo Program

You also learned to use the disk preparation program called

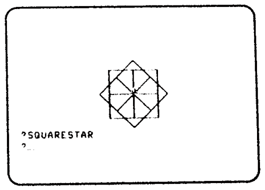
FORMAT

The Turtle and Text on the Screen

The Turtle and Text on the Screen

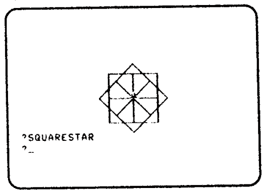
When you're using Logo, your monitor screen can be available solely for text, solely for pictures, or for a combination of the two. You'll learn how to control the screen with commands in this chapter.

The Text Screen and the Split Screen



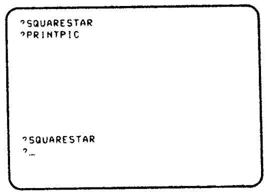
```
?SQUARESTAR
?_
```

Split Screen



```
?SQUARESTAR
?_
```

(CONTROL)-(S)



```
?SQUARESTAR
?PRINTPIC



?SQUARESTAR
?_
```

(CONTROL)-(T)

Before you give Logo any turtle commands, the whole screen is available for printing words, like commands and messages. This is called the **text screen**.

As soon as you give a turtle command, the screen is divided into a large turtle space and a small text space. In fact, only four lines at the bottom are available for text. You saw this **split screen** in Chapter 3 when you told Logo to run the procedure SQUARE. When the turtle finished drawing the square, you saw the ? (prompt) and _ (cursor) in the bottom text space of the screen.

After you have been using the split screen, you can easily get back to the whole screen for text. The command TEXTSCREEN gets you the whole screen for text. SPLITSCREEN gets you back to the turtle space and the four-line text space. Neither of these commands destroys what was in the two spaces. They only change what is visible to you.

There are special action keys that have the same effect as these screen commands. (CONTROL)-(T) shows the whole text screen and (CONTROL)-(S) shows the split screen.

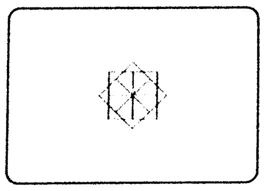
Try switching. First press CONTROL-S. Then run the procedure SQUARESTAR by typing

```
SQUARESTAR
```

Getting SQUARESTAR: If SQUARESTAR is no longer in your workspace, load the file MYSQUARES from your program disk. Then run it.

Now go back and forth between the text screen and the split screen. Press CONTROL-T and then CONTROL-S and then CONTROL-T, and so on.

The Full Graphics Screen


CONTROL-L

FULLSCREEN (or CONTROL-L) gives the whole screen to the turtle. No text is visible. So if you type FULLSCREEN, you don't see the text characters typed on the screen, but they are there all the same.

Try this: CONTROL-L and then CONTROL-S and then CONTROL-L.

You can press CONTROL-T, CONTROL-S, and CONTROL-L while a procedure is running.

The Editor Screen

When you edit a procedure using EDIT, the turtle screen and whatever lines are on it become invisible. The editor screen takes its place. When in the editor, you can press CONTROL-L to see the original turtle screen. To return to the editor screen, press CONTROL-T.

Logo Vocabulary

These are the Logo primitives you learned in this chapter:

TEXTSCREEN
FULLSCREEN
SPLITSCREEN

Special Keys

You learned to use a few special keys in this chapter:

CONTROL-T
CONTROL-S
CONTROL-L

The Turtle's Pen and Color

The Turtle's Pen and Color

The turtle leaves a track whenever you tell it to move FORWARD or BACK a certain number of steps. This is because the turtle has a "pen" to draw with. If you want the turtle to move without leaving a track, you can tell the turtle to lift its pen. You can change the color of the track by changing the color of the pen. And you can tell the turtle to fill in a space between tracks with solid color. This chapter explains how to use the pen and how to make color graphics.

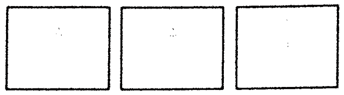
Pen Commands

These are commands that directly affect the action of the turtle pen.

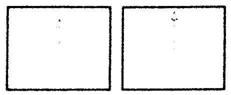
PENUP and PENDOWN

To lift the pen up, use the command PENUP (or PU). To put the pen down again, use the command PENDOWN (or PD). Try experimenting to see the results produced by these two commands. Draw a line using FD, then use PU, command FD again, use PD, and command FD once more.

```
FD 30
PU                      Or PENUP
FD 20
```



```
PD                      Or PENDOWN
FD 20
```



PENERASE

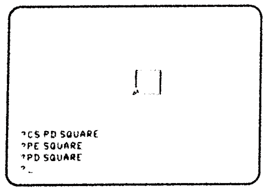
You can erase what the turtle has drawn by using the command PENERASE.

If you give the command PENERASE (or PE), the turtle becomes an eraser instead of a drawing tool. Then if you make it retrace a line it has drawn, the line is erased.

For example, clear the screen, make sure the pen is down, and draw a square. Type

```
CS PD SQUARE
```

More Than One Command Per Line: Here's a new idea: you can type more than one command on a line. This is true of commands inside a procedure as well as of commands you type directly to Logo from the keyboard.



```
?CS PD SQUARE
?PE SQUARE
?PD SQUARE
?_
```

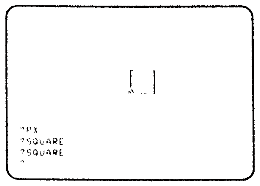
SQUARE

Now type

```
PE SQUARE               PE for PENERASE.
```

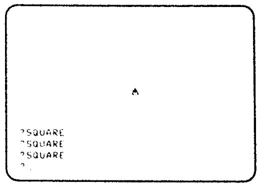
The turtle erases any lines on the screen until you tell it to PENUP or to PENDOWN. Notice the turtle does not draw any new lines.

PENREVERSE



```
?PX
?SQUARE
?SQUARE
?
```

Second SQUARE



```
?SQUARE
?SQUARE
?SQUARE
?
```

Third SQUARE

PENREVERSE (or PX) is a mixture of PENDOWN and PENERASE. When you give this command, the turtle draws a line whenever it moves over blank background. But when it moves along a previously drawn line, it erases it. You can use this to produce some spectacular effects. For example, draw a square by typing

```
PX SQUARE
```

and then typing

```
PX                      Or PENREVERSE
SQUARE
SQUARE
SQUARE
```

Now you must return the pen to its normal drawing state. So type

```
PD
```

Using Apple Color Graphics

This section describes Logo commands and features that take advantage of Apple color. Your Apple computer allows you to use six colors: black, white, green, violet, orange, and blue. If you have a black-and-white video monitor or television set, your color selection includes gray tones as well as black and white, but no other colors.

There are two types of color changes you can make. You can change the color of the background—the space that surrounds the turtle design—and you can change the color of the turtle's pen.

The command SETBG, with a number input, changes the background color. The command SETPC, with a number input, changes the pen color. The pen color must be different from the background color if you are to see the pen lines.

Both SETBG and SETPC take one input, which must be a number. Here are the number codes you use as input when you type a command to change either the background or the pen color.

To Get This Color	Type
black	0
white	1
green	2
violet	3
orange	4
blue	5
black (black-and-white TV)	6

Note that background colors 0 and 6 are both black; 6 is the recommended background for a black-and-white screen, since the pen draws thinner lines with a 6 background.

Changing the Background Color

Try

```
SETBG 1
SETBG 2
SETBG 3
SETBG 4
SETBG 5
```

Now define a procedure, CB, that cycles through background colors.

To see the colors more clearly use the Logo command WAIT. WAIT 60 makes Logo wait for 1 second before running the next command. Try different inputs to WAIT.

```
TO CB
SETBG 1 WAIT 20
SETBG 2 WAIT 20
SETBG 3 WAIT 20
SETBG 4 WAIT 20
SETBG 5 WAIT 20
SETBG 0 WAIT 20
END
```

Or define CB in editor.

Repeat CB a few times. Type

```
REPEAT 3 [CB]
```

You can always find out the number code of the current background color by typing BACKGROUND or its short name, BG. Type

```
PR BG
```

Logo responds

```
0
```

to tell you the background color is set to black.

Logo Operations: BACKGROUND is part of Logo's vocabulary, but it is different from SETBG or SETPC. It is not a command; it is an **operation**. It does not cause something to happen, but rather produces an **output** that can be used as an input by a command or another operation. (An output is the value an operation produces as a result of its action or computation.) Several other operations are introduced in this manual.

Changing the Pen Color

Use the command SETPC. Try different pen colors. Start with black as the background color and a clear screen.

```
SETBG 0 CS
SETPC 2 SQUARE
RT 90 SETPC 3 SQUARE
RT 90 SETPC 4 SQUARE
RT 90 SETPC 5 SQUARE
```

Now type

```
SETPC 0   SQUARE
```

The square disappears! This should not surprise you because the pen color and the background color are the same. To set the pen color to white, type

```
SETPC 1
```

For Thinner Black-and-White Lines: If you are using a black-and-white video monitor or television set, one way to get thinner pen lines is the following:

```
SETBG 6 SETPC 1
```

PENCOLOR or PC gives you the current number code for the pen color. Type

```
PR PC
```

Logo responds

```
1
```

to tell you the pen color is white.

Bug Box
Pen color green does not draw on background orange, and pen color orange does not draw on background green. Violet and blue do not show up on one another.

If you change the color of the background, lines already on the screen might change color. You can learn to use color well by trying different combinations.

Filling in Shapes or Background

Logo has a graphics command that allows you to fill in with color any space on the screen. The space can be one enclosed with turtle pen lines or it can be a background space. Either type of space is filled with whatever color the pen is currently set to.

Do you have a square on the screen? If not, clear the screen with

```
CS
```

and type

```
SQUARE
```

Now type

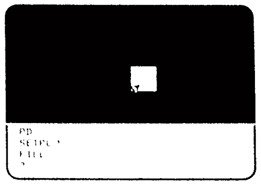
```
PU
RT 45 FD 2
PD
```

By moving the turtle inside the square, you've set up the turtle to fill in this space with solid color. In this case, the turtle is a kind of pointer. Next, set the pen color to white.

```
SETPC 1
```

Finally, you need to give the FILL command. Type

```
FILL
```



```
PD
SETPC 1
FILL
?
```

Filling SQUARE

You should now be looking at a white square. You can hide the turtle to get a clear picture. You can set the pen color to a different color and try FILL again.

Now clear the screen and run the procedure SQUARESTAR. Type

```
CS SQUARESTAR
```

Look at the design on the screen. It has many closed spaces. A closed space is an area on the screen that has pen lines completely enclosing it, with no break or opening to the background. SQUARE had one such space. SQUARESTAR has many.

You have to fill in one space at a time. Do the same as before. First move the turtle.

```
PU RT 20
FD 15 PD
```

Acting as a pointer, the turtle is now inside one of the closed spaces.

Now again set the color to white.

```
SETPC 1
```

Then color that space by giving the command

```
FILL
```

Move the turtle to another space in the SQUARESTAR design. Be sure to lift the pen first and to put it back down before filling.

If you have a color monitor or TV, you can use any color and can even fill in adjacent spaces, keeping in mind when mixing colors the limitations pointed out earlier.

Now fill in the background.

```
SETBG 4
PD FILL
```

If you have a black-and-white monitor or TV, you should now be looking at the SQUARESTAR design against a gray background. (Remember, color values other than black or white only produce tones of gray for you.) If you have a color monitor or TV, you should be looking at the SQUARESTAR design against an orange background.

You can also use FILL in procedures that draw turtle pictures. Use the editor to define or modify the procedure that will do the filling process. Remember that the filling commands you give should come after those that draw the picture. Your commands must tell Logo to do the following:

1. Lift the pen and move the turtle inside the background area or closed space you want to fill.
2. Put the pen down again.
3. Set the pen color.
4. Fill the area occupied by the turtle.

You repeat these commands if any other space is to be filled.

Try writing a procedure that uses FILL.

Logo Vocabulary

These are the Logo primitives you learned in this chapter (* denotes operations):

Full Name	**Short Name**
PENDOWN	PD
PENUP	PU
PENERASE	PE
PENREVERSE	PX
SETBG	
BACKGROUND*	BG
SETPC	
PENCOLOR*	PC
WAIT	
FILL	

Another Look at Editing

Another Look at Editing

chapter 7

The Logo editor allows you to change already defined procedures as well as define new ones. You may wish to change one of your procedures to fix a bug or to alter what the procedure does.

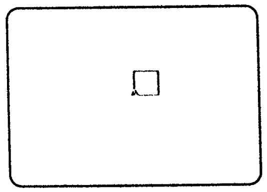
Changing a Procedure

For practice with the editor, first define a procedure to draw a diamond, but with a bug in it. Type

```
TO DIAMOND
SQUARE
RT 45
END
```

Now type

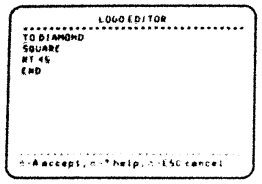
```
DIAMOND
```


This Is a Diamond?

This procedure is supposed to draw a diamond. You tried it and found it draws a square, not a diamond! The bug is obvious. The command RT 45 should be used *before* the turtle draws a square. To fix the bug, you need to edit the procedure. Type

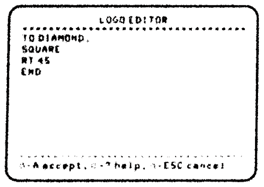
```
EDIT "DIAMOND
```

The text of the procedure DIAMOND is now displayed on the screen, with the _ (cursor) under the first T in the title line.



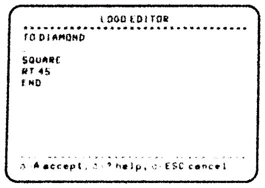
```
LOGO EDITOR
TO DIAMOND
SQUARE
RT 45
END
ó-A accept, ó-? help, ó-ESC cancel
```

DIAMOND in Editor



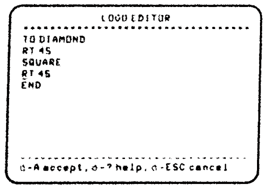
```
LOGO EDITOR
TO DIAMOND_
SQUARE
RT 45
END
ó-A accept, ó-? help, ó-ESC cancel
```

You Move Cursor



```
LOGO EDITOR
TO DIAMOND
_
SQUARE
RT 45
END
ó-A accept, ó-? help, ó-ESC cancel
```

You Press (RETURN)



```
LOGO EDITOR
TO DIAMOND
RT 45
SQUARE
RT 45
END
ó-A accept, ó-? help, ó-ESC cancel
```

You Move Cursor

```
TO DIAMOND
SQUARE
RT 45
END
```

Editing Reminder: To start editing, you first move the cursor to where you want to add or delete characters. You move the cursor to the right by pressing (→). You move it to the left by pressing (←).

To edit DIAMOND, move the cursor to the end of the title line.

```
TO DIAMOND_
```

Now press (RETURN) and type

```
RT 45
```

Press (RETURN).

Move the cursor down to the line before END by pressing (↓) two times. Now the cursor should be under the R in

```
RT 45
```

Press (CONTROL)-(F) five times to erase the characters on the line. Don't worry about the empty line. Logo will take it out when you leave the editor.

This is how the edited and redefined DIAMOND looks.

```
TO DIAMOND
RT 45
SQUARE
END
```

Useful Editing Actions: Here are some useful editing actions. The *Reference Manual* describes more of them. Those denoted by * are not discussed in this chapter, but are useful and will be mentioned later.

To Do This	**Use This Keypress**
Move the cursor down to the next line.	(↓)
Move the cursor up to the previous line.	*(↑)
Move the cursor and the following text to the beginning of the next line.	(RETURN)
Move the cursor to the beginning of the current line.	*(ó)-(<)
Move the cursor to the end of the current line.	*(ó)-(>)
Move the cursor one space left.	(←)
Move the cursor one space right.	(→)
Erase the character directly under the cursor as the cursor moves to the next space to the right.	*(CONTROL)-(F)
Erase the character to the left of the cursor.	(DELETE)
Erase all the characters on the line where the cursor is.	*(CONTROL)-(X)
Erase all the characters to the right of the cursor.	*(CONTROL)-(Y)
Print a copy of the line that was most recently deleted using (CONTROL)-(X) or (CONTROL)-(Y).	*(CONTROL)-(R)

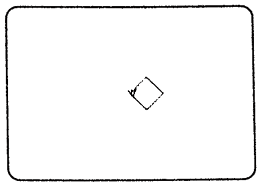
Leaving the Editor

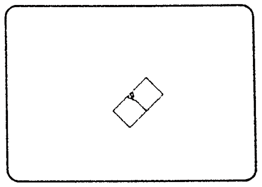
Press (ó)-(A). This keypress asks Logo to accept the changes you've made. Logo prints a message saying

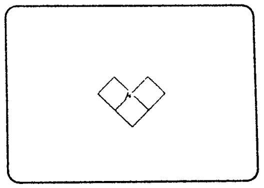
```
DIAMOND DEFINED
```

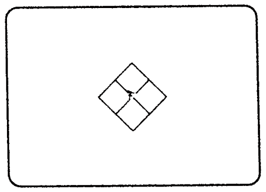
Bug Box
If you are editing and make typing mistakes and want to start again, press (ó)-(ESCAPE). Logo halts the editing and forgets all the changes you made so far.

Editing Outside of the Editor


DIAMOND


RT 45 DIAMOND


(Again) RT 45 DIAMOND


(Yet Again) RT 45 DIAMOND

You can use most of the editing keystrokes listed earlier to edit instructions you type to Logo when you are not in the editor. But one of these, (CONTROL)-(R), has a special function that works only outside the editor. Type

```
DIAMOND                    Press (RETURN).
```

The turtle draws a diamond. Now press (CONTROL)-(R). Logo responds

```
DIAMOND_
```

This command displays for editing purposes a copy of the last line you typed. Each such line of text on the screen is in a mini-editor, only one line long. The _ (cursor) is at the end of the line, ready in case you want to make any changes in the line.

Try this change. Press (ó)-(<) to move the cursor to the beginning of the line.

```
DIAMOND
```

Now type

```
RT 45
```

Press the (SPACE) bar and (RETURN). By this editing action, you move RT 45 in front of the DIAMOND command, adding a space to separate the two commands. By pressing (RETURN), you instruct the turtle to draw the design in that order, RT 45 first and then DIAMOND.

Press (CONTROL)-(R) again and press (RETURN). Logo responds by drawing another diamond. Do it once more and watch a larger diamond emerge on the screen.

(CONTROL)-(R) (RETURN)

Experiment with other editing actions both in the editor and outside of it. For more details consult the *Reference Manual*.

Remember to Save It: This is a good time to save your new procedure DIAMOND. If you don't save it now, you might forget to do it before turning off your Apple. If you've forgotten the steps for saving procedures on your program disk, go back to Chapter 4. Remember, if the prefix is currently set to another volume name, you first have to change the prefix with the SETPREFIX command.

Special Keys

You learned to use a few special keys in this chapter:

CONTROL-F
Ó-<
CONTROL-R

Your Workspace

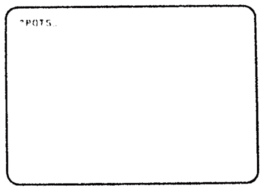
chapter 8

Your Workspace

As you define procedures, you create new Logo primitives—like SQUARE and SQUARESTAR. Logo puts these new primitives in a part of your Logo workspace, the temporary storage place in your Apple's memory.

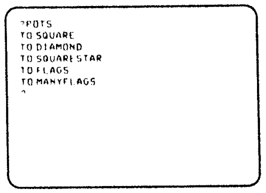
If you want to see what you have in your workspace, Logo provides several ways to do so. For example, you can print out the title lines of all the procedures you have written or you can print out their definitions.

Printing Out Procedures



```
?POTS
```

You Type



```
?POTS
TO SQUARE
TO DIAMOND
TO SQUARESTAR
TO FLAGS
TO MANYFLAGS
?
```

You See

The command POTS (for print out titles) prints out the title lines of all of the procedures in the workspace. Type

```
POTS
```

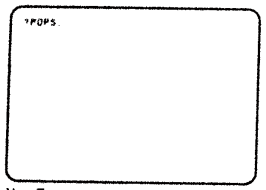
Logo responds

```
TO SQUARE
TO DIAMOND
TO SQUARESTAR
.
.
.
```

and so on.

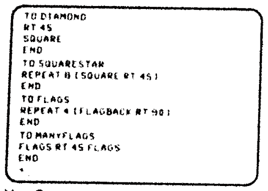
Bug Box
If you don't see the same lists displayed on your screen as printed here in this chapter, this may be because you

- **erased one or more procedures from the workspace**
- **didn't type definitions for all the procedures in this list.**



```
?POPS
```

You Type



```
TO DIAMOND
RT 45
SQUARE
END
TO SQUARESTAR
REPEAT 8 [SQUARE RT 45]
END
TO FLAGS
REPEAT 4 [FLAGBACK RT 90]
END
TO MANYFLAGS
FLAGS RT 45 FLAGS
END
?
```

You See

POPS (for print out procedures) prints out the definitions of all the procedures in the workspace. Type

```
POPS
```

Logo responds

```
TO SQUARE
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END

TO DIAMOND
RT 45
SQUARE
END

TO SQUARESTAR
REPEAT 8 [SQUARE RT 45]
END
.
.
.
```

and so on.

Stopping the Scrolling: If you have more procedures in the workspace than fit on one screen, the text scrolls upward or disappears so fast you don't have time to read it. In this case, press (CONTROL)-(W) whenever you want to stop it to take a close look. (The W in (CONTROL)-(W) means wait.) You start the scrolling again by pressing any key.

You can print out the definition of any particular procedure with PO (for print out). For example, type

```
PO "SQUARESTAR
```

Logo responds

```
TO SQUARESTAR
REPEAT 8 [SQUARE RT 45]
END
```

PO can also take a list of procedure names. For example, PO [SQUARE SQUARESTAR DIAMOND] prints out the definitions of the three procedures whose names are in brackets. Remember that (CONTROL)-(T) or TEXTSCREEN lets you use the entire screen for text.

Erasing From the Workspace

You can erase procedures from your workspace. If you do not save these procedures on a disk, you will have to type them in again. So make sure you really want to erase before you do so.

There are several Logo commands to erase your work. The most commonly used is ERASE or ER. You use it to erase one or a select few procedure definitions from the workspace. Try it, if you wish. You can always retype the definitions you erased.

```
ERASE "DIAMOND
```

erases the procedure DIAMOND.

```
ERASE [SQUARE SQUARESTAR]
```

erases the procedures named in brackets. Usually you use ERASE to

- conserve workspace if you are filling it with many procedures
- delete all but one or a few procedures you intend to save on a disk.

To check to see that your procedures were erased, type POTS.

The command ERPS (for erase procedures) erases all the procedures in the workspace. You use ERPS only if the workspace is so full that you need to empty it before you can type or load other procedures.

Logo Vocabulary

These are the Logo primitives you learned in this chapter:

Full Name	**Short Name**
ERASE	ER
ERPS	
POTS	
POPS	

Special Key

You learned to use this special key in this chapter:

CONTROL-W

First Program: Drawing a Spider

First Program: Drawing a Spider

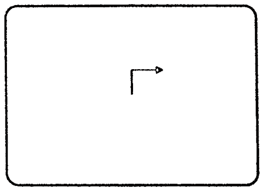
You know enough Logo to write your first **program**! You will define several procedures designed to work together as a computer program. The result will be a turtle graphics picture of a spider.

Your spider will have four legs on each side. Each leg is made by two lines joined to form a 90-degree angle.

The First Step

As a first step, make a RIGHTLEG.

```
TO RIGHTLEG
FD 30                    Any number will do.
RT 90
FD 30
END
```


RIGHTLEG With Bug

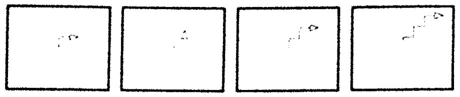
Now test this new procedure. Type

```
RIGHTLEG
```

Although this procedure makes a leg, the turtle stops at a funny place for attaching another spider leg to this one.

Actually, at this point RIGHTLEG could be used to make stairs. Try it.

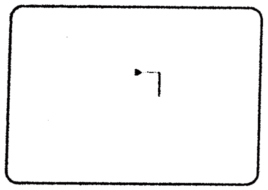
```
RIGHTLEG
LT 90
RIGHTLEG
LT 90
RIGHTLEG
```



But you want spider legs.

When in doubt about where the turtle should be when the procedure stops, put the turtle where it was before the procedure was run. Do that by fixing or debugging RIGHTLEG, using the editor.

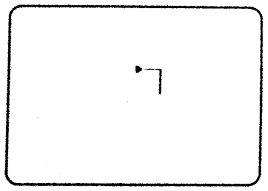
```
EDIT "RIGHTLEG
```


RIGHTLEG Debugged

The Logo editor shows this procedure with the _ (cursor) under the T of TO.

```
TO RIGHTLEG
FD 30
RT 90
FD 30
END
```

Move the cursor down to be under the E in END. Now you can type in the new commands.


RIGHTLEG

```
TO RIGHTLEG
FD 30
RT 90
FD 30
BK 30
LT 90
BK 30
END
```

These three commands put the turtle where it was at the start of RIGHTLEG.

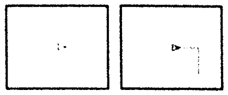
Try RIGHTLEG.

```
RIGHTLEG
```

Another Procedure or Two

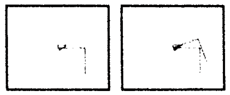
Now, before you define it, plan RIGHTSIDE, the procedure that will draw all the legs on the spider's right side. You need one leg horizontally positioned. Type

```
CS
RT 90
RIGHTLEG
```



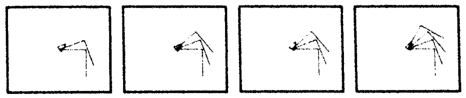
Do the second leg.

```
LT 20
RIGHTLEG
```



Good. Continue in this way until the turtle has drawn four legs. Now make a procedure for RIGHTSIDE:

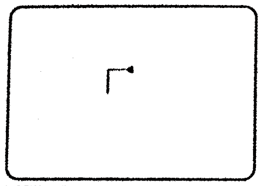
```
TO RIGHTSIDE
RT 90
REPEAT 4 [RIGHTLEG LT 20]
LT 10
END
```



Notice the last command, LT 10, returns the turtle to the same position and heading it was in at the start of the procedure. It is good practice to adopt this rule: "Leave the turtle in the state in which you found it."

Try

```
RIGHTSIDE
```

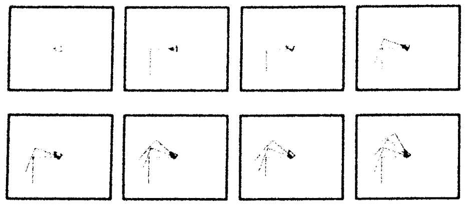

LEFTLEG

Now work on a left leg. LEFTLEG will be similar to RIGHTLEG.

```
TO LEFTLEG
FD 30
LT 90
FD 30
BK 30
RT 90
BK 30
END
```

Try it. Then use LEFTLEG in the definition of LEFTSIDE.

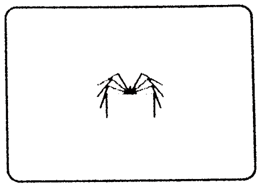
```
TO LEFTSIDE
LEFT 90
REPEAT 4 [LEFTLEG RT 20]
RT 10
END
```



The Superprocedure

Finally, you can write the superprocedure SPIDER to put them together. (A superprocedure is the principal procedure of a program. It mainly calls, in the proper order, the procedures that do the actual drawing of the picture.)

```
TO SPIDER
LEFTSIDE
RIGHTSIDE
FD 10 BK 10
HT
END
```

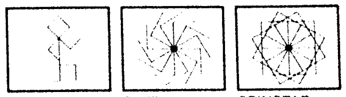

SPIDER

Now run your first full computer program by typing

```
SPIDER
```

Recognize this little fellow?

Here are other designs you can make using RIGHTLEG and LEFTLEG:


MAN SWIRL SPINSTAR

Save Your New Program: This is a good time to save your new program SPIDER. If you have procedures in the workspace other than those used to make SPIDER, either erase those first or use the command SAVEL that allows you to save part of the workspace contents. Remember, if the prefix is currently set to another volume name, you'll first have to change the prefix with the SETPREFIX command.

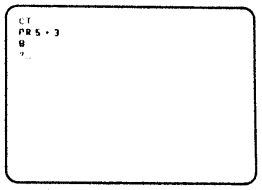
Logo Does Arithmetic

chapter 10

Logo Does Arithmetic

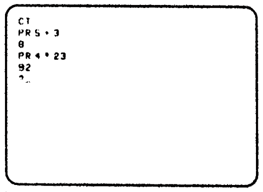
In chapters to follow, you'll occasionally tell Logo to do some arithmetic for you. This is a good place to find out how.

Doing the Four Basic Operations



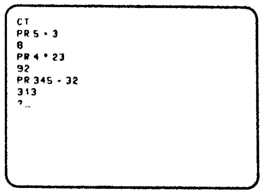
```
CT
PR 5 + 3
8
?_
```

Logo Adds



```
CT
PR 5 + 3
8
PR 4 * 23
92
?_
```

Logo Multiplies



```
CT
PR 5 + 3
8
PR 4 * 23
92
PR 345 - 32
313
?_
```

Logo Subtracts

For example, Logo can add numbers. To see this happen, first clear the text screen.

```
CT
```

Then type

```
PR 5 + 3
```

Logo prints

```
8
```

Logo can multiply. Type

```
PR 4 * 23
```

Logo prints

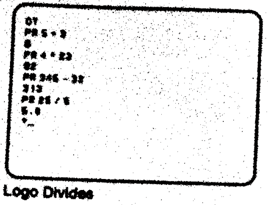
```
92
```

Logo can subtract. Type

```
PR 345 - 32
```

Logo prints

```
313
```



```
CT
PR 5 + 3
8
PR 4 * 23
92
PR 345 - 32
313
PR 25 / 5
5.0
?_
```

Logo Divides

And Logo can divide. Type

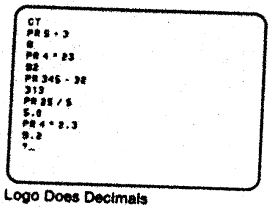
```
PR 25 / 5
```

Logo prints

```
5.0
```

Decimal and Integer Numbers

Notice that the result of 25 divided by 5 is a decimal number. Logo has both decimal and integer numbers.



```
CT
PR 5 + 3
8
PR 4 * 23
92
PR 345 - 32
313
PR 25 / 5
5.0
PR 4 * 2.3
9.2
?_
```

Logo Does Decimals

Some computations always result in a decimal answer. Division (/) is one of those. Other arithmetic operations depend on what they are given as inputs. For example, if you type

```
PR 4 * 2.3
```

Logo prints

```
9.2
```

For more discussion about arithmetic in Logo, consult the *Reference Manual*.

Special Characters

You learned these arithmetic characters:

/	Means divide. Tells Logo to divide the first number by the second number.
-	Means subtract. Tells Logo to subtract the second number from the first number.
+	Means add. Tells Logo to add the two numbers.
*	Means multiply. Tells Logo to multiply the two numbers.

Some Geometry: Triangles

Some Geometry: Triangles

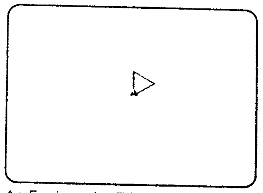
chapter 11

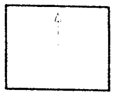
The turtle can draw different triangle shapes. The triangle you learn about below, called an **equilateral** or **equiangular** triangle, is like a square in that all its sides are equal and all its angles are equal.

A Procedure for a Triangle

In this sample triangle, the turtle will first take 30 steps forward, the same amount it took in SQUARE. Type

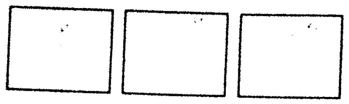
```
FD 30
```


An Equiangular Triangle



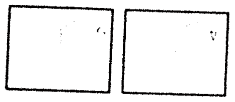
Now comes the big decision. How many degrees does the turtle have to turn to draw this triangle? Sometimes people think that equiangular triangles have 60-degree angles. Look what happens when the turtle turns 60 degrees. Try it.

```
RT 60
FD 30
RT 60
```

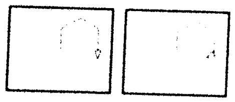


Interesting, but this won't make a triangle! But you might as well finish it.

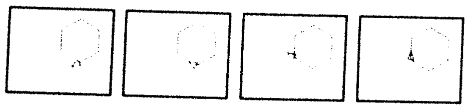
```
FD 30
RT 60
```



```
FD 30
RT 60
```



```
FD 30
RT 60
FD 30
RT 60
```

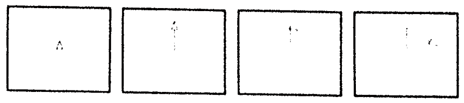


The figure is a hexagon; it has six rather than three sides. To make a triangle, the turtle needs to turn 120 degrees at each corner.

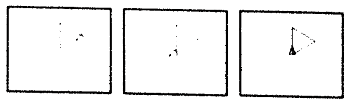
Why 120 Degrees? Why 120 and not 60? The answer is simple. The internal angle of each corner *is* 60 degrees measured from inside the triangle, side to side—but the turtle has to make a much larger turn than that. To complete its triangle trip, the turtle must turn 360 degrees (a complete circle) before it returns to its starting state. But it only turns three times altogether, since a triangle has only three angles. That means each turn has to be larger than 60 degrees if three turns has to add up to 360 degrees. Now 360 divided by 3 is 120—that's the amount the turtle has to turn each time.

Now you can finish drawing the triangle.

```
CS
FD 30
RT 120
FD 30
```



```
RT 120
FD 30
RT 120
```

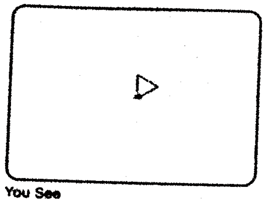


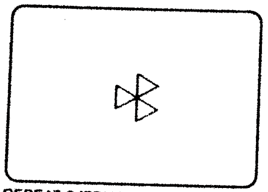
Now you can define the procedure TRIANGLE. Use the editor to do this.

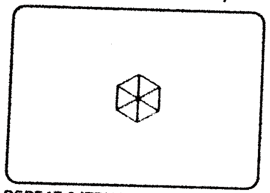
```
EDIT "TRIANGLE
```

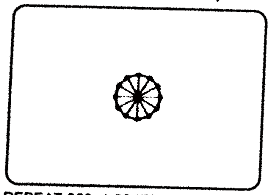
Below the TRIANGLE title line, type

```
REPEAT 3 [FD 30 RT 120]
END
```


You See


REPEAT 3 [TRIANGLE RT 120]


REPEAT 6 [TRIANGLE RT 60]


REPEAT 360 / 30 [TRIANGLE RT 30]

Play with this procedure. For example,

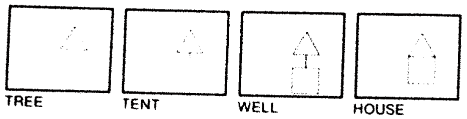
```
REPEAT 3 [TRIANGLE RT 120]
REPEAT 6 [TRIANGLE RT 60]
REPEAT 100 [TRIANGLE RT 30]
```

In this last example, the turtle retraces its path many times. You can always stop the turtle by pressing (ó)-(ESCAPE).

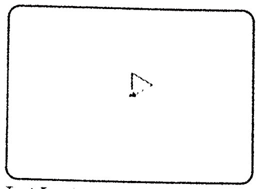
You might want to figure out how many times the turtle needs to repeat a set of commands in order to make a 360-degree trip and complete a shape. For example, if the turtle turns 30 degrees each round, it has to repeat the set of commands 360 divided by 30 or 12 times. Logo can do arithmetic and so can divide 360 by 30 for you.

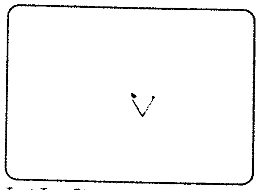
```
REPEAT 360 / 30 [TRIANGLE RT 30]
```

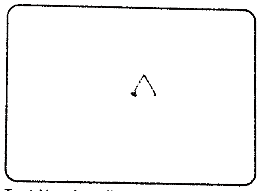
Here are some designs you can make by using TRIANGLE:



TENT From TRIANGLE


Tent Tips Left


Tent Tips Right


Tent Now Level!

Make the turtle draw TENT. Just running TRIANGLE is not enough; the TENT will be tipped. Try it anyhow.

```
CS
TRIANGLE
```

Try turning the turtle RT 90 and then running TRIANGLE. Maybe this will work.

```
CS
RT 90
TRIANGLE
```

This time the tent is upside down. Here's a better idea. Remember that when you made TRIANGLE, the inside angle was 60 degrees. Try turning half that angle or 30 degrees.

```
CS
RT 30
TRIANGLE
```

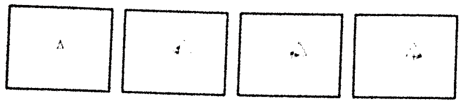
The procedure is

```
TO TENT
RT 30
TRIANGLE
END
```

TREE1 From TENT

Now you can use TENT to help make TREE1.

```
CS
TENT
RT 60
FD 15
```



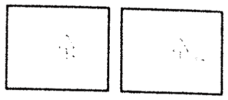
```
RT 90
FD 15
RT 180
```

Now define TREE1.

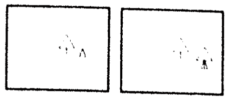
```
TO TREE1
TENT
RT 60
FD 15
RT 90
FD 15
RT 180
END
```

You can make three or four trees appear on the screen. For example,

```
RT 90
PU
FD 30
```



```
LT 90
PD
TREE1
```

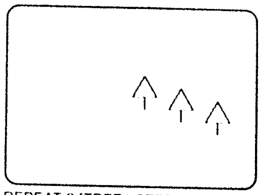


The commands that came before you typed TREE1 are called **setup instructions**. It is a good programming habit to put setup instructions in a separate procedure. SETTREE will set up the turtle for drawing a new tree on the screen.

```
TO SETTREE
RT 90
PU FD 30
LT 90 PD
END
```

You can then use TREE1 and SETTREE repeatedly.

```
REPEAT 3 [TREE1 SETTREE]
```

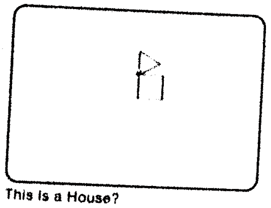

REPEAT 3 [TREE1 SETTREE]

If you want to change the distance between trees, then edit SETTREE and change the amount the turtle goes FORWARD.

HOUSE From SQUARE and TENT

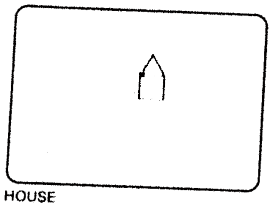
You have taught the turtle to make a square and a triangle. Now put them together to make a house.

```
SQUARE
FD 30
TRIANGLE
```


This is a House?

You put them together, but the turtle didn't draw a house! There is a bug. The fix is simple. Use TENT instead of TRIANGLE.

```
SQUARE
FD 30
TENT
```


HOUSE

Now make HOUSE a procedure.

```
TO HOUSE
SQUARE
FD 30
TENT
END
```

You might like to try making WELL on your own. Here's a tip. Look at the picture a few pages back.

A Reminder About Saving: This is a good time to save your new procedure TRIANGLE and the other procedures you defined during this chapter. You might like to save them under the new filename TRIANGLES1. (Later you'll be making some different triangle designs.) Be sure to type `POTS` first, to see what you have in the workspace.

Variables: Changing Sizes of Shapes

Variables: Changing Sizes of Shapes

chapter 12

In this chapter, you will learn to make procedures that allow you to change the size of a shape each time you run it. All you do is type a number for the size you want.

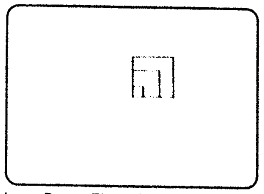
Big Squares and Small Squares

You might want the turtle to draw squares with sides of 60, 50, 100, 10, and so on. One way of doing this is to have many procedures—SQUARE100, SQUARE50, SQUARE33, and others like this—for each new size of square.

But there is a shortcut. You can change SQUARE (the procedure you defined in Chapter 3) so that it is like FORWARD in that it takes an input. Then you can tell SQUARE how long to make its sides by typing an input each time you run SQUARE. You would type

```
SQUARE 50
SQUARE 33
SQUARE 13
```

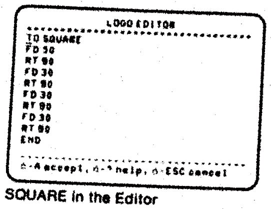
and so on.


Logo Draws Three Squares

So, change SQUARE to make a procedure for drawing different-sized squares. BOX might be a suitable name because it reminds us of squares. But if you call it BOXR (pronounced box-are), you'll meet a new idea—the idea of twin procedures. BOXR can be a square where the turtle makes all right turns. You can make a corresponding square where the turtle makes all left turns, BOXL (pronounced box-ell).

A shortcut method for typing in the definition of BOXR is to modify SQUARE in the editor. If you change the name of the procedure before you leave the editor, you will not change the definition of SQUARE.

```
EDIT "SQUARE
```

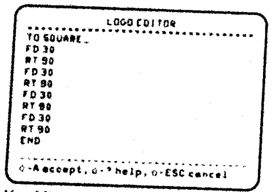


```
LOGO EDITOR
..........................................
TO SQUARE
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
------------------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

SQUARE in the Editor

Now the Logo editor shows the procedure with the _ (cursor) under the T of TO.

```
TO SQUARE
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
```



```
LOGO EDITOR
..........................................
TO SQUARE_
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
------------------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

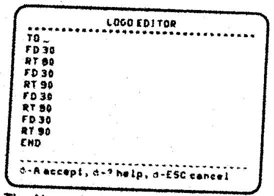
You Move the Cursor

First, change the name of the procedure from SQUARE to BOXR. Using (→), move the cursor right, to the space after SQUARE.

```
TO SQUARE_
```

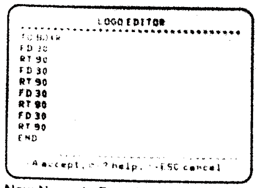
Now erase the word SQUARE, using (DELETE). Next, type BOXR.

```
TO BOXR_
```



```
LOGO EDITOR
..........................................
TO _
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
------------------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

The Name Is Erased



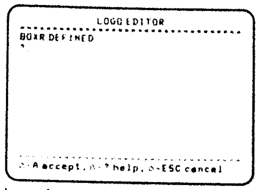
```
LOGO EDITOR
..........................................
TO BOXR
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
------------------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

New Name Is Typed

Press (ó)-(A) and Logo accepts your definition by responding

```
BOXR DEFINED
```

Check the Workspace: You might like to type POTS now, to see if the workspace contains BOXR as well as the original SQUARE procedure.



```
LOGO EDITOR
..........................................
BOXR DEFINED
?
------------------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

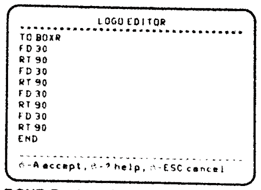
Logo Accepts BOXR

At this point, BOXR and SQUARE have the same definition. You need to change the definition of BOXR. Type

```
EDIT "BOXR
```

The Logo editor now shows

```
TO BOXR
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
```



```
LOGO EDITOR
..........................................
TO BOXR
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
------------------------------------------
ó-A accept, ó-? help, ó-ESC cancel
```

BOXR Back in the Editor

You first need to change BOXR so that it requires an input like FORWARD does. Then the procedure will be able to draw squares of various sizes. How do you tell Logo to do this?

The first instruction has to be "FORWARD some amount." But when you are only at the stage of defining the procedure, you don't know what the amount will be. We handle this situation by giving the amount a name.

Call it :SIDE, because that's the part of the square you want to change each time. The : (colon) tells Logo that SIDE is a **variable**, a word that can contain different values.

Now replace each 30 with :SIDE.

```
TO BOXR :SIDE
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
END
```

Space only before colon.

One more thing is needed to make this new procedure work. To tell Logo that :SIDE is BOXR's input, the title line must look like this:

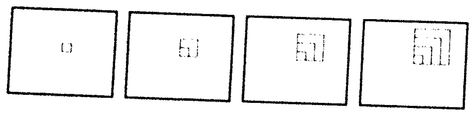
```
TO BOXR :SIDE
```

You type a number after the procedure name BOXR whenever you run the procedure. Wherever the same variable name (:SIDE) appears in the definition, Logo knows that it must supply the number you gave as input.

Now press ()-(A) and Logo accepts the newly defined procedure.

BOXR makes the turtle draw a square of any size, depending upon the number you give it as an input. Run each of these, one at a time.

```
BOXR 10

BOXR 20

BOXR 30

BOXR 40
```



How Variables Work: You have just used a powerful mathematical idea—the idea of a variable. A variable is a container for different values. Instead of using a mysterious x for the variable, as you do in school algebra, you have used a meaningful name, :SIDE.

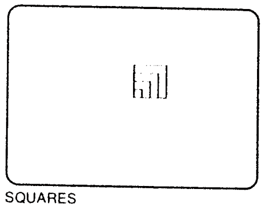
The expression :SIDE means "whatever happens to be in the container called :SIDE." If Logo is to carry out the command FORWARD :SIDE, there must be something in the container.

The container is filled when you run BOXR and type an input such as 10 or 15. When Logo obeys that command, 10, 15, or whatever you type as the input is put in the container named :SIDE. BOXR can then look in the container whenever one of its lines needs an input to FD.

Bug Box

If BOXR doesn't work correctly, here are possible bugs:

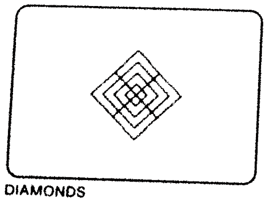
- You typed :SISE or some other spelling instead of :SIDE.
- You forgot to use a : (colon).
- You did not put a space between FD and :.
- You put a space between : and SIDE.
- You inserted an extra instruction in BOXR.
- You accidentally erased an instruction in BOXR.
- You typed a : in front of a number.

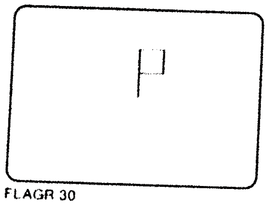


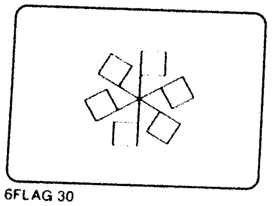
SQUARES

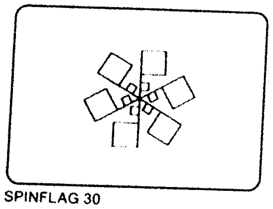
Here are some designs you can make using BOXR:

```
TO SQUARES
BOXR 10
BOXR 20
BOXR 30
BOXR 40
END
```


DIAMONDS


FLAGR 30


6FLAG 30

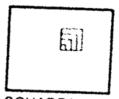

SPINFLAG 30

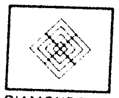
```
TO DIAMONDS
RT 45
REPEAT 4 [SQUARES RT 90]
END
```

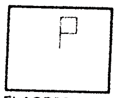
```
TO FLAGR :SIZE
FD :SIZE
BOXR :SIZE
BK :SIZE
END
```

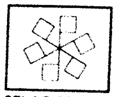
```
TO 6FLAG :SIZE
REPEAT 6 [FLAGR :SIZE RT 60]
END
```

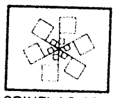
```
TO SPINFLAG :SIZE
6FLAG :SIZE
6FLAG :SIZE - 20
END
```


SQUARES


DIAMONDS


FLAGR30


6FLAG 30


SPINFLAG 30

Being able to control the size of a shape makes that procedure much more useful and interesting.

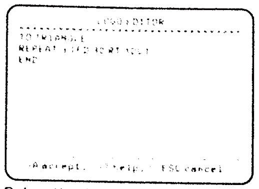
A Reminder About Saving: Before you go on, save your new procedure BOXR and the other procedures you defined so far in this chapter. But first erase from the workspace any procedures already saved or that you don't want. Think up a filename that will remind you what group of procedures you saved in this particular file. Later on it may be hard to remember.

Big Triangles and Small Triangles

You can also define a triangle procedure that takes an input. Type

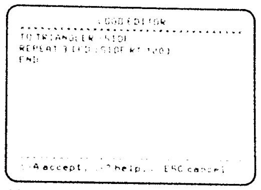
```
ED "TRIANGLE
```

Bug Box
If the screen displays the words `TO TRIANGLE` but no lines with commands, this means that the procedure TRIANGLE is not in the workspace. You need to load the file holding TRIANGLE.



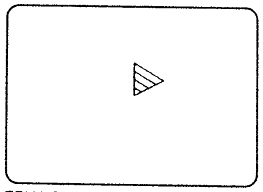
```
LOGO EDITOR
TO TRIANGLE
REPEAT 3 [FD 30 RT 120]
END

^A accept, ^? help, ESC cancel
```

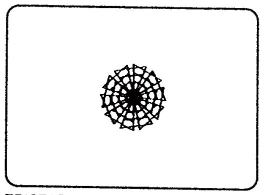
Before You Edit



```
LOGO EDITOR
TO TRIANGLER :SIDE
REPEAT 3 [FD :SIDE RT 120]
END

^A accept, ^? help, ESC cancel
```

After You Edit


TRIANGLES


TRISTAR

Now the Logo editor shows this procedure with the _ (cursor) under the T of TO.

```
TO TRIANGLE
REPEAT 3 [FD 30 RT 120]
END
```

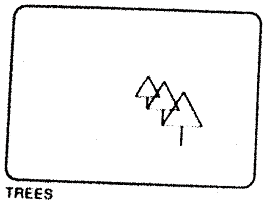
Change the name of TRIANGLE. And change `30` to `:SIDE`.

```
TO TRIANGLER :SIDE
REPEAT 3 [FD :SIDE RT 120]
END
```

You can make different designs using TRIANGLER. (Note that the title TREE2 distinguishes this procedure from the TREE1 procedure you defined in the last chapter.)

```
TO TRIANGLES
TRIANGLER 10
TRIANGLER 20
TRIANGLER 30
TRIANGLER 40
END
```

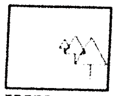
```
TO TRISTAR
REPEAT 10 [TRIANGLES RT 36]
END
```


TREES

```
TO TREE2 :SIDE
RT 30
TRIANGLER :SIDE
RT 60 FD :SIDE / 2
LT 90 BK :SIDE / 2
END

TO TREES
TREE2 30
TREE2 40
TREE2 50
END
```

TRIANGLES

TRISTAR


TREES

Another Reminder About Saving: Now save TRIANGLER and your other new procedures. Use the filename TRIANGLES2. But first use ERASE to get rid of TRIANGLE and the other procedures you already saved, like BOXR and DIAMONDS. Then type POTS to make sure you have in the workspace what you want to save.

Special Character

You learned this special character in this chapter:

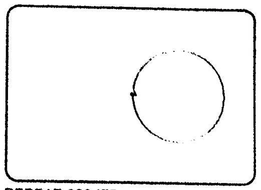
: (colon)	Prefixes a variable name.

Circles and Arcs

Circles and Arcs

The turtle can make curved as well as straight lines. Curves are made by repeatedly taking a small step and turning just a little bit.

■ The Turtle Draws Circles


REPEAT 360 [FD 1 RT 1]

To make a complete circle, the turtle has to turn 360 degrees. Try

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

The circle looks fine, but it takes a long time to draw. That's because the turtle needs to repeat the two instructions 360 times, as many times as drawing 90 squares!

To draw circles faster, you can make a little compromise.

```
REPEAT 36 [FD 10 RT 10]
```

This circle is less perfect. In fact, it is a 36-sided polygon. (A **polygon** is a shape with many sides. *Poly* means "many.") But the turtle draws it 10 times faster than the first circle.

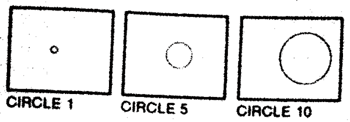
Why change the FD input to 10? What would happen if you left it as 1?

You can try some experiments to answer questions like that. Write a procedure for experimenting with circles of different sizes.

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

Now, try it with various inputs.

```
CIRCLE 1
CIRCLE 5
CIRCLE 10
```



Notice that the circle's size changes in proportion to its input. The larger the input number, the larger the circle. This is not surprising, because each circle has the same number of FORWARDs in it. The FORWARD distance determines the length of the circumference, the outer boundary of the circle.

Bug Box

If your circles look more egg-shaped than round, the bug is in your video monitor or TV set and not in Logo. The Logo command .SETSCRUNCH allows you to correct the picture by changing the height-to-width ratio of the screen. If you want to know what the current ratio is, type

```
PR .SCRUNCH
```

Include the . (period).

Logo responds by printing a number (normally 0.8) that is the current setting. To change the number, type

```
.SETSCRUNCH .9
```

Notice the space and . (period) before the 9.

Then clear the screen with

```
CS
```

and type

```
CIRCLE 10
```

If your circle still is not round, try

```
.SETSCRUNCH 1.0
```

Try other settings until you are satisfied.

Sometimes it is more convenient to choose the size of a circle by its radius (the distance from the center to a point on the circumference). When you use the CIRCLE procedure, you need to calculate the turtle's step size to get a circle of a particular radius. Why not let the computer calculate it? To do this, write another procedure that uses CIRCLE. Call it CIRCRAD.

```
TO CIRCRAD :RADIUS
CIRCLE 2 * 3.14 * :RADIUS / 36
END
```

How does CIRCRAD work? The input to CIRCLE in the first instruction line is a complex arithmetical operation but not too difficult to understand. The operation 2 * 3.14 * :RADIUS computes the circumference of a circle. The circumference has 36 FORWARDs. So divide the circumference by 36 to get the step size.

Now try

```
CIRCRAD 30
RT 90 FD 30
FD 30 HT
```

See if you can do any of these projects, using circles:

FLOWER TARGET FACE PEACE

The Turtle Draws Arcs

Many projects require only pieces of circles, called arcs. There is a simple way to get a piece of a circle right now. Run the CIRCLE procedure and quickly press (Ó)-(ESCAPE) to stop the turtle before it finishes drawing.

As you found out, this method has a little problem. You need very good coordination to produce the right sized arcs. The best way to control this situation is to add another input variable to the CIRCLE procedure. The input will allow you to vary the number of times the small steps and turns are repeated.

Change the procedure name to ARC at the same time.

```
EDIT "CIRCLE
```

The Logo editor shows the CIRCLE procedure.

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

Change the procedure so it looks like this:

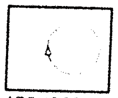
```
TO ARC :STEP :TIMES
REPEAT :TIMES [FD :STEP RT 10]
END
```

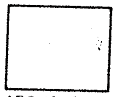
Leave the editor and try ARC with different inputs.

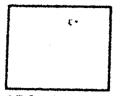
```
ARC 10 36
```

```
ARC 10 18
```

```
ARC 10 9
```


ARC 10 36


ARC 10 18


ARC 10 9

The way ARC works right now, you have to tell it how many times to repeat the little FORWARDs and RIGHTs to make an arc of the length you want. Another way is to input the length as so many degrees and let Logo calculate how many times to repeat the step and turn. So change ARC.

```
EDIT "ARC
```

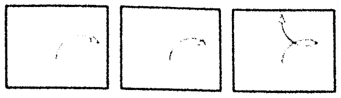
Change the variable name from :TIMES to :DEGREES.

```
TO ARC :STEP :DEGREES
REPEAT :DEGREES / 10 [FD :STEP RT 10]
END
```

Using :DEGREES as the second variable helps you think about what length you want an arc. For example, an input of 90 :DEGREES (which is one-fourth of 360), produces an arc that is a quarter of a circle in length. (Your input of so many degrees is divided by 10 to take into account that the turtle turns 10 degrees right each time the instruction in brackets is repeated.)

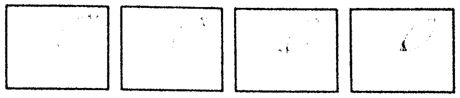
Now try these instructions:

```
ARC 6 120
RT 120
ARC 6 120
```



A fish!

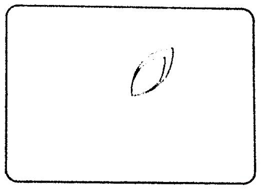
```
ARC 6 90
RT 90
ARC 6 90
RT 90
```

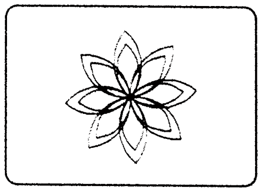


A petal!

Here are some more designs to make:

```
TO PETAL :SIZE
ARC :SIZE 90 RT 90
ARC :SIZE 90 RT 90
END
```


PETAL 10 PETAL 8


REPEAT 8 [PETAL 10 PETAL 8 RT 45]

With PETAL defined, try each of these instruction lines separately:

```
PETAL 10 PETAL 8
```

```
REPEAT 8 [PETAL 10 PETAL 8 RT 45]
```

```
REPEAT 4 [PETAL 8 RT 45 PETAL 10 RT !
45]
```

Notice that Logo lines can extend beyond one screen line. Logo marks **continuation lines** by putting an ! (exclamation point) in the last character position of the line. The rest of the text flows onto the next screen line.

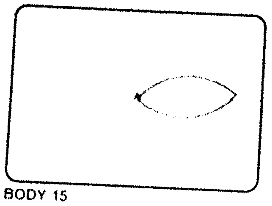
You might want to make a FLOWER procedure out of one of the designs you just did. If you already made a procedure named FLOWER, be sure to use a different name.

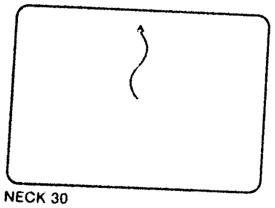
Here are some projects using arcs. See if you can make them! You will need to refer to Appendix A for the CIRCLER, CIRCLEL, ARCR, and ARCL procedures.

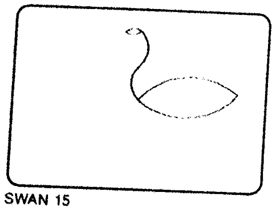
PACMAN SWAN HEART

Try making SWAN.

You need only two different shapes, PETAL and NECK. You can use PETAL to make BODY and HEAD. Type the definitions to make the parts of the swan.


BODY 15

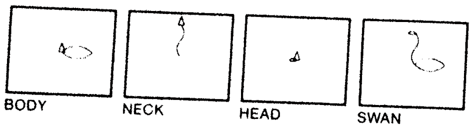

NECK 30


SWAN 15

```
TO BODY :SIZE
RT 45
PETAL :SIZE
LT 45
END

TO NECK :SIZE
LT 45
ARCR :SIZE 90
ARCL :SIZE 90
RT 45
END

TO HEAD :SIZE
LT 135
PETAL :SIZE
RT 135
END
```

Now do SWAN, the superprocedure.

```
TO SWAN :SIZE
BODY :SIZE
NECK :SIZE * 2
HEAD :SIZE / 6
END
```



Now you can draw different sized swans. In fact, you might like to make a setup procedure, like you did for TREE1 in Chapter 11. Then you could place swans in different places on the screen.

A Reminder About Saving: Before going to the next chapter, save your new procedures. A filename like CIRCARC would be good, but you can use separate files, if you prefer, for your circle and arc procedures. Maybe you want to keep SWAN and its procedures in a separate file. Be sure to use POTS to check your workspace before and after saving.

Logo Vocabulary

These are the Logo primitives you learned in this chapter (* denotes operation):

.SCRUNCH*
.SETSCRUNCH

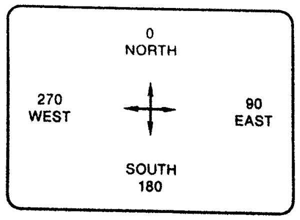
The Turtle's World

The Turtle's World

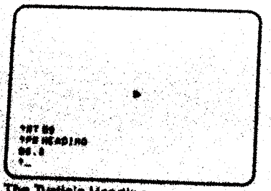
Recall from Chapter 2 that the turtle has a position and a heading. This chapter looks at precise ways to describe and control the turtle's position and heading.

The Turtle's Heading

The turtle's heading is described in degrees like a compass reading with 0 or north at the top of the screen. Then 90 degrees is directly east, 180 degrees is directly south, and 270 degrees is directly west. If you could mark the screen, this is how it would look:



When Logo starts up and after CS, the turtle's heading is 0. If the turtle's heading changes and you want to know exactly where it is pointing, you can get a compass reading of the turtle's heading at any time.



```
?RT 90
?PR HEADING
90.0
?_
```

The Turtle's Heading

Try

```
CS
RT 90
PR HEADING
```

Logo responds

```
90.0
```

HEADING outputs the direction the turtle is pointing.

SETHEADING or SETH is a command that sets the turtle's heading in a particular direction. SETH acts differently from RT or LT in that the end result does not depend on the current heading of the turtle. Try

```
SETH 90
RT 90
SETH 90
```

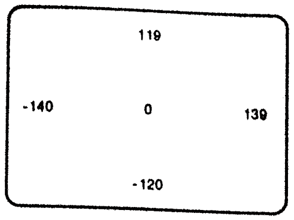
The Turtle's Position

Think of the screen as a grid divided into **coordinates**. The x coordinates run along the horizontal or side-to-side dimension of the screen. The y coordinates run along the vertical or top-to-bottom dimension of the screen.

Any point on the screen can be defined by an x-coordinate number and a y-coordinate number. When the turtle is at the screen's center, it is defined by [0 0]. (Logo displays the numbers as 0.0 0.0. It always shows the numbers on both sides of the decimal point.) These are its coordinates at that position. Any time the turtle moves to a new position, its location on the screen is defined by a new set of two numbers.

If the turtle is left of the center, then the first number (the x-coordinate) is prefixed by - (minus sign). If the turtle is below the center, then the second number (the y-coordinate) is prefixed by - (minus sign). If the turtle is both left of and below the center, then both coordinate numbers are prefixed by - (minus sign).

The screen's dimensions are approximately those in the illustration.



The command POSITION or POS outputs the coordinate numbers defining the turtle's current position. For example, type

```
PR POS
```

Logo responds

```
0.0 0.0
```

Change the turtle's heading back to 0 by typing

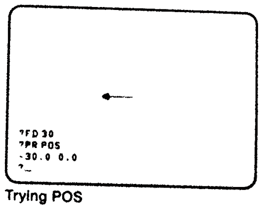
```
SETH 0
```

Then change its position and use POS to ask Logo to print its position.

```
LT 90
FD 30
PR POS
```

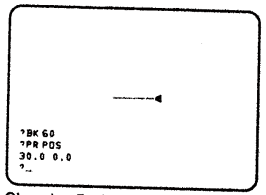
Logo responds

```
-30.0 0.0
```



```
?FD 30
?PR POS
-30.0 0.0
?_
```

Trying POS



```
?BK 60
?PR POS
30.0 0.0
?_
```

Changing Position

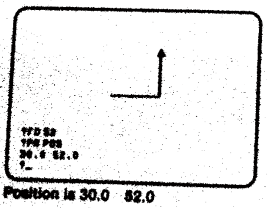
The turtle is 30 steps left of the center along the horizontal dimension. Vertically, it is at the center.

If you type

```
BK 60
PR POS
```

Logo responds

```
30.0 0.0
```



```
?FD 52
?PR POS
30.0 52.0
?_
```

Position is 30.0 52.0

Now the turtle is 30 steps right of the center along the horizontal dimension. Type

```
RT 90
FD 52
PR POS
```

Logo responds

```
30.0 52.0
```

The turtle is 30 steps right of the center and 52 steps above the center.

Bug Box
If you typed CS in between the above sets of instructions, you will not get these results. Remember, CS brings the turtle back to a [0.0 0.0] position.

SETPOS, which stands for set position, is a command that sets the turtle at a specific position on the screen. SETPOS acts differently from FORWARD or BACK in that the end result does not depend on the turtle's initial position. SETPOS does not change the turtle's heading. Type

```
SETPOS [50 -52]
```

Be sure to leave a space before the -52.

50 is the x-coordinate and -52 is the y-coordinate. This moves the turtle to 50 steps east of the center and 52 steps south.

Using POSITION to Draw

POS and SETPOS have many good applications. One of these is an easy way to draw a right-angle triangle. You just decide the lengths of the two sides joining in a right angle, and Logo does the rest with the help of POS and SETPOS.

You first record the starting position of the turtle. You do this by using the Logo command MAKE to create a variable name or container that stores the start position.

```
CS
MAKE "START POS
```

If you now say

```
PR :START
```

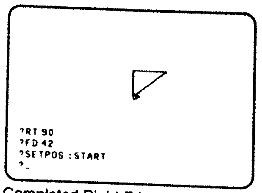
and the turtle was in the center of the screen when you typed MAKE "START POS, Logo responds

```
0.0 0.0
```

Your second step in drawing the right triangle is to have the turtle draw the two sides with the right angle in between.

```
FD 33
RT 90
FD 42
```

A right angle is 90 degrees.



```
?RT 90
?FD 42
?SETPOS :START
?_
```

Completed Right Triangle

Use the command SETPOS to bring the turtle back to its starting position.

```
SETPOS :START
```

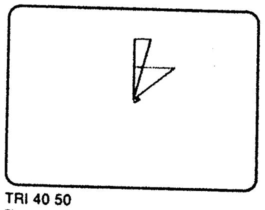
Remember, :START holds the coordinates POS gave it before.

The turtle is moved to :START. And since the pen is down, a line is drawn as the turtle returns to the starting point. This line is the third side of the triangle. And you didn't have to calculate the length! A procedure for this is

```
TO TRI :SIDE1 :SIDE2
MAKE "START POS
FD :SIDE1
RT 90
FD :SIDE2
SETPOS :START
END
```

Try

```
CS
TRI 40 50
SETH 0
TRI 75 20
```

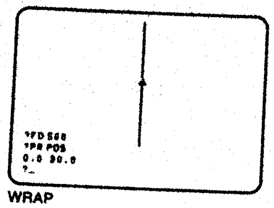


TRI 40 50
TRI 75 20

Changing the Turtle's Boundaries

There are three commands you can use to change the turtle's screen boundaries: WRAP, FENCE, and WINDOW.

Whenever you start up Logo, the turtle is able to WRAP. This means it can walk off one edge of the screen and on at the opposite edge without changing direction.



```
?FD 500
?PR POS
0.0 20.0
?_
```

WRAP

Type

```
CS
FD 500
PR POS
```

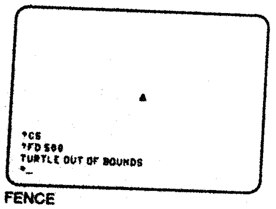
Logo responds

```
0.0 20.0
```

Notice that the turtle is actually 20 steps and not 500 steps from the center. It "wrapped" all the way around the screen, came back on at the bottom, and moved to a position 20 steps north of the center.

The screen boundaries can also be set up so that the turtle cannot move off the screen. Type

```
FENCE
```



```
?CS
?FD 500
TURTLE OUT OF BOUNDS
?_
```

FENCE

Now type

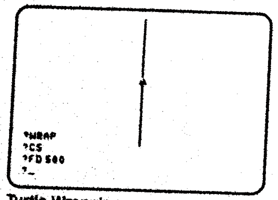
```
CS
FD 500
```

Logo responds

```
TURTLE OUT OF BOUNDS
```

The turtle screen acts this way until you type

```
WRAP
```



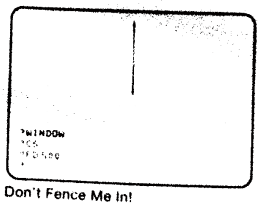
```
?WRAP
?CS
?FD 500
?_
```

Turtle Wrapping

Do it. Now the turtle again wraps around the screen. Type

```
FD 500
```

The WINDOW command allows the turtle to move off the screen without wrapping and without causing an OUT OF BOUNDS message. When you give the command WINDOW



```
?WINDOW
?CS
?FD 500
```

Don't Fence Me In!

and send the turtle out of screen bounds, it is invisible to you. The x and y coordinates are very large. Try

```
WINDOW
CS
FD 500
PR POS
```

Logo responds

```
0.0 500.0
```

The turtle is now 500 steps north of the center and out of view.

A CS command always restores the turtle to its center position on the screen. Changing WINDOW to FENCE or WRAP also restores the turtle to its center position.

Any time you want the turtle to wrap again, type `WRAP`.

A Reminder About Saving: This is a good time to save your new procedure TRI. Check first to see what is in your workspace. Procedures already saved in a file shouldn't be saved again.

Logo Vocabulary

These are the Logo primitives you learned in this chapter (* denotes operations):

Full Name	**Short Name**
SETHEADING	SETH
SETPOS	
WINDOW	
WRAP	
FENCE	
MAKE	
HEADING*	
POS*	

Exploring Polygons and Spirals

Exploring Polygons and Spirals

chapter 15

You can vary the size (angle) of each turn the turtle makes in a complete circuit of 360 degrees. A small angle, like 10, produces a circular shape, as you saw in Chapter 13. If you use angles that are considerably larger, you produce circles that have corners, or polygons. By varying the size of the forward step, as well as the size of the angle, you can get beautiful and surprising designs.

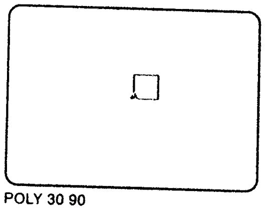
A Polygon Procedure

The following procedure takes two inputs. One specifies the number of turtle steps, and one specifies the amount to turn.

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE
END
```

Now try it!

```
POLY 30 90
```


POLY 30 90

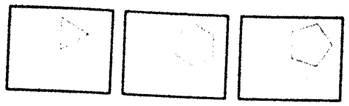
The turtle does not stop, because POLY keeps telling it to go forward and turn. To stop POLY and the turtle, press -ESCAPE. Logo responds

```
STOPPED! IN POLY
```

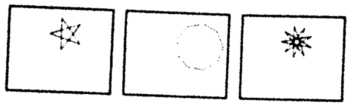
The message also tells you which instruction Logo was running when you stopped the procedure.

Here are some examples of POLY shapes. Try POLY with other inputs. You may want to use CS between each drawing.

```
POLY 30 120
POLY 30 60
POLY 30 72
```



```
POLY 30 144
POLY 30 45
POLY 30 160
```



Recursive Procedures: POLY is a **recursive procedure**. This means that instead of "calling" another procedure to help out, POLY calls itself.

A **call** is when one procedure names another, calling on it to do its work. This can be done because each procedure has its own name and is a completely separate piece. A procedure can call other procedures or even itself; procedures that call themselves are recursive.

You will learn more about recursion in Chapter 17.

A Spiral Procedure

The POLY procedure makes the turtle draw **closed shapes**. (A closed shape is made by a line that turns and comes back to its starting position.) In POLY, the turtle goes forward and rotates to get back to where it started. An exception occurs when the turtle turns 0 or 360 degrees (or a multiple of 360) on each round. In that case, it walks in a straight line.

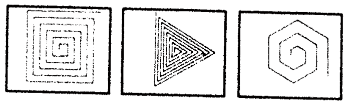
A spiral, on the other hand, is an **open shape**. The line does not return to its starting position. To draw a spiral, the turtle should not go back to where it started. Instead, the turtle should increase its forward step on each round so it gets farther and farther away from that point.

You can do this by increasing :STEP a little on the recursion line of the POLY procedure. Modifying the procedure POLY, you can make up a spiral-drawing procedure. Change POLY so the procedure looks like this:

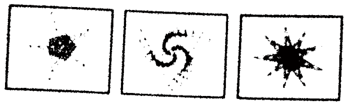
```
TO SPI :STEP :ANGLE
FD :STEP
RT :ANGLE
SPI :STEP + 6 :ANGLE
END
```

Now try SPI.

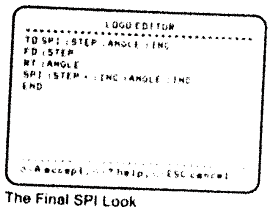
SPI 5 90	Remember, pressing (ó)-(ESCAPE) stops the procedure.
SPI 5 120	
SPI 5 60	(CONTROL)-(L) shows the whole turtle field.



```
SPI 5 144

SPI 5 125

SPI 5 160
```



You can change SPI and give it a third input, :INC, which SPI will add to :STEP instead of 6. Then you can change how much the turtle's step increases by choosing different numbers for the third input.



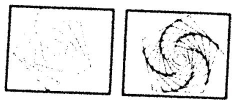
```
LOGO EDITOR
TO SPI :STEP :ANGLE :INC
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END

@-A accept, @-? help, @-ESC cancel
```

The Final SPI Look

```
TO SPI :STEP :ANGLE :INC
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END
```

Now try

```
SPI 5 75 1
SPI 5 75 2
```



You might want to stop the turtle at different places. Try other inputs. Remember to give three inputs each time.

A Reminder About Saving: Now save your new procedures, perhaps under the filename POLYSPI. Be sure to check your workspace before and after saving.

A Game Project

A Game Projec

chapter 16

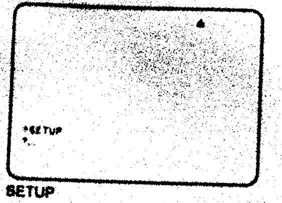
Using Logo, you can make a computer game you can play alone or with someone else. At the start of the game, a target and a turtle appear somewhere on the screen. The object is for you to try to get the turtle into the target with the fewest moves. Then the target moves to a new position and you try again.

For a first version, the moves can be directed by regular Logo commands like LT 45 or FD 80. Later you can refine the game by assigning keys to direct the turtle. Developing the game in stages illustrates the kind of **project management** to which Logo is well suited.

Setting Up the Basic Project

First you need to set up a target, and then you need to set up the turtle. You can write one procedure that can be used for both tasks. An example of a setup procedure is printed below. SETUP sets the turtle up in a random position on the screen, heading straight up.

```
TO SETUP
PENUP
RT RANDOM 360
FD RANDOM 100
SETHEADING 0
PENDOWN
END
```



```
?SETUP
?
```

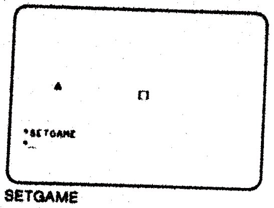
SETUP

The Logo operation RANDOM outputs a number that Logo makes up. This number is always smaller than the one you give RANDOM as its input. RANDOM changes the turtle's heading and position each time SETUP is executed.

In SETUP, the turtle turns some angle that can be as small as 0 degrees or as large as 359 degrees. The actual number is computed each time RANDOM is used. The input to FD is also a random number. Here the number can be no larger than 99.

Notice that the next-to-last instruction in SETUP leaves the turtle facing north. This is done to challenge the player to estimate the angle required to hit the target.

As you will see, you can use SETUP to set up the turtle as well as the target. It's a good idea to put the turtle back in the center before you use SETUP the second time.



```
?SETGAME
?
```

SETGAME

The following procedure, SETGAME, uses SETUP twice, first to set up the target position and second to set up the turtle position.

```
TO SETGAME
CS
SETUP
TARGET
PU SETPOS [0 0]
SETUP
END
```

SETGAME calls TARGET, so now you should define TARGET.

```
TO TARGET
BOXR 10
END
```

Go Get BOXR: Notice that TARGET uses BOXR, a procedure you defined in Chapter 12. Load the file that contains BOXR from your disk or else retype the definition now.

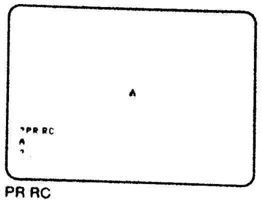
Use SETGAME a few times. It is hard at first. First run

```
SETGAME
```

and then when turtle and target are set up, turn the turtle right or left to aim and move it forward to shoot. As you practice, your skill in aiming and shooting improves.

Making a Key Into a Game Button

There are many kinds of **interactive programs** that you can write. These are programs where you and your Apple computer talk back and forth. You can have Logo ask questions that you answer in words or sentences. Sometimes you might want to respond by the press of a key. This requires using the operation READCHAR or RC. Type



```
?PR RC
A
?
```

PR RC

`PR RC` Or `READCHAR`

When Logo comes to READCHAR or RC, it waits for you to press a key.

Now type the character

```
A
```

RC receives the character A and passes it to the PR or PRINT command. The PRINT command then puts the character on the screen.

```
A
```

Notice that when you type the character, Logo does not wait for you to type anything else. It acts immediately. Try READCHAR or RC a few more times. Note that if you type only RC followed by (RETURN) and then type in a character like X, Logo responds

```
I DON'T KNOW WHAT TO DO WITH X
```

RC is an operation like HEADING or POS. It is used as an input to another command. For example, using MAKE, you can name a container (variable) to hold RC's output until it is needed, just as you named an output container in the procedure TRI.

```
MAKE "KEY RC
```

Now type a character. (Notice that the character does not appear on the screen when you type it. In other words, Logo does not "echo" what you type to it.)

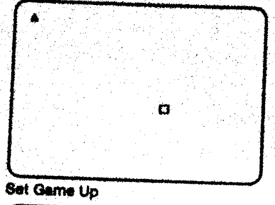
Now you can talk about this character you typed by using :KEY.

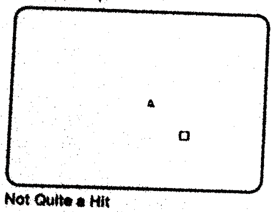
```
PRINT :KEY
```

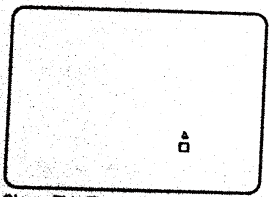
Logo responds

```
Z
```

or whatever character you typed.


Set Game Up


Not Quite a Hit


Closer This Time!

In the following procedure, you can use this idea of giving things names so you can talk about them.

```
TO PLAY
MAKE "ANSWER RC
IF :ANSWER = "S [STOP]
IF :ANSWER = "F [FD 10]
IF :ANSWER = "R [RT 15]
IF :ANSWER = "L [LT 15]
PLAY
END
```

If you type F, L, or R, the following happens:

- F makes the turtle move forward 10 steps.
- R makes the turtle turn right 15 degrees.
- L makes the turtle turn left 15 degrees.

Try it. Type

```
PLAY
```

and type R and L, followed by F.

In PLAY, :ANSWER is the variable name, the container for RC's output. PLAY then checks :ANSWER using the Logo primitive IF. IF requires two inputs. The first input is either TRUE or FALSE. The second input is a list of instructions to be carried out when the first input is TRUE.

In this procedure, you use the Logo operation = (equal to), which compares two inputs. This = operation outputs TRUE when the inputs are the same. It outputs FALSE when the inputs are not the same. This operation, like the arithmetic operations (+ - / *), has to be placed between its inputs.

Notice that PLAY is recursive. That is, the last line of the procedure calls PLAY. PLAY does not stop unless it has a bug or you press S.

Try it. Make turtle tracks all over the screen.

Expanding the Game Project

In this section, you will build a better target game out of SETGAME and PLAY. Some of the techniques used in this game are unfamiliar to you; others are not. Make a procedure, GAME, that uses SETGAME and then PLAY.

```
TO GAME
SETGAME
PLAY
END
```

Try

```
GAME
```

Perhaps you should raise the turtle's pen. It would also be nice if GAME printed some instructions or rules. Type

```
TO GAME
RULES
SETGAME
PU
PLAY
END

TO RULES
SPLITSCREEN
PRINT [HIT THE TARGET WITH THE TURTLE]
PRINT [TYPE R OR L TO TURN AND F TO ADV!
ANCE]
END
```

Now try it.

```
GAME
```

This is much better, but there is room for improvement. The game plays too slowly. Make it more challenging.

Give the player only *one* chance to land on the target. The player can turn the turtle many times, but will have only one chance to tell the turtle how far to go forward.

Here is the plan. After Logo sets up the scene for the game, it lets you start playing. You first make a try, then look to see if the turtle landed in the target. Logo should leave the screen unchanged for a time and then start the game again with a new target and position.

You should use a **top-down approach** to plan this revision of the game. That means you should plunge in and write the overall structure of the game before you know how you are going to write all the details.

```
TO GAME
RULES
SETGAME                This sets up each game.
PU
PLAY
WAIT 100               Logo waits a little while.
GAME                   This starts a new game.
END
```

You need to edit the procedure PLAY to allow only one chance to move the turtle forward. The challenge of the game is to judge the distance correctly the first time you try.

```
TO PLAY
MAKE "ANSWER RC
IF :ANSWER = "R [RT 15]
IF :ANSWER = "L [LT 15]
IF :ANSWER = "T [TRYLANDING STOP]
PLAY
END
```

The STOP command is very important. It makes the procedure stop after you have tried landing. Now edit RULES and change F to T.

```
TO RULES
PR [HIT THE TARGET WITH THE TURTLE]
PR [TYPE R OR L TO TURN AND T TO TRY LA!
NDING]
END
```

You've used the top-down approach again. You've changed PLAY to use a procedure named TRYLANDING. But you haven't defined TRYLANDING yet! Do it now.

```
TO TRYLANDING
PR [HOW FAR DO YOU WANT TO MOVE FORWARD!
]
FD FIRST READLIST
END
```

READLIST is like RC except you can type a word or list instead of a single character. (But unlike RC, it waits for you to press RETURN to signal that you are done.) READLIST outputs what you typed in the form of a list. FIRST READLIST outputs only the first word you typed (in this case a number). That is the only output you need to play the game.

Now you have written the whole game. To try it, type

```
GAME
```

Remember that you give the commands R and L to turn the turtle, and T to try landing on the target. After you type T, Logo waits for you to type a number and press RETURN.

You may be able to adapt this game and the techniques used in it to make other interactive games. You can also make many improvements to this game. For example, have Logo do a calculation of distance to figure out whether the turtle landed on the target. Logo could also keep track of your score.

A Reminder About Saving: Now save all your GAME procedures in a disk file. Be sure to check your workspace before and after saving.

Logo Vocabulary

These are the Logo primitives you learned in this chapter (* denotes operations):

Full Name	**Short Name**
IF (command or operation)	
STOP	
RANDOM*	
READCHAR*	RC
READLIST*	RL
FIRST*	

Special Character

This is a new arithmetic character you learned in this chapter:

=	Means equal to. Tells Logo to compare two numbers to see if they are equal.

Recursive Procedures

chapter 17

Recursive Procedures

One of the most powerful features of Logo is that you can divide a project into procedures. Each of these procedures has its own name and is a completely separate piece. A procedure can be called (used) by or can call any other procedure. Some procedures call themselves; such procedures are recursive. This chapter helps you understand recursive procedures and tells you how to stop them.

Understanding Recursive Procedures

You have already used recursive procedures. For example, POLY and SPI are both recursive. The fourth line of each procedure is the recursive call.

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE
END
```

```
TO SPI :STEP :ANGLE :INC
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END
```

POLY calls POLY as part of its definition, and SPI calls SPI.

Think about this process. Imagine that Logo has an unlimited supply of helpers who are creatures living in the computer. Every time you run a procedure or one procedure calls another, a Logo helper is asked to look up the definition of the procedure that was called. The helper then begins to carry out the instructions. It does this by calling on other helpers. Usually, several helpers are needed to carry out one procedure.

For example, when POLY is called, a Logo helper looks up the definition of POLY to see what commands it contains. Then this helper calls an FD helper. After the FD helper finishes, an RT helper is called. When the RT helper finishes, the first Logo helper calls a second set of FD and RT helpers. The same process occurs a third time, a fourth time, and so on. The first POLY helper never finishes its job, at least not as long as the procedure is running.

When you use POLY, the process continues until you press ⓓ-ESCAPE. Not all recursive procedures work this way. You can stop them in different ways. In fact, making up appropriate **stop rules** is an important part of writing recursive procedures.

Stopping Recursive Procedures

There are many kinds of stop rules. Here is a simple example, using the KEYP (for keypress) operation and the primitive STOP. Load the SPI procedure from your program disk, or if you didn't save it, type the following definition. If you did save it and have loaded it, change it so it looks like the following definition:

```
TO SPI :STEP :ANGLE :INC
IF KEYP [STOP]
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END
```

KEYP outputs TRUE whenever you press any key on the keyboard. Using it with IF lets you stop SPI by pressing any key. When you do so, Logo carries out the STOP command.

Test it by running SPI. Remember to give three inputs when you do. After SPI is running, press any key on the keyboard to test this stop rule.

Here is a different kind of stop rule. You could decide that SPI should stop if :STEP is greater than 150. So place SPI in the editor and replace the first line with

```
IF :STEP > 150 [STOP]
```

The character > means "is greater than." It is an arithmetic character that works similarly to the = character you used in the last chapter. But it compares two numbers to see if the first number is greater than the second number.

After you edit SPI, it looks like this:

```
TO SPI :STEP :ANGLE :INC
IF :STEP > 150 [STOP]
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END
```

Try the new SPI. If you don't like the stop rule, change the 150 to a different number.

Designing a stop rule for POLY is a little trickier. POLY completes a figure when the turtle returns to its starting state. This means that the turtle must turn a complete rotation or 360 degrees. But sometimes the turtle turns a multiple of 360 degrees—so you can't simply tell it to stop when it's turned 360 degrees.

In any case, you only need to know what the turtle's heading was when it started, and then compare it to the turtle's heading after each turn. So before the turtle starts drawing, you have to make Logo remember the turtle's heading. (This is what you did when you made TRI in Chapter 14. You told Logo to store the turtle's starting position.) Type

```
MAKE "START HEADING
```

Then POLY can regularly check to see if the turtle's current heading is the same as :START. When it finds they are the same, it knows that the turtle has made the polygon shape and returned to its starting place.

```
IF HEADING = :START [STOP]
```

When you put the stop rule into the procedure, make sure to place it right after the RT command. What happens if you put it before? POLY stops right away—even before the turtle starts drawing.

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
IF HEADING = :START [STOP]
POLY :STEP :ANGLE
END
```

Now try POLY.

There is a problem here. You didn't make Logo remember the starting heading before you ran POLY. That's why POLY doesn't work.

It is a good idea to put that action into a procedure. One way would be to write a superprocedure that remembers the heading and then runs POLY.

```
TO SUPERPOLY :STEP :ANGLE
MAKE "START HEADING
POLY :STEP :ANGLE
END
```

There is a better way of doing this. Instead of using MAKE, use :START as another input to POLY. Change POLY to look like this:

```
TO POLY :STEP :ANGLE :START
FD :STEP
RT :ANGLE
IF HEADING = :START [STOP]
POLY :STEP :ANGLE :START
END
```

To use POLY now, you must type **HEADING as the third input**. Try

```
CS RT 30
POLY 50 120 HEADING
```

You can write a controlling procedure that runs HEADING and gives its result to POLY. Then you won't have to type the third input name. This is a procedure separate from POLY but one that calls and runs POLY.

```
TO SUPERPOLY :STEP :ANGLE
POLY :STEP :ANGLE HEADING
END
```

Now SUPERPOLY does the whole job. When you run SUPERPOLY, you supply numbers for :STEP and :ANGLE, and the operation HEADING reads the turtle's current heading and supplies that number to POLY for its third input.

With their recursive and stop features, these new versions of the spiral and polygon procedures are very powerful. Try inventing new designs with them.

A Reminder About Saving: Now save your new poly and spiral procedures. Because they are improvements on the originals (which you defined in Chapter 15), save these new ones under the same filename. So first use ERASEFILE to delete the old POLYSPI file, then save the new procedures under POLYSPI. Be sure to check your workspace before and after saving.

■ *Logo Vocabulary*

This is the Logo primitive you learned in this chapter (* denotes operation):

KEYP*

Special Character

This is a new arithmetic character you learned in this chapter:

>	Means "is greater than." Tells Logo to compare two numbers to see if the first number is greater than the second number.

Sounds and Music

Sounds and Musi

chapter 18

You know how to make all kinds of turtle designs. Here you will learn how to make your own special sound effects and musical compositions. Apple Logo II has a command, TOOT, that produces sound through the speaker inside the case of your Apple computer. You can use TOOT for sound effects and music.

Using TOOT for Sound Effects

TOOT requires two inputs: **frequency** and **duration**. Frequency is the number of times in a second a sound wave vibrates. A high frequency produces a high-pitched sound, a low frequency a low-pitched sound. Duration is the length of a sound. A large duration value produces a long sound, a small duration produces a short sound. Try

```
TOOT 400 300
```

Now double the frequency with

```
TOOT 800 300
```

Now shorten the duration to half.

```
TOOT 800 150
```

Use REPEAT to make a series of notes.

```
REPEAT 8 [TOOT 800 300 WAIT 10]
```

WAIT helps you hear the note length.

Limits of Frequency and Vibration: When you use TOOT, the frequency can be in the range 3 through 65,535. Duration can be in the range 0 through any four-digit number. You may not find the extreme low or high values of frequency and duration very useful.

A frequency of 262 produces the middle C note, and a duration of 60 holds a note for approximately one second. The tuning A note has a frequency of 440.

This time try a pattern of sound. To save yourself work, define a procedure that you can use again.

```
TO WARBLE
TOOT 446 2 TOOT 830 2 TOOT 573 1 TOOT 3!
70 1
END
```

Now try

```
REPEAT 6 [WARBLE]
```

Heard that sound before? Using the Logo editor, define this procedure that uses WARBLE:

```
TO PINBALL
IF KEYP [STOP]
CLEARTEXT
REPEAT RANDOM 10 [WARBLE]
PR [YOUR SCORE IS OUT OF SIGHT!]
WAIT RANDOM 35
PINBALL
END
```

Simulation pinball! The CLEARTEXT command makes it easier to read the score! The next line changes the length of each warble. The WAIT and RANDOM commands in the sixth line make different length pauses between each warble.

Using TOOT for Music

Musical effects are also easy to make using TOOT. In fact, with TOOT and other commands, you can define a procedure that gives you a musical keyboard. Type

```
TO ALPHAPLAY
MAKE "NOTE ASCII RC
TOOT :NOTE * :NOTE / 10 10
ALPHAPLAY
END
```

ASCII is a primitive that assigns a code number to each key of the keyboard. When you press a key, the RC primitive determines which key was pressed, and then Logo places the ASCII code number of that key in the variable called NOTE. The value of NOTE is used to calculate the frequency input of TOOT. The duration is 10. (You can change the duration input if you want to play music at a faster or slower tempo.)

Now try

```
ALPHAPLAY
```

Press various keys on your keyboard. Any keys. After you get familiar with the key arrangement, you can play a little tune by ear.

But you can also play a tune by first defining a procedure. For example, define this procedure that plays a familiar tune:

```
TO TUNE
TOOT 262 56 / 8
TOOT 349 56 / 4
TOOT 262 56 / 8
TOOT 262 56 / 8
TOOT 294 56 / 4
TOOT 262 56 / 2
TOOT 330 56 / 4
TOOT 349 56 / 2
END
```

Run

```
TUNE
```

Recognize it?

Now see if you can make another procedure that plays a tune, one that is familiar or one that you invent.

Musical Tips: If you know the musical scale, you can use these tips:

- The frequency for middle C is 262, as you learned earlier. To go up eight notes (an octave) to the C above middle C, double the frequency of middle C to 524. (You can double the frequency of any note to go up the scale an octave.) To go down an octave from C or any note, divide the frequency of that note by 2.
- To know what frequency value to give to notes in the range less than one octave, like D, E, F, and so on, use the table of frequency values under the TOOT entry in the *Reference Manual* in the chapter called "The Outside World." The table also gives you the frequency values for sharps. (The same sharp values also stand for flats.)
- The duration varies with the tempo of a tune. TUNE has a slow tempo, making 56 the duration of a whole note. You may use a smaller value to get a faster beat or tempo, or a larger value to get a slower beat or tempo. Whatever tempo you choose, divide the whole note duration by 2 for a half note (56 / 2), by 4 for a quarter note (56 / 4), by 8 for an eighth note (56 / 8), and so on. Remember, Logo can do the division for you, as it does in TUNE.

A Reminder About Saving: Don't forget to save any of these music procedures you want to keep. But be sure to check your workspace before saving.

Logo Vocabulary

You learned these Logo primitives in this chapter (* denotes operation):

TOOT
ASCII*

What Next?

What Next?

Now that you have begun your Logo explorations, you may be interested in exploring additional projects in Logo and in learning more about what other people are doing with Logo. Sharing ideas is an important part of the Logo culture.

Clubs, Newsletters, and Magazines

There are clubs set up all over the world for people using Apples. Many of these user groups have or know of spin-off groups for Logo users. The International Apple Core, a nonprofit organization that provides support and information to Apple users, can provide you with a listing of clubs in your area.

International Apple Core, Inc.
908 George Street
Santa Clara, CA 95050

Several newsletters are published for Logo users. You may wish to subscribe to one of them. Here is a partial listing:

The National Logo Exchange. Published by the Posy Collection, P.O. Box 5341, Charlottesville, Virginia 22905.

Turtle News. Published by the Young People's Logo Association, P.O. Box 855067, Richardson, Texas 75085.

Logo Newsletter. Published for adults by the Young People's Logo Association, P.O. Box 855067, Richardson, Texas 75085.

Logophile. Published by the Educational Computing Organization of Ontario, Logo Special Interest Group, Queens University, Kingston, Ontario, Canada.

Newsstand computer magazines and many educational publications regularly publish articles about Logo. Also, your local library is a good source of information on articles and publications about Logo. Here is a list of some of the articles that have been written about Logo:

"Why Logo?: Logo Is Designed to Encourage Development of Problem-Solving Skills," by Brian Harvey. Published in *BYTE* magazine, Vol. 7, No. 8 (Aug. 1982), pages 163-193.

"What is Logo?" by Molly Watt. Published in *Creative Computing* magazine, Vol. 8, No. 10 (Oct. 1982), pages 112-129.

"Logo Music Projects: Experiments in Musical Perception and Design," by Jeanne Bamberger. Published in *Logo Memo 52*. Cambridge, Massachusetts: MIT D.S.R.E. (May 1979).

"One Child's Learning: Introducing Writing With a Computer," by Robert W. Lawler. Published in *Logo Memo 56*. Cambridge, Massachusetts: MIT D.S.R.E. (Mar. 1980).

Books

Here are just some of the books that have been written about Logo. Check your library or bookstore for more.

Logo for the Apple II, by Harold A. Abelson. Published by Byte Books, McGraw-Hill, 1983.

Turtle Geometry: The Computer as a Medium for Exploring Mathematics, by Harold A. Abelson and Andrew diSessa. Published by MIT Press, 1981.

Mindstorms: Children, Computers, and Powerful Ideas, by Seymour Papert. Published by Basic Books, 1980.

Discovering Apple Logo, An Invitation to the Art and Pattern of Nature, by David Thornburg. Published by Addison-Wesley, 1983.

Learning With Apple Logo, by Daniel Watt. Published by Byte Books, McGraw-Hill, 1983.

You can get a complete bibliography on Logo materials by writing directly to

Logo Computer Systems Inc.
220 Fifth Avenue, Suite 1604
New York, NY 10001

or

Logo Computer Systems Inc.
9960 Cote de Liesse
Lachine, Quebec H8T 1A1
Canada

And Now...

This introduction to Apple Logo II is finished, but your adventure starts now. It's your turn to create your own projects and discover the world of Logo.

Logo has many other capabilities that are not covered in this manual. All of them are described in the *Apple Logo II Reference Manual*.

Have fun!

Special Circle and Arc Procedures

This appendix defines six special circle and arc procedures: CIRCLER, CIRCLEL, ARCR, ARCL, ARCR1, and ARCL1. R and L at the end of each name tells which way the curve is turning.

You need two of these special procedures—ARCR and ARCL—to successfully use the SWAN procedure in Chapter 13. These procedures can also be useful for any of your own projects requiring very precise angles or symmetry.

Here are the procedures:

```
TO CIRCLER :RADIUS
ARCR :RADIUS 360
END

TO CIRCLEL :RADIUS
ARCL :RADIUS 360
END

TO ARCR :RADIUS :DEGREES
ARCR1 .174532 * :RADIUS :DEGREES / 10
REMAINDER :DEGREES 10
END

TO ARCL :RADIUS :DEGREES
ARCL1 .174532 * :RADIUS :DEGREES / 10
REMAINDER :DEGREES 10
END
```

```
TO ARCR1 :STEP :TIMES :REM
REPEAT :TIMES [RT 5 FD :STEP RT 5]
FD :STEP / 10 * :REM RT :REM
END
```

```
TO ARCL1 :STEP :TIMES :REM
REPEAT :TIMES [LT 5 FD :STEP LT 5]
FD :STEP / 10 * :REM LT :REM
END
```

The number .174532 in ARCR and ARCL is the result of computing 2 x pi / 36 when pi is rounded to 3.1416 and 36 is the number of sides of the polygon.

When two arcs curving in opposite directions are drawn, one right after the other, the result is a graceful s-curve. Try ARCR followed by ARCL. The variable :REM in ARCR1 and ARCL1 draws a special extension (a less curved line) at the end of the arc. Try ARCR1 followed by ARCL1. (You may need to make the input values smaller than those you used with ARCR and ARCL.)

Glossary

bug box: A special box in this manual, marked with an image of the Logo spider, to help you diagnose and correct difficulties you may encounter during the tutorial.

call: A request to Logo from the keyboard or from a procedure to execute a named procedure. For example, in the SWAN program, the SWAN superprocedure calls the procedure HEAD (as well as others) and the HEAD procedure calls the procedure PETAL. See also **procedure** and **recursive procedure**.

closed shape: A shape or form made by a line that returns to its starting point; for example, a triangle. Compare with **open shape**.

command: A procedure, either a primitive or one you define, that carries out your instruction and does not return a value. FORWARD (FD) is an example of a command. Compare with **operation**.

continuation line: A line of Logo text that extends or wraps around to a second line, marked at the break point with an ! (exclamation point).

coordinate: A number that describes the horizontal or vertical position of the turtle on the Logo graphics screen grid. In its "home" state, the Logo turtle is at coordinates [0 0]. See also **position**.

cursor: The blinking bar character (_) that shows you where the next character you type will appear on the screen.

define: To make up a Logo procedure by listing Logo instructions using the TO and END commands, followed by (ó)-A. See also **procedure**.

duration: The length or persistence of a sound in time, as in music. Compare with **frequency.**

edit: To modify text you have already typed, by deleting, adding, and so on. You can edit text as you type, or you can edit already defined procedures by using the Logo editor. See also **editor**.

editor: That portion of Logo that holds procedures you would like to modify. Normally entered using the EDIT (ED) command; for example, ED "SQUARE.

equiangular triangle: A triangle whose three angles are of equal size, and consequently, whose sides are of equal length. Same as an equilateral triangle.

equilateral triangle: A triangle whose three sides are of equal length, and consequently, whose angles are of equal size. Same as an equiangular triangle.

execute: To command Logo to carry out or run the instructions you've placed in a procedure. You do this by typing the name of the procedure from the keyboard or inserting it within another procedure.

file: A collection of information stored on a disk as a unit with its own name.

format: To prepare the surface of a disk so that Logo can store files on it.

frequency: The number of times in a second (or other measure) a sound wave vibrates, as in music. Compare with **duration**.

full screen: The screen used wholly for graphics—no text. Compare with **split screen** and **text screen**.

heading: The exact direction, given in degrees, the turtle is pointed at any given moment. Defined by a number from 0 through 360 and given as output by the operation HEADING. Compare with **position**.

input: The value you give to a variable. You can provide an input directly from the keyboard when you type a command that requires one (for example, REPEAT 8 [PETAL 8]) or from within a procedure when an operation or another procedure assigns a value to a particular variable (for example, FD FIRST READLIST, where FIRST READLIST provides input to FD). See also **variable**.

interactive program: A program that pauses to ask a question or otherwise to prompt the user to respond.

open shape: A shape or form made by a line that does not come back to its starting position; for example, a spiral. Compare with **closed shape**.

operation: A procedure, either a primitive or one you define, that outputs a value. For example, POS is an operation. Compare with **command**.

output: A value produced by an operation or a procedure. It can be read by you from the screen (for example, the number printed on the screen if you type PR PENCOLOR and (RETURN)), stored in a variable in a procedure (for example, MAKE "START POS in the procedure TRI), or used in a procedure as input to a command (for example, FD :STEP, where the variable STEP provides an output value that becomes input to FD).

polygon: A shape with many sides.

position: The exact location of the turtle on the screen at any given moment. Defined by two coordinates and given as output by the operation POS. Compare with **heading**.

prefix: Usually the first word after a Logo disk command; for example, /MYLOGO in SAVE "/MYLOGO/SQUARES. The volume name you establish with the Logo command SETPREFIX. See also **volume name**.

primitive: A procedure that Logo always knows how to execute because it is built into the Logo system, like HIDETURTLE (HT). See also **procedure**.

procedure: A single instruction or set of instructions (defined by you or built into Logo as a primitive) designed to perform a specific task. See also **superprocedure**.

program: Several procedures designed to work together, called and controlled by a superprocedure. See also **procedure** and **superprocedure**.

project management: The design and development of a programming project in a systematic way.

prompt: The ? character on the screen that tells you Logo is waiting for you to type something.

recursive procedure: A procedure that calls itself by naming itself and thus continues to run. See also **call**.

run: See **execute**.

setup instructions: Instructions that position the turtle and perhaps draw a shape before the principal command is given.

split screen: The combined turtle graphics and text screen, where the bottom four lines, below the picture, are reserved for text. Compare with **text screen**.

state: The exact position and heading of the turtle at a given moment. See also **position** and **heading**.

superprocedure: A procedure that calls one or more other procedures. See also **procedure** and **program**.

text screen: The screen used only for text, no graphics. It can hold 24 lines from top to bottom. Compare with **split screen**.

top-down approach: An approach to designing a computer program by starting with the overall structure of the program before working out all the details.

turtle: The triangular shape on the Logo graphics screen that shows where lines are being or will be drawn and in what direction.

turtle graphics: Pictures drawn on the screen using the Logo turtle. See also **turtle**.

variable: A word or character that acts as a container to hold a value. It can be used in the title line (for example, TO SQUARE :SIDE) or body of a procedure. It is always preceded directly by a colon (:).

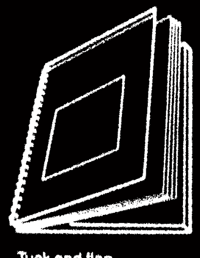
volume name: The name given a formatted disk. It is also the name of the directory that lists the file contents of the disk. For example, LOGO is the name of the Logo volume and disk directory. See also **prefix**.

workspace: The part of the computer's memory where Logo programs and variables are stored temporarily as you define them or after you load them from a disk.

Index

■ *Acknowledgments*

This project would not have been completed without the significant contribution of the Apple Logo team of Apple Computer, Inc. All the staff members of Logo Computer Systems Inc. have been extremely important to the development of this project. Special contributions were made by the following persons: Renaud Nadeau, Mario Carriere, Sharnee Chait, Brian Silverman, Michael Quinn, Rene Yelle, Effie Maniatis, Pascal Bekaert, and Barbara Mingie.



Tuck end flap
inside back cover
when using manual.

apple computer

20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010
TLX 171-576

Printed in U.S.A.

030-0863-A
A2L4034