ATARI LOGO

Introduction to Programming Through Turtle Graphics

ATARI LOGO

**Introduction to Programming
Through Turtle Graphics**

Disclaimer of all Warranties and Liabilities

Logo Computer Systems, Inc., makes no warranties, expressed or implied, concerning the quality, performance, merchantability or fitness of use for any particular purpose of this manual and the software described in this manual. This manual and the software described in this manual are sold ''as is ''. The entire risk as to the quality and performance of these goods is with the buyer; if the goods shall prove defective following their purchase, the buyer and not the manufacturer, distributor or retailer assumes the entire cost of all necessary servicing, repair and replacement and any incidental or consequential damages. In no event will Logo Computer Systems, Inc. be responsible for direct, indirect, incidental or consequential damages relating to the purchase or use of these goods, even if Logo Computer Systems, Inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Notice

Logo Computer Systems, Inc. and ATARI, Inc., reserve the right to make any improvements and changes in the product described in this manual at any time and without notice.

Copyright and Trademark Notices

This product and all software and documentation in this package (ROM cartridge, Manuals and Reference Guide) are copyrighted under United States Copyright laws by Logo Computer Systems, Inc.

© 1983 Logo Computer Systems, Inc.

ATARI is a registered trademark of Atari, Inc. (a Warner Communications Company), used by permission.

Table of Contents

ATARI Logo:
An Introduction

Compared with languages such as English or French, the computer language Logo has a small number of words and rules of grammar. Logo comes with a core of primitives — simple commands built into the language. These primitives let you perform operations such as addition, subtraction, multiplication, division, and manipulating words and lists of words. ATARI Logo is easy to work with. The primitives and initial programs you create, help to build other more complex programs.

This guide focuses on Turtle Graphics. Turtle Graphics provide a friendly introduction to programming. It lets you see how your programs work so you can develop an intuitive understanding of programming.

The aspects of Turtle Graphics unique to ATARI Logo are discussed in Chapters 16 and 17, and Appendices B, C of this book. You can also refer to Chapters 1 and 6 of the *ATARI Logo Reference Manual.*

Using This Guide

Though not a comprehensive guide to ATARI Logo programming, this guide gives you the information you need to get started. The activities presented here will give you a sense of the power of this high-level language. You'll quickly learn how to write procedures, how to edit, and how to save and retrieve your work. Refer to the Table of Contents and Index in the *ATARI Logo Reference Manual* for details.

Throughout this manual, green text is used to represent what you type on the computer. Black text is used to represent what the computer displays.

The sections labeled "Bug Box" address potential problems and suggest ways to fix them. A "bug" is something that doesn't work. Happily, ATARI Logo provides a relaxed learning environment where you can explore different ways of doing things. "Debugging" or editing a program can be a fun, and rewarding experience.

Getting Started

To use the ATARI Logo Cartridge, you need an ATARI Home Computer and a TV set or monitor. If you want to save programs, you need an ATARI Disk Drive or ATARI Program Recorder.

For specific questions about the operation of your ATARI Home Computer, refer to the computer owner's guide. You'll find your computer and ATARI Logo easy to operate. To load ATARI Logo into your computer:

1. With the computer off, turn on your TV set or monitor. If you have one, turn on your ATARI Disk Drive and wait for the busy light to go off. If you are not using a disk drive, skip to step 3.

2. Insert the ATARI Master Diskette in the disk drive and close the disk drive door. You may also use a data diskette if it contains DOS (Disk Operating System) files.

3. Insert the ATARI Logo Cartridge into the console's cartridge slot and turn the computer on.

After a moment you'll see on the screen:

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

The ? (question mark) is the prompt symbol. When ? is on the screen, you can type something. The ■ is the cursor. It shows where the next character you type will appear.

Bug Box

If you have problems getting started, make sure the computer is properly connected to the TV or monitor, and that the ATARI Logo Cartridge is correctly inserted. Then repeat steps 1, 2 and 3.

The Keyboard

The ATARI Home Computer keyboard is set up like a typewriter. For practice, type any word or sentence, though it probably won't make sense to Logo. For example, type:

HELLO THERE

Press RETURN. Logo will respond

I DON'T KNOW HOW TO HELLO

Feel free to experiment on the keyboard. You won't hurt the computer or the ATARI Logo Cartridge. You can always start Logo again by turning the computer off and on again.

Character Keys

Character keys — A, B, C, 7, ;, $, etc. They include letters of the alphabet, numbers and punctuation marks.

RETURN

In Logo the RETURN key serves a programming function. It tells Logo: "Now do what I just typed." Press RETURN when you want Logo to obey your instructions.

SPACE BAR

The SPACE BAR prints an invisible but important character called space. Logo uses spaces as word separators. For example, Logo would interpret THISISAWORD as a single word and would interpret THIS IS A WORD as four words.

SHIFT

Holding down SHIFT while pressing some character keys changes that particular character key's meaning in Logo. For example, if you hold the SHIFT key down and press], Logo will print] (close bracket) on the screen.

The bracket, [ ], symbols are very important in Logo. Do not confuse them with parentheses, ( ), which are SHIFT (and SHIFT).

To make a shift character, always press the SHIFT key first and then hold it down while typing the other key.

CTRL (CONTROL)

The CTRL key can change character keys into function keys. Press it alone and nothing happens; hold it down and press a certain character key, and something happens. These key combinations do not always print out on the screen, but Logo responds to them.

CTRL Arrow Keys

CTRL ← will move the cursor one space to the left and CTRL → will move the cursor one space to the right.

The arrow keys are useful editing keys. They move the cursor in the direction in which they point without affecting the text already there. Note: CTRL ↑ and CTRL ↓ only work in the ATARI Logo Editor. Once the cursor is positioned, you can insert or delete characters. To insert text simply position the cursor and begin typing.

DELETE BACK S (DELETE BACK SPACE)

Erases the character to the left of the cursor.

BREAK

The BREAK key tells Logo to stop whatever it is doing. It will also get you out of the ATARI Logo Editor without executing the changes. When you press BREAK, Logo types

```
STOPPED!
?■
```

then, lets you type the next instruction.

ESC

The ESC (ESCape) key is used to exit the ATARI Logo Editor. This key is discussed along with other special editing keys in Chapter 4, page 26 .

ATARI Key (人) or Reverse Video Key (◨)

If you press the (人) or (◨) key and then type a character key, the character appears in reverse video on the screen (dark character on a light background). You can return to the regular display by pressing the key a second time.

CAPS LOWR (CAPS)

When you first turn on your ATARI Home Computer, anything you type will appear in all uppercase letters. Press the CAPS LOWR key, now only lowercase letters are produced. ATARI Logo primitives must all be typed in uppercase letters. Therefore, if you accidentally press the CAPS LOWR key, Logo will no longer understand your instructions.

SHIFT CAPS LOWR Combination

To lock the keyboard in uppercase, simply hold the SHIFT key and then press the CAPS LOWR key.

SYSTEM RESET (RESET)

Do not use this key once you have booted Logo. You will lose everything in memory.

The Print
Command

Try the following to get a feel for Logo:

PRINT [HELLO THERE]

The instruction appears on the screen, but is not obeyed until you press RETURN. Logo will then respond

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]

Do not press RETURN.

Press DELETE BACK S until the screen shows

PRINT [I AM THE█

Now type the rest of the line.

PRINT [I AM THE GREATEST]

Press RETURN and your instruction is obeyed.

I AM THE GREATEST

The DELETE BACK S key is one of several editing functions Logo provides so that you can change what you have typed without having to type the entire instruction over again. You will be introduced to more editing keys in later sections of this guide.

You can have Logo print some other sentences by typing PRINT and enclosing what you want typed in [ ] (brackets) as illustrated in the previous example.

Clearing off the Screen

A useful command is CT (which stands for Clear Text). It clears the screen of text and starts the text at the top of the screen.

CT   Press RETURN.

The screen clears and the *cursor* appears at the top of the text screen.

Writing Procedures

You can make up *new* commands in Logo by defining procedures. You can use the command TO to signal your intention to Logo. On the same line you tell Logo the name of the procedure. Then, you tell Logo what you want the procedure to do. For example, let's define a procedure, GREET, so that whenever you type GREET Logo will type

HI THERE
BYE NOW
?█

Type TO followed by the name of the procedure to start defining the procedure.

TO GREET   Type a space to separate the words TO
                                  and GREET.
                                  Press RETURN.

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

>PRINT [HI THERE]   Press RETURN after each instruction.
>PRINT [BYE NOW]
>END

The word END signals the procedure TO GREET that you have finished the procedure definition. Logo will now type

```
GREET DEFINED
?■
```

Logo again uses ? as the *prompt* symbol. Run the procedure by typing

GREET   Press RETURN

Logo types

```
HI THERE
BYE NOW
?■
```

Bug Box
If GREET does not work, you probably typed something incorrectly. Soon you will learn how to edit your procedure, that is, change the parts you don't like. In the meantime, just try writing another procedure. Give it another name, let's say GREET1. (Note that numbers can be used as letters in Logo commands.)

```
TO GREET1
>PR [HI THERE]
>PR [BYE NOW]
>END            Press RETURN.
```

Run this procedure by typing

GREET1

Logo should type

```
HI THERE
BYE NOW
?■
```

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

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

Press RETURN.

If you have a GREET1 procedure, you can use it instead of GREET.

Logo responds

```
HI THERE
BYE NOW
HI THERE
BYE NOW
HI THERE
BYE NOW
HI THERE
BYE NOW
HI THERE
BYE NOW
?■
```

You could have REPEAT run GREET a thousand times.

```
REPEAT 1000 [GREET]
```

If you want to interrupt the process, press BREAK. Logo responds

```
STOPPED! IN GREET
?█
```

Note: In the next chapters of this guide, we will not always remind you to press RETURN or to type a space between words.

ATARI Logo Vocabulary

The following Logo primitives were introduced in this chapter.

PRINT (PR)
CT
TO
END
REPEAT

We have also used several special keys.

[
]
BREAK
DELETE BACK S
RETURN

Meet
the Turtle

In this section, you will learn how to program by controlling a computer creature known as a turtle. The first Logo turtle was a robot; it looked like a big canister and moved about on wheels. It was attached to a computer by a long cable and could draw lines on the floor, which normally was covered with paper.

Our turtle, however, lives on the computer screen. It also has a pen and can write with it on the screen. Logo has many commands that you can use to control the turtle. This section introduces you to some of the most important turtle commands.

To see the turtle, give the command ST. ST stands for Show Turtle.

ST   Press RETURN.

A small turtle will appear at the center of the screen. Notice that the turtle is shaped in a way that shows its position and its heading. The *position* and *heading* are called the turtle's *state*. At any time, the graphics turtle is at a specific position and is facing in a specific heading. 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.

Notice also that the *prompt* symbol and the *cursor* are now near the bottom of the screen. The commands you type will now appear on the last five lines of the screen.

Changing the Turtle's State

FORWARD
Now, let's get the turtle to do something using the command FORWARD. FORWARD needs to be followed by a number. This number is called an input and indicates how many steps the turtle moves.

Type the following command and remember to press RETURN when you want Logo to "do it".

FORWARD 50

Notice that the turtle changes its position but not its heading.

We chose 50 as the input, but you could use any number.

The space between FORWARD and 50 is very important. It distinguishes the word FORWARD from the word FORWARD50. Actually, you can type extra spaces between words and Logo will ignore them.

Bug Box
Often as you interact with Logo you will make mistakes. Many of these will be typing errors. Perhaps the most common typing bug is not leaving a space between a command and its inputs. For example, FORWARD is a command that expects a number as its input. FORWARD is part of Logo's vocabulary. Thus, FORWARD 50 is guaranteed to cause the turtle to move forward 50 steps. But, FORWARD50 is a different word from FORWARD, and one that is not defined (unless you have defined it yourself).

The difference between the two instructions is merely a space between words. The difference between FRWARD and FORWARD is merely an O, but to Logo these differences are very significant.

If you type

FRWARD

Logo responds

I DON'T KNOW HOW TO FRWARD

Check what you have typed with what you meant to type.

RIGHT

To change the turtle's heading, we tell it to turn **RIGHT** or **LEFT** a specified number of degrees. You can, of course, tell it to turn any number of degrees.

In the following example, we tell the turtle to turn **RIGHT 90** degrees.

RIGHT 90

The turtle turns 90 degrees to the right of where it had been heading previously. Notice the turtle changes its heading, not its position on the screen.

BACK

BACK makes the turtle back away from its current position; it changes the turtle's position only.

BACK 50

Again, 50 is chosen as an example, but you can choose any number.

FORWARD 50   RIGHT 90   BACK 50

LEFT

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

LEFT 45

The turtle turns 45 degrees to its left. It changes only its heading, not its position. The effect of the turn is seen more clearly if you now tell the turtle to go 25 steps.

FORWARD 25

CS

You might want to clear the screen and start again. The command **CS**, which stands for Clear Screen, will erase the turtle tracks from the screen and put the turtle in its startup state at the center of the screen heading straight up.

CS

Try experimenting with these state-change commands on your own. Remember, you can always use **CS** to clear the screen, put the turtle back in the center of the screen and start over.

LEFT 45   FORWARD 25

ATARI Logo Vocabulary

The following Logo primitives were introduced in this chapter. Some are followed by a short form, in parentheses, that can be used in place of the full primitive word.

BACK (BK)
CS
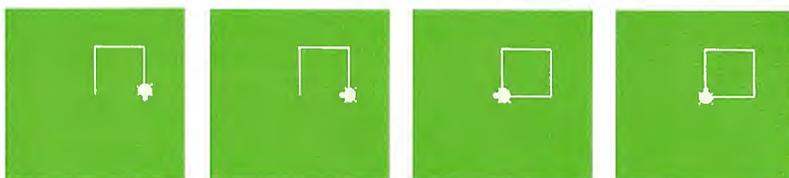FORWARD (FD)
LEFT (LT)
RIGHT (RT)
ST

Using the commands FORWARD and RIGHT or LEFT, we can make the turtle draw a square. For convenience, we will often use the short names for those commands.


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



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

If we used 50 instead of 30, the turtle would draw a bigger square. We could use any number.

Let's define a new command in Logo to get the turtle to draw a square. New commands in Logo are programs or procedures written by you. Each time you want a square, you can use your new procedure rather than retyping the individual instructions.

To define a new command, you should first choose a name. Let's use SQUARE because that seems natural, but any name will do. You could then use the Logo command TO and define SQUARE 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 just what you want to type in advance.

Introducing the ATARI Logo Editor

There is another way to define a procedure. You could use the ATARI Logo Editor. Then, if you make typing mistakes, you can remove them easily. When you are using the editor, Logo carries out only editing actions.

There is a disadvantage to using the editor. Your turtle drawings will be replaced by the editing screen and the picture will be lost.

On the other hand, the disadvantage of using TO to define your new procedure is that you will not be able to fix typing errors, except on the line where the *cursor* rests.

EDIT

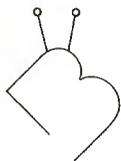EDIT or ED signals Logo that you want to edit. Follow it by the name of the procedure you want to edit. You must prefix the name with a " (quote mark). Do not type a space between " and the name of the procedure.

EDIT "SQUARE   Remember the " (quote mark).

After you press RETURN, you will be using the ATARI Logo Editor, and only editing functions will be carried out.


```
?EDIT "SQUARE
```

Bug Box

If you do not remember to prefix **SQUARE** with
a ″ (quote mark) and type

EDIT SQUARE

Logo will respond

I DON'T KNOW HOW TO SQUARE

If you had turtle drawings on the screen, they will disappear
when you start editing.

When the editor starts up, the *title line* of the procedure will
appear at the top of the screen.

TO SQUARE   This is the title line.

TO informs Logo that the following text
is part of a procedure definition.

SQUARE is the name of the procedure.
You are free to choose another name.

The *cursor* is at the beginning of the title line. Notice that Logo
does not use any *prompt* symbol while you are using the
editor.

Since you do not want to change the title line, press the CTRL ↓
key combination.

Now type in the commands that make up **SQUARE**. They are
the commands you used previously.

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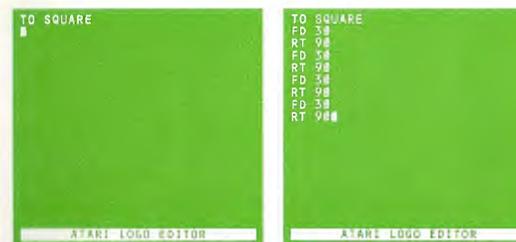FD 30
RT 90
FD 30
RT 90
FD 30
RT 90

Think of everything you type in the editor as a stream of
characters. If you want to add something to the stream, move
the *cursor* to that place.

If you want to move the *cursor* backwards in the stream, use
CTRL ←. You can move all the way to the beginning of the
stream.

If you want to move the *cursor* forward, use CTRL →.

Then type the characters you want and they will become part
of the stream. Thus, if you have made a typing error in a
previous line, use CTRL ← to move the *cursor* backwards to
where the typing error is. When the *cursor* passes over
characters, they remain unchanged.

If you want to erase a character, move the *cursor* on top of that
character and press the CTRL DELETE BACK S key combination.
The character hidden by the *cursor* will disappear.

Notice if the *cursor* is at the end of a text line, pressing CTRL DELETE BACK S will move the next line of text to the end of this line. Thus

```
FD 30█
RT 90
```

becomes

```
FD 30RT 90
```

Press RETURN to separate the lines again.

```
FD 30
RT 90
```

If you had pressed the SPACE BAR instead of RETURN, you

would have seen

```
FD 30 RT 90
```

You can move the *cursor* from the last character typed to the T in TO on the title line by pressing CTRL ← several times.
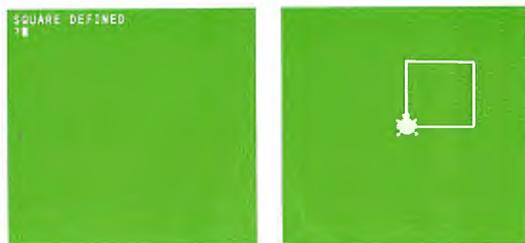
When you have finished editing, type END and press ESC. If you do not type END, Logo will insert the word END when you press ESC. Logo will now type

```
SQUARE DEFINED
```

Using the New Command

Try your new command. Type

```
SQUARE
```


SQUARE

Again, type

```
SQUARE
```

This time the turtle simply retraced its path.

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

```
RIGHT 45
```

and now type

```
SQUARE
```

Continue to repeat these two commands (RT 45 and SQUARE). To do this, you could use the Logo command, REPEAT. REPEAT requires two inputs. For example,

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

The first input indicates how many times to repeat a list of instructions. The second input is the list of instructions. The instructions must be enclosed in brackets. Think of the brackets as making an envelope. Complete the design by typing

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


RT 45   SQUARE   REPEAT 3 [RT 45 SQUARE]

Now, let's make a procedure for this design and call it **SQUARESTAR**.

`EDIT "SQUARESTAR`

You will now be using the editor; and the title line will be displayed on the screen.

`TO SQUARESTAR`

Notice that the *cursor* is at the beginning of the title line. Since you do not want to change the title line, press the **CTRL ↓** key combination and type

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

Don't forget to press **ESC** when you are finished editing.

It is always a good idea to try out your new procedure. Put the turtle in its startup state in the center of the screen facing straight up. Remember that **CS** will do this. Type **CS**
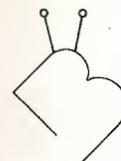
`CS`

and then

`SQUARESTAR`

If you don't want the turtle showing in the middle of your drawing, type

`HT`

HT, which stands for Hide Turtle, makes the turtle invisible. Remember to use **ST** to make the turtle visible again.

REPEAT 8 [RT 45 SQUARE]

Bug Box
If your squares look like rectangles, the bug is in your TV, and not in Logo. The Logo command **.SETSCR** allows you to change the aspect ratio of the screen. Try

`.SETSCR .8`

Then type

`CS SQUARE`

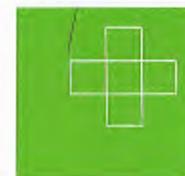If your square looks worse, try

`.SETSCR 1`

Try other settings until you are satisfied.

Other Uses of SQUARE

Once you have defined a procedure, you can use it as you would any Logo primitive such as **FD**, **BK**, **LT**, **RT**, etc. Thus, a procedure you define can be used as part of the definition of other procedures. This is one of the powerful features of Logo. For example, there are many designs that can use **SQUARE**. Some more examples are:
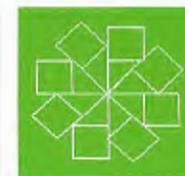
FLAG   CROSS   FLAGS   MANYFLAGS

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

FLAGBACK, on the other hand, leaves the turtle in the same position on the screen as it started. We can see the effect of these differences in **CROSS** and **FLAGS**. **CROSS** runs **FLAG** four times while **FLAGS** runs **FLAGBACK** four times.

```
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

TO MANYFLAGS
FLAGS
RT 45
FLAGS
END
```
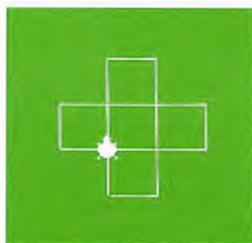
If you turn the computer off now, all the procedures you have written will be erased. See the next chapter for saving your procedures if you don't want to type them in again.

FLAG   CROSS   FLAGBACK

ATARI Logo Vocabulary

The following Logo primitives were introduced in this chapter.

EDIT (ED)
HT
.SETSCR

We have also used several special keys.

BREAK
CTRL ←
CTRL →
CTRL ↑
CTRL ↓
CTRL DELETE BACK S
ESC

FLAGS   MANYFLAGS

Saving
and Retrieving
Your Work

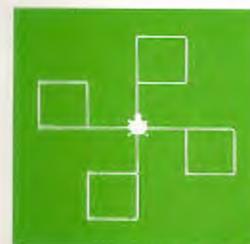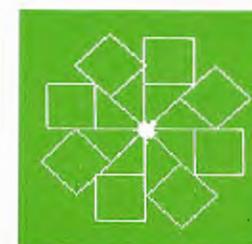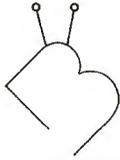When you define procedures, Logo puts them in your *workspace*. A workspace is a space in the computer memory that lasts only for the time the computer is turned on. You can save your procedures permanently on a diskette or cassette using the SAVE command. You can later retrieve your work using the LOAD command.

Bug Box

To save your work, you need an Atari program recorder or disk drive. If you do not have either, skip this chapter and go on.

Saving and Retrieving Your Work on Cassette

When you are saving your work on cassette, everything in your workspace is saved in one cassette file. To create a file of your workspace, first insert a blank cassette into the program recorder, rewind the tape to the beginning of the magnetic part (*not* the leader part), and set the counter on your program recorder to 000. Next, type

SAVE "C:

and press RETURN on the computer keyboard.

The C: indicates to the computer that you are saving your procedures on a cassette.

When you type SAVE "C:, you hear the computer beep twice. Then, press the PLAY and RECORD buttons on the program recorder simultaneously, and then press RETURN on the computer keyboard. Everything presently in your workspace will be saved. When the saving is complete, the *prompt* and *cursor* reappear on the screen, and you can turn the computer off.

The same side of your blank cassette can be used to save several different files. In this case, write down the beginning and end counter numbers each time you save. It's a good idea to also write down a name or description of each file. Always advance the tape by approximately 10 counts before saving the contents of another workspace.

To retrieve your work from the cassette, set the counter of your recorder to the beginning number of the file that you want to retrieve. Then type

LOAD "C:

and press RETURN. When you hear the computer beep, press the PLAY button on your program recorder and then hit RETURN on your keyboard. Everything that you saved in the file will now be loaded back in your workspace. The *prompt* and *cursor* reappear on the computer screen when the loading is complete.

Saving and Retrieving Your Work on Diskette

In order to save your work, you need to use a formatted diskette. If one has not yet been prepared, you will need to format a blank diskette with the ATARI Disk Operating System. Instructions on how to complete this procedure can be found in an *ATARI Disk Operating System Reference Manual*.

When you are saving your work on diskette, everything in your workspace (each of your procedures) is saved in one disk file. Each file on your diskette must be given a different name. A filename can be 1 to 8 characters long with an optional 3 character extension. The first character of the filename must be a letter. All letters in the filename and extension must be upper case. If an extension is used, a period must be used to separate the filename from the extension. For example, to create a file named DESIGN with an extension .001, type

`SAVE "D:DESIGN.001`

The input "D:DESIGN.001 indicates that you are saving onto a diskette and the file is named DESIGN.001. A file name must be one word and preceded by a " (quote mark) and device code (D: in this case).

```
?SAVE "D:DESIGN.001
```

When you give this command, the disk drive makes noise and the red light marked BUSY lights up. Wait until the light is off and the cursor reappears. Now that you have saved your work, you can turn the computer off.

To load your work from a diskette, use the command LOAD and the file name as its input the same way you did with SAVE.

`LOAD "D:DESIGN.001`

Everything that you saved in the file DESIGN.001 is now back in your workspace.

Bug Box
If you have turned on your ATARI Home Computer before you turned on the disk drive, Logo will respond with

`I CAN'T OPEN D:DESIGN.001`

Refer to Chapter 1 — Getting Started.

Saving Your Work on a Printer

In order to get a hard copy (paper copy) of your workspace, you must have a printer connected to your ATARI Home Computer. If you do, typing

`SAVE "P:`

prints out all the procedures in your workspace.

```
?LOAD "D:DESIGN.001
```

Listing the File Names

You can see the names of files already on the diskette by using the command CATALOG with "D: as the input.

```
CATALOG "D:
```

Notice that your file, DESIGN.001, has been added to the list of files.

CATALOG used with a cassette will give you a listing of the procedures.

Erasing Disk Files

Files on a diskette can be erased by the command ERF. Its input is the file name preceded by "D: (as with SAVE and LOAD). Typing

```
ERF "D:DESIGN.001
```

will erase the file named DESIGN.001 from the diskette in the disk drive.

You can't erase cassette files with ERF. To erase a cassette file, you must save another workspace in the same place as the file you want to erase.

ATARI Logo Vocabulary

The following Logo primitives were introduced in this chapter.

CATALOG
ERF
LOAD
SAVE

The Turtle and the Text on the Screen

Before you give Logo any turtle commands, the whole screen is available for text. As soon as you give a turtle command, the screen is divided into a large turtle field and a small text field. Only five lines at the bottom are available for text.

The command TS (which stands for Text Screen) will get you the whole screen for text. SS (for Split Screen) will get you back to the turtle field and the five-line text field. Neither of these commands destroys what was on the two fields. They only change what is *visible* to you. Try going back and forth. There are special action keys that have the same effect as these commands. For example, CTRL T is for TS and CTRL S is for SS.

FS (which stands for Full Screen) gives the whole screen to the turtle. No text is visible. So if you type FS, you will not see the characters typed on the screen, but they will be there all the same. CTRL F has the same effect as FS.

CTRL T, CTRL S, and CTRL F can be typed while a procedure is running.

Try this.

CTRL F   CTRL S

Next we go back and forth between TS and SS.

CTRL T   CTRL S

Note: When you edit a procedure, the turtle screen is erased; it becomes the edit screen.

Chapter **6**

The Turtle
and the Text
on the Screen

When you first start up ATARI Logo or when you leave the ATARI Logo Editor, neither CTRL S nor CTRL F will switch to the turtle screen until you give a turtle command.

ATARI Logo Vocabulary
The following Logo primitives were introduced in this chapter.

FS
SS
TS

We have also used several special keys.
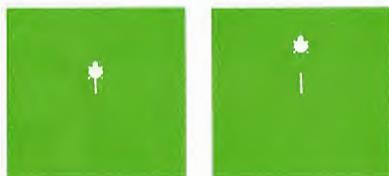
CTRL F
CTRL S
CTRL T

The Turtle's Pen and Color

The turtle leaves a trace 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 trace, you can tell the turtle to lift its pen. You can change the color of the trace by changing the color of the pen. This chapter explains how to use the pen and the ATARI color graphics.

Pen Commands

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 effects produced by these two commands.


```
FD 15
PENUP
FD 15
```



```
PENDOWN
FD 20
```

When you tell the turtle to put its pen up (PU) or down (PD), you are changing the *state* of the pen. There are two other commands that change the state of the turtle's pen. These are PE (for Pen Erase) and PX (for Pen Reverse).

The command PE turns the turtle's pen into an eraser instead of a drawing instrument. If you now make it retrace a line it has drawn, the line will be erased.

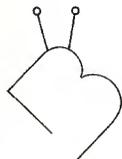For example, clear the screen, make sure the pen is down, and draw a square.

```
CS
PENDOWN
SQUARE
```

Now

```
PE
SQUARE
```

The turtle will erase any lines it passes over until you tell it to PENUP or to PENDOWN. Notice the turtle does not draw any new lines.

Bug Box

If Logo responds

`I DON'T KNOW HOW TO SQUARE`

when you type SQUARE, look SQUARE up in the Index to find out where SQUARE is defined. This same situation may happen again if you type the name of a procedure that is not defined in your workspace. If the procedure has been defined in this book, you can look up its name in the Index.

PX is a mixture of PENDOWN and PE. When this command has been given, the turtle will draw a line whenever it moves over blank background. But if it moves over a previously drawn line, it will erase it. This can be used to produce some spectacular effects. For example, type
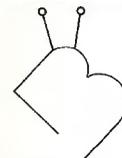
```
PX
SQUARE
SQUARE
SQUARE
```

PENDOWN (or PD) returns the pen to its normal drawing state. So, type

```
PENDOWN
```

Using ATARI Color Graphics

This section describes features which take advantage of ATARI color. The ATARI Home Computer allows you to use 128 colors. These 128 colors are actually composed of 16 colors each having 8 shades. The colors are number-coded as follows:

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

Bug Box

Colors may vary depending upon the type of TV, monitor, condition, and color adjustments.

For each color, the lowest number is the darkest shade of that color, and the highest number is the lightest shade of the color. For example, 0 is black and 7 is white.

There are three types of color changes you can make. You can change the color of the turtle field or BG (BackGround) and the color of the turtle's pen. The color of the turtle itself can also be changed (see Chapter 16).

Changing the Background Color

When ATARI Logo starts, the background color is 74 (blue). To change the background color use the SETBG command with the color number as input.

Try

```
SETBG 1
SETBG 40
SETBG 120
```

You can define a procedure that cycles through several 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. CB is an example of such a procedure.

```
TO CB
SETBG 0 WAIT 20
SETBG 35 WAIT 20
SETBG 48 WAIT 20
SETBG 60 WAIT 20
SETBG 98 WAIT 20
SETBG 126 WAIT 20
END
```

Repeat CB a few times. Type

```
REPEAT 3 [CB]
```

You can always find out the number code of the current background color by printing BG. Type

```
PRINT BG
```

Logo responds

```
126
```

Now, set the background color to black.

```
SETBG 0
PRINT BG
0
```

Changing the Pen Color

The turtle can use three different pens to draw. The pens are numbered 0, 1, and 2. When Logo starts, the turtle is using pen 0 and its color is gold (color number 15). If you haven't changed the pen, the following commands should draw a square in gold.

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

You can find out which pen the turtle is using by printing PN (Pen Number). Type

```
PRINT PN
0
```

The pen number can be changed by the command SETPN. Using three different pens, you can have turtle drawings in three different colors. Pen number 1 draws in purple (color number 47) and pen number 2 draws in red (color number 121). The following commands will display a purple square and a red square by using pens 1 and 2.

```
SETPN 1
REPEAT 4 [FD 30 RT 90]
LT 180
SETPN 2
REPEAT 4 [FD 30 RT 90]
```

Note that if you change the background color, the pen colors may not correspond exactly to the numbers in the color table.

If you want to find out the color numbers, you can ask Logo to print the Pen Color (PC) of each pen. Type

```
PR PC 0
15
PR PC 1
47
PR PC 2
121
```

ATARI Logo allows you to change the color of the turtle's drawing already on the screen. This is done by the command SETPC (SET Pen Color). SETPC tells Logo to change the color of a particular pen (and drawings made with it). Thus typing

SETPC 0 40

will change the color of the first square (previously drawn in gold) to magenta. Notice that the other squares remain their original colors. That is because they were drawn with other pens. Now pen number 0 will draw in magenta (40). To try this out, set the pen to 0.

SETPN 0
FD 50

You should see a magenta-colored line on the screen.

ATARI Logo Vocabulary
The following Logo primitives were introduced in this chapter.

BG
PC
PE
PENDOWN (PD)
PENUP (PU)
PN
PX
SETBG
SETPC
SETPN
WAIT

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

First, define a procedure to draw a diamond, but with a bug in it. For example, type

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

When we try this procedure, we find that 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 we edit the procedure. Type

```
EDIT "DIAMOND
```

The text of the procedure, DIAMOND, is now displayed on the screen.

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

The *cursor* is positioned at the top left corner of the screen on the letter T of the word TO.

To edit, move the *cursor* to where you want to add or delete characters. Move the *cursor* forward by pressing the CTRL→ key combination. Move it back by pressing the CTRL← key combination.

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

```
TO DIAMOND█
```

Now press RETURN and type

```
RT 45
```

Move the *cursor* down to the line before END by pressing the CTRL ↓ key combination twice.

```
RT 45█
```

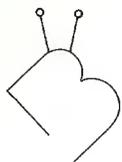Press DELETE BACK S (five or six times) to erase the characters on the line.

Leaving the Editor

Press ESC. This signals Logo that you have finished editing. Logo prints out a message saying

```
DIAMOND DEFINED
```


DIAMOND with bug

Bug Box

If you are editing and don't like the changes you are making or decide not to make changes and want to start again, press BREAK. Logo will leave the editor and forget the changes that you made so far. The definition of the procedure will be the same as before you started editing.

Here are some useful editing actions. The *Reference Manual* describes others.

CTRL ↓ (down arrow)	moves the *cursor* down to the next line.
CTRL ↑ (up arrow)	moves the *cursor* up to the previous line.
CTRL ← (left arrow)	moves the *cursor* one space to the left.
CTRL → (right arrow)	moves the *cursor* one space to the right.
RETURN	moves the *cursor* and the following text to the beginning of the next line.
*CTRL A	moves the *cursor* to the beginning of the current line.
*CTRL E	moves the *cursor* to the end of the current line.
DELETE BACK S	erases the character to the left of the *cursor*.
CTRL DELETE BACK S	erases the character directly under the *cursor*.
*SHIFT DELETE BACK S	erases the rest of the line (to the right of the *cursor*).

*This editing action was not discussed in this section, but it is useful and will be used later.

The *cursor* will move only where text is on the screen. If you try to make the *cursor* go where there is no text ATARI Logo will beep.

Editing Outside of the Editor

You can use most of the editing functions to edit instructions you type to Logo when you are not in the editor. For example, type

DIAMOND

Now press the CTRL Y key combination. This gets you a copy of the last line you typed. Each line of text on the screen is like a mini-editor, only one line high. Logo responds

DIAMOND

The *cursor* is at the end of the line.

You can now press the CTRL A key combination to move the *cursor* to the beginning of the line.
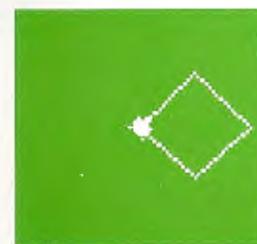
DIAMOND

Now type
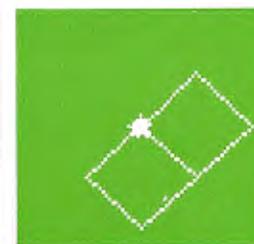
RT 45   Type a space.

Now, press RETURN. Press CTRL Y again and then press RETURN.
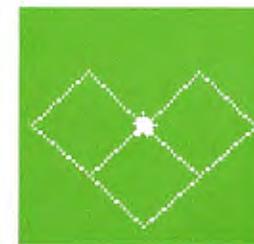
Experiment with other editing actions both in the editor and outside of it. For more details, consult the *ATARI Logo Reference Manual*.

DIAMOND   RT 45 DIAMOND   RT 45 DIAMOND

Your Workspace

As you interact with Logo and give words meaning, Logo puts these new words in what we refer to as your *workspace.*

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 and you can print out their definitions.

Printing Out Procedures

POTS (Print Out TitleS) prints out the title lines of each of the procedures in the workspace. Type

POTS

Logo responds

```
TO DIAMOND
TO CB
.
.
.
```

and so on.

POPS (Print Out ProcedureS) prints out the definitions of all the procedures in the workspace. Type

POPS

Logo responds

```
TO DIAMOND
RT 45
SQUARE
END

TO CB
SETBG 0 WAIT 20
SETBG 35 WAIT 20
SETBG 48 WAIT 20
SETBG 60 WAIT 20
SETBG 98 WAIT 20
SETBG 126 WAIT 20
END
.
.
.
.
```

and so on.

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

PO "SQUARESTAR

Logo responds

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

PO can also be given a list of procedure names. For example, PO [SQUARE SQUARESTAR DIAMOND] would print out the definitions of the three procedures whose names are in the input list. Remember that you can use TS or CTRL T to get a full screen of text.

Erasing From the Workspace

You can erase procedures from your workspace. If you have not saved these procedures on a cassette or diskette, you will have to type them in again. So, be sure you really want them erased.

There are several Logo commands to erase your work. The most commonly used is ERASE, whose short form is ER.

```
ERASE "DIAMOND
```

erases the procedure DIAMOND.

```
ERASE [SQUARE SQUARESTAR]
```

erases all the procedures named in the input list.

```
ERPS
```

erases all procedures from the workspace.

ATARI Logo Vocabulary

The following Logo primitives were introduced in this chapter.

POTS
POPS
PO
ERASE (ER)
ERPS

A Graphics Project Drawing a Spider

Let's make a spider with 4 legs on each side. A first step is to look more closely at a right leg and a left leg. Each leg is made by 2 lines joined to form a 90 degree angle.
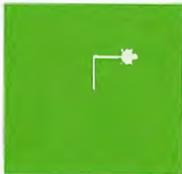
As a first step, let's make a RIGHTLEG.

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

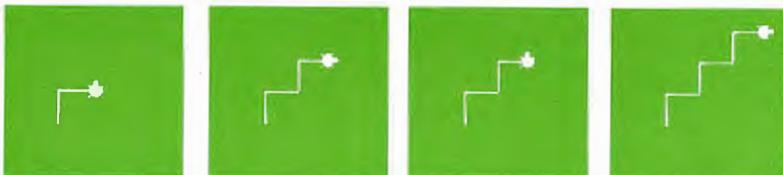You can choose any number as input to FORWARD. We use 30 here.

Now try it.

```
RIGHTLEG
```

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

At this point RIGHTLEG could be used to make stairs.

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

... but we want spider legs.

When in doubt about where you think the turtle should be when the procedure stops, put the turtle where it was before the procedure was run. Now fix or *debug* RIGHTLEG using the editor.

EDIT "RIGHTLEG

Now the ATARI Logo Editor shows this procedure with the *cursor* on the T of TO.
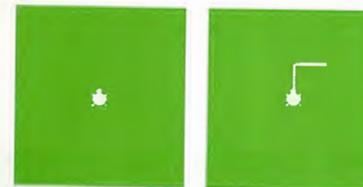
```
TO RIGHTLEG
FD 30
RT 90
FD 30
END
```

Position the *cursor*, then type in the new commands.

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

These last 3 commands return the turtle to where it was at the start of RIGHTLEG.
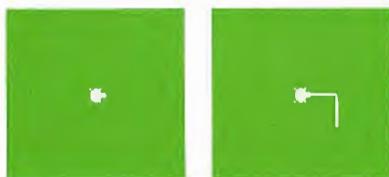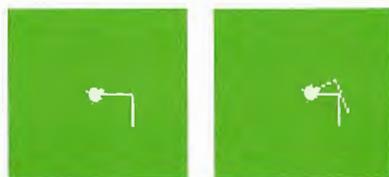
Try RIGHTLEG.

```
CS
RIGHTLEG
```

Now plan RIGHTSIDE, the procedure that will draw all the legs on the spider's right side. We want one leg horizontal, so ...

```
CS
```



```
RT 90
RIGHTLEG
```

Now for the second leg ...


```
LT 20
RIGHTLEG
```

Continue in this way until the turtle has drawn 4 legs. You can now make a procedure for RIGHTSIDE.

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


Notice the last command in RIGHTSIDE, 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 force procedures to adopt the

rule of good behavior: "Leave the turtle in the state you found it."

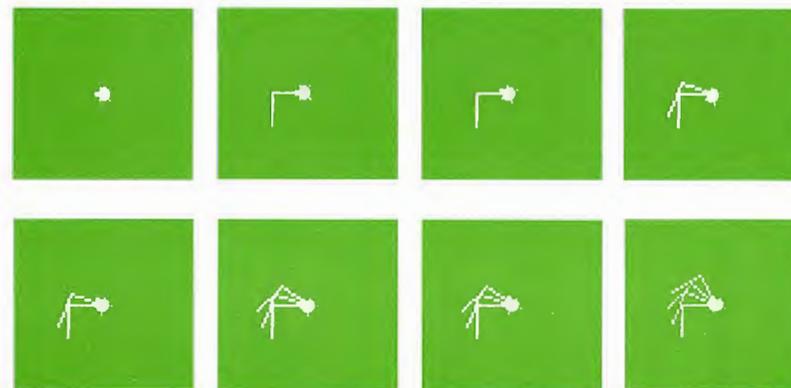Now work on a left leg. LEFTLEG will be similar to RIGHTLEG.

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

Try it out.

Use LEFTLEG to write LEFTSIDE.

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


and finally ...

```
TO SPIDER
LEFTSIDE
RIGHTSIDE
FD 10 BK 10
HT              Hide Turtle now that the job is done.
END
```


Other Designs Using RIGHTLEG and LEFTLEG:

MAN   SWIRL   SPINSTAR

Some Turtle Geometry: Triangles

The turtle can draw different triangle shapes. The triangle we discuss below is like a square in that all its sides are equal and all its angles are equal. In this example, the turtle will take 30 steps forward, the same amount it took in SQUARE.
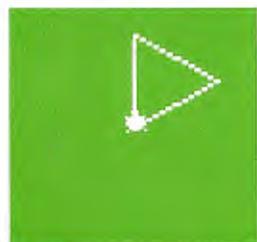


```
FD 30
```

Now comes the big decision. How many degrees does the turtle have to turn to draw this triangle? In school we learned that equilateral triangles have 60 degree angles. Look what happens when the turtle turns 60 degrees.
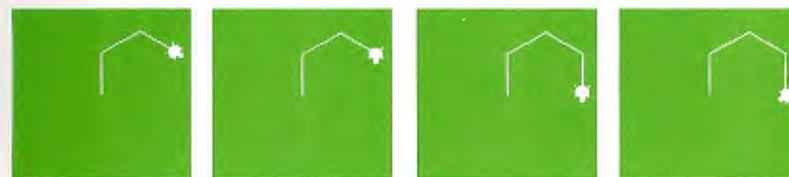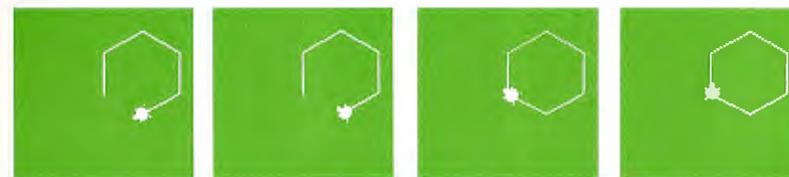


```
RT 60
FD 30
RT 60
```


Equilateral Triangle

Interesting, but not a triangle! But we might as well finish it.
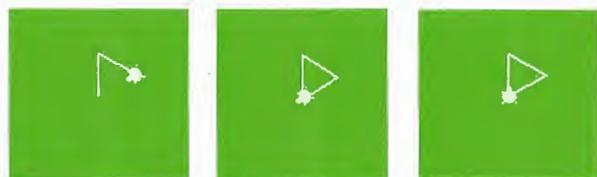


```
FD 30
RT 60
FD 30
RT 60
```



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

The figure is a hexagon; it has 6 rather than 3 sides. To make a triangle, the turtle needs to turn 120 degrees at each corner. Why 120 and not 60? The answer is simple. You have to look at the problem from the turtle's viewpoint. When the turtle starts its triangle trip, it must turn 360 degrees (a complete circle) before it returns to its starting state. It walks along an edge of the triangle and pivots about (turns around at) the external, not the internal, angles of the triangle. The turtle does this three times. (3 times 120 is 360; 6 times 60 is 360.) When the turtle draws a square, it turns four times instead of three; 4 times 90 is 360!

Now we can finish drawing the triangle.
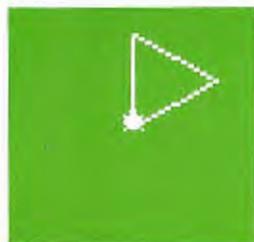
```
CS
FD 30
RT 120
FD 30
```



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

Now you can define the word TRIANGLE. Let's use the editor to do this.

```
EDIT "TRIANGLE
```

and now insert the lines

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


TRIANGLE
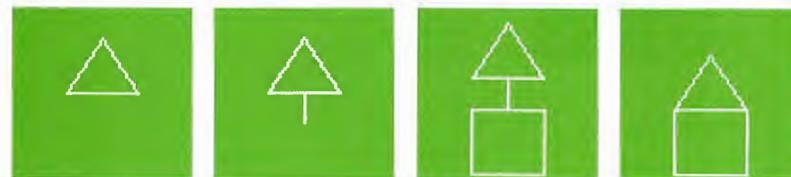
Play with this procedure a bit. For example

```
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 **BREAK**.

You might want to figure out how many times the turtle needs to repeat a set of commands. 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 made by using TRIANGLE.
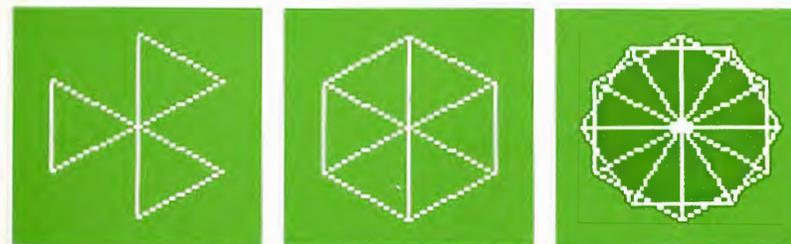

TENT   TREE   WELL   HOUSE

Let's make the turtle draw TENT. Running TRIANGLE alone is not enough. The TENT will be tipped.

REPEAT 3 [TRIANGLE RT 120] REPEAT 6 [TRIANGLE RT 60] REPEAT 360/30 [TRIANGLE RT 30]

```
CS
TRIANGLE
```

Turn the turtle **RT 90** and run **TRIANGLE**. This time the tent is upside down.


```
CS
RT 90
TRIANGLE
```

Remember when we made **TRIANGLE** the inside angle was 60 degrees. If we now turn the turtle **RT 90** and then **LT 60**, the turtle is set up for drawing a tent. Of course, we could just turn the turtle **RT 30**.
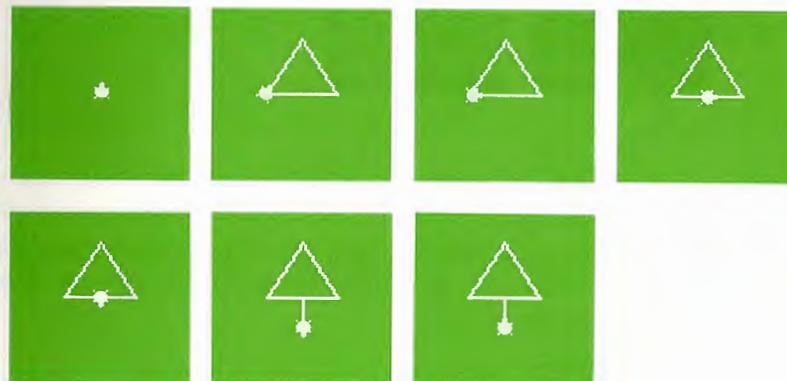


```
CS
RT 30
TRIANGLE
```

The procedure, then, is:

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

74

TENT into TREE

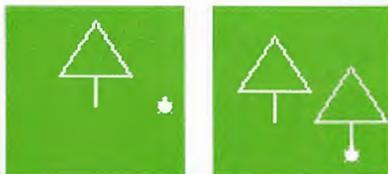Now, you can use **TENT** to help make **TREE**.


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

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


```
CS
TREE
RT 90
PU
FD 30
```

75

```
LT 90
PD
TREE
```

It is a good programming habit to put the 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
```

We can then use TREE and SETTREE repeatedly.

```
REPEAT 3 [TREE SETTREE]
```

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

REPEAT 3 [TREE SETTREE]

Turtle Makes a House

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

```
CS
SQUARE
FD 30
TRIANGLE
```

You put them together but the turtle didn't draw a house. There is a bug, but it is easy to fix. We should use TENT instead of TRIANGLE.

```
CS
SQUARE
FD 30
TENT
```


HOUSE

Variables:
Big Squares and
Small Squares

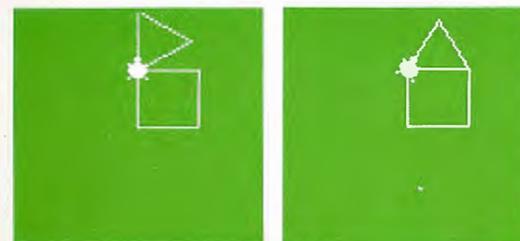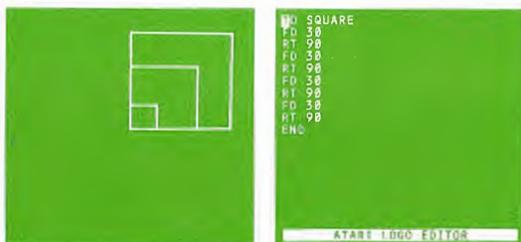You might want the turtle to draw squares with sides of 60, 50, 100, 10 and so on. One way of doing this is to have many procedures: SQUARE100, SQUARE50, SQUARE33, etc. But there is a shortcut. We can change SQUARE so that it takes an input. Then we can tell SQUARE how long to make its sides by typing

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

So let's make a procedure for drawing variable sized squares. BOX might be a suitable name since it reminds us of squares. But, we'll meet a new idea by calling it BOXR (pronounced box-are) which makes us think of a right-turning square. We could also define a left-turning box and call it BOXL (pronounced box-ell).

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

```
EDIT "SQUARE
```

Now the Logo editor shows this procedure with the *cursor* on the T of TO.

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

First, let's change the name of the procedure from SQUARE to BOXR. The *cursor* is sitting on the T of TO. Move it to the space after SQUARE. Use CTRL E.

```
TO SQUARE
```

Now, erase the word SQUARE using the DELETE BACK S key and type the word BOXR.

```
TO BOXR
```

Press ESC. Logo responds

```
BOXR DEFINED
```

Remember, your original definition SQUARE still exists. At this point BOXR and SQUARE have the same definition. Now we change BOXR. Type
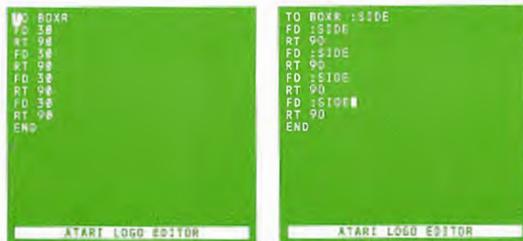
```
EDIT "BOXR
```

The ATARI Logo Editor now shows

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

We change BOXR so that it requires an input like FORWARD does. The procedure will be able to draw squares of various sizes. How do we tell Logo to do this?

The first instruction has to be FORWARD some amount, but when we are writing the procedure we don't know what amount it will be. We handle this situation by giving the amount a name. For example, let's call it SIDE. If we choose this name, we write FD :SIDE to mean FORWARD whatever number happens to be called SIDE. So we can write

```
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
```

One more idea is needed to turn this into a procedure. When we use the command BOXR, we will now have to follow it with an input like this:

```
BOXR 20
BOXR 100
```

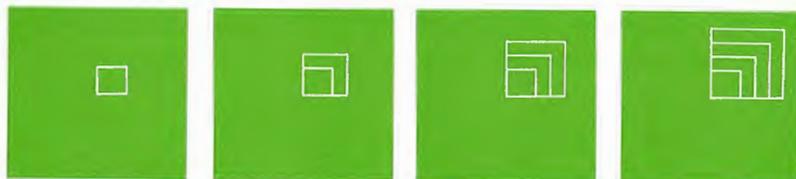To indicate that BOXR needs an input and that this input will be called SIDE, we do the whole procedure like this:

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

Press ESC. Logo responds

```
BOXR DEFINED
```

BOXR makes the turtle draw a square of any size depending upon the number you give it as an input.
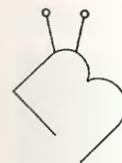

BOXR 20   BOXR 100

```
BOXR 10
BOXR 20
BOXR 30
BOXR 40
```

We have just used a powerful mathematical idea — the idea of a variable. But instead of using a mysterious X for the variable as we did in school algebra, we have used a meaningful name, SIDE, which reminds us of what the variable does.

When you were defining this procedure you wanted to tell the turtle how to draw a square, but you did not know what size square you might need. Indeed, you wanted to be able to draw squares of all possible sizes. When you came to typing the FORWARD command, you knew that FORWARD needed an input. You couldn't just type FORWARD without an input. You had to type FORWARD something. To give this something a name, we called it SIDE. In Logo 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 use BOXR and type BOXR 10 or BOXR 15. When Logo obeys that command, 10, 15, or whatever you typed as the input is put in the container named SIDE. BOXR can then look in the container at a later time.

Bug Box
Possible bugs:

1. You typed :SISE or some other spelling different from the way the input on the title line was spelled.
2. You forgot to use : (colon).
3. You inserted an extra instruction in BOXR.
4. You accidentally erased an instruction in BOXR.
5. You typed a space between : (colon) and SIDE, or a : (colon) in front of a number.
6. You forgot to use a space before the :(colon).

The character, : (colon), informs Logo that the word to which it is prefixed names a container that can have in it a number, another word, a list of words, or a list of lists.

Some Uses of BOXR

| SQUARES | DIAMONDS | 6FLAG 30 | SPINFLAG 30 |

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

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
```

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

Big Triangles and Small Triangles

We can also define a triangle procedure which takes an input. Type

```
ED "TRIANGLE
```

Now, the Logo editor shows this procedure with the *cursor* on the T of TO.

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

Then change TRIANGLE:

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

Use the procedure to make designs like:

TRIANGLES   TRISTAR   TREES

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

TO TRISTAR
REPEAT 10 [TRIANGLES RT-36]
END

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

TO TREES
TREE 30
TREE 40
TREE 50
END
```


TRIANGLES   TRISTAR   TREES

Arithmetic

As you have learned from the examples above, you can do arithmetic in Logo. For example, if you type

```
PR 5 + 3
```

Logo types

8

```
PR 4 * 23
```

Logo types

92

```
PR 345 - 32
```

Logo types

313

```
PR 25 / 5
```

Logo types

5

Logo Numbers

Logo has both decimal and integer numbers and will perform arithmetic with decimals or integers. We have seen some examples of arithmetic with integers. Here are other examples:

```
PR 25 / 5
5

PR 25 / 6
4.16666666

PR 4 * 2.3
9.2

PR 25/2
12.5
```

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

ATARI Logo Vocabulary
The following Logo primitives were introduced in this chapter.

/ (divide)
— (subtract)
+ (add)
* (multiply)
: (colon)

Circles
and Arcs

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

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

```
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, we can make a little compromise.
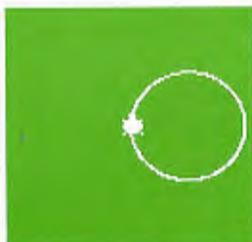
```
REPEAT 36 [FD 10 RT 10]
```

This circle is less perfect; in fact, it is a 36-sided polygon. But the turtle draws it 10 times faster than the first circle.

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

You can try some experiments to answer these questions. Let's 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.

REPEAT 360 [FD 1 RT 1]

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

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

Circle with Radius

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 circle). With the CIRCLE procedure, we need to calculate the radius of each circle. Why not let the *computer* calculate it? To do this, write another procedure that uses CIRCLE. Let's call it CIRCRAD.

```
TO CIRCRAD :RADIUS
CIRCLE 2 * 3.14 * :RADIUS / 36
END
```
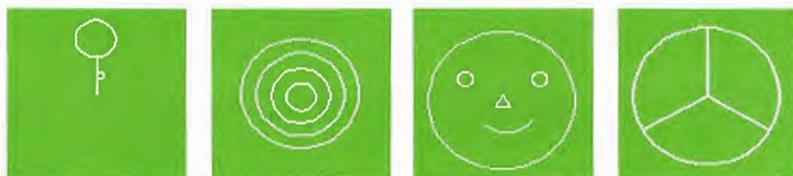
Note: 2 * 3.14 * :RADIUS represents the circumference of a circle ($2\pi r$). The circumference has 36 FORWARDs. So divide it by 36 to get the step size.

Now try


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

Here are some projects using circles. See if you can make them!

FLOWER   TARGET   FACE   PEACE

Turtle Draws Arcs

Many projects require only pieces of circles. There is a simple way to get a piece of a circle right now. Run the CIRCLE procedure and quickly press BREAK to stop the turtle before it finishes drawing.

This method has a little problem. You need very good coordination to produce the right size arcs. The best way to control this situation is to give another input to the CIRCLE procedure. The input allows us to vary the number of times the small steps and turns are repeated.

Let's change the procedure name to ARC at the same time.

```
EDIT "CIRCLE
```

The ATARI 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
```


ARC 10 36   ARC 10 18   ARC 10 9

We can use the number of degrees that you want in your arc as the input and let Logo calculate how many times to repeat.

```
EDIT "ARC
```

Change the input to REPEAT so the whole procedure looks like this.

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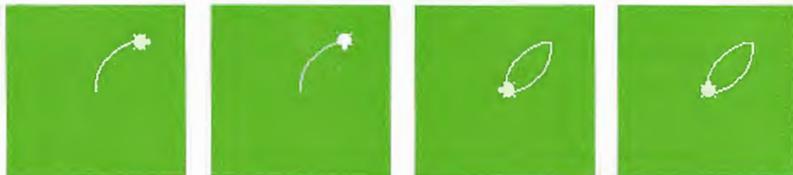
Using Arcs


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

A Fish!


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

A Petal! Let's write a procedure to draw petals.

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

Try

```
PETAL 40 PETAL 30
REPEAT 8 [PETAL 40 PETAL 30 RT 45]
REPEAT 4 [PETAL 30 RT 45 PETAL 40 RT→
45]
```

Notice that Logo lines can extend beyond one screen line. Logo marks continuation lines by putting a → (arrow) 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 out of one of these designs.

Now let's do a SWAN. You will need to use the CIRCLER, CIRCLEL, ARCR, and ARCL procedures listed in Appendix A.

SWAN

There are really only 2 different shapes, PETAL and NECK. We can use PETAL to make BODY and HEAD. So

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

```
TO NECK :SIZE
LT 45
ARCR :SIZE 90
ARCL :SIZE 90
RT 45
END
```
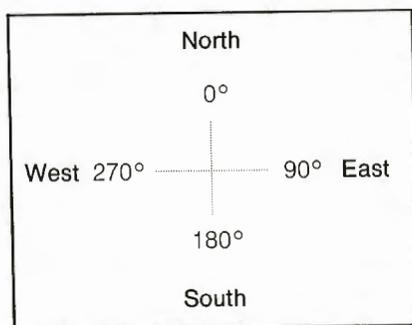
```
TO HEAD :SIZE
LT 135
PETAL :SIZE
RT 135
END
```

SWAN, the controlling procedure, is

```
TO SWAN :SIZE
BODY :SIZE
NECK :SIZE / 2
HEAD :SIZE / 4
END
```

**The Turtle's
Field**

The turtle has a *position* and a *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. We could mark the screen:

```
          North

           0°

West 270° ----+---- 90°  East

          180°

          South
```

When the turtle starts up, its heading is 0. After CS, the turtle has a heading of 0. At any time you can get a compass reading. Try

```
CS
RT 90
PR HEADING
```

Logo responds

90

HEADING outputs the turtle's direction.

HEADING is part of Logo's vocabulary. It is different from PRINT or FORWARD. It is not a command; it is an *operation*. It does not

cause something to happen, but rather outputs something that can be used as an input. In this section, several other operations are introduced.

SETH, which stands for SET Heading, 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
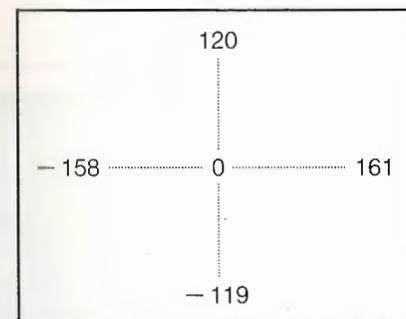
```
SETH 90
RT 90
SETH 90
```

The turtle's position is described by two numbers which indicate how far the turtle is from the center. For example, POS, which stands for POSition, is [0 0] at the start. The first number indicates the turtle's location along the horizontal or x-axis. If the turtle is west of the center, then the number will be negative.

The second number indicates the turtle's location along the vertical or y-axis. If the turtle is south of the center, then the number will be prefixed by a - (minus sign).

The turtle screen can be represented by a grid divided into coordinates. The x coordinates run along the horizontal, and the y coordinates run along the vertical. The turtle at the center has both XCOR and YCOR equal to 0. XCOR and YCOR are Logo operations.

The initial screen dimensions are approximately:

```
              120

 -158 --------- 0 --------- 161

             - 119
```

For example, if you type

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

Logo responds

```
-30 0
```

The turtle is 30 steps west (left) of the center along the horizontal.

If you type

```
BK 60
PR POS
```

Logo responds

```
30 0
```

Now the turtle is 30 steps east (right) of the center along the horizontal.

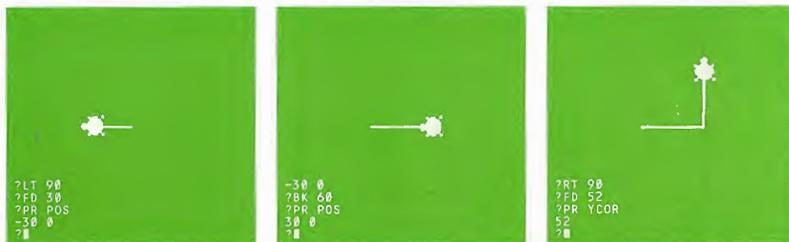You could also find either coordinate by itself.

```
PR XCOR
```

Logo responds

```
30
```

Then, continue with

```
RT 90
FD 52
PR YCOR
```

Logo responds

```
52
```

The turtle is 30 steps east of the center and 52 steps north of the center.

Bug Box
Remember **CS** brings the turtle back to [0 0]. Therefore, if you type **CS** you will not get these results.

SETPOS, which stands for SET POSition, is a command that **SET**s the turtle at a specific **POS**ition on the screen. **SETPOS** acts differently from **FORWARD** or **BACK** in that the end result does not depend on the turtle's initial position. Also this command 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.

WRAP and WINDOW

The turtle starts out being able to **WRAP**, which means it can walk off one edge of the screen and on at the opposite edge. It does not change direction.

Type

```
CS
FD 500
PR POS
```

Logo responds

```
0 20
```

Notice that the turtle is 20 steps and not 500 steps from the center.

The WINDOW command allows the turtle to move off the screen without wrapping. Thus the turtle might often be invisible to you, but still be carrying out your orders. The x and y coordinates can be very large.
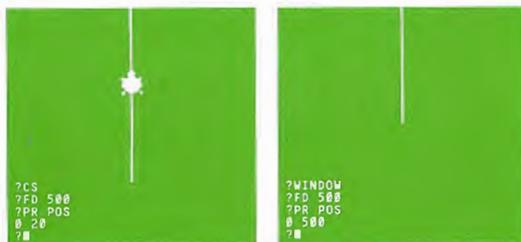
```
WINDOW
FD 500
PR POS
```

Logo responds

```
0 500
```

The turtle is now 500 steps from the center and out of view. A CS always restores the turtle to its center position on the screen.

If you want to get back in the WRAP mode, type WRAP. Typing WRAP or WINDOW clears the screen and brings the turtle to its center position on the screen.

Using POS to Draw

There is an easy way to draw a right triangle provided you know the lengths of the two sides joining in a right angle. We first record the starting position of the turtle. We do this by using the Logo command MAKE.

```
CS
MAKE "START POS
```

MAKE does two things. It puts the output from POS in your workspace and makes START its name. Thus if you say

```
PR :START
```

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

```
0 0
```

Now, have the turtle draw the two sides with the right angle in between.

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

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

```
SETPOS :START
```

The turtle is moved to :START, and since the pen is down, a line is drawn. 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
```

ATARI Logo Vocabulary

The following Logo primitives were introduced in this chapter.
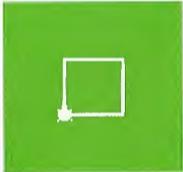
Commands

SETH
SETPOS
WINDOW
WRAP

Operations

HEADING
POS
XCOR
YCOR

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

Exploring Polygons and Spirals

Just as you can vary the number of steps the turtle takes, you can also vary how much it turns. In fact, you can get beautiful and surprising designs by varying these two components of the turtle's state. The following procedure takes two inputs: one which specifies the number of turtle steps and one which specifies the amount to turn.

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

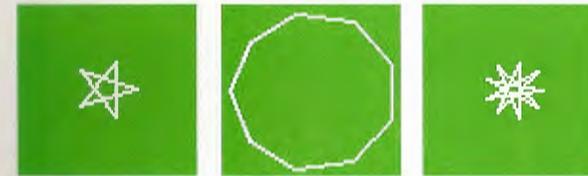Now try it!


```
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 BREAK. Logo responds

```
STOPPED! IN POLY
```

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 40
POLY 30 160
```

POLY is *recursive*. This means that, instead of calling another procedure to help out, POLY calls itself. In Logo, 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.

Spiral Procedures

The POLY procedure makes the turtle draw closed figures. 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.

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 further and further away from that point.

We can do this by adding a little bit to :STEP on the recursion line of the POLY procedure. Modifying the procedure POLY, we 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
SPI 5 120
SPI 5 60
```

Remember the **BREAK** key stops the procedure. **CTRL F** shows the whole turtle field.
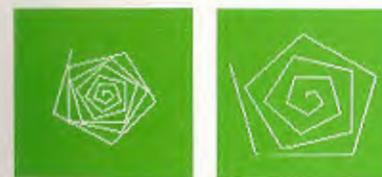


```
SPI 5 144
SPI 5 125
SPI 5 160
```

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

```
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.

**Extended
Turtle Graphics
for ATARI Logo**

ATARI Logo has extended turtle graphics capabilities. There are four turtles. The turtle you have been talking to so far is one of the four turtles. In addition to producing turtle drawings, all four turtles can perform dynamic actions.

This chapter describes how to talk to these turtles together or separately, and shows you how to use some of the special actions they can perform.

Multiple Turtles

The four turtles are numbered 0 through 3. You can tell Logo which turtle you want to talk to by using the TELL command. For example, to line up four turtles on the screen, begin with the following instructions.


```
ST
FD 40
```

When you start Logo, it's always turtle 0 that carries out your commands until you specify another one.


```
TELL 1
```

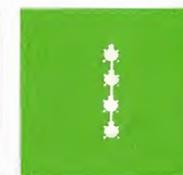Turtle 1 appears at the center of the screen.

Bug Box

Turtle 1 may not appear on the screen. This is because TELL makes a turtle appear only the first time it is addressed. If the turtle doesn't appear after TELL, it is hidden. Use

```
ST
```

Now the turtle appears.


```
FD 20
TELL 2
BK 20
TELL 3
```

In order to talk to more than one turtle at a time, use the [ ] (square brackets) around the turtle numbers.


```
TELL [0 1 2 3]
RT 90
FD 20
```

All the turtles turn right and go forward.

Turtles in Motion

The turtles can be set in motion with the SETSP (for SET SPeed) command.

```
SETSP 20
```

This command makes all the turtles move at the speed of 20. Larger numbers make the turtles move faster and smaller numbers make them move slower. Try varying the speed.

```
SETSP 50
SETSP 10
```

The turtles leave a trace behind them because their pens are down. If you want to erase these lines, you can use the pen eraser.
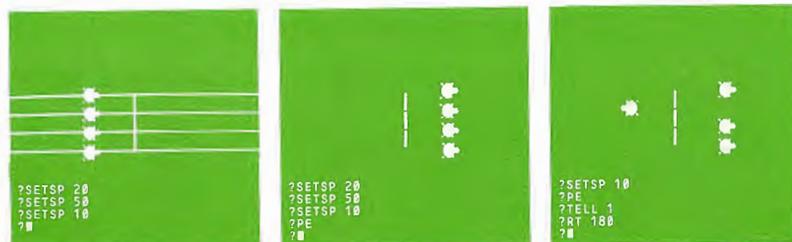
```
PE
```

When you set a turtle in motion, it always moves in the direction it is heading. That's why the turtles are now moving from the left to the right side of the screen. If you change the heading of a turtle in motion, the direction of the movement changes.

Let's make turtle 1 move in the opposite direction.

```
TELL 1
RT 180
```

You have probably noticed that Logo is free to do other things while the turtles are moving. This is very different from the turtle graphics we have been working with up until now.

If you want to get back to a clear screen with a single turtle, type

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

All the turtles are piled up one on top of the other. We'll hide them

```
HT
```

and then call up turtle 0.

```
TELL 0 ST
```

To change the turtle's state, put its pen down.

```
PD
```

Turtle's Color

When you call up the turtles, each one appears in a different color. You can change a turtle's color using the command SETC (for SET Color). For example,

```
TELL 0
SETC 20
```

makes turtle 0 gold. There are 128 colors numbered in the same way as the screen background and pen colors. The color table is found in Chapter 7.


```
?SETSP 20
?SETSP 50
?SETSP 10
?■
```

```
?SETSP 20
?SETSP 50
?SETSP 10
?PE
?■
```

```
?SETSP 10
?PE
?TELL 1
?RT 180
?■
```
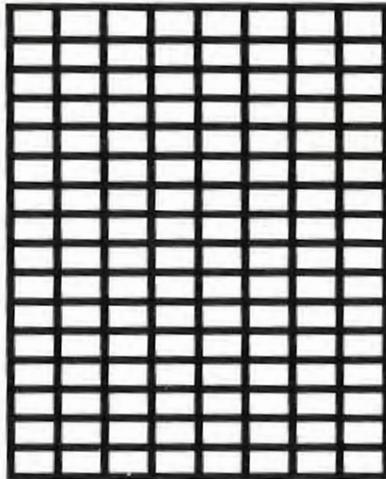


```
?TELL [0 1 2 3]
?CS
?HT
?TELL 0 ST
?■
```

Turtle's Shape

You can change the turtle into another shape: a rocket ship, a star, or a bird. You can create an unlimited amount of shapes using the shape editor, but only fifteen shapes are available at one time.

To use the shape editor, give the command **EDSH** (EDit SHape) followed by the shape number (1 through 15). The shapes start out blank each time Logo starts up. Let's try it.
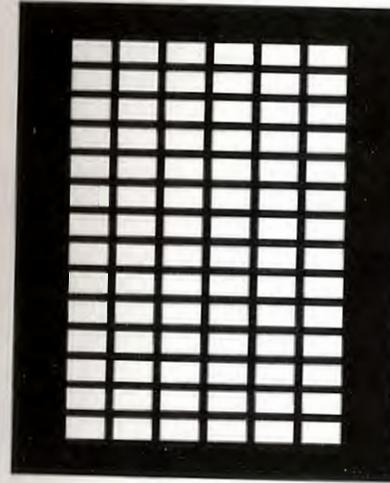
EDSH 1

Each shape appears as an 8 column by 16 row rectangular grid with the *cursor* in the top left corner. All the boxes in the grid start out empty. You can move the *cursor* anywhere within the grid by using the four **CTRL** arrow key combinations, **CTRL→**, **CTRL ←**, **CTRL ↑** and **CTRL ↓**. Pressing **CTRL** with any of the arrow keys moves the *cursor* within the grid without affecting the shape you are defining.

You can fill in the box under the *cursor* by pressing the **SPACE BAR**. The filled-in boxes within the grid define the shape you are creating. If you press the **SPACE BAR** when the *cursor* is on an empty box, the box fills in. If you press the **SPACE BAR** when the *cursor* is on a filled-in box, the box becomes empty. Try

moving the *cursor* around and filling in individual boxes using the **CTRL** arrow key combinations and the **SPACE BAR** alternately. Remember that the **SPACE BAR** does not move the *cursor*.

Try making shape 1 a hollow rectangle like the one in the illustration below.

When you have completed the shape, press **ESC** to exit from the shape editor.

Now use the **SETSH** (SET SHape) command to make turtle Ø appear as a hollow rectangle.

```
CS
TELL 0
ST
SETSH 1
```

To change the turtle back to its original turtle shape, type

SETSH 0

When a turtle is carrying a shape that you made, it can do everything that it does when it is carrying the regular turtle shape with one small exception. When you change the turtle's heading, your shape will not rotate to face that heading as the turtle shape does. But the turtle (with your shape) will move in the heading that you have given it.

Saving Shapes

When you exit from the shape editor (by pressing **ESC**), the shape that you have defined is in your workspace. However, it is not permanently saved. If you turn the computer off and then reboot Logo, all fifteen shapes will be blank again.

One way to save your shapes is by using the **GETSH** (GET SHape) operation and giving each defined shape a name. For example,

```
MAKE "BOX GETSH 1
```

gives the name **BOX** to the hollow rectangle you defined as shape number 1. **GETSH 1** *gets* your shape number 1, which is then named **BOX** by the **MAKE** command.

When you **SAVE** your work on cassette or diskette, the name **BOX** (whose value is your shape 1 — the rectangle) is saved.

When you later **LOAD** your work into your workspace, you can *put* one of your named shapes back into a shape number by using **PUTSH** (PUT SHape) followed by a shape number. For example,

```
PUTSH 1 :BOX
```

puts the **BOX** shape back into shape number 1.

Now when you type

```
TELL 0
SETSH 1
```

turtle 0 will take on the rectangle shape named **BOX**.

Collision Detection

ATARI Logo can detect *when* a turtle collides with a turtle drawing or another turtle. The possible types of collisions are numbered from 0 through 21 depending on which turtles (or which turtle and which drawing) are involved in the collision.

There are three kinds of drawings that Logo can distinguish. Logo can recognize these three kinds of drawings by the pen number (0, 1 or 2) used to draw.

The numbers symbolizing the different types of collision are outlined in Appendix B. Using these numbers with the **WHEN** command, you can set up a system of collision detection.

Let's see how it works. First, clear the screen completely and set your turtles back to their normal shape.
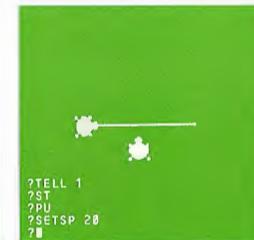
```
TELL [0 1 2 3]
CS
SETSH 0
HT
```

Call up turtle 0 and make it draw a horizontal line.

```
TELL 0 ST
SETPN 0
PD
RT 90
FD 50 BK 100
```

Call up turtle 1 and set it in motion.

```
TELL 1
ST
PU
SETSP 20
```

We've set up a collision. Now, we want turtle 1 to change direction each time it hits the line. You could type the command RT 180 and press RETURN when the turtle hits the line. To make Logo do this, type

WHEN 4 [RT 180]

Number 4 is the *symbol* for a collision between turtle 1 and a drawing made by pen number 0. You can find out the number for a collision by using the primitives OVER and TOUCHING. See Appendix B for more details as well as a list of all the possible types of collisions and their numbers.

The WHEN command calls up a "WHEN Demon". The WHEN Demon sits inside the Logo world and spends all its time watching for a collision. Whenever this collision happens, it jumps up and gives the command(s) in its list to the turtle(s) you are using at the moment. After that, the WHEN Demon resumes its watch. It is relieved of this task only when the CS command is given or when you clear the action by typing

WHEN 4 [ ]

While the WHEN Demon is watching the turtle involved in this collision, you may be talking to another turtle. If the turtle you are talking to and the one that the WHEN demon is watching are different turtles, the turtle you are talking to will carry out the commands in the WHEN list. To see this effect, type

TELL 0

Now turtle 0, instead of turtle 1, turns around whenever turtle 1 hits the line.

ATARI Logo Vocabulary
The following Logo primitives were introduced in this chapter.

Commands

EDSH
PUTSH
SETC
SETSH
SETSP
TELL
WHEN

Operations

GETSH

A Game
Project

This chapter explains how to develop a game project using the dynamic turtles. The project chosen here is a simple navigation game. A target, a few obstacles and a turtle appear somewhere on the screen. The player tries to guide the turtle around the obstacles to reach the target with the smallest number of moves.

The target can be a turtle drawing, or another turtle. Here, we use turtle 0 for the target and turtle 1 for the navigator, and set a WHEN Demon to watch for a collision between them.

Developing this kind of project can be simpler and easier to manage when you approach it in stages. For instance, our first version will not have obstacles. Turtle 1 will be set in motion and the player must use regular Logo commands like RT 45 or LT 30 to steer the turtle towards the target. In the second stage, we will refine the game by assigning keys or paddles to direct the turtle, and in the last stage we will introduce the obstacles.

Setting Up the Game

The first thing we need to do is to set up a target. Then we need to set up the turtle. One procedure will be good for both tasks. An example of a setup procedure is printed below. SETUP sets the turtle at a random position on the screen. It leaves the turtle's heading at 0 degrees (north).

```
TO SETUP
PENUP HOME
RT RANDOM 360
FD RANDOM 80
SETH 0
END
```

The Logo operation RANDOM outputs a number which Logo makes up. This number is always less than the one RANDOM is given as its input.

In SETUP, for example, the turtle turns RIGHT some angle which 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 79.

Notice that the last instruction in SETUP leaves the turtle facing north.

SETUP can be used to set up the target and the turtle as well as the obstacles. The following procedure, SETGAME, puts the target and the turtle at randomly selected positions using SETUP twice.

```
TO SETGAME
CS
TELL 0 ST
SETUP
TELL 1 ST
SETUP
END
```

Now try SETGAME. The game will be clearer if the target does *not* look like a turtle. So, let's use the hollow rectangle shape we edited as shape 1 in the last chapter. To use this shape, we add SETSH 1 in SETGAME. (Make sure that the shape is made before you try out this procedure.)

```
TO SETGAME
CS
TELL 0 ST
SETUP
SETSH 1
TELL 1 ST
SETUP
END
```

The procedure GAME uses SETGAME and then sets up the movement and the WHEN Demon.

```
TO GAME
SETGAME
SETSP 30
WHEN 19 [SETSP 0]
END
```

The last line makes turtle 1 stop its movement when it reaches the target. 19 is the collision number between turtle 0 and 1.

Try GAME a few times. For example,

```
GAME
RT 45
LT 10
```

Making a Key into a Game Button

There are many kinds of interactive programs that you can write. You can have Logo ask questions and receive answers in words or sentences. Here, we want to trigger Logo into action by the touch of a key. This requires using the operation RC. Type

```
PR RC
```

Logo waits for a key to be pressed.

Type the character A. RC receives the character A and passes it to the PRINT command.

The PRINT command then puts an A on the screen.

A

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

```
YOU DON'T SAY WHAT TO DO WITH X
```

RC is an operation like HEADING or POS. It is used as an input to another command or operation. For example, we could name RC's output using MAKE.

```
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.)

We can talk about this character by :KEY.

```
PRINT :KEY
```

and Logo will respond

```
Z
```

or whatever character you typed.

We can use this idea of naming in the following procedure. Imagine we have a procedure called LISTEN. If you type L or R it will listen to your order according to the following rules.

R means turn right 15 degrees
L means turn left 15 degrees

```
TO LISTEN
MAKE "ANSWER RC
IF :ANSWER = "R [RT 15]
IF :ANSWER = "L [LT 15]
END
```

In LISTEN, :ANSWER is what RC outputs. LISTEN then checks :ANSWER using the Logo primitive IF. IF requires two inputs: the first one must be a condition, either TRUE or FALSE, and the second, an instruction list. If the condition turns out to be TRUE, it carries out the instruction. In this example, we use the Logo operation =. The = (equal) compares two inputs and outputs TRUE when the inputs are the same, FALSE when they are not.

Logo must run the procedure LISTEN if a character is typed on the keyboard. KEYP is an operation to do just that. It outputs TRUE if a key is pressed or FALSE if no key is pressed. Using KEYP you can write a procedure that runs LISTEN only if a character is pressed. Type

```
TO PLAY
IF KEYP [LISTEN]
PLAY
END
```

Notice that PLAY is recursive. That is, the last line of the procedure PLAY calls PLAY. PLAY will continue running even though the turtle stops its movement. You must press BREAK to stop PLAY.

Try it with the GAME procedure.

```
GAME
PLAY
```

Press R or L to steer the turtle.

Bug Box

If you press R or L and nothing happens, you may have accidentally pressed the CAPS LOWR key. ATARI Logo will no longer understand your instructions and you will have to change to uppercase letters. To do this, simply hold the SHIFT key and then press the CAPS LOWR key.

Expanding the Game Project

In this section, we build a better target game out of GAME, SETGAME and PLAY. First, we will add a line in SETGAME to make the turtle draw some obstacles.

```
TO SETGAME
CS
TELL 0 ST
REPEAT 3 [SETUP PD FD 20]
SETUP
SETSH 1
TELL 1 ST
SETUP
END
```

The third line in SETGAME uses the command REPEAT to draw 3 short lines at random positions.

The GAME procedure should call up another WHEN Demon to specify what should happen when turtle 1 hits an obstacle. The collision number for turtle 1 and pen number 0 is 4. When that happens, the turtle will simply back up a few steps. It would also be nice if GAME printed some instructions.

```
TO GAME
RULES
SETGAME
SETSP 10
WHEN 19 [SETSP 0]
WHEN 4 [BK 6]
PLAY
END
```

```
TO RULES
SS
PRINT [STEER THE TURTLE TO HIT THE TARGET]
PRINT [TYPE R OR L TO TURN]
END
```

Try GAME now.

GAME

This is much better, but there is room for improvement.

PLAY continues running even when the turtle stops its movement. You must press BREAK to stop the game. You can change the procedure so that it will stop when the game is over.
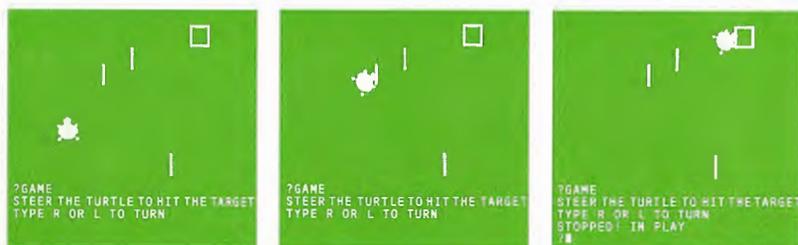
How can Logo know when the game is over? One way is by looking at the speed of turtle 1. Notice that the WHEN Demon set up in GAME orders SETSP 0 whenever the two turtles meet. PLAY must keep checking whether the turtle's speed is at 0. When the turtle's speed is 0, the game is over and PLAY must stop.

```
TO PLAY
IF SPEED = 0 [STOP]
IF KEYP [LISTEN]
PLAY
END
```

SPEED is an operation that outputs the speed of the turtle you are talking to at the moment.

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

GAME

Remember that you can give the commands R and L to turn the turtle. When the turtle reaches the target, the procedures stop and Logo displays the prompt on the screen.

You may be able to adapt this game and the techniques used in it to make other interactive games. You can also improve this game. For example, you could replace turtle 0 with another shape, or have Logo keep track of your score and change your rules so that the player will lose a point each time the turtle hits an obstacle.

ATARI Logo Vocabulary
The following Logo primitives were introduced in this chapter.

Commands

IF
STOP

Operations

KEYP
RANDOM
RC
SPEED
=

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 any other procedure; it can also call other procedures. Some procedures call themselves; these procedures are *recursive*. We have already used recursive procedures in Chapter 15. For example, POLY and SPI are both recursive.

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE          This is the recursive call.
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.

Let's think about this process. Imagine that Logo has an unlimited supply of helpers who are computer creatures living in the computer. Every time a procedure is called, a helper is called upon to look up the definition of the procedure. 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, its helper calls an FD helper. After the FD helper finishes, an RT helper is called. When it finishes, another POLY helper is called. The second POLY helper calls an FD helper, an RT helper, and a third POLY helper. Meanwhile, the first POLY helper is still waiting for the second POLY helper to finish. In this process, the FD helpers and the RT helpers finish their jobs (we don't know who they call for help). The POLY helpers never finish; they keep on calling for new POLY helpers.

When you use POLY, the process continues until you press BREAK. Not all recursive procedures work this way. They can be made to stop. 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 operation. We have used KEYP in the previous chapter.

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

KEYP outputs TRUE whenever any key on the keyboard is pressed. Using it with IF lets you stop SPI by pressing any key.

Here is a different kind of stop rule.

We could decide that SPI should stop if :STEP is greater than 150. So, we replace the first line by

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

After editing 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 it.

Designing a stop rule for POLY can be 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, that is, 360 degrees. But sometimes the turtle turns a multiple of 360 degrees.

In any case, we 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 POLY is called, we have to make Logo remember the turtle's heading. Type

```
MAKE "START HEADING
```

Then POLY can check to see if the turtle's current heading is the same as :START.

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

When you put the stop rule into the procedure, make sure that it's placed after the RT command. What will happen if you put it before? It will stop 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. If you forget to make Logo remember the starting heading before you run POLY, POLY does not work.

It is best to put that action into a procedure. Let's rename POLY and call it POLY1. We should also add :START as another input.

```
TO POLY1 :STEP :ANGLE :START
FD :STEP
RT :ANGLE
IF HEADING = :START [STOP]
POLY1 :STEP :ANGLE :START
END
```

Then, we can make a new POLY, which runs HEADING and give its result to POLY1.

```
TO POLY :STEP :ANGLE
POLY1 :STEP :ANGLE HEADING
END
```

Now POLY will do the whole job. Note that HEADING is an operation. That is why we don't put a : (colon) in front of it like we do to indicate the contents of a variable.

You can create all sorts of designs with SPI and POLY. Combining spirals and polygons, you can make new and dazzling designs.

This introduction to Logo is finished; but we hope you will explore other turtle projects on your own.

The ATARI Logo Reference Manual describes more advanced graphics features as well as many other features of ATARI Logo that you may want to try now.

Have fun!

**Useful Tools:
Circle and Arc
Procedures**

```
TO ARCRIGHT :RADIUS :DEGREES
ARCR1 .174532 * :RADIUS :DEGREES / 10
IF 0 = REMAINDER :DEGREES 10 [STOP]
FD .174532 * :RADIUS / 10 * REMAINDER→
 :DEGREES 10
RT REMAINDER :DEGREES 10
END

TO ARCR :RADIUS :DEGREES
ARCRIGHT :RADIUS :DEGREES
END

TO ARCLEFT :RADIUS :DEGREES
ARCL1 .174532* :RADIUS :DEGREES / 10
IF 0 = REMAINDER :DEGREES 10 [STOP]
FD .174532 * :RADIUS / 10 * REMAINDER→
 :DEGREES 10
LT REMAINDER :DEGREES 10
END

TO ARCL :RADIUS :DEGREES
ARCLEFT :RADIUS :DEGREES
END

TO ARCR1 :STEP :TIMES
REPEAT :TIMES [RT 5 FD :STEP RT 5]
END

TO ARCL1 :STEP :TIMES
REPEAT :TIMES [LT 5 FD :STEP LT 5]
END

TO CIRCLEL :RADIUS
ARCL1 .174532 * :RADIUS 36
END

TO CIRCLER :RADIUS
ARCR1 .174532 * :RADIUS 36
END
```

Comments: These arc and circle procedures actually draw a
36-sided polygon. (The number .174532 is the result of
computing $2 * \pi / 36$; π is rounded to 3.1416; and 36 is the
number of sides of the polygon.)

ATARI Logo has the capability of collision detection. This is done by the WHEN command which sets up a WHEN demon to watch for a particular kind of collision and to carry out a set of commands when it happens. (See Chapter 16.)

You can use the primitives OVER or TOUCHING as the first input to WHEN. OVER takes two inputs: the first is the turtle number and the second is the pen number. For example, if you want a turtle to turn around each time there is a collision between turtle 1 and a drawing made by pen number 0, you could type

`WHEN OVER 1 0 [RT 180]`

This is the same as typing

`WHEN 4 [RT 180]`

TOUCHING works the same way as OVER except its inputs are two turtle numbers. For example, if you want the current turtle to turn around each time turtle 1 and 2 collide, type

`WHEN TOUCHING 1 2 [RT 180]`

This is the same as typing

`WHEN 21 [RT 180]`

The types of collisions, numbered 0 through 21 are indicated in the table below. Numbers 0 to 14, (with the exception of numbers 3, 7 and 11) indicate a collision between a turtle and a drawing made by one of the three pens. (See Chapter 7 for more information on pen number.) Collision numbers 16 to 21 indicate collisions between two turtles.

The numbers 3, 7 and 15 are used as the codes for special events that are not collisions on a graphics screen but can be recognized by WHEN Demons.

Table of Collisions and Events

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

Collision number	Turtle number	Turtle number
16	3	0
17	3	1
18	3	2
19	0	1
20	0	2
21	1	2

**Using Joysticks
for Animation**

ATARI Logo Car Race

Here's an animated graphics program that takes advantage of some of ATARI Logo's exciting features. It will create two cars that race around a track at joystick controlled speed making noise as they go.

To start, copy the following patterns from grids 2 and 4 onto grids 1 and 3 respectively. To do this, use the Shape Editor, as described in Chapter 16.
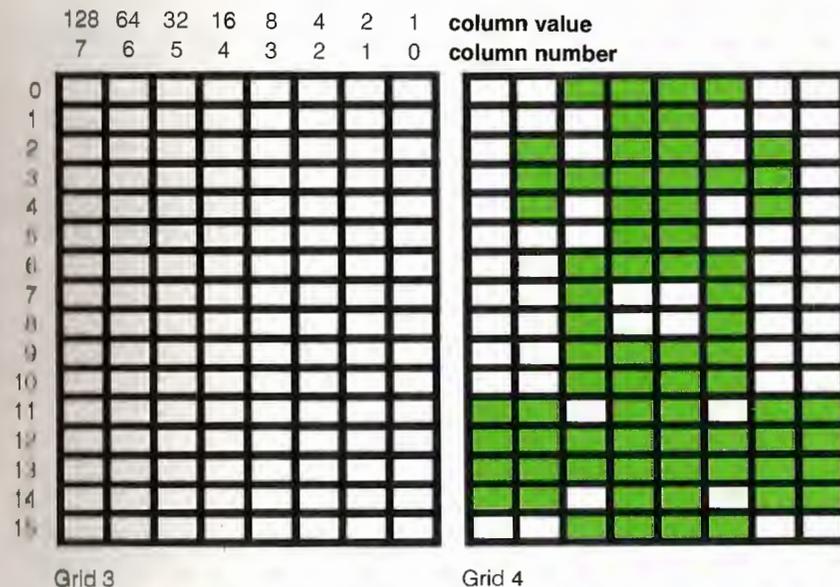
Type

```
EDSH 1
```


Grid 1   Grid 2

Type

```
EDSH 2
```


Grid 3   Grid 4

Type the following MAKE statements that set up variables "CAR1 and "CAR2. These will be the race car shapes. These statements make it possible for you to save the shapes when you save your workspace.

```
MAKE "CAR1 GETSH 1
MAKE "CAR2 GETSH 2
```

If you want to save these shapes by themselves at this point, type

```
SAVE "D:CARS
```

You can reload them later by typing

```
LOAD "D:CARS
```

Type the following procedures that set up the ATARI Logo Car Race. The comments will explain how they work.

```
TO RACE
SETUP
SET.COLLISION
TRIALS
END
```
The RACE procedure is the superprocedure of the car race program. It calls all the subprocedures.

```
TO SETUP
TELL 0 HT
FS
RACER1
RACER2
RACETRACK
FINISHLINE
PAINT.CARS
END
```
The SETUP procedure sets up the race track and puts the race cars at the starting line.

```
TO RACER1
TELL 1
PUTSH 1 :CAR1
PU ST
SETSH 1
SETPOS [-30 -100]
END
```
The RACER1 procedure sets up shape and position for the first car. The first car uses turtle 1 with a new shape.

```
TO RACER2
TELL 2
PUTSH 2 :CAR2
PU ST
SETSH 2
SETPOS [30 -100]
END
```
The RACER2 procedure sets up shape and position for the second car using turtle 2.

```
TO RACETRACK
SETBG 0
TELL 0
SETSP 150 WAIT 100 SETSP 0
END
```
The RACETRACK procedure draws the center line which divides the race track.

```
TO FINISHLINE
TELL 3 PU HT
SETPOS [-50 115]
PD
RT 90 SETPN 0
FD 100
SETH 0
END
```
The FINISHLINE procedure draws the finish line.

```
TO PAINT.CARS
TELL 1
SETC 7
TELL 2
SETC 27
END
```
The PAINT.CARS procedure sets the color of the first car to white and the second car to red. There are 128 colors to choose from and you can choose your own colors. (See Chapter 7).

```
TO SET.COLLISION
WHEN OVER 1 0 [SETBG BG + 1 IF BG = 1→
27 [SETBG 0]]
WHEN OVER 2 0 [SETBG BG + 1 IF BG = 1→
27 [SETBG 0]]
END
```
The SET.COLLISION procedure sets up the WHEN demons to change the background color when a car hits the finish line. One demon watches the first car and the other watches the second car.

```
TO TRIALS
GASPEDAL1
NOISE1
GASPEDAL2
NOISE2
TRIALS
END
```
The TRIALS procedure keeps the cars moving and the noise going continuously.

```
TO GASPEDAL1
TELL 1
SETSP (JOY 0) * 5
IF JOYB 0 [SETPOS [-30 -100]]
END
```
The GASPEDAL1 procedure moves the first car at a speed which is 5 times the position of the joystick in Controller Jack 1.

```
TO NOISE1
TOOT 0 (JOY 0) * 10 + 30 15 4
END
```

The NOISE1 procedure creates the engine sound for the first car.

```
TO GASPEDAL2
TELL 2
SETSP (JOY 1) * 5
IF JOYB 1 [SETPOS [30 - 100]]
END
```

The GASPEDAL2 procedure moves the second car at a speed which is 5 times the position of the joystick in Controller Jack 2.

```
TO NOISE2
TOOT 1 (JOY 1) * 10 + 30 15 4
END
```

The NOISE2 procedure creates the engine sound for the second car.

To Start the Race: Each player must hold down the joystick button. When ready to start, release the button and use the joystick to control the speed. If you do not use your joystick, the car will go backwards.

Index