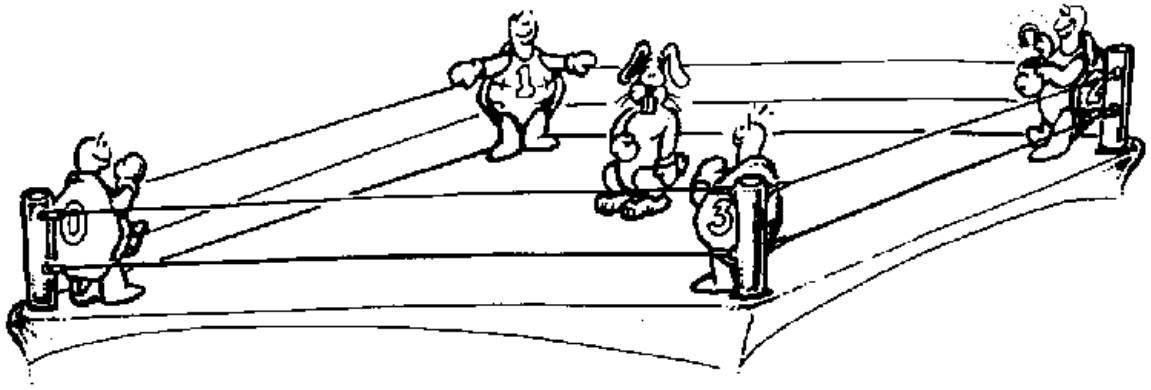
Chapter 11. Animating Multiple Turtles

"Wow, working with one turtle was bad enough. But working with hundreds?

"That's worse than working with rabbits!"



Simulating Multiple Turtles

Yes, there are lots of turtles in MSW Logo, 1024 of them. You can animate them, change their shapes, add sound effects, and all sorts of things.

But before you start getting busy with animating all those turtles, let's take a look at how you can use just one turtle to simulate multiple turtles that do lots of different things. It's a great review of things you've been doing up until now. And it will give you some ideas for working with UCB Logo, which only has one turtle.

The KALEIDOSCOPE procedure (KALEID.LGO) shows a good use of coordinate, color, and other commands.

The resulting picture looks as if it were drawn by multiple turtles.

```
TO KALEID :ANG :CNT
IF :CNT < 1 [STOP]
SETPC (LIST (RANDOM 128)+128 (RANDOM 128) ~
  +128 (RANDOM 128)+128)
;Try SETPC RANDOM 16 with other Logos
MOVE
KALEID :ANG + 5 :CNT - 1
END
```

The pen colors for red, green, and blue are set from 128 to 255 (0 to 127 plus 128). You may recall that the higher the number, the lighter the color. The lighter pen colors go with the darker screen colors set in the START procedure below. In that procedure, the colors are set randomly from 0 to 100 to produce a darker screen color.

```
TO START
CS HT
SETSC (LIST RANDOM 100 RANDOM 100 ~
  RANDOM 100)
;Try SETBG RANDOM 16 with other Logos
KALEID 0 50
WHATNOW
END
```

In the MOVE procedure, the turtle remembers it’s coordinates and then moves. The four simulated turtles each move in the four quadrants (the four quarters) of the screen based on the (+) positive and (-) negative values assigned to the X and Y variables.

```
TO MOVE
(LOCAL "X1 "Y1 "Y1 "Y2)
MAKE "X1 XCOR
```

```
MAKE "Y1 YCOR
FD RANDOM 100 RT :ANG + 15
MAKE "X2 XCOR
MAKE "Y2 YCOR
PU SETPOS (LIST (- :X1) :Y1) PD
SETPOS (LIST (- :X2) :Y2)
PU SETPOS (LIST (- :X1) (- :Y1)) PD
SETPOS (LIST (- :X2) (- :Y2))
PU SETPOS (LIST :X1 (- :Y1)) PD
SETPOS (LIST :X2 (- :Y2))
PU SETPOS (LIST :X2 :Y2) PD
END
```

Below is the MSW Logo procedure for running the procedure again. It creates a typical Windows dialog box where you can click on your response to run the kaleidoscope again or quit.

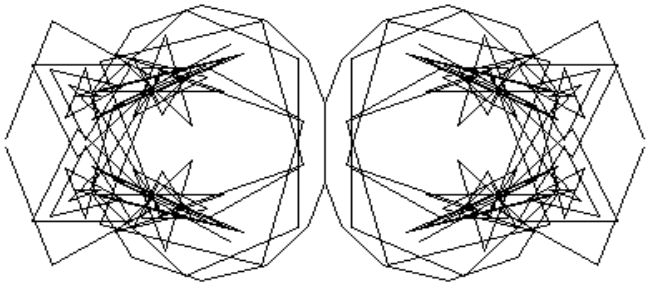
```
TO WHATNOW
CT MAKE "ANS YESNOBOX [AGAIN?] ~
  [RUN THE KALEIDOSCOPE AGAIN?]
IFELSE :ANS = "TRUE [START][CT STOP]
END
```

Try this procedure with non-windows versions of the language.

```
TO WHATNOW
CT
PR [ANY KEY TO RUN IT AGAIN, ESC TO
  TERMINATE]
IF RC = CHAR 27 [CT STOP] [START]
END
```

If you press the Esc key — CHAR 27 — the text is cleared and the procedure stops. Any other key and the START procedure is called.

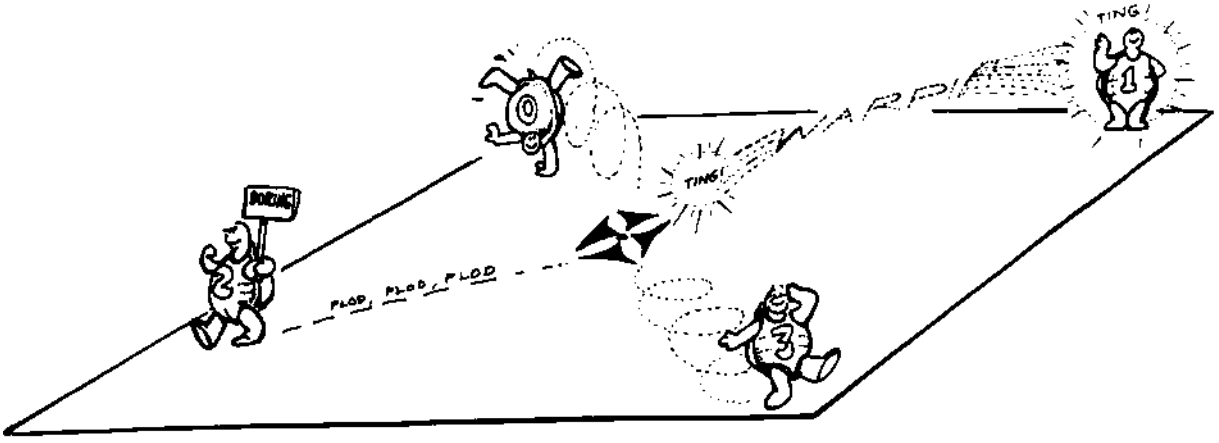
The drawing shown below was made with the KALEID2.LGO procedure. This procedure includes a couple of different ways of simulating multiple turtles. Imagine what this would look like if added different colors to it.



In fact, why not give that a try!

Independent Turtles

The MULTI.LGO procedure in the \procs\chpt11 subdirectory is another good example of using coordinate commands in a procedure. In the KALEIDOSOPE procedure, the simulated turtles all acted according to plan. In MULTI.LGO, you tell each one what to do.



Four turtles are defined in this new procedure even though only one is actually used. The SETUP procedure defines each turtle by defining its position and heading. XCOR, YCOR, and HEADING are each spelled out for each turtle so you can see how this procedure works.

Type START to begin. Then ASK a turtle — 0, 1, 2, or 3 — to do something.

You'll notice that ASK requires a turtle number (:TNUM) and a :COMMAND.LIST. So be sure to enclose your instructions to the turtle in brackets.

ASK 2 [REPEAT 4 [FD 100 RT 90]]

If nothing else, this little exercise is a great little demonstration of what you can do with turtle positions and headings.

A New Target Game

There's a new target game in \procs\chpt11 called ZAP.LGO. This is a more practical application of simulated turtles. Ernestine gets the chance to shoot herself down.

This simple procedure is also just the beginning of what you can do with this game. There are lots of things you can do to dress it up. But first let's see how it works.

The ZAP procedure get's you started. It tell's you what you have to do.

To start the actual game, press **Z**.

The Z procedure sets up the game. Turtle 0 is put in a random position on the screen. It's pen is put down so that it will draw a short line to

show you the direction
in which it's moving.

Your job is to guess the direction and speed of your turtle to intercept the first turtle. The target turtle is going to keep moving across the screen in the direction it's heading, once you guess how to catch it.

You have to set the heading to intercept Turtle 0. And you have to set the speed. You'll get used to the speed by trial and error. The higher the speed, the greater the chance for error. Try keeping it in the 5 to 20 range.

Once you enter the direction and speed, the computer takes over. If you come within 10 turtle steps of Turtle 0, you'll get the CHEERS procedure. Otherwise, you'll have to try again.

Read the procedures carefully. Then try this game a few times.

- What can you do to make it easier?
- How would you make it harder?
- What would you do to make the CHEERS procedure a bit flashier. Printing Congratulations is a bit dull.

There are lots of things you can do to make this game better. Go do it!

Simulations

Before we go too far, there is another realm to explore, the realm of simulations. Among other things, simulations are ways to get the computer to act out your "What if?" wishes. Actually, you have already been simulating multiple turtles.

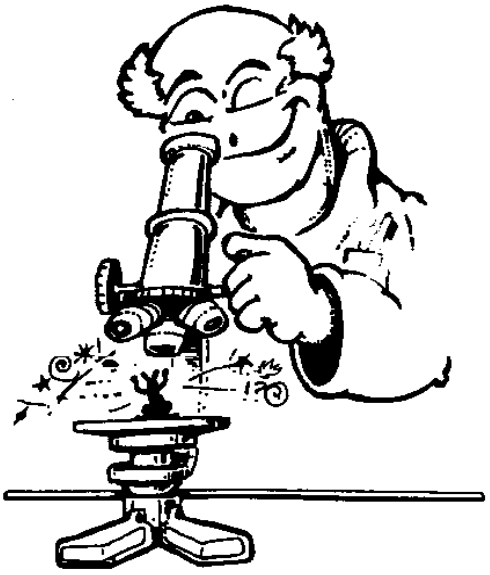
Maybe now you can get the simulated turtles to simulate others kinds of behavior.

You've heard about aircraft simulators. These are computer-controlled airplane cockpits that allow airline pilots to train in all sorts of emergency situations without ever leaving the ground.

You may have gone to arcades where they have driving or flying simulators. And, of course, there are the flight simulator software programs you can buy for your own computer.

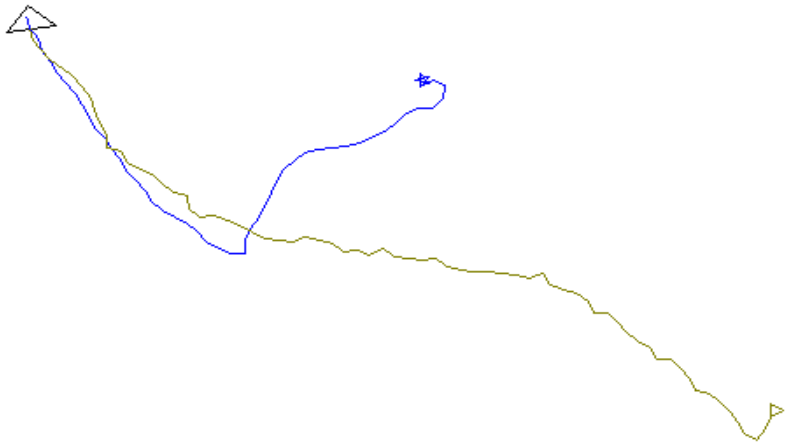
Logo Science

An interesting use of simulators is in analyzing behavior. For example, how would a mouse find its way out of a maze.



One way that researchers discovered was that every time this one mouse came to a wall, it turned right. It eventually found its way out.

You'll find a behavior simulation on the CD that came with this book, BEHAVIOR.LGO. This offers three animal simulations: Find By Smell, Find by Sight, Chase and Evade.



The "Find" simulations are rather straight forward. The Chase and Evade simulation is fun. Will Find.By.Sight catch Avoid.By.Smell before Avoid can get out of the playing area?

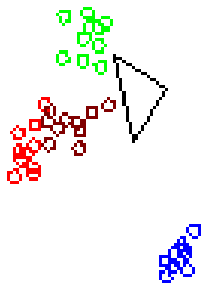
Here's a challenge! BEHAVIOR.LGO uses one turtle to simulate the actions of two. Change the procedures to actually use two turtles.

Cellular Behavior

Another interesting simulation can be found in CELLS.LGO. This is an example of Logo used in medical education. The START procedure lists a message that was posted on CompuServe's Logo Forum.

CELLS.LGO is the response.

Three groups of cells are drawn randomly on the screen. The turtle always seeks out the red cells on which



additional cells are
grown.

Your Challenge

Create an AVOID procedure. Currently, the turtle will move right over the green and blue cells to find the red ones. Your job is to create a procedure that makes the turtle move around the green and blue cells while still seeking the red cells.

Yes, it can be done. Give it a try.

Working With Multiple Turtles

OK! You've got the idea. Now the question is: how do you go about doing that — work with multiple turtles, that is?

In MSW Logo, there are 1024 turtles numbered from 0 to 1023. To talk to a specific turtle, use the SETTURTLE command. Or why not write a simple TELL procedure? In that way, you keep talking the same language as you did when simulating multiple turtles.

```
TO TELL :TNUM
SETTURTLE :TNUM
END
```

You can set the heading, position, size, and pen control for each turtle, but not the pencolor. All the turtles use the same pen color.

We started this book with a race between the tortoise and the hare. So how about a race just for turtles? Ernestine and her cousins love to race.

```
TO RACE
TELL RANDOM 4 FD RANDOM 10
TELL 0 IF YCOR > 200 ~
```

```
  [PR [TURTLE 0 IS THE WINNER!] STOP]
TELL 1 IF YCOR > 200 ~
  [PR [TURTLE 1 IS THE WINNER!] STOP]
TELL 2 IF YCOR > 200 ~
  [PR [TURTLE 2 IS THE WINNER!] STOP]
TELL 3 IF YCOR > 200 ~
  [PR [TURTLE 3 IS THE WINNER!] STOP]
RACE
END

TO START
TELL 0 PU SETPOS [-100 -150] ST
TELL 1 PU SETPOS [-50 -150] ST
TELL 2 PU SETPOS [0 -150] ST
TELL 3 PU SETPOS [50 -150] ST
RACE
END
```

This race procedure is very simple. All it does is place four turtles in a race from the bottom to the top of the screen.

1. Why not draw a race track on which the turtles can run?
2. Why not add some pizzazz to the announcement of a winner? Some flashing lights? Maybe some music?
3. Change the shapes of the turtles into race cars.

Changing the Shape of the Turtle

What about changing the shape of the turtle? This is where animation and multimedia starts to creep into Logo.

CAR.LGO and BIPLANE.LGO are procedures you can use to change the shape of the turtle. You can edit the procedures to make them bigger or smaller depending on how you are going to use them.

Morf's just an old fashioned guy who likes the old stunt planes. So he made one of his own. He also made a bitmap files of the race car and the biplane that you'll find in the graphics directory, in case you want to use it as a turtle shape.

Let's stick with the racing theme.

1. Run the CAR procedure.

   The car image is displayed.

2. Pick up the pen and move to the lower left corner of the car image.

   PU SETXY -2 -6

3. Type BITCUT 25 45

   The car image disappears. It is cut to the clipboard.

   More about loading and cutting graphic images in the next section. For now, let's focus on the turtle.

4. Type BITMAPTURTLE

   The car image is taken from the clipboard and "mapped" or changed into a turtle.

   Now you can drive your race car around the screen.

   PD FD 100 HOME

Remember, you raised the pen before you cut the image to the clipboard. You have to put it back down again before you can start drawing.

Saving Your Drawings

Before you go too far, there's something important you need to know.

- You've written a procedure to create a great drawing.
- You already know about saving the procedure.
- What about saving the drawing?

Let's explore this using Morf's biplane, OK? BIPLANE.LGO is in the \procs\chpt11 subdirectory. Erase everything else in your workspace using the ERALL command. Then load the biplane. After it is loaded into Logo, type **BIPLANE** and press **Enter**.

Morf's biplane is displayed.

You can save this drawing as a bitmap using the Bitmap menu or the BITSAVE command. However, you'll be saving one humongous file, typically over 2 megabytes. Most of that will be wasted white space. To check this out:

1. Click on the Bitmap menu.

   The Bitmap menu is displayed.

2. Select **SaveAs...**

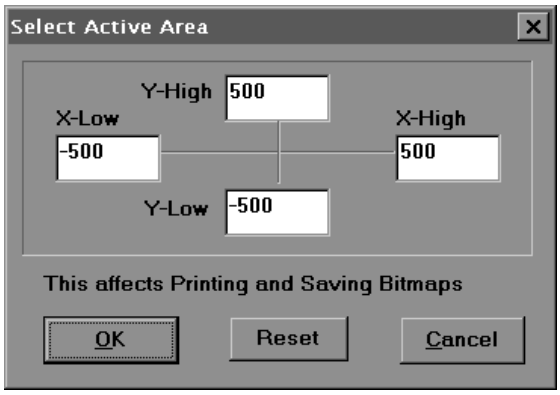
   The SaveAs dialog box is displayed.

3. Type the name of the image. Name it whatever you want, for example TEST.BMP. Then click on OK.

   The image is saved as a BMP file.

Now that the picture is saved, go take a look at its size, using the Windows File Manager, Explorer, or MS-DOS. Pretty big, isn't it?

So let's cut it down. You do this by reducing the active area to something just larger than the picture. As we said earlier, unless you redefine the active area as something smaller, MSW Logo sets it as 1000 pixels (turtle steps) wide by 1000 high.



How big does it have to be to accommodate Morf's biplane? You'll have to do some exploring to determine that. The first place to look is at the procedures. What do they tell you about the dimensions of the drawing?

Or you can just guess?

1. With the biplane displayed on the screen, type ST to show the turtle.

   The turtle is displayed near the tail of the biplane.

2. Lift the pen using PU. Then estimate how far it is to just beyond the propeller. For example, type SETX 100.

   The turtle moves to the left just beyond the propeller.

3. Good guess! Now you want to move down to just below the wheels. How far do you think that is?

   Type SETY - <your guess>. How'd you do?

4. Now you are at the lower left corner of the picture.

   You've got two of the four numbers required for the active area.

5. How high above 0 does the biplane extend? One way to experiment is to try using BITCUT.

   BITCUT 100 50

   That doesn't quite do it, does it?

6. Try something just a bit bigger. How about 102 and 56?

   The entire biplane is cut. Works for me!

7. Type **CS** and run the biplane procedure again.

   The biplane graphic is displayed.

8. Open the **Bitmap** menu and select **Active Area...** You can also use the SETACTIVE AREA command.

   SETACTIVEAREA [<*Xlow Ylow Xhigh Yhigh*>]

   SETACTIVEAREA [-100 -25 2 31]

9. Now open the Bitmap menu and select SaveAs...

   The current directory is displayed.

10. Save the new active area as TEST.BMP, writing over the existing file.

    This new BMP file is much smaller, isn't it?

Now you have a BMP image that you can use as a turtle. However, you've still got a problem!

When you start moving a bitmap image around the screen as a turtle, you'll notice that the turtle trail comes from the lower left corner of the bitmap. That's where the center of the turtle would be if it were visible.

However, you're going to run into trouble when you want to turn. The picture doesn't turn the way that Ernestine does. Try it.

FD 1OO RT 90 FD 100

Looks like the car went into a 100 step skid sideways. If you're using the biplane, it all of the sudden became a helicopter.

To show the car or biplane facing other directions, you have to add additional bitmaps. The easiest way to add these is to load your bitmap into a program such as *Paintbrush* (it comes with Windows) or *Paint Shop Pro*, a popular shareware product that's included on the CD that comes with this book. Rotate the bitmap and save each view as a separate bitmap.

But then, how do you load each bitmap into Logo? Let's take a look at that next.

Animating the Turtle

Graphics are much more fun when they seem to come to life. Animating the turtle gives you the chance to play movie producer, director, script writer, and even actor all at the same time.

Making movies isn't really all that complicated. Even with all the exotic computer-generated special effects, movies are simply a series of still pictures or photographs that are run in front of a lens at speeds generally from 32 to 64 frames (or pictures) per second.

Take a look at ANIMATE.LGO in the \procs\chpt11 subdirectory. Type START to watch a simple stick figure appear to wave its arms.

When you look at the procedure, you see that the arms are drawn and then erased. The turtle turns 10 degrees and then

repeats the process. If the motion seems too slow on your computer, take out the WAIT 2 commands.

For some other example of turtle animation, see Puff, the magic dragon, and Gretchen's Balloon. You saw them in Chapter 8.

Animating Other Shapes

An easy place to start animating other shapes is with a bouncing ball. There are five colored balls in the graphics directory on the CD that came with this book. Copy those <*color*>.BMP files into your MSW Logo directory. Then we can get down to business.

Let's start with one bouncing ball and a procedure named BOUNCE.LGO.

```
TO START
BITLOAD "REDW.BMP
BITCUT 32 32
BITMAPTURTLE
MAKE "HGHT 200
PU SETY :HGHT
BOUNCE :HGHT
END

TO BOUNCE :HGHT
IF :HGHT < 20 [HOME STOP]
DOWN
UP :HGHT
BOUNCE :HGHT - 20
END

TO DOWN
IF YCOR = 0 [STOP]
BK 5 WAIT 1
DOWN
```

```
END

TO UP :HGHT
IF YCOR = :HGHT [STOP]
FD 5 WAIT 1
UP :HGHT
END
```

Most of this is old stuff. The new commands are right there at the beginning of the START procedure. This time you're going to load another image into Logo and then change it into a turtle.

```
BITLOAD "REDW.BMP
BITCUT 32 32
BITMAPTURTLE
```

These commands are key to changing the shape of the turtle.

1. BITLOAD <*Bitmap File*> Loads a graphics file such as REDW.BMP

   You can load just about any graphic image into MSW Logo. It doesn't have to be something you created in Logo. It can be clip art from another program. It can be a photograph or artwork that you scanned into the computer. There's only one catch.

   Graphics files must be in the BMP format.

   You can use the *Paintbrush* program that comes with Windows or *Paint Shop Pro*, a shareware program included in the CD, to convert graphics from other formats to the BMP format.

   BITLOAD is like LOADPIC, LOADPICT, LOADSNAP and similar commands used in other versions of Logo.

2. BITCUT <*width*> <*height*> cuts the rectangular shape

that you define to the clipboard.

You can cut rectangular shapes of most any size and then change them into turtles. This is particularly useful when you want to precisely position a larger graphic as a background on which you draw other designs or as the background over which you move other turtles.

Graphics programs as well as clip art and scanning utilities usually show or allow you to ask the size of a graphic image. If you don't know the size of the image, MSW Logo offers the advantage of the BITLOADSIZE command. This tells you the width and the height of the bitmap you loaded into Logo.

BITCUT is like the SNAP command used in some other versions of Logo.

3. BITMAPTURTLE maps, or changes, the bitmap image to Turtle 0 or to the turtle number set by the SETTURTLE command. More on multiple turtles is coming up.

SPECIAL NOTE: Want to learn more about computer graphics? Read the GRAPHICS.PDF file that's on the CD that came with this book. That may well tell you more than you ever wanted to know.

Before we go too far, take a look at what you've got here. The red ball appeared and began to bounce, gradually coming to a stop. Now you've got a red ball sitting on a white background. This is your turtle. Ernestine is taking a rest.

Give the turtle a few commands to see how it works with this new shape. You'll see that the turtle's pen is in the lower left corner of the shape. To get a better idea of what you're dealing with, change the screen color.

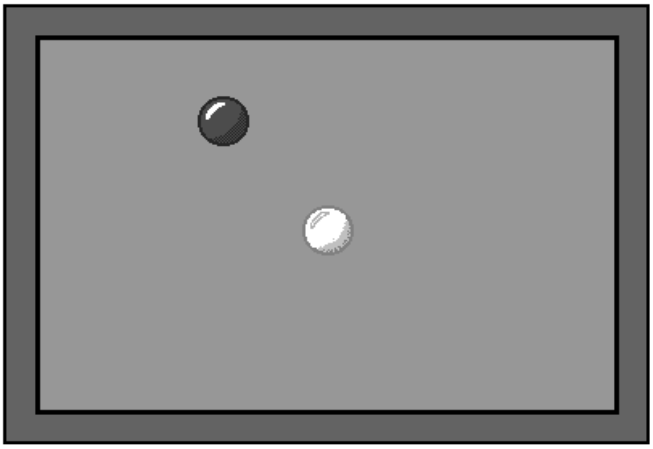
What happens?

Now it’s easier to see that the turtle doesn’t actually turn the way that Ernestine does. It just moves off in the direction in which you send it.

After you play around with the red ball turtle, see what else the other Bounce procedures offer. They’re in the \projects\chpt11 subdirectory of the CD that came with this book.

From One to Two Turtles

Here’s another procedure that can get you started on a billiards game.



Here you have a red ball and a white ball that bounce around the table and off of each other when they collide. It’s the BILRDS.LGO procedure in the \procs\chpt12 subdirectory on the CD. This procedure goes a step beyond the bouncing ball procedure.

```
TO SETUP
...
SETBITINDEX 0
BITLOAD "REDG.BMP
```

```
BITCUT 32 32
SETTURTLE 0
BITMAPTURTLE
...
SETBITINDEX 1
BITLOAD "WHITEG.BMP
BITCUT 32 32
SETTURTLE 1
BITMAPTURTLE
...
END
```

When you start working with multiple turtles you have to add some additional commands. Other versions of Logo have their own way of doing this. MSW Logo may seem to be a bit more complicated. But actually, it tells you more about what's actually going on inside Windows.

- SETBITINDEX <*index*> indexes or reserves a space on the clipboard for the next image to be cut (up to 1024 images).
- BITLOAD loads the bitmap image from your hard drive.
- BITCUT cuts the image to the clipboard as the indexed image.
- SETTURTLE <*number*> identifies the turtle by number.
- BITMAPTURTLE maps the most recently indexed image to turtle number identified by SETTURTLE.

In the BILRDS.LGO procedure, two balls images are set up as turtles 0 and 1.

- What would it take to make a realistic billiard game?
- How would you make a pocket billiards game?

Racing Cars

Early in this chapter when you first read about working with multiple turtles, you played with a RACE procedure. Let's fix up that procedure so you're racing multiple cars rather than multiple turtles.

Take a look.

```
TO START
CS
REPEAT 4 ~
  [
  SETBITINDEX REPCOUNT - 1
  BITLOAD "RACECAR.BMP
  BITCUT 30 58
  SETTURTLE REPCOUNT - 1
  BITMAPTURTLE
  ]
SETTURTLE 0 PU SETPOS [-100 -150] ST
SETTURTLE 1 PU SETPOS [-50 -150] ST
SETTURTLE 2 PU SETPOS [0 -150] ST
SETTURTLE 3 PU SETPOS [50 -150] ST
RACE_CAR
END

TO RACE_CAR
SETTURTLE RANDOM 4
WAIT 20
FD RANDOM 20
IF YCOR > 200 [FLAG STOP]
RACE_CAR
END

TO FLAG
(PR "TURTLE TURTLE [IS THE WINNER!])
```

```
MAKE "ANS YESNOBOX [RACE] ~
  [WANT TO RACE AGAIN?]
IFELSE :ANS = "TRUE [START] ~
  [CS CT PR [SEE 'YA AT THE RACES.]]
END
```

This is quite a bit different from the first RACE procedure.

First of all, the RACE_CAR procedure is much simpler. There's really no reason to test all four turtles after each step. You only need to test the last one that moved.

Did that turtle — or car — go past the Y coordinate of 200 or not? If so, the FLAG procedure is called.

The TURTLE command tells you which turtle is active. In this case, the active turtle is the one that crossed the finish line first.

```
(PR "TURTLE TURTLE [IS THE WINNER!])
```

Why [FLAG STOP]?

When the winning car crosses the 200 Y-coordinate, the FLAG procedure takes control from the RACE procedure. When you stop the FLAG procedure, control passes back to where Logo left the RACE_CAR procedure. The next line says RACE_CAR, which isn't exactly what you wanted.

In the FLAG procedure, when you left-click on NO in the RACE_CAR box, the first thing that happens is that the screen and text are cleared. When you clear the screen or type NOBITMAPTURTLE, the turtle reverts back to her original form.

What About REPCOUNT

When you want to use one or more other shapes as the turtle, you have to have a way to know which bitmap works with which turtle. Let's review for a minute.

To talk to each turtle, you use the SETTURTLE command — SETTURTLE 0, SETTURTLE 1, all the way up to SETTURTLE 1023, if you want.

To help match bitmaps to turtles, you use the SETBITINDEX command.

```
REPEAT 4 ~
  [
  SETBITINDEX REPCOUNT - 1
  BITLOAD "RACECAR.BMP
  BITCUT 30 58
  SETTURTLE REPCOUNT - 1
  BITMAPTURTLE
  ]
```

SETBITINDEX takes one input, an index number. In the START procedure, there's a new way of assigning that number. It's the REPCOUNT command.

REPCOUNT is used with the REPEAT command. In short, it counts the number of repeats. For example:

```
REPEAT 3 [(PRINT [THIS IS REPEAT NUMBER] ~
  REPCOUNT)]
THIS IS REPEAT NUMBER 1
THIS IS REPEAT NUMBER 2
THIS IS REPEAT NUMBER 3
```

REPCOUNT outputs the number of repeats including the current one. It starts at number 1. Since the goal of the procedure is to match the index numbers with the turtle

numbers (that start at 0), the START procedure uses REPCOUNT - 1.

The next step is to load the race car bitmap and cut it to the clipboard.

```
BITLOAD "RACECAR.BMP
BITCUT 30 58
```

Next, you match the bitmap index to a turtle. Again, you use the REPCOUNT command to count the four race cars.

```
SETTURTLE REPCOUNT - 1
BITMAPTURTLE
```

Finally, you assign the bitmap that is currently on the clipboard to the selected turtle using BITMAPTURTLE. This process is repeated four times to assign a race car bitmap to each of four turtles.

Awfully simple? Or simply awful?

You find a number of examples of REPCOUNT in the \projects\chpt11 subdirectory.

Keyboard Control

In a game or an animated show, you may want the user to provide a direction, a distance, or some other form of response while the procedure is running. Most versions of Logo use READCHAR or RC, READWORD or RW, READLIST or RL. Each of these commands stops the action and awaits your input. Nothing happens until you type a character, a word, or a list.

MSW Logo provides another form of keyboard control. This one is an example of Windows modeless programming that you'll read more about in the next chapter.

To get you started, here's a simple procedure for doodling on the screen.

```
TO DOODLE
CS
MAKE "STEP 0
KEYBOARDON [COMMAND ]
SETFOCUS [MSWLOGO SCREEN]
ST
END

TO COMMAND
MAKE "KEY CHAR KEYBOARDVALUE
IF :KEY = "R [RIGHT 30]
IF :KEY = "L [LEFT 30]
IF :KEY = "U [PENUP]
IF :KEY = "D [PENDOWN]
IF :KEY = "C [DOODLE]
IF :KEY = "Q [CS KEYBOARDOFF]
FD :STEP
IF NUMBERP :KEY [MAKE "STEP :KEY]
END
```

To run the procedure, type DOODLE and then press a number key to set the speed of the turtle. Make sure the **Caps Lock** key is On. Then press and hold any key to move the turtle forward. You can then change directions or lift the pen up whenever you want.

This procedure introduces some new commands: KEYBOARDON, SETFOCUS, and KEYBOARDVALUE. KEYBOARDON allows you to trap events, such as pressing a key, that take place in the window that has "focus." That's the window that's selected at the time; the one with the highlighted title bar. Other windows have a grey title bar.

Want to see which window has focus? Use the GETFOCUS command.

1. Open MSW Logo.
2. Click in the Commander window.
3. Type GETFOCUS. What happens?
4. Type EDALL in the Input Box.
5. Type SETFOCUS "EDITOR in the Input Box. Which title bar is highlighted?

Typically, the Commander window has focus when you're running Logo procedures. Logo interprets the commands and executes them one after the other. KEYBOARDON traps events that take place in the MSW Logo screen, which is why you must SETFOCUS to that screen.

KEYBOARDVALUE outputs the ASCII value of the key that is pushed. You'll read about the ASCII code, CHAR, and ASCII commands in the next chapter. Here's a variation of DOODLE that shows you how those values are used.

DOODLER doesn't care if the Caps Lock key is on. The numbers in the list (in the brackets) are the upper and lower case ASCII values of the first letter of each command.

```
TO COMMAND :KEY
IF MEMBERP :KEY [70 102] [ FD 30 STOP ]
IF MEMBERP :KEY [66 98] [ BK 30 STOP ]
IF MEMBERP :KEY [82 114] [ RT 30 STOP ]
IF MEMBERP :KEY [76 108][ LT 30 STOP ]
IF MEMBERP :KEY [67 99] [ CS STOP ]
IF MEMBERP :KEY [81 113] [ QUIT STOP ]
```

```
END

TO DOODLER
ICON "COMMANDER
SETFOCUS [ MSWLOGO SCREEN ]
KEYBOARDON [ COMMAND KEYBOARDVALUE ]
END

TO QUIT
KEYBOARDOFF
UNICON "COMMANDER
END
```

To get your keyboard back to normal, use KEYBOARDOFF. This disables the trapping of events by KEYBOARDON.

For another look at how these keyboard control commands can be used, take a look at the MIDI.LGO procedure in the Examples directory. This was set up when you installed MSW Logo.

Controlling the Turtle's Speed

There are times when the turtle seems to go too fast. For example, if you have a fast Pentium computer, the DOODLE procedure moves along pretty fast even when the STEP variable is 1.

There are various ways to handle that using WAIT commands and other timing procedures.

```
TO TIMER :TIME
IF :TIME = 0 [STOP]
TIMER :TIME - 1
END
```

If your version of Logo doesn't have speed control, put a TIMER procedure in between moves the turtle might make. It's one simple way to slow things down.

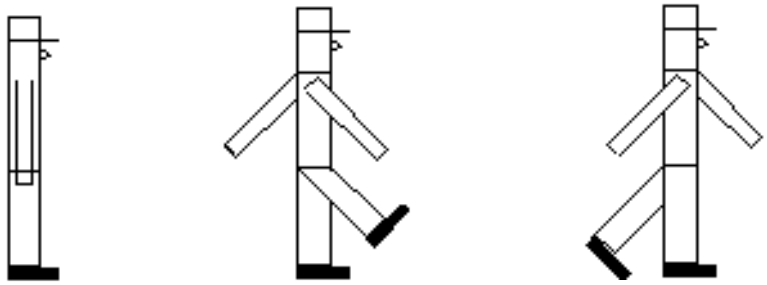
For better control, here's a procedure from George Mills, the person responsible for MSW Logo.

```
TO DOKEY
MAKE "KEY CHAR KEYBOARDVALUE
IF :KEY = "+ [MAKE "DELAY :DELAY - 50]
IF :KEY = "- [MAKE "DELAY :DELAY + 50]
IF :DELAY < 50 [MAKE "DELAY 50]
SETUP.TIMER
END

TO DOSTEP
FD 10
END

TO MAIN
MAKE "DELAY 1000
SETUP.TIMER
KEYBOARDON [DOKEY]
SETFOCUS [MswLogo Screen]
END

TO SETUP.TIMER
SETTIMER 1 :DELAY [DOSTEP]
END
```

Now go see what you can do.

Logo Animation

One way to create animation is to use different turtles, each showing part of an action. Here are three very simple drawings of a soldier walking.



Look at the SOLDIER.LGO procedure. If you have a fast computer, run soldier and watch the soldier as he seems to walk across the screen.

You can also create three turtles and have them appear to be marching across the screen.

Whoever saw a marching turtle?

Well, here's a very simple start that you can build upon. It uses STAND.BMP, LWALK.BMP, and RWALK.BMP to march across the screen.

```
TO START
SETUP
SETTURTLE 0
SETX -200 ST WAIT 30
CT PR [FORWARD, MARCH!]
WAIT 10
REPEAT 15 [MARCH]
CT PR "HALT
SETTURTLE 0 ST
END
```

```
TO MARCH
LOCAL "X
MAKE "X POS HT
SETTURTLE 0
ST SETPOS :X
WAIT 10 FD 25
MAKE "X POS HT
SETTURTLE 1
ST SETPOS :X
WAIT 10 FD 25
MAKE "X POS HT
SETTURTLE 2
ST SETPOS :X
WAIT 10 FD 25
MAKE "X POS HT
END

TO SETUP
CS CT
SETBITINDEX 0
BITLOAD "STAND.BMP
BITCUT 30 110
SETTURTLE 0
BITMAPTURTLE
PU SETH 90 HT
SETBITINDEX 1
BITLOAD "RWALK.BMP
BITCUT 80 110
SETTURTLE 1
BITMAPTURTLE
PU SETH 90 HT
SETBITINDEX 2
BITLOAD "LWALK.BMP
BITCUT 80 110
```

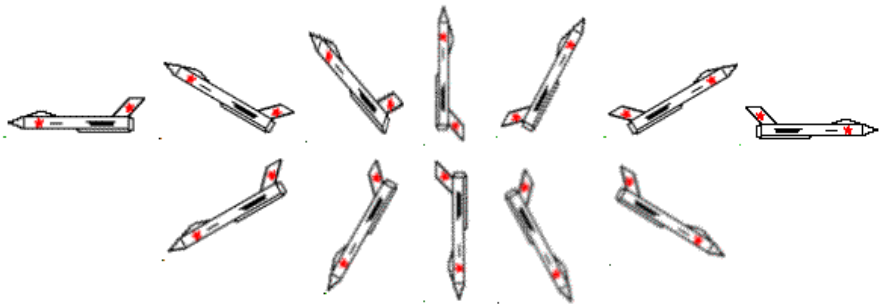
```
SETTURTLE 2
BITMAPTURTLE
PU SETH 90 HT
END
```

The SETUP procedure loads the three bitmaps, cuts each to the clipboard, and assigns each to a turtle. The MARCH procedure tells each turtle to take a step forward and then tells the next turtle where it is so that it can take a step from that position.

You can add additional views to make the action smoother. You can also create pictures using MS PAINT or another graphics program and then animate those pictures.

Flying the Firefox

Do you remember the FIREFOX procedure? You might want to run it from the CD that came with this book if you don't remember. In the graphics directory, you'll find twelve separate files for each view of the Firefox.



Each bitmap file (*.BMP) is named for the direction it is headed: N.BMP for North, rth East, and so on.

There's also another one called

HIT.BMP. You can use this one for when you Zap the Firefox.

Ready to start flying? Take a look at the FLY.LGO procedure in the \procs\chpt11 subdirectory on the CD. Read the START procedure to learn how to fly the jet.

Here are three of the procedures that control the flight of the Firefox. They show another use of the Keyboard commands.

```
TO FLY
CS READY
MAKE "STEP 0
KEYBOARDON [CONTROL]
SETFOCUS [MswLogo Screen]
TELL 3 PU RT 90 SETX -400 ST
ICON "COMMANDER
END

TO CONTROL
MAKE "XY POS
MAKE "KEY CHAR KEYBOARDVALUE
IF :KEY = "R [RT 30 MAKE "H HEADING HT ~
  TURN PU SETPOS :XY SETH :H ST STOP]
IF :KEY = "r [RT 30 MAKE "H HEADING HT ~
  TURN PU SETPOS :XY SETH :H ST STOP]
IF :KEY = "L [LT 30 MAKE "H HEADING HT ~
  TURN PU SETPOS :XY SETH :H ST STOP]
IF :KEY = "l [LT 30 MAKE "H HEADING HT ~
  TURN PU SETPOS :XY SETH :H ST STOP]
IF :KEY = "Q [QUIT STOP]
IF :KEY = "q [QUIT STOP]
FD :STEP
IF NUMBERP :KEY [MAKE "STEP :KEY]
END
```

```
TO QUIT
KEYBOARDOFF
UNICON "COMMANDER
CS
END
```

Logo Flight Simulator

With all this talk about airplanes, why not develop a flight simulator?

It's not really that hard to do. You can use the Firefox graphics or you can develop new ones.

Of course, you'll need to create an airport graphic with maybe a hanger, airstrip, and trees. It's easier if you create this as a picture file and then just load the picture rather than have this drawn each time you decide to go flying.

Now let's go flying. The simplest thing to do is make a group of key controls for the aircraft: Up, Down, Level flight, Slow Down, Speed up, Stop

You want the aircraft to start rolling down the runway, picking up speed as it goes. When you think you're going fast enough, you want to take off. If you're not going fast enough and you try to take off, you crash. Make sure you create a picture of a crash scene.

How fast is fast enough? That's something you can set with a conditional statement.

When you're flying around the screen, you want to be able to go through some maneuvers. To add some realism to your

simulator, you can add speed changes if the plane climbs or dives.

In real flying, if a plane starts to climb and doesn't maintain its speed, it stalls. The opposite is just as bad. If you go into a dive and start going to fast, you may not be able to pull in time. Last but not least, you want to be able to bring the plane down on the runway and have it roll to a stop.

There's nothing that hard here. It's just going to take some planning and experimenting to make it work as much like a real airplane as you want.

Masking the Image

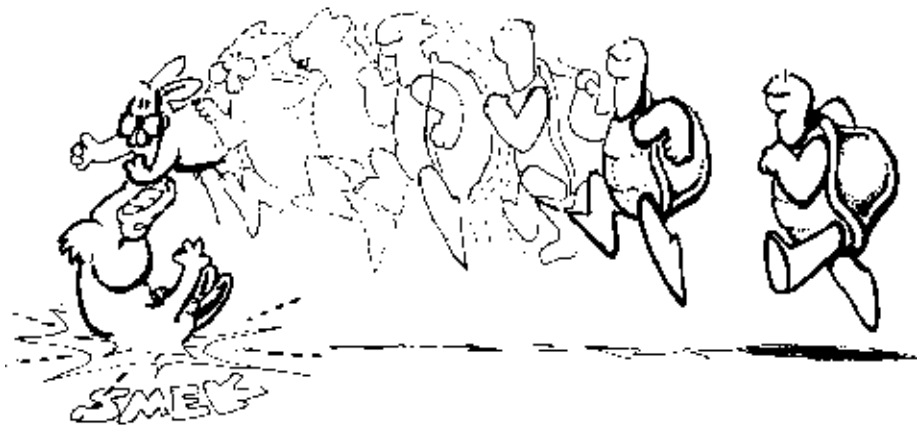
You've got one more problem, however. And it can get a bit tricky to solve it.

The problem is masking.

When you cut an image to the clipboard, you cut a rectangle. The Firefox images all have a picture of the jet on a white background. That's not so bad if you're flying in a white sky. But when you start flying over a multi-colored landscape, that can become a problem.

There's a short introduction to masking on the CD in the \projects\chpt11 subdirectory. This introduces you to the Bitmodes in which MSW Logo operates, and to the BITMODE commands.

Check it out and see what these features can do for you. Read the SPRITE.LGO procedure from George Mills for more information.



You’re getting pretty good at this stuff now; changing shapes and all. So why not try a few other things using your own multiple turtle procedures?
