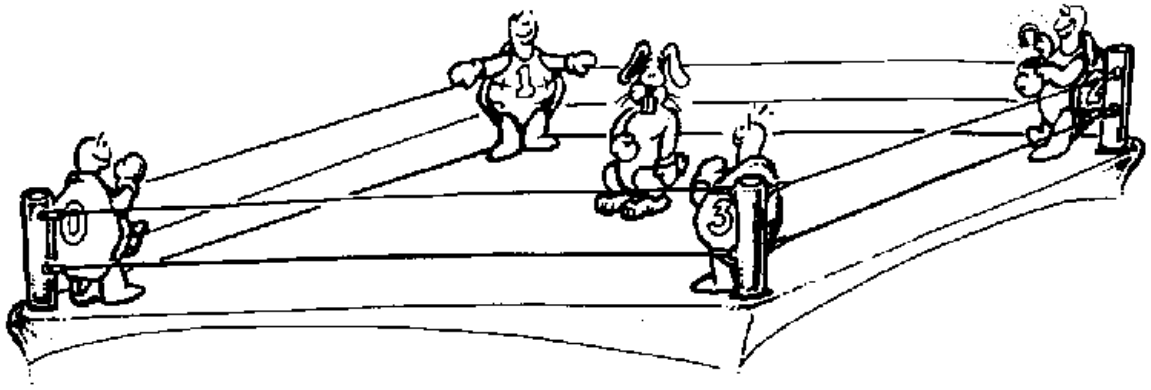
Chapter 10. Multiple Turtles

"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. But before you start getting busy with all those turtles, let's take a look at how you can use just one turtle to simulate multiple turtles. It's a great review of things you've been doing up until now.

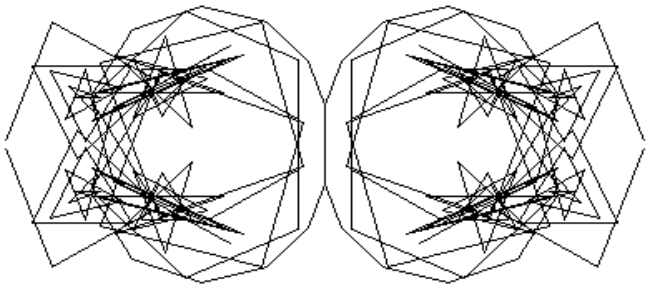
The KALEIDOSCOPE procedure (KALEID.LGO) shows a good use of coordinate, color, and other commands. The resulting picture looks as if it were drawn by multiple turtles.

```
TO KALEID :ANG :CNT
IF :CNT < 1 [STOP]
SETPC COLOR
```

```
MOVE
KALEID :ANG + 5 :CNT - 1
END

TO COLOR
OP (LIST (RANDOM 128)+128 (RANDOM 128)+128
(RANDOM 128)+128)
END

TO MOVE
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

TO SCREEN
OP (LIST RANDOM 100 RANDOM 100 RANDOM
100)
END

TO START
CS HT
SETSC SCREEN
KALEID 0 50
WHATNOW
END
```

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

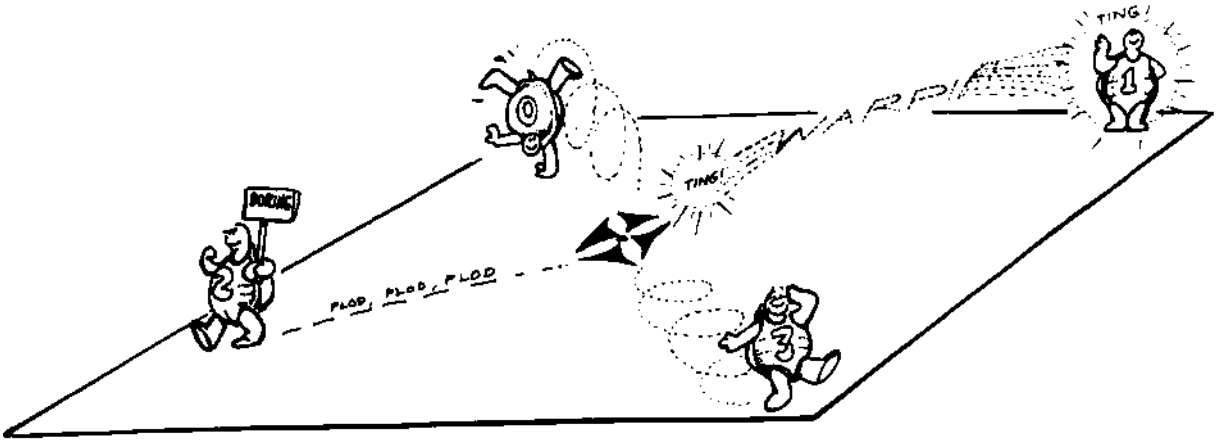
The drawing shown below was made with the KALEID2.LGO procedure. This procedure includes a couple of different ways of simulating multiple turtles. Imagine what this would look like if added different colors to it.



In fact, why not give that a try!

Independent Turtles

Here's another good example of using coordinate commands in a procedure. In the KALEIDOSOPE procedure, the simulated turtles all acted according to plan. In the procedure listed on the next page, 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.

```
TO START
   CT PRINT [Welcome to the illusion of multiple turtles!]
TIMER
   PRINT [This procedure lets you play with four]
   PRINT [turtles: 0, 1, 2, and 3.  Just "ASK"]
   PRINT [them TO do what you want each TO do.]
TIMER
```

```
PRINT [For example...]
PRINT "
PRINT [ASK 0 [REPEAT 3 [FD 100 RT 120]]] TIMER
PRINT "
PRINT [Ready?  OK, give it a try...]
SETUP
END

TO TIMER
WAIT 150 CT
END
TO SETUP
SET.ZERO
SET.ONE
SET.TWO
SET.THREE
END

TO SET.ZERO
MAKE "OLDX0 XCOR
MAKE "OLDY0 YCOR
MAKE "OLDH0 HEADING
END

TO SET.ONE
MAKE "OLDX1 XCOR
MAKE "OLDY1 YCOR
MAKE "OLDH1 HEADING
END

TO SET.TWO
MAKE "OLDX2 XCOR
MAKE "OLDY2 YCOR
MAKE "OLDH2 HEADING
END

TO SET.THREE
```

```
MAKE "OLDX3 XCOR
MAKE "OLDY3 YCOR
MAKE "OLDH3 HEADING
END

TO ASK :TNUM :COMMAND.LIST
IF :TNUM = 0 [ZERO :COMMAND.LIST]
IF :TNUM = 1 [ONE :COMMAND.LIST]
IF :TNUM = 2 [TWO :COMMAND.LIST]
IF :TNUM = 3 [THREE :COMMAND.LIST]
END
TO ZERO :COMMAND.LIST
SETXY :OLDX0 :OLDY0 SETH :OLDH0
PD RUN :COMMAND.LIST PU
MAKE "OLDX0 XCOR
MAKE "OLDY0 YCOR
MAKE "OLDH0 HEADING
END

TO ONE  :COMMAND.LIST
SETXY :OLDX1 :OLDY1 SETH :OLDH1
PD RUN :COMMAND.LIST PU
MAKE "OLDX1 XCOR
MAKE "OLDY1 YCOR
MAKE "OLDH1 HEADING
END

TO TWO :COMMAND.LIST
SETXY :OLDX2 :OLDY2 SETH :OLDH2
PD RUN :COMMAND.LIST PU
MAKE "OLDX2 XCOR
MAKE "OLDY2 YCOR
MAKE "OLDH2 HEADING
END

TO THREE :COMMAND.LIST
SETXY :OLDX3 :OLDY3 SETH :OLDH3
```

```
PD RUN :COMMAND.LIST PU
MAKE "OLDX3 XCOR
MAKE "OLDY3 YCOR
MAKE "OLDH3 HEADING
END
```

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 commands. Or, why not write a simple TELL procedure? In that way, we're all talking the same language.

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

We started this book with a race. So how about a simple turtle 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 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.

A New Target Game

Here's a new target game. You get the chance to hit a moving target.

```
TO ZAP
CT
PR [WELCOME TO THE GAME OF...]
PR [* * * * * ZAP THE TURTLE * * * * *]
PR []
PR [ONE OF THE TURTLES WILL APPEAR ON
THE]
PR [SCREEN. CAN YOU GUESS THE PROPER
  DIRECTION]
PR [AND SPEED TO HIT THE MOVING TURTLE?]
PR []
WAIT 200 CT
PR [AW, C'MON!] WAIT 50
PR [GIVE IT A TRY!] WAIT 50
PR [TO PLAY, PRESS 'Z AND THEN ENTER.]
END
```

```
TO Z
CS CT PU ST
MAKE "ANS1 (RANDOM 500) - (RANDOM 250)
MAKE "ANS2 (RANDOM 200) - (RANDOM 100)
TELL 1 HOME TELL 0 PU
SETPOS [:ANS1 :ANS2] PD
SETH (RANDOM 180) - (RANDOM 45)
IF HEADING < 45 [SETH HEADING + 45]
MAKE "DIR RANDOM 2
IF :DIR = 1 [SETH HEADING - 180]
PR [CAN YOU HIT THE TARGET?]
WAIT 80 CT
AIM
END

TO AIM
REPEAT 3 [TELL 0 FD RANDOM 25]
PR [GUESS THE HEADING TO THE TARGET?]
MAKE "ANS3 READNUMBER
IF NOT NUMBERP :ANS3 ~
  [PR [NOT A NUMBER. TRY AGAIN.]]
TELL 1 SETH :ANS3
PR [GUESS THE SPEED TO INTERSECT THE ~
  TARGET?]
MAKE "ANS4 READNUMBER
WRAP SHOOT
END

TO SHOOT
TELL 0 FD RANDOM 10
TELL 1 IF OR XCOR > ABS 450 YCOR > ABS 150 ~
  [OOPS Z]
MAKE "ANS1 XCOR
MAKE "ANS2 YCOR
TELL 1 FD :ANS4
MAKE "ANS5 XCOR
MAKE "ANS6 YCOR
```

```
IF OR :ANS5 < (:ANS1 - 10) :ANS5 > (:ANS1 + 10) ~
  [SHOOT]
IF OR :ANS6 < :ANS2 - 10 :ANS6 > (:ANS2 + 10) ~
  [SHOOT]
CHEERS
END

TO READNUMBER
OUTPUT FIRST RL
END

TO CHEERS
CT
REPEAT 5 [PR [CONGRATULATIONS!]]
WAIT 100 Z
END

TO OOPS
CT
PR [SORRY! TRY AGAIN. ]
WAIT 80
END
```

This is just a 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.

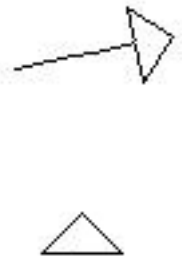
How Does It Work

Now, let's take this procedure apart to 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.

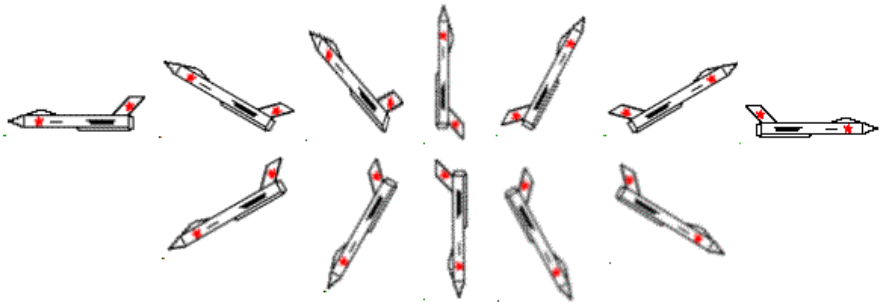
- What can you do to make it easier?
- How would you make it harder?
- What would you do to make the CHEERS procedure a bit flashier. Printing Congratulations is a bit dull.

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

Changing the Shape of the Turtle

What about changing the shape of the turtle? You could have two aircraft in an aerial duel.

Do you remember the FIREFOX procedure? You might want to run it from the diskette that came with the Sourcebook if you don't remember. In the PCX 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, ENE.BMP for East North East, and so on.

There's also another one called HIT.BMP. You can use this one for when you Zap the Firefox.

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.

Keyboard Control

To get you started, here's a simple procedure for doodling on the screen. It might give you some ideas for flying your Firefox around the screen.

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

TO COMMAND
MAKE "KEY CHAR KEYBOARDVALUE
IF :KEU = "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.

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.

DOODLE1 doesn't care if the Caps Lock key is on. The numbers in the list (in the brackets) are the upper and lower case 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 DOODLE1
ICON "COMMANDER
SETFOCUS [ MSWLOGO SCREEN ]
KEYBOARDON [ COMMAND KEYBOARDVALUE ]
END

TO QUIT
KEYBOARDOFF
UNICON "COMMANDER
END
```

KEYBOARDOFF 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.

Auto Racing

If you don't like jet planes, here are a few other examples of bitmaps you can use as turtles. You can edit the procedures to make them bigger or smaller depending on how you are going to use them.

```
TO CAR
CS HT WHEEL RWING BODY
WHEEL FD 20 WHEEL PU SETX XCOR - 15 PD
WHEEL FD 10 FWING COCKPIT
END

TO BODY
FD 5 RT 90 FD 5 LT 90
REPEAT 2 [FD 30 RT 90 FD 10 RT 90]
RT 90 FD 10 LT 90
END
```

```
TO COCKPIT
PU SETX XCOR + 8 SETY YCOR - 24 PD
REPEAT 2 [FD 12 RT 90 FD 4 RT 90]
FILLUP
END

TO FILLUP
PU FD 2 RT 90 FD 2 PD
SETFC [000 000 000] FILL
BK 2 LT 90 BK 2
END

TO FWING
REPEAT 2 [FD 5 RT 90 FD 20 RT 90]
END

TO RWING
BK 5
REPEAT 2 [FD 5 RT 90 FD 20 RT 90]
END

TO WHEEL
REPEAT 2 [FD 10 RT 90 FD 5 RT 90]
FILLUP
END
```

Morf's Biplane

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 file that you'll find in the PCX directory, in case you want to use it as a turtle shape.

```
TO PLANE
SETPENSIZE [2 2]
FD 30 LT 90 FD 15 LT 60 FD 20 RT 60 FD 40
RT 90 FD 5 LT 135 FD 6 RT 45 FD 20 LT 90 FD 15
LT 90 FD 90 LT 135 PU FD 15 PD
```

```
SETFC [0 0 0] FILL PU HOME PD
GEAR
WING
PROP
END

TO GEAR
SETPENSIZE [3 3]
PU LT 90 FD 70 LT 90 PD FD 20 RT 90
REPEAT 12 [FD 2 RT 36]
PU HOME PD
END

TO PROP
PU SETPOS [-95 -7] SETH 0 PD
FD 25 PU HOME PD SETPENSIZE [1 1]
END

TO WING
PU SETPOS [-45 25] PD
SETPENSIZE [5 5] LT 90 FD 30
PU HOME PD
END
```

You've already seen how MSW Logo allows you to cut a portion of the screen and hold it temporarily on the Windows clipboard. The BITCUT command cuts an area that you define and saves it as a bitmap image.

1. Run the CAR procedure.
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
4. Type BITMAPTURTLE

Now you can drive your race car around the screen.

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.

Each of the 1024 available turtles can have its own shape, its own heading and position. However, there can only be one pen color for all the turtles.

Use SETTURTLE to talk to an individual turtle. Or use the TELL :TNUM procedure. Use CLEARSCREEN or NOBITMAPTURTLE to change Ernestine back to her regular shape.

There is one noticeable problem, however. You're going to run into trouble when you want to turn. The picture remains the same. Try it.

FD 1OO RT 90 FD 100

Looks like the car went into a 100 step skid sideways.

To show the car 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.

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.

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
MAKE "WIN TURTLE
(PR "TURTLE :WIN [IS THE WINNER!])
MAKE "ANS YESNOBOX [RACE] ~
  [WANT TO RACE AGAIN?]
```

```
IFELSE :ANS = "TRUE [START] ~
  [CS CT PR [SEE 'YA AT THE RACES.]]
STOP
END
```

This is quite a bit different than 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. You can with the :WIN variable and then use the variable to announce the winner.

```
MAKE "WIN TURTLE
(PR "TURTLE :WIN [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 procedure. The next line says RACE. So you start another race with Ernestine and her cousins racing up a single path.

Why Ernestine? Why not four cars?

In the FLAG procedure, when you left-click on NO in the RACE 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.

Creating Bitmapped Turtles

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. 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.

```
[
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.

Turtles With Different Shapes

"But what if I want each turtle to have a different shape?"

No problem! Let's say you want to use the twelve Firefox bitmaps, or different colored race cars, or different hot air balloons. The process is the same:

```
SETBITINDEX index
BITLOAD "<new file>.BMP
BITCUT <width><height>
SETTURTLE number
BITMAPTURTLE
```

Whether you use this sequence of commands in a separate procedure for each shape or include the sequence in one

procedure is up to you. However, the sequence remains the same.

Why not give it a try?

Revise the ZAP game to use the twelve different pictures of the Firefox. Rather than be quite so random, have Logo select one of the 12 directions the jet is facing.

You may remember the DOODLE procedure. Now try this:

1. Make each one of the bitmaps into a turtle.
2. Change the COMMAND procedure so that each time you turn the turtle right or left, a procedure is called that checks the turtle's heading. Then the correct turtle is used until you change directions again. For example, you might add a CHECK procedure.

```
TO CHECK
IF HEADING = 30 [SETTURTLE 1
IF HEADING = 60 [SETTURTLE 2]
IF HEADING = 90 [SETTURTLE 3]
...and so on until
IF HEADING = 0 [SETTURTLE 0]
END
```

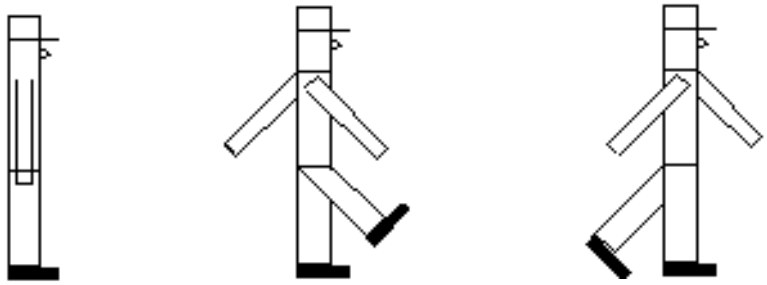
3. You would then change the COMMAND procedure to something like this:

```
IF :KEY = "R [RIGHT 30 CHECK STOP]
IF :KEY = "L [LEFT 30 CHECK STOP]
```

Why not see what you can do with this now.

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 seem to walk.

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

Whoever saw a marching turtle?

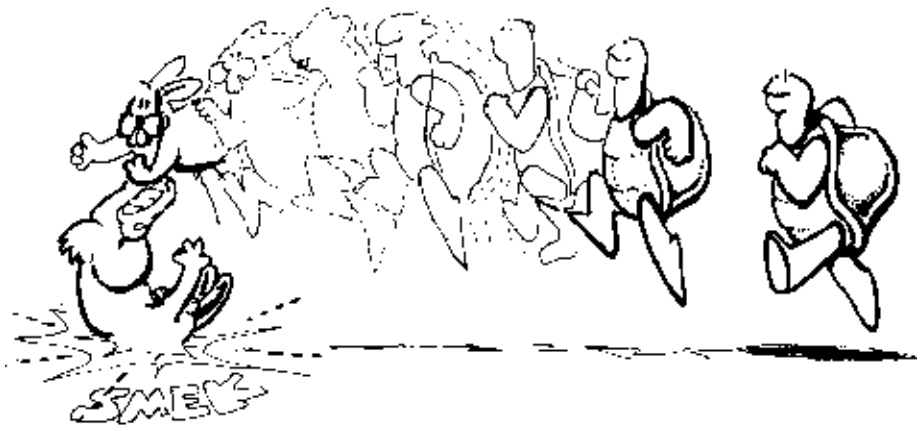
Well, here's a very simple start that you can build upon. It uses LWALK.BMP and RWALK.BMP to march across the screen.

```
TO MARCH
LOCAL "X
MAKE "X POS HT
SETTURTLE 1
ST SETPOS :X
WAIT 20 FD 50
MAKE "X POS HT
SETTURTLE 2
ST SETPOS :X
WAIT 20 FD 50
MAKE "X POS HT
END
```

```
TO START
CS CT
SETBITINDEX 0
BITLOAD "STAND.BMP
BITCUT 30 110
SETTURTLE 0
BITMAPTURTLE
PU SETH 90
SETBITINDEX 1
BITLOAD "RWALK.BMP
BITCUT 80 110
SETTURTLE 1
BITMAPTURTLE
PU SETH 90
SETBITINDEX 2
BITLOAD "LWALK.BMP
BITCUT 80 110
SETTURTLE 2
BITMAPTURTLE
PU SETH 90
SETTURTLE 0
WAIT 100
CT PR [FORWARD, MARCH!]
REPEAT 10 [MARCH]
END
```

The START 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 tell the next turtle where it is so that the next turtle 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.



You're getting pretty good at this stuff now. So why not try a few other things using your own multiple turtle procedures? You'll find some more ideas in the next chapter.
