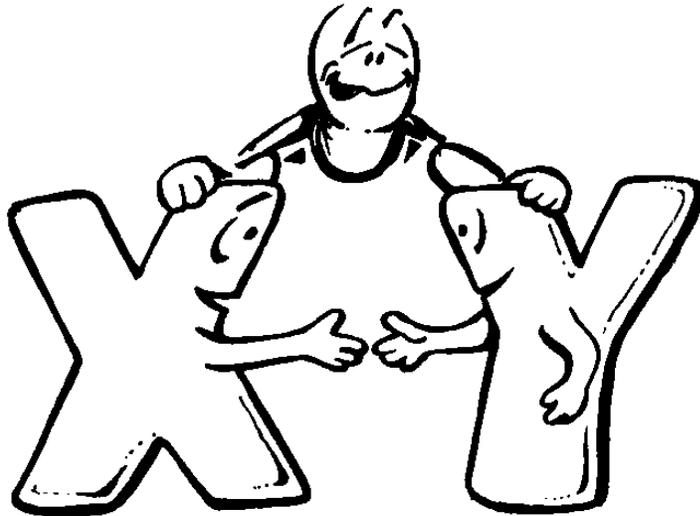


Chapter 8. Turtle Positions and Coordinates

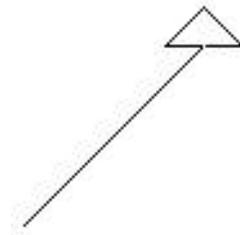
So far, we've moved the turtle around the screen using the direction commands: FORWARD, BACK, LEFT, RIGHT. Well, there are two other ways to move the turtle around that give you all sorts of new things to do. These new choices are turtle headings and X-Y coordinates.



Turtle Headings

When you want the turtle to move from HOME up to the right, you can tell her go RIGHT 45 FD 100 LEFT 45.

Another way to do the same thing is to tell the turtle to set her heading to 45 and then go forward 100 steps and then set her heading back toward the top of the screen.



Turtle Positions and Coordinates

```
SETHEADING 45 FD 100 SETHEADING 0
```

Or you can use shorthand. Clear the screen and type

```
SETH 45 FD 100 SETH 0
```

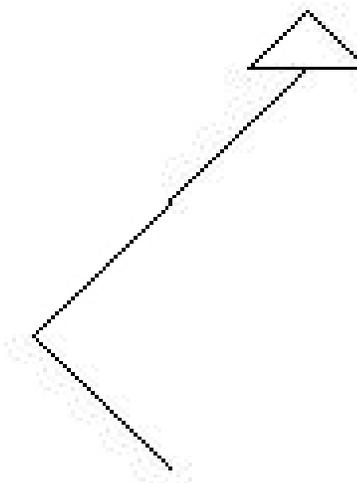
When you typed `SETH 45`, what direction did the turtle turn? She turned the same direction as when you told her `RIGHT 45`, correct?

If you typed `SETH -45 FD 100 SETH 0`, what do you think would happen? Go ahead — try it and see. Remember to clear the screen first.

OK, now type `SETH 45 FD 100 SETH 0` again, but without clearing the screen. What happened?

Hmmmm? This is interesting. Now type `SETH 45 FD 100 SETH 0` again. What happened this time?

Well, it seems as if you can use the `SETH` command to turn the turtle. But it certainly doesn't act like the `RIGHT` and `LEFT` commands.



SHOW HEADING

If you ever want to know what heading the turtle is on, type

```
SHOW HEADING
```

Turtle Positions and Coordinates

HEADING is a new command. You can easily guess what it means.

SHOW, you may remember, is a command that is a lot like PRINT. The big difference is that PRINT shows lists of things without brackets. SHOW displays them with brackets.

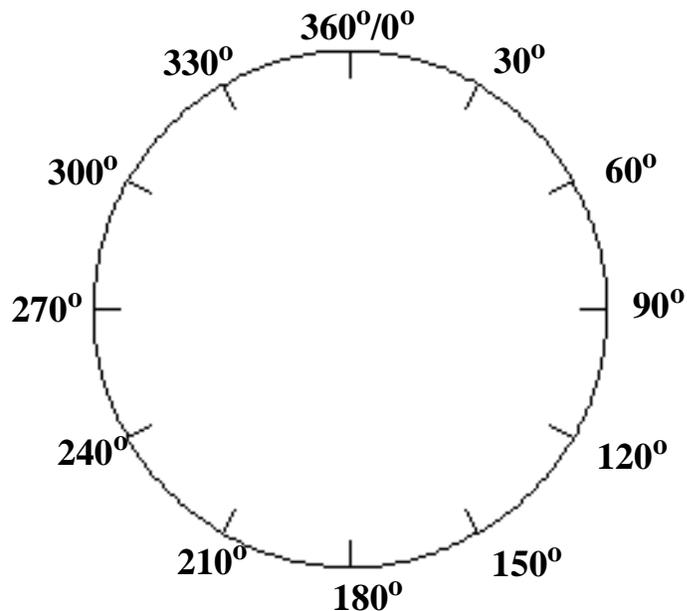
```
PRINT [1 2 3 4 5]
```

The result is 1 2 3 4 5

```
SHOW [1 2 3 4 5]
```

The result is [1 2 3 4 5]

OK? Back to HEADINGS. Remember the string clock you made a while ago, the one you made with a piece of board and some nails? This is a good tool for learning about headings. Well, let's look at that clock again.



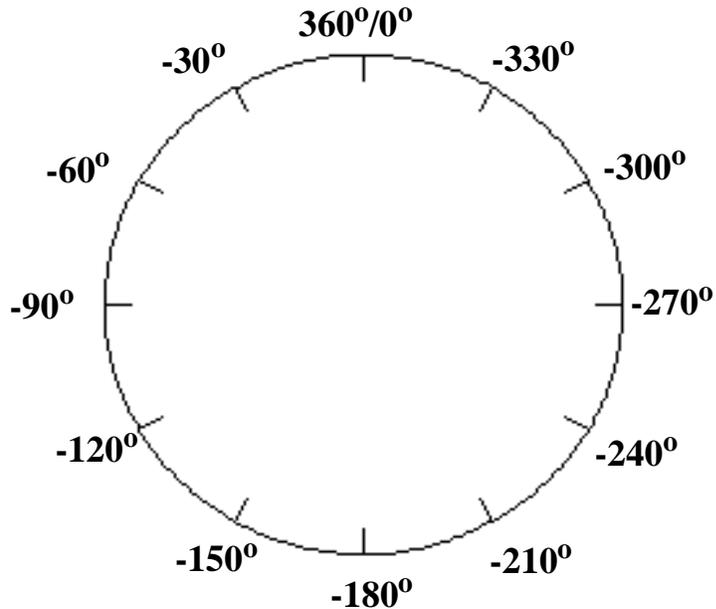
Turtle Positions and Coordinates

What heading is at 12:00 o'clock? 3:00 o'clock? 6:00 o'clock? 9:00 o'clock?

Get the idea? Turtle headings follow the hands of the clock. The heading at 12:00 o'clock is zero. At 3:00 o'clock, it's 90. At 6:00 o'clock, it's 180, and at 9:00 o'clock, it's 270.

This isn't to say that you can't move counter clockwise, to the left. You've already seen what happens when you typed

SETH -45.



What time would it be at SETH -90?

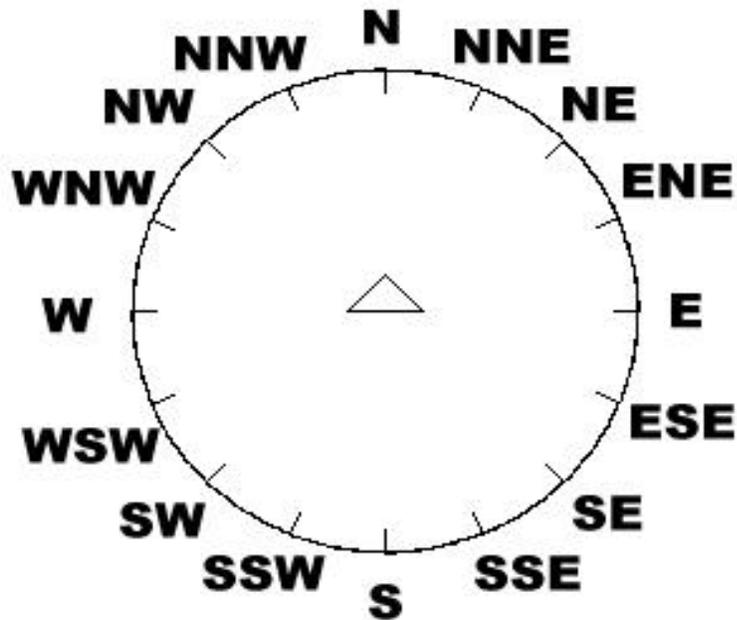
What about SETH -180? SETH -270?

The Turtle's Compass

When you look at a clock like this, with all the hours marked off in degrees, what does it make you think of?

How about a compass?

Yes, that's it. It's like a compass. Now, what do you think you can learn from a compass?



The compass needle always points to North. If North is at 12:00 o'clock or 360 degrees, where's East? South? West?

Now let's get tricky. Where's Northeast? How many degrees is that? What about Southwest?

Morf's got a great game that will help you out. You'll learn about the procedures to draw this marked-off circle a little later in this chapter.

Rabbit Trail 21. The Turtle Rally



Have you ever heard of a car rally?

Well, car rallies are special types of races. Only this type of race doesn't depend on how fast you can drive. It depends on how well you can follow directions. Each car has a driver and a navigator. The team that can find its way through the course closest to the time limit is the winner.

In the Turtle Rally, you are given a compass and a list of directions. You'll need a digital watch that shows the seconds or a watch with a second hand so you can time yourself.

Here's how to play.

Have one person set up the course, maybe your teacher, your mom, or a person elected to be Rally Captain. This can be a course through the house, through the neighborhood, around the school yard, or anywhere you can set up a safe course to follow.

The directions will read something like this, only there should be many more.

1. You have 15 minutes to complete the race.
2. Start from Home. Set your heading to 0 and go forward for 40 seconds.
3. Set your heading to 90 and walk forward 120 steps.
4. Set your heading to 30 and run forward for 10 seconds.

You can make up any kind of directions you want. They can be simple. They can be silly. Or they can be tough. It makes the race more fun when you mix them up: walking, running, turning — and don't forget Back!

Turtle Positions and Coordinates

Another thing you can do is make a map for each team. Only maybe you leave off some names to make it tough. Maybe you change the names to make it silly.

Use compass directions, such as North, Southwest, Northeast. Maybe you can mix up compass directions with degrees and headings. There are all sorts of things you can do to get practice. And you can have a fun rally at the same time.

The Rally Captain times each team — the time each team starts and the time each team finishes. Each team starts a minute or two apart so that teams don't bump into each other along the path.

It is a good thing to have checkpoints along the path where teams have to pick up something like a piece of paper and return it to the finish line. Maybe the paper can have a question each player has to answer correctly. Use your imaginations. The main thing is to have some fun while practicing with headings and directions.

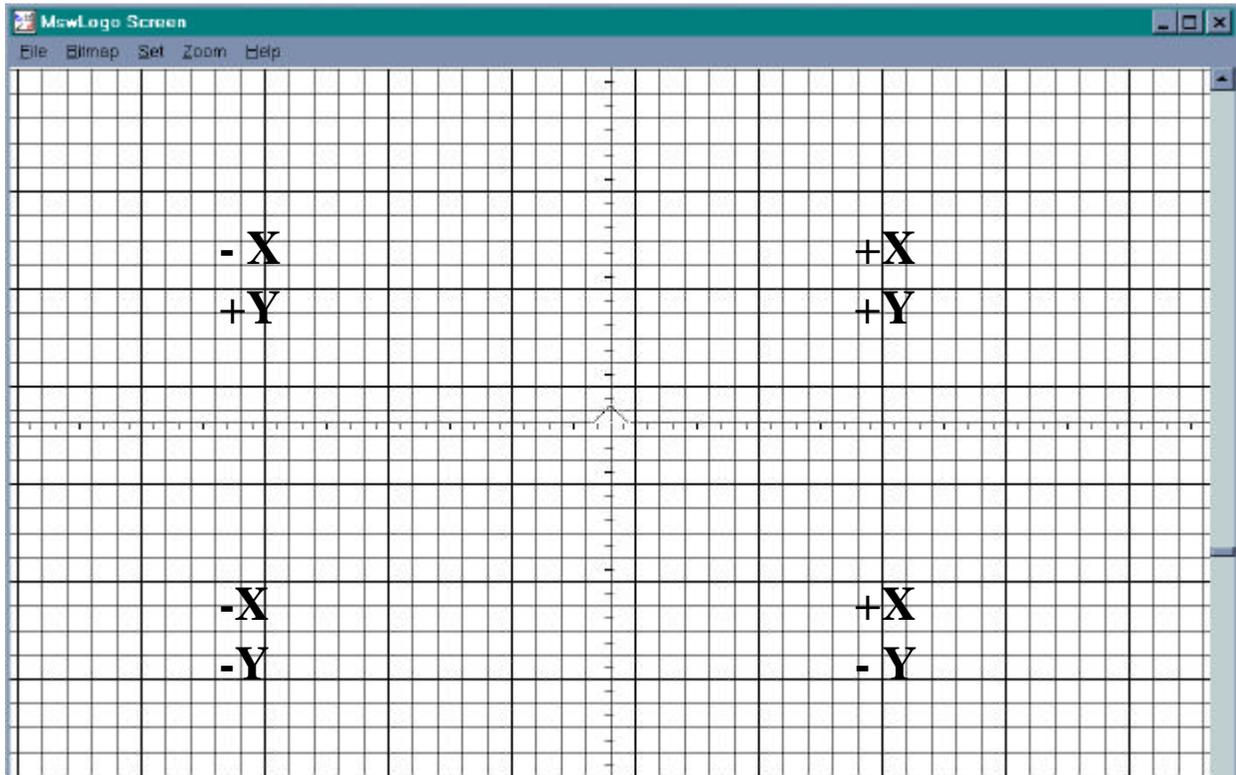
After you've run your Turtle Rally, why not try one on the computer? You'd be surprised what you can create using headings and directions. In fact, that's just what you'll do a little later in this chapter.

X - Y Coordinates

Imagine this screen with 1000 columns drawn on it and 1000 rows. All of these columns and rows may not show on your computer screen but they are there. You can change the numbers if you want. Read the instructions in the

Turtle Positions and Coordinates

RELEASE.TXT file that was installed when you installed in the **mswlogo** directory when you installed MSW Logo.



Columns go up and down. Rows go from side to side. To find a position, you'd have to know which column and which row to pick.

This is exactly what you do when using the X- and Y-coordinates. X-coordinates count the number of columns from side-to-side. Y-coordinates count the number of rows up and down.

There are two lines that divide the screen in half, one divides the screen from top to bottom and one divides the screen down the middle, from side to side. Where these lines meet is HOME, where X is 0 and Y is 0.

If you move up to the right, both the X-coordinate numbers and Y-coordinate numbers are positive (+).

If you move up to the left, the X-coordinate becomes negative (-) but the Y-coordinate remains positive.

If you move down to the left, both the X- and Y-coordinates become negative.

If you move down to the right, the X-coordinate is positive and the Y-coordinate is negative.

If you'd like to put the same type of grid on your computer screen, use the GRID procedure (GRID.LGO on the CD that came with this book). Make sure you maximize the graphics screen to run this. Left-click on the button at the far left of the Graphics screen title bar. Then left-click on "Maximize." You can also use the command

ICON "COMMANDER

This reduces the Commander windows to a small icon.

XCOR and YCOR

HOME is at X-coordinate 0 and Y-coordinate 0. In Logo, we use the shorthand names:

XCOR

YCOR

Clear the screen. Then type SHOW XCOR. What happened?

Hmmmmm. Very interesting.

What do you think you need to do to send the turtle to another X-coordinate? Give yourself a Gold Star if you said SETX, not SET XCOR.

Go ahead, try it. Type SETX and a number.

Turtle Positions and Coordinates

Now try moving the turtle up and down. What's the command? That one should be real easy.

I bet you've already guessed that you can combine these two commands. Sure!

SETXY _____

That's right. It's SETXY.

Now, where were we?

Was the pen down when you typed the SETX, SETY, and SETXY commands? What happened? Did you end up with a crazy drawing on the screen?

If the pen was up, put it down and then do a few more SETXY commands to get some practice using the commands.

SETPOS

SETPOS is like SETXY except that it takes a list of x- and y-coordinates as inputs. It moves the turtle to the coordinates you put in the list.

SETPOS [_____] or

SETPOS LIST :X :Y

POS and TOWARDS

There are times when you want to remember the turtle's position while she goes off and does something else. That's where POS comes in. This command remembers the

Turtle Positions and Coordinates

coordinates of where the turtle was when you issued the command. For example:

```
TO S
SETXY -100 50 SETH 0 FD 25 LT 90 FD 15
MAKE "A POS
RT 90 FD 200 SETH 135 FD 25
MAKE "B POS
PU SETPOS :A PD
SETH TOWARDS :B
FD DIST :A :B
END
```

```
TO DIST :A :B
OP DIST1 (FIRST :A) - (FIRST :B)~
      (LAST :A) - (LAST :B)
END
```

```
TO DIST1 :DX :DY
OP SQRT (:DX * :DX ) + (:DY * :DY )
END
```

The first and third lines of the S procedure are whatever you want them to be. The idea is to make the turtle off somewhere, make the variable A the POS or coordinates of the final location, and then go off to another location, which is named B.

PU SETPOS :A PD takes the turtle back to the coordinates of :A.

SETH TOWARDS :B tells the turtle to turn towards the coordinates that were remembered as position :B.

Turtle Positions and Coordinates

Finally, the procedure tells the turtle to go FD the DISTance to :B. You know that :A and :B are lists of coordinates. Then you should be able to figure out what this line does:

```
OP DIST1 (FIRST :A) - (FIRST :B)~  
        (LAST :A) - (LAST :B)
```

FIRST :A and FIRST :B are the first elements of :A and :B or the X-coordinates. So what's LAST :A and :B?

Remember these examples. POS, TOWARDS, and the DIST procedures can come in real handy.

Drawing an Ellipse

“Do you remember back in Chapter 3 when you were talking about that vertical axis thing?” Morf asked. “The axis was going through what looked like a circle on its side, sort of like it was flattened a bit. Can you show us more about that now?”

“Sure. I never really thought about it that way, but an ellipse does look like a flattened circle.”



To get a more accurate picture of an ellipse, take a look at the Ellipse procedures in the Logolib directory that was setup when MSW Logo was installed. That will give you an idea of how complex ellipses are.

First of all, if you really want to know what the ELLIPSA2 procedure is doing, you have to know an ellipse is.

An ellipse is a closed plane curve (something like a circle) that is made by a point moving in such a way that the sum of its distances from two fixed points is a constant.

“I knew I wasn’t going to understand it.” said Morf. “But what the heck. I can still use it to draw things.”

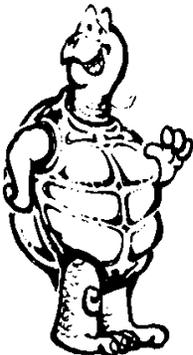
"No, now wait a minute. Ellipses aren’t that tough. Do you remember when we talked about a circle being a closed curve where all points on that curve were the same distance from the center?

"Do you remember when you drew the circle on the floor? You stretched a piece of string from the center and moved the chalk around the center. The length of the string remained the same, didn’t it?

"That means it remained constant.

"Well, now it’s my turn to give you a Turtle Trail to follow off the computer. It will help you make sense out of ellipses and that ellipse procedures.

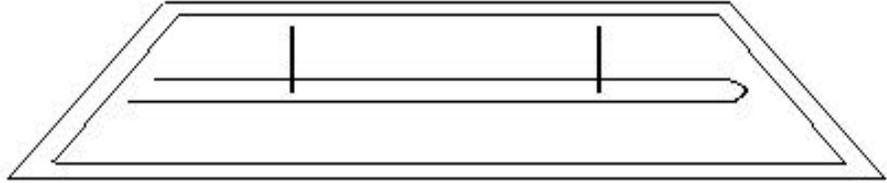
Logy’s Turtle Trail: Drawing an Ellipse



Find yourself a flat surface in which you can stick some tacks or pins. Of course, you have to find the tacks or pins, too. Small nails will also do just fine. You can use a wide piece of shelf board, a piece of plywood, or you can even use the side of a corrugated box.

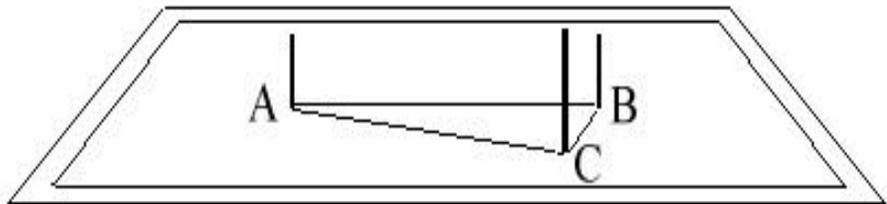
Turtle Positions and Coordinates

Now tape a piece of paper on the flat surface. Stick the pins in the surface so you end up with something looking like this.



Now get yourself a piece of string about as long as the one shown in the picture above. Tie the two ends together to make a loop. Drop the loop over the two pins. Then put the tip of your pencil inside the loop and stretch it outward so that the loop of string is tight.

It should look something like this.



The two pins are at points A and B. That's your pencil at point C. Now move the pencil around the two pins keeping the string tight. What's the picture look like?

Looks like an ellipse, doesn't it?

It should. Because that's exactly what you've drawn. Go back and check the definition again to be sure. To draw a different shape of ellipse, just move the pins.

An ellipse is a curve that is made by a point — the point of your pencil — moving in such a way that the sum of its

distances from two fixed points — the two tacks or pins — is a constant. That's another way of saying the sum of its distances remains the same.

Check it out. The pins are the two fixed points. So the distance from A to B is always the same, right?

The string doesn't change its length, does it? So the sum of the two distances, AC and CB, must also remain the same. The individual distances will change as the pencil moves around the pins. But the sum of the distances will always be the same.

Make sense? Sure it does. You just proved it.

The ELLIPSE Procedure

Now take a look at this ELLIPSE procedure.

```
TO ELLIPSE :A :B
  ELL - :A :A :B 1
END
```

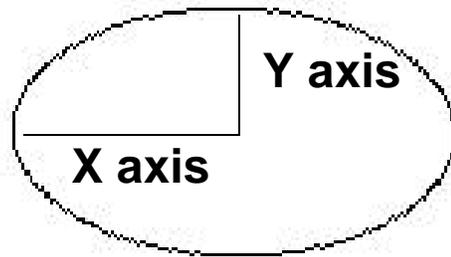
```
TO ELL :X :A :B :INC
  IF (:X * :X) > (:A * :A) [STOP]
  IF :X = :A [MAKE "INC -1]
  SETPOS LIST :X :INC * :B * SQRT (1 - (:X * :X) ~
    / (:A * :A))
  ELL :X + :INC :A :B :INC
END
```

This procedure draws an ellipse based on the position of the turtle. To run it, you enter the horizontal radius or X-coordinate of the ellipse, and the vertical radius or Y-coordinate. For example, try

```
ELLIPSE 100 40
```

Turtle Positions and Coordinates

The turtle starts from the center of the ellipse, moves along the X-axis to the edge and then heads for the Y-axis, drawing an ellipse as it goes.



The ELLIPSE procedure is on the CD that came with this book, along with a second ELLIPSE2 procedure that draws an ellipse starting from the edge. If you want to see just how these procedures work, you can do some editing and test them from the MSW Logo Editor.

Here's the other procedure.

```
TO ELLIPSE2 :S :E
MAKE "N 0
REPEAT 120 [RIGHT :N FD :S LT :N LT :N ~
  FD :S * :E RT :N MAKE "N :N + 3]
END
```

Now let's see how these work. We'll start with this one since it's the easiest.

1. Load ELLIPSE2.LGO from the CD that came with this book.
2. Type **EDALL** in the Input Box.

ELLIPSE2.LGO is displayed in the Editor Window.

3. Drag your mouse over **REPEAT 120 [** and press **Delete**.
REPEAT 120 [is deleted.
4. Delete the closing bracket also.
5. Since you're not going to run the procedure, the variables

don't get entered. So you'll have to enter them separately.

Type

MAKE "S 2

MAKE "E 4

MAKE "N 0

6. Left-click in the Editor window title bar and drag it out of the way so that you can watch the turtle.
7. Highlight the line in the procedure that begins, "RIGHT :N."
8. Left-click on **Test** on the Editor to start the procedure. At this point, the procedure should be easy to follow. Just keep on clicking on Test to watch the Ellipse being drawn.

You can also left-click on any one or more commands in that line to watch each command be carried out. The main thing is to watch each line. Think about what happens to each variable as you left-click on Test.
9. To stop the procedure or to try different variables, simply type CS in the Input Box.

The screen is cleared and you're ready to start again.

Type a negative number as the :E variable. What happens? Why does the turtle act that way? Carefully trace the procedure using Test to see how and why the procedure works the way it does.

Now for the other ELLIPSE procedure.

```
TO ELLIPSE :A :B
```

```
ELL -:A :A :B 1
```

```
END
```

```
TO ELL :X :A :B :INC
```

```
IF (:X * :X) > (:A * :A) [STOP]
```

```
IF :X = :A [MAKE "INC -1]
```

Turtle Positions and Coordinates

```
SETPOS LIST :X :INC * :B * SQRT ~
  (1 - (:X * :X) / (:A * :A))
ELL :X + :INC :A :B :INC
END
```

This time we're going to do just about the same thing.

1. Load the ELLIPSE procedure.
2. Type EDALL to display the ELLIPSE procedure in the Editor window.
3. You have four variables to enter: X, A, B, and Inc.

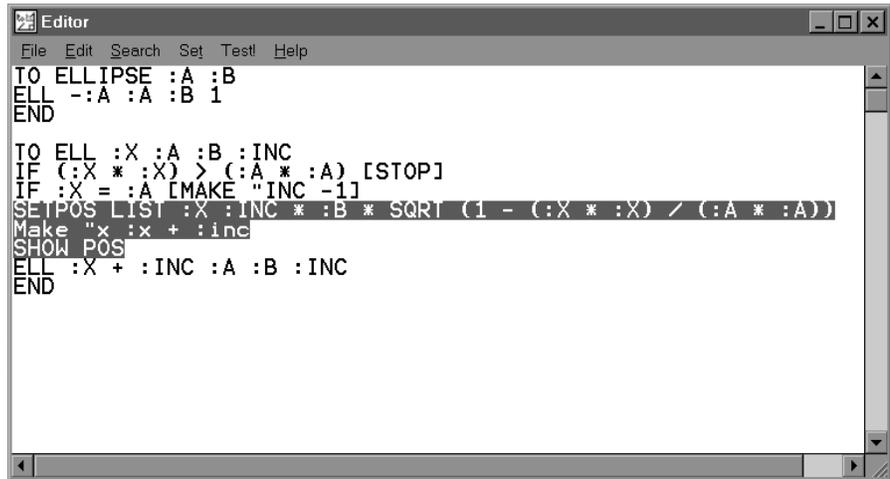
```
MAKE "X -100
MAKE "A 100
MAKE "B 40
MAKE "INC 1
```

To check out this procedure properly, there's a couple of lines you need to add to the ELL procedure.

```
TO ELL :X :A :B :INC
IF (:X * :X) > (:A * :A) [STOP]
IF :X = :A [MAKE "INC -1]
SETPOS LIST :X :INC * :B * SQRT~
  (1 - (:X * :X) / (:A * :A))
MAKE "X :X + :INC
SHOW POS
ELL :X + :INC :A :B :INC
END
```

Now highlight the three lines that begin with SETPOS, MAKE "X, and SHOW.

Then left-click on Test and watch the turtle move.



```
Editor
File Edit Search Set Test Help
TO ELLIPSE :A :B
ELL -:A :A :B 1
END

TO ELL :X :A :B :INC
IF (:X * :X) > (:A * :A) [STOP]
IF :X = :A [MAKE "INC -1]
SETPOS LIST :X :INC * :B * SQRT (1 - (:X * :X) / (:A * :A))
Make "x :x + :inc
SHOW POS
ELL :X + :INC :A :B :INC
END
```

While you can't break the procedure apart like you could with the previous one, you can display the changes in the variables as well as the changing coordinates.

Lessons Learned

“Gee, I think I just learned everything I never really wanted to know about ellipses,” Morf commented.

“Yes, but look at the other lessons you learned along the way. You saw different ways to make ellipses including the use of the SETPOS command. You also got some practice breaking complex procedures apart and editing them to see how they work. That's something you can use in many situations.

“Now that's something worthwhile, isn't it?”

More Circles and Arcs

Remember the marked off circles you saw at the beginning of this chapter? Well, here are the procedures used to create those circles. They may look a bit familiar.

Turtle Positions and Coordinates

```
TO ARCL :CENTER :RADIUS :ANGLE
LOCAL "AMT
MAKE "AMT :RADIUS * PI / 180
PU SETXY :CENTER LT 90 FD :RADIUS LT 90 PD
REPEAT :ANGLE [FD :AMT LT 1]
END
```

```
TO ARCR :CENTER :RADIUS :ANGLE
LOCAL "AMT
MAKE "AMT :RADIUS * PI / 180
PU SETPOS :CENTER LT 90 FD :RADIUS RT 90 PD
REPEAT :ANGLE [FD :AMT RT 1]
END
```

```
TO CIRCLEM :CENTER :RADIUS :ANGLE
REPEAT 360 / :ANGLE [ARCR :CENTER :RADIUS
:ANGLE MARK]
PU SETPOS :CENTER PD
END
```

```
TO MARK
RT 90 FD :RADIUS/10
BK :RADIUS/10 LT 90
END
```

CIRCLEM draws a circle as a series of arcs with a mark between the arcs. To draw the individual arcs that are between the hours on the clock, type:

```
ARCR [100 20] 100 30
```

In this procedure, the variable :CENTER is a list representing the X- and Y-coordinates. The turtle moves to the X-coordinate of 100 and the Y-coordinate of 20. It then turns and moves to the edge of the arc and draws it one degree at a time.

Turtle Positions and Coordinates

This is a bit different from the circles you did before. With those, you simply started at the edge and drew a circle. You had to tell the turtle how far to go forward and how many degrees to turn.

In these new circle and arc procedures, you tell the turtle where the center of the circle is to be and what the radius is to be. You know what a radius is, don't you? It's the distance from the center of a circle to the edge. It's one-half the diameter of the circle. The diameter stretches from edge to edge through the middle of the circle.

What's that :AMT variable?

It's just like the variable you used in the arc and circle procedures in the last chapter. That's part of the mathematical formula for drawing a circle. It calculates the size of the circle based on the size of the radius you give it.

```
TO CIRCLE :CENTER :RADIUS
LOCAL "AMT
MAKE "AMT :RADIUS * PI/180
PU SETPOS :CENTER
SETX XCOR - :RADIUS SETH 0 PD
REPEAT 360 [FD :AMT RT 1]
PU SETPOS :CENTER PD
END
```

What does this line mean?

```
MAKE "AMT :RADIUS * PI/180
```

Change the PI / 180 to PI / 360. What happens? Do you remember the relationship between the diameter of the circle and PI?

Turtle Positions and Coordinates

Diameter * PI = Circumference

So, maybe `:RADIUS * PI / 180` will make more sense if we write it as:

`(:RADIUS * 2) * PI / 360`

Radius * 2 equals the diameter, right? Then we have the diameter * PI that equals the distance around the circle. That's divided by 360. How does this fit in the command:

`REPEAT 360 [FD :AMT RT 1]`

That makes `:AMT` equal to 1. You end up with the more familiar command for drawing a circle:

`REPEAT 360 [FD 1 RT 1]`

Awfully simple? Or simply awful?

Go ahead. Play with these procedures a bit. Make sure you understand how they work. Then we'll get creative.

Flower Gardens

Here's a picture that Olga created and sent to Logy and Morf. It's on the CD that came with this book as OLGA.LGO.



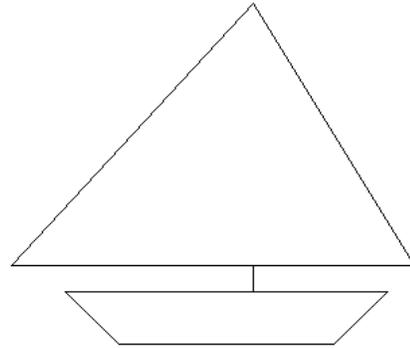
Some of the other procedures Logy and Morf received showed butterflies and birds flying around.

Creative Coordinates

You saw how coordinates can be used to create pictures in the GRID procedure. Here's a simple procedure that uses the SETPOS command.

Turtle Positions and Coordinates

```
TO SAILBOAT
CS PU SETPOS [ 20 -40] PD
SETPOS [-160 -40] SETPOS [ 20 160]
SETPOS [ 140 -40] SETPOS [ 20 -40]
PU SETPOS [ -120 -60] PD
SETPOS [ -80 -100]
SETPOS [ 80 -100]
SETPOS [ 120 -60]
SETPOS [ 20 -60]
SETPOS [ 20 -40]
SETPOS [ 20 -60]
SETPOS [ -120 -60]
END
```



A Coordinate Challenge

Now, how about a challenge. Can you identify what the WHAT.LGO procedure will draw. It's on the CD that came with this book. START gets you started. WHAT1, WHAT2, and WHAT3 are simply subprocedures to break a large procedure into pieces.

Here's a hint — it's a circus animal. Want to make a guess as to what type of animal?

Erase your workspace and then load WHATISIT.LGO. What do you think this procedure will draw? You deserve First Prize if you said a row of circus animals. To draw our row of animals, we use variables with the SETXY command.

But, wait a minute. The X-coordinate is the only variable. Why?

The reason for this is that the X-coordinate is the only coordinate that changes as each animal is drawn. The Y-coordinates stay the same for each animal drawn.

Remember, Y-coordinates are the lines going up and down. If you changed them, some of the animals might end up walking in mid-air. And that would sure look funny!

A Target Game

Do you want to have some fun practicing with X- and Y-coordinates? Then try this target game. It's TARGET.LGO.

The turtle draws a small target on the screen. You are then asked to guess the X- and Y-coordinates of the target. If you guess correctly, the CHEERS procedure runs. If you don't guess right, you get another chance.

Go ahead. It's fun.

There are a lot of things to talk about in this game.

Defining the Target

Look at this line in the TARGET procedure. It starts with PU HT:

```
SETXY (RANDOM 900) - 450 (RANDOM 580) - 290
```

This seems complicated, doesn't it. Remember that RANDOM only picks numbers between 0 and the number you select. Since coordinates can be less than 1, we need random numbers that can put the target anywhere on the screen.

So let's read this line:

```
SETXY (RANDOM 900) - 450 (RANDOM 580) - 290
```

Turtle Positions and Coordinates

SETXY takes a list of two inputs, the X- and the Y-coordinate. First, Logo looks for the X-coordinate. You know the MSW Logo screen is 1000 turtle steps wide. So, the X-coordinates can be from -500 on the left of the screen, to 500 on the right.

But since we don't want our target right on the edge, we make the biggest number RANDOM can select a little smaller than 500.

$(\text{RANDOM } 900) - 450$

- If RANDOM selects the largest number [900] Logo subtracts 450 from it giving you an X-coordinate of 450.
- If RANDOM selects the smallest number [0] Logo subtracts 450 giving you an X-coordinate of -450.

NOTE: These coordinates are for an SVGA monitor set at 1024 x 768 resolution. You may have to adjust these coordinates to fit your screen.

Once Logo has the X-coordinate, it reads to the right to find the Y-coordinate.

$(\text{RANDOM } 580) - 290$

If we do the same arithmetic with the Y-coordinate as we did with the X-coordinate, we can have a Y-coordinate from 290 to -290. So, with this line, we can have a target just about anywhere on the screen.

Once the target position has been selected, a circle is drawn around that spot.

Testing Your Target Skills

There's the TEST command again. TEST is a conditional command that is a lot like IF, remember?. TEST checks to see if something is true or false. IFTRUE, or IFT, then it does one thing. IFFALSE, or IFF, it does something else.

```
TEST OR (ABS :X1) > 450 (ABS :Y1) > 290
IFTRUE [PRINT [YOUR NUMBER IS TOO BIG. ~
  TRY AGAIN.] WAIT 100 PLAY.GAME]
IFFALSE [PD ST SETXY :X1 :Y1]
```

If you want, you can use IF...THEN...ELSE instead.

```
IF OR (ABS :X1) > 450 (ABS :Y1) > 290 ~
  [[PR [YOUR NUMBER IS TOO BIG. TRY AGAIN.]~
  WAIT 100 PLAY.GAME]] [PD ST SETXY :X1 :Y1]
```

I don't know about you, but that gets a bit tough to read. That's why TEST can be a handy alternative.

```
TEST OR (ABS :X1) > 450 (ABS :Y1) > 290
```

This command tests your inputs for :X1 and :Y1 to find out if the coordinates you entered are within the target zone.

Reading from the left, Logo first reads the word TEST. To find out what needs to be tested, Logo continues reading.

```
TEST OR
```

OR is what they call a logical operator. It asks if either of two conditions is true:

```
(ABS :X1) > 450
(ABS :Y1) > 290
```

Turtle Positions and Coordinates

ABS stands for absolute value, which means it is neither a positive or a negative number. What ABS says is that it doesn't care if the X-coordinate :X1 is a positive number or a negative number. It drops the positive or negative symbol and just tests the random number.

For example, $(ABS :X1) > 450$ tests to see if the coordinate you input is within the target zone, no more than 450 or less than -290.

There's an ABS procedure in the Logolib directory or you can use this procedure.

```
TO ABS :NUM
  IFELSE :NUM < 0 [OP :NUM * -1] [OP :NUM]
END
```

If :NUM is less than zero — if it is a negative number — then output :NUM times -1. Two negative numbers multiplied together give you a positive number. If :NUM is positive, then output :NUM. Does that help?

Let's get back to the game procedures.

```
IFTRUE [PRINT [YOUR NUMBER IS TOO BIG.
  TRY AGAIN.] WAIT 20 PLAY.GAME]
```

If the test is true — the number you typed is too big — print “YOUR NUMBER IS TOO BIG. TRY AGAIN.” If it is false, or within the proper range, then the turtle draws a line out to the coordinates you guessed.

SPECIAL NOTE: Why not take a few minutes and make up some of your own tests? Here's one Morf did:



```
TO CHECK
TEST HEADING = 0
IFFTRUE [FORWARD 100]
IFFALSE [HIDETURTLE]
END
```

Another way to write this test condition is to use the IFELSE command.

```
TO CHECK
ST
IFELSE HEADING = 0 [FD 100] [HIDETURTLE]
END
```

Hitting the Target

Did you hit the target or not?

Actually, in this game, you don't have to hit the target exactly. If you come within 12 steps of the target coordinates, you score a hit. If not, you start over again.

Here's how Logo checks.

```
IF OR (:X1 > (:X + 12) ) (:X1 < (:X - 12))
  [PU HOME PD PLAY.GAME]
IF OR (:Y1 > (:Y + 12)) (:Y1 < (:Y - 12))
  [PU HOME PD PLAY.GAME]
CHEERS
```

The first two lines check to see if the X coordinate you picked is greater than X + 12 or less than X - 12. If so, the turtle gets sent home to start again.

Turtle Positions and Coordinates

If your X coordinate is on target, Logo then moves to the next line to check the Y coordinate. If your Y coordinate is also on target — less than $Y + 12$ and less than $Y - 12$ — Logo moves to the CHEERS procedure.

But first, why use 12? Why not 10? Or 15?

No reason. It's just that the target CIRCLE has a diameter of 24. The center is at the halfway point or 12.

```
REPEAT 36 [FD 2 RT 12]
```

How would you handle a circle with a radius of 10 or 15? Check back to More Circles and Arcs on page 313. There you'll find a circle procedure that starts from the center and lets you draw a circle with whatever radius you want.

Add Some Pizzazz

This CHEERS procedure doesn't do much, does it? It just prints CONGRATULATIONS! 24 times. So why not soup it up a bit.

Before you do that, take a look at the last part of the procedure. This is a standard way to ask if you want to play the game again. This works in just about all versions of Logo.

```
TO CHEERS
  CT REPEAT 24 [PRINT [CONGRATULATIONS!]]
  WAIT 100
  PRINT [WANT TO PLAY AGAIN? (Y/N)]
  MAKE "ANS READCHAR
  IF :ANS = "Y [GAME]
  IF :ANS = "N [PRINT [BYE FOR NOW!]]
  END
```

However, MSW Logo offers a rather special way to do the same thing. You can use the YESNOBOX. To check it out, run this simple procedure.

```
TO CHECK
MAKE "CHECK YESNOBOX [QUESTION] ~
  [PLAY AGAIN?]
IFELSE :CHECK = "TRUE [PR "GOOD][PR "BYE]
END
```

Now, why not change your CHEERS procedure.

```
TO CHEERS
<Add your own Pizzazz here.>
WAIT 100
MAKE "ANS [TARGET][WANT TO PLAY AGAIN?]
IFELSE :ANS = "TRUE [GAME] ~
  [PRINT [BYE FOR NOW!]]
END
```

See how the YESNOBOX works? First you give the box a name. In this case, the box is called TARGET after the game. In the box, the question is asked, "WANT TO PLAY AGAIN?" YES outputs TRUE so you go back to the game. NO outputs FALSE which send you to the option.

BYE FOR NOW!

The YESNOBOX is just one of many useful windows commands you can discover using MSW Logo. You'll read more of these in Chapter 11.

Turtle Positions and Coordinates

From the Center One More Time

The target in the game above is drawn around a center position. Do you remember drawing rectangles and things around a center point in Chapters 2 and 6? Now you have another way to look at that problem.

- Draw a rectangle that is 50 turtle steps wide and 120 turtle steps high.
- Put the center of the rectangle 100 turtle steps from HOME and 100 turtle steps to the left.

Rather than take turtle steps to find the center, you can use SETPOS.

```
TO RECTANGLE :CENTER :WIDTH :HEIGHT
  PU SETPOS :CENTER FD :WIDTH / 2 RT 90
  FD :HEIGHT / 2 RT 180 PD
  REPEAT 2 [ FD :HEIGHT RT 90 FD :WIDTH RT 90 ]
  END
```

This is like what you did before. But there's another way to draw that rectangle.

```
TO RECTANGLE :X :Y :WIDTH :HEIGHT
  PU SETXY :X :Y
  SETXY :X - :WIDTH / 2 :Y - :HEIGHT / 2 PD
  SETY YCOR + :HEIGHT SETX XCOR + :WIDTH
  SETY YCOR - :HEIGHT SETX XCOR - :WIDTH
  END
```

Here's one more way to draw this rectangle.

```
TO RECTANGLE :X :Y :WIDTH :HEIGHT
  (LOCAL "A "B "C "D)
  PU SETXY :X :Y
  SETXY :X - :WIDTH / 2 :Y - :HEIGHT / 2
  MAKE "A POS
```

Turtle Positions and Coordinates

```
SETY YCOR + :HEIGHT MAKE "B POS
SETX XCOR + :WIDTH MAKE "C POS
SETY YCOR - :HEIGHT MAKE "D POS
SETPOS :A PD
SETPOS :B
SETPOS :C
SETPOS :D
SETPOS :A
END
```

Now you can even draw the diagonal lines between the corners.

```
TO DIAGONALS :X :Y :WIDTH :HEIGHT
RECTANGLE :X :Y :WIDTH :HEIGHT
SETPOS :C
SETPOS :B
SETPOS :D
END
```

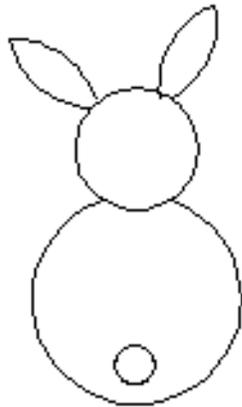
This will give you something new to think about. Can you find an easier way to write these procedures? Surely you can do better than this. What about doing the same thing with other polygons?

Turtle Positions and Coordinates

Who's Who in the Zoo

Now let's take a look at what some other young people did with creative coordinates. These procedures were sent to Logy and Morf as entries into contests they used to run.

One of the contests they ran a few years ago was to find the best animal procedures. Some of them are pretty simple. Others get pretty complicated. The procedures are on the disk that comes with this book



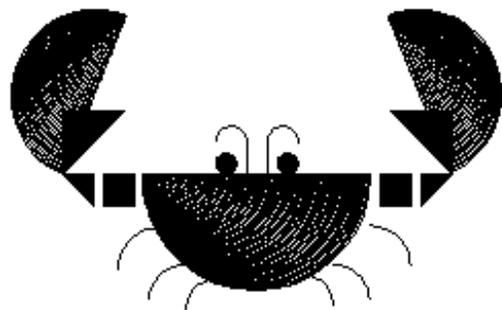
You can start off with one of the simpler ones, MORPH.

Did you know that the scientific name for a rabbit is "lagomorph?"

That sounded a lot like Logomorph. So a young lady from Virginia just shortened the last part and came up with a new name for Logy's friend.

Some young friends from New Jersey worked on a series of animal drawings for the contest.

I bet this is one of the cutest crabs you ever saw. It comes from one of the Ron Eberly books. Take a look at CRAB.LGO to see the procedures that created the crab.



Turtle Positions and Coordinates



The young people that designed the crab and the tiger cub (CUB.LGO) used a version of Logo that has no FILL command. So they wrote their own.

Here are two procedures they wrote to fill a curved area. They're a nice use of recursion to create a drawing.

```
TO FILL.ARCR :N :S
IF :N = 0 [STOP]
MAKE "SAVEX XCOR MAKE "SAVEY YCOR
MAKE "SAVEHEADING HEADING
ARCR :N :S
SETXY LIST :SAVEX :SAVEY
SETH :SAVEHEADING
FILL.ARCR :N - 1 :S
END
```

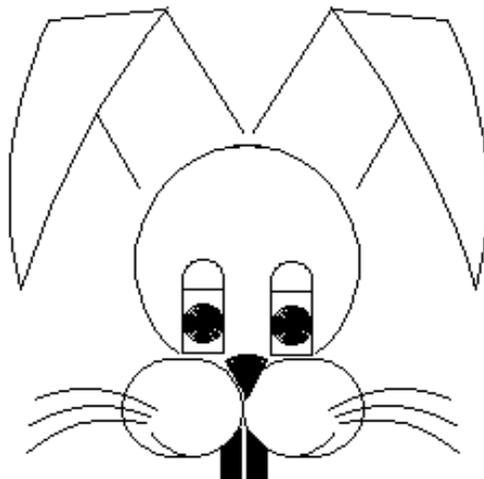
```
TO FILL.ARCL :N :SIF :N = 0 [STOP]
MAKE "SAVEX XCOR MAKE "SAVEY YCOR
MAKE "SAVEHEADING HEADING
ARCL :N :S
SETXY LIST :SAVEX :SAVEY
SETH :SAVEHEADING
FILL.ARCL :N - 1 :S
END
```

Turtle Positions and Coordinates

Learn Another Language

Want to learn another language?

The HAAS procedure has procedure titles written in Dutch. Can you figure out what those titles mean by what they do?



The main procedure shows you some of the Dutch procedure names.

```
TO HAAS
SETUP
LINKERWANG
RIMPEL.LINKS
PU LT 90 FD 40 RT 90 PD
SNOR.LINKS
PU RT 90 FD 40 LT 90 PD
RECHTERWANG
RIMPEL.RECHTS
PU RT 90 FD 40 LT 90 PD
SNOR.RECHTS
PU LT 90 FD 31 RT 90
PU BK 33 PD
TANDEN
PU FD 39 PD
NEUS
PU BK 4 PD PU FD 12 PD
HOOFD
PU BK 12 PD PU FD 25 LT 90 FD 30 RT 90 PD
OOG
PU RT 90 FD 41 LT 90 PD
```

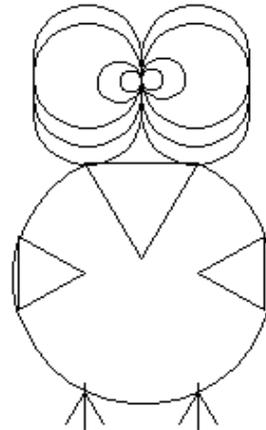
Turtle Positions and Coordinates

```
OOG
PU LT 90 FD 11 RT 90 PD
PU FD 80 PD
OREN
  PU BK 80 PD
END
```

```
TO HAAR.L
LT 60
REPEAT 6 [FD 10 LT 10]
PU REPEAT 6 [RT 10 BK 10] PD
RT 60
END
```

```
TO RIMPEL.LINKS
RT 180 PU REPEAT 18 [RT 5 FD 2]
PD REPEAT 12 [RT 5 FD 2]
PU REPEAT 30 [BK 2 LT 5]
RT 180 PD
END
```

Did you ever see a DODOBIRD? That seemed like a good name for this crazy bird.

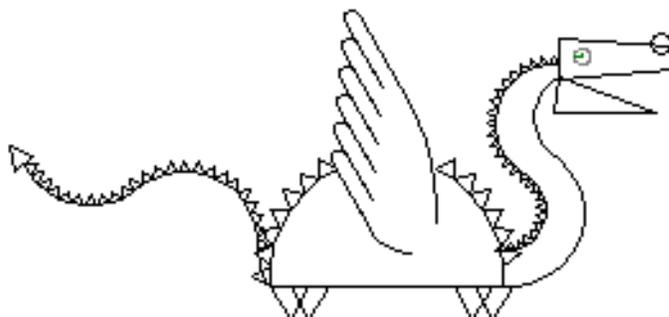


Now, what kind of animal can you dream up? Why not do some sketches. Then put them on the computer.

Puff, The Magic Dragon, may give you an idea or two. Puff was sent in as an entry to a Logo animation contest Logy

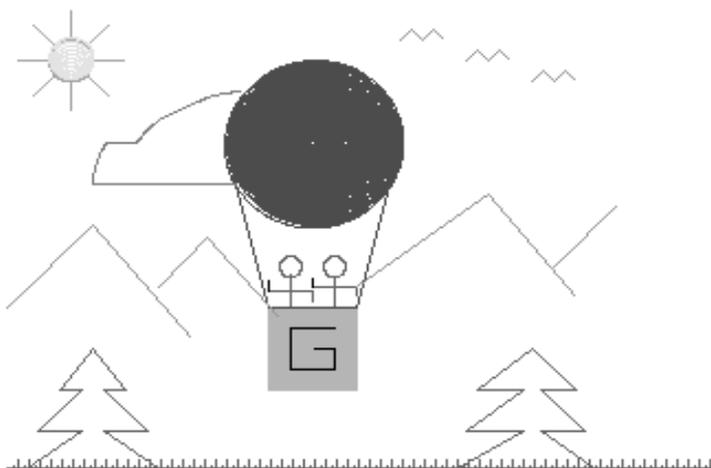
Turtle Positions and Coordinates

and Morf had some years back. As you will see from the PUFF.LGO procedure, Puff breathes smoke and spits fire.



Up, Up, and Away

A very popular contest that Logy and Morf held was called the Up, Up, and Away Contest. It celebrated the birthday of the first flight of a hot air balloon. Young people were told to let their imaginations run wild and create a drawing that featured a hot air balloon. Here's Gretchen's balloon (BALLOON.LGO).

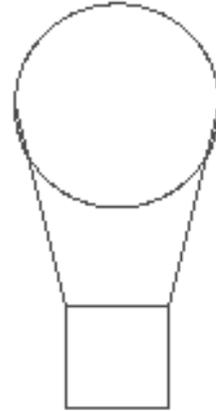


Gretchen was among the winners because her entry was the only one that included animation. The two people in the balloon wave goodbye to you.

Why not try a balloon yourself? Start with a basic balloon shape like this one.

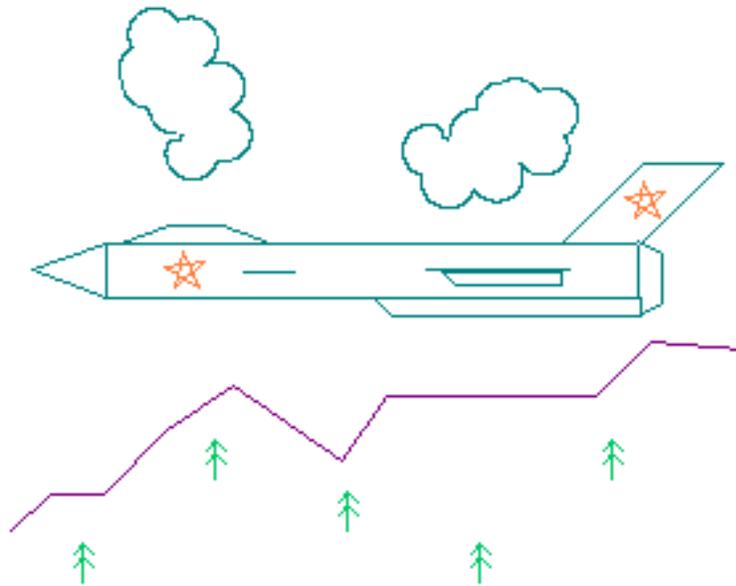
Then see what you can do with it.

How about making the balloon into a turtle that floats over the landscape?



Fly, Fly Away

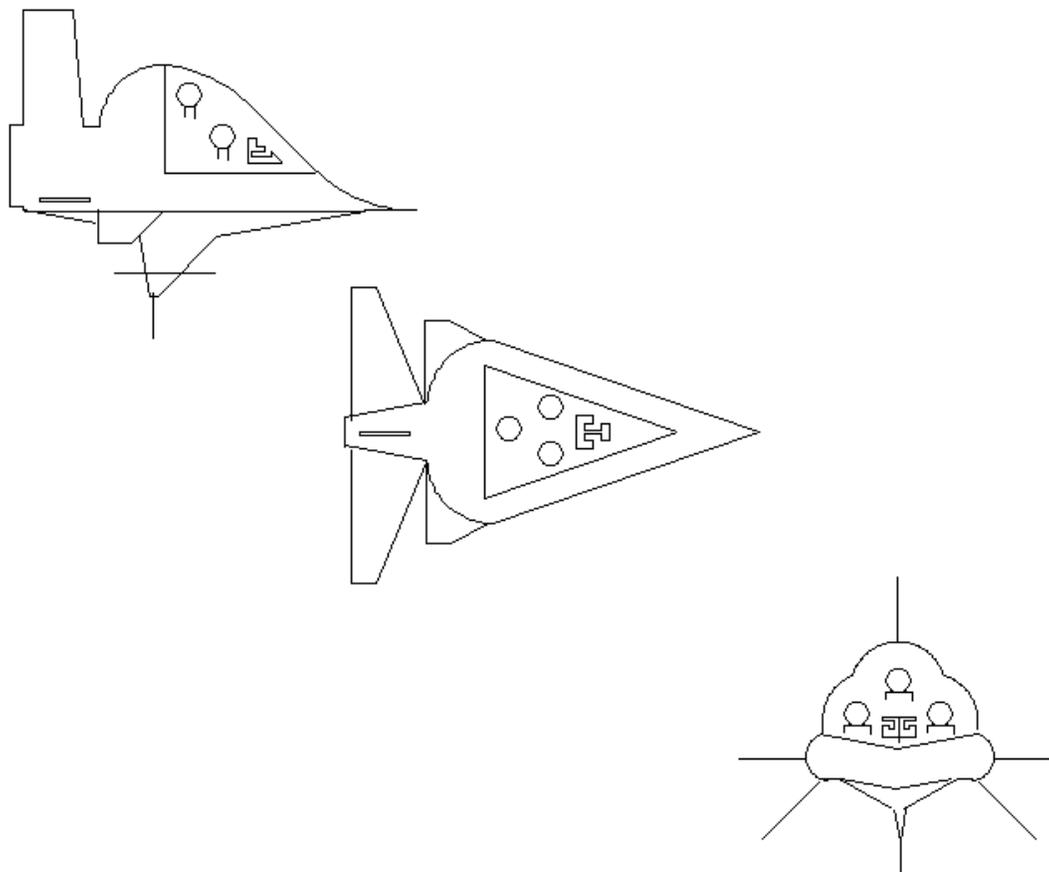
The Up, Up, and Away Contest was so successful, Logo and Morf held a Fly, Fly Away Contest to see who could create the best drawing of an aircraft. This offered young people another great chance to let their imaginations run wild.



FIREFOX.LGO is a translation of one of the many procedures sent in by young people at the Utah State School for the Deaf. You'll find pictures based on this procedure in the PCX directory of the CD that came with this book. These may give you some ideas for ways to use this and other procedures mentioned in this chapter.

Turtle Positions and Coordinates

Here are the three views of X2.LGO, a very creative procedure from an 11-year-old from San Diego.

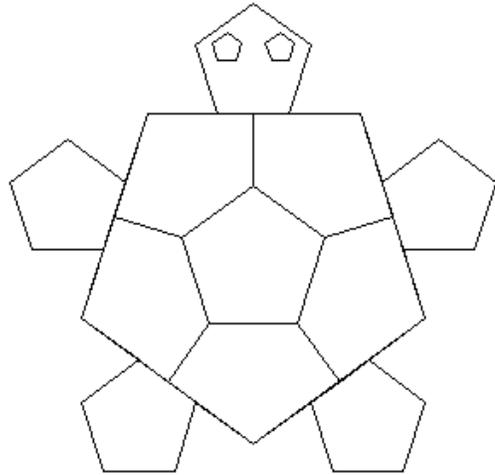


Have you ever been on an airplane? Why not create a picture of it? Or maybe you want to try a helicopter? How about a blimp? Then you can put your favorite advertising message along the side of it.

The Variable Turtle

Here's another use of turtle positions. This draws a turtle any size that you want it to be.

Logy and Morf have used this drawing on software and other things they have published over the years. .



```
TO YPLA :SIZE
TURTLE1
MAKE "P []
REPEAT 6 [TURTLE2]
TURTLE3
END
```

```
TO TURTLE1
CS PD HT RT 18
MAKE "TSIZE :SIZE / 3
REPEAT 5 [FD :TSIZE LT 108~
  REPEAT 3 [FD :TSIZE RT 72] ~
  FD :TSIZE LT 108 FD :TSIZE RT 72]
REPEAT 5 [FD :SIZE RT 72]
END
```

```
TO TURTLE2
FD :SIZE / 2 RT 90 FD :TSIZE
MAKE "Q POS
IF :P = [] [MAKE "P POS]
SETXY :P SETXY :Q
MAKE "P :Q BK :TSIZE LT 90
FD :SIZE / 2 RT 72
END
```

Turtle Positions and Coordinates

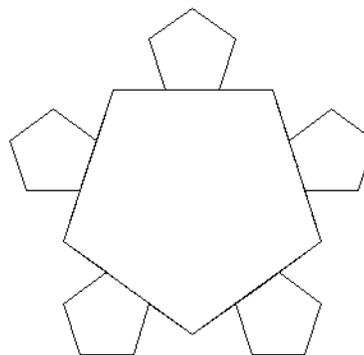
```
TO TURTLE3
PU FD :TSIZE LT 90 FD :SIZE / 4 LT 18 PD
REPEAT 5 [FD :SIZE / 12 RT 72]
PU RT 18 BK :SIZE / 4 RT 90
PU FD :TSIZE LT 90 FD :SIZE / 4 RT 18 PD
REPEAT 5 [FD :SIZE / 12 LT 72]
END
```

To understand how this procedure work, you have to pick them apart and run them one piece at a time. Try this:

```
TO YPLA :SIZE
TURTLE1
END
```

Before you run this procedure, what do you think it does? What will the turtle draw?

There's one thing you can see right away. Since it uses REPEAT 5, it probably draws some pentagons.



The next thing worth noticing is the use of POS in TURTLE2.

```
TO TURTLE2
FD :SIZE / 2 RT 90 FD :TSIZE MAKE "Q POS
IF :P = [] [MAKE "P POS]
SETPOS :P SETPOS :Q
MAKE "P :Q BK :TSIZE LT 90 FD :SIZE / 2 RT 72
END
```

Turtle Positions and Coordinates

This procedure draws the pentagon shapes in the center of the turtles shell. This seems easy enough but you have to remember; this procedure draws any size turtle you want.

TURTLE2 could probably be written using some fancy variables. POS makes it easier. The second line says, “I don’t care where you are after doing what it says in the first line. Now, just remember the coordinates where you are and call them Q.

Now let’s take a look at :P.

The easiest way to see how :P and the entire YPLA procedure works is to erase all the instructions that include :P. Then run the procedure. What do you see? Everything is there except the pentagon in the center. Now, if you read the procedures carefully, you see how TURTLE2 uses SETXY to draw the center pentagon.

No, this isn’t easy. However, don’t get discouraged. Exploring like this is what makes it such a Great Logo Adventure.

Logo Geography

It started with the idea of Wrapping.



NO, that's a different kind of wrapping!

Some young people in a local elementary school computer club got curious as to what happened when the turtle went off one edge of the screen and then showed up on the opposite edge. Did it travel behind the screen?

This is the same group that explored the soccer ball. They already understood the idea of flattening three-dimensional objects on the two-dimensional screen. So it was easy enough to picture flattening the world into a two-dimensional map, that if they travelled off one edge of the map, they would automatically “wrap” to the same spot on the opposite edge.

But how can you stop the turtle from wrapping?

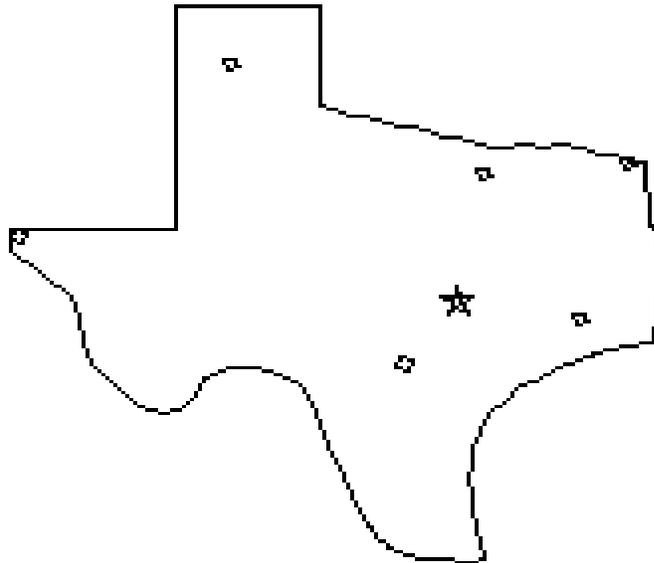
`WINDOW` is the command that tells the turtle not to wrap, just keep on going off into the world beyond the screen.

`FENCE` is the command that stops the turtle at the edge of the screen. It stops the action and tells you that the turtle is “Out of Bounds.”

Exploring Texas

Let's get back to geography. You may remember Turtle Town from Chapter 2. Pictures of Turtle Town are in the graphics directory on the CD that came with this book.

After talking about all sorts of maps, and creating an elaborate Turtle Town game, the students decided to create their own maps on the screen. Here's a map of Texas that they developed. It's part of the TEXAS.LGO procedure



The map of Texas includes the cities of Amarillo, Austin, Dallas, El Paso, Houston, San Antonio, and Texarkana. And since game design was a very important activity with this group, they made a game out of it.

The GAME procedure, which is only a small part of the complete procedure, describes the game action. Logy and Morf start out at a randomly selected city and have to travel to the target city, also randomly selected.

TO GAME

PR [Logy, What direction do we turn?]

PR [How many miles do we have to go?]

Turtle Positions and Coordinates

```
WAIT 80
PR [Can you give Logo a hand?]
PR [What direction do they turn?]
MAKE "DIR RW
PR [How far?]
MAKE "TURNS RW
IF :DIR = TURNS [LT :TURNS]
IF :DIR = "RT [RT :TURNS]
PR [How many miles do they have to travel?]
MAKE "FAR RW PU
FD :FAR * .187
CHECK
END
```

It took a lot of discussion but the group finally figured out that the actual miles on the screen map were the number of turtle steps times 0.187.

$550 \text{ miles} * 0.187 = 103 \text{ turtle steps}$
(Actually, it's 102.85. But 103 is close enough.)

The game was good practice in guessing directions and drawing mileage to scale. After working with Texas, the group did several other states. Each time, they kept adding new features.

1. Take a look at the Texas procedure and see if you can develop the same type of procedure for your state.
2. Add other landmarks from your state...maybe lakes, rivers, parks, and things.
3. If a player moves from one place to another correctly, have them answer a random question about that place. Think up five or six questions for each location you put on your map.

Can you think of other ways to explore Logo geography?