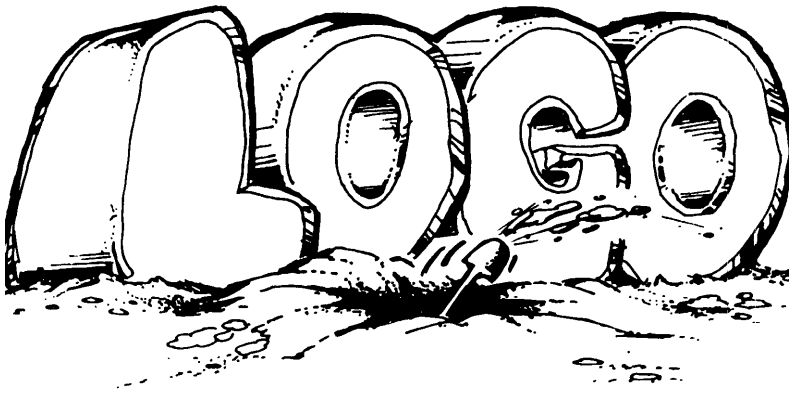
Chapter 12. What's Next

Believe it or not, you've just started to dig into Logo.



You've seen a lot. You have hopefully explored the many procedures and commands described in these pages. But keep one important thought in mind.

The procedures offered in this book and on the diskette that comes with it are how others solved a problem. That doesn't mean that these solutions are the best way to do things. They may not be the easiest or the most efficient way. These are just places for you to start on your own adventures.

How many of your own rabbit trails did you discover?

Even though you may have found a bunch, there is so much more to discover. But at least now you have a few tools to use now.

Before we go, here are a few more things to think about.

From Two to Three Dimensions

Here's a challenge for you. Draw pattern of a soccer ball on the screen.

The first thing you see, looking at a soccer ball, is a bunch of hexagon shapes. When some 3rd and 4th grade computer club members were asked to draw this pattern on the screen, they thought it would be easy.

```
TO SOCCER.BALL :DIS
REPEAT 6 [REPEAT 6 [FD :DIS RT 60] FD :DIS LT 60]
END
```

The boys thought that all they had to do was draw a series of hexagons. But the center was a pentagon, not a hexagon. Try their SOCCER.BALL procedure, the one above. It's not quite right, is it?

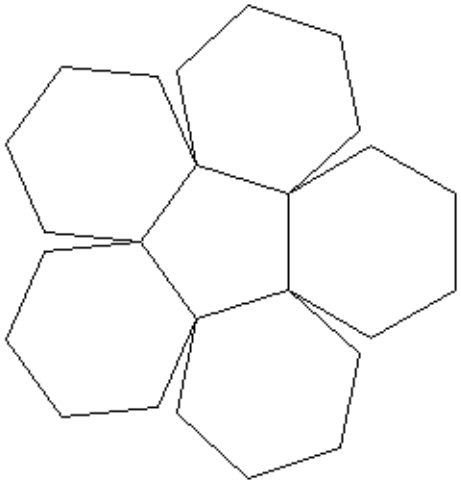
The girl's team was the first to figure out that they could not make the soccer pattern on the screen as it appears on the ball. They had to flatten it out.

At first, they thought this procedure was wrong. But then they discovered it was really correct.

```
TO SOCCER :DIS
REPEAT 5 [REPEAT 6 [FD :DIS RT 60] FD :DIS LT 72]
END
```

The girls printed twelve of their patterns, colored them, taped them together, and made their own soccer ball. When they were finished, they decided it made a better pinata.

So they filled it with candy and had a party.



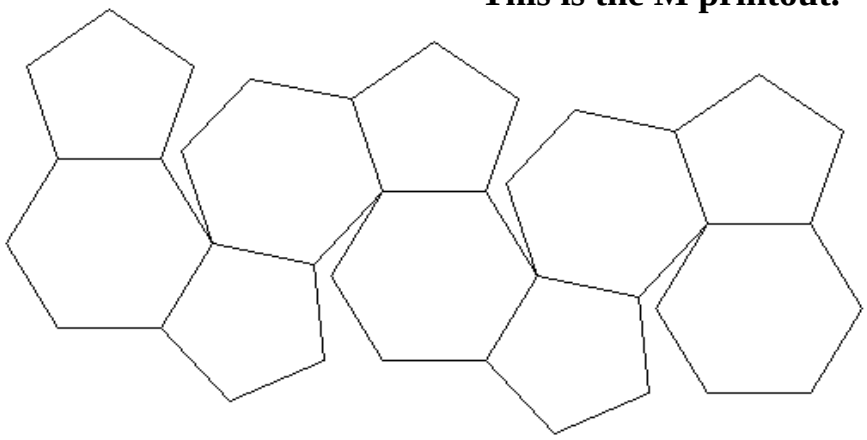
When this story was published in former YPLA newspaper, Turtle News, different groups and classes around the world had fun with it.

Here's one young person's response. His challenge was to reduce the pattern to the fewest number of parts. Two was the best he could do.

```
TO START
ADAM'S.SOCCER.BALL
WAIT 100 CS
M
PR [THIS IS THE FIRST SIDE.] WAIT 100 CS CT
M2
PR [TAPE THIS SIDE TO THE FIRST SIDE.]
END

TO ADAM'S.SOCCER.BALL
CT
PR [This Logo procedure is from Adam Johnson,]
PR [age 12, The Computer Learning Center,]
PR [Caldwell, Idaho 83605.]
END
```

```
TO M
RT 90 PU FD 180 PD
H FD 60 LT 90 P
K H L P K2 H P2 P
K H L P K2 H P2 P
END
```

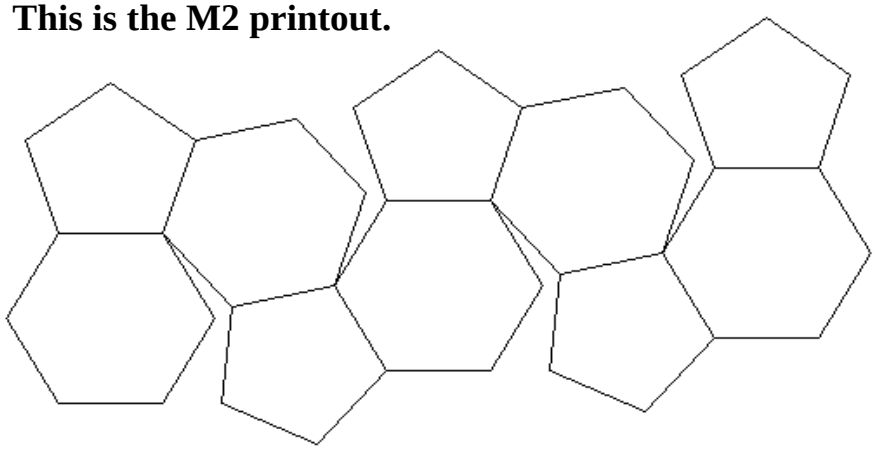
This is the M printout.



```
TO K
FD 60 RT 72 FD 60 LT 180
END

TO H
REPEAT 6 [FD 60 RT 60]
END

TO L
FD 60 RT 60 FD 60 RT 42
END

TO P
LT 90
REPEAT 5 [FD 60 RT 72]
END
```

```
TO K2
LT 72 BK 60 RT 108
END

TO P2
LT 60 BK 60 RT 210
END

TO M2
PU LT 90 FD 240 RT 180
PD H FD 60 LT 90
P RT 108 H P2 P K H
L P K2 H P2 P K H L P
END
```

This is the M2 printout.



Rabbit Trail 25. Folded Paper Fun

Making the soccer ball out of paper is just one of many things you can do with Logo and folded paper. The computer club that made the first flattened soccer ball pattern found that you can make all sorts of three dimensional objects from folded paper.

How about a simple cube? This takes you from the two dimensional square to a three dimensional cube.

```
TO CUBE  :D :X1 :Y1
HT PU SETXY :X1 :Y1 SETH 0 PD
REPEAT 4 [SQUARE :D SETX :X1 + :D MAKE "X1
      XCOR]
MAKE "X1 XCOR - (:D * 3 )
MAKE "Y1 YCOR - :D
PU SETXY :X1 :Y1 PD
REPEAT 3 [SQUARE :D FD :D]
END

TO CUBES  :D :X1 :Y1
CUBE :D :X1 :Y1
CUBE :D :X1 + ( :D * 4 ) :Y1
END

TO SQUARE  :D
REPEAT 4 [FD :D RT 90]
END
```

The group first cut out a number of cardboard squares. Then they taped them together to see what kind of shapes they could make. The next step was to transfer the pattern to the computer.

Making 3-D shapes from triangles really got interesting

```
TO TETRAHEDRON  :D
RT 30 TRI :D MOVER :D TRI :D
MOVEL :D TRI :D
END
```

```
TO MOVER  :D
RT 60 FD :D LT 60
END

TO MOVEL  :D
LT 60 FD :D RT 60
END

TO TRIR  :D
RT 60 FD :D TRI :D
END

TO TRI  :D
REPEAT 3 [FD :D RT 120]
END

TO OCTAHEDRON  :D
LT 30 TRI :D RT 30 TETRAHEDRON :D
LT 60 TRI :D TRIR :D TRIF :D
END

TO TRIF  :D
FD :D RT 60 TRI :D
END
```

TETRAHEDRON and OCTAHEDRON are just the beginning of what you can do with Logo and a printer.

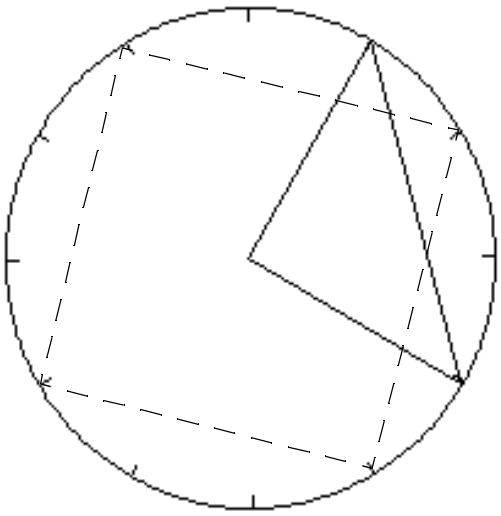
Go ahead. Try these. Print them. Fold them up. And then design your own 3-D figures.

The whole idea is to explore, to discover what you don't know and then go find the answers.

Rotating Planes in Space

"No, Morf. This is not about rotating or rolling your favorite Biplane. This is about moving shapes through space.

Remember this picture? You saw it in Chapter 2 and also in Chapter 9 of this book., only it didn't have the dotted-line square in the picture then.



The original idea was to use string and make a triangle; first from 12:00 o'clock to 3:00 o'clock, the center, and back to 12:00 'oclock. Next you moved to 1:00 o'clock and made the triangle shown in the picture. Then you moved to 2:00 o'clock and so one, all the way around the clock.

The triangles moved around an axis that was at the center of the string clock. When you did the same thing with squares, they moved around on an axis at the center of the square.

Can you write Logo procedures that shows the shapes rotating around an axis?

Sure you can! Here's a start.

```
TO START
CS
PR [THESE PROCEDURES ALLOW YOU TO SEE]
PR [TWO AND THREE DIMENSIONAL FIGURES]
PR [ROTATE AROUND AN AXIS.]
PR "
PR [ROTATE: ROTATE FOUR SQUARES AROUND]
PR [A CENTRAL POINT.]
PR "
PR [ROTATE.CUBE: ROTATE A CUBE AROUND A]
PR [CENTRAL POINT.]
PR "
PR [CENTER.ROTATE: ROTATE A SQUARE AROUND]
PR [A POINT IN THE CENTER OF THE PLANE.]
PR "
PR [ROTATE.SQUARE: ROTATE A SQUARE AROUND]
PR [A POINT OUTSIDE THE SQUARE.]
END

TO ROTATE
ROTATE.REC 0
END

TO ROTATE.REC  :ANGLE
CS
DRAW.ANG :ANGLE
ROTATE.REC :ANGLE + 3
END

TO DRAW.ANG  :ANGLE
RECT 100 ( 100 * COS :ANGLE )
RECT 100 ( 100 * COS ( :ANGLE + 90 ) )
RECT 100 ( 100 * COS ( :ANGLE + 180 ) )
RECT 100 ( 100 * COS ( :ANGLE + 270 ) )
END
```

```
TO ROTATE.SQUARE
ROTATE.SQ 0
END

TO ROTATE.SQ  :ANGLE
CS
DRAW.SQUARE ( 100 * ( COS :ANGLE ) ) ~
        ( 50 * ( COS ( :ANGLE + 90 ) ) )
ROTATE.SQ ( :ANGLE + 5 )
END

TO DRAW.SQUARE  :WIDTH :CENTER
PU HOME RT 90
FD :CENTER
LT 90 PD
CENTERRECT 100 :WIDTH
END

TO ROTATE.CUBE
HT ROT.CUBE 0
END

TO ROT.CUBE  :ANGLE
CS
DRAW.SQUARE ( 100 * ( COS :ANGLE ) ) ( 50 * ( COS (
:ANGLE + 90 ) ) )
DRAW.SQUARE ( 100 * ( COS ( :ANGLE + 90 ) ) ) ( 50 * (
COS ( :ANGLE + 180 ) ) )
DRAW.SQUARE ( 100 * ( COS ( :ANGLE + 180 ) ) ) ( 50 *
( COS ( :ANGLE + 270 ) ) )
DRAW.SQUARE ( 100 * ( COS ( :ANGLE + 270 ) ) ) ( 50 *
( COS ( :ANGLE + 360 ) ) )
ROT.CUBE :ANGLE + 5
END
```

```
TO CENTERRECT  :LENGTH :WIDTH
CENTER.CORNER :LENGTH :WIDTH
RT 180
RECT :LENGTH :WIDTH
RT 10
CENTER.CORNER :LENGTH :WIDTH
END

TO CENTER.ROTATE
CENTER.ROTATE.RECT 0
END

TO CENTER.ROTATE.RECT  :ANGLE
CLEARSCREEN
DRAW.CENTER.RECT ( 100 * ( COS :ANGLE ) )
CENTER.ROTATE.RECT ( :ANGLE + 1 )
END

TO DRAW.CENTER.RECT  :WIDTH
CENTER.RECT 100 :WIDTH
END

TO RECT  :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90]
END

TO CENTER.CORNER  :LENGTH :WIDTH
PU LT 90 FD :WIDTH / 2
LT 90 FD :LENGTH / 2 PD
END

TO CENTER.RECT  :LENGTH :WIDTH
CENTER.CORNER :LENGTH :WIDTH
RT 180
RECT :LENGTH :WIDTH
RT 180
CENTER.CORNER :LENGTH :WIDTH
END
```

This procedure shows squares and cubes rotating around an axis. It also gives you another example of trigonometry at work. How about a challenge or two?

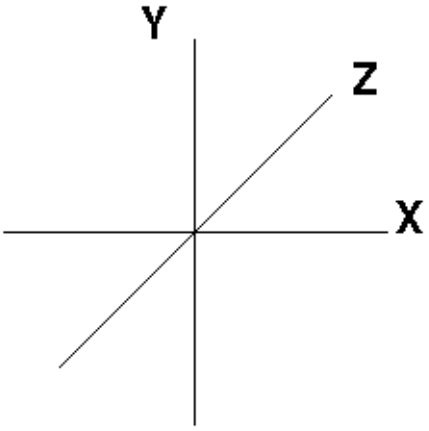
1. These procedures show the "side view" of the shapes rotating around a vertical axis. Write procedures to show a top view of the shapes rotating around the axis — as if you were looking down on the shapes.
2. Write procedures for other shapes, such as a rotating triangle. Use different types of triangles.
3. What about moving a pyramid through space?

Well, maybe you had beter leave that job to the next project.

How About Turtle CAD

When some junior high students saw the work that the third grade students had done creating and folding soccer ball patterns, they wondered if it would be possible to work in three dimensions on the Logo screen. They were thoroughly familiar with the two dimensions of the x - y coordinate system.

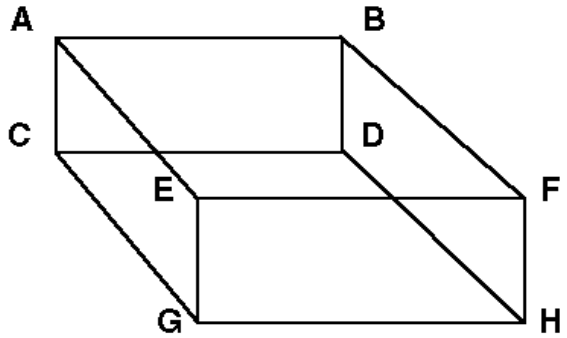
Could this be expanded to serve three dimensions: X, Y, and Z?



Yes, it can. In addition, the resulting procedure offers a good look at property lists, an often confusing feature of Logo.

In the procedure listed on the following pages, (ECONOBOX.LGO on the diskette) the basic unit is the coordinate point as defined by the POINT procedure. Points have letter names and x, y, and z coordinates to position them in three-dimensional space.

Take a look at the DIAMOND and CAR procedures. To display a 3-D object, you must define each point in space and each line. Once you have defined all the required points by name and position, you can construct shapes like this one.



The FIGURE procedure takes the shape name and a list of two-point lists; for example, [[A B][A C][A E][B F][B D] [C D][C G]…]. The two-point lists represent the line segments of the shape with each letter representing an endpoint.

Here's a simple procedure to develop a pyramid. Each point is defined along with the lines linking those points.

```
TO PYRAMID
POINT "A [0 0 0]
POINT "B [0 0 50]
POINT "C [50 0 50]
POINT "D [50 0 0]
POINT "E [25 50 25]
FIGURE "PYRAMID [[A B][B C][C D][D A][A E]
  [B E][C E][D E]]
END
```

The procedure allows you to create as many shapes as you want. Each can be as complex as you want. But only one can be manipulated at a time.

Once you have defined your shape, you can expand it or contract it, rotate it, magnify it, shrink it, and then restore it to its original shape.

To expand a shape, use EXPAND. Tell the procedure which shape to expand, which axis the expansion will operate on, and how much to expand it.

MAGNIFY is very similar to EXPAND. However, you don't specify an axis since the figure is magnified in all directions.

ROTATE operates on a plane: xy, xz, or yz. Specify the shape, the plane, and the degrees of rotation you want to see. As you move your shape through space, the turtle remembers the position of your shape and moves it from its last position. When you want to start over with a new shape, or start from your shape's original position, use

RESTORE "<*figure name*>.

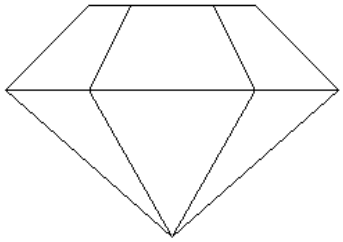
To get you started, there are three examples provided in the procedure below: a diamond, a pyramid, and an econobox-like car. The entire procedure has been printed here so you can follow it a bit easier.

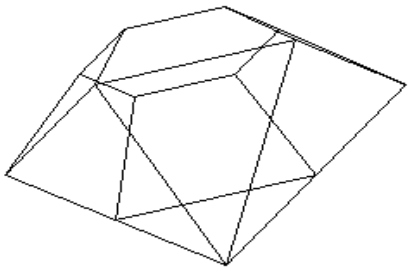
Type DIAMOND or CAR to see a front view of the figures. Then rotate the figures using commands such as:

ROTATE "<*figure name*> "XY 45
ROTATE "<*figure name*> "YZ 30

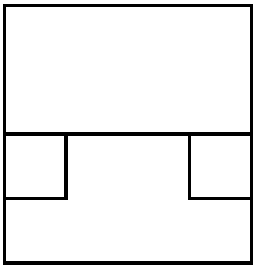
Now you're ready to start off on your own.

```
TO DIAMOND
POINT "A [60 0 60]
POINT "B [0 51.961 51.961]
POINT "C [30 51.961 103.92]
POINT "D [90 51.961 103.92]
POINT "E [120 51.961 51.961]
POINT "F [90 51.961 0]
POINT "G [30 51.961 0]
POINT "I [30 81.961 51.961]
POINT "J [45 81.961 77.941]
POINT "K [75 81.961 77.941]
POINT "L [90 81.961 51.961]
POINT "M [75 81.961 25.98]
POINT "H [45 81.961 25.98]
FIGURE "GEM [[A B] [A C] [A D] [A E] [A F] [A G] ~
        [B C] [B I] [C J] [C D] [D K] [D E] [E L] [E F] ~
        [F M] [F G] [G H] [G B] [I J] [J K] [K L] [L M] ~
        [M H] [H I]]
END
```

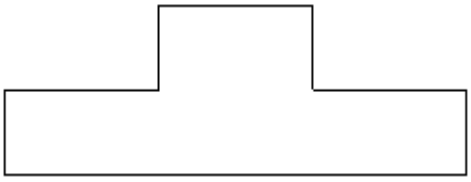




```
TO CAR
POINT "A [0 0 0]
POINT "B [0 20 0]
POINT "C [40 20 0]
POINT "D [40 0 0]
POINT "E [0 20 40]
POINT "F [0 40 40]
POINT "G [40 40 40]
POINT "H [40 20 40]
POINT "I [0 20 80]
POINT "J [0 40 80]
POINT "K [40 40 80]
POINT "L [40 20 80]
POINT "M [0 0 120]
POINT "N [0 20 120]
POINT "O [40 20 120]
POINT "P [40 0 120]
POINT "Q [0 10 0]
```

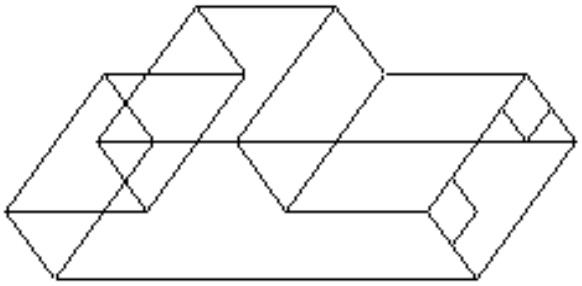


Front View



ROTATE "AUTO "XZ 90

```
POINT "R [10 10 0]
POINT "S [10 20 0]
POINT "T [40 10 0]
POINT "U [30 10 0]
POINT "V [30 20 0]
FIGURE "AUTO [[A B] [A D] [A M] [B C] [S R] [R Q] ~
      [B E] [C D] [T U] [U V] [C H] [D P] [P O] [P M] ~
      [O L] [O N] [N M] [N I] [I J] [I L] [J F] [J K] [K L] ~
       [K G] [G H] [G F] [F E] [E H]]
END

TO POINT :POINTNAME :COORDS
MAKE :POINTNAME :COORDS
PPROP :POINTNAME "POINT "TRUE
PPROP :POINTNAME "ORIG :COORDS
END

TO FIGURE :SHAPENAME :LP
IF (GPROP :SHAPENAME "POINT) = "TRUE ~
      [(PR :SHAPENAME [IS ALREADY A POINT ~
      NAME.]) STOP)]
MAKE :SHAPENAME :LP
PPROP :SHAPENAME "FIGURE "TRUE
MAKE "SX (WORD :SHAPENAME "PTS)
MAKE :SX [] MAKE "N9 1
REPEAT COUNT :LP ~
      [TSF :N9 TSL :N9 MAKE "N9 :N9 + 1]
MAKE "MATRIX [1 0 0 0 1 0 0 0 1]
DRAW:SHAPENAME
(PR :SHAPENAME [IS NOW A SHAPE.])
END
```



ROTATE "AUTO "XY 45
ROTATE "AUTO "XZ 45

```
TO DRAW:FIGURE
MAKE "S9 THING (WORD :FIGURE "PTS)
REPEAT COUNT :S9 [MAKE "P9 FIRST :S9 ~
        MAKE :P9 (LIST (ITEM 1 :MATRIX) * ~
        (ITEM 1 THING :P9) + (ITEM 2 :MATRIX) * ~
        (ITEM 2 THING :P9) + (ITEM 3 :MATRIX) * ~
        (ITEM 3 THING :P9) (ITEM 4 :MATRIX) * ~
        (ITEM 1 THING :P9) + (ITEM 5 :MATRIX) * ~
        (ITEM 2 THING :P9) + (ITEM 6 :MATRIX) * ~
        (ITEM 3 THING :P9) (ITEM 7 :MATRIX) * ~
        (ITEM 1 THING :P9) + (ITEM 8 :MATRIX) * ~
        (ITEM 2 THING :P9) + (ITEM 9 :MATRIX) * ~
        (ITEM 3 THING :P9)) MAKE "S9 BF :S9]
CS ST MAKE "S9 THING :FIGURE
REPEAT COUNT :S9 [PU SETXY LIST BL THING  ~
        (FIRST FIRST :S9) PD SETXY LIST BL THING ~
        (LAST FIRST :S9) MAKE "S9 BF :S9]
END

TO RESTORE :FIGURE
IF NOT (GPROP :FIGURE "FIGURE) = "TRUE ~
        [(PR :FIGURE [IS NOT A SHAPE.]) STOP]
MAKE "N9 THING (WORD :FIGURE "PTS)
REPEAT COUNT :N9 [MAKE FIRST :N9  ~
        GPROP (FIRST :N9) "ORIG MAKE "N9 BF :N9]
MAKE "MATRIX [1 0 0 0 1 0 0 0 1]
DRAW:FIGURE
END
```

```
TO ROTATE :FIGURE :AXIS :AMT
IF NOT MEMBERP :AXIS [XY XZ YZ] [PR ~
      [THE AXIS MUST BE XY, XZ, OR YZ.] STOP]
IF NOT (GPROP :FIGURE "FIGURE) = "TRUE ~
      [(PR :FIGURE [IS NOT A SHAPE.]) STOP]
IF :AXIS = "XY [MAKE "MATRIX (LIST ~
      (COS :AMT) 0 - (SIN :AMT) 0 (SIN :AMT)
      (COS :AMT) 0 0 0 1)]
IF :AXIS = "XZ [MAKE "MATRIX (LIST ~
      (COS :AMT) 0 0 - (SIN :AMT) 0 1 0 (SIN :AMT) ~
      0 (COS :AMT) 0)]
IF :AXIS = "YZ [MAKE "MATRIX (LIST 1 0 0 0 ~
      (COS :AMT) 0 - (SIN :AMT) 0 (SIN :AMT) ~
      (COS :AMT))]
DRAW:FIGURE
END

TO MAGNIFY :FIGURE :AMT
IF NOT (GPROP :FIGURE "FIGURE) = "TRUE ~
      [(PR :FIGURE [IS NOT A SHAPE.]) STOP]
MAKE "MATRIX (LIST :AMT 0 0 0 :AMT 0 0 0 :AMT)
DRAW :FIGURE
END

TO EXPAND :FIGURE :AXIS :AMT
IF NOT MEMBERP :AXIS [X Y Z] [PR ~
      [THE AXIS MUST BE X, Y, OR Z.] STOP]
IF NOT (GPROP :FIGURE "FIGURE) = "TRUE ~
      [(PR :FIGURE [IS NOT A SHAPE.]) STOP]
IF :AXIS = "X [MAKE "MATRIX ~
      (LIST :AMT 0 0 0 1 0 0 0 1)]
IF :AXIS = "Y [MAKE "MATRIX ~
      (LIST 1 0 0 0 :AMT 0 0 0 1)]
IF :AXIS = "Z [MAKE "MATRIX ~
      (LIST 1 0 0 0 1 0 0 0 :AMT)]
DRAW :FIGURE
END
```

```
TO TSL :N9
IF NOT MEMBERP LAST (ITEM :N9 :LP)  ~
       THING :SX [MAKE :SX FPUT LAST  ~
       (ITEM :N9 :LP) THING :SX]
END

TO TSF :N9
IF NOT MEMBERP FIRST (ITEM :N9 :LP)  ~
       THING :SX [MAKE :SX FPUT FIRST  ~
       (ITEM :N9 :LP) THING :SX]
END
```

Understanding Property Lists

For the longest time, Property Lists were a major source of confusion — until Logy and Morf discovered a way to read them.

Put the PROPerty of :PROPERTY, which has the value of :VALUE, with the name :NAME. It helped when they saw PPROP written as a procedure.

```
TO PPROP :NAME :PROPERTY :VALUE
MAKE (WORD :NAME CHAR 32 :PROPERTY) :VALUE
END
```

Another way to look at the PPROP procedure is to use some more familiar terms.

```
PPROP "TEXAS "CAPITAL "AUSTIN
PPROP "TEXAS "ABBREVIATION "TX
PPROP "TEXAS "CITIES [HOUSTON DALLAS
  AMARILLO EL PASO]
PPROP "TEXAS "REGION "HILL.COUNTRY
```

Put the property of CAPITAL with the value of AUSTIN with the name, TEXAS. Put the ABBREVIATION TX with TEXAS. Put the cities of Houston, Dallas, Amarillo, and El Paso with Texas.

In the POINT procedure used on the previous pages, you have

```
PPROP :POINTNAME "POINT "TRUE
PPROP :POINTNAME "ORIG :COORDS
```

Put the property of POINT with the value of TRUE with the name :POINTNAME.

Put the property of ORIG with the value of COORDS with the name :POINTNAME. Take a look at the RESTORE procedure to see how this is used.

It's not nearly as scary as you think!

Now that you have the properties defined, what can you do with them? For one thing, you can recall them using the GPROP procedure. That's exactly what's done in the RESTORE procedure.

```
TO GPROP :NAME :PROPERTY
OUTPUT THING (WORD :NAME CHAR 32
  :PROPERTY)
END
```

This outputs the THING (the value) defined in the PPROP procedure. For example:

GPROP "TEXAS "CAPITAL results in AUSTIN.

GPROP (FIRST :N9) "ORIG results in the original coordinates for :NAME.

The THREED procedures are one good examples of property lists. Play around with them on your own. You're bound to find other uses.

Bury and Unbury

BURY is one of those Logo primitives that is often ignored. But it can be very useful.

Let's try something.

1. Load any procedure.
2. Type BURYALL and press Enter.
3. Type EDALL and press Enter.

   Where'd the procedures go?

4. Try to run the buried procedure. What happened?
5. Now load another procedure.
6. Type UNBURYALL and press Enter.
7. Type EDIT ALL and press Enter.

Both the procedures are now visible in the Editor, aren't they?

What this means is that you can bury certain conditions and then erase everything else. If you are writing an adventure game, you can have your character carry things from one situation, one procedure, to another.

Experiment with these commands. You'll find lots of uses for them.

Why not write some color procedures and bury those? For example:

```
TO BLACK
SETPC [0 0 0]
END

BURY "BLACK

SETSC BLACK
```

The screen color turns black.

If you do a number of these, you don't have to remember color numbers anymore. Use the names.

Why not bury the tools you use regularly? See the Appendix for some tool and utility ideas.

Logo and Artificial Intelligence

The whole idea of artificial intelligence gets very confusing. It makes you wonder, just what is intelligence? And how can it be artificial?

One of the things that makes humans intelligent is the ability to learn. And if we can teach a computer to learn, then maybe it's intelligent?

This is not the place to discuss whether computers can or ever will be able to really learn. Leave that to the computer scientists and philosophers.

For our purposes, computers of today don't really learn. It is the software that computers run that make them appear to

learn. So, if you look at it, the learning is really kind of artificial. Maybe we can call that artificial intelligence.

Have you ever played the game States and Capitals? Someone names a state. You have to name the capital of that state. This edition of the game shows how the computer can appear to be learning.

```
TO STATEQUIZ
GREET
QUIZ
END

TO GREET
CLEARTEXT
PRINT [WHAT'S YOUR NAME?]
PRINT []
MAKE "NAME READLIST
PRINT []
PRINT SENTENCE "HI :NAME
PRINT []
END

TO QUIZ
ASK.CAPITAL
IF CHECKANSWER [QUIZ]
END

TO ASK.CAPITAL
MAKE "ITEM ((RANDOM COUNT :SLIST) + 1)
PRINT SENTENCE [WHAT'S THE CAPITAL OF]
      FIRST SELECT :ITEM :SLIST
END

TO CHECKANSWER
MAKE "CHECK READLIST
PRINT []
```

```
IF EMPTYP :CHECK [OP "TRUE]
IF :CHECK = [QUIT] [OP "FALSE]
TEST :CHECK = LAST SELECT :ITEM :SLIST
IFTRUE [PRINT (SENTENCE [THAT'S RIGHT,]
        :NAME)]
IFFALSE [PRINT (SENTENCE "SORRY :NAME "IT'S
        LAST SELECT :ITEM :SLIST)]
PRINT []
OP "TRUE
END

TO SELECT :N :LIST
TEST :N = 1
IFTRUE [OUTPUT FIRST :LIST]
IFFALSE [OUTPUT SELECT (:N - 1) (BUTFIRST
        :LIST)]
END

TO TEACH
CLEARTEXT
TYPE [WHAT IS THE STATE?]
MAKE "QUEST READLIST
PRINT []
(TYPE SENTENCE [WHAT IS] :QUEST ['S
        CAPITAL?])
MAKE "ANSWER READLIST
MAKE "GROUP []
MAKE "GROUP LPUT :QUEST :GROUP
MAKE "GROUP LPUT :ANSWER :GROUP
CLEARTEXT
PRINT (SENTENCE [THE STATE IS] :QUEST [THE
        CAPITAL IS] :ANSWER)
PRINT []
PRINT [SHALL I ADD THEM TO THE LIST?]
TYPE [(Y OR N) ?]
IF RC = "Y [MAKE "SLIST LPUT :GROUP :SLIST]
PRINT []
PRINT [< < < < NEW LIST > > > >]
```

```
PRINT []
SHOW :SLIST
PRINT []
TYPE [ADD MORE? (Y OR N)]
IF RC = "Y [TEACH] [QUIZ]
END

TO INIT
MAKE "SLIST []
END
```

By now you should have little if any trouble figuring out how this procedure works. Sure, it might take some time. But you can do it.

The main feature of this game is "lists within lists within lists." First, there is the list of States and Capitals — SLIST. If you ever want to erase this list and start over, type INIT.

Secondly, there is a list that matches each state with its capital — GROUP.

Thirdly, there is a list of each state generated from the variable :QUEST and one for each capital that comes from :ANSWER.

Together, these look like this:

```
MAKE "SLIST [[[Oklahoma] [Oklahoma City]]
      [[New York] [Albany]]
      [[Texas][Austin]]
      [[Massachusetts][Boston]]
      [[California][Sacramento]]]
```

Logo "learns" new states and capitals from the TEACH procedure.

The first thing that TEACH does is ask you to create the variables, :QUEST and :ANSWER. It then creates a new empty list named GROUP.

MAKE "GROUP []
Next, it adds the state (:QUEST) to the :GROUP list.

MAKE "GROUP LPUT :QUEST :GROUP

LPUT and FPUT are interesting commands. They are used to add words or other lists to a list. For example:

LPUT "Logo [MSW]
results in the list [MSW Logo].

FPUT "MSW [Logo]
also results in the list [MSW Logo].

In the case of States and Capitals, LPUT tells Logo to add :QUEST at the end of the list :GROUP. Once you have the state listed, you need to add the capital.

MAKE "GROUP LPUT :ANSWER :GROUP

This line adds :ANSWER as the second list within the list :GROUP.

MAKE "SLIST LPUT :GROUP :SLIST

And finally, this line adds the list of two lists to the master list :SLIST.

Time to experiment.

Change the TEACH procedure to add a third element to the GROUP list , maybe the county of the capital or the population of the state?

How would you change the other procedures to ask about that third element?

Go ahead — try it. It's really not that hard.

Rather than look for the LAST element of the GROUP list, you might want to look for the LAST BUTLAST element, or the FIRST BUTFIRST element, or select an ITEM from a list that matches something else.

More AI Applications

Simulations

Before we go too far, there is another realm to explore, the realm of simulations. These are ways to get the computer to act out your "What if?" wishes.

Several years ago, Logy and Morf visited Tombstone, Arizona, the town made famous by the gunfight at the OK Corral. That's where Wyatt Earp and his brothers shot it out with the Clantons.

Everything about the fight has been well-documented including where every one involved was standing, who shot first, who killed who. Billy Clanton was the first to draw his

pistol and fire. But because he was such a poor shot, Wyatt didn't shoot Billy. He shot Billy's brother with his first shot.

Well, Morf asked, "What if Billy hadn't missed?"

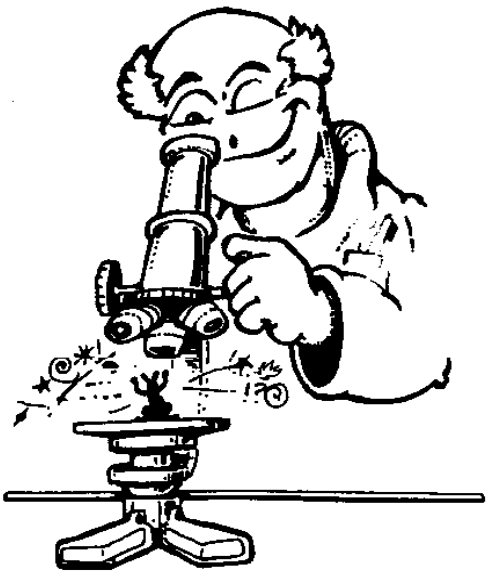
This raised an interesting question. And so the two friends sat down at the computer and plotted the whole thing out. They made a simulation to answer their questions.

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

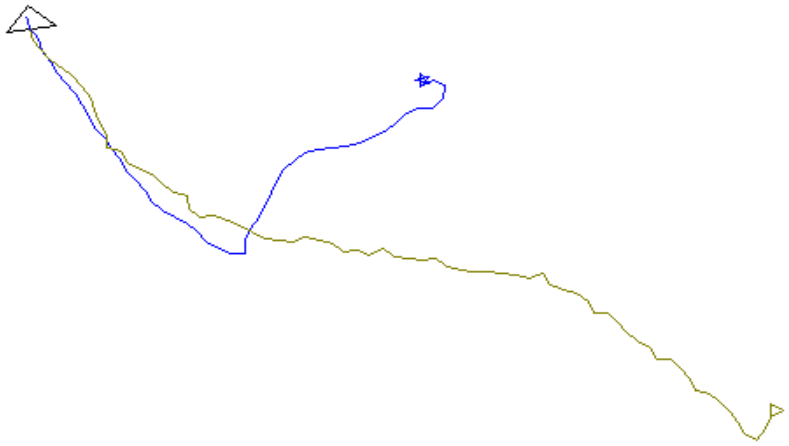
You may have gone to arcades where they have driving or flying games. These are simulators. And, of course, there are the flight simulator software programs you can buy.

Logo Science

Another 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 disk 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?

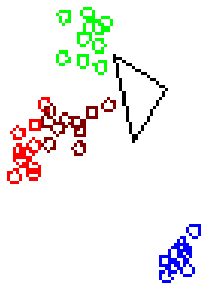
Here's a challenge! BEHAVIOR.LGO uses one turtle to simulate the actions of two. Change the procedures to actually use two turtles.

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.

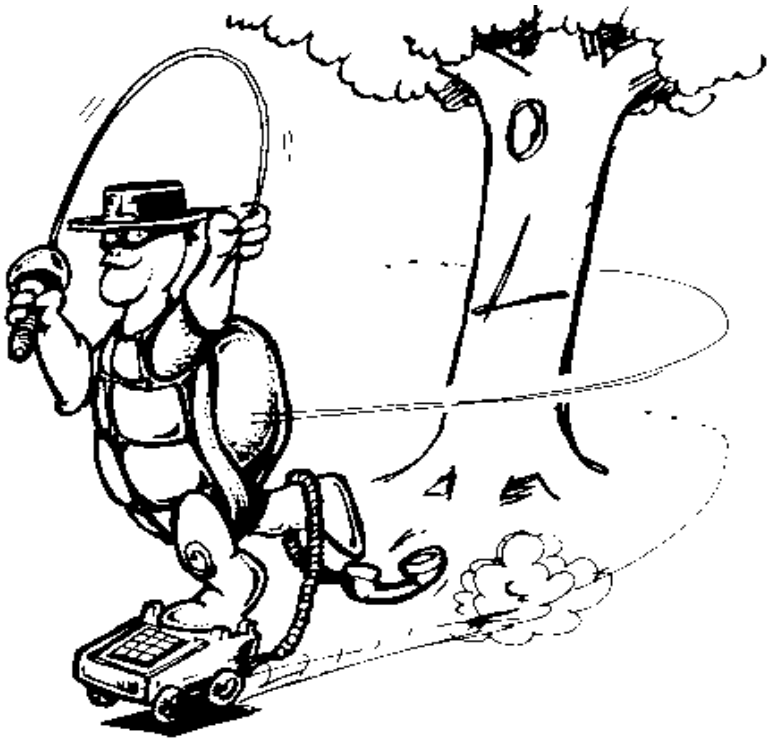
Yes, it can be done. Give it a try.

What's Next

Seems we just got started and here we are. And there are so many other things to do. And what about all the other games and projects?

Well, there has to be something left for you to explore on your own. You can start with the procedures on the diskette that came with this book. This is the part of Logo that Logy and Morf like best, exploring new ways to do things, finding new and better ways to make things work.

In fact, by now you can do just about anything you want with Logo. Doesn't that make you feel great!



You'll find a number of interesting Logo sights on the Internet. There's Logy and Morf's Home Page at

http://www.cyberramp.net/~jmul

Send e-mail to jmul@cyberramp.net. Other addresses include:

http://www.softronix.com

for information on MSW Logo and other products from George Mills.

There is a Logo news group at comp.lang.logo. There's also an on-line Logo mailing list, actually a discussion group. To subscribe, send

subscribe logo-l to majordomo@gsn.org

Most important! What ever you do, enjoy your very own

GREAT LOGO ADVENTURE!


MY JOURNAL
OR HOW I DID IT!