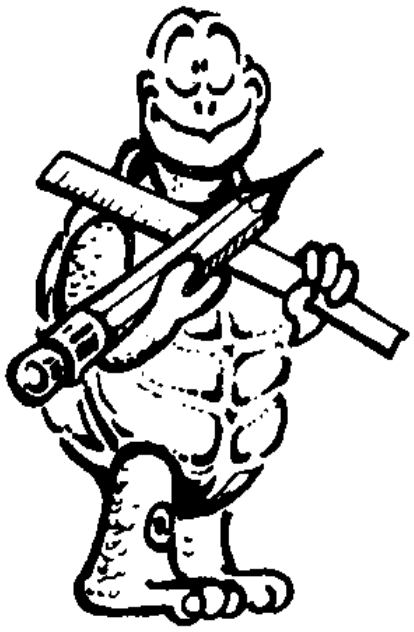
Chapter 9. The Great Math Adventure

"For some reason, when I'm doing my arithmetic homework, it seems more like a pain in the neck than a Great Math Adventure."



True enough! But think about this for a moment! What part of your life does not involve mathematics, some form of calculation, counting, or measurement?

What about time, the counting and measurement of seconds and hours? Or distance, the measurement of space in inches and feet, or centimeters and meters? What about music, measured in frequencies?

Think about Logo and the computer for a moment. Everything you do on the computer is translated into electrical signals that become a mathematical code of zeros and ones.

Everything you have been doing so far in our Great Logo Adventure has been part of turtle geometry, right? I hate to tell you this, but that's mathematics.

Here's something else to think about.

Just what is mathematics? Is it just some number tables that you have to memorize? Is it just a bunch of formulas and equations?

Or is it a way to express ideas and relationships using common symbols such as +, -, *, /, and others? That sounds like a language, doesn't it? Isn't it sort of like Logo? A way

to explore and express ideas on and off the computer? The language that you read on the screen is expressed by numbers, by a mathematical code that is translated into electrical signals. Think about it.

Look beyond your homework into all the realms where numbers and math are used. You'll see a number of examples in this book. On the other hand, talking about Logo, numbers, math, and a universal language is something for a whole new book. With the examples you'll see here, maybe by the time you finish this book, you may just agree that math *is* a universal language.

Come on. Let's start with arithmetic.

Logo Arithmetic

Remember back in the first Rabbit Trail where you took a Turtle Walk. You used the multiplication symbol then.

```
FD 10 * 5
```

What would happen if that line said

```
FD 10 + 5  or

FD 10 - 5  or

FD 10 / 5
```

Try these commands using variables.

```
MAKE "A 50
MAKE "B 100
MAKE "C :A + :B
```

FD :A + :B or FD 100 + 50

FD :C / :A * 10

This one's a bit more interesting.

FD 150 / 50 * 10

That's FD 150 divided by 50 = 3. 3 * 10 = 30 or FD 30.

FD :C / (:A * 3)

Does this instruction do the same thing? Why? Or why not?

When Logo looks at a command that uses arithmetic, it does the arithmetic in the standard mathematical order: multiplication and division followed by addition and subtraction.

So, when Logo reads that line, the first thing it sees is FD :C or FD 150. This is divided by 50 * 3 or 150. So you have FD 150 / 150 or FD 1. Looks like the parentheses change things.

Parentheses are among the Logo delimiters that can be used to change the order of operations.

"Delimiters. That's a funny word!"

It may seem a bit strange. But when you think about it, that's just what they do. Delimiters ***de***fine the ***limits*** of an operation. Take a look.

The commands listed below use arithmetic signs to tell the turtle to go FD 200.

```
FD 100 + 1000 / 10
FD 10 * (5 + 15)
FD (20 - 10) * (18 + 2)
```

Write down some other commands that will take the turtle FD 200. Make sure you use parentheses in your examples. Then test them out.

_____

Positive and Negative Numbers

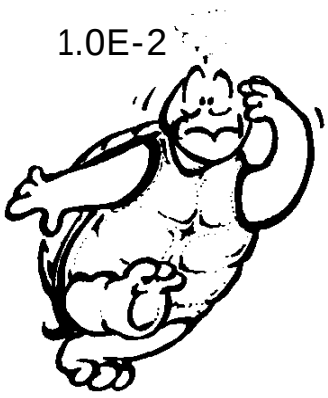
When you add or multiply two positive numbers together, what do you get? You get another positive number. That makes sense, doesn't it?

Add a positive and a negative number together. What do you get? Why not try it and see.

```
SHOW 10 + (-2)
8

SHOW 10 + (-12)
-2
```

What about multiplication?

```
SHOW 10 * -2
-20

SHOW -10 * -2
20
```



Decimal numbers include a part of a whole number, such as 1.25, 3.24, 89.23. Logo lets you use decimals such as

FD 100.125 BK 21.75

Logo writes very big and very small numbers using what they call engineering notation, such as 1.0E-2.

Engineering Notation

COOKIE.LGO is a simple game that adds some fun to mathematics. It's a great test to see who can make the most money selling cookies.

The full procedure is on the diskette that came with this book. Only a few of the subprocedures are listed here. You'll need to look at the whole thing to understand what's going on.

Right now, let's just focus on that strange stuff in the Cost procedures:

```
TO COSTA
OUTPUT 1.E-2 * (19 + RANDOM 7)
END

TO COSTB
OUTPUT 1.E-2 * (12 + RANDOM 8)
END

TO COSTC
OUTPUT 1.E-2 * (9 + RANDOM 7)
END
```

Engineering notation looks strange. That's what they call the output in the Cost procedures above. But it's not all that complicated. It's really pretty easy.

Time to experiment.

Trying playing around with some engineering numbers.

SHOW 1.E + 2 * 9

SHOW 1.E - 5 * 9

SHOW 1.E + 14 * 128

What kind of answer is that? 1.28e+16?

If you play around with engineering notation, you'll discover how it works. Try adding lots of numbers to 1.E. Subtract a bunch also. What happens?

You'll find it's shorthand for writing very big or very small numbers. And soon you'll be able to read them just like you read other numbers.

1.28e+16 is 128 with 14 zeros, sixteen places to the right of the decimal point.

What's SHOW 1.E-14 * 128?

Mathematics doesn't have to be dull, meaningless stuff. It can be fun. It can even get exciting!

Mathematical Operations

There are lots of other ways you can use arithmetic with Logo. Here are the math commands used in MSW Logo.

SUM	DIFFERENCE	MINUS
PRODUCT	QUOTIENT	REMAINDER
INT	ROUND	SQRT
POWER	EXP	LOG10
LN	SIN	RADSIN
COS	RADCOS	ARCTAN
RADARCTAN		

Let's take a look at some examples:

```
FD SUM 50 50
```

What do you think that means? You're right — FD 100. Forward the sum of 50 and 50 or 50 + 50. How about

```
FD DIFFERENCE 300 200
```

Forward the difference between 300 and 200 or 300 - 200.

```
FD PRODUCT 10 10
```

Forward the product of 10 times 10 or 10 * 10.

```
FD QUOTIENT 1000 10
```

Forward the quotient of 1000 divided by 10 or 1000 / 10.

```
FD REMAINDER 1000 300
```

Forward the remainder of 1000 divided by 300. How much is that?

MSW Logo uses INT for integer. But what's an integer? That's a whole number, one without decimals or fractions.

FD INT (121.8 - 21.1)

Forward the integer of 121.8 - 21.1. That equals 100.7. But, since the command is FD INTeger, or whole number, the decimal is dropped.

FD ROUND 121.8 - 21.5

Forward 121.8 - 21.5 rounded off. That equals 100.3, which rounds to 100.

Change that to

FD ROUND 121.8 - 22.1

That equals 99.7, which is rounded to 100.

All these examples would be much simpler if they just said FD 100. After all the arithmetic is done, they each tell Ernestine, the turtle, to go FD 100.

So what?

Well, what if you want to add or multiply a bunch of variables?

FD SUM (PRODUCT :A :X)(QUOTIENT :B :Y)

```
REPEAT (PRODUCT :A :B) [FD SUM :C :D RT 90]
```

You'll see some examples of this type of thing later on.

Factorials

Factorials give you the chance to explore recursion again as well as multiplication. A factorial is the product of all the whole numbers from 1 to whatever.

For example, factorial 5 is just another way of saying
5 * 4 * 3 * 2 * 1 which equals 120

To write that as a procedure:

```
TO FACTORIAL :N
IF :N = 1 [OUPUT :N]
OUPUT :N * (FACTORIAL :N - 1 )
END
```

The easiest way to make sense of this is to use the recursive pages approach like you did with AMAZE in the last chapter.

When you type FACTORIAL 5, this is what Logo sees.

```
TO FACTORIAL 5
IF 5 = 1 [OUPUT 5]
OUPUT 5 * (FACTORIAL 5 - 1 )
END
```

This is saved on the first page as the procedure is called again. This time, it reads

```
TO FACTORIAL 4
IF 4 = 1 [OUPUT 4]
```

OUPUT 4 * (FACTORIAL 4 - 1)
END

This continues until the second line of the procedure reads

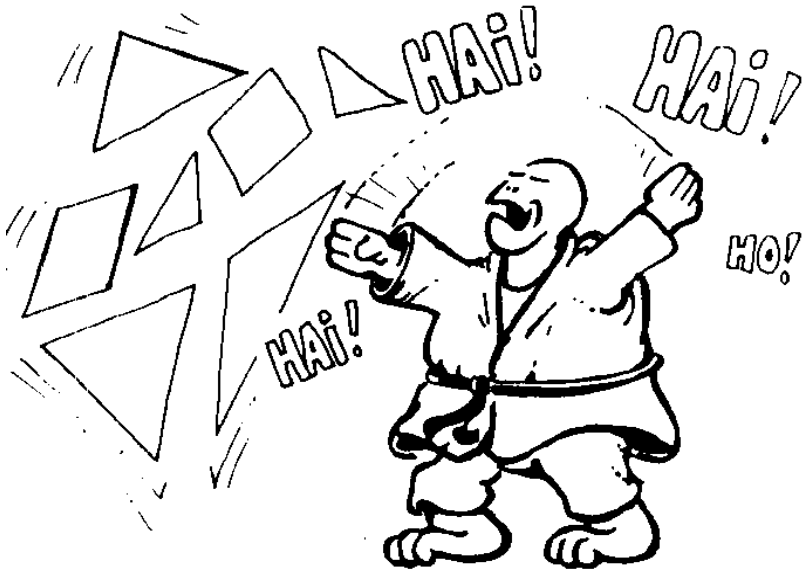
IF 1 = 1 [OUTPUT 1]

Then Logo reads back through the pages:

5 * 4 = 20 * 3 = 60 * 2 = 120 * 1 = 120

The Tangram Procedures

Now let's take a closer look at something else you've seen before. Do you remember the Tangram puzzles? Well, let's take a look at the procedures to draw the Tangram pieces you saw earlier.



```
TO TRIANGLE.RT :SIDE
FD :SIDE RT 135
FD :SIDE / SQRT 2 RT 90
FD :SIDE / SQRT 2 RT 135
END

TO TRIANGLE.LT  :SIDE
FD :SIDE LT 135
FD :SIDE / SQRT 2 LT 90
FD :SIDE / SQRT 2 LT 135
END

TO SQUARE.LT  :SIDE
MAKE "SIDE1 :SIDE / (2 * SQRT 2 )
REPEAT 4 [FD :SIDE1 LT 90 ]
END

TO SQUARE.RT  :SIDE
MAKE "SIDE1 :SIDE / (2 * SQRT 2 )
REPEAT 4 [FD :SIDE1 RT 90 ]
END

TO MED.TRI.RT  :SIDE
FD 2 * (:SIDE / (2 * SQRT 2 ) ) RT 135
FD :SIDE / 2 RT 90
FD :SIDE / 2 RT 135
END

TO MED.TRI.LT  :SIDE
FD 2 * (:SIDE / (2 * SQRT 2 ) ) LT 135
FD :SIDE / 2 LT 90
FD :SIDE / 2 LT 135
END
```

```
TO SMALL.TRI.RT  :SIDE
FD :SIDE / 2 RT 135
FD (:SIDE / SQRT 2 ) / 2 RT 90
FD (:SIDE / SQRT 2 ) / 2 RT 135
END

TO SMALL.TRI.LT  :SIDE
FD :SIDE / 2 LT 135
FD (:SIDE / SQRT 2 ) / 2 LT 90
FD (:SIDE / SQRT 2 ) / 2 LT 135
END

TO PARGRAM.LT  :SIDE
REPEAT 2 [FD :SIDE / (2 * SQRT 2 ) LT 45 ~
  FD :SIDE / 2 LT 135 ]
END

TO PARGRAM.RT  :SIDE
REPEAT 2 [FD :SIDE / (2 * SQRT 2 ) RT 45 ~
  FD :SIDE / 2 RT 135]
END
```

To give you an idea of what you can do with tangram shapes, try this procedure.

```
TO TANGRAM  :SIDE
SETH 90 TRIANGLE.LT :SIDE
FD :SIDE SETH 0
TRIANGLE.LT :SIDE
FD :SIDE SETH 270
SMALL.TRI.LT :SIDE
FD :SIDE / 2 SETH 225
MED.TRI.RT :SIDE
SQUARE.LT :SIDE FD :SIDE1
```

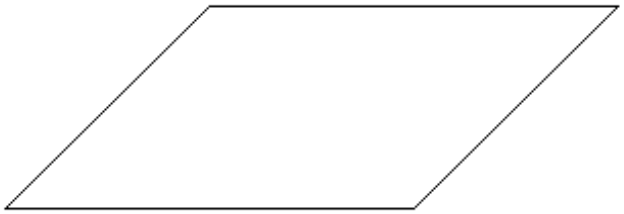
```
PARGRAM.LT :SIDE
END
```

This procedure uses the different shape procedures to make one big shape. Which one?

You'll have to run the procedure to see.

What's a Parallelogram

"Logy, there's something strange here? You've got a rectangle that looks like it's falling over."



"You're right, you know. I never thought of a parallelogram like that," said Logy.

"Para-who?"

"That's another shape, a parallelogram. You might call that the granddaddy of a square," Logy answered.

"I don't get it? What do you mean, granddaddy?"

"Take a look at this procedure. It's called PARGRAM for short. It's a bit different from the one in the TANGRAM procedure."

```
TO PARGRAM  :SIDE :INC :ANGLE
REPEAT 2 ~
 [
  FD :SIDE RT :ANGLE
  FD :SIDE / :INC RT 180 - :ANGLE
 ]
END
```

Go ahead and try this:

PARGRAM 100 2 45

Look familiar? It's draws a parallelogram just like the one above.

"What would happen if I changed the angles from RT 45 to RT 90?"

"Hey, that would be a rectangle," said Morf, jumping up and down excitedly.

"So, you can say that a rectangle is sort of like the 'child' of a parallelogram. A parallelogram can take many shapes, one of which is a rectangle."

"Now look at the sides. You've got two that are the length of :SIDE and two that are the length of :SIDE divided by :INC. What would happen if you changed :INC to 1?"

"Well, let's see. You'd have two sets of sides that all use the same variable. That means they'd all be the same."

"And if all the angles are RT 90, what's that?"

"Hey, that's a square!"

"OK, then. Is it fair to say that a square is the child of a rectangle?"

"Seems that way."

"Right! So now we can add a new shape rule."

- A square has four equal sides and four equal angles.
- A rectangle has two sets of equal sides and four equal angles.
- A parallelogram has two sets of equal sides and two sets of equal angles.

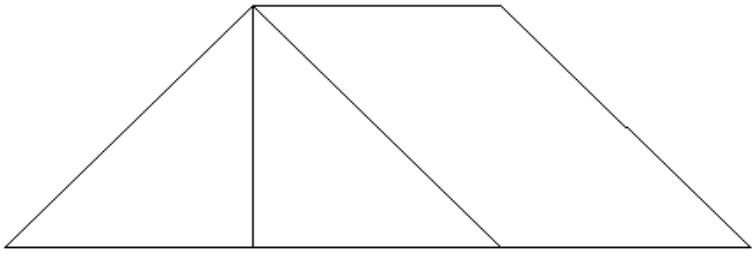
Fun With Tangrams

Enough of this stuff. Let's have some more fun with tangrams! Put two small triangles together. What shape do you get? Can you make a square from the two small triangles? How about a larger triangle? A parallelogram?

Put the parallelogram and two small triangles together. What shape is that? Can you make a square? What about a trapezoid?

"A what?"

"A Trap-e-zoid! That's another shape, Morf. It's like a parallelogram but it only has one set of parallel sides instead of two."



"That's no trap-e-whatever. That's a picture of the pup tent we use out in the back yard!"

"Get serious, Morf. Can you make a triangle using five pieces of the puzzle?

You'll find lots of puzzle books that have tangrams in them. But you don't really need those books, do you? I'll bet you can think up all sorts of shapes on your own.

Making Crazy Shapes

Why not have the computer think up some shapes for you?

These might come out a bit crazy. But who cares? That's the fun of having the turtle do things for you. That's why CRAZY.SHAPES was included in the TANGRAM.LGO procedure.

```
TO CRAZY.SHAPES  :SIDE
SHAPES :SIDE
MOVE :SIDE
CRAZY.SHAPES :SIDE
END

TO SHAPES :SIDE
MAKE "SHAPE RANDOM 10
IF :SHAPE = 0 [TRIANGLE.RT :SIDE]
IF :SHAPE = 1 [TRIANGLE.LT :SIDE]
IF :SHAPE = 2 [MED.TRI.RT :SIDE]
IF :SHAPE = 3 [MED.TRI.LT :SIDE]
IF :SHAPE = 4 [SMALL.TRI.RT :SIDE]
IF :SHAPE = 5 [SMALL.TRI.LT :SIDE]
IF :SHAPE = 6 [SQUARE.RT :SIDE]
IF :SHAPE = 7 [SQUARE.LT :SIDE]
IF :SHAPE = 8 [PARGRAM.RT :SIDE]
IF :SHAPE = 9 [PARGRAM.LT :SIDE]
END
```

```
TO MOVE :SIDE
MAKE "MOVE INT RANDOM 5
IF :MOVE = 0 [SETH HEADING + 45]
IF :MOVE = 1 [SETH HEADING + 90]
IF :MOVE = 2 [FD :SIDE]
IF :MOVE = 3 [FD :SIDE / 2]
IF :MOVE = 4 [FD (:SIDE / SQRT 2 ) / 2]
END
```

RANDOM, RERANDOM, Picking, and Shuffling

There's that RANDOM command again. It takes one input. Remember that in MSW Logo, RANDOM selects a number between zero and the number you input. Try this:

```
REPEAT 20 [SHOW RANDOM 10]
```

You'll see a list of 20 numbers randomly selected from the a list of numbers between 0 and 9.

There may be times when you want to randomly select a sequence of numbers and then use that same sequence over again. Ordinarily the sequence of random numbers is different each time Logo is used. However, if you need the same sequence repeatedly, type RERANDOM before you use the RANDOM primitive. For example:

```
RERANDOM REPEAT 2 [SHOW RANDOM 10]
9
3

RERANDOM REPEAT 2 [SHOW RANDOM 10]
9
3
```

When you need different sets of random numbers, you can give RERANDOM a seed number as input. This is simply a number that identifies the sequence you're working with. For example:

```
(RERANDOM 1234)
REPEAT 2 [SHOW RANDOM 10]
6
2

(RERANDOM 1234)
REPEAT 2 [SHOW RANDOM 10]
6
2
```

If you want to change the sequence, simply change the seed number.

```
(RERANDOM 4321)
REPEAT 2 [SHOW RANDOM 10]
9
3
```

There are times that you want to randomly rearrange a list, like a sequence of numbers, letters, cities, or whatever — like shuffling a deck. To do that, you have to give RANDOM a little help.

Here's a procedure to do just that. It has some new commands that you'll get into a bit later. But since we're talking about RANDOM and RERANDOM, go ahead and try it out now. You'll find lots of uses for it.

```
TO SHUFFLE :DECK
MAKE "X []
```

```
REPEAT COUNT :DECK ~
   [CHECK MAKE "DECK BUTFIRST :DECK]
REPEAT (RANDOM 4) ~
   [MAKE "X LPUT FIRST :X BUTFIRST :X]
OP :X
END

TO CHECK
IFELSE (RANDOM 3) = 1 ~
  [MAKE "X FPUT FIRST :DECK :X] ~
  [MAKE "X LPUT FIRST :DECK :X]
END
```

Why not see what happens when you deal a shuffled list of shapes?

```
TO DEAL
CS
MAKE "LIST SHUFFLE [SQ TRI HEX]
RUN :LIST
END

TO HEX
REPEAT 6 [FD 50 RT 60] WAIT 60
END

TO SQ
REPEAT 4 [FD 50 RT 90] WAIT 60
END

TO TRI
REPEAT 3 [FD 50 RT 120] WAIT 60
END
```

Now that we've confused you with this procedure, let's confuse you even more. There's another way to shuffle things around. You saw that in the SHAPES :SIDE procedure.

```
TO SHAPES :SIDE
MAKE "SHAPE RANDOM 10
IF :SHAPE = 0 [TRIANGLE.RT :SIDE]
IF :SHAPE = 1 [TRIANGLE.LT :SIDE]
  on through...
IF :SHAPE = 9 [PARGRAM.LT :SIDE]
END
```

In effect, this procedure shuffles the Tangram procedures. It makes the variable :SHAPE a random number. It then matches the random number with the conditional statement to find a procedure to run.

Using SHUFFLE, the whole thing would be a bit easier.

One last point of confusion. There's a PICK command that randomly selects an element from a word or list. For example:

```
MAKE "CHOICES [A B C D E]
SHOW PICK :CHOICES
D
```

Go ahead and explore how you could use all these choices. They all do things a bit differently. So you've got lots of choices for whatever you want to do.

Squares and Square Roots

Look back at the TRIANGLE.RT procedure. Got any idea what the SQRT 2 means?

That number is used to figure out how long the two short sides of the triangle are. The left side is the longest side, right? And you know that you have two equal sides connected by an angle of 90 degrees.

A long time ago, some mathematician figured out that when you know the long side of a triangle that has two equal sides and a right angle, then the short sides equal

<*Long side*> / SQRT 2

There are lots of rules like this for triangles and other shapes. You've already figured out a bunch of them.

"But what does SQRT 2 mean?"

Actually, it stands for the square root of 2. That sounds a lot worse than it really is. It doesn't have anything to do with the square shape. It's part of an arithmetic problem that asks what number, multiplied by itself, gives you the answer of 2.

What's SQRT 100? SQRT 9? SQRT 16?

Think about it for a minute. What number multiplied by itself equals 100?

10 * 10 = 100

What number multiplied by itself equals 9?

3 * 3 = 9

What number multiplied by itself equals 16?

4 * 4 = 16.

Now let's turn the square around. Square roots are like saying, "Here's the answer. Tell me what the question is. Here's 16, tell me what number multiplied by itself gives me that answer?"

So let's take a look at another question: 4 multiplied by itself equals what? MSW Logo has a POWER command.

FD POWER 4 2

That's like saying forward 4 to the power of 2, or 4 squared, or 4 times 4.

FD POWER 10 3

This is like saying Forward 10-cubed or 10 * 10 * 10 or 10 to the power of 3. The 2 and the 3 are called "exponents." This might make you think that POWER and EXP (or EXP) are the same. EXPN is a trigonometric function that calculates the natural base *e* (2.7183. . .) raised to the power specified by its input.

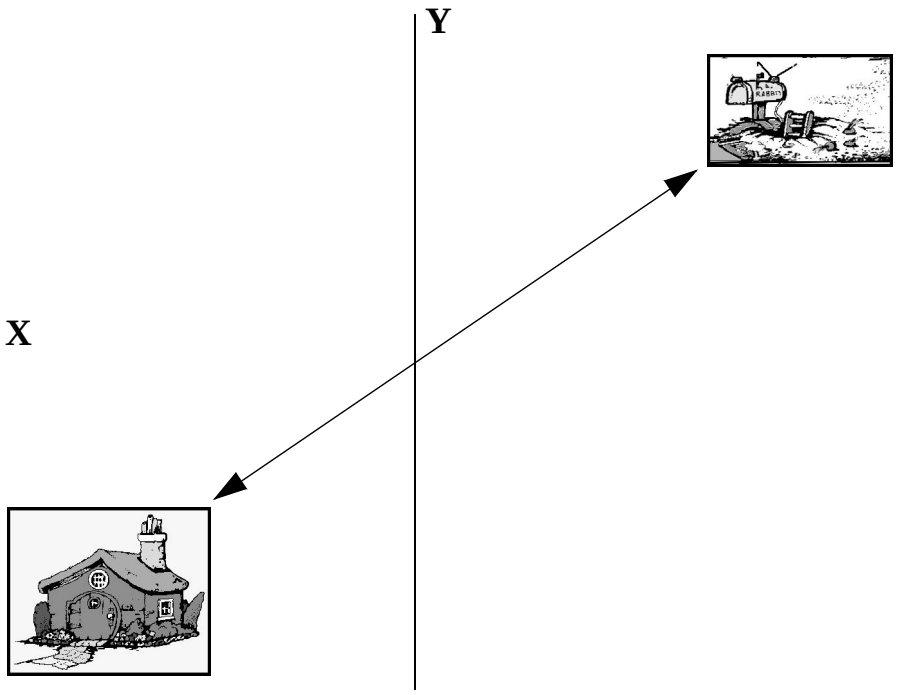
"I didn't understand a word of that."

Walking Distance

Let's first do some coordinate calculations to follow up on a Chapter 7 project.

Do you remember in Chapter 1 when you explored Turtle Town? You were asked to draw a map to your friend's house.

Now let's figure out the distances between two houses. Let's use Logy's and Morf's homes.



If you know the coordinates of each home, it's easy to use Logo and trigonometry to calculate the distance between the two homes. Here's a variation of the DISTance procedure.

```
TO DIST  :X1 :Y1 :X2 :Y2
OP DIST1 :X1 - :X2 :Y1 - :Y2
END

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

Gee, when you look at those procedures, it seems as if trigonometry has something to do with coordinates — which, of course, it does. Does that DIST1 procedure look familiar?

```
SQRT (:DX * :DX ) + (:DY * :DY)
```

is the same as

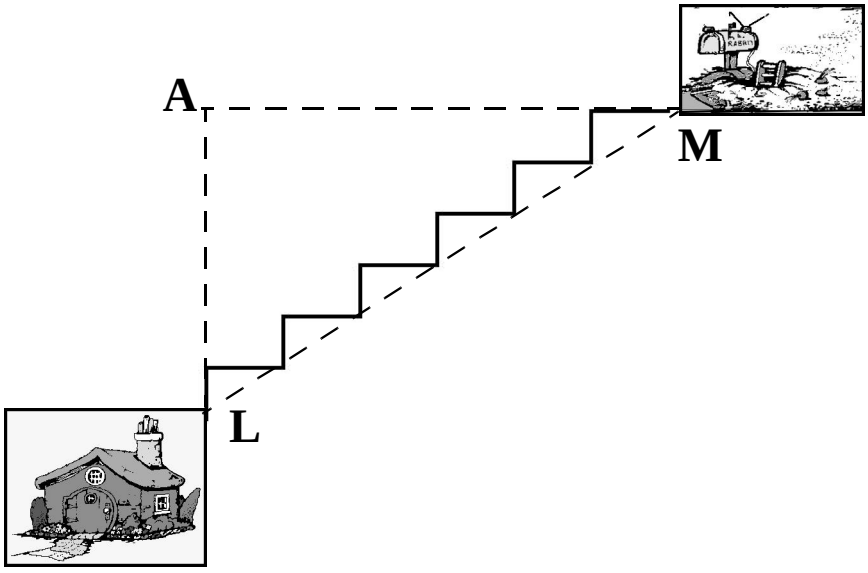
SQRT :DX^2 + :DY^2

which looks just like the Pythagorean theorem, which says that the hypotenuse squared equals the sum of the adjacent and opposite sides squared. You'll read moe about this theorem in the next section that's on Trigonometry.

Logy's home is at [-100 -80] and Morf's Den is at [150 60]. What's the distance between the two homes?

SHOW DIST -100 -80 150 60
286.530975637888

This answer is fine if you're flying between homes. But since you're walking, you've still got some figuring to do.



Walking between the two homes means that you have to go up one block, turn right, go another block, turn left, and so forth. That's not the same as the straight line distance from L to M.

So how can you calculate the walking distance?

1. Start with what you know.
2. Define what you need to know.
3. Go find it.

You know the coordinates of each home, and you calculated the distance between the two homes.

- Logy's Home is at [-100 -80].
- Morf's Home is at [150 60].
- The direct distance between the two homes is 286.53.

You also know that turn from one block to the next, that the total distance you must walk going North (top of the screen) is the distance from L to A. The total distance you will walk going East (right side of the screen) is the distance from A to M.

Think about this. You also know the coordinates of point A. Take the Y coordinate of Morf's home and the X coordinate of Logy's home, and where are you?

Point A is at [-100 60].

The easy way to calculate the walking distance would be to use the DIST procedure. Rather than do that, let's write a procedure to calculate the total walking distance.

```
TO WALK.DIST
MAKE "MX 150
MAKE "MY 60
MAKE "LX -100
MAKE "LY -80
MAKE "WALK ABS (:LX - :MX) + (:LY - :MY)
(PR [THE WALKING DISTANCE IS] :WALK)
END
```

The walking distance is 390 steps.

You can use the DIST procedure to prove this. First calculate the distance from Logy's Home to A.

```
DIST -100 -80 -100 60 = 140
```

Now from Point A to Morf's Home.

```
DIST -100 60 150 60 = 250
```

One last test: does 286.53 squared equal 140 squared plus 250 squared.

```
SHOW SQRT 140 * 140 + 250 * 250
286.530975637888
```

Works for me!

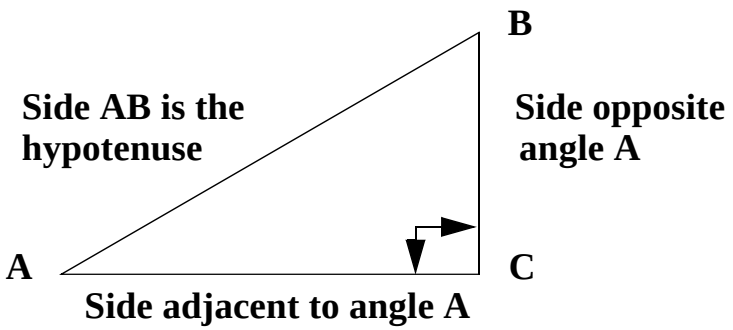
A Quick Look at Trigonometry

"You know, you can't really get away from trigonometry if you're going to explore angles. In simplest terms, trigonometry is the study of triangles. And what's a triangle other than three connected angles?"

Trigonometry — *trig* for short — also includes the study and use of what they call *trigonometric functions*. These functions include strange names such as sine, cosine, tangent, arctangent, and cotangant. They describe the functions of an angle that is described in terms of the ratios of pairs of sides or angles in a right triangle. You remember the right triangle, don't you?

A right triangle includes one right angle — right angles are 90 degrees — and two acute angles. Acute angles are less than 90 degrees. Obtuse angles are more than 90 degrees.but less than 180 degrees.

So here's a right triangle:



A gentleman for Ancient Greece named Pythagoras came up with the rule that lets you determine the length of any side of a right triangle if you know the length of any two sides. The relationship that Pythagoras came up with says that side AC squared plus side BC squared equals the hypotenuse, side AC, squared.

$$A^2 + B^2 = C^2$$

Defining Trig Functions

Once you know how the different sides of a right triangle relate to each other, you can define the trigonometric functions in terms of their right triangle relationships. Using the angles A, B, and C in the right angle shown on the last page, here's the definitions of functions with MSW Logo commands:

Sine (sin):	Sine of angle A = the ratio between the side opposite angle A and the hypotenuse. BC / AB.
Cosine (cos):	Cosine angle A = the ratio between the adjacent side and the hypotenuse. AC / AB

Tangent (tan): Tangent angle A = the ratio between the opposite side and the adjacent side. BC / AC

Arctangent: (Arctan) Arctangent angle A = The inverse of the tangent or the ratio between the adjacent side and opposite side. AC / BC

Using this information, you can define just about any trig function you may need. It also may help you understand some of the more complex procedures and commands in MSW Logo.

There are many books on trigonometry, if you really want to dig into it. There are also books on advanced Logo that talk about it much more than space allows in this book. For now, let's take a quick look at trigonometry in action.

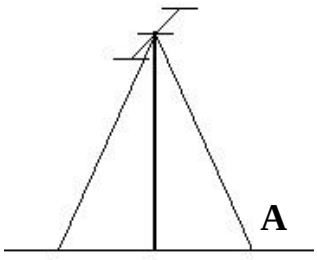
Morf's TV Antenna

Here's a problem that uses the sine function.

Morf has a TV antenna that is twelve feet tall. The instructions say that to keep it steady, he has to use four guy wires, each at a 65 degree angle from the ground.

How much wire does he need to buy?

Here's what the antenna looks like. Just remember that there are four wires and not just two as shown here.



The sine of angle A equals the opposite side (the height of the antenna) divided by the hypotenuse (the length of the wire).

SINE A = 12 / WIRE LENGTH
or
TOTAL WIRE = (12 / SINE A) * 4

In Logo, that's written as

```
SHOW (12 / SIN 65) * 4
52.9621401101996
```

Looks like Morf needs about 13-1/4 feet of wire for each of the four guy wires.

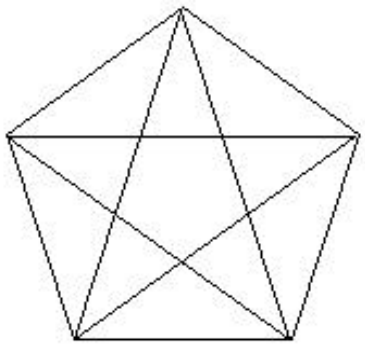
Side of the Star

Do you remember the problem with the stars back in Chapter 6? You started with a pentagon like this:

```
REPEAT 5 [FD 100 RT 72]
```

How did Ernestine know that the side of the star was 160? Maybe you can figure it out now.

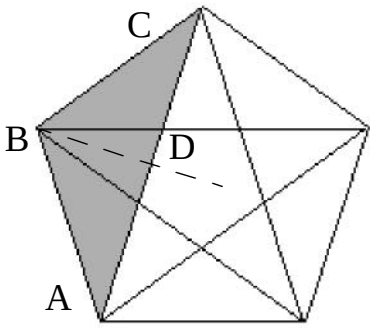
What do you know about that pentagon and the star?



The sides of the pentagon are 100.

The three angles at each corner equal 36 degrees.

Now, how can you prove that the long side of the shaded triangle is 160?



From HOME in the lower left, type

LT 18 FD 100 RT 108 / 2 FD 80

You already know that the big angle in the shaded triangle is 108. In Chapter 6, you proved that it's 180 - 72. So you cut that angle in half and go FD 80 — any distance, just so long as you cross the long side of the triangle.

You're left with two right angles, ABD and CBD.

Now you can calculate half of the long side of the shaded triangle using this trigonometric function.

COSINE 36 = ADJACENT / 100 (hypotenuse)

To put this in Logo terms:

SHOW 100 * COS 36 or
80.9016994374947

Now let's do some checking.

Using the Pythagorean theorem of $A^2 + B^2 = C^2$, you can calculate the length of the short side of the right triangle BD.

$100^2 = 80.9^2$ + What?

```
SHOW SQRT (POWER 100 2) - (POWER 80.9 2)
58.7808642331839
```

To check this out, try this:

```
REPEAT 5 [FD 162 RT 144]
LT 36
REPEAT 5 [FD 100 RT 72]

FD 100 RT 72 + 54  (Half the inside angle of 108)
FD 58.78
RT 90
FD 80.9
```

Where are you?

The length of the long side of the shaded triangle is 80.9 * 2 or 161.8, which is just about 162. Ernestine likes to work with rounded numbers. So now you know where she got it.

This is a bare taste of trigonometry. But maybe as you look at some of the procedures in the Logolib directory and elsewhere, they won't seem quite so strange now.

You never know. You might just be able to figure them out and do some really neat things with them.

Here's a challenge for you. Do you remember the From the Center exercises. In Chapter 7, you found the center point of different polygons. Now that you've had some experience with trigonometry, can you find out the distance from the edge to the center of any of the shapes?

Counting Numbers and Stuff

COUNT is another very useful Logo command. It outputs the number of elements in its input. That input can be a word or a list.

```
SHOW COUNT “LOGO
4   (There’s four letters in LOGO.)
SHOW COUNT [LOGY AND MORF]
3   (There’s three words in the list.)
```

Here’s an example from a procedure we talk more about in Chapter 11. Since you’re only interested in the first command, the one with COUNT in it, we left that second part off . Let’s see if we can use the REPEAT command to help make some sense out of COUNT.

```
REPEAT ( COUNT :NUMS2 ) - 1  [...
```

You have a variable named :NUMS2. So for our explorations, let’s make :NUMS2 equal to a list of numbers.

```
MAKE "NUMS2 (LIST 22 11 30 567 982)

SHOW :NUMS2
Result: [22 11 30 567 982]

SHOW COUNT :NUMS2
Result: 5

REPEAT ( COUNT :NUMS2 ) - 1 [FD 100 RT 90]
```

What would this command draw? You should know that. You learned about this shape back in Chapter 2.

Items, Members, and Things

There are some other neat things you can do with words and lists. In the example above, you used the COUNT of the variable :NUMS2 to create a square. You can also select an item from a word or list and use that, too.

Here's an example. I bet you can guess what this is going to look like. It also tells you what ITEM does in a Logo procedure.

```
REPEAT ITEM 3 :NUMS2 [SQUARE RT 12]

TO SQUARE
REPEAT ( ( COUNT :NUMS2 ) - 1 ) [FD 100 RT 90]
END
```

What do you think ITEM 3 :NUMS2 is?

You know that :NUMS2 is a list — [22 11 30 567 982]. So what is ITEM 3 :NUMS2?

Another double-dip ice cream cone if you said 30.

ITEM outputs the third element of the variable :NUMS2. It doesn't matter whether the variable is a word or a list.

```
SHOW ITEM 2 "CAT
  A

SHOW ITEM 2 7861236
  8
```

Get the idea?

In the :NUMS2 example, you knew what NUMBER you were looking for — the third element, or 30. But what if you didn't know?

Logo lets you ask. Take a look.

```
TO CHECK :X
IFELSE MEMBERP :X :NUMS2~
  [REPEAT ITEM 3 :NUMS2~
  [SQUARE RT 12]] [SQUARE]
END

TO SQUARE
REPEAT ( ( COUNT :NUMS2 ) - 1 ) [FD 100 RT 90]
END

MAKE "NUMS2 [22 11 30 567 982]
```

In the CHECK procedure, Logo asks if :X is a member of the variable :NUMS2. If it is, it runs the REPEAT command

```
[REPEAT ITEM 3 :NUMS2 [SQUARE RT 12]]
```

If not, it just runs the SQUARE procedure.

Logo picks up these instructions from the IFELSE command. It's like saying if a condition is true, then do this, or else do this.

Ask Logo

You just saw MEMBERP. Well, there are a number of other questions you can ask Logo.

EQUALP

Are two words or lists equal or identical? For example:
IF EQUALP :X (ITEM 3 :NUMS2) [REPEAT …

EMPTYP

Is a word or list empty (") or ([ ])? For example:
IF EMPTYP :NUMS2 [STOP] [REPEAT …

If :NUMS2 is an empty list STOP, else run the REPEAT command.

NUMBERP

Is the object a number? For example:

IF NUMBERP (ITEM 3 :NUMS2) [REPEAT…

If ITEM 3 :NUMS2 is a number, then continue with the REPEAT command. Otherwise skip it and go on to the next line.

WORDP

Is something a word.

LISTP

Is something a list?

Logical Operations

There are three other primitives you need to look at before you leave Logo arithmetic: AND, OR, NOT.

AND

AND tests to see if all the conditions following the command are true.

```
MAKE "X 210
MAKE "Y 724
MAKE "Z 910
IF AND :X > 200 :Y < 800 [FD 100]
```

The conditions are true so the turtle moves forward 100.

Where you have more than two conditions, the commands and the command AND must be enclosed in parentheses.

```
MAKE "Z 555
IF (AND :X > 200 :Y < 800 :Z >500) [FD 100]
```

The conditions are true so the turtle moves forward 100.

OR

Where AND tests if all conditions are true, OR tests to see if any of the conditions is true. If you have more than two conditions to test, use parentheses as shown below.

```
IF OR :X > 200 :Z >1000 [FD 100]
IF (OR :X > 200 :Y < 800 :Z >1000) [FD 100]
```

Because at least one of the conditions is true, the turtle moves forward 100.

You could rewrite the Serpinski curve procedure from the last chapter using OR by changing

```
IF :LEVEL = 0 [FD :SIZE STOP]
IF :SIZE < 2 [FD :SIZE STOP]
```

to

```
IF OR (:LEVEL = 0)(:SIZE < 2) [FD :SIZE STOP]
```

NOT

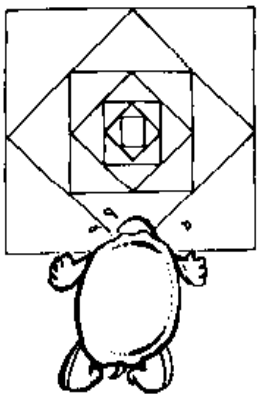
If the condition is false, NOT outputs True. In other words:

```
IF NOT :Z > 1000 [FD 100]
```

Since Z is *not* greater than 1000, the turtle moves forward 100.

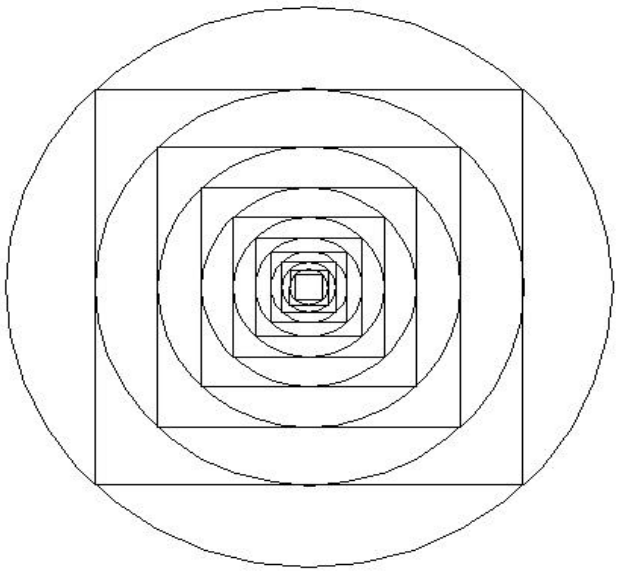
Math Challenges

Math Challenges may sound a bit like homework, but these problems are fun!



Here's a fairly easy challenge for you. Draw a series of squares within squares like the ones that are causing Logy to shake her head. It's tricky, sure. But it's really not that tough.

Here's another bit of a challenge to see what you've learned so far.



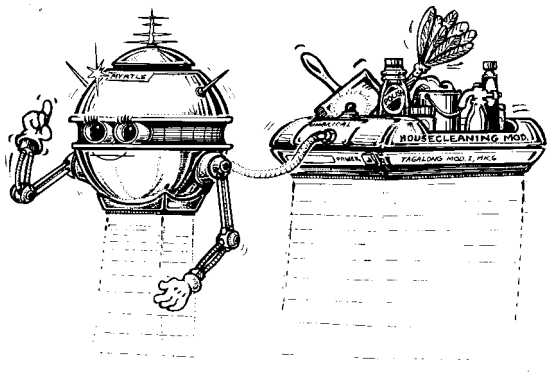
That drawing is a Mandala. People in India believe this is a symbol of the endless universe. Take some time to figure out how this procedure. It's an interesting exercise in turtle geometry.

```
TO MANDALA :RADIUS :CENTER
CIRC
SQUARE
IF :RADIUS < 10 [STOP]
MANDALA :RADIUS :CENTER
END

TO SQUARE
SETH 225 FD :RADIUS SETH 0
MAKE "SIDE SQRT (2 * (:RADIUS * :RADIUS))
PD REPEAT 4 [FD :SIDE RT 90]
MAKE "RADIUS :SIDE / 2
END

TO CIRC
PU SETXY :CENTER
SETX XCOR - :RADIUS
PD CIRCLER :RADIUS
PU SETXY :CENTER
END

TO CIRCLER :RADIUS
LOCAL "STEP
MAKE "STEP 2 * :RADIUS * 3.1416 / 36
REPEAT 36 [RT 5 FD :STEP RT 5]
END
```

OK. Now that you've got the Mandala procedure all figured out, try doing the same thing using triangles instead of squares.

Number Systems

Converting numbers from one number system to another is something better left to Mrtle, the affable robot who never learned to speak computerese.



However, since number systems ar part of The Great Math Adventure, it might be fun to explore a procedure that converts numbers from one number system to another. It's called CONVERT.LGO. To run it, type something like

SHOW (or PRINT) CONVERT 12 10 2

In other words, convert 12 from base 10 to base 2. Base 10 is the standard decimal number system people use in day-to-day living. Base 2 is the binary number system that computers use.

```
TO ANYBASE.TO.DEC :N :BASE :POWER
IF EMPTYP :N [OP 0]
OP (:POWER * C.TO.N LAST :N) + ~
   ANYBASE.TO.DEC BL :N :BASE :POWER * :BASE
END

TO C.TO.N :N
IF NUMBERP :N [OP :N]
OP (ASCII :N) - 55
END
```

```
TO CONVERT :N :FRBASE :TOBASE
OP DEC.TO.ANYBASE ANYBASE.TO.DEC :N~
  :FRBASE 1 :TOBASE
END

TO DEC.TO.ANYBASE :N :BASE
IF :N < :BASE [OP N.TO.C :N]
OP WORD DEC.TO.ANYBASE INT QUOTIENT :N~
  :BASE :BASE N.TO.C REMAINDER :N :BASE
END

TO N.TO.C :N
IF :N < 10 [OP :N]
OP CHAR 55 + :N
END

TO DIVISORP :A :B
OP 0 = REMAINDER :B :A
END
```

Two sets of procedures were tacked on at the end. Two convert numbers from base 10 to base 16 and two convert numbers from base 10 to binary numbers (base 2) and back.

```
TO HEXTODEC :N
OP CONVERT :N 16 10
END

TO DECTOHEX :N
OP CONVERT :N 10 16
END

TO BINTODEC :N
OP CONVERT :N 2 10
END
```

```
TO DECTOBIN :N
OP CONVERT :N 10 2
END
```

Two others that might be useful are

```
TO OCTTODEC :N
OP CONVERT :N 8 10
END

TO DECTOOCT :N
OP CONVERT :N 10 8
END
```

Why add octal numbers? Because computers (especially older personal computers) use binary, octal, and hexadecimal numbers.

Logo Physics

Physics? This is supposed to be a math adventure, not science!

You're about to see how science and math work together. Did you ever play a lunar module game, where you land a spaceship on the moon or another planet?

You were dealing with physics.

Did you ever play an artillery game where you fired your cannon at the enemy?

You were working with physics.

The laws of physics describe how things work, how planets stay in orbit, how different types of force work including the force of gravity.

Most of physics is a bunch of mathematical equations. Unless you're a mathematician, those equations don't mean much. You need to see physics in action to understand how things work. And if you're going to see things in action, why not have some fun doing it!

Logo Gravity

When you drop a stone from a tall tower, gravity pulls it to the ground. It accelerates as it falls, moving faster and faster.

Then splat! It hits the ground.

Just what is acceleration? What's the difference between acceleration and speed? What's the difference between speed and velocity, if any?

Speed and velocity mean the same thing. Morf uses the term Speed because it's easier to spell.

Speed measures the rate at which your position changes. If you run 100 yards in 10 seconds, your rate of speed is

Distance/Time = Speed

100 yards/10 seconds = 10 yards per second

This is your average speed over the 100 yard track. But you started at 0 yards per second. Let's take what we know and figure the acceleration.

Acceleration

Speed measures the rate at which your position changes. Acceleration measures the rate at which the speed changes.

Change in speed/Time = Acceleration

Another way of saying this is

Final speed - Beginning speed/Time = Acceleration

Let's say that the speed crossing the finish line was 50 yards per second. This gives us

50 yards per sec. - 0 yards per sec. / 10 seconds = an acceleration of 5 yards per second per second.

What this says is that your speed increases by 5 yards per second during every second you travel.

- Just before you start, your speed is 0 yards per second.
- At the end of the first second, it's 5 yards per second.
- At the end of the second, it's 10 yards per second.
- At the end of the third, 15 yards per second.
- At the end of the fourth, 20 yards per second.

The speed changes by 5 yards per second giving you an acceleration of 5 yards per second per second. Get it?

Now let's put this information into a procedure to demonstrate what acceleration is all about. When you drop a rock from a tower, the laws of physics tell us that the acceleration is going to be 32 feet per second per second.

That's too fast to see on the computer. So we're going to slow this down a lot. If this is too slow or too fast for your computer, you can change it. In the FREEFALL procedure, there's a line that says

MAKE "ACC 0.2

Change that 0.2 to whatever works well on your computer You might want to do procedures that simulate the gravity on different planets. How would you do that? What information do you need?

```
TO GROUND
HT PU SETPOS [-100  -200] PD
SETPOS [100 -200]
PU HOME PD
END

TO FREEFALL :HEIGHT
CS PD GROUND PU SETY :HEIGHT
MAKE "VEL 0
MAKE "ACC 0.2
MAKE "TIME 0
ST PD
ACCELERATE
PRINT "
PRINT [The turtle has landed.]
END

TO ACCELERATE
(PR "TIME :TIME "VELOCITY :VEL "DISTANCE
  (:HEIGHT - YCOR ) )
BK :VEL + :ACC / 2
MAKE "VEL :VEL + :ACC
MAKE "TIME :TIME + 1
IF YCOR < -200 [STOP]
ACCELERATE
END
```

To run the procedure, type FREEFALL and a height from which the turtle will fall. Now, how can you make good use of this procedure? Can you make up a game involving FREEFALL?

Artillery Practice

One of the first, if not the very first computer game was an artillery game played long before personal computers were even thought of. Here's a procedure that describes the basic movement of an artillery shell. (Because SHELL is a command, we use the term SHEL.)

To fire an artillery shell, you need to know three things: the velocity of the shell out of the gun barrel, the angle at which it was fired, and the local gravity (or acceleration) factor. This gets more complicated than we need to worry about here. Let's just say that as you go higher, the effects of gravity get smaller.

Try this to start:

SHEL 10 45 0.4

Then experiment with some different numbers.

```
TO ACCEL :SX :SY
IF YCOR < -50 [STOP]
SETH 90 FD :SX
SETH 0 FD :SY - :ACC/2
ACCEL :SX (:SY - :ACC)
END

TO GROUND
PU SETPOS [150 -50] PD
SETPOS [-150 -50]
END

TO SHEL :VEL :ANGLE :ACC
HT GROUND
MAKE "SX :VEL * COS :ANGLE
MAKE "SY :VEL * SIN :ANGLE
```

```
ACCEL :SX :SY
(PR [Range = ] XCOR + 150)
END
```

When you fire an artillery shell, it flies through the air. And air produces resistance. So let's add in some air resistance to our procedure.

Have you ever heard of the sound barrier? As a jet fighter flies through the air, it pushes air out in front of it. The faster it goes, the more air it pushes.

When a jet flies at the speed of sound, it actually pushes through that big bubble of air that it has pushed out in front of it. If you're under that airplane, you'll hear a sound like thunder.

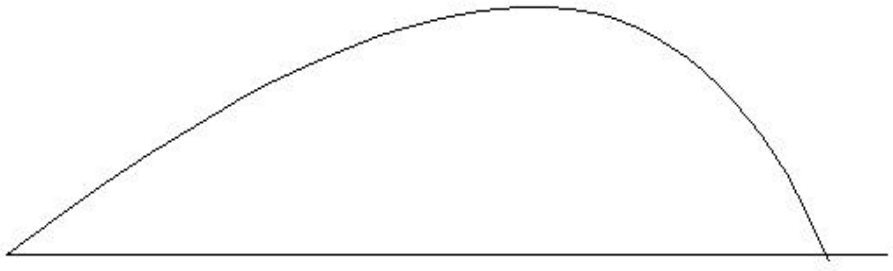
Our artillery shells are not going to break the sound barrier. But they are going to push air out in front of them. And that air is going to slow them down.

In our procedures, we can add an "air friction factor" that will slow the shell down. We call it :FRIC. Watch and see.

```
TO ACCEL :SX :SY
SETPOS LIST XCOR + :SX YCOR + :SY
IF YCOR < -50 [STOP]
ACCEL (:SX - :FRIC * :SX) (:SY - :FRIC * :SY - :ACC)
END

TO GROUND
PU SETPOS [150 -50] PD
SETPOS [-150 -50]
END
```

```
TO SHEL2 :VEL :ANGLE :ACC
MAKE “FRIC 0.1
HT GROUND
MAKE "SX :VEL * COS :ANGLE
MAKE "SY :VEL * SIN :ANGLE
ACCEL :SX :SY
(PR [Range = ] XCOR + 150)
END
```

This gives you enough information to design your own artillery game. Why not see what you can do to design a game where two artillery units fire at each other.



Think about it. There’s all sorts of possibilities.


MY JOURNAL
OR HOW I DID IT!