LOGO physics

JAMES P. HURLEY

Logo Physics

Logo Physics

James Hurley

HOLT, RINEHART AND WINSTON
New York Chicago San Francisco Philadelphia
Montreal Toronto London Sydney Tokyo
Mexico City Rio de Janeiro Madrid

Address correspondence to:
383 Madison Avenue, New York, NY 10017

Library of Congress Cataloging in Publication Data

Hurley, James P.
Logo physics.

Includes index.
Summary: Programming techniques, using Logo, for solving physics problems.
1. Physics—Computer-assisted instruction. 2. Atari computer. 3. LOGO (Computer program language)
[1. Physics—Computer-assisted instruction. 2. LOGO (Computer program language) 3. Programming (Computers)]
I. Title.
QC23.H888 1984 530'.028'5425 84-25312

ISBN 0-03-002913-9

Printed in the United States of America

Published simultaneously in Canada

5 6 7 8 039 9 8 7 6 5 4 3 2 1

CBS COLLEGE PUBLISHING
Holt, Rinehart and Winston
The Dryden Press
Saunders College Publishing

Contents

Acknowledgments

I would like to thank Tom Laugh for reading an earlier version of the manuscript and for his many helpful comments. Several of the problems and projects at the end of each chapter were included at his suggestion.

I would also like to thank Charles Rosengard for translating the programs into Commodore Logo and for his many suggestions for improvement.

Preface

Practice may or may not make perfect, but physics without practice is sterile. The laws of physics tell us how the universe works. To truly understand these laws we must *see* them at work. We must see, for example, how the laws of motion and the laws of universal gravitation combine to generate the elliptical paths of the planets. Without this confirmation, the laws of physics are just the laws of physics and not the laws of nature.

The basic laws of physics are not particularly difficult to understand. However, their application to anything but the simplest situation is difficult. To determine the planetary orbits, for example, we must solve a pair of coupled, nonlinear, second-order differential equations. Although a mathematical solution to these equations is difficult, a numerical solution is straightforward. Furthermore, the ease of obtaining a numerical solution is more or less independent of the nature of the forces, or, for that matter, the nature of the physical situation. We may apply the same numerical methods to projectile motion, planetary motion, space travel, harmonic oscillators, radioactive decay, foxes eating rabbits eating grass, and many others.

While numerical solutions are straightforward, they present two problems: They are tedious and difficult to interpret. Now the computer can surely remove the tedium. Computers thrive on tedium. When the computer is presented with a long repetitious task in arithmetic you can almost feel it quiver with excitement as it nervously awaits your command to begin.

But once the computer has determined the numerical solution, you find it has created a monster. You are deluged with a mass of data and must cope with the second problem generated by numerical solutions. How do you interpret all those numbers? The most satisfying answer is graphics. If it is a trajectory, then draw it. If not a trajectory, then find some way to plot the results: Number of radioactive particles as a function of time, position as a function of time, slope as a function of displacement, number of rabbits as a function of the number of foxes, and so on. A thousand pixels of graphics are worth a million numbers.

There are a variety of computer languages that are capable of generating graphics. As of this writing, Logo is the most flexible in this application and the easiest to learn. It is the most flexible because of its extensive graphics vocabulary. It is the easiest to learn because it allows one to communicate with the computer in much the same way people communicate with each other. Logo is a procedural (or modular) language. It permits one to define new words which then have the same status as the fundamental commands built into the language. These new words may be executed by themselves or as part of another procedure. "Words" are not bound irrevocably into "sentences" as in BASIC.

Language has an important effect on our powers of reason. Sometime when you are mulling over a problem, stop for a moment and notice how your thought process is carried by language. Try to think without using words. The fact that the Logo language structure is close to conventional language structure is a considerable asset.

It must be pointed out that this is neither a Logo manual nor a physics text. Rather, some previous exposure to Logo is presumed and, at the very least, a physics text shoud be available for consultation. While an effort has been made to include in each chapter a discussion of the relevant phys-

ics, there are wide gaps which can only be filled by a more comprehensive treatment.

The prerequisites for this book are minimal. Some trigonometry and a little algebra are necessary. However, the chapters are graded in order of mathematical difficulty. The earlier chapters assume less sophisticated programming skills and mathematical background. In fact, much of this early material was used in a pilot program with fifth- and sixth-grade school children.

Since their invention, high-speed computers have been used by scientists to solve problems that are too difficult to solve by normal methods. The languages that have been used to communicate with the computer were not easy to learn or use. With the advent of Logo it is now possible for anyone to speak with the computer. I would not like to represent this book as a revolutionary approach to the study of physics; it is quite traditional. In each chapter some physical principles are described and a problem is presented for solution. A method for its solution is then demonstrated, and, in conclusion, additional problems are posed to allow the student to test and expand his or her grasp of the subject. The only novelty in our presentation is the level at which the student is introduced to the computer. Ten years ago, only graduate students of physics were taught programming techniques. Today, every engineering student is introduced to computer programming. With the advent of simpler computer languages, it is time to offer this capability at an even earlier level. I recall when I was a lad in high school, hearing of some very bright person who had studied "THE CALCULUS" in college. Knowing "THE CALCULUS" was the trademark of great intellect. Today we teach calculus in high school.

We do not want to replace analytical methods with numerical methods. Analytical methods must remain the primary goal in any scientific discipline. Certainly the subject of free fall in a gravitational field is easily dealt with by algebraic means. If the student has the algebraic background, this should be the tool of choice. The purpose of introducing free fall in this book is to allow students without the necessary algebraic skills to solve such problems *and* to use this simple problem as an illustration of the basic algorithms for generating a solution in Logo. These same tech-

niques will later apply to more complicated problems where algebraic methods are inadequate.

The primary objective of this book is to demonstrate that any person with a curiousity about the laws of nature may solve a wide variety of problems using Logo to overcome the mathematical hurdles. Logo can be for the novice what FORTRAN is for the professional. The subjects dealt with here only scratch the surface and we hope our readers will be inspired to explore the laws of nature with the reassurance of having a turtle at their command.

Notes to the User

Not all versions of Logo were created equal. The Apple and IBM Logos lack multiple turtles (sprites). Atari Logo has multiple turtles but lacks a **TOWARDS** command. Commodore and Sprite Logos have sprites and a **TOWARDS** command but lack a **WINDOW**. Terrapin and Krell Logos lack both sprites and a **WINDOW**. The Radio Shack and TI Logos are not suitable for our application because they employ only integer arithmetic, or lack trigonometric functions, list processing, or all three. No Logo can be all things to all persons.

It is possible to write Logo procedures to supply some of the missing primitives. We may write a **TOWARDS** procedure for the Atari Logo which works in precisely the same way as the **TOWARDS** command in Apple or IBM Logo.

Simulating multiple turtles is not quite as straightforward. We have included an **ASK** procedure for the Apple and IBM Logos which is similar to that for the Atari and Commodore Logos.

The procedures **TOWARDS** and **ASK** are listed below. It is not important that you understand the programs, only how they are used. If your version of Logo lacks either, it

will be essential to store the missing command on disk or tape and load it into memory whenever the project requires. These procedures are slower than their machine language counterparts, but they will suffice. Generally speaking, speed is not a problem with the projects included in this book.

The syntax used in this book is LCSI (Logo Computer Systems Inc.). All major programs have been translated into the MIT syntax and appear in Appendix B.

TOWARDS

The following **TOWARDS** procedure is a modification of a program written by Harold Abelson of the Massachusetts Institute of Technology. It will be a necessity for all users of Atari Logo.

The effect of the command **TOWARDS** (position) is to output the heading of the turtle when facing the input position. For example:

```
SETHEADING TOWARDS [30 40]
```

will direct the turtle toward the point whose x and y coordinates are 30 and 40 respectively.

As a second example, if the turtle is home and we call

```
PRINT TOWARDS [30 40]
```

the heading of 38.87 is printed on the screen.
Incidentally, the **TOWARDS** procedure also gives you an **ARCTAN** procedure as well.

```
TO TOWARDS :POS
OP TOWARDS1 ( ( FIRST :POS ) - XCOR ) ( ( LAST :POS
 ) - YCOR )
END

TO TOWARDS1 :DX :DY
OP TOWARDS3 :DX :DY TOWARDS2 ABS :DX ABS :DY
END

TO TOWARDS2 :DX :DY
IF :DX = 0 [OP 0]
IF :DY = 0 [OP 90]
OP ARCTAN ( :DX / :DY )
END

TO TOWARDS3 :DX :DY :ANG
IF :DY < 0 [MAKE "ANG 180 - :ANG]
```

```
IF :DX < 0 [MAKE "ANG 360 - :ANG]
OP :ANG
END

TO ARCTAN.RAD :X
MAKE "X2 :X * :X
IF :X > 1 [OP 1.5707963 - ARCTAN.RAD ( 1 / :X)]
IF :X < -1 [OP -1.5707963 - ARCTAN.RAD ( 1 / :X)]
OP :X * ( 0.999866 + :X2 * ( - 0.3302995 + :X2 * (
 0.180141 + :X2 * ( - 0.085133 + :X2 * 0.0208351 )
 ) ) )
END

TO ARCTAN :X
OP 57.2957795 * ARCTAN.RAD :X
END

TO ABS :X
OP IF :X < 0 [- :X] [:X]
END
```

ASK

If your Logo does not possess multiple turtles (sprites), the following procedure will simulate the effect. It should be saved, then loaded whenever the project requires. To understand **ASK**, you should imagine that you have four turtles (0, 1, 2, 3) that can be addressed independently. Unless you specify otherwise, your instructions are addressed to turtle 0. If, on the other hand, you type

```
ASK 1 [SETPOS [30 40]]
```

turtle 1 goes to the coordinate position (30,40).

In general, when you type:

```
ASK (turtle number) (instruction list)
```

the specified turtle carries out the instructions contained in the list. The program to accomplish this is given below:

```
TO ASK :N :CMD
MAKE "OLD.POS.HEADING LIST POS HEADING
PU SETPOS FIRST THING :N
SETHEADING LAST THING :N
PD
RUN :CMD
MAKE :N LIST POS HEADING
PU
```

```
SETPOS FIRST :OLD.POS.HEADING
SETH LAST :OLD.POS.HEADING
PD
END

TO HOME.ALL
LOCAL "N
MAKE "N 0
REPEAT 4 [MAKE :N [[0 0] 0] MAKE "N :N +1]
END
```

In the procedure **ASK**, notice the line:

```
MAKE :N LIST POS HEADING
```

If N were 1, the position (30,40), and the heading 90°, then the variable 1 becomes the list [[30 40] 90]. The first list within the list is the position and the second is the heading. To see how the process works, first initialize the turtle numbers by calling **HOME.ALL**. If you print out the names **(PONS)** you will see:

```
0 is [[0 0] 0]
1 is [[0 0] 0]
2 is [[0 0] 0]
3 is [[0 0] 0]
```

If you now type:

```
ASK 1 [RT 90 FD 100 LT 90 FD 50 RT 90] PONS
```

you will see a right angle drawn on the screen followed by:

```
0 is [[0 0] 0]
1 is [[100 50] 90]
2 is [[0 0] 0]
3 is [[0 0] 0]
```

In this example, turtle 1 is at (100,50) with a heading of 90° and the other turtles are at home.

If *all* turtles are addressed by the **ASK** procedure it is not necessary to keep track of the old position and heading of the "main" turtle. In such cases the **ASK** procedure may be abbreviated as follows:

```
TO ASK :N :CMD
PU SETPOS FIRST THING :N
SETHEADING LAST THING :N
PD
RUN :CMD
MAKE :N LIST POS HEADING
END
```

Remember: In all programs that employ **ASK** be sure to initialize the variables by calling **HOME.ALL** first.

There is one unfortunate feature of this **ASK** command. It does not show the turtle. This may be remedied with the following program. It runs much more slowly than that above and should be used only when it is necessary to see the turtle. Save it under a different name (for example, **ASK.SHOW**).

```
TO ASK :N :CMD
MAKE "OLD.POS.HEADING LIST POS HEADING
PU SETPOS FIRST THING :N
SETHEADING LAST THING :N
DRAW.TURTLE [PE]
PD
RUN :CMD
DRAW.TURTLE [PD]
MAKE :N LIST POS HEADING
PU
SETPPOS FIRST :OLD.POS.HEADING
SETH LAST :OLD.POS.HEADING
PD
END

TO DRAW.TURTLE :CMD
RUN :CMD
LT 45
REPEAT 4 [FD 5 RT 90]
RT 45
END

TO HOME.ALL
LOCAL "N
MAKE "N 0
REPEAT 4 [MAKE :N [[0 0] 0] MAKE "N :N +1]
END
```

Logos of the Future

The Logo language is evolving rapidly. New versions of Logo or upgraded versions of older Logos are appearing regularly. Some of the comments about the limitations of existing Logos may be outdated by the time you read this. The following list is both a collection of things to look for when you select a version of Logo and a wish list. I have not

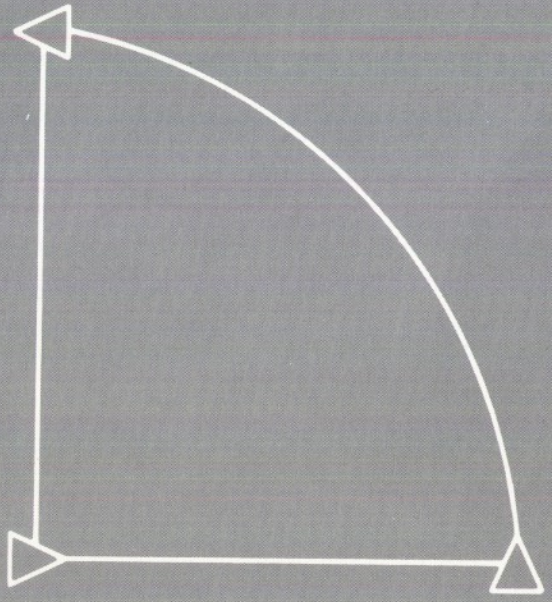
included those features common to Apple, Atari, Commodore, Krell, and Terrapin Logos. The most important features are listed first and less important features last.

1. **ASK** or **TELL** (The ability to address more than one turtle. The turtle need not be dynamic.)
2. **WINDOW** (The ability of the turtle to leave the screen without wrapping or giving an error message.)
3. **TOWARDS** (The ability to direct the turtle toward any position.)
4. **SETSCALE** (The ability to choose the representative size of the screen. If one is to do physics problems ranging from atomic orbitals to planetary motion it is helpful to choose a scale that suits your needs. In many cases it is necessary to set independent scales for the x and y axes.)
5. **CARTESIAN** (The ability to measure angles according to the more traditional methods of Cartesian geometry. It is difficult to explain to students why Logo formulas appear differently from those in their mathematics and physics texts. It would be helpful to be able to toggle between **CLOCK**, in which angles are measured clockwise from the y axis, and **CARTESIAN**, in which angles are measured counterclockwise from the x axis.)
6. Logarithms and exponentials.
7. Implementation of arrays and iteration. (Although lists and recursion are more generally applicable, arrays and iteration are usually easier to work with and more customary in mathematics applications.)

With the exception of **WINDOW** and **SETSCALE**, all of the above features can be written in Logo with varying degrees of success. (Arrays and iteration already exist in the parent language LISP.) However, as the Logo language matures it will begin to incorporate more and more features at the machine language level, and this will ease the task of communication with the computer in a manner which is both logical and literate. It is truly remarkable that Logo, as it exists today, is capable of filling such a wide variety of needs. This is due in part to the care that went into its design and in part to the extensible nature of Logo itself.

Logo Physics

chapter



1 Vectors

Introduction

We are all accustomed to making calculations with numbers. Two plus two is four. Numbers are useful to describe many things in physics. The temperature of boiling water is 100° Celsius. The weight of a bag of sugar is 10 pounds. The atmospheric pressure is 14 lb/in^2. Numbers are old friends.

However, there are many things in physics which are not conveniently described by simple numbers. You cannot describe the wind velocity by a simple number. It is not sufficient to say that the velocity is 10 mph since this tells us nothing about the direction of the wind. We might say however, that the wind velocity is 30° east of north with a magnitude of 10 mph. To give a complete description of the wind velocity, we must specify both its magnitude and direction. We cannot do this with a single number.

There are many things that fit into this category. Such things as displacement, velocity, acceleration, and force are all quantities that are specified by both a magnitude and a direction. There is a mathematical object ideally suited to

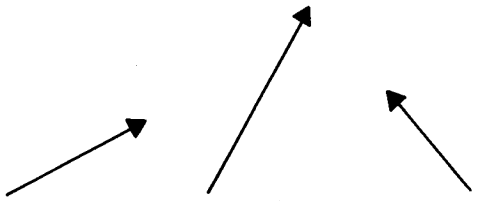


Figure 1.1 Three Vectors

describe such physical quantities. A *vector* in mathematics is defined as a directed line segment. It possesses both a magnitude (the length of the line) and a direction (the direction of the line segment). In Figure 1.1 we see an illustration of three vectors. Each has a different magnitude (length) and direction.

By contrast, a physical quantity that is determined by a single number is called a *scalar*. Examples of scalars are mass, temperature, density, and pressure. These quantities have only a magnitude—not a direction. We will use boldface type (**A**) to distinguish vectors from scalars.

We will illustrate the need for vectors by looking at the motion of a projectile. A ball is thrown into the air and follows the trajectory of the dotted line in Figure 1.2. The ball has a net displacement **D**, a velocity **v** (tangent to the path), an acceleration **a** (directed downward) in the direction of the gravitational force **F** acting on the ball.

The ball also has a number of physical attributes described by scalar quantities as illustrated in Figure 1.3. The weight of the ball is 3 lb, its volume is 20 in^3. Its temperature is 80° F and it strongly reflects light which has a wavelength of 7×10^{-6} m (a fancy way of saying that the ball is red).

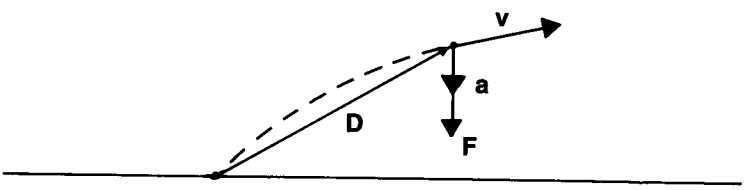


Figure 1.2 Vector Quantities of a Ball in Flight

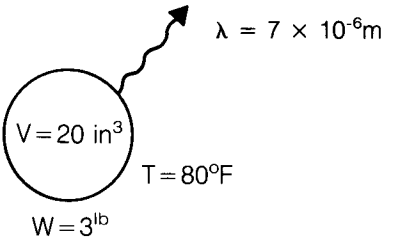


Figure 1.3 Scalar Quantities of a Ball

Addition of Vectors

We know how to do mathematical operations with scalars. We can add, subtract, multiply, divide, and square scalars. How do we perform mathematical operations with vectors? Well, what kind of mathematical operation do we wish to perform? Certainly, we must be able to add vectors. To see how this vector addition is done, let us consider a physical problem for which the appropriate definition is clear.

We have observed that displacements are vectors. A displacement from New York to Chicago is a vector (roughly 1000 mi due west). A displacement from Chicago to Indianapolis is a vector (roughly 200 mi due south). The net effect of two such displacements is a vector from New York to Indianapolis. The addition of these displacements is represented in Figure 1.4.

Polygon Method of Vector Addition

We see that the two displacement vectors are added by placing the vectors tail to tip and the sum is the vector from the tail of the first to the tip of the second. We may also

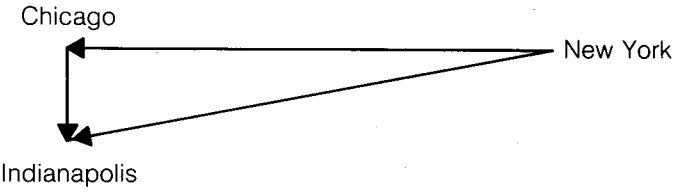


Figure 1.4 Displacement Vectors

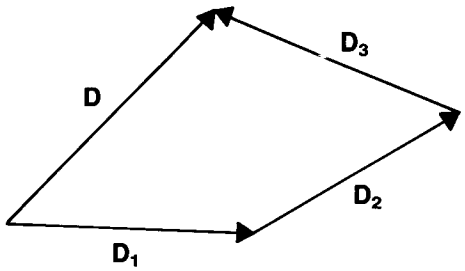


Figure 1.5 Polygon Method of Vector Addition

express this sum symbolically by the vector equation:

$$\mathbf{D} = \mathbf{D}_1 + \mathbf{D}_2$$

This rule may be generalized to addition of any number of vectors. In Figure 1.5 the sum of the three vectors $\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3$ is the vector **D** obtained by the above tail-to-tip rule. You will notice that the figure forms a closed polygon, hence the name "polygon method of vector addition."

To give you some examples of vector addition by the polygon method we have constructed a **VECTOR.ADDING.MACHINE** program. There are few calculations that Logo is better suited for than the addition of vectors.

```
TO VECTOR.ADDING.MACHINE
HOME
WINDOW
CS HT PD
DRAW.VECTORS
FIND.RESULTANT
PRINT [DO YOU WISH ANOTHER EXAMPLE? ( Y OR N )]
MAKE "ANSWER RC
IF :ANSWER = "Y [VECTOR.ADDING.MACHINE]
END

TO DRAW.VECTORS
PRINT [ENTER MAGNITUDE ( SPACE ) DIRECTION.]
MAKE "VEC RL
IF:VEC = [] [STOP]
SETH LAST :VEC
FORWARD FIRST :VEC
DRAW.TIP
DRAW.VECTORS
END

TO FIND.RESULTANT
SETH TOWARDS [0 0]
```

```
RIGHT 180
PRINT [RESULTANT VECTOR]
(PRINT [MAGNITUDE =] DISTANCE.FROM.HOME [HEADING =]
 HEADING)
MAKE "TIP POS
PU HOME
SETH TOWARDS :TIP PD
SETPOS :TIP
DRAW.TIP
END

TO DRAW.TIP
RIGHT 25
BACK 15 FORWARD 15
LEFT 50
BACK 15 FORWARD 15
END

TO DISTANCE.FROM.HOME
OP SQRT (SQ XCOR) + SQ YCOR
END

TO SQ :N
OP :N * :N
END
```

In **VECTOR.ADDING.MACHINE** the screen is cleared and the turtle sent **HOME** and **DRAW.VECTOR** is called where a list containing the magnitude and direction of the vector is input. For example, to characterize a vector whose magnitude is 100 units and whose direction is 45°, we would in effect **MAKE "VEC [100 45]**. (To check this, stop the program after entering a vector and enter the names editor

```
ENTER MANITUDE ( SPACE ) DIRECTION.
100 70
ENTER MAGNITUDE ( SPACE ) DIRECTION.
50 300
RESULTANT VECTOR
MAGNITUDE = 77.9238 HEADING = 40.5591
```

Figure 1.6 VECTOR.ADDING.MACHINE

(**EDNS**). You should see **MAKE "VEC [100 45]**. The ability to put any kind of information into lists is one of the more powerful features of Logo.) The heading of the new vector is set and the turtle moves forward a distance equal to the magnitude of the vector. An arrow head is added to the vector and the procedure calls itself to repeat the process for the next vector. When all vectors have been entered, the resultant is obtained by entering an empty list. The **IF** statement then stops **DRAW.VECTORS** and we return to **VECTOR.ADDING.MACHINE** where we **FIND** the **RESULTANT** by setting the heading away from the origin to obtain the heading of the resultant. The length of the resultant is just the **DISTANCE.FROM.HOME**.

To run this program, type **VECTOR. ADDING.MACHINE**. Enter the magnitude and heading of the vectors you wish to add. When you want the sum, enter an empty list (simply press RETURN without entering any data) and the sum will be printed and then drawn.

Component Method of Vector Addition ▲

Although the polygon method is easy to visualize, it is difficult to execute in practice (unless you have a turtle to do your work) because of the complexity of the polygon. A simpler method is to "resolve the vectors into components" and then add the components. Components are easy to add. They add the same way as scalars.

The components (D_x and D_y) of a vector **D** are obtained by projecting **D** on the x and y coordinate axes. We see from

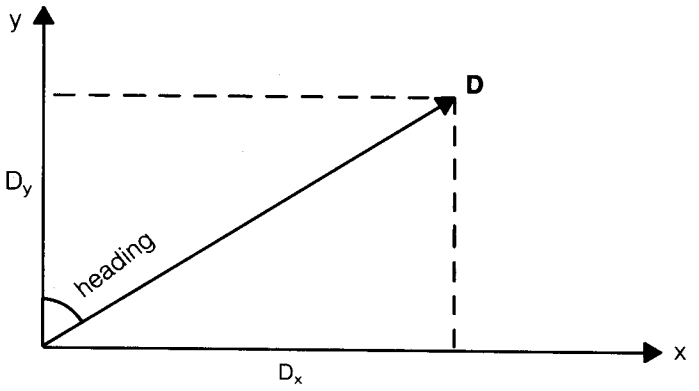


Figure 1.7 Component Method of Vector Addition

Figure 1.7 that

$$D_x = D \sin(\text{heading})$$

and

$$D_y = D \cos(\text{heading})$$

So if we know the magnitude and heading of any vector we can find its x and y components.

Consider now the sum of three vectors: $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{D}_3$. It should be clear from Figure 1.8 that

$$D_x = D_{1x} + D_{2x} + D_{3x}$$

and

$$D_y = D_{1y} + D_{2y} + D_{3y}$$

So the x and y components of the resultant is just the sum of the x and y components of the three vectors $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{D}_3$. The reason this method is simple is that the x and y components add just like scalars. We do not have to do any fancy vector addition to determine $D_{1x} + D_{2x} + D_{3x}$ or $D_{1y} + D_{2y} + D_{3y}$.

We will use this method in many of the dynamics problems in upcoming chapters.

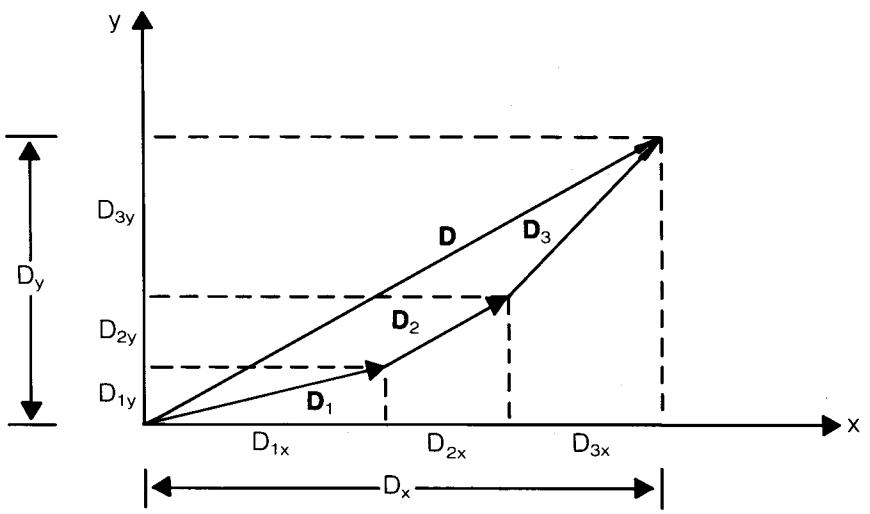


Figure 1.8 The Components of a Vector

Projects

1. Write a procedure which generates the x and y components of any vector. Here is a sample of how the program might start:

```
TO COMPONENTS :MAGNITUDE :HEADING
(PRINT [X COMPONENT =] . . .
. . .
. . .
END
```

All you will need is a four- or five-line program. To obtain the components of a vector of magnitude 50 and heading 70 you should be able to type **COMPONENTS 50 70**, and the computer will print **X COMPONENT = 47, Y COMPONENT = 17.1**.

Use your procedure to find the components of the following vectors:

a. 100 at 45°
b. 100 at 135°
c. 100 at 225°
d. 100 at 315°

You do not need to use any graphics for this project; text alone will do.

2. Can you write a procedure that will reverse the process carried out in Project 1? You input the x and y components and the computer generates the magnitude and direction of the vector.

Problems

1. Use the **VECTOR.ADDING.MACHINE** program to find the sum of the following vectors:
   a. 100 at 30°, 150 at 180°, and 160 at 290°. (Answer: 100.7 at 265°)
   b. 100 at 0°, 100 at 90°, 200 at 280°, and 200 at 270°. (Answer: 141.4 at 135°)
   c. 30 at 90° and 40 at 0°. (Answer: 50 at 36.9°)
2. Suppose all roads in your neighborhood run E-W or N-S. To get to town you have to travel 30 mi due east and then 40 mi due north. How much shorter would your trip be if you could travel as the crow flies? (Answer: 20 mi)
3. We have seen how to add vectors. How do we subtract vectors? What is $\mathbf{A} - \mathbf{B}$? Let us try some traditional arithmetic. Suppose we write $\mathbf{A} - \mathbf{B} = \mathbf{A} + (-\mathbf{B})$. We have

converted the difference of **A** and **B** into a sum of **A** and $-\mathbf{B}$. Now we know how to do sums. But what is the vector $-\mathbf{B}$? We will define $-\mathbf{B}$ to be a vector equal in magnitude to **B** but opposite in direction. There are two ways you can use the **VECTOR.ADDING.MACHINE** program to subtract vectors. One is to determine the direction opposite to **B** and the other is to use the same direction as **B** but go **FORWARD** the negative of the magnitude of **B**.

Let **A** be 50 units at 60° and **B** 70 units at 30°. What is **A** − **B**? (Answer: 36.6 at 166.9°)

4. We have noted that velocities are vectors. As an example of velocity addition and relative velocities, consider the following. A car is moving down a straight road at 60 mph. It is raining but windless so the rain is falling vertically. The average speed of the rain drops is 15 mph. What is the direction of the track left by the raindrops on the side window of the car?

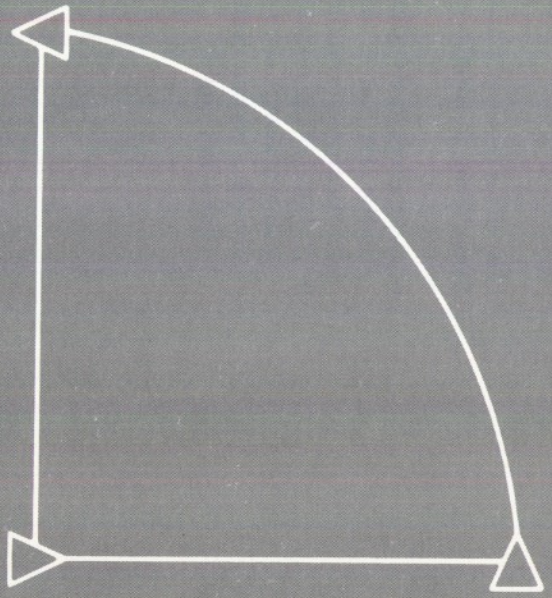
   To solve such problems we must know how to add velocities. Let **V**(c) (read "v of c") be the velocity of the car and **V**(r) the velocity of the rain. We denote the velocity of the rain relative to the car by **V**(r/c) (read "V of r relative to c"). Now the velocity of the raindrops is equal to the velocity of the raindrops relative to the car plus the velocity of the car, so that

$$\mathbf{V}(r) = \mathbf{V}(r/c) + \mathbf{V}(c)$$

   (The virtue of this notation is that the terms in the parentheses form an equality under multiplication; that is, r = (r/c)c.)

   a. Determine the direction of the track of the raindrops on the car; that is, the direction of **V**(r/c). (Answer: 76° measured from the vertical)
   b. A police officer is stationed in the turnpike toll booth giving tickets to cars on which the rain track angle exceeds a certain value. If the speed limit is 55 mph, what is this angle?

chapter



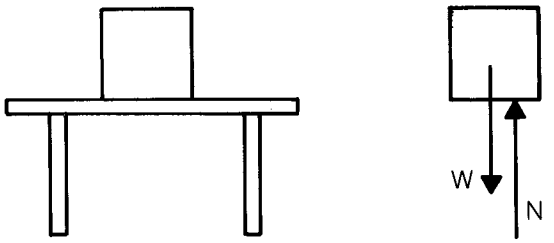
2 Equilibrium of Forces

Introduction

If a body is to be in equilibrium, the vector sum of all the outside forces acting on it must be zero. If a 100 lb body rests on a flat table the forces acting on the body are the gravitational force of 100 lbs and the normal supporting force of the table; also 100 lbs. The gravitational force (W) is directed downward and the normal force (N) is directed upward (see Figure 2.1). The vector sum of these two forces is zero.

When there are more than two forces acting on the body the solution is not quite as simple, but the same rule applies. The vector sum of the forces must be zero. Consider the example of a 100 lb block resting on a smooth inclined plane. The block is prevented from sliding down the incline by a cord parallel to the incline. The problem is to determine the tension in the cord (T) and the normal support force (N) that the incline exerts on the block.

To solve this problem we must first choose a body whose equilibrium we wish to study; draw a vector diagram of all the forces acting on the body; and finally equate the vector



W = N = 100 lb

Figure 2.1 The Vector Sum is Zero

sum of all the forces to zero. In this problem the body to study is the block itself. The three forces acting on the block are illustrated in Figure 2.2. Let us use the polygon rule of vector addition to find what the tension in the cord and normal force have to be. The polygon must form a closed figure if the sum of the vectors is to be zero. This polygon is illustrated in Figure 2.3. The problem is to determine a tension **T** and normal force **N** that are consistent with the figure. Before we do this, let us look at another problem. We would like to show that our method is applicable to a wide variety of problems and not just the ones discussed above.

A Second Example

A 100 lb weight is suspended from a weightless cord. The cord is tied to a horizontal weightless pole. The pole is attached to a vertical wall. The free end of the pole is held up by a weightless cord tied to the wall at an angle of 45°

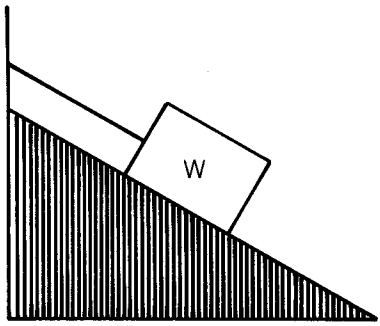


Figure 2.2 A Block on an Incline

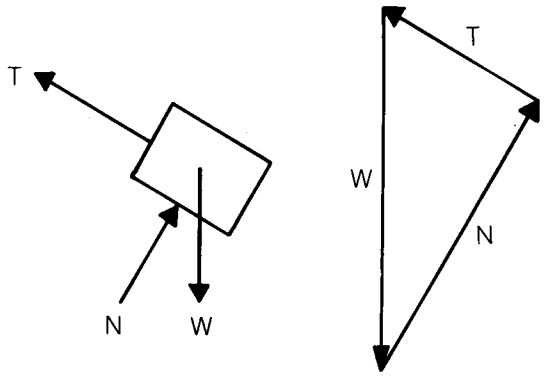


Figure 2.3 Tension and Normal Forces

as illustrated in Figure 2.4. The task is to determine the force **F** of the pole and the tension **T** in the cord. Again we must pick a particle or body and examine the forces acting on it. We choose the small particle where the two cords and the pole meet. The forces acting on this particle are shown in Figure 2.5. The sum of these forces will be zero if they form a closed polygon, like that shown in Figure 2.6. Once again, the task is to find the forces consistent with those in the figure.

To solve these and other similar problems we will construct a Logo program.

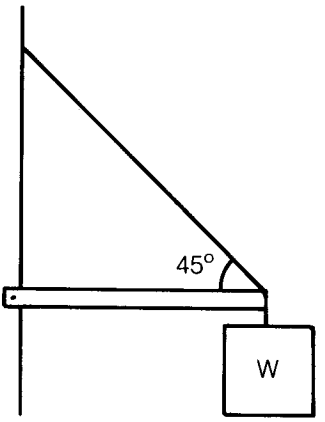


Figure 2.4 Suspended Block

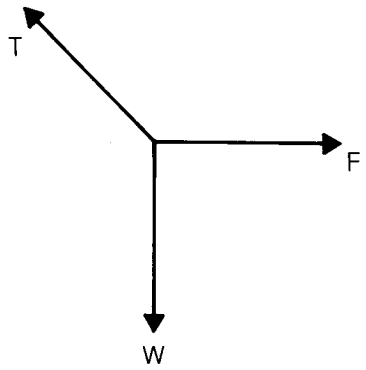


Figure 2.5 Forces acting on Particle

The Logo Program

We will take as a general problem the equilibrium of a body under the action of three forces; **W, T** and **F**. **W** is a weight and is directed downward. **F** and **T** are forces whose directions are known. We will denote their headings by **F.ANGLE** and **T.ANGLE**. The problem is to determine the magnitude of **F** and **T** so that they form a closed triangle with **W** as illustrated in Figure 2.7.

Suppose we were to try to solve this problem with a pencil, a piece of paper, and a ruler. We would draw a vector of length W in the downward direction. We position the ruler at the tip of this line and set the ruler to the angle **F.ANGLE**. We must then draw a line along this heading of such a length that a line from its far end to the base of W possesses the proper heading of **T.ANGLE**. The turtle can

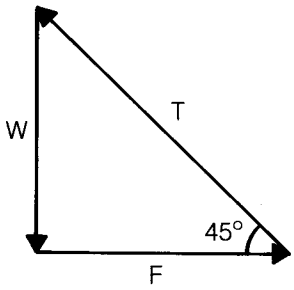


Figure 2.6 Polygon

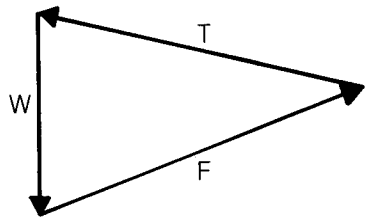


Figure 2.7 Forces

execute this process in the following manner. Start the turtle from the bottom of W with a heading of **F.ANGLE**. Move forward a distance of 10 turtle units. Ask the turtle to set its heading toward the base of **W** (that is, toward **[0 0]**). Check to see if the heading is less than **T.ANGLE**. If so, the turtle has not gone far enough and so proceeds forward another 10 turtle units. When the heading does become less than **T.ANGLE** we step back 10 units and divide the length of the step by 2. (For technical reasons it is actually faster if you divide the step length by 3.5.) This process is repeated until the step length is less than .01. This should bring us within .01 units of the end of **F**. (If you want greater accuracy make this number smaller.) The magnitude of **F** is just the distance between the present position and the tip of **W**. We record the length of **T** as the distance between the tip of **F** and the tail of **W** and draw the vector **T**. This process is carried out in the following program:

```
TO EQUILIB :W :F.ANGLE :T.ANGLE
WINDOW
HOME
BACK :W MAKE "OLD.POS POS
STEP 10
(PRINT [F =] DIST.BETWEEN POS :OLD.POS )
MAKE "T DIST.BETWEEN POS [0 0]
( PRINT [T =] :T )
SETH :T.ANGLE
FORWARD :T
END

TO STEP :S
IF :S < 0.01 [STOP]
SETH :F.ANGLE
FORWARD :S
SETH TOWARDS [0 0]
IF HEADING < :T.ANGLE [SETH :F.ANGLE BACK :S MAKE "S :S/2]
```

```
STEP :S
END

TO DIST.BETWEEN :P1 :P2
OP SQRT ( SQ ( FIRST :P1 ) - FIRST :P2 ) +SQ (LAST
:P1 ) - LAST :P2
END

TO SQ :X
OP :X * :X
END

TO START
EQUILIB 60 45 270
END
```

With this program we can solve a number of problems in equilibrium dynamics.

Problems

1. A 100 lb block is suspended by ropes as shown in Figure 2.8. Find the tension in the ropes. (Answer: 60 lb and 80 lb)

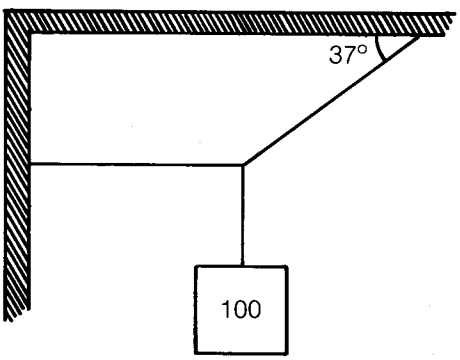


Figure 2.8 Problem 1

2. A 100 lb block hangs from a boom as shown in Figure 2.9. Find the thrust of the boom and the tension in the supporting cord.

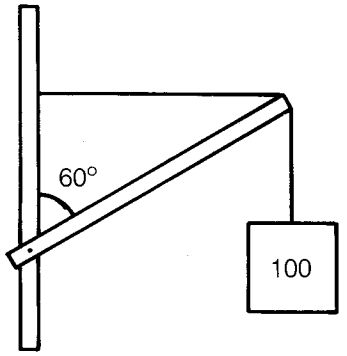


Figure 2.9 Problem 2

3. A driver is using a rope to pull her automobile out of a rut. Rather than pull directly on the rope, the driver ties the rope to a tree and makes the rope as taut as possible. She then applies a 100 lb force at the midpoint in a direction perpendicular to the straight line between the automobile and the tree. The rope bends, making an angle of 5 degrees with the straight line (see Figure 2.10). What is the force on the auto?

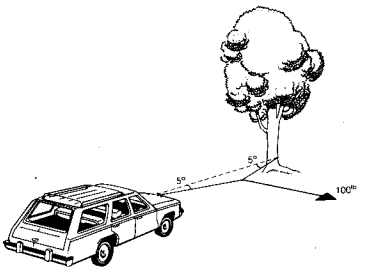


Figure 2.10 Problem 3

4. A 200 lb man is walking a tight rope. At his current position, the rope makes an angle of 5 degrees on one side and 10 degrees on the other as shown in Figure 2.11. What is the tension in the rope on each side?

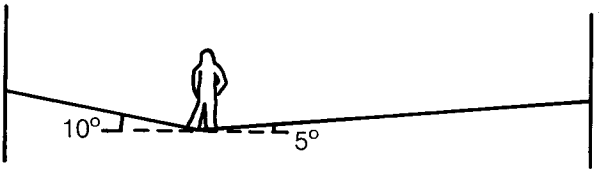
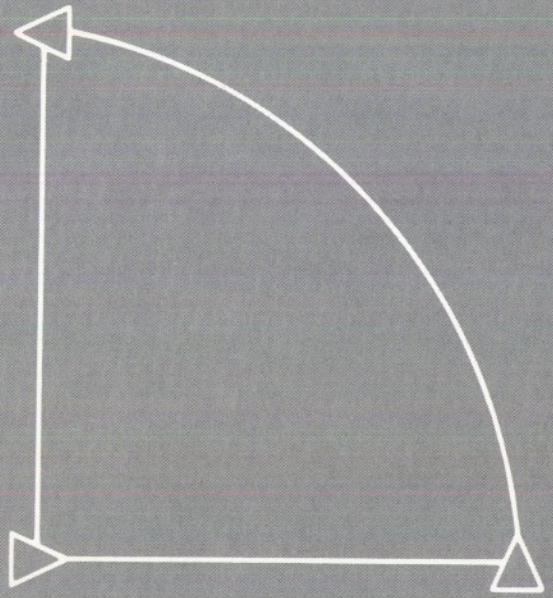


Figure 2.11 Problem 4

Project

1. We have written a Logo program to determine the magnitude of **F** and **T** given the direction of **F** and **T** and the vector **W**. Can you write a program that determines the direction of **F** and **T** given **W** and the magnitude of **F** and **T**? (The answers will not be unique.)

chapter



3 Free Fall

Introduction

One of the most important subjects in physics is the study of motion. How do bodies move when forces are applied to them? How does a baseball move after it leaves the bat? How does a ball swing at the end of a string? How does a satellite revolve about the Earth and the Earth about the sun? We will examine these questions and more, but first we must begin with a simple problem: the free fall of a body in the Earth's gravitational field.

There are three variables that we must consider when we study motion: position, velocity, and acceleration. We want to know where the body is, how fast it is moving, and how fast the velocity is changing. Let us take a moment to define the concepts of velocity and acceleration of a body moving along a straight line.

Velocity

The velocity of a body is defined as the distance the body moves in a unit time. For example, a car traveling at

a steady rate that moves 60 miles in one hour has a velocity of 60 mi/hr. A ball rolling on a table that covers 20 ft every second has a velocity of 20 ft/sec. In general, the velocity of a body traveling at a constant rate is equal to the distance covered divided by the time. We may express this relation by the equation:

$$\text{velocity} = \frac{\text{distance covered}}{\text{time}}$$

Question: A man runs the hundred yard dash in 9 sec (see Figure 3.1). What is his average velocity?

Answer: The distance is 100 yards and the time 9 sec so the velocity is 100 yd/9 sec = 11.1 yd/sec. Since there are 3 ft in a yard we might also express this velocity as 33.3 ft/sec.

Velocity can be either positive or negative. If a body is moving along some directed line, the velocity is positive if it is moving in the forward direction on the directed line and negative if it is moving in the opposite direction. The x and y axes in the turtle coordinate system are such directed lines. A turtle moving to the right along the x-axis has a positive x-component of velocity. A turtle moving to the left has a negative x-component of velocity.

Acceleration

A body that is changing its velocity is said to be accelerating. The acceleration is a measure of the *rate* at which the velocity is changing. By comparison, the velocity is the *rate* at which the position is changing. As an example, imagine a ball rolling down an incline with a velocity of 20 ft/sec at some point (as shown in Figure 3.2). One second later the velocity has increased to 22 ft/sec. The velocity has increased by 2 ft/sec during one second. The acceleration therefore is 2 ft/sec/sec. (This is often written as 2 ft/sec^2.) The velocity changes by 2 ft/sec every second.

In general, the acceleration is defined as the change in velocity divided by the time it takes. We may define the

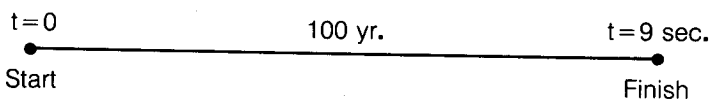


Figure 3.1 100-yard dash

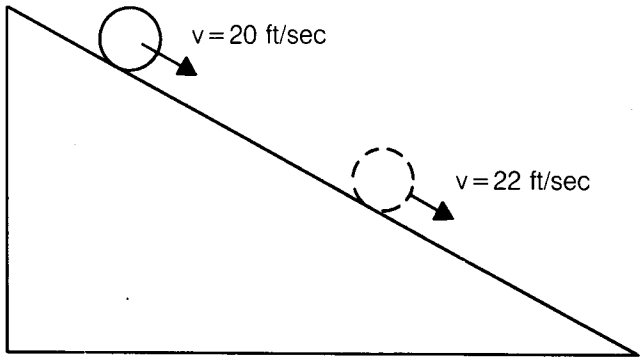


Figure 3.2 A Ball Rolling Down an Incline

acceleration with the following equation:

$$\text{acceleration} = \frac{\text{change in velocity}}{\text{time}}$$

Question: A car moves from 30 mi/hr to 40 mi/hr in 1/2 hr at a steady rate. What is the acceleration?

Answer: The acceleration is the change in velocity, 40 mi/hr − 30 mi/hr or 10 mi/hr divided by the time of 1/2 hr. That is,

$$\text{acceleration} = \frac{40\,\text{mi/hr} - 30\,\text{mi/hr}}{1/2\,\text{hr}} = 20\,\text{mi/hr/hr}$$

Question: The velocity of a ball thrown into the air changes from 90 ft/sec to 58 ft/sec during the first second. What is the acceleration?

Answer: The acceleration is the change in velocity (final velocity − initial velocity), 58 ft/sec − 90 ft/sec or −32 ft/sec. The acceleration is the change in velocity divided by the time; that is, −32 ft/sec/sec. (This is often called the acceleration due to gravity. It is the same for all bodies when the air friction is negligible.) We sometimes say that a body is *decelerating* if the acceleration is negative.

Free Fall

A body is in free fall if there are no forces acting on it other than the gravitational pull of the Earth. The simplest motion of this type is vertical free fall. To be specific, suppose a body is released from a height of 50 ft above the ground. The body falls with constant acceleration. For convenience we will choose an acceleration of .5 ft/sec/sec. (This is not the acceleration of gravity on Earth. That would be

32 ft/sec/sec. However, if we choose such a large acceleration things would happen too fast on our small screen. We will consider realistic problems when we take up scaling techniques.) Two of the questions we would like to answer are: What is the velocity when the body strikes the ground? How long does it take before it strikes the ground?

We cannot answer these questions using the above definitions only. We can ask the turtle to help. We will ask the turtle to execute the following set of commands once each second:

```
TO STEP
FORWARD :VELOCITY
MAKE "VELOCITY :VELOCITY + :ACCELERATION
STEP
```

To see what this procedure does and why it solves our problem, we will follow the steps one at a time. The first command is **FORWARD :VELOCITY**, which translated says: "Move forward a distance equal to the value of the velocity variable." Suppose the velocity is 10 ft/sec. The turtle will move 10 units in one second.

The next command is **MAKE "VELOCITY :VELOCITY + :ACCELERATION**, which translated says: "Increase the value of the velocity variable by an amount equal to the acceleration variable." If the velocity is 10 ft/sec and the acceleration (the change in the velocity per second) is .5 ft/sec/sec, then the new value of the velocity variable will be 10.5 ft/sec. This gives us the increase in the velocity in one second.

The next command is **STEP**. This calls the procedure again, with a new velocity of 10.5 ft/sec. The turtle moves 10.5 ft during the next second. At the end of this time the velocity has further increased to 11 ft/sec. This process is repeated over and over again, moving the turtle by ever increasing distances due to the ever increasing velocity. We must, of course, find some way to get the turtle out of this infinite loop. We do this with a conditional statement (an "if" statement): **IF YCOR < 0 [STOP.]**

Our procedure is now:

```
TO STEP
IF YCOR < 0 [STOP]
FORWARD :VELOCITY
```

```
MAKE "VELOCITY :VELOCITY + :ACCELERATION
STEP
END
```

This **STEP** procedure will follow the turtle to the ground. But we must take care of some preliminaries before it can be run. In **FREE.FALL** the ground is drawn, and the turtle headed downward (**RT 180**). Our entire program is:

```
TO FREE.FALL :HEIGHT
WINDOW
CS PD
DRAW.GROUND
PU
SETY :HEIGHT
MAKE "ACC .5
MAKE "VEL 0
MAKE "TIME 0
RIGHT 180
PD
STEP
PRINT "
PRINT [THE TURTLE HAS LANDED]
END


TO DRAW.GROUND
PU
SETPOS [-100 0] PD
SETPOS [100 0]
HOME
PU
END

TO STEP
(PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE
 (:HEIGHT - YCOR))
IF YCOR < 0 [STOP]
FORWARD :VEL
MAKE "VEL :VEL + :ACC
MAKE "TIME :TIME + 1
STEP
END

TO START
FREE.FALL 50
END
```

Notice that we have amplified the **STEP** procedure by adding a command to print the current time, velocity, and distance.

We have also included a **START** procedure. (We will often include a **START** procedure in the chapters to follow. Its purpose is to simplify initial program execution.) If you run **START** the turtle is released from a height of 50 ft above the ground and the following results are printed:

After sec	Velocity	Distance covered
0	0	0
1	.5	0
2	1.0	.5
3	1.5	1.5
4	2.0	3
5	2.5	5
6	3.0	7.5
7	3.5	10.5
8	4.0	14
9	4.5	18
10	5.0	22.5
11	5.5	27.5
12	6.0	33
13	6.5	39
14	7.0	45.5
15	7.5	52.5

We see that sometime between the 14th and 15th second, the turtle traveled the 50 ft to the ground. His speed increased to between 7 and 7.5 ft/sec. Try some new values for the height and see how the turtle picks up speed as he falls.

We may clean up the **STEP** procedure by using this simple tail recursion:

```
STEP :VEL :TIME
(PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE
 (:HEIGHT - YCOR))
IF YCOR < 0 [STOP]
FORWARD :VEL
STEP :VEL + :ACC :TIME + 1
END
```

Each time **STEP** calls itself it does so with the increased

values of the velocity and the time. (Notice also that you must change the call **STEP** in **FREE.FALL** to **STEP :VEL :TIME**.) We will continue to use this more compact form.

The Bouncing Turtle and the Midpoint Approximation

Let us look at an elastic turtle; a bouncing turtle. We would like the turtle to rebound rather than stop when he hits the ground. To rebound we mean that the turtle reverses his velocity when **YCOR** < **0**. Therefore we simply have to change the line

```
IF YCOR < 0 [STOP]
```

in **STEP** to

```
IF YCOR < 0 [MAKE "VEL -:VEL]
```

Try this and watch him bounce. You should notice that he bounces higher and higher; this is not right. If the turtle is turned around at the bottom and sent up with the same velocity as when he came down, he should rebound to the same height. What went wrong?

The source of our difficulty is something that will pursue us throughout this book. The turtle's solution to the free fall problem is only an approximation. It is the error in this approximation that causes the turtle to continue to bounce higher and higher. To see where the error arose, let us examine the **STEP** procedure.

When we ask the turtle to go **FORWARD :VEL** we should ask: What velocity? The velocity after 10 sec is 5 ft/sec. The velocity after 11 sec is 5.5 ft/sec. So what is its velocity *during* the 10th second? We cannot really say.

There are two ways to deal with this problem. One is to make the time interval between steps shorter. If the time interval were just .1 sec then the change in velocity would be just one tenth of the acceleration, or .05 ft/sec. The velocity after 10 sec is 5 ft/sec. The velocity after 10.1 sec is 5.05 ft/sec. The difference in velocity is so small that it would not make a big difference which velocity we used. We could

use a still smaller time interval of .01 sec and reduce the error even further.

The trouble with using such small time intervals is that the program would take a long time to run. A better solution is to use the *midpoint approximation*.

If the velocity after 10 sec is 5.0 ft/sec and the velocity after 11 sec is 5.5 ft/sec, then a good approximation of the velocity during the 10th sec is 5.25 ft/sec—the average velocity. Another way to express this result is to set the midpoint velocity to the velocity at 10 sec plus one half the change in the velocity. Remember, the change in the velocity is the acceleration. Therefore, instead of moving **FORWARD :VEL**, we move **FORWARD :VEL + :ACC/2**. The improved program is **BETTER.FREE.FALL**.

```
TO BETTER.FREE.FALL :HEIGHT
WINDOW CS PD
DRAW.GROUND
PU
SETY :HEIGHT
MAKE "ACC .5
MAKE "VEL 0
MAKE "TIME 0
RIGHT 180
PD
STEP :VEL :TIME
PRINT "
PRINT [THE TURTLE HAS LANDED]
END

TO DRAW.GROUND
PU SETPOS [-100 0] PD
SETPOS [100 0]
HOME PU
END

TO STEP :VEL :TIME
(PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE
 (:HEIGHT - YCOR))
IF YCOR < 0 [STOP]
FORWARD :VEL + :ACC / 2
STEP :VEL + :ACC :TIME + 1
END

TO START
BETTER.FREE.FALL 50
END
```

We will make frequent use of this midpoint approximation in the material that follows. It is particularly suitable in this problem because the acceleration is constant. In fact, by using the midpoint approximation the calculation is exact.

Try bouncing the turtle again using the midpoint approximation. You should find that he no longer continues to bounce higher and higher.

A Simpler Midpoint Approximation

There is a simpler way to apply the midpoint approximation that will both clean up the appearance of our programs and make them run faster. Examine **BEST.FREE.FALL**.

```
TO BEST.FREE.FALL :HEIGHT
WINDOW CS PD
DRAW.GROUND
PU
SETY :HEIGHT
MAKE "ACC .5
MAKE "VEL 0 + :ACC / 2
MAKE "TIME 0
RIGHT 180
PD
STEP :VEL :TIME
PRINT "
PRINT [THE TURTLE HAS LANDED]
END

TO DRAW.GROUND
PU SETPOS [-100 0] PD
SETPOS [100 0]
HOME PU
END

TO STEP :VEL :TIME
(PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE
 (:HEIGHT - YCOR))
FORWARD :VEL
STEP :VEL + :ACC :TIME + 1
END

TO START
BEST.FREE.FALL 50
END
```

Notice that it is the same as **FREE.FALL** except for one statement. In **FREE.FALL** we said:

```
MAKE "VEL 0
```

In **BEST.FREE.FALL** we say:

```
MAKE "VEL 0 + :ACC / 2
```

The only difference between the two programs is that we have applied the midpoint approximation to the initial velocity. Notice that **STEP** is unchanged. It would seem that we have not been very consistent. We applied the midpoint approximation to the first turtle step but not to the subsequent steps as we did in **BETTER.FREE.FALL**. Try running **BEST.FREE.FALL**. When the turtle lands, compare the printed values of the distance and time with the results of **BETTER.FREE.FALL**. (Forget about the velocity for the moment.) *The results are precisely the same.* The reason they are the same is that in both **BETTER. FREE.FALL** and **BEST.FREE.FALL** we get started on the right foot and on each subsequent step, increment the midpoint velocity by the acceleration thereby obtaining the new *midpoint velocity*. (Since it is the midpoint velocities that are computed, they will not be the same as the velocities in **BETTER.FREE.FALL**.) We will use this simpler algorithm in the future when we need the improved accuracy of the midpoint approximation.

Projects

1. We have seen how to make the turtle bounce off the ground. Suppose we wanted to place a shelf at any point and allow the turtle to bounce off it (either on the way up or the way down; see Figure 3.3). Try inserting an **IF KEYP**

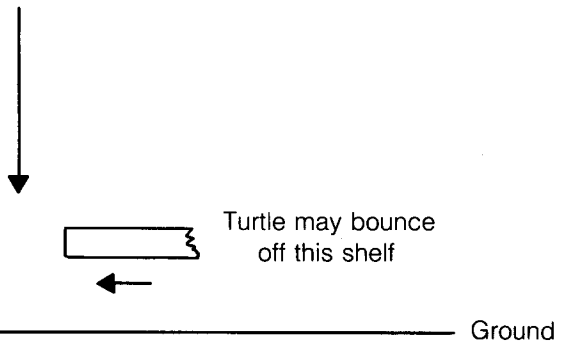


Figure 3.3 Free Fall Turtle Bouncing Off Shelf

statement to reverse the turtle at any point in his motion. (Do not forget that **KEYP** will not proceed until the key is printed.) Press the space bar and notice that he continues to rise to the height from which he was dropped. (This is a good example of energy conservation.)

2. Write a program **TOSS :HEIGHT :VELOCITY** that allows you to set the turtle at any height and give him any initial velocity (either up or down).

Problems

1. Using the midpoint approximation, make the following calculation. (You will need to estimate the correct answers since the turtle passes the ground level before stopping.)
   a. If the turtle is released from a **HEIGHT** of 100 ft, what is the velocity when the turtle strikes the ground?
   b. Repeat (a) with **HEIGHT = 200** ft.
   c. Repeat (a) with **HEIGHT = 50** ft.

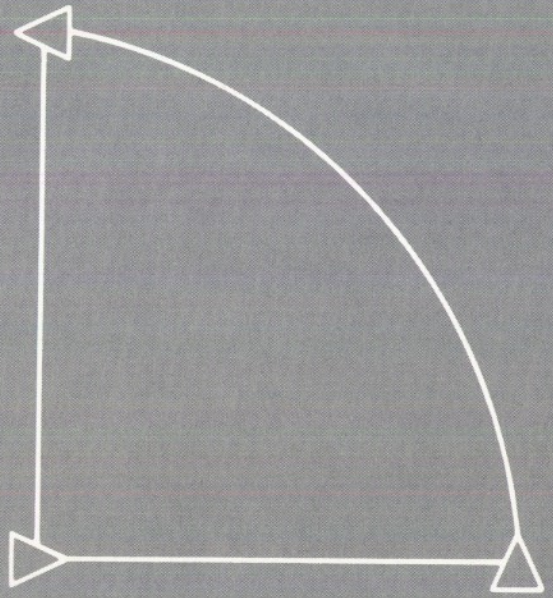
2. From the results of Problem 1 can you determine how the velocity depends on the height? We have listed several possibilities below.
   a. velocity = (some constant) × (height)
   b. velocity = (some constant) × $(\text{height})^2$
   c. velocity = (some constant) × $(\text{height})^{1/2}$
   d. velocity = (some constant) / $(\text{height})^{1/2}$
   e. velocity = (some constant) / (height)

   To determine which formula is correct, notice that if (a) is correct the velocity should double if you double the height. If (b) is correct the velocity should quadruple if you double the height. If (c) is correct you should double the velocity if you quadruple the height. And so on and so on.

3. Can you determine how the time of flight depends on the height? For example, does **TIME** = (some constant) × (**HEIGHT**)?

4. How does the velocity depend on the time?

chapter



4 Projectile Motion and the CRT

Introduction

Next to free fall, projectile motion is the simplest example of motion in a gravitational field. The basic physical law is that the acceleration of any body in a uniform gravitational field is constant.

$$\mathbf{A} = \text{constant}$$

The acceleration is a vector directed vertically downward. If we use the turtle coordinate axes, the two components of the vector **A** are

$$A_x = 0$$

and

$$A_y = -(\text{some constant})$$

The value of this constant depends on the strength of the local gravitational field. It will be smaller on Pike's Peak than in Death Valley. It will be smaller still on the Moon.

Turtle Trajectory

Let us write a program that will move the turtle through the air in the same way a projectile would move. There are three things we need to know in order to find the turtle trajectory. We need to know the initial velocity with which the projectile was thrown, the initial angle, and the local gravitational constant. Let us define a procedure with these three quantities as variables.

```
TO PROJECTILE :VELOCITY :ANGLE :GRAVITY
HT DRAW.GROUND
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
(PRINT [RANGE =] XCOR + 130)
END

TO DRAW.GROUND
PU SETPOS [130 -50]
PD SETPOS [-130 -50]
END
```

The ground is drawn, the turtle is placed at the left side of the screen and given the appropriate x- and y-components of velocity, and the procedure **STEP** is called.

```
TO STEP :VX :VY
SETH 90 FD :VX
SETH 0 FD :VY
IF YCOR < -50 [STOP]
STEP :VX :VY - :GRAVITY
END
```

In **STEP** the turtle steps through his trajectory. We set a time interval of 1 sec between steps. During 1 sec **:VX** will remain constant but **:VY** will decrease by an amount equal to the y-component of the acceleration (**:GRAVITY**). If the turtle falls below ground level (**YCOR** = -50) the procedure stops and control returns to **PROJECTILE** where the range (the net horizontal displacement) is printed. To see how the program works run **START**, where

```
TO START
PROJECTILE 9 45 .4
END
```

Try a few different values for the velocity, angle, and gravity. To smooth the trajectory try

```
TO STEP :VX :VY
SETPOS LIST XCOR + :VX YCOR + :VY
IF YCOR < -50 [STOP]
STEP :VX :VY - :GRAVITY
END
```

This will remove the ragged look of the path.

One important experiment we will cover in the Projects section is the determination of the maximum range of a projectile for a given initial velocity.

A More Accurate Trajectory

We noted in Chapter 3 that we make a small error by assuming that the velocity is constant when we issue the command **FORWARD :VELOCITY**. We can improve the accuracy by using the velocity at the beginning of the step plus half the change in the velocity. Thus the turtle moves by going **FORWARD :VELOCITY + :ACCELERATION / 2**. In our projectile problem there is no acceleration in the x-direction and the acceleration in the y-direction is **-:GRAVITY**. A more accurate procedure therefore is:

```
TO ACCURATE.PROJ :VELOCITY :ANGLE :GRAVITY
HT
DRAW.GROUND
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * (SIN :ANGLE) - :GRAVITY / 2
STEP :VX :VY
(PRINT [RANGE =] XCOR + 130)
END

TO DRAW.GROUND
PU SETPOS [130 -50]
PD SETPOS [-130 -50]
END

TO STEP :VX :VY
SETPOS LIST XCOR + :VX YCOR + :VY
IF YCOR < -50 [STOP]
STEP :VX :VY - :GRAVITY
END
```

```
TO START
ACCURATE.PROJ 9 45 .4
END
```

Let us put this more accurate program to work in order to reexamine the motion of a bouncing ball.

When an elastic ball rebounds from a hard surface, the x-component of the velocity is unchanged but the y-component is reversed. In the **SETP** procedure above, the projectile stopped when it reached ground level. To simulate the bouncing ball we require:

```
IF YCOR < -50 [MAKE "VY - :VY]
```

With this change, try **ACCURATE.PROJ 9 80 .4**. You should see a bouncing ball. (If you had used **PROJECTILE** instead of **ACCURATE.PROJ**, the ball would not rebound to the same height each time.)

Air Friction

You will notice that all the trajectories of **PROJECTILE** are symmetric. The right half of the trajectory is the same as the left half. If you observe the trajectory of any real projectile you will see that there is no symmetry. The trajectory will look more like that in Figure 4.1.

We can make the trajectory more realistic by including the effect of air friction. As a good approximation we may say that the frictional force is proportional to the velocity. For example, the x-component of velocity decreases by an amount proportional to **:VX** and the y-component decreases by an amount proportional to **:VY**. We have included this frictional force in the following listing.

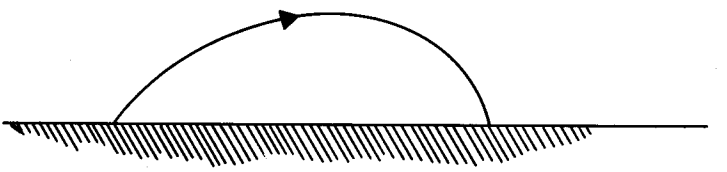


Figure 4.1 A Real Trajectory

```
TO PROJECTILE.FRICTION :VELOCITY :ANGLE :GRAVITY
HT DRAW.GROUND
MAKE "FRICTION .1
MAKE "VX :VELOCITY * COS :ANGLE
```

```
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
(PRINT [RANGE =] XCOR + 130)
END

TO DRAW.GROUND
PU SETPOS [130 -50]
PD SETPOS [-130 -50]
END

TO STEP :VX :VY
SETPOS LIST XCOR + :VX YCOR + :VY
IF YCOR < -50 [STOP]
STEP (:VX - :FRICTION * :VX) (:VY - :FRICTION * :VY
 - :GRAVITY)
END

TO START
PROJECTILE.FRICTION 20 30 .4
END
```

Try **START** once again and see what the trajectory looks like.

The Cathode Ray Tube

A television picture tube is a cathode ray tube (CRT). Electrons are heated in the cathode filament, accelerated toward the screen, and pass through a pair of deflecting plates. These deflecting plates carry a positive charge on one plate and a negative charge on the other. As shown in Figure 4.2, the top plate is positive. Since the negative electron is attracted to the positive charges and repelled by the negative charges, this causes the electron to move upward.

The amount of charge on the deflecting plates is determined by the potential difference (or the voltage) across the plates. The greater this voltage, the greater the deflection of the charges as they pass between the plates. By varying the potential, the electron beam can be directed to any chosen point on the screen.

The equations for the x- and y-components of acceleration of the charge between the plates are very similar to that of the projectile.

$$A_x = 0$$
$$A_y = \text{CONSTANT} * \text{VOLTAGE}$$

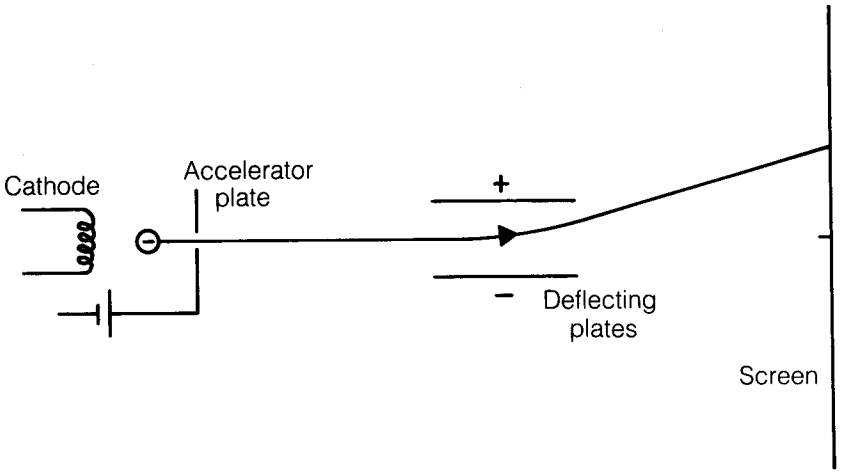


Figure 4.2 An Electron in Cathode Ray Tube

For simplicity we will assume that

$$A_y = .03 * :VOLTAGE$$

The factor .03 has been chosen to keep the turtle on the screen. The voltage can be either positive or negative.

These equations for the acceleration are of exactly the same form as those for the projectile. Therefore, the dynamics of the electron is the same as that of any other projectile. The sole difference is that the deflecting plates act on the electron only when the electron is between them. This variation has been included in the following listing for the cathode ray tube:

```
TO CRT :VOLTAGE
MAKE "ACCELERATION 3.N2 * :VOLTAGE
CLEARSCREEN
WINDOW HT FULLSCREEN
DRAW.SCREEN
DRAW.PLATES
SETPOS [-130 0]
PD ST
STEP 4 0
SPLITSCREEN
PRINT [GO AGAIN ( Y / N ) ?]
IF RC = "Y [PRINT [WHAT VOLTAGE?]] [STOP]
MAKE "V FIRST RL
CRT :V
END
```

```
TO DRAW.SCREEN
PU SETPOS [130 -120]
PD
SETH 0 FD 120 LT 90 FD 5 BK 5 RT 90 FD 120
PU
END

TO DRAW.PLATES
SETPOS [-40 -40]
SETH 90
REPEAT 2 [PD FD 80 LT 90 PU FD 80 LT 90]
IF :VOLTAGE > 0 [DRAW.PLUS 0 50 DRAW.MINUS 0 -50]
 [DRAW.PLUS 0 -50 DRAW.MINUS 0 50]
END

TO DRAW.PLUS :X :Y
SETPOS LIST :X :Y
PD SETH 0
FD 6 BK 12 FD 6
SETH 90
FD 6 BK 12 FD 6
PU
END

TO DRAW.MINUS :X :Y
SETPOS LIST :X :Y
PD FD 6 BK 12
PU
END

TO STEP :VX :VY
IF XCOR > 130 [STOP]
SETPOS LIST XCOR + :VX YCOR + :VY
STEP :VX :VY + ACCY
END

TO ACCY
IF (ABS XCOR) < 40 [OP :ACCELERATION] [OP 0]
END

TO ABS :NUM
IF (:NUM < 0) [OP -NUM] [OP :NUM]
END
```

The electron steps through its trajectory with **ACCX = 0** and **ACCY = :VOLTS** whenever the electron is between the two deflecting plates. This occurs when the absolute value (**ABS**) of the **XCOR** is < **40**. Notice that in

DRAW.PLATES the plates range from **XCOR = −40** to **XCOR = +40**. You may enter either positive or negative voltages. The plus or minus sign will be drawn next to the appropriate deflecting plate. Run **START** to see the CRT in action (as illustrated in Figure 4.3).

```
TO START
CRT 5
END
```

Problems

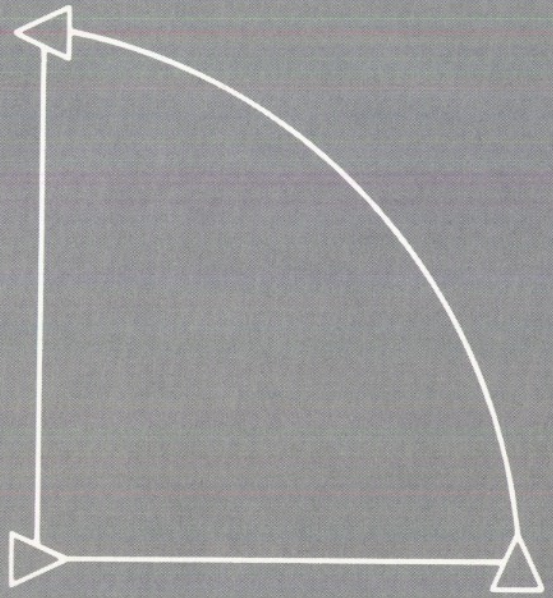
1. Use the **PROJECTILE** listing to determine the projection angle that produces the maximum range for a given velocity and acceleration of gravity. Use various fixed values of the velocity and gravity and show that the optimal projection angle is approximately constant. (There will be a small numerical error due to the fact that the turtle will fall slightly below ground before he stops.)
2. Using the results of Problem 1, determine how the maximum range depends on the magnitude of the initial velocity. Choose from one of the answers below:
   a. R(max) = (constant) * v
   b. R(max) = (constant) * $v^{1/2}$
   c. R(max) = (constant) * v^2
   d. R(max) = (constant) / v
3. Using the results of Problem 1, determine how the maximum range depends on **GRAVITY** for a fixed initial velocity. Choose from one of the selections below:
   a. R(max) = (constant) * G
   b. R(max) = (constant) * $G^{1/2}$
   c. R(max) = (constant) / G
   d. R(max) = (constant) / G^2
4. Using the results of Problem 3, how far could you throw a baseball on the Moon if you could throw the same ball 200 yards on Earth? The lunar gravity is .16 times the gravity on Earth.

Projects

1. We have shown how a ball can be made to bounce along the ground. Can you write a program to keep a ball bouncing around a room?
2. Write a procedure that will automate the CRT. Send out 11 electrons with voltages on the deflecting plates set at **5, 4, 3, 2, 1, 0, -1, -2, -3, -4,** and **-5**. There are many

ways of doing this. As an exercise in list manipulation try writing two procedures, **AUTO** and **BEGIN :VOLTAGES**. In **AUTO** you define a list called **"VOLTAGES** which is **[5 4 3 2 1 0 -1 -2 -3 -4 -5]. AUTO** then calls the recursive procedure **BEGIN :VOLTAGES** that runs **CRT FIRST :VOLTAGES** and then calls itself recursively with **BEGIN BF :VOLTAGES**. You will need to find some way to stop the program.

chapter



5 Projectile Motion II

Introduction

There are two kinds of problems we run into in generating numerical or graphical solutions to physics problems. One is that the turtle runs off the screen and the other is that the accuracy of the numerical algorithm breaks down when there are large changes in velocity over our one second time interval between steps. The midpoint approximation helps but it is not always good enough.

There are two remedies. If the turtle runs off the screen we can scale the problem down to screen size. If the velocity changes are very great over the one second step interval we can choose a smaller time interval. Actually both problems are solved by means of scaling techniques; in one case we scale the length and in the other we scale the time.

Scaling the Time

We have always assumed in stepping **FORWARD :VELOCITY** that the velocity is approximately constant over the given time interval (1 sec in our previous applica-

tions). But suppose the time interval is 1 sec, the velocity 40 ft/sec, and the acceleration 32 ft/sec/sec. During a 1 sec interval the velocity would change from 40 ft/sec to 72 ft/sec. It is hardly appropriate to assume that the velocity is constant over this 1 sec interval.

The solution to this problem is to use a shorter time interval—perhaps 0.1 sec. During 0.1 sec, the velocity would change from 40 ft/sec to 43.2 ft/sec if the acceleration were 32 ft/sec/sec. It is certainly a much better approximation to assume that the velocity is constant over the smaller time interval. For still greater accuracy, use an even smaller time interval.

If we choose a time interval of 1 sec, the turtle will move a distance equal to the velocity in 1 sec. This follows from the very nature of the definition of velocity as the distance moved per second. If the Difference in Time (**DT**) between steps is not equal to 1 sec, then the distance moved in a time **DT** is the distance moved in a unit time multiplied by the time **DT**. So where we moved **FORWARD :VX** before, we now move **FORWARD :VX * :DT**.

In a similar way, the change in the velocity in 1 sec is the acceleration. The change in the velocity during a time **DT** is **:ACCELERATION * :DT**.

These two modifications have been incorporated in **PROJECT.DT**:

```
TO PROJECT.DT :VELOCITY :ANGLE :GRAVITY
WINDOW HT
DRAW.GROUND
MAKE "DT .1
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
(PRINT [RANGE =] (XCOR +130))
END

TO STEP :VX :VY
INC.XY :VX * :DT :VY * :DT
IF YCOR < -50 [STOP]
STEP :VX :VY - :GRAVITY * :DT
END

TO DRAW.GROUND
PU SETPOS [130 -50]
PD SETPOS [-130 -50]
END
```

```
TO INC.XY :DX :DY
SETPOS LIST XCOR + :DX YCOR + :DY
END

TO START
PROJECT.DT 70 45 32
END
```

We have made the two fundamental changes in **STEP**. Instead of incrementing **XCOR** by **:VX** and **YCOR** by **:VY**, we now increment **XCOR** by **:VX** * **:DT** and **YCOR** by **:VX** * **:DT**. This modification also applies to the acceleration, so the second change is that **:VX** is now incremented by **−:GRAVITY** * **:DT** rather than **−:GRAVITY**. We have included the procedure **INC.XY :DX :DY** as well. The effect of this procedure is to increment the values of the x and y coordinates by **:DX** and **:DY**. The variable **:DX** stands for the *D*ifference between the old and new values of x. A similar definition applies to **:DY** in reference to the y values.

By choosing **:DT** small enough we may achieve any desired degree of accuracy. There remains, however, the problem of keeping the turtle on the screen when the distances are very large; for example, drawing the orbit of the planet Pluto. This problem is solved by drawing the orbit to scale. To see how this is done let us take up the general problem of scaling.

Scaling the Distance

When we draw a house on a piece of paper we are applying principles of scaling. We recognize that we cannot draw a full size house on an 8 1/2 × 11 inch sheet of paper. We choose some reasonable *scale* and draw the house accordingly. We will define the variable **SCALE** to be the number of turtle units per foot (or mile, kilometer, and the like). A turtle unit is the distance between an **XCOR** of 0 and an **XCOR** of 1.

If we fire a projectile with a range of 1000 ft, the trajectory will not fit on the screen if we let 1 t.u. (turtle unit) be equal to 1 ft. The screen is approximately 250 t.u. wide. To get a range of 1000 ft on a 250 t.u. screen it is necessary to choose a scale factor of 250/1000 t.u./ft or 1/4 t.u./ft. If we multiply the range of 1000 ft by the scale factor of 1/4

t.u./ft we get a screen range of 1000 * 1/4 = 250 t.u., which will fit on the screen.

The trajectory calculations are carried out as usual without regard to scaling considerations. There are only two instances where the scale factor must be applied. First, when we draw to the screen, we must convert the distance (in feet, meters, or miles) to turtle units. Second, when we extract information from the screen we must convert the turtle units into distance (measured in feet, meters, or miles). This process is illustrated in **PROJECT.SCALE**.

```
TO PROJECT.SCALE :VELOCITY :ANGLE :GRAVITY
WINDOW HT
DRAW.GROUND
MAKE "DT .1
MAKE "SCALE .25
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
(PRINT [RANGE =] (XCOR + 130) / :SCALE)
END

TO DRAW.GROUND
PU SETPOS [130 -50]
PD SETPOS [-130 -50]
END

TO STEP :VX :VY
INC.XY :VX * :DT * :SCALE :VY * :DT * :SCALE
IF YCOR < -50 [STOP]
STEP :VX :VY - :GRAVITY * :DT
END

TO INC.XY :DX :DY
SETPOS LIST XCOR + :DX YCOR + :DY
END

TO START
PROJECT.SCALE 70 45 32
END
```

The two places the scale factor is applied are where we move **FORWARD** and where we print **XCOR**. In the first instance we convert feet into turtle units and, in the second, we convert turtle units back into feet.

Figure 5.1 PROJECT.SCALE 70 45 (.16*32)
PROJECT.SCALE 70 45 32

To see how far you could throw a baseball on the Moon where gravity is 16% of gravity on the Earth, run **PROJECTILE.SCALE 70 45 .16 * 32**. The velocity is 70 ft/sec (about 50 mph), the projection angle 45°, and the acceleration of gravity .16 * 32. To get a visual comparison with the same trajectory on the Earth, run **PROJECT.SCALE 70 45 32**. You should see a significant difference in the range (960 ft versus 153 ft), as shown in Figure 5.1.

Problems

1. If you can throw a baseball with a speed of 88 ft/sec (60 mph), how high can you throw the ball? (Answer: 144 ft)
2. If you can throw a baseball with a speed of 88 ft/sec, what is the maximum range on Jupiter? On Jupiter, gravity is 2.56 times as great as gravity on the Earth. (Answer: 100 ft)
3. Assume that you can jump 3 ft off the ground (that is, raise your center of gravity by 3 ft). How high could you jump on the Moon? On Jupiter?
4. If you can long jump 10 ft on the Earth, how far could you long jump on the Moon? On Jupiter?

Projects

1. Artillery practice: A cannon is being fired from a building of arbitrary height. The target is a small abandoned house. The house should disappear if it is hit (**PENERASE**). The muzzle velocity of the cannon is fixed and equal to 80 ft/sec. The configuration is illustrated in Figure 5.2.
2. Can you modify the projectile program so that both the range and height are printed when the program ends? (Hints: You might **MAKE "SET.FLAG? "YES** in **PROJECTILE**. In **STEP**, check to see whether **:VY** is nega-

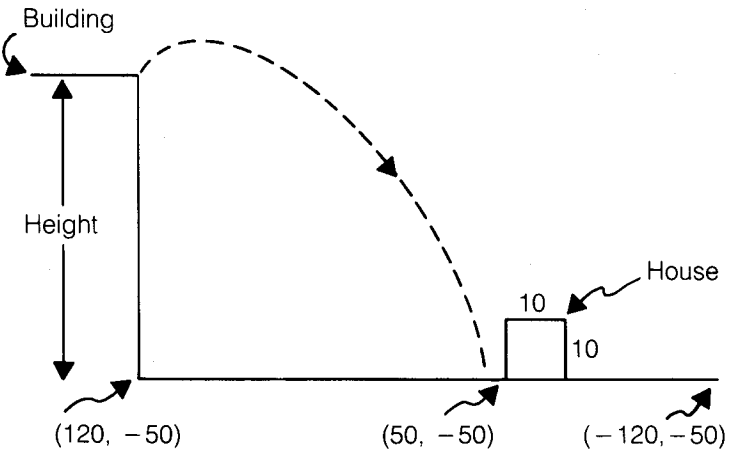
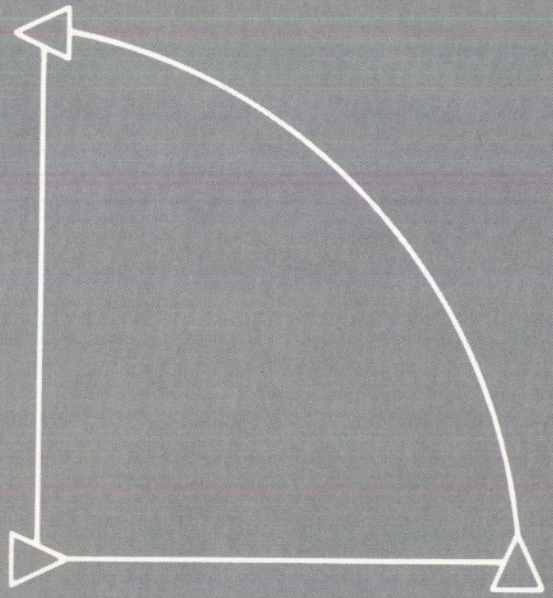


Figure 5.2 Shell Trajectory

tive *and* whether the flag is set. If so, set the flag to **NO** and note the height of the turtle. Another technique is to note that when **VY.OLD * VY.NEW** is negative, there is a change in the direction of motion.)

chapter



6 The Monkey, the Hunter, and Einstein's Principle of Equivalence

Introduction

In this chapter, we will describe an experiment for you to perform—an expanded project. The experiment consists of solving the following problem. A monkey rests on the limb of a tree 200 ft above the ground. The base of the tree is 200 ft from a hunter. The hunter is attempting to shoot the monkey. The monkey, however, is wise in the ways of avoiding the hunter's bullets. The moment the monkey sees the flash of the gun he drops from the limb of the tree so the bullet passes harmlessly over his head.

The hunter is no fool either. She sees what is happening. The monkey is dodging the bullet by jumping from the tree limb. "Surely," she thinks to herself, "there must be some point at which I might aim the gun so that the bullet will strike the monkey even if he does drop from the limb."

The hunter returns to her safari tent to consult with the turtle over this perplexing problem. She writes a Logo program to simulate the situation. In this program, the turtle is placed on the limb of the tree 200 units above the ground with instructions to drop the moment the projectile

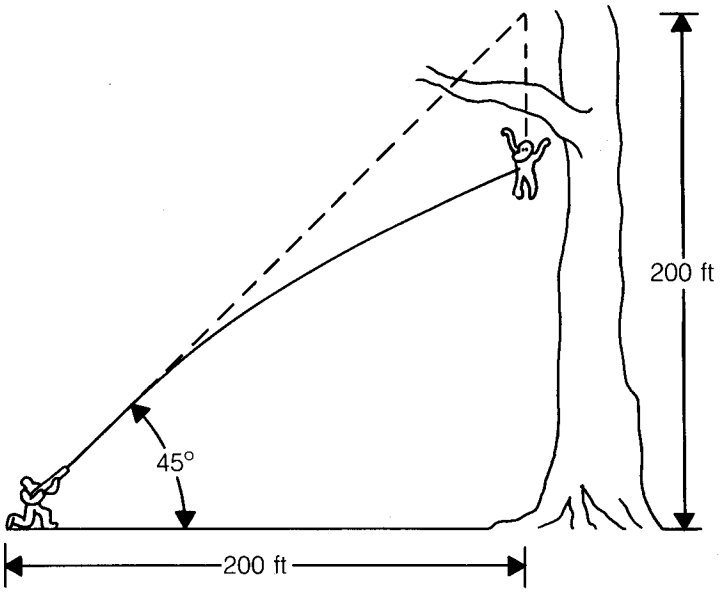
is fired. The projectile comes from a point 200 units from the base of the tree with a velocity and direction to be specified by the hunter. By trial and error she hopes to find the particular angle and velocity which will bag her the monkey.

The results of the hunter's Logo program are surprising. She finds that if she sets the angle so that the gun is pointed below the turtle on the tree limb, the projectile always passes below the falling turtle. If the angle is set above the tree limb the projectile always passes above the falling turtle. These results are independent of the muzzle velocity of the projectile. However, the hunter finds that if she aims the gun directly at the turtle on the limb, the projectile strikes him every time—and this striking result is independent of the muzzle velocity of the projectile. If the muzzle velocity is low, the projectile strikes the turtle close to the ground. If the muzzle velocity is high, the projectile strikes the turtle close to the limb. Both examples are illustrated in Figure 6.1.

Your task is to reconstruct the hunter's Logo program and verify the results of her experiments. You may find the **PROJECTILE** program in Chapter 5 useful. You will also require two turtles. If your Logo does not have sprites, use the **ASK** procedure from *Notes to the User*.

Einstein's Principle of Equivalence

It is not too difficult to solve the equations of motion for the monkey and the projectile and confirm the fact that the monkey is struck if the gun is aimed directly at him. It is, however, easier to convince ourselves of the correctness of this result in another way. Imagine that the whole system of hunter, monkey, and tree are enclosed in a gigantic elevator as shown in Figure 6.2. Let us examine the dynamics inside the elevator if the cable is cut the moment the gun is fired and the elevator falls freely. As soon as the cable is cut, the elevator begins to fall at a rate determined by the local acceleration of gravity. If an object in the elevator were dropped the moment the cable was cut, it would remain at rest. Relative to the ground, it would fall with the local value of the gravitational acceleration; but so does the elevator.



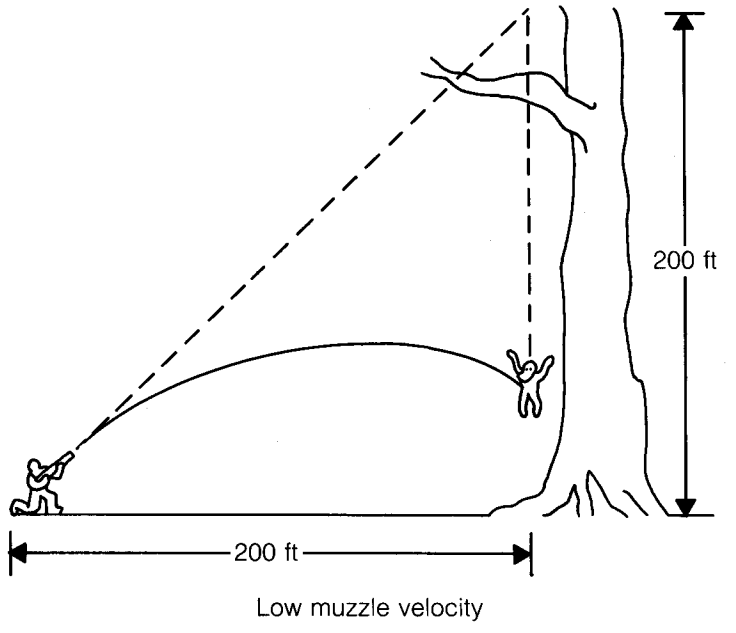


Figure 6.1 High and Low Muzzle Velocity

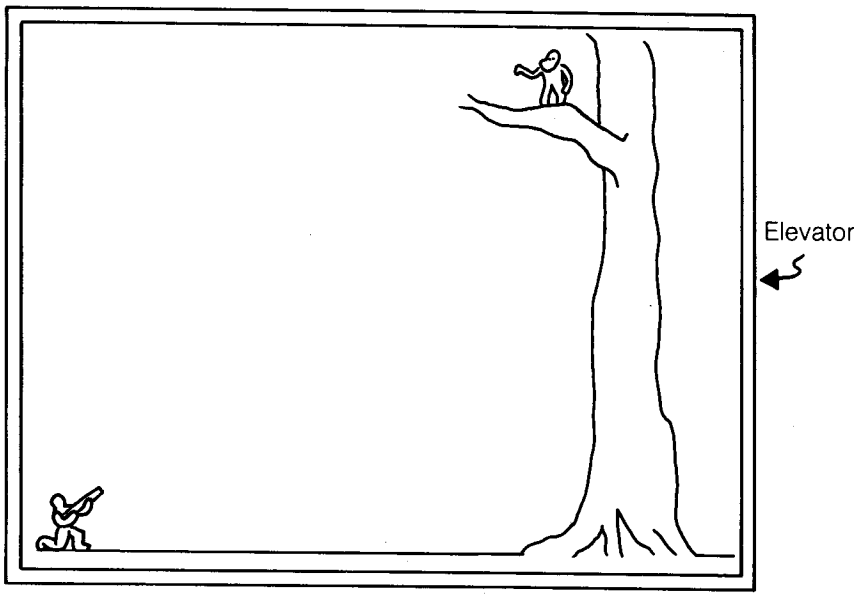


Figure 6.2 Monkey and Hunter in Elevator

Thus the two accelerate together and therefore there is no relative motion between the object dropped and the falling elevator. Everything within the elevator appears to be weightless. If at rest, it remains at rest. If in motion, it moves with constant velocity in a straight line. To the hunter within the elevator there are two possible interpretations of the events that follow the cutting of the cable. She might assume that she was in an elevator whose cable had just been cut *or* that in some mysterious way, gravity had suddenly been suspended—everything within the elevator had just become weightless. Indeed this is but a special case of Einstein's principle of equivalence:

> It is impossible to distinguish between an accelerated frame of reference and a uniform gravitational field.

Armed with this principle, Einstein was able to demonstrate the association between gravity and the geometrical properties of space.

A Different Point of View

In the frame of reference in which the monkey and the hunter are located in the elevator, it is easy to understand

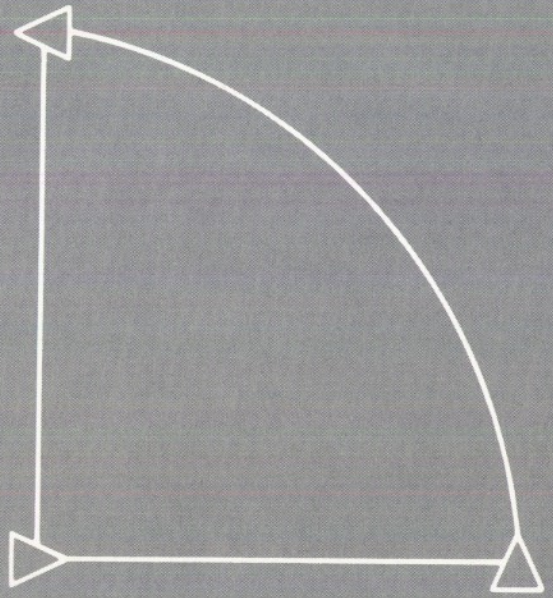
why the projectile hits the monkey. First we observe that *after the cable is cut there is no effective gravity in the elevator car.* Second, *the cutting of the cable in no way affects the motion of either the projectile or the monkey.* Neither the projectile nor the monkey is touching the walls of the elevator so the elevator has no affect on them. (The hunter and the tree are very much affected by the cutting of the cable. They begin to accelerate. But then they are in contact with the elevator.) Let us study the motion of the monkey and the projectile from the point of view of an observer within the freely falling elevator. This observer feels weightless. Everything in the elevator is weightless. If the monkey is weightless, he will remain at rest after letting go of the tree limb. (The tree accelerates upward but the monkey remains at rest.) The projectile, after it has left the barrel of the gun is moving in a weightless environment and so, moves with constant velocity *in a straight line.* If the gun is aimed directly at the monkey, the monkey remains at rest, and the projectile moves in a straight line, then the projectile must clearly strike the monkey. The phenomenon is very easy to understand in the accelerating frame of reference.

The laws of physics should be independent of the interpreter. Both the monkey and the hunter should be able to apply basic physical principles and come to the same ultimate conclusion. The only real difference between the two is their point of view. To the hunter the elevator is at rest and everything in it subject to a uniform gravitational field. To the monkey, the elevator is accelerating upward and there is no gravity. It so happens that for this particular problem it is easier to obtain a solution by taking the monkey's point of view.

Project

1. Simulate this experiment. Let the muzzle velocity and angle be input variables. **ASK 0** to be the projectile and **ASK 1** to be the falling monkey.

chapter



7 Escape Velocity

Newton's Law of Universal Gravitation

In Chapter 3 we studied the motion of a falling body. Now we will study the reverse motion—a body projected straight up. The projection velocity will be so great that we will have to consider the possibility that the body will move far from the Earth's surface. If it gets very far away, we cannot assume that the acceleration due to gravity will be constant. Certainly if the body gets *very* far away we can *neglect* the Earth's pull on the body. Therefore, our first task is to study the effect of the Earth's pull on a body that is not close to the surface of the Earth.

This problem of gravitational forces had puzzled scientists for over two thousand years. To the Greek astronomers there were many questions to be answered. Why does a stone fall to the ground? (Aristotle's answer to this question was: That's where it belongs. That is its "natural state.") Why does the Moon rotate about the Earth? Is there any connection between the motion of the Moon about the Earth

and the trajectory of a stone? If so, why doesn't the Moon just fall to the Earth the way a stone does?

Newton recognized the connection between the motion of a projectile and the motion of planetary bodies. In his treatise *Principia Mathematica*, we find an illustration very similar to that of Figure 7.1. It represents a body being thrown from a *very* tall pole. For low velocities the trajectory is just what we would expect. As the velocity increases, however, the point at which the body strikes the Earth is further and further from the pole. It seems quite reasonable that at some very high velocity it will return to the pole without ever striking the ground. Perhaps this is precisely how the Moon behaves. It has a velocity which permits it to "fall" in the Earth's gravitational field in such a way that its orbit is a circle. To verify this conjecture what Newton needed was a quantitative law that he could test. The law he proposed is called the *law of universal gravitation* and states that:

Any two bodies are attracted to each other with a force which is proportional to the product of the mass of each body and inversely proportional to the square of the distance separating the centers of the two bodies. The force is attractive and is directed along the line joining the two bodies.
or

$$F \propto \frac{mM}{r^2}$$

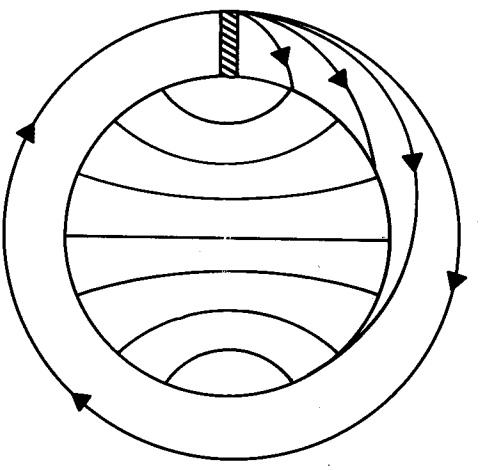


Figure 7.1 Ball Thrown From Tall Pole

where F is the force on *either* body, m and M are their respective masses and r the separation between the mass centers. This law applies to all bodies. Your body is attracted to the Earth with a force proportional to your mass and the mass of the Earth. If your body was on the Moon, your mass would be the same but the mass of the Moon is much less than that of the Earth and so the gravitational attraction between your body and the Moon would be much less. Therefore, you would weigh less on the Moon.

You can measure this force of attraction between yourself and the Earth by putting a spring under your feet and measure the compression. This is essentially what you do when you weigh yourself on a bathroom scale.

In order for the gravitational force to be large there must be at least one large mass. Although there is a gravitational force between you and your car, it is very small, since neither you nor your car have a very great mass (certainly nothing like the mass of the Earth). It is not altogether negligible and so, with very delicate instruments, it is possible to measure the attractive force between two bodies of known mass m and M and so measure the factor of proportionality in the above equation. The constant is called the univeral gravitational constant and the symbol G is used. We may now write:

$$F = \frac{GmM}{r^2} \qquad [1]$$

where G is a measurable quantity. (Its value is 6.67×10^{-11} in metric units.)

Newton's Law of Motion

Newton's law of universal gravitation gives us the force between any two bodies. But if we are to determine the escape velocity of a rocket we must know how this force affects the motion. This is given to us by *Newton's Law of Motion* which states that:

The sum of the external forces acting on a body is equal to the product of the mass and the acceleration of the body.
or

$$\text{Sum of the forces} = ma$$

Generally we write:

$$F = ma \qquad [2]$$

where F represents the sum of all the external forces. This law tells us how a body responds to a force. In our particular example of a body acted on by a gravitational force of a body of mass M at a distance between centers of r, we may combine Newton's laws of motion and universal gravitation and write,

$$\frac{GmM}{r^2} = ma$$

Since the mass (m) is common to both sides of the equation we have

$$a = \frac{GM}{r^2}$$

One last observation and we are finished. This acceleration is attractive. Thus if the turtle is moving along the y = axis the acceleration will be directed down toward the origin where the planet is located. In this case the acceleration will be negative. We have then

$$a = \frac{-GM}{r^2} \qquad [3]$$

This is the rule we need. It tells us the acceleration of any body at whatever distance (r) from the Earth's center. We see that the acceleration decreases as the body gets further

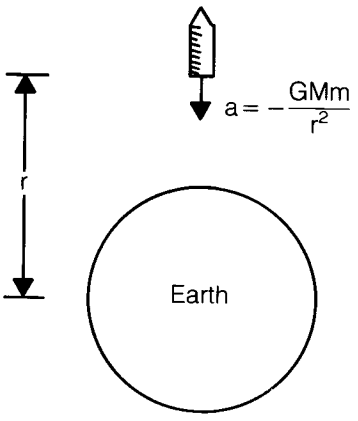


Figure 7.2 Acceleration Due to Gravity

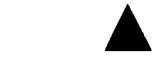
and further from the Earth. At *very* great distances the acceleration due to the Earth's gravity is negligible.

Of course this result applies to any two bodies. In general it says:

The acceleration of a body due to the gravitational attraction of another body of mass M at a distance r between centers is GM/r^2.

Notice that the acceleration of a given body depends only on the mass of the *other* body. This is due to the cancellation of the mass (m) in equation 3.

If we apply equation 3 to the *surface* of the Earth and substitute the appropriate numbers we find that a = 32 ft/sec/sec. If we did the same calculation for the Moon we would find an acceleration of only 5.1 ft/sec/sec and on the planet Jupiter a = 85 ft/sec/sec. If you fell out of bed on Jupiter you might break your pajamas.

Escape Velocity 

Now that we have an expression for the acceleration let us tackle the escape velocity problem. What must be the velocity of a space ship when it has burned up its fuel if it is to escape the Earth's gravitational field? By "escape the Earth's gravitational field" we mean that the rocket is capable of coasting to indefinitely great distances from the Earth and will never fall back. We shall make the assumption (generally a good assumption) that the fuel is burned out at a point not far from the Earth's surface. When we speak of *the escape velocity* we mean that for any velocity greater than this value the rocket will coast to infinity, and for any velocity less than this value it will fall back to Earth.

Our procedure for solving the escape velocity problem will be very similar to the method we used to solve the free fall problem. The basic motion will be determined by the procedure:

```
TO STEP :VEL
FORWARD :VEL
STEP :VEL + ACC
END
```

The main difference is that the acceleration is no longer a constant but will be determined by equation 3. Another minor complication is that we will be blasting the turtle

very far from the Earth and if we are to keep him in sight we will have to use the **WRAP** mode. This means that we cannot keep track of his position using **YCOR** as before. But we shall see that Logo is more than a match for this difficulty.

The **ESCAPE** program tests to see if the turtle has been given a sufficiently high velocity that will allow him to escape the Earth's gravity.

```
TO ESCAPE :VEL :MASS
WRAP
CS FULLSCREEN
MAKE "RADIUS 40
DRAW.EARTH :RADIUS
SETH 0
MAKE "Y :RADIUS
ST
STEP :VEL + ACC / 2
SPLITSCREEN PRINT [SPLAT!]
END

TO STEP :VEL
IF AND :VEL < 0 :Y < 279 [PE]
IF :Y < :RADIUS [STOP]
FD :VEL
MAKE "Y :Y + :VEL
STEP :VEL + ACC
END

TO ACC
OP -:MASS / (:Y * :Y)
END

TO CIRCLE :RAD
MAKE "PI 3.14159
HT CS
PU FD :RAD RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :RAD / 12]
LT 90 - 15
END

TO DRAW.EARTH :RADIUS
CIRCLE :RADIUS
END

TO START
ESCAPE 6.6 1000
END
```

The first procedure, **ESCAPE :VEL :MASS**, sets the stage.

We use the **WRAP** mode so that we can always see the turtle. The radius of the Earth is set at 40 (this may be changed later) and a circle of radius 40 is drawn. The turtle is placed on top of the Earth directed vertically, ready for blast off.

The next command, **MAKE "VEL :VEL + ACC / 2** changes the initial velocity to the velocity after 1/2 sec. To understand the reason for this command we must go back to our earlier observation (see Free Fall) that all of our solutions are approximations. The inaccuracies develop because of the fact that the velocity is not constant but changes continuously from the beginning of the step to the end of the step. By using the average velocity between steps, we improve the accuracy of our results.

Another point to notice in the **STEP** procedure is the command **MAKE "Y :Y + :VEL**. Since the turtle moves along the y=axis and the Earth is centered at the origin, the distance from the center of the Earth to the turtle is equal to the y coordinate. In the past we have used the built-in **YCOR** to keep track of the y coordinate. We cannot do that here since the turtle uses the **WRAP** mode. Each second, the value of y increases by **:VEL** and so we simply add it to y during each cycle. The conditional (**IF**) statements are used to allow the turtle to erase his trail if he falls back to Earth and to stop him when he reaches the Earth.

The units have been chosen to keep the turtle on the screen. We have arbitrarily set G equal to one. In the next section we will consider the problem of how to cope with astronomical dimensions on a small TV or monitor screen.

Problems

1. Run the **START** procedure to see what happens. In **START** the velocity is set equal to 6.6 and the mass of the planet is chosen to be 1000. Estimate (approximately ± .5) the minimum velocity needed for escape if the mass of the planet is 400.
2. Repeat Project 1 with a planetary mass of 1600.
3. Can you predict how the minimum escape velocity will depend on the mass of the planet? Use the results of Problems 1 and 2 and choose from the following possibilities. The escape velocity is proportional to:
   a. the square of the mass of the planet.
   b. the square root of the mass of the planet.
   c. the mass of the planet.

d. the inverse of the mass of the planet.
e. the reciprocal of the square of the mass of the planet.

Projects

1. Modify **ESCAPE** so that the radius of the planet is an input variable (that is, **TO ESCAPE :VEL :MASS :RADIUS**). Change the radius to several different values and see if you can determine how the escape velocity depends on the radius of the planet.

2. A great hole is dug in the Earth which passes straight through the center and out the other side. Let us study the motion of the turtle as he falls through this hole.

   We need a new rule for the acceleration of the turtle due to the pull of the Earth. Suppose the turtle is a distance y from the center of the Earth as in Figure 7.3. It can be shown from Newton's law of universal gravitation that the acceleration due to gravity is given by

   $$A = -GM/y^2$$

   where M is now the mass *inside* a sphere of radius y. Let M_0 be the *total* mass of the Earth and R the radius of the Earth.

   If the density of the Earth were constant the mass inside a sphere of radius y is

   $$M = M_o \frac{(4/3)\pi y^3}{(4/3)\pi R^3}$$

   or more simply

   $$M = M_o y^3/R^3$$

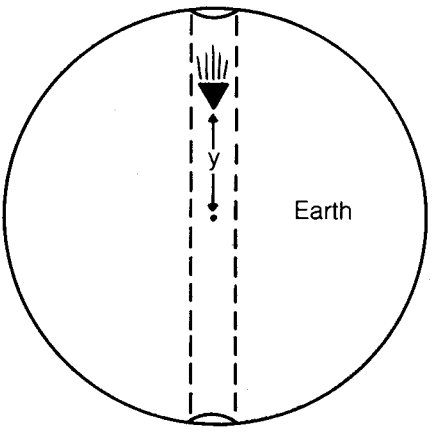


Figure 7.3 Turtle Passing Through the Earth

The acceleration therefore is

$$a = -GM/y^2 = -GM_0y/R^3$$

Notice that the acceleration is a linear function of y rather than an inverse square function as before.

Let G = 1, M_0 = 400, and R = 40. Let the turtle fall in the hole and see what becomes of him.

Scaling

In the previous section we discussed the escape velocity using turtle graphics. We would now like to do some realistic problems. What, for example, is the escape velocity from the Earth, or the Moon, or Mars?

To answer such questions we must modify the **ESCAPE** program. We will call this program **ESCAPE.SCALE**.

```
TO ESCAPE.SCALE :VELOCITY :MASS :RADIUS
WRAP
CS FULLSCREEN
MAKE "G 6.67N11
MAKE "SCALE 50 / :RADIUS
DRAW.PLANET :RADIUS * :SCALE
SETH 0
MAKE "Y :RADIUS
MAKE "DT :RADIUS / (5 * :VELOCITY)
ST
STEP :VELOCITY + :DT * ACC / 2
SPLITSCREEN PRINT [SPLAT!]
END

TO DRAW.PLANET :R
CIRCLE :R
END

TO CIRCLE :RAD
MAKE "PI 3.14159
HT CS
PU FD :RAD RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :RAD / 12]
LT 90 - 15
END

TO STEP :VEL
IF AND :VEL < 0 (:Y < 279 / :SCALE + :RADIUS) [PE]
IF :Y < :RADIUS [STOP]
FD :VEL * :DT * :SCALE
```

```
MAKE "Y :Y + :VEL * :DT
STEP :VEL + :DT * ACC
END

TO ACC
OP - :G * :MASS / (:Y * :Y)
END

TO START
ESCAPE.SCALE 10300 6.0E24 6300000
END
```

The first thing we must do is to **MAKE "G 6.67E−11** which is the numerical value of G in metric units. Next we recognize that we cannot draw a circle of radius 6,300,000 m (the radius of the Earth in meters) on the screen. We need to scale the dimensions so that the planetary body will fit on the screen. You will see in **ESCAPE.SCALE** the command **MAKE "SCALE 50 / :RADIUS** where **:RADIUS** is radius of the planetary body so that later, when we **DRAW.PLANET** with a radius of **:RADIUS * :SCALE** we will obtain a circle whose radius is 50 turtle units. This is a reasonable size for the planet on the screen. We will apply the scale factor only when we do something that applies to the screen. In all of our numerical calculations we will use the actual numbers that apply to the particular planetary body. There are only two places in the program where we address the screen. The first is when the planet is drawn and the second is when the turtle goes **FORWARD** on the screen (see the **STEP** procedure). All other calculations are carried out as before with no scaling.

There is a second modification necessary. We must use a time interval between **STEP**s which is greater than the 1 sec interval we have used earlier. In **ESCAPE.SCALE** you will see **MAKE "DT :RADIUS / (5 * :VELOCITY)**. With this choice of **DT** the distance moved in each step (**:VELOCITY * :DT**) will be **:RADIUS / 5** or one-fifth of the planetary radius. This is a reasonable step length. Not so small that the program takes too long to run and not so long that the numerical algorithm breaks down.

With these modifications we can now consider some realistic applications. Some of the many possible applications are included in the Projects and Problems sections.

Problems

4. The mass and radius of several planetary bodies are listed in the table. Fill in the table by determining the escape velocities.

	Mass	Radius	Escape Velocity
Earth	6×10^{24} kg	6,300,000 m	11,000 m/sec
Moon	7.3×10^{22} kg	1,740,000 m	?
Mars	6.4×10^{23} kg	3,300,000 m	?
Mercury	3.2×10^{23} kg	2,400,000 m	?

5. You have paused in your space journey to visit a small asteroid. The asteroid is a spherical mass of ice with a radius of 10 mi (16,000 m) and mass of 8.5×10^{15} kg. You must be very careful how you move about on this asteroid. If you jump for joy you may achieve escape velocity and fly off into space to become just another planetary body. To what velocities should you limit your jump for joy?

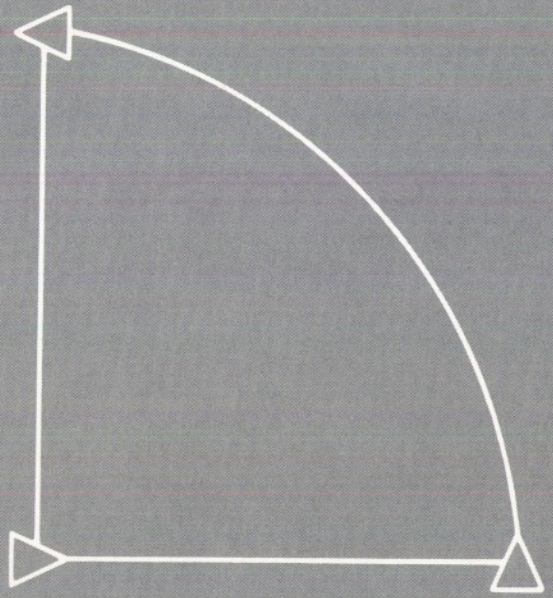
6. It is possible to escape the Earth's gravitational pull but not escape from our solar system. The sun, although much further away than the Earth, has a much larger mass. Let us suppose that your rocket ship is located at a distance of 1.5×10^{11} m from the sun. (This is the mean distance between the Earth and the sun.) Starting from this point (set **RADIUS = 1.5 10^11**) what velocity at burnout is required to escape the sun's pull? The mass of the sun is 2.0×10^{30} kg.

Projects

3. The upward and downward motion of a rocket are symmetric. The velocity at any point on the way up is equal to the velocity at the same point on the way down. If the rocket just barely escapes the Earth's gravity it will coast to a very great distance where its velocity is practically zero. From the symmetry of the trajectory we might determine the escape velocity of a rocket by releasing it from a very great distance with zero velocity and note its velocity when it strikes the Earth. This relieves us of the tedium of trying many different velocities to see which allows the rocket to escape the Earth's gravity. Modify **ESCAPE.SCALE** to determine the escape velocity in this way.

4. Use the results of Project 2 to determine the time it would take for a trans–earth train to pass through the Earth.

chapter



8 Planetary Motion

Introduction

The dynamical laws that govern the motion of a baseball and the flight of a rocket ship are the same as the laws that govern the motion of the planets. Each planet is attracted toward every other body in the solar system. The sun is by far the most massive body in our solar system and therefore has the greatest effect on the motion of the planets. In fact, the mass of the sun is so great (300,000 times the mass of the Earth) that we may neglect, for the most part, the effect of all other bodies on the motion of the planets.

We would like to study what effect the sun has on the motion of the planets. What is the shape of the planetary orbits? How does their speed depend on the distance from the sun? How does the Martian year compare with the Venusian year?

The Equations of Motion

We have seen that there is a gravitational force between any two bodies. This force is given by Newton's law of uni-

versal gravitation:

$$F = \frac{GmM}{r^2}$$

where r is the distance between centers. It is Newton's law of motion which tells us how this force affects the motion of the body. It produces an acceleration given by:

$$F = mA$$

Combining these two laws we see that the acceleration of the body of mass m is given by

$$A = \frac{GM}{r^2}$$

Now the acceleration is a vector. Since the gravitational force is attractive, the acceleration of a given planet is directed toward the sun. If we place the sun at the origin of the x-y coordinate system, we can see from Figure 8.1 that the x and y components of the acceleration vector are

$$A_x = -A \sin(\text{heading})$$

$$A_y = -A \cos(\text{heading})$$

But we also see from the figure that

$$\sin(\text{heading}) = x/r$$

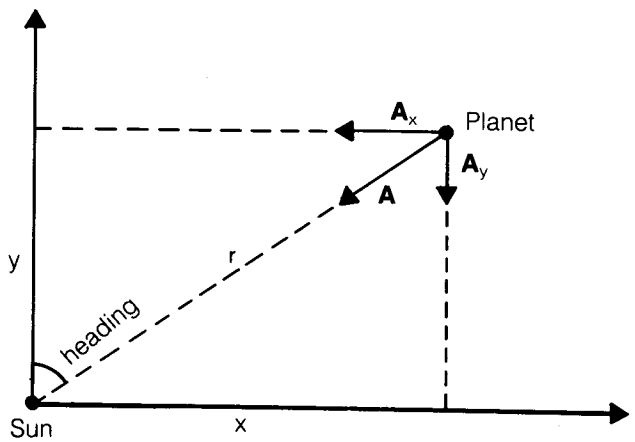


Figure 8.1 Components of the Acceleration Vector

and

$$\cos(\text{heading}) = y/r$$

Therefore

$$A_x = -A\,x/r$$

and

$$A_y = -A\,y/r$$

or since $A = GM/r^2$

$$A_x = -GMx/r^3$$

and

$$A_y = -GMy/r^3$$

It is useful to compare these equations to the fundamental equations of motion we used to determine escape velocity in Chapter 7. If we consider the special case of a body moving along the y-axis, then $x = 0$ and $y = r$. Therefore

$$A_x = 0$$

and

$$A_y = -GMy/r^3 = -GM/r^2$$

which is exactly the equation of motion we employed to study the escape velocity of a rocket ship.

The Logo Procedure

Now that we have the laws for the acceleration we may determine the trajectory of any planetary body. We do this just as we have done in the past: **FORWARD** velocity, change the velocity by adding the acceleration and go back to **FORWARD** velocity. In **ORBIT** we use units chosen to keep the turtle conveniently on the screen. Later we will consider more realistic problems by scaling the distances.

```
TO ORBIT :X :Y :SPEED :DIRECTION
WINDOW CS HT
MAKE "MASS 4000
CIRCLE 20
PENUP
SETX :X SETY :Y
FIND.R
MAKE "VX (:SPEED * SIN :DIRECTION) + ACCX / 2
```

```
MAKE "VY (:SPEED * COS :DIRECTION) + ACCY / 2
FULLSCREEN PENDOWN
STEP :VX :VY
END

TO FIND.R
MAKE "R SQRT ((SQ XCOR) + SQ YCOR)
MAKE "R3 :R * :R * :R
END

TO SQ :X
OP :X * :X
END

TO STEP :VX :VY
INC.XY :VX :VY
FIND.R
STEP :VX + ACCX :VY + ACCY
END

TO INC.XY :DX :DY
SETPOS LIST XCOR + :DX YCOR + :DY
END

TO ACCX
OP -:MASS * XCOR / :R3
END

TO ACCY
OP -:MASS * YCOR / :R3
END

TO CIRCLE :R
MAKE "PI 3.14159
HT PU HOME
FD :R RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :R / 12]
LT (90 - 15)
END

TO CIRCULAR
ORBIT 50 0 8.9 0
END

TO ELLIPSE
ORBIT 120 0 5 0
END

TO COMET
```

```
ORBIT -150 -90 8 31
END
```

In **ORBIT** you may assign any value to the x and y coordinates of the planetary body, give it any speed, and set its direction of motion as you please. The mass of the central body (the turtle's sun) is chosen to be 4000 and we have set G = 1. The central body has a radius of 20 turtle units (**CIRCLE 20**). The orbiting body is placed in position and given the appropriate x and y components of velocity. Notice that once again we have adjusted the velocity to be the midpoint velocity by adding 1/2 the change in the velocity (**ACCX** and **ACCY**). (Take care to enclose the sine function in parentheses. It acts on everything that follows until it reaches a parenthesis or an end of line.) The turtle then begins to step off the trajectory of the orbiting body.

One of the things we will examine in the Projects section is the shape of planetary orbits. Most of the planets in our solar system move in roughly circular orbits, but there are other shapes as well. The path may be an ellipse (with the sun at one of the focii), a hyperbola, or a parabola. Another thing we would like to examine is the relationship between the period of the orbit (the time it takes to rotate about the sun) and the distance of the planet from the sun. What is the length, for example, of the Martian year? (One Martian year, you say? Quite right, but how many Earth days in a Martian year?)

Vector Addition of Velocities

The basic problem in determining the trajectory of the orbiting body is to determine the effect of the acceleration in changing the velocity. We have solved this problem in **ORBIT** by resolving vectors (velocity and acceleration) into their components. The virtue of doing this is that the components add in the same way that ordinary numbers add—two plus two is four. But a vector two units due east plus a vector two units due north is not a vector four units in length. Vectors are a little more difficult to add. We have seen in Chapter 1 that turtle graphics is made to order for adding vectors. We can use this capability to solve our orbit problem.

In Figure 8.2, turtle 0 represents the orbiting planet. Its current velocity is v and its acceleration A. We obtain its new position by the command: **FORWARD :V**. We must then obtain the velocity at the new position in order to repeat the operation. We may set turtle 1 the task of finding this new velocity. He does this by adding the acceleration A to the velocity v. The sum is the velocity at the new position. This basic process is carried out repeatedly in **ORBIT.NEW**.

We use turtle 0 as the orbiting body and turtle 1 to keep track of the velocity for us.

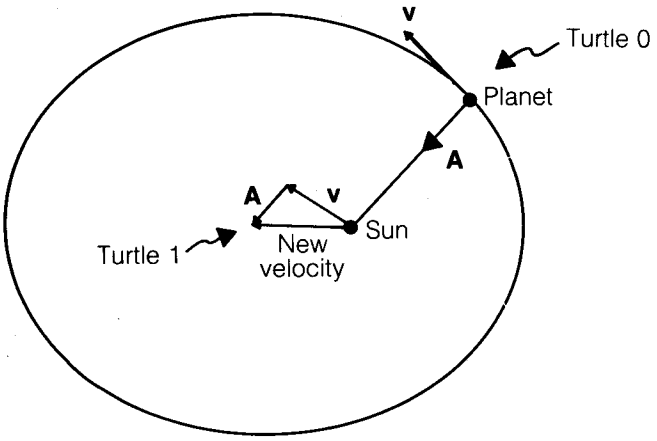


Figure 8.2 Turtle 0 Orbiting and Turtle 1 Computing Velocity

```
TO ORBIT.NEW :X :Y :SPEED :DIRECTION
HOME.ALL
CS WINDOW
MAKE "MASS 4000
PENUP
SETPOS LIST :X :Y
ASK 1 [PENUP HT HOME PENDOWN SETH :DIRECTION FD
 :SPEED]
FULLSCREEN PENDOWN HT
STEP
END

TO STEP
SETH :DIRECTION
FD :SPEED
CHANGE.VEL
STEP
END
```

```
TO CHANGE.VEL
SETH TOWARDS [0 0]
MAKE "ACC.ANGLE HEADING
MAKE "ACC :MASS / ( ( SQ XCOR ) + SQ YCOR )
ASK 1 [ADD.ACC]
END

TO ADD.ACC
SETH :ACC.ANGLE
FD :ACC
MAKE "SPEED SQRT ( SQ XCOR ) + SQ YCOR
SETH TOWARDS [0 0] RT 180
MAKE "DIRECTION HEADING
END

TO SQ :X
OP :X * :X
END

TO START
ORBIT.NEW 120 0 4 0
END
```

Notice in **ORBIT.NEW** that we **ASK 1** to go **HOME**, set itself in the direction of the velocity, and move forward a distance equal to the speed. In effect this gives us a vector at the origin that determines the velocity. Later in **ADD.ACC** this velocity will be increased by an amount equal to the acceleration.

In **STEP**, turtle 0 (the orbiting body) points itself in the direction of the velocity and moves forward a distance equal to the speed. We then call **CHANGE.VEL** where turtle 0 sets its heading toward home. This is the direction of the acceleration and determines the acceleration angle. The magnitude of the acceleration is as before. Now turtle 1 is called on to add this acceleration to the old velocity in order to generate the new velocity. When this is completed turtle 0 proceeds to step again with the new velocity.

The advantage of this method is that we can see clearly the nature of the vector addition of velocities. (If you would like to see the velocity on the screen, you may wish to insert a scale factor for turtle 1.)

(It is interesting to note that the trajectory of turtle 1 is always a circle; see Figure 8.3. The planetary orbit is a circle in velocity space but an ellipse in configuration space. The center of the circle is not necessarily coincident with

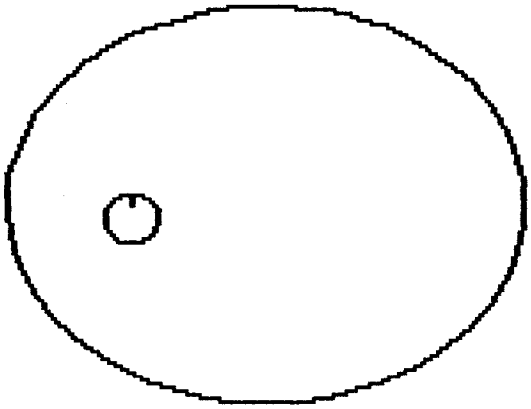


Figure 8.3 ORBIT.NEW 120 0 4 0

the force center. For further discussion of this subject see Harold Abelson, Andrea di Sessa, and Lee Rudolph, "Velocity, space and the geometry of planetary orbits," *American Journal of Physics* 43 (1975), p. 579.)

Problems

1. Try some orbits and see what you get. Choose different values of x, y, speed, and direction and see what happens. Try the given procedures of **CIRCULAR**, **ELLIPSE**, and **COMET** and see what happens.
2. Change the mass to 3500 and try **CIRCULAR** again. What is the shape of the new orbit?
3. We would like to determine how the velocity of the planet in a circular orbit is related to the **MASS** of the central body and the radius of the orbit. To do this, we first observe that a radius of 50 and a velocity of 8.9 give a circular orbit. Change the radius to 100 and see what new velocity is needed to produce a circular orbit. Can you use this result to select from the following choices the relationship between the velocity and the radius for circular orbits?
   a. v = constant * $r^{1/2}$
   b. v = constant / $r^{1/2}$
   c. v = constant * r
   d. v = constant / r
4. Next we want to see how the velocity of a circular orbit depends on the mass of the central body. To do this, change the **MASS** to 2000, set the radius at 100 and see

what velocity is necessary to achieve a circular orbit. Can you tell from this result which of the following relationships between the velocity and the mass of the central body is correct?

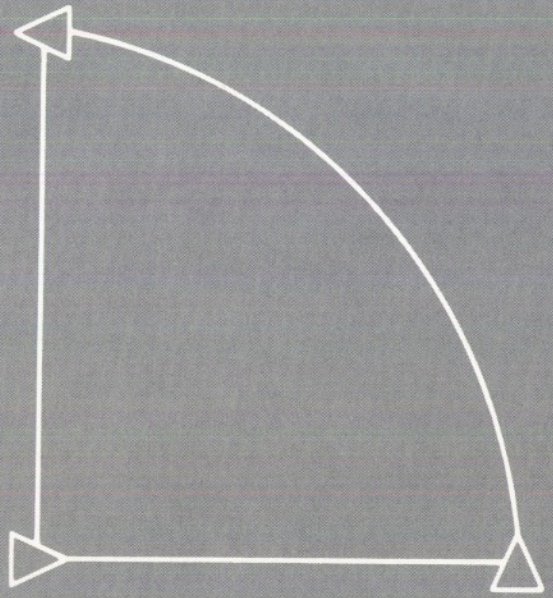
a. $v = \text{constant} * m^{1/2}$
b. $v = \text{constant} / m^{1/2}$
c. $v = \text{constant} * m$
d. $v = \text{constant} / m$

5. Next see if you can put Projects 3 and 4 together and find exactly how v depends on m and r together. Can you determine the constant as well? For example, is $v = 2\ m^{1/2}\ r^{1/2}$ a correct formula?

Projects

1. It is interesting to see what would happen if we lived in a universe in which the gravitational force were not proportional to $1/r^2$. Consider the shape of the orbits if the force is proportional to $1/r^3$. To do this, change **MAKE "R3 :R * :R * :R** to **MAKE "R3 :R * :R * :R * :R** in the **FIND.R** procedure. Also (to keep the turtle on the screen) **MAKE "MASS 4000 * 50** in **ORBIT**. Now try **CIRCULAR**. (These orbits are known as Cotes' spirals in honor of the man who first investigated this problem.)
2. This project is similar to Project 1. Let the force be proportional to $1/r^4$. (Change the mass to **4000 * 50 * 50**; that is, 1×10^7.) You will find only two types of orbits: Those that go right into the sun and those that go off to infinity. Be happy that you do not live in such a universe.

chapter



9 The Music of the Spheres

Introduction

In Chapter 8 we limited ourselves to imaginary planetary bodies. We were restricted in our choice of parameters by the necessity of keeping the turtle in view. We would like to consider more realistic problems now—in particular, the orbits of all nine planets in our solar system as well as the motion of the Earth's moon and artificial satellites. In order to solve our problem with the limitations of the screen size we will scale all distances that are to be drawn on the screen.

Scaling

Our objective is to represent the solar system on the screen. This cannot be done all at once. If we were to draw Pluto on the screen (orbital radius 5.9×10^{13} m) at a distance of 100 turtle units, the radius of Mercury (orbital radius 5.8×10^{10} m) would be approximately 1 turtle unit. Now we clearly could not see a circle so small. We will therefore consider the planets in two groups. The first will be Mercury, Venus, Earth, and Mars. The second will be

Jupiter, Saturn, Uranus, Neptune, and Pluto. Of the first group, Mars is farthest from the sun. For this reason we choose the scale in such a way that the orbit of Mars fills the screen. The largest circle we can draw on the screen has a radius of 119 turtle units. Thus our scale factor is 119 divided by the radius of the orbit of Mars (**MAKE "SCALE 119 / MARS**). To represent the second group we will change the scale factor to 119 divided by the orbital radius of Pluto.

We will also have to make an adjustment in our time interval DT or the orbit will take too long to develop. We will do this by adjusting the time DT so that the distance the turtle moves in each step is some reasonable screen distance. The distance the turtle moves in each step is **:SPEED * :DT * :SCALE**. If this distance is too small, the orbit will take too long. If the distance moved is too large, the orbit will not be very accurate since our approximation assumes that the velocity is fairly constant over the length of the step. As a reasonable compromise we set

$$(:\text{SPEED}) * (:\text{DT}) * (:\text{SCALE}) = 8$$

The step length is therefore 8 turtle units, not too large and not too small. If we solve this equation for :DT we find

$$:\text{DT} = \frac{8}{(:\text{SPEED}) * (:\text{SCALE})}$$

This should be a reasonable time interval between steps.

We have taken care of two of the problems in constructing a program to draw the planetary orbits. A scale has been chosen for both the distances and the time. There is one more question to deal with and then we will begin.

All of the planets in our solar system travel in orbits which are very nearly circular. Since these orbits are nearly circular we would like our program to generate only circular orbits. We have examined in earlier problems the relationship between the speed, mass of the central body (now the Sun), and the radius in order to obtain a circular orbit. The result of that work shows that this relationship is

$$v = (M/r)^{1/2}$$

In that problem we had set G = 1. Now we use the appropriate value of G in metric units; $G = 6.67 \times 10^{-11}$. This G must multiply the mass so that

$$v = (GM/r)^{1/2}$$

In **CIR.ORBIT** we will input the value of the planetary radius and require that the velocity satisfy this equation. This assures us that the orbits will be circular and we do not have to resort to the trial and error approach of Chapter 8. Our program **CIR.ORBIT** becomes

```
TO CIR.ORBIT :R
CS
MAKE "X :R MAKE "Y 0
MAKE "DIRECTION 0
MAKE "TIME 0
MAKE "RAD.OF.SUN 6.91998E8
MAKE "MASS.OF.SUN 2.E30
MAKE "G 6.67N11
MAKE "SCALE 119 / MARS
MAKE "SPEED SQRT :G * :MASS.OF.SUN / :R
MAKE "DT 8 / (:SPEED * :SCALE)
CIRCLE :RAD.OF.SUN * :SCALE
PU
SETX :R * :SCALE
FIND.R
MAKE "VX (:SPEED * SIN :DIRECTION) + :DT * ACCX / 2
MAKE "VY (:SPEED * COS :DIRECTION) + :DT * ACCY / 2
FULLSCREEN PD
STEP :VX :VY
SPLITSCREEN PRINT :TIME / (24 * 60 * 60)
END

TO STEP :VX :VY
IF KEYP [STOP]
MAKE "X :X + :VX * :DT
MAKE "Y :Y + :VY * :DT
SETPOS LIST (:X * :SCALE) (:Y * SCALE)
MAKE "TIME :TIME + :DT
FIND.R
STEP :VX + ACCX * :DT :VY + ACCY * :DT
END

TO FIND.R
MAKE "R SQRT (SQ :X) + SQ:Y
MAKE "R3 :R * :R * :R
END

TO SQ :X
OP :X * :X
END
```

```
TO ACCX
OP -:G * :MASS.OF.SUN * :Y / :R3
END

TO ACCY
OP -:G * :MASS.OF.SUN * :Y / :R3
END

TO CIRCLE :R
MAKE "PI 3.14159
HT PU HOME
FD :R RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :R / 12]
LT (90 - 15)
END

TO MERCURY
OP 5.79997E10
END

TO VENUS
OP 1.08E11
END

TO EARTH
OP 1.5E11
END

TO MARS
OP 2.3E11
END

TO JUPITER
OP 7.89997E11
END

TO SATURN
OP 1.43E12
END

TO URANUS
OP 2.97E12
END

TO NEPTUNE
OP 4.49999E12
END

TO PLUTO
```

```
OP 5.89997E12
END
```

As the planet steps through its orbit the time is incremented by **DT** after each step. One of the things that we want to examine is the period of rotation. The elapsed time (measured in Earth days) may be determined at any point in the orbit by pressing any key. This will stop the **STEP** procedure. Control will then return to the procedure that called **STEP**; that is, **CIR.ORBIT**. The next command is **PRINT :TIME / (24 * 60 * 60)**, which is just the time in days.

We have included nine procedures, **MERCURY** through **PLUTO**, that output the appropriate radius of the planet. If we call **CIR.ORBIT EARTH** we obtain the Earth's orbit. By pressing the space bar (or any key) when the orbit closes we should find the length of the Earth's year.

If you attempt to determine the period of any of the planets outside the orbit of Mars you will not be able to see the turtle—he will be off the screen. To deal with this second group of planets (Jupiter, Saturn, Uranus, Neptune, and Pluto) change the scale from

```
MAKE "SCALE 119 / MARS
```

to

```
MAKE "SCALE 119 / PLUTO
```

Drawing Orbits

Up to this point we have constructed the planetary orbits by solving Newton's laws of motion and universal gravitation. For circular motion we already know the shape of the orbit; it is a circle. We do not really need Newton's laws to draw a circle. The only thing we do need from Newton is the speed and we have that:

$$v = (GM/r)^{1/2}$$

We will write a program which draws the circular orbit of any planet. We choose to draw the orbit not because it is better than calculating it from the laws of motion; we do so because it is much faster and our next objective is to write a program that will put several planets on the screen at the

same time in order to see how the planets move relative to one another. With several planets on the screen at once the turtle gets very busy, and so it is a kindness to simplify his task.

In order to draw circular orbits we replace

```
STEP :VX :VY
```

in **CIR.ORBIT** with

```
MAKE "ANGLE 360 * 8 / (2 * :PI * :R * :SCALE)
DRAW.ORBIT
```

where **DRAW** is defined by the procedure

```
TO DRAW.ORBIT
IF KEYP [STOP]
FD 8
LT :ANGLE
MAKE "TIME :TIME + :DT
DRAW.ORBIT
END
```

Try this modification and see how much faster it is, but not before you understand why **ANGLE** is defined the way it is.

Four Planets

As mentioned earlier, it is not possible to put all nine planets on the screen at once because of scaling problems. We can, however, put up four at a time. To speed up the movement we draw the orbits as demonstrated in the previous section.

```
TO PLANETS :R0 :R1 :R2 :R3
WINDOW
HOME.ALL
CS FULLSCREEN ST
MAKE "PI 3.14159
MAKE "R ( SE :R0 :R1 :R2 :R3 )
MAKE "TURTLE.NO [0 1 2 3]
MAKE "MASS.OF.SUN 2.E30
MAKE "G 6.67N11
MAKE "SCALE 119 / MAX :R
MAKE "V []
MAKE "ANGLE []
SET.VELOCITIES :R
MAKE "DT 8 / ( ( MAX :V ) * :SCALE )
```

```
SET.PLACE :R :TURTLE.NO
SET.ANGLE :R :V
PD HT
DRAW.ORBITS :V :TURTLE.NO
END

TO SET.VELOCITIES :R
IF EMPTYP :R [STOP]
MAKE "V LPUT SQRT (:G * :MASS.SUN / FIRST :R) :V
SET.VELOCITIES BF :R
END

TO SET.PLACE :R :TURTLE.NO
IF EMPTYP :R [STOP]
ASK FIRST :TURTLE.NO [PU SETX :SCALE * FIRST :R PD]
SET.PLACE BF :R BF :TURTLE.NO
END

TO SET.ANGLE :R :V
IF EMPTYP :R [STOP]
MAKE "ANGLE LPUT (360 * :DT * FIRST :V) / (2 * :PI
* FIRST :R) :ANGLE
SET .ANGLE BF :R BF :V
END

TO DRAW.ORBITS :V :TURTLE.NO
TURN :TURTLE.NO :ANGLE
STEP :V :TURTLE.NO
DRAW.ORBITS :V :TURTLE.NO
END

TO TURN :TURTLE.NO :ANG
IF EMPTYP :TURTLE.NO [STOP]
ASK FIRST :TURTLE.NO [LEFT FIRST :ANG]
TURN BF :TURTLE.NO BF :ANG
END

TO STEP :V :TURTLE.NO
IF EMPTYP :TURTLE.NO [STOP]
ASK FIRST :TURTLE.NO [FD ( FIRST :V ) * :DT *
 :SCALE]
STEP BF :V BF :TURTLE.NO
END

TO MERCURY
OP 5.79996E10
END

TO VENUS
```

```
OP 1.08E11
END

TO EARTH
OP 1.5E11
END

TO MARS
OP 2.3E11
END

TO JUPITER
OP 7.79999E11
END

TO SATURN
OP 1.43E12
END

TO URANUS
OP 2.97E12
END

TO NEPTUNE
OP 4.49999E12
END

TO PLUTO
OP 5.89999E12
END

TO START
PLANETS MERCURY VENUS EARTH MARS
END
```

In **PLANETS** there are four inputs, the radii of any four planets. There are also four lists used in **PLANETS** to keep track of the necessary information. First there is a list **TURTLE.NO ([0 1 2 3])** that simply keeps track of which turtle we are talking to. Second, there is a list **R** in which we store the values of the four radii of the orbits. Third, there is a list **V** containing the four velocities of the planets. Finally, there is a list **ANGLE** that stores the angles through which the four planets should turn with each step.

Notice that **PLANETS** is self-scaling. The **SCALE** is set at **119 / MAX :R** where **MAX :R** is the maximum of the four radii. Later when the planets are **SET** in **PLACE** with the command **SETX :SCALE * FIRST :R**, we can

be sure that for any **FIRST :R** the value of the x-coordinate will be 119 or less. We may call **PLANETS 1 2 3 4** or **PLANETS 1E10 2E10 3E10 4E10** and the scale factor will make the appropriate adjustment. (It is not necessary to order the radii. The only precaution that you must take is not to choose radii which differ by too great a factor. If one radius is 1/1000 of the largest radius then its screen radius will be approximately .1 turtle units, which is too small to see. You cannot put Pluto and the Earth on the screen at the same time.)

Problems

1. Determine the period of rotation for Mercury. To do this, run **CIR.ORBIT MERCURY** and press the space bar after the orbit is completed. The elapsed time will be printed. Do the same for Venus, Earth, and Mars.
2. Change the scale so that Pluto, the farthest planet from the Sun, is at 119 turtle units. Determine the period of rotation for Jupiter, Saturn, Uranus, Neptune, and Pluto. As a hint, the Pluto year is about 248 Earth years.
3. Determine the period of rotation of the Moon about the Earth. The mass of the Earth is 6.0×10^{24} kg and the distance between the Moon and the Earth is 3.8×10^{6} m. You will need to make some modifications of **CIR.ORBIT** to determine the answer to this question.
4. A communication satellite stays over the same spot on the Earth at all times. It appears to be stationary, which makes it more useful for purposes of communication. If it appeared to rotate about the Earth it would be available for use, at best, only 50% of the time. Of course it is not actually sitting still. It is in fact rotating about the Earth with a period of 24 hours. Such an orbit is called "geosynchronous" (synchronous with the Earth). Assuming that the orbit is circular, what is the radius of the orbit? (Hint: Use the information given in Problem 3. The result should be about 6 to 7 times the radius of the Earth.)
5. Call **PLANETS MERCURY VENUS EARTH MARS** and see if you can explain why Venus is seen from Earth only in the early morning or late evening hours. Mars, on the other hand, may be visible at any time of the night.
6. Call **PLANETS MERCURY VENUS EARTH MARS**.

Next, compare with **PLANETS JUPITER SATURN URANUS PLUTO**. This should give you some idea of the size of the solar system.

Projects

1. In Appendix A you will find a table giving the planetary positions on January 1, 1985. Construct a program which places the Earth, Mars, Jupiter, and Saturn in their proper positions at the first of the year. As you run the program the planets should rotate about the center of the screen and the elapsed time should be printed at the bottom. Let the program run until the current date is reached. Check the night sky and see if the positions of the planets agree with your calculations.
2. It was noted as long ago as the third century B.C. by Greek astronomers that at certain times of the year Mars, Jupiter, and Saturn trace out retrograde orbits. A retrograde orbit is one in which the planet, viewed from the Earth, reverses its apparent motion relative to the fixed stars (see Figure 9.1). Can you construct a program which illustrates the reason for the retrograde motion? The program would draw a line connecting the Earth to the planet of your choice. Would you see retrograde motion of Mercury and Venus? How many times, during one Martian year, will Mars exhibit retrograde motion?

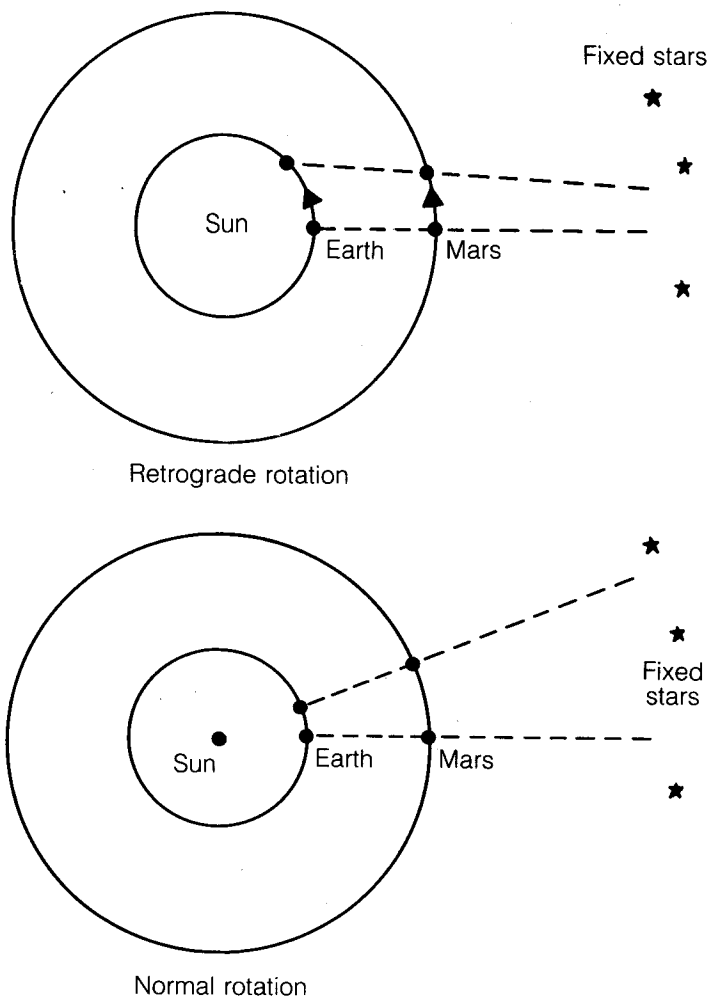
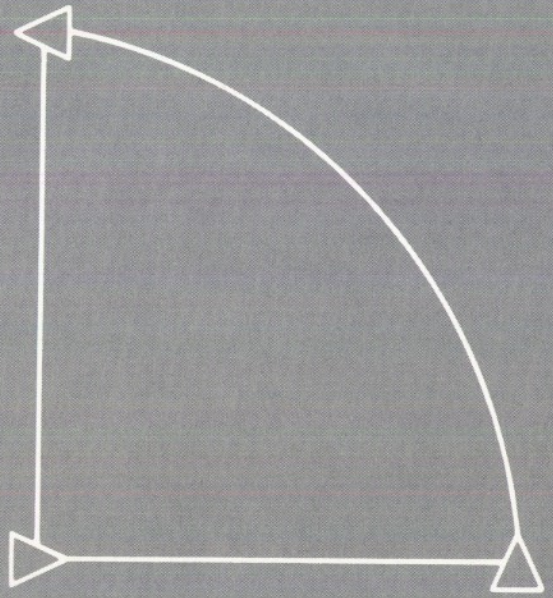


Figure 9.1 Normal Rotation and Retrograde Rotation

chapter



10 Voyager II and Lunar Orbits

Introduction

Sending a space ship to Uranus requires a very big rocket and takes a long time. Voyager II is scheduled to reach Uranus on January 14, 1986 and will have spent eight years in space to cover the eight billion miles. Clearly, any trick which would speed up the flight would be welcome. One method we would like to explore is bouncing Voyager off the other planets; in essence using the planets as large paddles to accelerate Voyager. Now you clearly cannot bounce a space ship off a planet. What you can do, however, is fly the space craft very close to the planet and use the gravitational field to pull the space craft around the planet. The scheme is illustrated in Figure 10.1. On the left, a space ship is accelerated by a planet and on the right a ball is accelerated by a paddle. The mechanism by which the two bodies interact (the space ship and the planet or the paddle and the ball) is not important. What is important is that in an energy conserving collision one body may gain energy and the other lose a comparable amount, thereby conserving the total. As a general rule we can say that when a light object

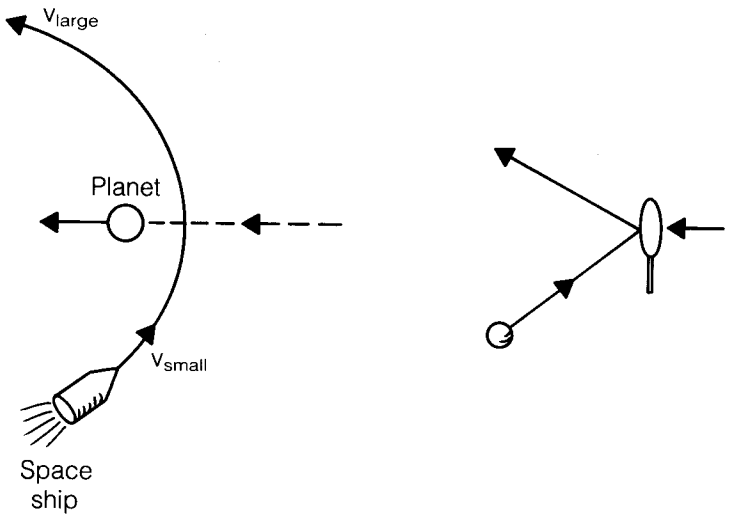


Figure 10.1 Space Ship Bouncing off Planet like Ball off Paddle

is "struck" by an advancing heavy object, the light object will gain energy. On the other hand, when a light object strikes a retreating heavy object, the light object will lose energy. This is illustrated in Figure 10.2. Conversely, if the truck were approaching the ball, the ball would gain energy after impact, as illustrated in Figure 10.3.

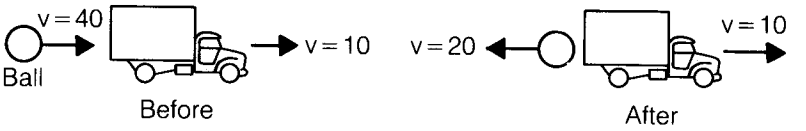


Figure 10.2 A Ball Bounces Off a Receding Truck

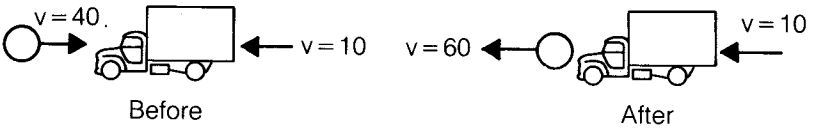


Figure 10.3 A Ball Bounces Off an Advancing Truck

The general rule in such head-on collisions is that if energy and momentum are conserved, the relative velocity of approach is equal to the relative velocity of recession. In Figure 10.2 this relative velocity is 30 mph before and after the collision. In Figure 10.3 the relative velocity is 50 mph before and after the collision.

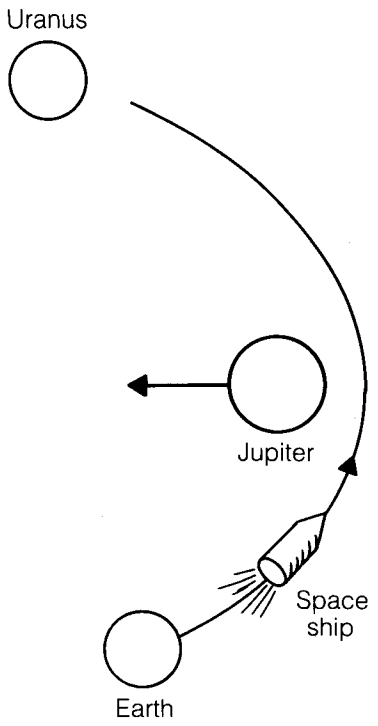


Figure 10.4 The Space Ship Gets a Boost From Jupiter

Now we do not want to bounce our space ship off the planet the way the ball bounces off the truck. But we might get a boost on our trip to Uranus by employing a tactic such as that illustrated in Figure 10.4. The space ship will increase its velocity by "colliding" with the advancing planet, Jupiter. If the launch of the space ship is timed just right we may boost it on its way toward Uranus by using Jupiter as a great paddle.

The Turtle Soars

This effect is not unlike the soaring of birds rising in a thermal uplift. The strategy is to find something going your way and take a free ride. We may observe the effect with the program **VOYAGER**:

```
TO VOYAGER :X0 :Y0 :V0 :ANG0 :X1 :V1
WINDOW HOME.ALL
MAKE "MASS 8000
```

```
CS PU
SETPOS LIST :X0 :Y0
ASK 1 [PU SETPOS LIST :X1 0 PD]
FIND.R
MAKE "VX (:V0 * SIN :ANG0) + ACCX / 2
MAKE "VY (:V0 * COS :ANG0) + ACCY / 2
MAKE "DT .1
FULLSCREEN PD HT
STEP :VX :VY :X1
END

TO STEP :VX :VY :X1
PD
SETPOS LIST :X0 :Y0
ASK 1 [SETPOS LIST :X1 0]
MAKE "X0 :X0 + :VX * :DT
MAKE "Y0 :Y0 + :VY * :DT
FIND.R
(PRINT [SPEED =] SQRT ((SQ :VX) + SQ :VY))
STEP :VX + ACCX * :DT :VY + ACCY * :DT :X1 + :V1 *
 :DT
END

TO FIND.R
MAKE "R SQRT ((SQ (:X0 - :X1)) + SQ :Y0)
MAKE "R3 :R * :R * :R
END

TO ACCX
OP -:MASS * (:X0 - :X1) / :R3
END

TO ACCY
OP -:MASS * :Y0 / :R3
END

TO SQ :X
OP :X * :X
END

TO START
VOYAGER -120 -120 10 60 120 -8
END
```

VOYAGER takes six inputs. The first four relate to the space ship and specify its position and velocity. The last two specify the x-coordinate and velocity of the planet. The planet is given a mass of 8000. (You may wish to change this.) The rest of the program is similar to that for planetary orbits.

The main difference is that we have two moving, interacting bodies.

Run **START** to see how the space ship can pick up energy from an advancing planet. Use the split screen to observe the increase in velocity. In this example, the space ship begins with a velocity of 8 and after bending around the planet it leaves the screen with a velocity of about 18. Experiment with other initial parameters to see if you can further increase the velocity. (Remember, if you get too close to the planet, the numerical approximations break down and your results may be due more to a numerical error.) In the Problems section we will prove that the space ship will lose velocity when it bounces off a receding planet.

The Lunar Orbit

Suppose you were to view the solar system from a great distance in space. You see the planets revolving about the sun. You also see the planetary moons rotating about their parent planets. From the point of view of someone on the Earth, the Moon rotates in a circle about the Earth. But what does the orbit look like to the observer in space? From space we see the Moon rotating about a planet which is itself rotating about the sun. Might the orbit look something like that in Figure 10.5? Surprisingly, the path looks quite different. It will more closely resemble that shown in Figure 10.6.

We would like to show how these figures can be generated as a special application of the **VOYAGER** program. In that program there was one massive body (a planet) moving with constant velocity in a straight line and one light body (the space ship) whose motion was affected by its gravitational interaction with the planet.

We now imagine the Moon as a space ship orbiting about the Earth. The Moon does not greatly affect the motion of the Earth since the Earth is so massive, but the Earth

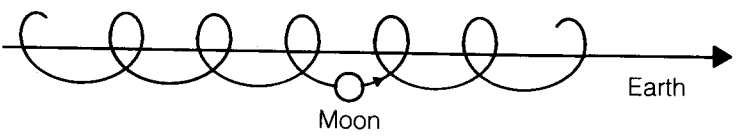


Figure 10.5 Possible Moon Orbit About the earth

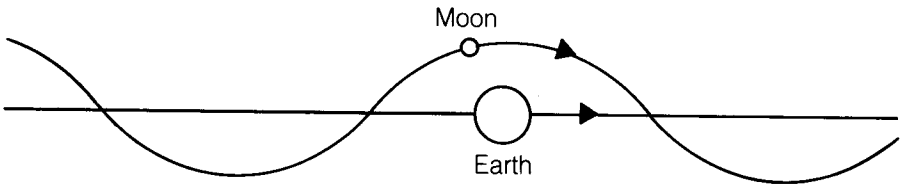


Figure 10.6 Actual Moon Orbit About the Earth

very definitely affects the motion of the Moon. Because of the pull of the Earth on the Moon, the Moon rotates about the Earth once every 27 days.

To see the shape of the Moon's orbit we make the following modification to the **VOYAGER** program. We change the **MASS** in **ORBIT** from **8000** to **750**. We also add the following procedure:

```
TO MOON.ORBIT :V.EARTH
MAKE "V.MOON 5
MAKE "Y.MOON 750 / ( :V.MOON * :V.MOON )
VOYAGER -120 :Y.MOON (:V.MOON + :V.EARTH) 90 -120
 :V.EARTH
END
```

In **MOON.ORBIT** there is one input: **V.EARTH**, which is the velocity of the Earth. The velocity of the Moon relative to the Earth is chosen to be 5. The distance between the Moon and the Earth (**Y.MOON**) is set in such a way that the orbit would be a circular orbit if the Earth were at rest. (Remember that we showed in Chapter 8 that the orbit is a circle when $R = M/V^2$.) When **ORBIT** is called, the Moon is set at **XCOR = −150** and **YCOR = Y.MOON**. The velocity of the Moon is set at **V.MOON + V.EARTH** which is the net velocity of the Moon. The heading of the Moon is 90. The Earth is set at **XCOR = −150, YCOR = 0**, with a velocity of **V.EARTH**. The initial condition is illustrated in Figure 10.7. The velocity of the Moon relative to the Earth is fixed (**V.MOON = 5**) as is the distance between the two. The only physical variable is the velocity of the Earth. If the Earth moves slowly, we expect the looping trajectory. If the Earth moves quickly, we expect the wave-like pattern. Try running **MOON.ORBIT 3** and **MOON.ORBIT 7**. They should generate first a looping trajectory and second a wave-like trajectory. In the first case the Earth is moving slowly and in the second the Earth is moving rapidly. To understand which of these two cases more closely represents the

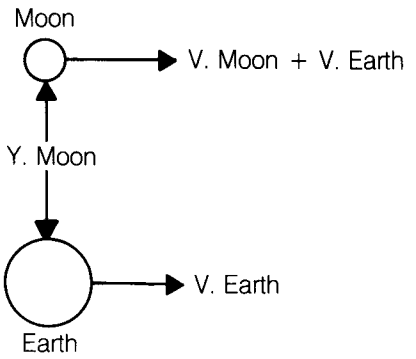


Figure 10.7 Earth and Moon Velocities

trajectory of the Earth's moon you have only to observe that the velocity of the Earth is 30 times the velocity of the Moon relative to the Earth. The trajectory therefore will be wave-like with a very long wave length.

As you allow these programs to run you should notice two things. First, the distance between the Moon and Earth remains constant and second, if you hold a toothpick parallel to the line formed by joining the Earth and the Moon, you should find that the toothpick rotates with steady angular velocity as it should.

Problems

1. We have shown that a space ship can pick up speed by colliding with an advancing planet. Demonstrate the converse. Show that a space ship will lose speed by colliding with a receding planet.
2. Try the **MOON.ORBIT** program with **V.EARTH = 5**. Recall that the velocity of the Moon relative to the Earth has been set to 5. The orbit trajectory has cusps and will look like Figure 10.8. This figure is called a cycloid.

   See if you can understand this result on the basis of the vector diagram shown in Figure 10.9. The net velocity of the Moon (**V**(m)) is the vector sum of the velocity of the Moon relative to the Earth (**V**(m/e)) and the velocity of

Figure 10.8 A Cycloid

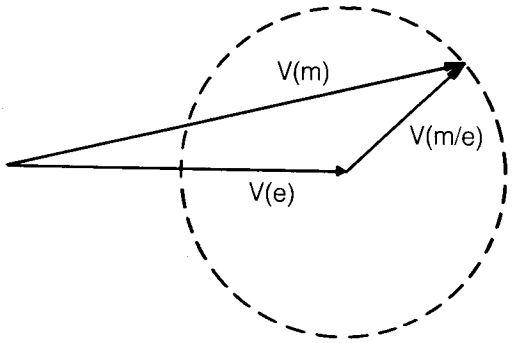


Figure 10.9 Vector Diagram of the Velocities

the Earth (**V**(e)). The vector **V**(e) remains fixed while the vector **V**(m/e) rotates with constant angular velocity. What happens to the vector **V**(m)? Can you also interpret the looping trajectories when **V**(m/e) is greater than **V**(e) and the wave-like trajectories when **V**(m/e) is less than **V**(e)?

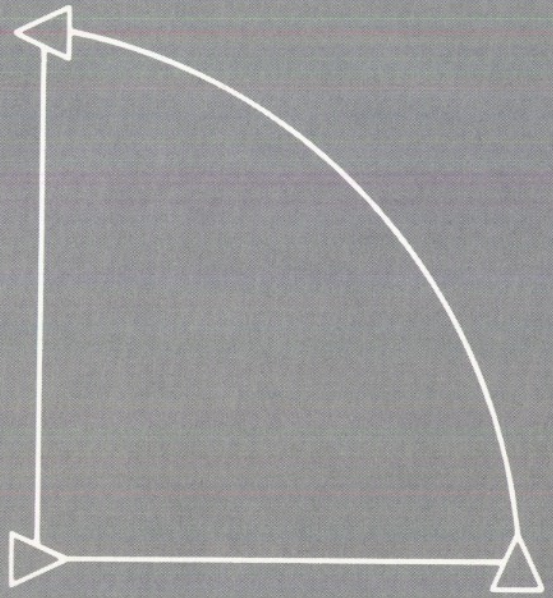
Projects

1. Construct a program which simulates the ball bouncing off the moving truck. The force should be zero until the ball is within some specified distance from the truck and also sufficiently strong to prevent the ball from striking the truck. You might use a repulsive inverse square force or, more realistically, a repulsive linear force. Demonstrate that for any force (any force independent of the velocity), the relative velocity of approach (the velocity of the ball before it begins to feel the force) is equal to the relative velocity of recession (the velocity after the ball has left the force field).
2. As an advanced project, write a program which draws the trajectories of two interacting planetary bodies. We have considered a special case of two interacting bodies: Voyager II and Jupiter. But Voyager is so small in comparison to Jupiter that we can surely neglect the effect of Voyager on the orbit of Jupiter. But suppose we were interested in the paths of a binary star. (A binary star is in fact two stars moving about one another.) Suppose one star were one half the mass of the other. What would their orbits look like?

   If you choose arbitrary values for the input velocities and headings of the two bodies, they will more than likely move off the screen. To avoid this, choose initial condi-

tions in such a way that the total momentum is zero. (This will fix the position of the center of mass of the system. The bodies may leave the screen, but they will return.) You pick one momentum (or velocity) and let the computer set the other in such a way that the total momentum is zero.

chapter



11 Jets, Rockets, and Conservation of Momentum

Introduction

If you have ever seen anyone fire a shotgun you must have noticed the "kickback" from the gun. That is the jolt you get in firing a large caliber gun. A cannon has an even larger kickback. If you could put a cannon to your shoulder and fire it, you might inflict as much damage on yourself as you do your target. This action (projecting a shell) and reaction (the kickback of the gun) are examples of one of the most fundamental principles of physics, the principle of conservation of momentum. We will see that the momentum given to the shell is lost by the gun so that the overall momentum is conserved; that is, remains constant.

To give these ideas greater precision we must define momentum. Momentum is a vector. As such, it has both magnitude and direction. A body of mass m and velocity **v** has a momentum **p** given by

$$\mathbf{p} = m\mathbf{v}$$

A Mack truck and a VW Bug traveling at the same speed do not have the same momentum. The truck has the larger

mass and therefore the larger momentum. But a VW, by traveling at high speed may have a greater momentum than a slowly moving Mack Truck.

The principle of momentum conservation says that:

In any isolated system, the total momentum remains constant in time.

An isolated system is one in which we may, for all practical purposes, ignore outside forces. If there are outside forces present, then our conservation of momentum principle is replaced by Newton's second law of motion:

The sum of all the outside forces is equal to the rate of change in the total momentum of the system.

Sometimes we may have a limited conservation of momentum. If there are no forces acting in a given direction then the component of the total momentum *in that direction* is conserved. For example, if two cars are traveling in opposite directions on a city street and collide, we may say that since there are no significant forces acting in the direction of travel (assuming the cars are coasting), the momentum in the direction of travel must be conserved in the collision.

Rockets and Jet Planes

As we have said, the recoil of a gun is an example of momentum conservation. Before the bullet is fired the momenta of the bullet and gun are zero. Imagine the gun suspended by a pair of cords as shown in Figure 11.1. There

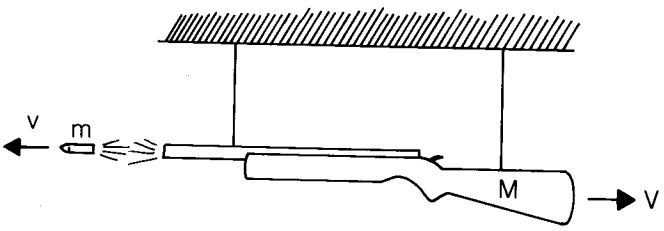


Figure 11.1 A Gun Fires a Bullet

is no horizontal force acting on the gun. Therefore the horizontal component of momentum must be conserved. If the horizontal momentum is zero before the bullet is fired the total horizontal momentum must be zero after the bullet is fired. We may write an equation

$$\mathbf{p}\ (\text{total, before}) = \mathbf{p}\ (\text{total, after})$$

or since $\mathbf{p}$ (total, before) is zero

$$0 = \mathbf{p}\ (\text{total, after})$$

Now the total momentum after firing the gun is the momentum of the bullet plus the momentum of the gun. Therefore

$$0 = \mathbf{p}\ (\text{gun, after}) + \mathbf{p}\ (\text{bullet, after})$$

Each momentum is the product of its mass and velocity. If the mass and velocity of the gun are M and $\mathbf{V}$, and the mass and velocity of the bullet are m and $\mathbf{v}$ we have

$$0 = M\mathbf{V} + m\mathbf{v}$$

If we chose the positive horizontal axis to be directed to the right then the components of this vector equation on the horizontal direction is

$$0 = MV - mv$$

since the velocity of the bullet has a negative horizontal component. We may calculate the velocity of recoil of the gun. It is

$$V = mv/M$$

and is therefore a small fraction of the velocity of the bullet (since m/M is a small number).

Now this "propulsion" of the gun by firing the bullet is very similar to the mechanism for the propulsion of rockets and jets. Both rockets and jets propel something out the back end which, to conserve momentum, increases the velocity of the jet or rocket. However, there are some important differences between the rocket, the jet plane, and the gun. First, the rocket and jet plane are in a continuous state of motion. The gun is not. There is also a difference between the jet and the rocket. The rocket carries all of its own propellant. The jet does not. A jet takes in air, mixes it with fuel, ignites the mixture, and expels the combustion products out the rear. The bulk of the mass ejected from the jet

is the air; very little of it is fuel. The rocket mixes, ignites, and ejects these propellants from the rear in much the same way as the jet. But the important distinction is the fact that the jet picks up the bulk of its propellant on the fly, while the rocket carries its own. The rocket is the engine of choice for space travel. There is nothing on the way to the moon to scoop up. There is no air in space. The rocket must bring whatever it needs.

To see how this difference between the rocket and the jet affects the motion we will study the two in some detail. In Figure 11.2 a rocket of mass M is moving with a velocity V. It emits a propellant of mass m, traveling at a velocity v *relative to the rocket*. The new velocity of the rocket is V′. The velocity of the propellant will be the velocity of the rocket (V′) minus the relative velocity of the propellant (v). Since momentum is conserved, the initial momentum must be equal to the final momentum. Therefore

$$MV = (M - m)V' + m(V' - v)$$

Solving for V′ we find

$$V' = V + mv/M$$

Therefore the velocity increases in proportion to the mass and velocity of the propellant and inversely with the mass of the rocket.

Next, let us compare this with the jet. In Figure 11.3 a jet of mass M is traveling with a velocity V. It picks up a mass of air m, initially at rest, and propells it out the rear with a relative velocity v. Once again, momentum is conserved and so

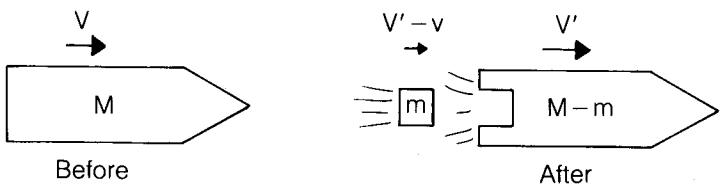


Figure 11.2 A Rocket Ejects Propellant

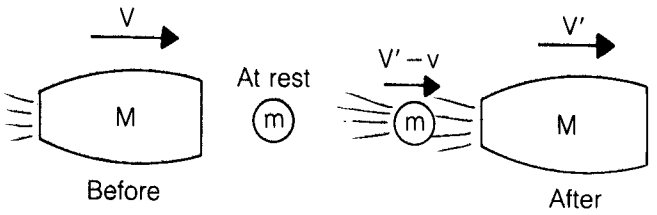


Figure 11.3 A Jet Engine

$$MV = MV' + m(V' - v)$$

or

$$V' = V + m(v - V')/M$$

We see that the increase in velocity for the jet is not the same as the increase in velocity for the rocket. We see in fact that unless $v - V'$ is positive, the jet will slow down (V' will be less than V). Therefore the jet must be throwing air out the back at a speed greater than the speed of the jet itself; otherwise the jet is scooping up air in front and ejecting it from the rear so that it is traveling in the same direction as the plane. This is not unlike the drag that any body experiences in moving through air: The object causes air around it to acquire a momentum at the expense of the momentum of the body. The trick is to get the air to travel in a direction *opposite* to the direction of the jet. Perhaps we can understand this best by considering the limiting case in which $v = V'$. Here the jet picks up the air at rest and leaves it at rest ($v - V' = 0$), and of course the plane neither speeds up nor slows down ($V' = V$).

An example very similar to this is that of the row boat. The function of the oars is to take water which is at rest along side the boat and give it a velocity directed away from the back of the boat. Initially the boat and the water are at rest. After the first stroke, the water is moving backward and so, if momentum is to be conserved, the boat must move forward. There is, however, an upper limit on the boat speed. It is possible to move the oars just so fast. Eventually there will come a boat speed which is just equal to the oar speed. When this happens, the velocity of the oars relative

to the water is zero. The oars no longer increase the velocity of the water relative to the boat. When the oars are dipped into the water, the oars and the water are moving at the same relative speed. We might as well not put the oars in the water at all. Thus the upper limit of velocity of the boat is equal to the maximum oar speed. In exactly the same way, the upper limit of the velocity of the jet is the velocity of the propellant relative to the jet.

Turtle-Rockets and Turtle-Jets 

Let us write a program to illustrate the differences between rockets and jets. The program **ROCKET** demonstrates the motion of a rocket.

```
TO ROCKET :VR :M :MM
WRAP
SETPOS [0 0]
HOME RIGHT 90
STEP 0
END

TO STEP :V
FORWARD :V
STEP :V + :M * :VR / :MM
END

TO START
ROCKET 10 5 1000
END
```

We see from the listing that the turtle is placed at the origin facing right and in the **STEP** procedure, begins to move forward with ever increasing velocity. The velocity increases in proportion to the ejected mass (**:M**), the relative velocity of the propellant (**:VR**), and in inverse proportion to the mass of the plane (**:MM**).

The program for the motion of a **JET** is very similar.

```
TO JET :VR :M :MM
WRAP
HOME RIGHT 90
STEP 0
END

TO STEP :V
FORWARD :V
```

```
STEP :V + :M * (:VR - :V) / :MM
END

TO START
JET 10 5 1000
END
```

If you wish to see precisely how the velocity increases you might insert in the **STEP** procedure the line: **(PRINT "VELOCITY :V)** and the velocity will be printed. (Be sure to include the parentheses.)

By using more than one turtle we may let the jet and the rocket move side by side. It is also instructive to plot the velocity as a function of the time. In **ROCKET.AND.JET** there are four turtles. The first two turtles represent the rocket and the jet. The second two plot the velocity of the rocket and the jet as functions of the time. You will notice that the velocity of the rocket increases linearly with the time while the velocity of the jet approaches a constant value. This limiting value is the relative velocity of the exhaust.

```
TO ROCKET.AND.JET :VR :M :MM
WRAP PU HT
HOME.ALL
ASK 0 [SETPOS [0 8] RT 90]
ASK 1 [SETPOS [0 -8] RT 90]
PD
MAKE "TIME 0 MAKE "DT 1
MAKE "SCALE 10
STEP 0 0
END

TO STEP :V0 :V1
ASK 0 [FORWARD :V0]
ASK 1 [FORWARD :V1]
ASK 2 [SETPOS LIST ( :TIME - 120 ) ( :V0 * :SCALE
 )]
ASK 3 [SETPOS LIST ( :TIME - 120 ) ( :V1 * :SCALE
 )]
MAKE "TIME :TIME + :DT
(PRINT "ROCKET :V0 " "JET :V1)
STEP (:V0 + :M * :VR /:MM) (:V1 + :M * (:VR - :V1)
 / :MM)
END

TO START
ROCKET.AND.JET 5 5 300
END
```

Problems

1. Vary some of the parameters in **ROCKET** to see what effect they have on the motion.
   a. Compare **ROCKET 10 5 1000** with **ROCKET 10 5 500**. Can you explain why a reduction of the rocket mass has the effect that it does?
   b. Compare **ROCKET 10 5 1000** with **ROCKET 10 10 1000**.
   c. Compare **ROCKET 10 5 1000** with **ROCKET 20 5 1000**.

2. Do Problem 1 for the **JET**.

3. If a rocket ship is operating in outer space, there is no air resistance. Within the Earth's atmosphere, however, the air resistance exerts a force on the rocket. This resistance is very nearly proportional to the velocity.
   a. Include a frictional force of $-.02 * V$ in the equation of motion of the rocket and see what happens.
   b. How does the motion of the rocket with air resistance compare with the jet without air resistance?
   c. Suppose you added some air resistance to the jet. How does this affect the maximum velocity of the jet?

4. Give the jet an initial velocity which is greater than the relative velocity of the exhaust. Describe the motion.

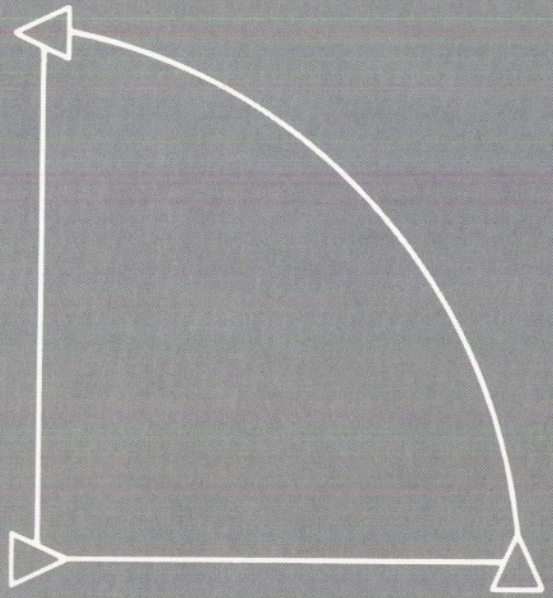
Projects

1. A 1-ton open-top freight car is moving along a level track with negligible friction at a speed of 40 mi/hr. It begins to rain and the car accumulates rain at a rate of 1 ton/hr.
   a. If the rain is falling vertically, can you write a Logo program to determine the motion of the railroad car?
   b. How long does it take for the speed to be reduced to 20 m/hr?

2. A railroad car weighing 1000 kg contains 100 kg of water. The car is at rest on a level frictionless track. Water is leaking from the back at the rate of 1 kg/sec. The velocity of the water relative to the car is 5 m/sec and is streaming away from the car parallel to the tracks. Determine the velocity of the car at the moment that all of the water has run out. To do this write a Logo program which describes the motion of the car. This program should stop when the water is all gone and then print the velocity of the car at that time. (Answer: 3.46 m/sec)

3. Can you improve on the **ROCKET** program to include the effect of the decreasing mass of the rocket as it exhausts its fuel? The mass **:M** decreases by an amount **:MM** every second.

chapter



12 The Harmonic Oscillator, Clocks, Rabbits, and Foxes

Introduction

Every form of clock relies on the repetitious motion of something: The rotation of the Earth, the oscillation of a spring or a tuning fork or a crystal, the swing of a pendulum, or in the case of the old water clocks, the drip of water from a bucket. All of these systems have one thing in common: They are repetitious. They repeat themselves over and over again. Such systems are said to be *periodic*. The period is denoted by the letter T and is defined as the time between the repetitions.

There is one periodic system of special interest: the harmonic oscillator. We will illustrate the harmonic oscillator with a few examples. In Figure 12.1 you see a block attached to a spring, a simple pendulum, and a ball rolling in a bowl. All of these systems are periodic (neglecting friction) and for small amplitudes exhibit the following common feature:

$$a \propto -(\text{positive constant})\, x$$

where a is the acceleration and x is the displacement from

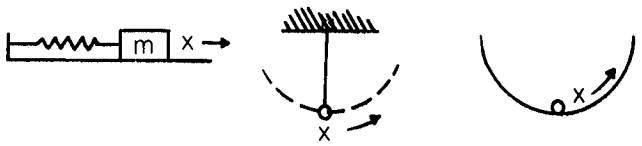


Figure 12.1 Three Oscillatory Systems

the equilibrium position. If we call the positive constant w^2 we have

$$a = -w^2 x$$

any physical system satisfying this equation is said to be a harmonic system. The value of w will be different for different systems.

It is easy to see that any system that satisfies the above equation will be periodic. It says that the velocity will decrease as long as x is positive and increase as long as x is negative. Clearly therefore, if the velocity continuously decreases for positive x, it must become negative eventually. If the velocity becomes negative (and continues to become more negative) the object must pass back through the origin. When x becomes negative, the velocity constantly increases and must eventually become positive and thus pass back through the origin again. This process is repeated over and over again so that the object oscillates about the origin.

So what is so special about the *harmonic* oscillator? In Figure 12.2 we illustrate a *general* relationship between the acceleration and the displacement which satisfy the conditions discussed above; that is, the acceleration is positive when the displacement is negative and vice-versa. Notice that for small values of x the dotted line is a good approximation of the solid curve. But since the dotted line is a straight line it follows that on this line

$$a \propto -x$$

or

$$a = -w^2 x$$

We can see therefore, that for small displacements (small values of x) we may always approximate *periodic* motion (represented by a curve for which a is negative when x is positive and vice-versa) by *harmonic* motion (represented by a linear relationship between a and x). This is one reason

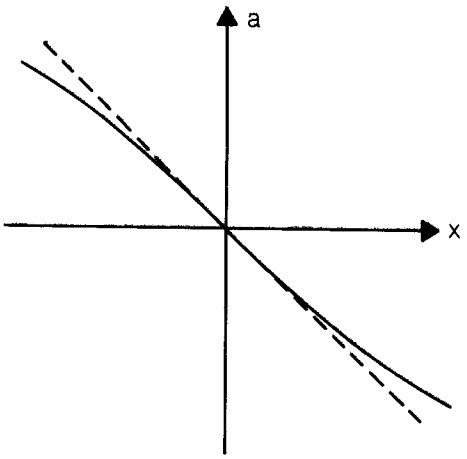


Figure 12.2 Relationship Between Acceleration and Displacement

why harmonic motion is special. The other is the wide variety of systems that satisfy this equation: The flow of electrical charges back and forth in your television circuits, the vibration of the electrons on the phosphors on the surface of the screen which forms the television picture, the rocking chair of the old gentleman watching the television picture, the interacting chemical components in the body of the old gentleman, and so on.

An Oscillating Turtle

Let us write a Logo program to solve the harmonic oscillator equation. We may rewrite the fundamental equation of the harmonic oscillator as follows:

$$\text{The change in velocity per second} = -w^2 x$$

If we choose our time interval to be one second, the motion defined by this equation is approximated by the Logo program **OSC**.

```
TO OSC :AMP
MAKE "W .5
PU SETPOS LIST 0 :AMP
PD
STEP 0
END
```

```
TO STEP :VEL
FORWARD :VEL
STEP :VEL - :W * :W * YCOR
END

TO START
OSC 50
END
```

In **OSC** we have chosen to move along the y-axis rather than the x-axis. The value you select for **:AMP** (the amplitude) determines the maximum displacement from the equilibrium position. The value of w has been chosen to be 0.5. As you run the program you will see that the acceleration is positive when **YCOR** is negative (the velocity decreases continuously for all positive values of **YCOR**). Conversely, the velocity increases for all negative values of **YCOR**. These requirements are demanded by the basic equation ($a = -w^2 x$).

If the value of w is increased, we find that the turtle oscillates faster. The frequency therefore increases as w increases. Try making w = 0.1 and see what happens.

To get a better "picture" of the harmonic motion we will ask the turtle to leave a record of his trip by making a y versus t plot. This plot is accomplished with the **OSC.TIME** program:

```
TO OSC.TIME :AMP
MAKE "W .1
MAKE "DT 1
PU SETPOS LIST 0 :AMP
HT PD
STEP 0
END

TO STEP :VEL
FORWARD :VEL
RIGHT 90
FORWARD :DT
LEFT 90
STEP :VEL - :W * :W * YCOR
END

TO START
OSC.TIME 50
END
```

If you run **START** the turtle will produce the curve shown in Figure 12.3. We will leave it as an exercise to verify that the period T of the wave is related to w by the equation $T = 2\pi/w$. You may recognize this curve as the cosine function.

We would like to demonstrate that

$$y = (\text{constant}) * \cos(\text{constant} * t)$$

It can be shown that the constants are related to the amplitude and period as follows:

$$y = \textbf{AMP} * \cos(180\ w\ t\ /\ 3.14)$$

or equivalently:

$$y = \textbf{AMP} * \cos(360\ t/T)$$

To verify this equation we add the following command to the **STEP** procedure:

```
IF XCOR > 110 [PU SETPOS LIST 0 :AMP PD VERIFY 0]
```

This line follows immediately after **STEP :VEL**. We also define **VERIFY** as follows:

```
TO VERIFY :TIME
SETPOS LIST :TIME :AMP * COS 180 * :W* :TIME / 3.14
VERIFY :TIME + :DT
END
```

In **VERIFY** the turtle should retrace the earlier curve. He will if $y = AMP \cos(180\ w\ t\ /\ 3.14)$.

Rabbits and Foxes

Periodic motion is not limited to pendula and tuning forks. Many systems in nature exhibit this phenomenon. As

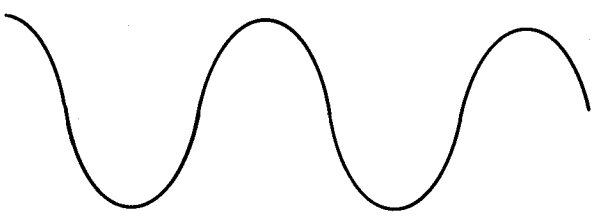


Figure 12.3 Displacement as a Function of Time

one interesting example, there is an area in biology called "preditor-prey theory." It deals with the ecological balance of a natural system. As a special example of this theory let us imagine an island on which there are only two animals—rabbits and foxes. The rabbits eat grass and the foxes eat rabbits. In order to study the population dynamics of this system we ask the following question: If there are currently 80 rabbits and 40 foxes on the island, how many rabbits and foxes will there be next year, the year after that, and so on?

Of course we need some more information to solve this problem. Let us use R to represent the number of rabbits and F the number of foxes. We assume that the growth rates for the rabbits and foxes satisfy the following equations (called the Volterra equations):

Rate of increase of R = a R − c R F
Rate of increase of F = − b F + d R F

The first term on the right hand side of each equation represents the natural growth rate of each species in the absence of the other. If there were no foxes, the rabbits would grow at a rate of a R (assuming unlimited grass). The coefficient a is the growth rate per rabbit per year of the rabbit populations. If the growth rate per rabbit is multiplied by the number of rabbits (R) we get the growth rate of the entire population. In a similar way, if there were no rabbits to feed on, the foxes die out at a rate −b F.

The second term in each equation represents the number of encounters between rabbits and foxes. These encounters (not pleasant to behold) will contribute to the reduction of the rabbit population and the nourishment of the fox population. Now the number of encounters between rabbits and foxes will be proportional to the product RF. For example, if you double the number of rabbits *or* the number of foxes you will double the number of encounters.

The Volterra equations can be solved using the **PRED** Logo program for the indicated values of a, b, c, and d:

```
TO PRED :RABBITS :FOXES
WINDOW
CS PU HT
MAKE "A 4.N2
MAKE "B 4.N2
```

```
MAKE "C 4.N4
MAKE "D 4.N4
MAKE "R :RABBITS
MAKE "F :FOXES
PD
DRAW.AXES
SETPOS LIST :R - 100 :F - 100
PD FULLSCREEN
STEP :R :F
END

TO DRAW.AXES
PU FULLSCREEN
SETPOS [-100 -100]
SETH 90
PD FORWARD 200
DRAW.ARROW
PU FORWARD 15
WRITE.R
SETPOS [-100 -100]
SETH 0 PD FORWARD 200
DRAW.ARROW
PU FORWARD 10
WRITE.F
END

TO DRAW.ARROW
RT 30 BK 10 FD 10 LT 60 BK 10
FD 10 RT 30
PU
END

TO WRITE.R
PR SETH 0
FD 10 RT 90 FD 5 RT 90 FD 5 RT 90 FD 5
RT 50 BK 8 PU
END

TO WRITE.F
SETH 0
PD FD 10 RT 90 FD 8 BK 8
RT 90 FD 5 LT 90 FD 5 PU
END

TO STEP :R :F
SETPOS LIST :R - 100 :F - 100
MAKE "DR (:A * :R) - (:C * :R * :F)
MAKE "DF (-:B * :F) + (:D * :R * :F)
STEP :R + :DR :F + :DF
END
```

```
TO START
PRED 40 80
END
```

As usual the turtle steps through the solution of the rate equations in the **STEP** procedure. In **PRED** the coordinate axes are drawn, the initial conditions set, and some bookkeeping details are taken care of. If you run the **START** procedure you begin the population study with 40 rabbits and 80 foxes. You will obtain the solution illustrated in Figure 12.4. (Because the Logo solution to the rate equations is only an approximation, the Logo curve will not quite close on itself. The exact solution would produce a closed curve. To get a better solution you might shorten the time interval between steps.) We will leave the interpretation of this figure to the Problems section.

The Volterra equations are rough approximations of the actual events as they are found in nature. They demonstrate that the rabbit and fox populations do interact. When there are too many rabbits and too few foxes, the rabbits decline and the foxes grow. The two populations fluctuate about a mean with the same period. For small fluctuations about the equilibrium state the oscillations are harmonic. There are numerous examples of competing systems which may be similarly represented: Population densities of government agencies, political parties, political ideologies, countries, chemical reactants, biological organisms, wages and prices, and many more interacting and competing systems.

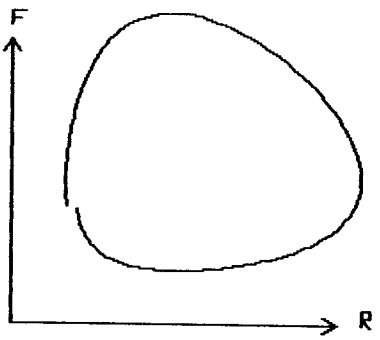


Figure 12.4 Rabbit and Fox Population

Problems

1. How is the period (T) of the oscillator related to w? To answer this question you must introduce the time into your program and have it printed to the screen. Write a program which will produce the figure and print the information shown in Figure 12.5. The equations represent the current values of the displacement Y, the velocity V, and the time. The specific numbers will not be the same but the format should be similar. To produce a print statement like that above you might consider:

```
(PRINT [Y=] TWO.DEC YCOR [V=] TWO.DEC :VEL [TIME=]
 :TIME)
```

This will print a single line and round **YCOR** and **:VEL** to two decimal places. We define **TWO.DEC** by the procedure:

```
TO TWO.DEC :X
OP (ROUND 100 * :X) / 100
END
```

which rounds a number to two figures after the decimal.

Armed with this information you should be able to find the period. By varying w, choose between the following alternatives:

a. T = constant * w
b. T = constant * $\sqrt{w}$
c. T = constant / w
d. T = constant / $\sqrt{w}$

Can you show that the constant is approximately 2π?

2. How is the period of oscillation related to the amplitude? Try different amplitudes for a fixed w and show that the period remains the same. This is why harmonic oscillators make such good clocks. They have the same period regardless of the amplitude. Even though the clock winds down (the amplitude becomes smaller and smaller) the

Figure 12.5 Y = −30.49 V = 5.18 TIME = 57

period remains constant. Unfortunately most physical systems are not exactly harmonic. The simple pendulum satisfies the harmonic oscillator equation

$$a = -w^2 x$$

for small x. (Here x represents the *angular displacement* and a the *angular acceleration* of the pendulum.) For larger x we find:

$$a = -w^2 \sin x$$

Using this relation, write a Logo program for the pendulum and see how the period changes with amplitude. Show that the period decreases with decreasing amplitude. (This means that the clock will run fast as it winds down.) Can you see why this is so by looking at the two equations above?

3. By experimenting with different initial values of R and F see if you can determine the "equilibrium state" of the system; that is, those initial values of R and F for which there is no change with time. Can you see why these are the equilibrium values by examining the rate equations for the rabbit/fox populations?

Projects

1. The system in Figure 12.6 is a two-dimensional harmonic oscillator. If all the springs are the same, the fundamental equations are

$$a_x = -w^2 x$$
$$a_y = -w^2 y$$

Write a program which will draw the path of a two-dimensional oscillator. Show that the path is an ellipse, a circle, or a straight line depending on the initial conditions. Other examples of two-dimensional harmonic motion are the pendulum bob on the end of a string and a ball rolling freely in a bowl. Examine for yourselves and see if the trajectories are indeed ellipses, circles, and straight lines.

2. If the springs in the Project 1 are not the same, the fundamental equations become

$$a_x = -(w_x)^2 x$$
$$a_y = -(w_y)^2 y$$

Repeat Problem 3 for this more general case. The figures you obtain will be complex curves called Lissajou figures.

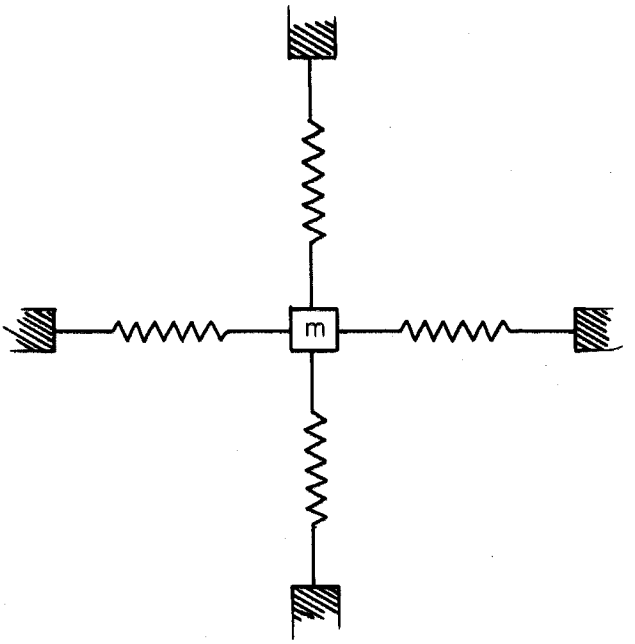
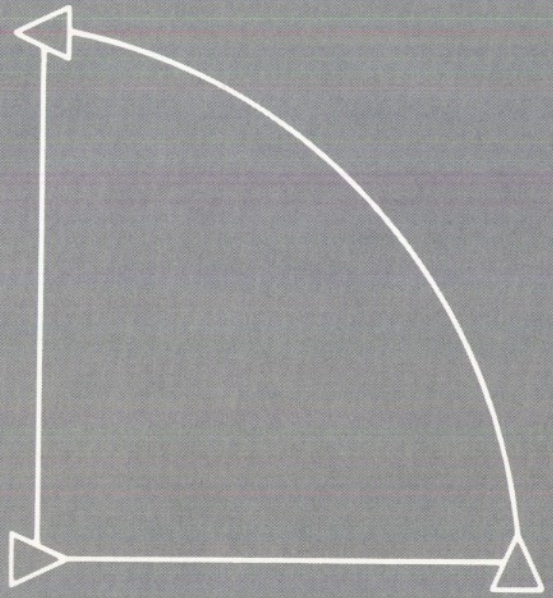


Figure 12.6 A Two-dimensional Harmonic Oscillator

3. Another way to represent the changing populations of rabbits and foxes is to plot R and F as functions of time. Write a program using two turtles to make this plot.

chapter



13 The Big Bang

Introduction

How did it all begin? How was the Earth formed? How did our solar system come to be? Sol, the star at the core of our solar system, is but one of many billions of stars in our galaxy. Our galaxy is but one of many billions in the universe. How did this whole colossus begin?

There are few questions to which we can give an ultimate answer. Is there an ultimate answer to this ultimate question? Can we look back in time to the beginning of the beginning? We surely can look into the past. We can look at the sun and see it as it was eight minutes ago—the time it takes for sunlight to reach the Earth. A look at Alpha Centauri, the nearest star, reveals its position as it was four years ago. Light from the nearest galaxy takes two hundred thousand years to reach us. Light from the farthest galaxy travels for one billion years before we see it. This gives us a considerable look into the past of that particular galaxy. But this is not quite what we are after. We would like to look all the way back to the beginning of time. Is there any light out there in space that was generated at the very beginning of

the universe that might give us a clue to how the universe began?

The most widely accepted theory is that there was indeed a beginning to the universe. In this theory the universe began some twenty billion years ago in a giant explosion. The density and temperature were enormous. The density was one billion times that of water and the temperature thirty billion degrees Celsius. The entire system exploded in a very big way. This explosion is referred to as the Big Bang. (Do not ask what came before the Big Bang. It is not possible to see beyond the Big Bang.) There was a great fire ball which produced an enormous flash of light. Now one might expect that this light would have been absorbed after all this time of flying about the universe at 360,000 mi/sec and there would be nothing left of it to testify to its early beginnings. However, there was so much light and there is so little matter in the universe to absorb the light that most of it has survived.

Now it might have survived, but light travels so much faster than matter that it must surely have moved so far away from the rest of the matter in the universe that there would be very little chance of our ever seeing it. But light is bound up with all the matter in the universe; it cannot escape. The reason it is trapped is that light is subject to the same laws of gravity as all the rest of matter. Light is attracted to the matter of the universe. The attraction is strong enough that all light is bound to the universe in much the same way that the Moon is bound to the Earth and the Earth to the sun.

As this great quantity of light generated by the Big Bang expanded with the rest of the matter in the universe, it cooled. It has had twenty million years to cool. It has cooled to a point where it is but a shadow of its former self. Its present temperature is about 3° above absolute zero, or 456° F below zero on the Fahrenheit scale. It was the 3° K background radiation that was first discovered by Penzias and Wilson and it was for this achievement that they were awarded the Nobel prize in physics.

The light they observed had all the properties that thermal radiation at this temperature should have. It has the proper spectral distribution; that is, the proper distribution of frequencies. At so low a temperature the dominant fre-

quency is very low. The most intense frequency is only 3×10^{11} Hz. This is ten thousand times smaller than the most intense frequency given off by our sun, which coincidentally lies right in the middle of the visible spectrum. (Aren't we lucky. Or maybe it isn't luck. Maybe the eye evolved in such a way that it was most sensitive to the most abundant frequencies given off by the sun.) Furthermore, Penzias and Wilson discovered that the 3° K background radiation was isotropic; there was as much light directed one way as any other. These two observations, the frequency distribution and the isotropy, were convincing evidence that this was thermal radiation.

The next time you gaze out into space on a dark night and dream the deep thoughts that occasionally come upon us when viewing the heavens, think of the 3° K background radiation out there. That light, if our eyes were sensitive enough to see it, allows us a view of the birth of our universe; an event which occured twenty billion years ago. That is a long look back into the past. It is as far back as man will ever be able to see.

Hubble's Law of the Expanding Universe

The 3° K background radiation is not the only evidence that our universe began as a great fire ball. Edwin Hubble has provided us with much more immediate evidence of the Big Bang theory.

Hubble made a study of all the known galaxies. He measured their distances from the Earth and their velocity. He discovered two curious things. The first was that all the galaxies are moving away from our galaxy and second, the further a galaxy is from us the faster it is receding from us (see Figure 13.1). If all the galaxies are moving away from us, does that mean that we are located in some special place in the universe? Are we located at the center of all galactic motion?

There was another time in history when man believed that he occupied a central place in the universe. In the now discarded Ptolemaic universe, the sun and all the planets revolved about the Earth. That theory did not work out very

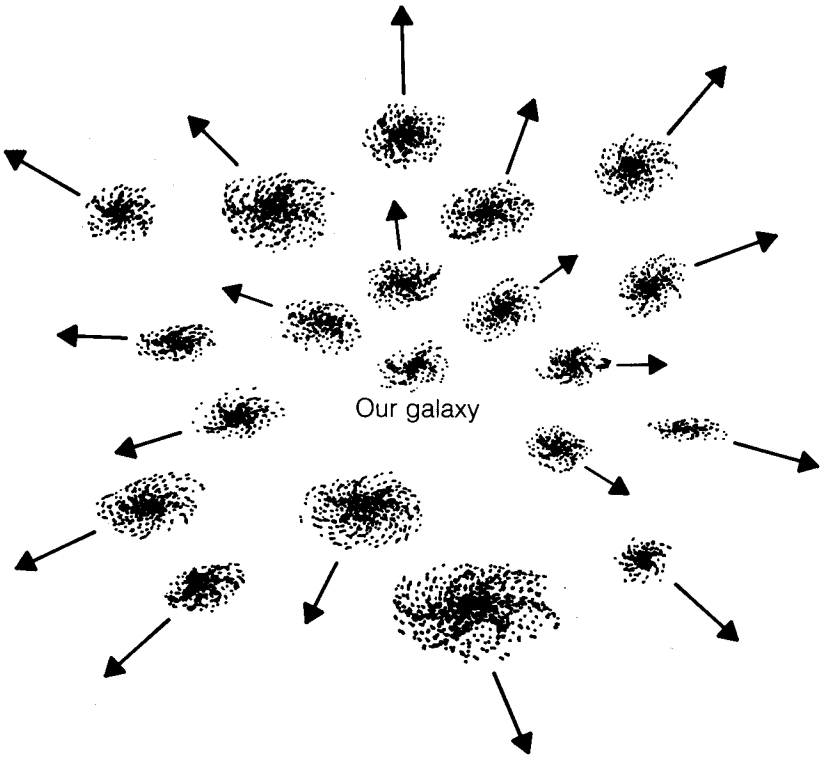


Figure 13.1 The Big Bang

well and having been once bitten we must be twice shy. The fact that all galaxies are moving away from our galaxy may not imply that our galaxy is central. Hubble observed that if all the matter in the universe was born in a mighty explosion, then *all* galaxies would be moving away from all other galaxies. To understand how this is not only possible but inevitable, let us examine an analogous situation for which Hubble's law is more easily derived.

Turtle Galaxies

We may illustrate Hubble's law by comparing it with a horse race (or perhaps a turtle race). In a horse race, all horses start from a common point at the same time (a samll bang). The horses bolt from the starting gate with different speeds. We will assume that in this race (unlike a real race) each horse continues to move with a constant velocity throughout the race. Clearly the fastest horse will win. Of

interest to us is the view of the race as seen by any one of the jockies. We maintain, and we will let the turtles show us presently, that at *any* time during the race, *each* jockey sees the other horses moving away from him. A jockey in the middle of the pack sees the horses ahead of him continually moving still further ahead. He sees the horses behind him continually moving still further behind. Moreover, the further a horse is from him, the faster it is moving away from him.

To see this in action, run the **BIG.BANG** program:

```
TO BIG.BANG
WINDOW
MAKE "V0 PICK.RAN
MAKE "V1 PICK.RAN
MAKE "V3 PICK.RAN
MAKE "V3 PICK.RAN
MAKE "V.LIST (SE :V0 :V1 :V2 :V3)
MAKE "V 0
CLEARSCREEN PU
SET.AT START 0
HT
STEP
END

TO STEP
ASK 0 [FD ( :V0 - :V )]
ASK 1 [FD ( :V1 - :V )]
ASK 2 [FD ( :V2 - :V )]
ASK 3 [FD ( :V3 - :V )]
IF KEYP [MAKE "V ITEM (1 + FIRST RC) :V.LIST]
STEP
END

TO PICK.RAN
OP (1 + RANDOM 100) / 40
END

TO SET.AT.START :N
IF :N > 3 [STOP]
MAKE :N [[-120 0] 90]
SET.AT.START : N + 1
END

TO ASK :N :CMD
PU SETPOS FIRST THING :N
SETHEADING LAST THING :N
DRAW.TURTLE [PE]
```

```
PD
RUN :CMD
DRAW.TURTLE [PD]
MAKE :N LIST POS HEADING
END

TO DRAW.TURTLE :CMD
RUN :CMD
LT 45
REPEAT 4 [FD 5 RT 90]
RT 45
END
```

(If you are using Apple Logo, load in the **ASK** procedure that draws the turtles.)

In **BIG.BANG** each horse is randomly given a velocity which lies between 1.00 and 3.25. (For Apple Logo it is advisable to increase these velocities. You might try **OP (1 + RANDOM 100) / 10** in **PICK.RANDOM**.) **V.LIST** is a list containing the velocities of the four horses. We will make use of this list and the variable **:V** later. All horses are put in the gate (**SETPOS [−120 0]**) and headed down the track (**SETH 90**).

In **STEP** the horses begin to step through their paces, each with its own constant velocity. Since **:V = 0** we can for the moment ignore it. Our objective is to view the race from the point of view of some particular jockey. (By analogy, we wish to view the other galaxies that started with us at the Big Bang and observe their motion relative to us. The horse race is a one-dimensional analogue to the real three-dimensional problem.) If we press the #2 key, for example, we are selecting turtle #2 which is the third horse. When we make this selection, **:V** is set equal to **:V2** (the third member of **:V.LIST**). From this point on all horses move relative to horse #2. Notice that **:V2 − :V** will be zero and so this horse comes to rest. The horse you are riding is always at rest *relative to you*. You should observe two things. First, all horses move away from the chosen horse. Second, the velocity at which they recede is proportional to the distance. The further a horse is from you, the greater its velocity relative to you.

Projects

1. If your version of Logo has the ability to edit the turtle shapes you might "number" the turtles.

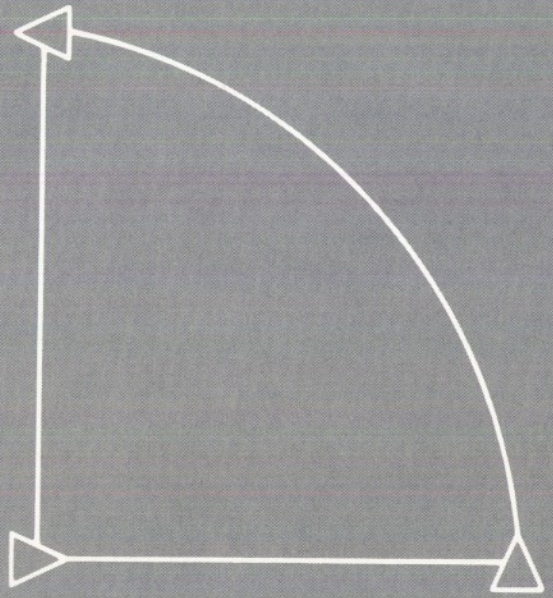
2. Hubble stated his law in the following quantitative form:

$$\frac{\text{relative velocity}}{\text{relative distance}} = K$$

and K is the same constant for all galaxies. See if you can verify this law from our horse race. Compute K for *each* horse relative to the chosen horse and see if it is a constant. To get the distance between horses you might **MAKE "X0 XCOR** just after you **ASK 0** to go **FORWARD. MAKE "X1 XCOR** just after you **ASK 1** to go **FORWARD**, and so on. The rest is up to you.

3. How about a two-dimensional model of the Big Bang?
4. Write a program which will place the turtles at random distances from the origin and give them velocities proportional to that distance. Reverse all velocities and confirm Hubble's law.
5. Add to **BIG.BANG** the option of pressing the key "R" which reverses the velocities of all turtles. You should retain the option of pressing 0, 1, 2, or 3. If you reverse the velocities after selecting horse 2 (that is, choosing to observe the race from the point of view of the jockey on horse 2), how does the race appear?

chapter



14 Radioactive Decay

Introduction

The nucleus of most atoms is stable. However, a few are unstable and decay by emitting a particle (generally an alpha particle or an electron). Such a nucleus is said to be radioactive and the particle emission is called *radioactive decay*. For example U^{238} decays into Th^{234} by emitting an alpha particle (the nucleus of helium). Carbon 14 radiates an electron and becomes nitrogen 14.

This process of radioactivity is spontaneous and random. The decay of a radioactive atom is independent of its environment and its past history. It is as if the nucleus was continuously spinning a roulette wheel or rolling a die and will radiate only when its lucky number comes up. Although the exact moment of decay is pure chance, it is nevertheless governed by strict rules of probability. Even though we cannot say when a given number will appear on a die, we can say with certainty that over the long haul it will appear with a frequency rate of one in six. If we roll the die a very large number of times we will see any given number appear one-sixth of the time.

In a similar way, if we have 1,000,000 radioactive atoms and the probability of each atom radiating during a 1 sec interval is .1 then we may expect with reasonable certainty that 100,000 atoms will radiate in the first second. This will leave 900,000 atoms of which approximately 90,000 will radiate during the next second, and so on. The rule therefore which determines decay of the system is the radioactive rate equation:

$$\text{Rate of increase in } N = -P\,N$$

where N is the current number of radioactive atoms and P is the probability of decay per atom per unit time. Notice that if $P = .1$ and $N = 1{,}000{,}000$, then the rate of increase in N is $-100{,}000$ as we found above.

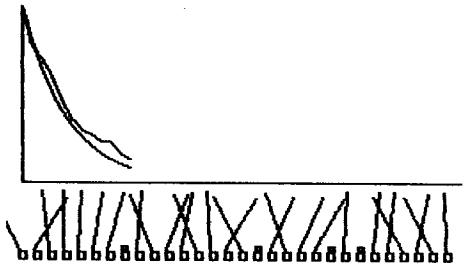
Radioactive Turtle

The program **RAD.DECAY** illustrates the radioactive process.

```
TO RAD.DECAY
WINDOW CLEARSCREEN HT
PRINT [PROB. OF DECAY IS ONE CHANCE IN ( ? )]
MAKE "CHANCE FIRST RL
MAKE "NUMBER 30 MAKE "TIME 0 MAKE "N.DECAY 0
MAKE "N.SCALE 4 MAKE "T.SCALE 6
MAKE "X.BOX -130 MAKE "Y.BOX -50
MAKE "SIZE 4 MAKE "SEPARATION 9
BOXES :X.BOX Y.BOX :SIZE :SEPARATION :NUMBER
BOXES :X.BOX :Y.BOX + :SIZE + 1 .75 * SIZE
 :SEPARATION :NUMBER
MAKE.STATE :NUMBER
DRAW.AXES
ASK 1 [HT PU SETPOS LIST :X.BOX :NUMBER * :N.SCALE
 PD]
ASK 2 [HT PU SETPOS LIST :X.BOX :NUMBER * :N.SCALE
 PD]
STEP :NUMBER
END

TO STEP :N
CYCLE 0
MAKE "TIME :TIME + 1
ASK 1 [SETPOS LIST ( :X.BOX + :T.SCALE * :TIME ) (
 :N.SCALE * ( :NUMBER - :N.DECAY ) ]
ASK 2 [SETPOS LIST :9 X.BOX + :T.SCALE * :TIME ) (
```

```
 YCOR - ( YCOR / :CHANCE ) ) ]
( PRINT [TIME =] :TIME [NO.DECAYED =] :N.DECAY )
STEP :N
END

TO MAKE.STATE :N
MAKE "STATE []
REPEAT :N [MAKE "STATE FPUT "R :STATE]
END

TO DRAW.AXES
PU SETPOS LIST :X.BOX :N.SCALE * :NUMBER
PD SETPOS LIST :X.BOX 0
RT 90 FD 300
END

TO CYCLE :N
IF :N = :NUMBER [STOP]
IF AND ( ( FIRST :STATE ) = "R ) ( ( RANDOM :CHANCE
 ) = 0 ) [MAKE "STATE ( LPUT "S BF :STATE ) ( EMIT
 :N )] [MAKE "STATE ( LPUT ( FIRST :STATE ) ( BF
 :STATE ) ) ]
CYCLE :N + 1
END

TO EMIT :N
BOX ( :X.BOX + :N * :SEPARATION ) ( :Y.BOX + :SIZE
 + 1 ) ( .75 * :SIZE ) [PE]
PD SETH ( -30 + RANDOM 61 ) FD 40 PU
MAKE "N.DECAY :N.DECAY + 1
END

TO BOXES :X :Y :SIZE :SEP :N
IF :N = 0 [STOP]
BOX :X :Y :SIZE [PENDOWN]
BOXES :X + :SEP :Y :SIZE :SEP :N - 1
END

TO BOX :X :Y :SIZE :CMD
SETH 0
PU SETPOS LIST ( :X - :SIZE / 2 ) ( :Y - :SIZE / 2
 )
RUN :CMD
REPEAT 4 [FD :SIZE RT 90]
END
```

If this program is run you will observe something like that illustrated in Figure 14.1. There are thirty atoms in the system represented by the thirty boxes at the bottom of the



```
TIME = 9 NO.DECAYED = 23
TIME = 10 NO.DECAYED = 25
TIME = 11 NO.DECAYED = 26
```

Figure 14.1 Radioactive Decay of 30 Atoms

figure. On top of some boxes there are still smaller boxes. These represent the particles that will be emitted by the radioactive atoms. On other atoms these smaller boxes are replaced by lines coming from the top at some random angle which represent the track of the emitted particles. The two curves plotted at the top of the figure represent both the actual number of radioactive atoms as a function of time and the theoretical curve as predicted by the radioactive rate equation. In the legend below the atoms the time and number of decayed atoms is printed.

Briefly, the individual procedures in this program function as follows. In **RAD.DECAY** the radiation probability is selected. It is the inverse of **CHANCE**; that is, one chance in **CHANCE**. For example, if there is 1 chance in 9 of the atom decaying then the probability of decay (P) is 1/9. (The reason for introducing this definition is that it simplifies the random selection process in **CYCLE**.) Also in **RAD.DECAY** scale factors for the particle number (**N.SCALE**) and time (**T.SCALE**) are introduced to improve the legibility of the plot. **X.BOX** and **Y.BOX** define the coordinates of the first box as well as the position of the origin of the coordinate axes. **SIZE** and **SEP** represent the size and separation of the boxes.

BOXES draws the boxes with centers at the coordinates **:X :Y**, of size (**:SIZE**), separation (**:SEP**), and number (**:NUMBER**). The second call to **BOXES** draws the three-quarter size boxes on top of the larger boxes.

In **MAKE.STATE** a list is created which will be used to determine the state of the collection of atoms. Initially

```
:STATE = [RRRRR . . . R]
```

There are 30 Rs for the 30 radioactive nuclei. If the third nucleus emits a particle and becomes stable, the third R is changed to S, so that **STATE** becomes

```
:STATE = [RRSRR . . . R]
```

In this way the program keeps track of the progress of the radioactive decay process.

DRAW.AXES does just that. The **STEP** procedure controls the evolution of the collection of radioactive atoms. It first calls **CYCLE 0** where the computer cycles through the list defined by the present **:STATE** and chooses at random which atoms will decay and which will not. For example, if **CHANCE** is 9, then **RANDOM 9** will yield a value between 0 and 8. One ninth of the time it yields a 0. When this happens the radioactive atom will **EMIT** a particle. The emission is accomplished by erasing (**PE**) the small box and drawing a tracer line 40 units in length at a random angle between $-30°$ and $+30°$.

Run the program and select various values for the radiation probability. Notice the change in the decay rate. Next reduce the number of nuclei by changing **NUMBER** and notice the increase in the discrepancy between the actual decay rate and the predicted decay rate. This is because statistical predictions are always much better with larger samples. It is for this reason life insurance companies make money every year.

Half Life

The rate at which radioactive atoms decay is often expressed in terms of their "half life." The half life is the time it takes for half of a large collection of radioactive atoms to decay. The reason we cannot define a "full life" is because of the probabalistic nature of the decay process and the unreliability of the statistics when the numbers become small.

The program **HALF.LIFE** will allow us to determine the relationship between the half life T and the radiation probability P.

```
TO HALF.LIFE
MAKE "P GET.P
```

```
DRAW.AXES
SETPOS [0 100] PD
STEP 0 100
END

TO GET.P
PRINT [DECAY PROBABILITY = ?]
OP FIRST RL
END

TO DRAW.AXES
PU SETPOS [0 100]
SETH 180 PD FD 100 LT 90 FD 320 PU
END
TO STEP :TIME :N
SETPOS LIST XCOR + 1 :N
IF :N < 50 [(PRINT [HALF LIFE =] :TIME) STOP]
STEP :TIME + 1 :N - :P * :N
END
```

In this example we begin with 100 atoms and record the time it takes for 50 to decay. In Figure 14.2 you see the results of **HALF.LIFE** with a probability of decay of .01.

Try different values of the radiation probability (any number between zero and one) and notice the change in the half life. In the Problems section we will consider the general relationship between T and P.

Carbon Dating

You are given a fragment of parchment purporting to be a portion of the Dead Sea scrolls. How might you verify the authenticity of the fragment? One thing you might do is verify its age by using a technique called "carbon dating."

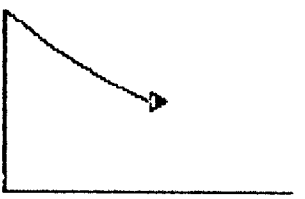


Figure Figure 14.2 Half Life = 69

Plants absorb carbon dioxide from the air. Radioactive carbon is generated in the air by cosmic ray neutrons bombarding nitrogen. This process happens at a continuous rate, so the proportion of radioactive carbon in the atmosphere is fixed and it is assumed to have been relatively constant over the past 10,000 years. The proportion of radioactive carbon to stable carbon is not large. It is about one part in 10^8. Thus if the sample contained 1.0×10^{10} carbon atoms, only 100 of them would be radioactive C^{14}. The rest will be C^{16}. When the plant or animal dies, the radioactive carbon will begin to decay at a rate determined by the radioactive rate equation:

$$\text{Rate of increase in } N = -PN$$

For C^{14} the value of P is 1.2×10^{-4} particles per year. The half life is 5,730 years.

Let us write a program to compute the age of a specimen given certain experimental data. To simplify the turtle graphics we will use convenient numbers for our ficticious radioactive sample. Let us assume that the fraction of the ficticious radioactive atoms in living matter is .1 and that the probability of decay is .2 per year. A Geiger counter tells us that the sample is radiating at a rate of .8 particles per year. (Remember, these are not realistic numbers and have been chosen simply to keep the turtle on the screen. We will consider scaling techniques in the Projects section.) By chemical analysis we determine that there are in all 1000 atoms (both stable and unstable). From this information we wish to determine the time that has lapsed since the specimen died.

If one tenth of the atoms were radioactive when the organism died then $.1 \times 1000 = 100$ were radioactive at its death. If the sample is decaying at a rate of .8 atoms per year, then because

$$\text{Rate of increase in } N = -PN$$

and P is .02 it follows that the current value of N is 40. To determine the age of the specimen we need to find a solution of the rate equation. We need to know how long it takes for N to decrease from its original value of 100 to its present value of 40. To do this we modify the **HALF.LIFE** program by removing the conditional statement (**IF N < 50 . . .**) and instead print the number of particles and the time.

```
(PRINT [NUMBER =] :N [TIME =] :TIME)
```

If we run this modified program with P = .02 we will find that **NUMBER = 40** when **TIME = 45**. Therefore the specimen is 45 years old.

Problems

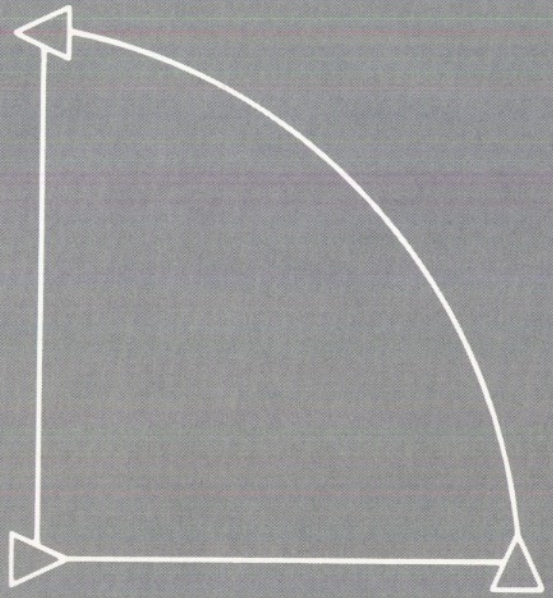
1. Use the **HALF.LIFE** program to determine the precise relationship between P and T. (Hint: 1n 2 = .6931. If you have not been exposed to logarithms, consider 1n 2 as a convenient abbreviation for .6931.)
2. A once living specimen currently radiates at the rate of .3 particles/year. When the organism died, .1 of the atoms were radioactive and the rest stable. The total number of such atoms is 1000. The radiation probability is .01. What is the current age of the sample? (Answer: About 120 years)
3. A bone fragment currently radiates at the rate of 2×10^6 particles/year. When the animal died, one tenth of the atoms were radioactive and the rest stable. The total number of such atoms is 9×10^{10}. The radiation probability is .001. Use the self-scaling program of Project 1 to determine the age of the bone fragment. (You will have to increase the time interval between steps. Do not forget to multiply **P * N** by the time interval to determine the change in N). (Answer: About 1500 years)
4. A radioactive specimen radiates at a rate of 10^8 particles/year with a radiation probability P = .01 particles/year. The specimen will not be safe to handle until the radiation level is reduced by a factor of 100; that is, until the rate is reduced to 10^6 particles/year. Use the self-scaling program of Project 1 to determine how long must one wait before the specimen is safe. (Answer: About 430 years)

Projects

1. Modify the **HALF.LIFE** program so that it is self-scaling and the original number of atoms (**:N**) is an input variable. By self-scaling we mean that for any choice of **:N**, the turtle does not go off the screen. Specifically, for any choice of **:N** the plot begins at **XCOR = 0** and **YCOR = 100**.
2. You can speed up the **RAD.DECAY** program by eliminating the graphics display of the atoms as boxes. Write a program which plots both the actual number of radiat-

ing atoms as a function of time and the number predicted by the radiation rate equation. Compare the predicted results with the actual numbers when the initial number of atoms is first 20 and then 100. Let the decay probability be one third.

chapter

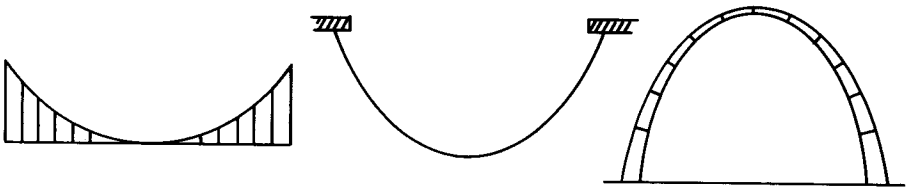


15 Bridges, Catenaries, and the Perfect Arch

Introduction

We would like to investigate the physics of three similar shapes: a suspension bridge, a hanging chain, and the perfect arch. They all look very much alike, as shown in Figure 15.1. In fact, if we turned the perfect arch upside down, it would have exactly the same shape as the hanging chain and a shape very similar to the suspension bridge cable although somewhat less blunt.

We will discuss the difference between these three shapes



Suspension bridge | Hanging chain | St. Louis arch **or** The perfect arch

Figure 15.1 Suspension Bridge, Hanging Chain, and Perfect Arch

later, but first we will focus our attention on just one—the suspension bridge.

The Suspension Bridge

The reason for the name "suspension bridge" is that the roadbed is suspended from the supporting cable. The cable is attached to two towers, one at either end. Vertical cables hang from the suspension cable and the roadbed is supported by these vertical cables. What we would like to understand is why the suspension cable is shaped the way it is. Furthermore, we would like to have the turtle draw a bridge for us.

To examine the physics of the cable, we will try to put the bridge together bit by bit. Our construction technique will be somewhat unorthodox, but no matter; we are looking for the basic physical laws and not their practical implementation.

We begin with a segment of the bridge already in place as illustrated in Figure 15.2. The bridge segment begins at the midpoint and runs to the right. The cable is held in place by two forces: a horizontal force H at the low point in the cable and a force T equal to the tension in the cable at the upper end. The only other force is the weight of the roadbed. (We will neglect the weight of the cables in com-

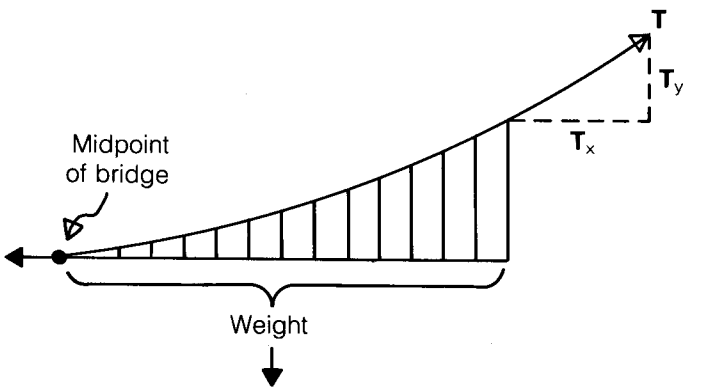


Figure 15.2 Segment of the Right Side of Bridge

parison to the roadbed.) If this system is to be in equilibrium, the net force must be zero. If we break up the cable tension T into its x- and y-components, we may balance the forces by setting:

$$T_x = H$$
$$T_y = \text{Weight}$$

Let us now imagine that we are to add the next segment of the roadbed of length DX (see Figure 15.3). This new segment adds an additional weight proportional to the length of roadbed. Let us define the density (DEN) of roadbed to be the weight per unit length. If we add a length DX, the additional weight is DEN * DX; that is, the weight per unit length multiplied by the length. This added weight must be supported by an increase in the vertical component of the tension (T_y). Therefore

$$\text{Increase in } T_y = \text{DEN} * \text{DX}$$

Since there has been no change in the horizontal forces we still have

$$T_x = H$$

By dividing these last two equations we obtain the following rate equation:

$$\text{Rate of increase of } T_y/T_x = (\text{DEN}) * (\text{DX}) / H$$

But we see from the similar triangles in Figure 15.3:

$$T_y/T_x = \text{DY/DX}$$

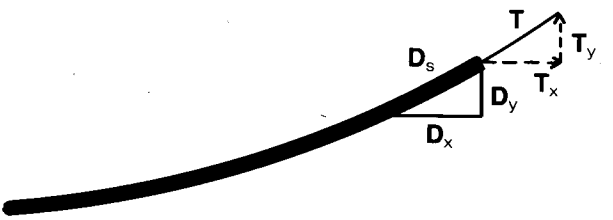


Figure 15.3 Adding to the Bridge

It is customary to define the slope of the cable by the ratio DY/DX so that

$$\text{SLOPE} = \text{DY/DX}$$

and our rate equation becomes

$$\text{Increase in SLOPE} = \text{DEN} * \text{DX/H}$$

and this is the *fundamental equation* which we shall ask the turtle to solve. It tells us that the slope of the supporting cable increases at a constant rate with the addition of each segment of roadbed. We will ask the turtle to step along the bridge cable, increasing the slope at a constant rate as he goes.

The Turtle Builds a Bridge

The solution of the fundamental equation is accomplished with the aid of the following recursive procedure:

```
TO STEP :SLOPE
SETPOS LIST ( XCOR + :DX ) ( YCOR + :DX * :SLOPE )
STEP :SLOPE + :DEN * :DX / :HOR
END
```

where we have noted that since **SLOPE = DY/DX** then **DY = SLOPE * DX**. We have denoted the horizontal force H by **:HOR**. Since the increment in the slope is a constant we may improve the readability of the procedure by **MAKE**ing **"INC.SLOPE :DEN * :DX / :HOR** so that our **STEP** procedure becomes:

```
TO STEP :SLOPE
SETPOS LIST ( XCOR + :DX ) ( YCOR + :DX * :SLOPE )
STEP :SLOPE + :INC.SLOPE
END
```

This procedure steps the turtle along the cable with the appropriate values of **DX** and **DY**. It continuously increases the slope by the amount required by the fundamental rate equation. But we need some way to draw both the left and right hand sides of the bridge as well as some way to stop the recursive procedure. Our complete program is:

```
TO BRIDGE :W
WINDOW
CS PU HOME
```

```
MAKE "DEN :W / 200
MAKE "HOR 50
MAKE "DX 10
MAKE "YO -50
MAKE "INC.SLOPE :DEN * :DX / :HOR
SETPOS LIST 0 :YO
PD STEP 0 1 1
STEP 0 1 -1
END

TO STEP :SLOPE :COUNTER :SIGN
SETPOS LIST (XCOR + :DX * :SIGN) (YCOR + :DX *
 :SLOPE)
MAKE "HEIGHT YCOR - :YO
BK :HEIGHT FD :HEIGHT
IF :COUNTER = 10 [BK :HEIGHT SETX 0 STOP]
STEP :SLOPE + :INC.SLOPE :COUNTER + 1 :SIGN
END

TO START
BRIDGE 200
END
```

BRIDGE takes one input, the weight (**W**) of the bridge. Our bridge is 200 units long and so the weight per unit length (the density) is **W / 200**. We have arbitrarily chosen the horizontal force (**H**) to be 50. To give our bridge more room in which to rise we have set the roadbed at **YO = −50**.

We have chosen three inputs for **STEP**. These are the **SLOPE** (initially zero), the **COUNTER** (initially 1), and the **SIGN** (initially 1). The **COUNTER** is used to enumerate the segments of the bridge as we add them one at a time. The **SIGN** is used to denote which side of the bridge we are building, +1 for the right side and −1 for the left side.

The value of **HEIGHT** is the height of the suspension cable above the roadbed. It determines the length of the support cables.

When the counter is equal to 10 we have completed the right side of the bridge and so we stop **STEP 0 1 1** and call **STEP 0 1 −1** to finish the left side.

Run **BRIDGE 200** to see Figure 15.4. **BRIDGE 300** produces a bridge whose cable rises more sharply. **BRIDGE 100** produces a bridge whose cable rises less sharply.

Figure 15.4 BRIDGE 200

The Perfect Arch

The great St. Louis arch, the "gateway to the west," was designed by Eero Saarine. It was a mammoth undertaking and twenty years elapsed between its design and completion. The shape of the arch is a catenary. We would like to explain what a catenary is and why it is such a good shape for an arch.

The word "catenary" comes from the Latin word for chain. A catenary is the shape which a chain assumes when suspended from its ends. An arch, on the other hand, looks more like an upside down catenary.

The St. Louis arch has been called "the perfect arch." Its perfection lies not in its beauty, although it is strikingly beautiful, but in its mechanical strength. An arch shaped like a catenary is stronger than an arch shaped like a parabola or a circle or any other shape.

Let us look at the chain and see what makes its shape so special. If a wire were suspended from its ends it might assume almost any form. This is because a wire is "stiff." A chain is not "stiff." A chain cannot support torsion or compressive forces. It can only support tensil (stretching) forces tangent to its length. There is an old saying: "You cannot push on rope" meaning: "You cannot make something or someone behave in an unnatural way." The natural property of a chain or a rope is that you can only pull on them.

In Figure 15.5 a rope hangs from its ends. Any segment of the rope exerts forces on adjoining segments. These forces must be stretching forces tangent to the rope at the point of contact. This is not the case for a hanging wire. Forces between adjoining segments may be in any direction. To make matters more complicated, there may also be a "couple." A couple is a measure of the tendency to cause rotation. By exerting a pair of forces on part of the wire, you can

Figure 15.5 Forces Acting Within a Rope

cause the rest of the wire to rotate. You cannot do this with a rope.

Now that we know what a rope or chain is we must determine the connection between the catenary and the "perfect arch." To see the connection we first observe that the shape of the catenary curve does not depend on the weight of the chain or on the strength of the gravitational force. All chains of a given length suspended from two given points will hang in the same shape. It will be the same shape on the Earth as on the Moon or on Mars. (You may simulate the reduction of the gravitational force by pinning the ends of a chain to a large piece of plywood. If the plywood is held vertically the chain assumes the shape of a catenary under 1 G. Tilt the board back at some angle, then tap the board until the chain assumes its equilibrium shape. This is the shape of a chain under less than 1 G. The tilting of the board effectively reduces the force of gravity. You will notice that the shape of the chain is unaffected by the angle of the board.)

The fact that the shape of the catenary is the same on all planetary bodies has an interesting generalization. Hang a chain. It takes the shape of a catenary. Reduce somehow the gravitational force (move it into outer space). The shape remains unchanged. It remains a catenary. Imagine now that the gravitational force is reduced to zero and then made *negative*. The gravitational force is now *acting in the opposite direction*. The forces acting within the chain will still be tangent to the chain but are now compressive forces. By reversing the gravitational force we *reverse all the forces*. Since the chain cannot support compressive forces it will collapse. But suppose that instead of the chain we had a rope with a bit of starch in it to give it some stiffness. Now if we inverted the gravitational force, or more simply we inverted the starched rope, it would still be shaped like a catenary (see Figure 15.6). All of the forces would be tangent

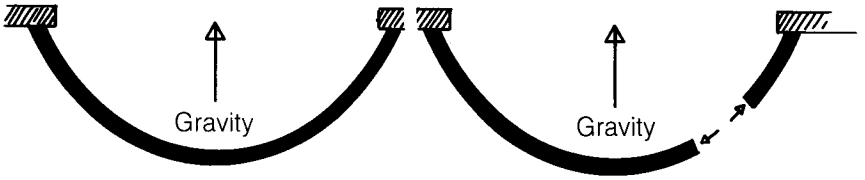


Figure 15.6 Tensile Forces Become Compressive Forces

to the catenary curve. The forces would all be compressive forces; no nontangential forces, no couples tending to cause rotation. Pure compression.

Let us now stack a pile of stone blocks alongside the starched rope in the same shape as the rope; that is, in the form of the catenary. The forces between the stones would be the same as the forces between the rope segments, purely compressive forces tangent to the arch. Now stone is very good at supporting compressive forces. No need to worry about the stones being crushed. But what about the individual stone sliding out of position? Or what about blocks tipping and so causing the arch to collapse? Again, no need to worry. The forces between stones is tangent to the arch and because of the way in which the stones are cut, the forces on each supporting face are perpendicular to that face. Thus no need for friction. No need for mortar to prevent tipping. (Mortar is not very good for this purpose anyway since it is very poor under tension. It will easily give way and the blocks would tip.)

In summary we see that by inverting the catenary we invert all the forces. The pulling forces in the rope become pushing forces as shown in Figure 15.7. The rope is good at withstanding pulling forces and the stone blocks are good at withstanding pushing forces. The inverted catenary is the perfect arch for stone blocks.

The Catenary Curve

There are two physical properties of a chain or rope which determine its shape when hung by its ends. The first is that the tension must be tangent to the curve and the second is that the tension cannot be compressive.

Let us give this information to the turtle and see if he can construct the catenary curve. We begin as we did in the construction of the bridge: start with a segment and add on

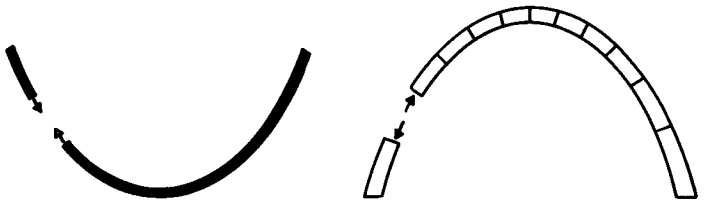


Figure 15.7 Forces Acting in a Rope and on an Arch

bit by bit. Figure 15.8 represents a segment of the right half of a hanging chain. The weight is not the weight of the roadbed as before, but the weight of the chain segment. This is an important difference. The weight of the chain is proportional to its length. The weight supported by the cable of a suspension bridge is proportional to the roadbed below or in effect, proportional to the projection of the cable on the horizontal axis.

If the forces in the figure are to balance, then

$$T_y = \text{Weight}$$
$$T_x = H$$

as before.

Let us add another link to the chain. In Figure 15.9, **DS** is the length of the link, **DX** its horizontal projection, and **DY** its vertical projection. Since we have added another link, the tension T must increase to support the weight of the new link. Therefore,

$$\text{Increase in } T_y = \text{weight of the new link}$$

Since the horizontal forces have not changed

$$T_x = H$$

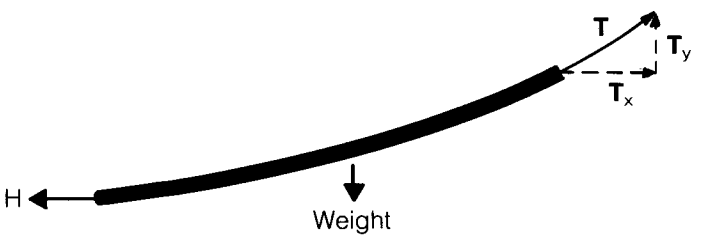


Figure 15.8 Segment of Right Side of Rope

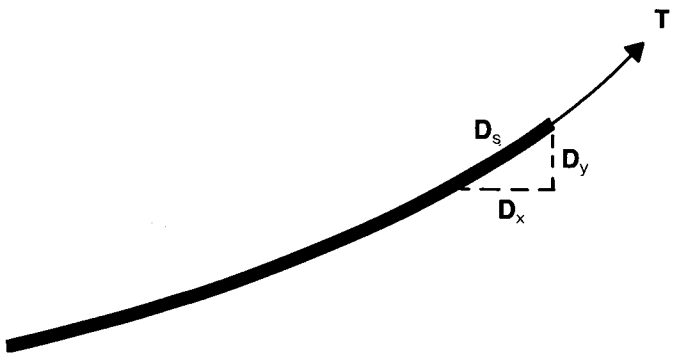


Figure 15.9 An Additional Segment is Added

Proceeding as in the bridge example we have

Increase in **SLOPE** = (weight of new link) / H

Now the weight of the link of length **DS** is **DS** * **DEN** where **DEN** is the weight of the chain per unit length. So finally

Increase in **SLOPE** = **DEN** * **DS/H**

The only difference between this equation and that for the bridge is that **DS** has replaced its horizontal projection **DX**.

The program to draw the catenary follows:

```
TO CAT :W
PU
MAKE "DEN :W / 200
MAKE "HOR 50
MAKE "DX 10
MAKE "YO -50
SETPOS LIST 0 :YO
PD
STEP.CAT 0 1 1
STEP.CAT 0 1 -1
END

TO STEP.CAT :SLOPE :COUNTER :SIGN
SETPOS LIST (XCOR + :DX * :SIGN) (YCOR + :DX *
 :SLOPE)
IF :COUNTER = 10 [PU SETPOS LIST 0 :YO PD STOP]
MAKE "DS SQRT (SQ :DX) + SQ :DX * :SLOPE
MAKE "INC.SLOPE :DEN * :DS / :HOR
STEP.CAT :SLOPE + :INC.SLOPE :COUNTER + 1 :SIGN
END
```

Notice that the **CAT** program is very similar to **BRIDGE**. Of course, we omit the roadbed and its supporting cables. We also omit **WINDOW** and **CS** so that you may superimpose the catenary on the bridge without erasing the bridge. If you run **BRIDGE 200** followed by **CAT 200** you should see something like that in Figure 15.10. As we observed earlier, the catenary rises more sharply than the bridge cable. You should be able to see this from the basic physics of the two systems without completing the calculations.

Projects

For the purposes of these projects we would like to recommend a modification which improves the accuracy of the **BRIDGE** program. The modification is identical to that used in our projectile motion program.

We recognize that the initial slope is 0. After one cable segment has been added the slope has increased by **INC.SLOPE**. Thus the average slope for this first segment should be **INC.SLOPE/2**. Therefore, in **BRIDGE** we call

```
STEP :INC.SLOPE / 2 1 1
STEP :INC.SLOPE / 2 1 (-1)
END
```

Not only does this get us started on a better foot, throughout the rest of the program the slope used in the calculations will be the average slope.

1. The first thing we would like to do is verify that the shape of the suspension bridge cable is a parabola. Pick some weight W, say 200. Plot the equation for a parabola, Y = A * X^2 where A is some chosen constant. See if you can find a value of A for which the parabola fits the bridge cable. You can do this by trial and error or by examining the coordinates of the end points of the bridge cable.

2. The St. Louis arch is not a true catenary. The thickness of the arch is not uniform but becomes progressively thinner toward the top. To study this effect, modify the **CAT** program to determine the shape of a hanging chain if the density of the chain varies uniformly from the center to the ends. Let there be 200 links, each 1 inch long. The first link to the right of center weighs 1/1000 lbs, the

second 2/1000 lbs, and so on. The last link on the right side weighs 100/1000 lbs. The chain is symmetrical about the center.

Problems

1. Can you build a bridge 200 units long between two towers 160 units high? The weight of the roadbed is 300. (Hint: You will have to change the tension **:HOR** in the cable at the midpoint.)
2. Determine the shape of the bridge cable if the central 100 ft of the bridge weighs 100 tons and the outside 100 ft (50 on each side) weighs 200 tons.

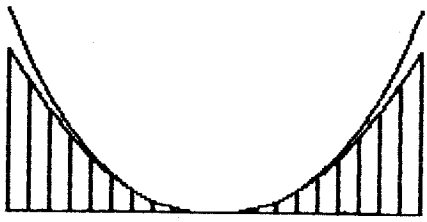
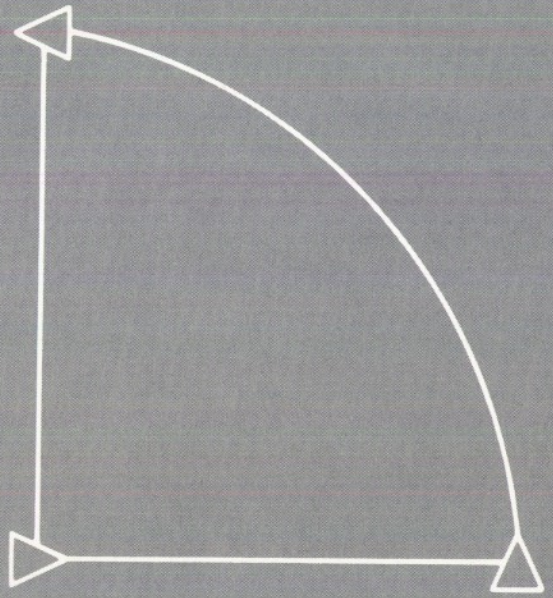


Figure Figure 15.10 Bridge and Catenary

chapter



16 Fishes and Optics

Introduction

The fish that got away is always bigger than the one we bring home to brag on. There may be many reasons for this, but at least one has a physical basis. Things do look bigger under water. By the same token, things above the water look smaller to the fish.

The reason for these optical illusions is that light rays bend as they pass from air to water or from water to air. The law that governs this bending is Snell's law.

Snell's Law

In Figure 16.1, a ray of light is incident from air and is refracted into the water. The angle between the incident ray and the normal is called the angle of incidence (i) and the angle between the refracted ray and the normal is the angle of refraction (r).

The way light bends as it passes from one medium to another depends on the velocity of light in the two media. The ratio of the velocity of light in a vacuum to that in the

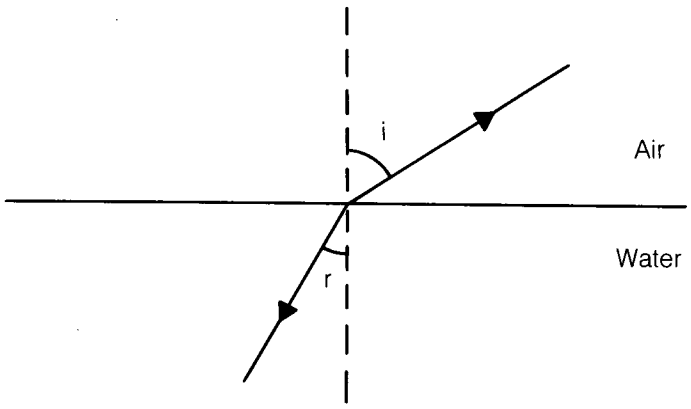


Figure 16.1 A Ray of Light Bends as it Enters the Water

medium is called *the index of refraction* and the symbol n is used:

$$n = c/v$$

The index of refraction of a vacuum is, by definition, one. The index of refraction of air is approximately 1, water 1.33, and glass 1.5.

We will call n_i the index of refraction of the incident medium and n_r that of the refractive medium. With these definitions we may state Snell's law; the fundamental law of geometrical optics. Snell's law says that:

$$n_i \sin i = n_r \sin r$$

Let us denote the index of refraction of water by N. Since the index of refraction of air is nearly one, Snell's law applied to the air/water interface in Figure 16.1 becomes:

$$\sin i = N \sin r$$

The Underwater Horizon

The Archer fish of Thailand derives its name from the curious way in which it makes its living. It has developed a technique for spitting a jet of water at unsuspecting bugs, knocking them out of the air into the water. It must take the Archer fish some time to learn this anti-aircraft technique because, as you see from Figure 16.2, the line of sight is not the same as the trajectory of the water jet.

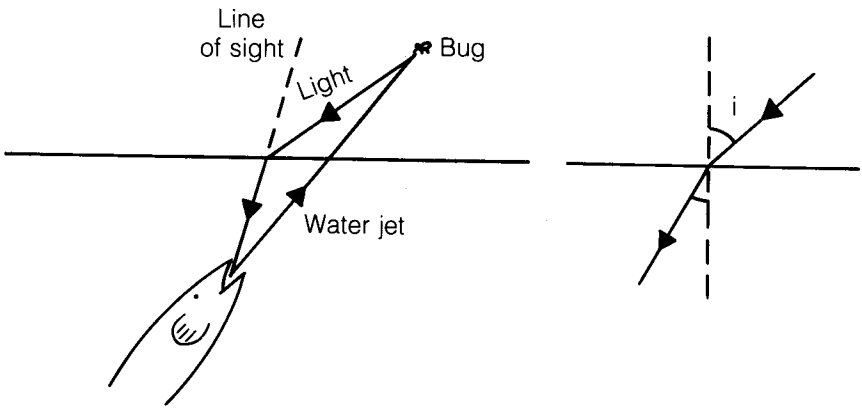


Figure 16.2 The Archer Fish

The Archer fish has apparently learned by trial and error to compensate for the discrepancy. We will write a program to determine the degree of compensation. In **ARCHER :DEPTH :R** the turtle draws the ray that enters the fish's eye. The fish is at a given **DEPTH** and looking in the direction **R**.

```
TO ARCHER :DEPTH :R
WINDOW
CS PU HT HOME
MAKE "N 1.33
DRAW.WATER
SETY -:DEPTH
SETH :R
PD FD DIST.TO.WATER
FD 100 BK 100
SETH I
FD 100
END

TO DRAW.WATER
PD RT 90 FD 100 BK 200 HOME PU
END

TO DIST.TO.WATER
OP :DEPTH / COS :R
END

TO I
MAKE "SINE.I :N * SIN :R
OP ARCTAN :SINE.I / SQRT (1 - SQ :SINE.I)
END
```

```
TO START
ARCHER 70 40
END
```

In **ARCHER** we have set n = 1.33, the index of refraction of water. The angles of incidence and refraction are I and R. To find the **DIST.TO.WATER** we do a little trigonometry. To determine the **ANGLE.R**, we first apply Snell's law:

$$\sin i = N \sin r$$

and then output the angle i. Since most versions of Logo do not have a primitive for the arcsine function, the procedure I instead outputs: arctan sin i / SQRT(1 − SIN^2 i), which is the same as outputting the angle i. (To see this, note that SQRT(1 − sin^2 i) = cos i and sin i / cos i = tan i.)

If we run **ARCHER 70 40** the turtle draws the ray that leads from the bug to the fish's eye (in the reverse order) and then extends the ray from the fish back into the air to make it easier to see the net deflection (as shown in Figure 16.3.) It is a simple matter to have the angles of incidence and refraction printed to the screen if you wish.

You will notice as you increase the angle of refraction R, that the ray above the water becomes more and more parallel to the surface of the water. The angle for which the ray in air becomes parallel to the water is called the critical angle. If you increase the angle of refraction beyond this critical angle you get an error message. If the fish were to look at such an angle it would see nothing above the water. (If it saw anyhing at all it would be objects at the bottom of the lake reflected off the surface.) The fish's horizon therefore is much less than our horizon (see Figure 16.4). We must turn through 180° to look from one horizon to the other. We leave it as an exercise to determine the fish's hori-

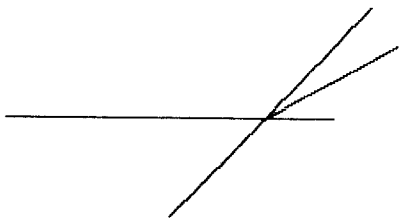


Figure 16.3 ARCHER 70 40

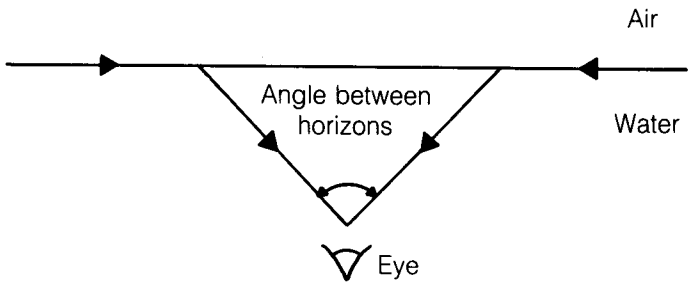


Figure 16.4 The Fish's Horizon

zon. Thus when the first amphibian emerged from the primordial oceans, it took a giant step toward broadening its horizons.

Apparent Depth

We have seen that the world above the water's surface appears distorted to the fish. By the same token, the underwater world appears distorted to the fisherman. For example, a fish appears to be closer to the surface than it actually is. In Figure 16.5 we show two rays emanating from a point P below the surface. These rays bend as they enter the air and are seen by the eye of the fisherman. To the eye, these rays appear to be coming from the point of intersection P′ and so the apparent depth is less than the actual depth.

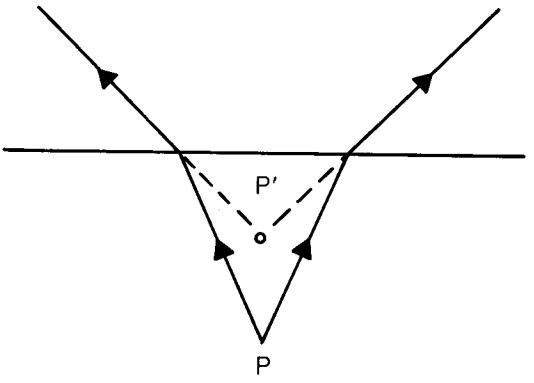


Figure 16.5 Rays Emerging From the Water

To see precisely how this comes about let us apply Snell's law and follow the rays as they pass from the water into the air. Since the rays are incident from the water and refracted into the air, Snell's law becomes:

$$N \sin i = \sin r$$

In the program **APPARENT.DEPTH** we apply this equation to determine the apparent depth of a fish whose actual depth is **:DEPTH**.

```
TO APPARENT.DEPTH :DEPTH
ARCHER.NEW :DEPTH 20
ARCHER.NEW :DEPTH -20
(PRINT [APPARENT DEPTH =] -YCOR)
(PRINT [ACTUAL DEPTH =] :DEPTH
END

TO ARCHER.NEW :DEPTH :I
PU HOME HT
MAKE "N 1.33
DRAW.WATER
SETY -:DEPTH
SETH :I
PD FD DIST.TO.WATER
SETH R
FD 100 BK 100
BK DIST.TO.Y.AXIS
END

TO DRAW.WATER
PD RT 90 FD 100 BK 200 HOME PU
END

TO DIST.TO.WATER
OP :DEPTH / COS :I
END

TO R
MAKE "SINE.R :N * SIN :I
OP ARCTAN :SINE.R / SQRT (1 - SQ :SINE.R)
END

TO DIST.TO.Y.AXIS
OP :DEPTH * (TAN :I) / SIN R
END

TO TAN :X
OP (SIN :X) / COS :X
END
```

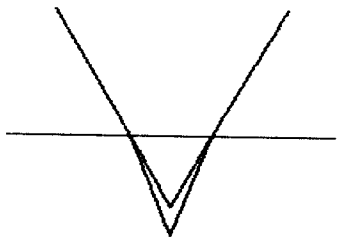
```
TO SQ :X
OP :X * :X
END

TO START
APPARENT.DEPTH 70
END
```

This program is very much the same as **ARCHER**. One of the primary differences is that the rays are now emerging from a source in the water (the fish) rather than a source in the air (the bug). Hence incident angles become refracted angles and refracted angles become incident angles.

The program draws two rays at 20° to the vertical. The emerging rays are extended back to their point of intersection on the y-axis. To an observer above the water the rays appear to be coming from this point of intersection. This is therefore the image point. If we run **APPARENT.DEPTH 70** we obtain Figure 16.6.

(These results are somewhat misleading. Two rays do not an image make. We should show that *most* of the rays leaving the source point P and entering the eye converge to the same image point P′. You can show this using the Logo program. Try three small angles; say 1°, 2°, and 3°. You will find that the apparent depth does not change much. For large angles there is a large variation in the depth but these rays do not enter the small pupil of the observers eye.)

Magnification



```
APPARENT DEPTH = 49.8789
ACTUAL DPETH = 70
```

Figure 16.6 APPARENT.DEPTH 70 20

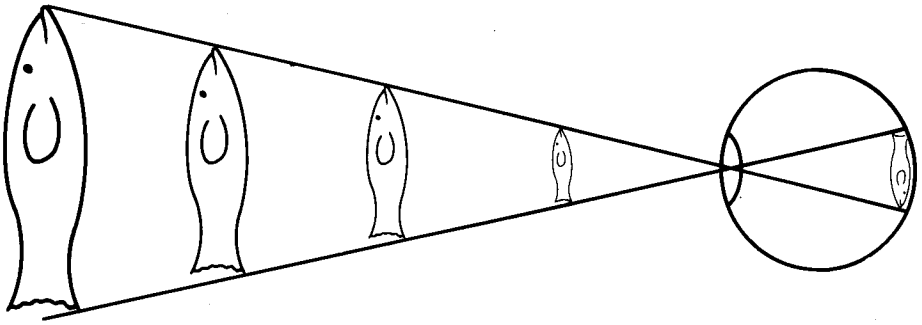


Figure 16.7 Retina Image of Fish

magnified. However, the question of magnification is quite different from the question of image location. A convex mirror brings an image closer to the observer but diminishes its apparent size. To illustrate the difference between apparent size and image position, you can see from Figure 16.7 that all the fish would create the same size image on the retina of the eye and so "appear" to be of the same size. To determine the apparent size of the fish we must examine the size of the image of the fish on the retina of the observer.

Figure 16.8 shows the two rays that leave each end of the fish and pass through the optical center of the eye. These

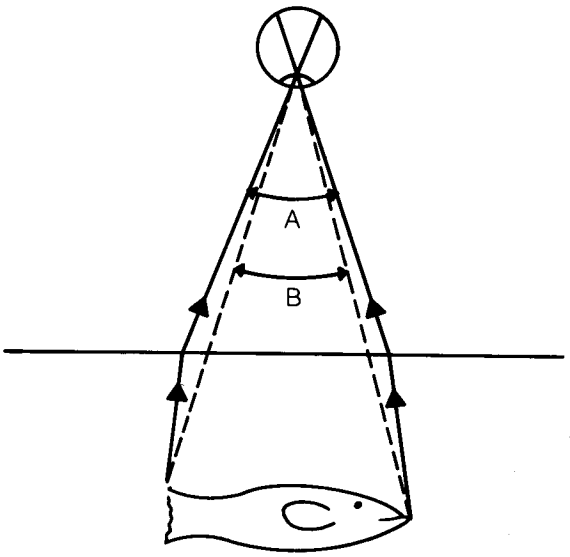


Figure 16.8 The Apparent Angle and Actual Angle

rays subtend an angle A. If there were no water, the rays entering the eye from either end of the fish would subtend a smaller angle B and so leave a smaller image on the retina. Magnification is defined as the ratio of the apparent angle to the angle in the absence of the water (or other optical system in general). So that

$$\text{Magnification} = A/B$$

To determine this magnification we may employ the program **MAG**. It will determine the magnification of a fish at a given depth below the surface as observed by an eye at a given height above the surface. The magnification depends also on the angle subtended by the fish.

```
TO MAG :HEIGHT :DEPTH :ANG.RAY
HT CS HOME
MAKE"EYE LIST 0 :HEIGHT
DRAW.WATER
MAKE "N 1.33
SETPOS :EYE
PD
DRAW.RAY :ANG.RAY
PU SETPOS :EYE PD
DRAW.RAY -:ANG.RAY
( PRINT [ANGLE IN AIR =] HEADING )
( PRINT [ANGLE IN WATER =] :ANG.RAY )
( PRINT [MAGNIFICATION =] :ANG.RAY / HEADING )
END

TO DRAW.WATER
PD RT 90 FD 100 BK 200 HOME PU
END

TO DRAW.RAY :ANG
SETH ( 180 - :ANG )
FD DIST.TO.WATER
SETH ( 180 - I )
FD DIST.TO.FISH
SETH TOWARDS :EYE
END

TO DIST.TO.WATER
OP :HEIGHT / COS I
END

TO DIST.TO.FISH
OP :DEPTH / ABS COS I
END
```

```
TO I
MAKE "SINE.I ( SIN :ANG ) / :N
OP ARCTAN :SINE.I / SQRT ( 1 - SQ :SINE.I )
END

TO ABS :X
IF :X > 0 [OP :X] [OP -:X]
END

TO START
MAG 100 60 20
END
```

In **MAG**, the eye is set at the given height above the surface, and a ray is drawn at an angle **:ANG.RAY**. The ray strikes the water where it is refracted to one end of the fish. A second ray is drawn at an angle **−:ANG.RAY** which intercepts the other end of the fish. (The early Greeks believed that objects were perceived by means of rays which were emitted from the eye. We begin the ray from the eye for convenience only. The rays perceived by the eye, of course, emanate from the fish.) The angular magnification is then printed.

By varying the three parameters (**:HEIGHT, :DEPTH,** and **:ANG.RAY**) we find different degrees of magnification. We explore this variety in the Problems section.

Problems

1. Determine the angle between opposite horizons for a fish.
2. Show that the apparent depth varies only slightly for small angles, but that its variability is greater for large angles. (Try 1° and 2° followed by 20° and 21°.)
3. Show that for small angles the ratio of the true depth to the apparent depth is the same for all depths. Can you identify the significance of this ratio?
4. Experiment with various values of **HEIGHT** and **DEPTH** in **MAG**. Show that the magnification lies between 1 and 1.33 (the value of N). Under what conditions is the magnification at maximum?

Project

1. The skin diver does not see the world as the fish does. This is because his eyes are encased in a mask and are therefore in air and not in water. Write a program that

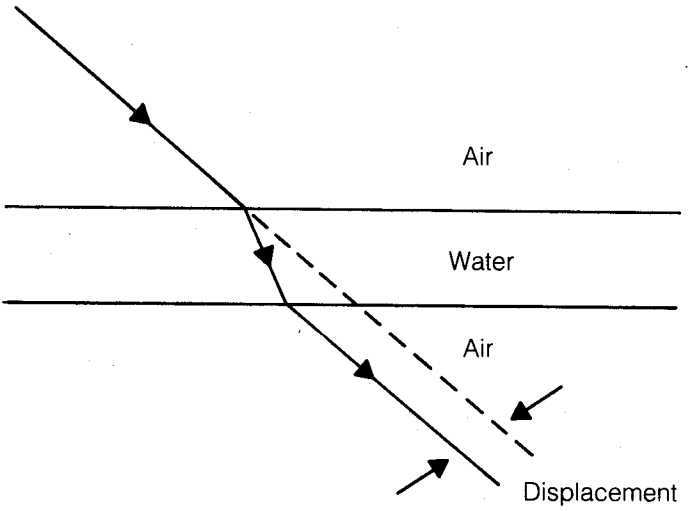
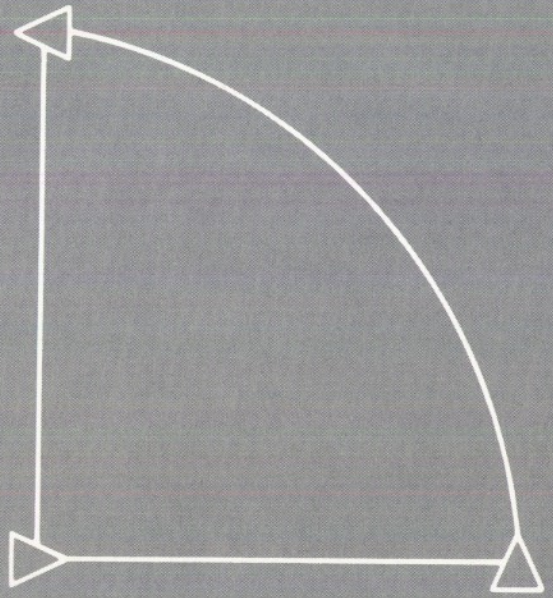


Figure 16.9 Ray of Light Passing Through Slab

shows how a ray through a slab is displaced but not deflected (see Figure 16.9). Get a printout of the displacement.

chapter



17 Rainbows

Introduction

There are few sights in nature as inspiring as a rainbow. It arches across the sky in glorious color. The end of the storm and gateway to a bright future.

The physical basis for the rainbow has been the subject of speculation for thousands of years. The myths that surround the rainbow are as colorful as the rainbow itself. There is even a bit of mythology to be found in some physics texts as well. The impression is often given that rain drops act like little prisms that split the white sunlight into its color constituents: red, orange, yellow, green, blue, and violet. If this were all there were to it we should see color wherever we look. Wherever there are illuminated raindrops there should be a spectrum of color. Instead we see color only in the rainbow arch.

As we shall show, we should see a rainbow even if the light from the sun were monochromatic (just one color). If we could somehow place a filter over the Sun that would allow only red light to penetrate, the rainbow would become a single bright red band arching across the sky. The first

thing we would like to explain is this red band and then the presence of the rest of the spectrum will be easy to understand.

The Red Band

So, let us investigate first the red band. To do so we must study what happens to a ray of light when it enters a raindrop.

When light encounters a sudden discontinuity such as an air/water interface, part of the light is reflected and part is refracted. In Figure 17.1, the dotted line is the normal to the surface. If the incident ray makes an angle i with the normal, the reflected ray also makes an angle i with the normal. The refracted beam is bent as it enters the water. If we denote the angle between the refracted beam and the normal by r, then i and r are related by Snell's law:

$$n_i \sin i = n_r \sin r$$

where n_i and n_r are the indices of refraction of light in their respective media.

Consider now the case of light entering a water drop. We wish to study the progress of a ray which is refracted as it enters the drop, reflected off the inside surface, and then refracted back into the air as illustrated in Figure 17.2. At each interface there is both a reflected and a refracted ray but we choose to follow only the indicated sequence. We

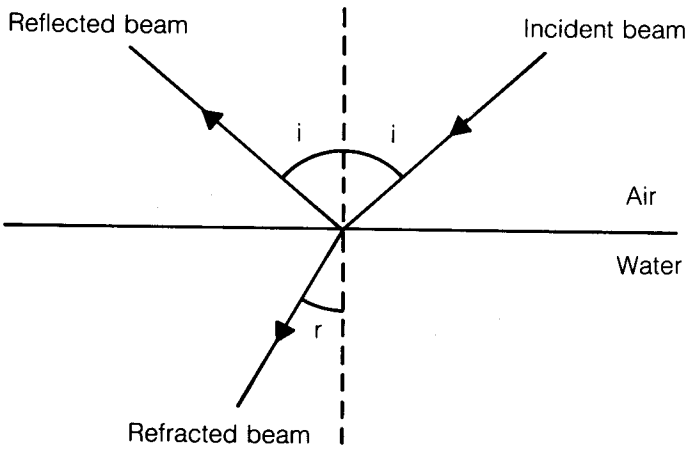


Figure 17.1 Light Reflected and Refracted off Water

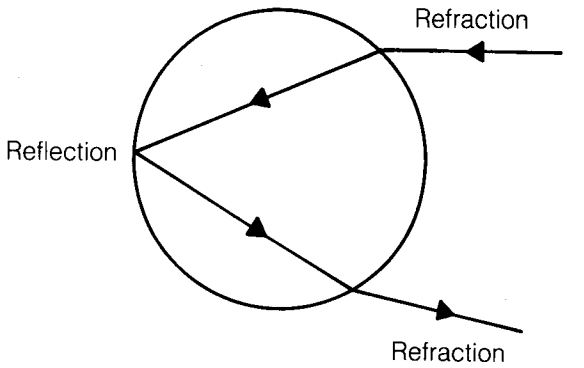


Figure 17.2 Ray of Light Through a Water Drop

shall see that this particular ray has a special property responsible for the rainbow.

To facilitate the calculation it is helpful to include some construction lines. The dotted lines in Figure 17.3 all represent radii of the circle. Because the radius of a circle intersects the circle perpendicularly, these radii are also the normals to the circle. The angles of incidence, refraction, and reflection are all measured relative to the extensions of the radii. Because the triangles within the circle are all isosceles triangles, it follows that all the indicated angles within the circle are equal to r.

Snell's law applied to the first refraction becomes

$$\sin i = n \sin r$$

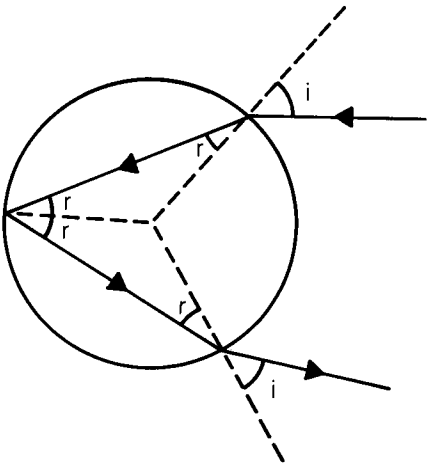


Figure 17.3 Construction Lines for the Rays

where we have set the index of refraction of air equal to one, and n is the index of refraction of water. We can therefore rewrite this equation as follows:

$$\sin r = (1/n) \sin i$$

We will use the equation in this form when we apply Snell's law to the second refraction.

The Ray Program

We would like to write a program to draw a series of rays coming from the sun which enter the drop at various points on its surface. The sun's rays are coming from the right in Figure 17.4. We will characterize each incident ray by the perpendicular distance between the ray and the center of the circle. We will call this distance the impact parameter (D).

The procedure **RAINBOW** draws a circle of radius **RAD** to represent the rain drop. In **DRAW.RAY.AT :D** we draw the ray whose impact parameter is D. The procedure then checks to see if the impact parameter (D) is greater than the radius (**RAD**) of the drop. If not, it **START**s a **NEW.RAY**. When the ray first meets the drop it undergoes a **REFRACTION** then moves to **CROSS** the **DROP**, where it undergoes a **REFLECTION**. The ray then **CROSS**es the **DROP** once again and undergoes a second **REFRACTION** with the index of refraction (N) inverted since the ray is now moving from water to air rather than from air to water. In **EXIT.RAY** the ray emerges from the drop at a heading that is printed on the screen. The next ray is drawn with the impact parameter increased by **:DD**.

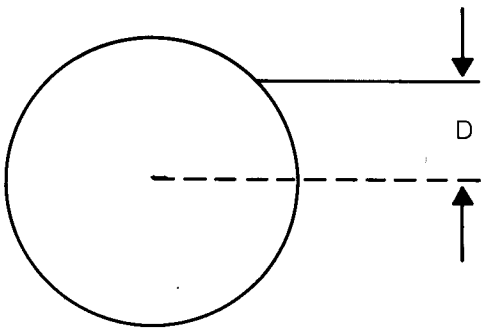


Figure 17.4 A Single Ray

```
TO RAINBOW :D :DD
CLEARSCREEN FULLSCREEN
MAKE "R 60
MAKE "N 1.33
WINDOW
CIRCLE :R
DRAW.RAY.AT :D
END

TO DRAW.RAY.AT :D
IF (ABS :D) > :R [STOP]
START.RAY.AT :D
REFRACTION :N
CROSS.DROP
REFLECTION
CROSS.DROP
REFRACTION 1 / :N
EXIT.RAY
PRINT HEADING - 90
DRAW.RAY.AT :D + :DD
END

TO START.RAY.AT :D
PU
SETPOS LIST (40 + SQRT ((SQ :R) - SQ :D)) :D
SETH 270
PD
FD 40
END

TO REFLECTION :N
FIND.ANGLE
MAKE "ANG ARCSIN ((SIN :ANG / :N)
SETH (HEADING + :ANG)
END

TO ARCSIN :X
OP ARCTAN (:X / (SQRT ((1 - SQ :X)))
END

TO FIND.ANGLE
MAKE "HEAD HEADING
SETH TOWARDS [0 0]
IF (ABS (:HEAD - HEADING)) > 90 [RT 180]
MAKE "ANG :HEAD - HEADING
END

TO REFLECTION
LT 180
```

```
MAKE "HEAD HEADING
SETH TOWARDS [0 0]
RT (HEADING - :HEAD)
END

TO CROSS.DROP
MAKE "L 2 * :R * COS :ANG
FD :L
END

TO EXIT.RAY
FD 160
END

TO START
RAINBOW 30 2
END
```

If this program is run by calling **RAINBOW 30 2** we see a series of rays emerging from the drop at various angles (see Figure 17.5). The important thing to notice is that there is a maximum deflection which occurs at about 42.5°. The angle of the exit rays increase up to a point and then begins to decrease. The ray with the greatest net deflection (42.5°) is called the *Cartesian ray* (after Rene Descartes, who first explained the physical basis for the rainbow). *Because of this reversal of the net deflection, there will be a greater density of rays emerging in the direction of the Cartesian ray.*

It may help to understand the enhancement in the intensity of the light in the Cartesian ray by means of the following analogy. Imagine that you are carrying a bucket of sand. There is a hole in the bucket and the sand is running out leaving a trail of sand on the ground behind you. If you were to slow down, stop, and back up, there would be more sand at the turn-around point than at any other point along the trail. You spend more time over this point than any other. This is precisely what happens in the rain drop. There is more light coming from the direction of the reversal in the deflection. If you look carefully at the exit rays you should be able to see this increase in intensity at the Cartesian angle.

If you were located below a collection of rain drops, you would notice a bright ray emitted from all drops in the direction of the Cartesian ray as seen in Figure 17.6. Other drops would be deflecting the sunlight toward the observer

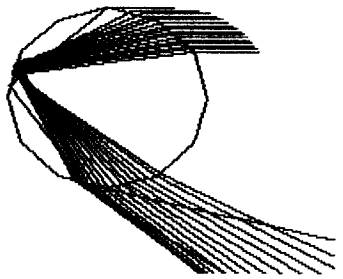


Figure 17.5 RAINBOW 30 2

as well, but the intensity of these rays would be much less than that of the Cartesian rays.

We have only shown in Figure 17.6 those rays and water drops that lie in the vertical plane. If you were to hold this figure upright you would have a representation of the vertical plane. Now grasp the figure with both hands at the two Xs marked in the figure. Rotate the page about a horizontal axis. The eye of the observer remains fixed as does the direction of the incident beam of sunlight. The Cartesian rays, however, rotate in an arc. This is the rainbow arc for the monochromatic red beam of light.

If the sun were high in the sky, the size of the rainbow is diminished. To see this, again hold the page by the Xs. Rotate the entire page to the left until the sunlight is at the appropriate angle. Now rotate about the line joining the Xs.

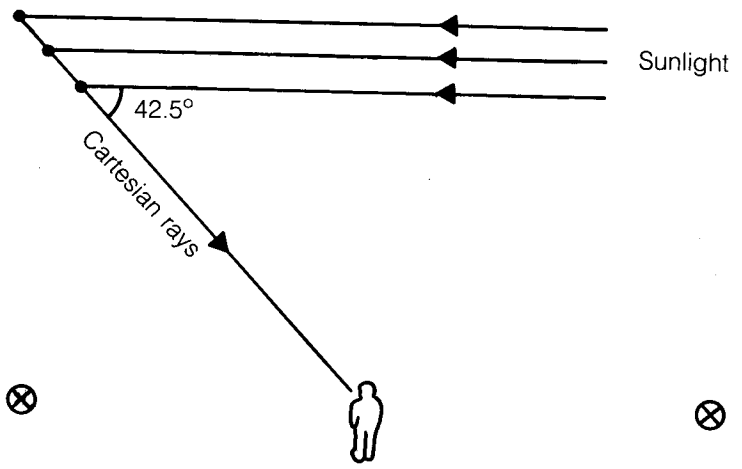


Figure 17.6 Enhancement of Cartesian Rays

Notice that the size of the rainbow arc is now diminished. Notice also that the angle between the observer, the rainbow, and the sun remains 42.5°.

The Color in the Rainbow

We have shown that an incident beam of monochromatic light will generate a bright arc at 42.5° relative to the incident beam. Now sunlight is not monochromatic. It scans the entire visible spectrum and much more. If we were to consider other frequencies we would find that the Cartesian rays emerge at different angles. This is due to the fact that the index of refraction depends on the frequency of the light. In **DRAW.RAY.AT** set **N = 1.34** and run **RAINBOW 30 2**. You will find that the Cartesian ray emerges at 41.06°.

The index of refraction increases with increasing frequency. Thus, the above experiment demonstrates that the angle of the Cartesian ray for high frequencies is less than the angle of the Cartesian ray for low frequencies. Blue light represents high frequencies and red light low frequencies, so the Cartesian blue ray emerges at a lower angle and the Cartesian red ray emerges at a high angle (see Figure 17.7). It is for this reason that we see red on the top of the rainbow and blue on the bottom (see Figure 17.8).

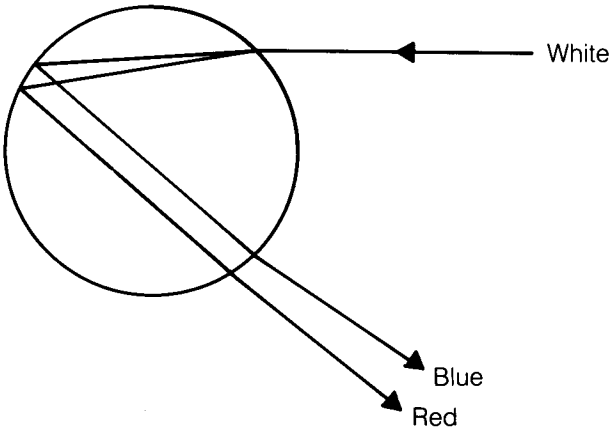


Figure 17.7 Red and Blue Rays

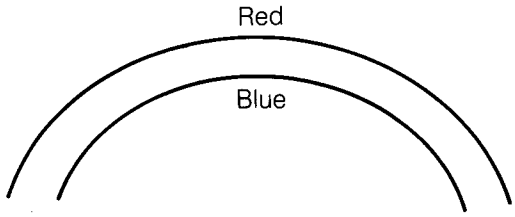


Figure 17.8 Red Appears at Top of Rainbow

A Mini-language

We have written the rainbow program in such a way that you may use the procedures as a small language; an optics language for spheres. The vocabulary of this language consists of five words:

```
START.RAY.AT :D
REFRACTION :N
REFLECTION
CROSS.DROP
EXIT.RAY
```

These five words may be called in the immediate mode or used in a procedure. For example, if you type **START.RAY.AT 30**, you will see a ray strike the drop with an impact parameter of 30. Next type **REFRACTION 1.33** and (if the turtle is visible) you will observe a change in the heading of the turtle. Next type **CROSS.DROP, REFRACTION 1 / 1.33**, and **EXIT.RAY** and you will see the ray pass through the drop and emerge from the far side. You have complete control of the course of the ray.

There are a variety of optics problems that you can solve using this mini-language. You may bundle the words together as we have done in the **DRAW.RAY.AT** procedure.

To understand the optics of a concave mirror consider the following modification:

```
DRAW.RAY.AT :D
IF (ABS :D > :R) [STOP]
START.RAY.AT :D
REFRACTION 1
CROSS.DROP
REFLECTION
CROSS.DROP
REFRACTION 1
EXIT.RAY
```

```
DRAW.RAY.AT :D + :DD
END
```

and call **RAINBOW 30 −60** to see Figure 17.9, which demonstrates that the focal point of a concave mirror is at about R/2 from the mirror. By using **REFRACTION 1** (index of refraction equal to one) we allowed the rays to pass through the sphere without deflection. In the problems that follow you may use similar techniques to find the focal point of a single refracting surface, a spherical lens, and a convex mirror.

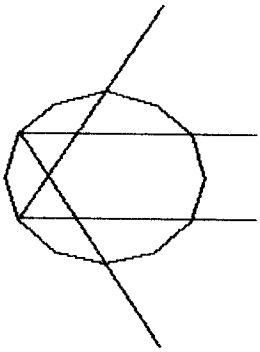


Figure 17.9 RAINBOW 30 −60

Projects

1. We have demonstrated the cause of the *primary* rainbow. There is also a *secondary* rainbow of lower intensity located above the primary at about 50°. The Cartesian ray which

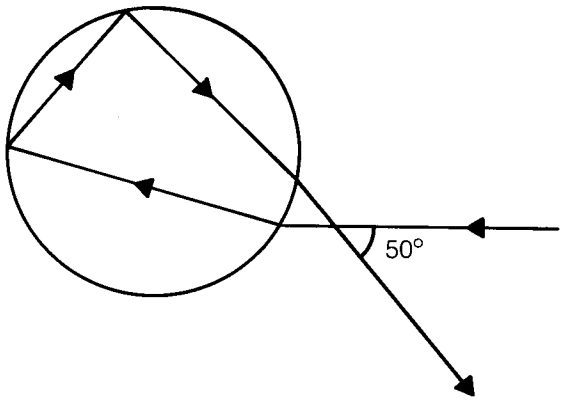


Figure 17.10 Cartesian Ray for the Secondary Rainbow

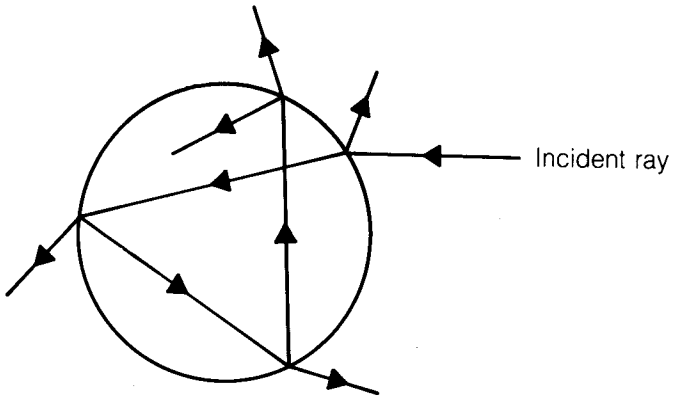


Figure 17.11 Refractions and Reflections from Incident Ray

is responsible for the secondary rainbow is reflected twice inside the drop and is illustrated in Figure 17.10. To find this ray, insert an additional **REFLECTION** and **CROSS.DROP** into **DRAW.RAY.AT** and try **RAINBOW −30 −1**. In the primary rainbow we see red on the top and blue on the bottom. Is this true for the secondary rainbow as well?

2. What is the angle of the tertiary rainbow?
3. A light ray that enters a water drop undergoes many internal reflections. With each reflected ray is a refracted ray (see Figure 17.11). Write a procedure that shows as many internal reflections as you choose to input. Show both the reflected and refracted rays.
4. Determine the focal point of a single refracting curved surface of radius 100. Let the lens be made of glass for which n = 1.5.
5. Determine the focal length of a spherical lens whose radius is 60 and index of refraction 1.5.
6. Determine the focal length of a convex mirror whose radius is 60.
7. As a more ambitious project, consider a converging thin lens of glass whose opposing faces have radii of 50 cm. Since it is a thin lens you may assume that both surfaces are on the y-axis (see Figure 17.12). Therefore to cross the lines you go **FORWARD 0**. To determine angles of incidence **SETHHEADING TOWARDS [−50 0]** and **[−50 0]**. The object is at a distance of 70 cm from the lens. Where is the image located?

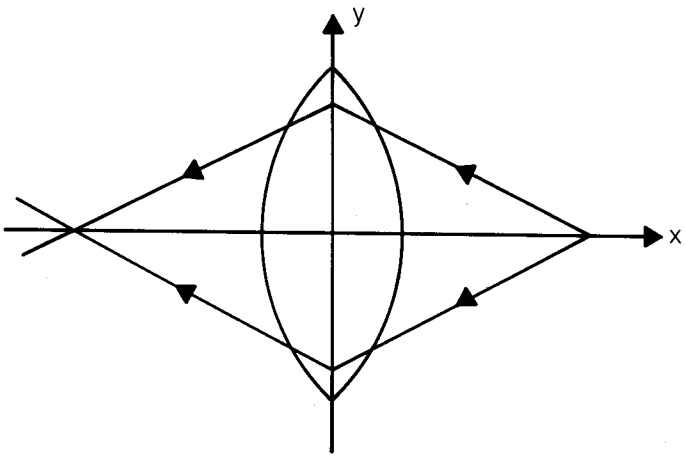
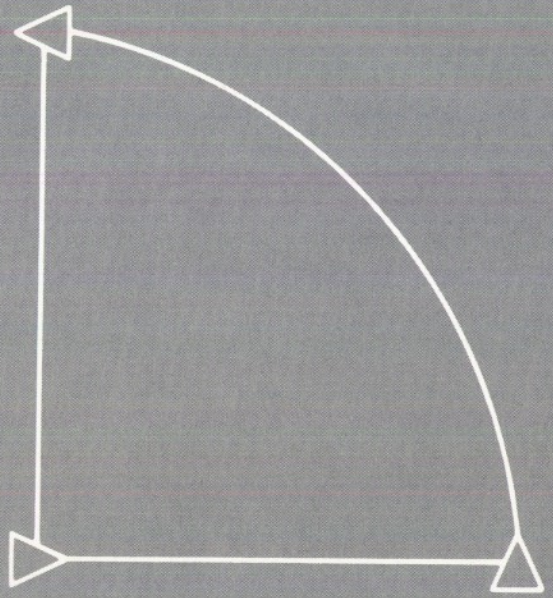


Figure 17.12 A Thin Lens

Problems

1. By examining all the rays that undergo one internal reflection and all rays that undergo two internal reflections, can you explain Alexander's dark band? (Alexander's dark band is the dark area that exists between the primary and secondary rainbows.)
2. We have shown for a single drop that the Cartesian ray emerges at 42.5°. Since raindrops come in many different sizes it is important to show that this result is independent of the radius of the drop. Set the radius to several different values and verify that the maximum deviation is independent of the radius.

chapter



18 Electric and Magnetic Field Lines

Introduction

Fields of one sort or another fill all of space. There are electric fields, magnetic fields, gravitational fields, and nuclear fields. We will investigate the nature of static electric and magnetic fields in this chapter.

The electric field of a single point charge of strength Q is given by the equation:

$$E = Q/r^2$$

where r is the distance between the charge and the field point. The electric field is a vector and so has a direction as well as a magnitude. The direction of the field of a point charge is directed radially away from a positive charge and radially toward a negative charge.

The static electric field of a group of point charges can be obtained by adding together the fields of the individual charges. For a large collection of charges this field can be quite complicated and so it is helpful to be able to visualize it in some way. This is accomplished with the aid of electric field lines. Electric field lines have the property of being

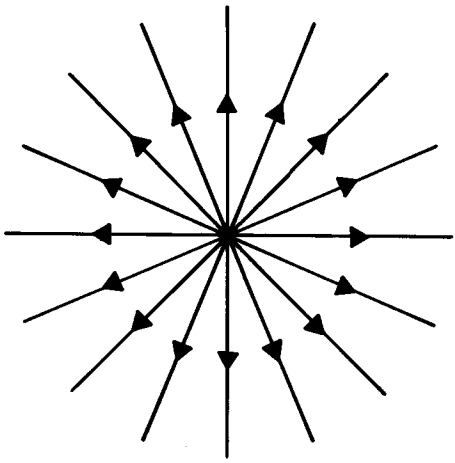


Figure 18.1 Electric Field Lines of a Single Positive Charge

everywhere tangent to the electric field. For example, the field lines of a positive point charge are illustrated in Figure 18.1. The electric field lines of two equal and opposite charges are shown in Figure 18.2.

Electric Field Lines

Let us consider a method by which the turtle might be employed to draw field lines. In order to be as definite as possible we will consider two charges of strength Q_1 and Q_2. Imagine the turtle to have somehow completed a por-

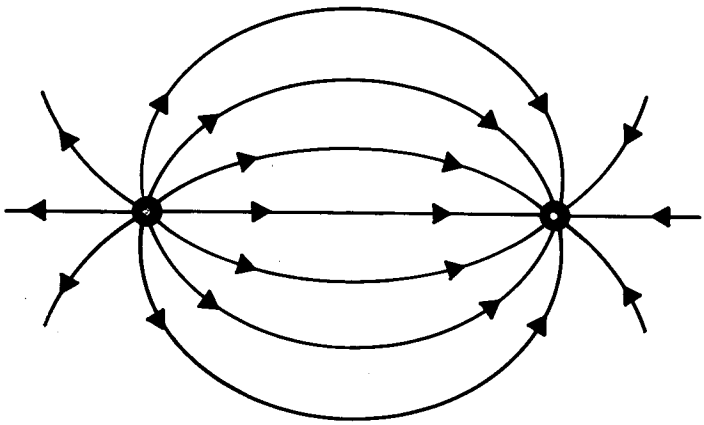


Figure 18.2 Field Lines of Equal and Opposite Charges

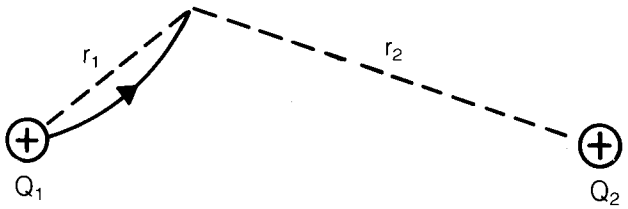


Figure 18.3 A Portion of an Electric Field Line

tion of one field line as illustrated in Figure 18.3. We would like to show how the turtle might add a small segment to this line. The first thing is to determine the direction of the electric field. Once we have this direction we simply extend the line a short distance at this heading and repeat the process over and over again.

To obtain the direction of the electric field we note the field is the vector sum of the fields produced by Q_1 and Q_2. These fields are directed radially away from the two charges and have magnitudes given by the equations:

$$E_1 = Q_1/(r_1)^2$$
$$E_2 = Q_2/(r_2)^2$$

where r_1 and r_2 are the distances between the charges and the field point (see Figure 18.4). We will use our tail-to-tip rule of vector addition to obtain the sum. The net field **E** is a vector from the tail of $\mathbf{E}_1$ to the tip of $\mathbf{E}_2$ as illustrated in Figure 18.5.

With this background we can now write a program to draw the electric field lines.

The Program

The electric field lines of any pair of charges may be drawn using the following program:

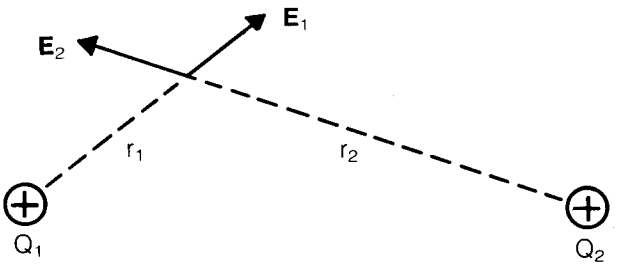


Figure 18.4 Electric Fields of Two Charges

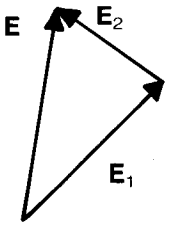


Figure 18.5 The Vector Sum of Two Fields

```
TO ELEC.FIELD :Q1 :X1 :Q2 :X2
WINDOW
HT FULLSCREEN CS
MAKE "POS.Q1 LIST :X1 0
MAKE "POS.Q2 LIST :X2 0
PU CHOOSE.LINE 0 330 :POS.Q1 10
IF ( :Q1 * :Q2 > 0 ) [CHOOSE.LINE 0 330 :POS.Q2 10]
 [CHOOSE.LINE ( 360 - :ANGLE + 20 ) ( -180 + :ANGLE
 - 30 ) :POS.Q2 -10]
END

TO CHOOSE.LINE :ANG :FINAL.ANG :Q.POS :LENGTH
PU
SETPOS :Q.POS
SETH :ANG
PD FD 10
STEP 0
IF :ANG > :FINAL.ANG [STOP]
CHOOSE.LINE :ANG + 20 :FINAL.ANG :Q.POS :LENGTH
END

TO STEP :N
FIND.FIELD.DIREC
FORWARD :LENGTH
IF ( OR ( ABS XCOR ) > 120 ( ABS YCOR ) > 120 )
 [STOP]
IF ( OR ( ABS :SQ.DIST.1 ) < 90 ( ABS :SQ.DIST.2 )
 < 90 ) [MAKE "ANGLE HEADING STOP]
STEP :N + 1
END

TO FIND.FIELD.DIREC
MAKE "OLD.POS POS
PU SETH TOWARDS :POS.Q1
MAKE "HEAD.1 HEADING
MAKE "SQ.DIST.1 SQ.DIST :POS.Q1
SETH TOWARDS :POS.Q2
MAKE "SQ.DIST.2 SQ.DIST :POS.Q2
BACK :Q2 / :SQ.DIST.2
```

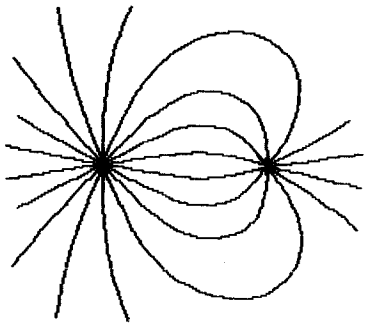
```
SETH :HEAD.1
BACK :Q1 / :SQ.DIST.1
SETH TOWARDS :OLD.POS
SETPOS :OLD.POS
PD RIGHT 180
END

TO SQ.DIST :POSITION
OP ( ( SQ ( ( FIRST POS ) - FIRST :POSITION ) ) +
 SQ ( LAST POS ) - LAST :POSITION )
END

TO START
ELEC.FIELD 20 -50 -10 50
END
```

The inputs to **ELEC.FIELD** are the charge and position of the two charges on the x-axis. For simplicity we always choose the greater charge to be positive and to the left of the second charge Q_2. The sign of Q_2 may be positive or negative. In the first call to **CHOOSE.LINE :ANG :FINAL.ANG :Q.POS :LENGTH** we begin to draw field lines which emanate from **:POS.Q1** at an angle of 0° and a step length of 10. (You may reduce this step length to obtain more accurate field lines.) The lines are drawn at 20° intervals until **:ANG > 330**. All lines continue until they near the boundary or terminate on the other charge. When the lines from **:Q1** have been completed a second call to **CHOOSE.LINE** will draw the lines which emanate from **:Q2**. If **:Q2** is opposite in sign from **:Q1**, some of the lines from **:Q1** will terminate on **:Q2**. Therefore, in this second call we must take care to choose only those headings away from **:Q2** which have not already been drawn. The angle at which the lines must begin is determined in **STEP** where **:ANGLE** is set to the heading at which the last field line terminated on **:Q2**.

The result of **ELEC.FIELD 20 −50 −10 50** is illustrated in Figure 18.6.

The Magnetic Dipole

Except for the region close to the dipole, the magnetic field of a magnetic dipole is very similar to the electric field of an electric dipole.



```
ELEC.FIELD 20 -50 -10 50
```

Figure 18.6 ELEC.FIELD 20 −50 −10 50

It is instructive to turn the magnetic dipole on its side as shown in Figure 18.7. These magnetic field lines are very similar to those generated in the Earth's core. By drawing a large sphere about the dipole, we can see the direction of the Earth's magnetic field at any latitude on the Earth's surface. Notice that the magnetic field is not parallel to the Earth's surface (except at the equator).

If we had a free-swinging compass, we would observe the true direction of the magnetic field. The usual compass is constrained to rotate in a horizontal plane and so we are unable to observe the vertical component. The angle between the magnetic field and the horizontal is called the *dip angle*. The dip angle is zero at the equator and 90° at the North Pole. In Washington D.C. the dip angle is 71° which may account for some of the strange goings on in the Capitol.

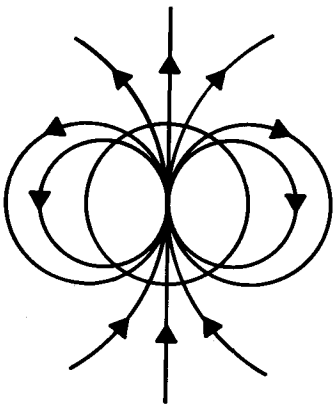


Figure 18.7 The Earth's Magnetic Dipole Field

Projects

1. Draw a dipole and a large sphere to illustrate the Earth's magnetic field.
2. Someplace on the line joining two positive charges the electric field must be zero. Run **ELEC.FIELD 2 −50 1 50**. Study the field and see if you can spot the place where the electric field must be zero.
3. Try to write a program that generates the field lines of three *positive* charges.
4. When the number of lines emanating from the charges is proportional to the quantity of charge one can show that the density of the field lines is proportional to the strength of the electric field. Modify **ELEC.FIELD** so that for two positive charges the number of lines emanating from each charge is proportional to the quantity of charge.
5. Construct a program that will draw arrows on the electric field line to indicate the direction of the field as in Figure 18.8. In this figure we have called **ELEC.FIELD 20 −50 −10 50** and the arrows were drawn according to the sign of the charges.
6. We have considered only the electric field of two charges. The objective of this project is to explore the field lines

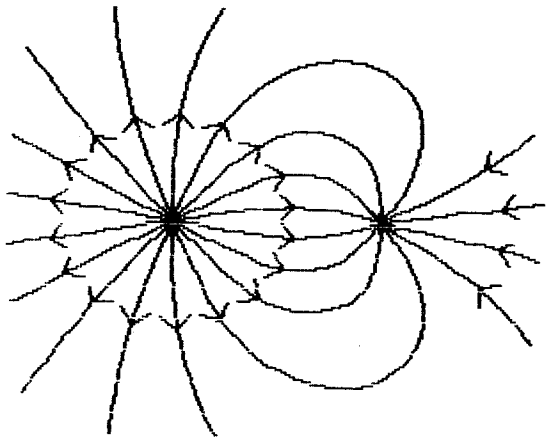


Figure Figure 18.8 Project 5

of *any* collection of charges. To simplify the task you might try to follow the steps outlined here:

a. Write a program which will execute the following procedure:

```
TO START
INITIALIZE
GO.TO 30 40
PLACE.CHARGE 10
GO.TO -20 80
DRAW.ELEC
END
```

where **INITIALIZE** allows you to initialize variables (for example, you may wish to place the position of the charge and its magnitude in a list **"CHARGE.LIST**—in this case **[30 40 10]**, where the position is (30,40) and the charge is 10. **GO.TO 30 40** sends the turtle to the point (30,40). **PLACE.CHARGE 10** places a charge of 10 at this point. You may wish to include a subprocedure to draw the charge (for example, draw an asterisk). **DRAW.ELEC** should draw an arrow in the direction of the electric field at the new turtle location of (−20,80). (Hint: To simplify parts (b) and (c) it is recommended that you use rectangular components for the positions and electric fields. You may then point the turtle in the direction of the electric field by **SETH TOWARDS LIST XCOR + EX YCOR + EY**, where **EX** and **EY** are the components of the electric field and **XCOR** and **YCOR** are the current location of the turtle.)

b. Repeat (a) but allow the turtle to place any number of charges. You may wish to keep track of the charges by constructing a **CHARGE.LIST**. For example, if **CHARGE.LIST** is **[[30 40 10] [10 90 −4]]**, there is a charge of 10 units at (30,40) and a charge of −4 at (10,90).

c. Repeat (b) except allow the turtle to move to any position in the field of preset charges and follow along the electric field line that passes through that point. For example, you should be able to run the following procedure:

```
TO START
FULLSCREEN
INITIALIZE
GO.TO 80 -80
PLACE.CHARGE 50
GO.TO -90 -90
```

```
PLACE.CHARGE 20
GO.TO 40 90
PLACE.CHARGE -10
GO.TO -80 -85
FOLLOW.FIELD.LINE
END
```

where **FOLLOW.FIELD.LINE** moves the turtle along the electric field line through the point $(-80,-85)$. (To stop the turtle you may try: **IF KEYP [STOP]**.) If you run this procedure you should obtain the field line illustrated in Figure 18.9.

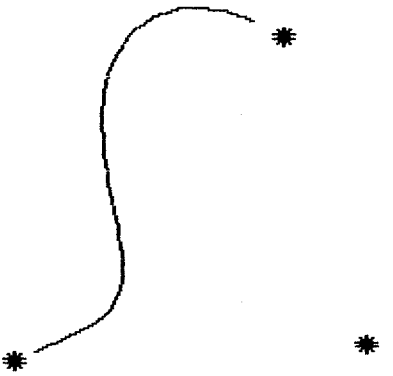


Figure Figure 18.9 Project 6

Appendix A

Astronomical Data

Physical Data for the Earth

Radius = 6.3×10^6 m
Mass = 6.0×10^{24} kg

Physical Data for the Sun

Radius = 6.9×10^8 m
Mass = 2.0×10^{30} kg

Physical Data for the Moon

Radius (Earth = 1) = .0123
Mass (Earth = 1) = .27
Surface Gravity (Earth = 1) = .16
Escape Velocity = 2,400 m/sec
Period = 27 days
Distance from Earth = 0.4×10^9 m

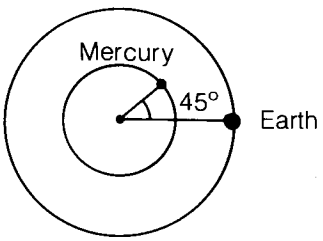
Physical Data for the Planets

	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune	Pluto
Dist. from sun (x 10^9 m)	58	108	150	228	778	1430	2900	4500	5900
Radius (Earth = 1)	.38	.97	1	.52	11	9.0	3.7	3.4	.45
Radius (x 10^3 m)	2.4	6.1	6.3	3.3	68	57	23	21	2.8
Surf. gravity (Earth = 1)	.38	.87	1.0	.39	2.6	1.2	1.1	1.2	.4
Escape Vel. (x 10^3 m/s)	4.3	10	11	5.1	61	37	22	26	5
Mass (Earth = 1)	.05	.81	1.0	.11	320	95	15	17	.1
Temperature (°K)	690	330	287	285	135	120	80	?	?
Length of year (yrs.)	.24	.61	1.0	1.9	12	29	84	165	248
Orbit vel. (× 10^3 m/s)	48	35	30	24	13	10	6.8	5.5	4.8

Planetary Positions

Position of the Planets on January 1, 1985

Mercury	45°
Venus	326°
Earth	0°
Mars	268°
Jupiter	194°
Saturn	130°
Uranus	156°
Neptune	172°
Pluto	113°



Appendix B

Terrapin and Commodore Logos

There are two dialects of Logo in common use. The first is represented by Apple, Atari, and IBM Logos; the second by Krell, Terrapin, and Commodore Logos. There are minor modifications within each of these two groups. It is the Apple/Atari/IBM Logo that is employed in the body of this book and these programs will run with little or no modification. The only additions necessary are a **TOWARDS** command for the Atari Logo and an **ASK** command for the Apple and IBM Logos. There will be other minor differences such as the use of abbreviations for certain commands but these differences should cause little difficulty.

The Krell, Terrapin, and Commodore Logos use a somewhat different syntax and for the benefit of these users we have included a translation of all major programs used in this book into Terrapin Logo. Modification to Krell and Commodore Logos will be minor. The one notable difference between the Terrapin and Commodore Logos is that

the Commodore Logo possesses sprites and so it is unnecessary to employ the **ASK** command presented below. Users of Commodore Logo should take advantage of their sprites (using the **TELL** command) and not force one turtle to do the work of four as is necessary in Apple, IBM, Terrapin, and Krell Logo.

ASK

Terrapin Logo does not like to use numbers as variables (although it is quite acceptable in Apple, Atari, and IBM Logos). It is not possible to **MAKE 1 "ANYTHING**. Since we wish to be able to **MAKE 1 [[0 0]0]** for example, it is necessary to resort to a subterfuge. Instead of the variable **1** we shall use the variable **WORD 1 "** . (Note the space after the quote.) This **WORD** command joins the 1 to the empty word and yields a word that Terrapin Logo will accept as a variable. For example, it is proper to **MAKE WORD 1 " "ANYTHING**. If we print out the names we see **"1** is **ANYTHING**. (While it would also be possible to **MAKE "1 "ANYTHING**, we would not be able to treat **"1** as both a variable and a number. This is a necessary feature of the **ASK** command as used in this book. Therefore the **ASK** command listed below is somewhat different from that included in *Notes to the User).*

```
TO HOME.ALL
LOCAL "N
MAKE "N "0
REPEAT 4 [MAKE :N [[0 0] 0] MAKE "N WORD :N + 1 " ]
END

TO ASK :N :CMD
MAKE "OLD.POS.HEADING LIST POS HEADING
PU SETPOS FIRST THING WORD :N "
SETHEADING LAST THING WORD :N "
PD
RUN :CMD
MAKE WORD :N " LIST POS HEADING
PU
SETPOS FIRST :OLD.POS.HEADING
SETH LAST :OLD.POS.HEADING
PD
END
```

```
TO SETPOS :LIST
SETXY FIRST :LIST LAST :LIST
END

TO POS
OP LIST XCOR YCOR
END
```

You will also notice in our **ASK** command the use of the two procedures **SETPOS** and **POS**. The use of the subprocedures is not necessary but makes the program a little more legible. You will find occasional use of **SETPOS** and **POS** in the listings below.

Note: The following program listings are complete with the exception of **HOME.ALL**, **ASK**, **SETPOS**, and **POS**. These subprocedures must be included where necessary.

```
TO VECTOR.ADDING.MACHINE
HOME
CS HT PD
DRAW.VECTORS
FIND.RESULTANT
PRINT [DO YOU WISH ANOTHER EXAMPLE? ( Y OR N )]
MAKE "ANSWER RC
IF :ANSWER = "Y VECTOR.ADDING.MACHINE
END

TO DRAW.VECTORS
PRINT [ENTER MAGNITUDE ( SPACE ) DIRECTION.]
MAKE "VEC RQ
IF :VEC = [] STOP
SETH LAST :VEC
FORWARD FIRST :VEC
DRAW.TIP
DRAW.VECTORS
END

TO FIND.RESULTANT
SETH TOWARDS 0 0
RIGHT 180
PRINT [RESULTANT VECTOR]
( PRINT [MAGNITUDE =] DISTANCE.FROM.HOME [HEADING
 =] HEADING )
MAKE "TIP POS
PU HOME
SETH TOWARDS FIRST :TIP LAST :TIP PD
SETPOS :TIP
DRAW.TIP
END
```

```
TO DRAW.TIP
RIGHT 25
BACK 15 FORWARD 15
LEFT 50
BACK 15 FORWARD 15
END

TO DISTANCE.FROM.HOME
OP SQRT ( SQ XCOR ) + SQ YCOR
END

TO SQ :N
OP :N * :N
END
```

```
TO EQUILIB :W :F.ANGLE :T.ANGLE
HOME
BACK :W
MAKE "OLD.POS POS
STEP 10
( PRINT [F =] DIST.BETWEEN POS :OLD.POS )
SETH :T.ANGLE
MAKE "T DIST.BETWEEN POS [0 0]
( PRINT [T =] :T )
SETH :T.ANGLE
FORWARD :T
END

TO STEP :S
IF :S < 0.01 STOP
SETH :F.ANGLE
FORWARD :S
SETH TOWARDS 0 0
IF HEADING < :T.ANGLE SETH :F.ANGLE BACK :S MAKE "S
 :S / 2
STEP :S
END

TO DIST.BETWEEN :P1 :P2
OP SQRT ( SQ ( FIRST :P1 ) - FIRST :P2 ) + SQ (
 LAST :P1 ) - LAST :P2
END

TO SQ :X
OP :X * :X
END
```

```
TO START
EQUILIB 60 45 270
END
```

```
TO FREE.FALL :HEIGHT
CS PD
DRAW.GROUND
PU
SETY :HEIGHT
MAKE "ACC 0.5
MAKE "VEL 0
MAKE "TIME 0
RIGHT 180
PD
STEP :VEL :TIME
PRINT "
PRINT [THE TURTLE HAS LANDED]
END

TO DRAW.GROUND
PU SETPOS [-100 0] PD
SETPOS [100 0]
HOME PU
END

TO STEP :VEL :TIME
( PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE (
 :HEIGHT - YCOR ) )
IF YCOR < 0 STOP
FORWARD :VEL
STEP :VEL + :ACC :TIME + 1
END

TO START
FREE.FALL 50
END
```

```
TO BETTER.FREE.FALL :HEIGHT
CS PD
DRAW.GROUND
PU
SETY :HEIGHT
MAKE "ACC 0.5
```

```
MAKE "VEL 0
MAKE "TIME 0
RIGHT 180
PD
STEP :VEL :TIME
PRINT "
PRINT [THE TURTLE HAS LANDED]
END

TO DRAW.GROUND
PU SETPOS [-100 0] PD
SETPOS [100 0]
HOME PU
END



TO STEP :VEL :TIME
( PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE (
 :HEIGHT - YCOR ) )
IF YCOR < 0 STOP
FORWARD :VEL + :ACC / 2
STEP :VEL + :ACC :TIME + 1
END

TO START
BETTER.FREE.FALL 50
END
```

```
TO BEST.FREE.FALL :HEIGHT
CS PD
DRAW.GROUND
PU
SETY :HEIGHT
MAKE "ACC 0.5
MAKE "VEL 0 + :ACC / 2
MAKE "TIME 0
RIGHT 180
PD
STEP :VEL :TIME
PRINT "
PRINT [THE TURTLE HAS LANDED]
END

TO DRAW.GROUND
PU SETPOS [-100 0] PD
```

```
SETPOS [100 0]
HOME PU
END

TO STEP :VEL :TIME
( PRINT "TIME :TIME "VELOCITY :VEL "DISTANCE (
 :HEIGHT - YCOR ) )
IF YCOR < 0 STOP
FORWARD :VEL
STEP :VEL + :ACC :TIME + 1
END

TO START
BEST.FREE.FALL 50
END
```

```
TO PROJECTILE :VELOCITY :ANGLE :GRAVITY
HT DRAW.GROUND
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
( PRINT [RANGE =] XCOR + 130 )
END

TO DRAW.GROUND
PU SETXY 130 ( - 50 )
PD SETXY - 130 ( - 50 )
END

TO STEP :VX :VY
SETH 90 FD :VX
SETH 0 FD :VY
IF YCOR < - 50 STOP
STEP :VX :VY - :GRAVITY
END

TO START
PROJECTILE 9 45 0.4
END
```

```
TO ACCURATE.PROJ :VELOCITY :ANGLE :GRAVITY
HT DRAW.GROUND
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * ( SIN :ANGLE ) - :GRAVITY / 2
```

```
STEP :VX :VY
( PRINT [RANGE =] XCOR + 130 )
END



TO DRAW.GROUND
PU SETXY 130 ( - 50 )
PD SETXY - 130 ( - 50 )
END

TO STEP :VX :VY
SETXY XCOR + :VX YCOR + :VY
IF YCOR < - 50 STOP
STEP :VX :VY - :GRAVITY
END

TO START
ACCURATE.PROJ 9 45 0.4
END
```

```
TO PROJECTILE.FRICTION :VELOCITY :ANGLE :GRAVITY
HT DRAW.GROUND
MAKE "FRICTION 0.1
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
( PRINT [RANGE =] XCOR + 130 )
END

TO DRAW.GROUND
PU SETXY 130 ( - 50 )
PD SETXY - 130 ( - 50 )
END

TO STEP :VX :VY
SETXY XCOR + :VX YCOR + :VY
IF YCOR < - 50 STOP
STEP ( :VX - :FRICTION * :VX ) ( :VY - :FRICTION *
 :VY - :GRAVITY )
END

TO START
PROJECTILE.FRICTION 20 30 0.4
END
```

```
TO CRT :VOLTAGE
MAKE "ACCELERATION 0.33 * :VOLTAGE
CLEARSCREEN
HT FULLSCREEN
DRAW.SCREEN
DRAW.PLATES
SETXY - 130 0
PD ST
STEP 4 0
SPLITSCREEN
PRINT [GO AGAIN ( Y / N ) ?]
IF RC = "Y THEN PRINT [WHAT VOLTAGE?] ELSE STOP
MAKE "V FIRST REQUEST
CRT :V
END

TO DRAW.SCREEN
PU SETXY 130 ( - 120 )
PD
SETH 0 FD 120 LT 90 FD 5 BK 5 RT 90 FD 120
PU
END

TO DRAW.PLATES
SETXY - 40 ( - 40 )
SETH 90
REPEAT 2 [PD FD 80 LT 90 PU FD 80 LT 90]
IF :VOLTAGE > 0 THEN DRAW.PLUS 0 50 DRAW.MINUS 0 (
 - 50 ) ELSE DRAW.PLUS 0 ( - 50 ) DRAW.MINUS 0 50
END


TO DRAW.PLUS :X :Y
SETXY :X :Y
PD SETH 0
FD 6 BK 12 PD 6
SETH 90
FD 6 BK 12 FD 6
PU
END

TO DRAW.MINUS :X :Y
SETXY :X :Y
PD FD 6 BK 12
PU
END

TO STEP :VX :VY
IF XCOR > 130 STOP
```

```
SETXY XCOR + :VX YCOR + :VY
STEP :VX :VY + ACCY
END

TO ACCY
IF ( ABS XCOR ) < 40 THEN OP :ACCELERATION ELSE OP
 0
END

TO ABS :NUM
IF ( :NUM < 0 ) THEN OP - :NUM ELSE OP :NUM
END
```

```
TO PROJECT.DT :VELOCITY :ANGLE :GRAVITY
HT
DRAW.GROUND
MAKE "DT 0.1
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
( PRINT [RANGE =] ( XCOR + 130 ) )
END

TO STEP :VX :VY
INC.XY :VX * :DT :VY * :DT
IF YCOR < - 50 STOP
STEP :VX :VY - :GRAVITY * :DT
END

TO DRAW.GROUND
PU SETXY 130 ( - 50 )
PD SETXY - 130 ( - 50 )
END

TO INC.XY :DX :DY
SETXY XCOR + :DX YCOR + :DY
END

TO START
PROJECT.DT 70 45 32
END
```

```
TO PROJECT.SCALE :VELOCITY :ANGLE :GRAVITY
HT
```

```
DRAW.GROUND
MAKE "DT 0.1
MAKE "SCALE 0.25
MAKE "VX :VELOCITY * COS :ANGLE
MAKE "VY :VELOCITY * SIN :ANGLE
STEP :VX :VY
SPLITSCREEN
( PRINT [RANGE =] ( XCOR + 130 ) / :SCALE )
END

TO STEP :VX :VY
INC.XY :VX * :DT * :SCALE :VY * :DT * :SCALE
IF YCOR < - 50 STOP
STEP :VX :VY - :GRAVITY * :DT
END

TO DRAW.GROUND
PU SETXY 130 ( - 50 )
PD SETXY - 130 ( - 50 )
END

TO INC.XY :DX :DY
SETXY XCOR + :DX YCOR + :DY
END

TO START
PROJECT.SCALE 70 45 32 * 0.16
END
```

```
TO ESCAPE :VEL :MASS
HOME WRAP
CS FULLSCREEN
MAKE "RADIUS 40
DRAW.EARTH :RADIUS
SETH 0
MAKE "Y :RADIUS
ST
STEP :VEL + ACC / 2
PENCOLOR 1
SPLITSCREEN PRINT [SPLAT!]
END

TO STEP :VEL
IF ALLOF :VEL < 0 :Y < 279 PENCOLOR 0
IF :Y < :RADIUS STOP
FD :VEL
MAKE "Y :Y + :VEL
```

```
STEP :VEL + ACC
END

TO ACC
OP - :MASS / ( :Y * :Y )
END

TO CIRCLE :RAD
MAKE "PI 3.14159
HT CS
PU FD :RAD RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :RAD / 12]
LT 90 - 15
END

TO DRAW.EARTH :RADIUS
CIRCLE :RADIUS
END

TO START
ESCAPE 6.6 1000
END
```

```
TO ESCAPE.SCALE :VELOCITY :MASS :RADIUS
HOME WRAP
CS FULLSCREEN
MAKE "G 6.67N11
MAKE "SCALE 50 / :RADIUS
DRAW.PLANET :RADIUS * :SCALE
SETH 0
MAKE "Y :RADIUS
MAKE "DT :RADIUS / ( 5 * :VELOCITY )
ST
STEP :VELOCITY + :DT * ACC / 2
PENCOLOR 1
SPLITSCREEN PRINT [SPLAT!]
END

TO DRAW.PLANET :R
CIRCLE :R
END

TO CIRCLE :RAD
MAKE "PI 3.14159
HT CS
PU FD :RAD RT 90 PD LT 15
```

```
REPEAT 12 [RT 30 FD 2 * :PI * :RAD / 12]
LT 90 - 15
END

TO STEP :VEL
IF ALLOF :VEL < 0 ( :Y < 279 / :SCALE + :RADIUS )
 PENCOLOR 0
IF :Y < :RADIUS STOP
FD :VEL * :DT * :SCALE
MAKE "Y :Y + :VEL * :DT
STEP :VEL + :DT * ACC
END



TO ACC
OP - :G * :MASS / ( :Y * :Y )
END

TO START
ESCAPE.SCALE 10300 5.99995E24 6300000
END
```

```
TO ORBIT :X :Y :SPEED :DIRECTION
CS HT
MAKE "MASS 4000
CIRCLE 20
PENUP
SETX :X SETY :Y
FIND.R
MAKE "VX ( :SPEED * SIN :DIRECTION ) + ACCX / 2
MAKE "VY ( :SPEED * COS :DIRECTION ) + ACCY / 2
FULLSCREEN PENDOWN
STEP :VX :VY
END

TO FIND.R
MAKE "R SQRT ( ( SQ XCOR ) + SQ YCOR )
MAKE "R3 :R * :R * :R
END

TO SQ :X
OP :X * :X
END

TO STEP :VX :VY
```

```
INC.XY :VX :VY
FIND.R
STEP :VX + ACCX :VY + ACCY
END

TO INC.XY :DX :DY
SETXY XCOR + :DX YCOR + :DY
END

TO ACCX
OP - :MASS * XCOR / :R3
END

TO ACCY
OP - :MASS * YCOR / :R3
END

TO CIRCLE :R
MAKE "PI 3.14159
HT PU HOME
FD :R RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :R / 12]
LT ( 90 - 15 )
END

TO CIRCULAR
ORBIT 50 0 8.9 0
END

TO ELLIPSE
ORBIT 120 0 5 0
END

TO COMET
ORBIT - 150 - 90 8 31
END
```

```
TO ORBIT.NEW :X :Y :SPEED :DIRECTION
HOME.ALL
CS
MAKE "MASS 4000
PENUP
SETXY :X :Y
ASK 1 [PENUP HT HOME PENDOWN SETH :DIRECTION FD
 :SPEED]
FULLSCREEN PENDOWN HT
```

```
STEP
END

TO STEP
SETH :DIRECTION
FD :SPEED
CHANGE.VEL
STEP
END

TO CHANGE.VEL
SETH TOWARDS 0 0
MAKE "ACC.ANGLE HEADING
MAKE "ACC :MASS / ( ( SQ XCOR ) + SQ YCOR )
ASK 1 [ADD.ACC]
END

TO ADD.ACC
SETH :ACC.ANGLE
FD :ACC
MAKE "SPEED SQRT ( SQ XCOR ) + SQ YCOR
SETH TOWARDS 0 0 RT 180
MAKE "DIRECTION HEADING
END

TO SQ :X
OP :X * :X
END

TO START
ORBIT.NEW 120 0 4 0
END
```

```
TO CIR.ORBIT :R
CS HOME
MAKE "X :R MAKE "Y 0
MAKE "DIRECTION 0
MAKE "TIME 0
MAKE "RAD.OF.SUN 6.91995E8
MAKE "MASS.OF.SUN 2E30
MAKE "G 6.67N11
MAKE "SCALE 119 / MARS
MAKE "SPEED SQRT :G * :MASS.OF.SUN / :R
MAKE "DT 8 / ( :SPEED * :SCALE )
CIRCLE :RAD.OF.SUN * :SCALE
PU SETX :R * :SCALE
```

```
FIND.R
MAKE "VX ( :SPEED * SIN :DIRECTION ) + :DT * ACCX /
 2
MAKE "VY ( :SPEED * COS :DIRECTION ) + :DT * ACCY /
 2
FULLSCREEN PD
STEP :VX :VY
SPLITSCREEN PRINT :TIME / ( 24 * 60 * 60 )
END

TO STEP :VX :VY
IF RC? STOP
MAKE "X :X + :VX * :DT
MAKE "Y :Y + :VY * :DT
SETXY ( :X * :SCALE ) ( :Y * :SCALE )
MAKE "TIME :TIME + :DT
FIND.R
STEP :VX + ACCX * :DT :VY + ACCY * :DT
END

TO FIND.R
MAKE "R SQRT ( SQ :X ) + SQ :Y
MAKE "R3 :R * :R * :R
END

TO SQ :X
OP :X * :X
END

TO ACCX
OP - :G * :MASS.OF.SUN * :X / :R3
END

TO ACCY
OP - :G * :MASS.OF.SUN * :Y / :R3
END

TO CIRCLE :R
MAKE "PI 3.14159
HT PU HOME
FD :R RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :R / 12]
LT ( 90 - 15 )
END

TO MERCURY
OP 5.79997E10
END
```

```
TO VENUS
OP 1.08E11
END

TO EARTH
OP 1.5E11
END

TO MARS
OP 2.3E11
END

TO JUPITER
OP 7.78E11
END

TO SATURN
OP 1.43E12
END

TO URANUS
OP 2.87E12
END

TO NEPTUNE
OP 4.49996E12
END

TO PLUTO
OP 5.89995E12
END
```

```
TO PLANETS :R0 :R1 :R2 :R3
HOME.ALL
CS FULLSCREEN ST
MAKE "PI 3.14159
MAKE "R ( SE :R0 :R1 :R2 :R3 )
MAKE "TURTLE.NO [0 1 2 3]
MAKE "MASS.OF.SUN 2E30
MAKE "G 6.67N11
MAKE "SCALE 119 / MAX :R
MAKE "V []
MAKE "ANGLE []
SET.VELOCITIES :R
MAKE "DT 8 / ( ( MAX :V ) * :SCALE )
```

```
SET.PLACE :R :TURTLE.NO
SET.ANGLE :R :V
PD HT
DRAW.ORBITS :V :TURTLE.NO
END

TO SET.VELOCITIES :R
IF EMPTY? :R THEN STOP
MAKE "V LPUT SQRT ( :G * :MASS.OF.SUN / FIRST :R )
 :V
SET.VELOCITIES BF :R
END

TO SET.PLACE :R :TURTLE.NO
IF EMPTY? :R THEN STOP
ASK FIRST :TURTLE.NO [PU SETX :SCALE * FIRST :R PD]
SET.PLACE BF :R BF :TURTLE.NO
END

TO SET.ANGLE :R :V
IF EMPTY? :R THEN STOP
MAKE "ANGLE LPUT ( 360 * :DT * FIRST :V ) / ( 2 *
 :PI * FIRST :R ) :ANGLE
SET.ANGLE BF :R BF :V
END

TO DRAW.ORBITS :V :TURTLE.NO
TURN : TURTLE.NO :ANGLE
STEP :V :TURTLE.NO
DRAW.ORBITS :V :TURTLE.NO
END

TO TURN :TURTLE.NO :ANG
IF EMPTY? :TURTLE.NO THEN STOP
ASK FIRST :TURTLE.NO [LEFT FIRST :ANG]
TURN BF :TURTLE.NO BF :ANG
END

TO STEP :V :TURTLE.NO
IF EMPTY? :TURTLE.NO THEN STOP
ASK FIRST :TURTLE.NO [FD ( FIRST :V ) * :DT *
 :SCALE]
STEP BF :V BF :TURTLE.NO
END

TO MERCURY
OP 5.79997E10
END
```

```
TO VENUS
OP 1.08E11
END

TO EARTH
OP 1.5E11
END

TO MARS
OP 2.3E11
END

TO JUPITER
OP 7.78E11
END

TO SATURN
OP 1.43E12
END

TO URANUS
OP 2.87E11
END

TO NEPTUNE
OP 4.49996E12
END

TO PLUTO
OP 5.89995E12
END

TO START
PLANETS MERCURY VENUS EARTH MARS
END
```

```
TO VOYAGER :X0 :Y0 :V0 :ANG0 :X1 :V1
HOME.ALL
MAKE "MASS 8000
CS PU
SETXY :X0 :Y0
ASK 1 [PU SETXY :X1 0 PD]
FIND.R
MAKE "VX ( :V0 * SIN :ANG0 ) + ACCX / 2
MAKE "VY ( :V0 * COS :ANG0 ) + ACCY / 2
FULLSCREEN PD HT
```

```
STEP :VX :VY :X1
END

TO STEP :VX :VY :X1
PD
SETXY :X0 :Y0
ASK 1 [SETXY :X1 0]
MAKE "X0 :X0 + :VX
MAKE "Y0 :Y0 + :VY
FIND.R
( PRINT [SPEED =] SQRT ( ( SQ :VX ) + SQ :VY ) )
STEP :VX + ACCX :VY + ACCY :X1 + V1
END

TO FIND.R
MAKE "R SQRT ( ( SQ ( :X0 - :X1 ) ) + SQ :Y0 )
MAKE "R3 :R * :R * :R
END

TO ACCX
OP - :MASS * ( :X0 - :X1 ) / :R3
END

TO ACCY
OP - :MASS * :Y0 / :R3
END

TO SQ :X
OP :X * :X
END

TO START
VOYAGER - 120 ( - 110 ) 10 60 120 ( - 8 )
END
```

```
TO ROCKET :VR :M :MM
WRAP
SETXY 0 0
HOME RIGHT 90
STEP 0
END

TO STEP :V
FORWARD :V
STEP :V + :M * :VR / :MM
END
```

```
TO START
ROCKET 10 5 1000
END
```

```
TO JET :VR :M :MM
WRAP
HOME RIGHT 90
STEP 0
END

TO STEP :V
FORWARD :V
STEP :V + :M * ( :VR - :V ) / :MM
END

TO START
JET 10 5 1000
END
```

```
TO ROCKET.AND.JET :VR :M :MM
WRAP PU HT
HOME.ALL
ASK 0 [SETXY 0 8 RIGHT 90]
ASK 1 [SETXY 0 ( - 8 ) RT 90]
PD
MAKE "TIME 0 MAKE "DT 1
MAKE "SCALE 10
STEP 0 0
END

TO STEP :V0 :V1
ASK 0 [FORWARD :V0]
ASK 1 [FORWARD :V1]
ASK 2 [SETXY ( :TIME - 120 ) ( :V0 * :SCALE )]
ASK 3 [SETXY ( :TIME - 120 ) ( :V1 * :SCALE )]
MAKE "TIME :TIME + :DT
( PRINT "ROCKET :V0 " "JET :V1 )
STEP ( :V0 + :M * :VR / :MM ) ( :V1 + :M * ( :VR -
 :V1 ) / :MM )
END

TO START
ROCKET.AND.JET 5 5 300
END
```

```
TO OSC :AMP
MAKE "W 0.5
PU SETXY 0 :AMP
PD
STEP 0
END

TO STEP :VEL
FORWARD :VEL
STEP :VEL - :W * :W * YCOR
END

TO START
OSC 50
END
```

```
TO OSC.TIME :AMP
WRAP
MAKE "W 0.1
MAKE "DT 1
PU SETXY 0 :AMP
HT PD
STEP 0
END

TO STEP :VEL
FORWARD :VEL
RIGHT 90
FORWARD :DT
LEFT 90
STEP :VEL - :W * :W * YCOR
END

TO START
OSC.TIME 50
END
```

```
TO PRED :RABBITS :FOXES
CS PU HT
MAKE "A 0.04
MAKE "B 0.04
MAKE "C 4N4
MAKE "D 4N4
MAKE "R :RABBITS
```

```
MAKE "F :FOXES
PD
DRAW.AXES
SETXY :R - 100 :F - 100
PD FULLSCREEN
STEP :R :F
END

TO DRAW.AXES
PU FULLSCREEN
SETXY - 100 ( - 100 )
SETH 90
PD FORWARD 200
DRAW.ARROW
PU FORWARD 15
WRITE.R
SETXY - 100 ( - 100 )
SETH 0 PD FORWARD 200
DRAW.ARROW
PU FORWARD 10
WRITE.F
END

TO DRAW.ARROW
RT 30 BK 10 FD 10 LT 60 BK 10
FD 10 RT 30
PU
END

TO WRITE.R
PD SETH 0
FD 10 RT 90 FD 5 RT 90 FD 5 RT 90 FD 5
RT 50 BK 8 PU
END

TO WRITE.F
SETH 0
PD FD 10 RT 90 FD 8 BK 8
RT 90 FD 5 LT 90 FD 5 PU
END

TO STEP :R :F
SETXY :R - 100 :F - 100
MAKE "DR ( :A * :R ) - ( :C * :R * :F )
MAKE "DF ( - :B * :F ) +( :D * :R * :F)
STEP :R + :DR :F + :DF
END

TO START
PRED 40 80
END
```

```
TO BIG.BANG
MAKE "V0 PICK.RAN
MAKE "V1 PICK.RAN
MAKE "V2 PICK.RAN
MAKE "V3 PICK.RAN
MAKE "V.LIST ( SE :V0 :V1 :V2 :V3 )
MAKE "V 0
CLEARSCREEN PU
SET.AT.START 0
HT
STEP
END

TO STEP
ASK 0 [FD ( :V0 - :V )]
ASK 1 [FD ( :V1 - :V )]
ASK 2 [FD ( :V2 - :V )]
ASK 3 [FD ( :V3 - :V )]
IF RC? THEN MAKE "V ITEM ( 1 + FIRST RC ) :V.LIST
STEP
END

TO PICK.RAN
OP ( 1 + RANDOM 100 ) / 2
END

TO SET.AT.START :N
IF :N > 3 STOP
MAKE WORD " :N [[-120 0] 90]
SET.AT.START :N + 1
END

TO ASK :N :CMD
PU SETPOS FIRST THING WORD " :N
SETHEADING LAST THING WORD " :N
DRAW.TURTLE [PENCOLOR 0]
PD
RUN :CMD
DRAW.TURTLE [PENCOLOR 1]
MAKE WORD " :N LIST POS HEADING
PU
PD
END

TO DRAW.TURTLE :CMD
PD
RUN :CMD
```

```
LT 45
REPEAT 4 [FD 5 RT 90]
RT 45
END

TO HOME.ALL
LOCAL "N
MAKE "N 0
REPEAT 4 [MAKE :N [[0 0] 0] MAKE "N THING WORD " :N
 + 1]
END
```

```
TO RAD.DECAY
HOME.ALL
CLEARSCREEN HT
PRINT [PROBABILITY OF DECAY IS ONE CHANCE IN ( ? )]
MAKE "CHANCE FIRST REQUEST
MAKE "NUMBER 30
MAKE "TIME 0
MAKE "N.DECAY 0
MAKE "N.SCALE 4 MAKE "T.SCALE 6
MAKE "X.BOX ( - 130 ) MAKE "Y.BOX ( - 50 )
MAKE "SIZE 4 MAKE "SEPARATION 9
BOXES :X.BOX :Y.BOX :SIZE :SEPARATION :NUMBER
BOXES :X.BOX :Y.BOX + :SIZE + 1 0.75 * :SIZE
 :SEPARATION :NUMBER
MAKE.STATE :NUMBER
DRAW.AXES
ASK 1 [HT PU SETPOS LIST :X.BOX :NUMBER * N.SCALE
 PD]
ASK 2 [HT PU SETPOS LIST :X.BOX :NUMBER * N.SCALE
 PD]
STEP :NUMBER
END

TO STEP :N
CYCLE 0
MAKE "TIME :TIME + 1
ASK 1 [SETXY ( :X.BOX + :T.SCALE * :TIME ) (
 :N.SCALE * ( :NUMBER - :N.DECAY ) )]
ASK 2 [SETXY ( :X.BOX + :T.SCALE * :TIME ) ( YCOR -
 ( YCOR / :CHANCE ) )]
( PRINT [TIME =] :TIME [NO.DECAYED =] :N.DECAY )
STEP :N
END
```

```
TO MAKE.STATE :N
MAKE "STATE []
REPEAT :N [MAKE "STATE FPUT "R :STATE]
END

TO DRAW.AXES
PU SETXY :X.BOX :N.SCALE * :NUMBER
PD SETXY :X.BOX 0
RT 90 FD 300
END

TO CYCLE :N
IF :N = :NUMBER THEN STOP
IF ALLOF ( FIRST :STATE ) = "R ( RANDOM :CHANCE ) =
 0 THEN MAKE "STATE ( LPUT "S BF :STATE ) ( EMIT :N
 ) ELSE MAKE "STATE ( LPUT ( FIRST :STATE ) ( BF
 :STATE ) )
CYCLE :N + 1
END

TO EMIT :N
BOX ( :X.BOX + :N * :SEPARATION ) ( :Y.BOX + :SIZE
 + 1 ) ( 0.75 * :SIZE ) [PD PENCOLOR 0]
PENCOLOR 1
PD SETH ( - 30 + RANDOM 61 ) FD 40 PU
MAKE "N.DECAY :N.DECAY + 1
END

TO BOXES :X :Y :SIZE :SEP :N
IF :N = 0 THEN STOP
BOX :X :Y :SIZE [PENDOWN]
BOXES :X + :SEP :Y :SIZE :SEP :N - 1
END

TO BOX :X :Y :SIZE :CMD
SET 0
PU SETXY ( :X - :SIZE / 2 ) ( :Y - :SIZE / 2 )
RUN :CMD
REPEAT 4 [FD :SIZE RT 90]
END

TO START
RAD.DECAY
END
```

```
TO HALF.LIFE
MAKE "P GET.P
```

```
DRAW.AXES
SETXY 0 0 PD
STEP 0 100
END

TO GET.P
PRINT [DECAY PROBABILITY = ?]
OP FIRST REQUEST
END

TO STEP :TIME :N
SETXY XCOR + 1 :N
IF :N < 50 THEN ( PRINT [HALF LIFE =] :TIME ) STOP
STEP :TIME + 1 :N - :P * :N
END
```

```
TO BRIDGE :W
CS PU HOME
MAKE "DEN :W / 200
MAKE "HOR 50
MAKE "DX 10
MAKE "Y0 ( - 50 )
MAKE "INC.SLOPE :DEN * :DX / :HOR
SETXY 0 :Y0
PD STEP 0 1 1
STEP 0 1 ( - 1 )
END

TO STEP :SLOPE :COUNTER :SIGN
SETXY ( XCOR + :DX * :SIGN ) ( YCOR + :DX * :SLOPE
 )
MAKE "HEIGHT YCOR - :Y0
BK :HEIGHT FD :HEIGHT
IF :COUNTER = 10 THEN BK :HEIGHT SETX 0 STOP
STEP :SLOPE + :INC.SLOPE :COUNTER + 1 :SIGN
END

TO START
BRIDGE 200
END
```

```
TO CAT :W
PU
MAKE "DEN :W / 200
```

```
MAKE "HOR 50
MAKE "DX 10
MAKE "YO ( - 50 )
SETXY 0 :YO
PD
STEP.CAT 0 1 1
STEP CAT 0 1 ( - 1 )
END

TO STEP.CAT :SLOPE :COUNTER :SIGN
SETXY ( XCOR + :DX * :SIGN ) ( YCOR + :DX * :SLOPE
 )
IF :COUNTER = 10 THEN PU SETXY 0 :YO PD STOP
MAKE "DS SQRT ( SQ :DX ) + SQ :DX * :SLOPE
MAKE "INC.SLOPE :DEN * :DS / :HOR
STEP.CAT :SLOPE + :INC.SLOPE :COUNTER + 1 :SIGN
END

TO START
CAT 200
END
```

```
TO ARCHER :DEPTH :R
CS PU HT HOME
MAKE "N 1.33
DRAW.WATER
SETY - :DEPTH
SETH :R
PD FD DIST.TO.WATER
FD 100 BK 100
SETH I
FD 100
END

TO DRAW.WATER
PD RT 90 FD 100 BK 200 HOME PU
END

TO DIST.TO.WATER
OP :DEPTH / COS :R
END

TO I
MAKE "SINE.I :N * SIN :R
OP ATAN :SINE.I SQRT ( 1 - SQ :SINE.I )
END
```

```
TO START
ARCHER 70 40
END
```

```
TO APPARENT.DEPTH :DEPTH
ARCHER.NEW :DEPTH 20
ARCHER.NEW :DEPTH ( - 20 )
( PRINT [APPARENT DEPTH =] ( - YCOR ) )
( PRINT [ACTUAL DEPTH =] :DEPTH )
END

TO ARCHER.NEW :DEPTH :I
PU HOME HT
MAKE "N 1.33
DRAW.WATER
SETY - :DEPTH
SETH :I
PD FD DIST.TO.WATER
SETH R
FD 100 BK 100
BK DIST.TO.Y.AXIS
END

TO DRAW WATER
PD RT 90 FD 100 BK 200 HOME PU
END

TO DIST.TO.WATER
OP :DEPTH / COS :I
END

TO R
MAKE "SINE.R :N * SIN :I
OP ATAN :SINE.R SQRT ( 1 - SQ :SINE.R )
END

TO DIST.TO.Y.AXIS
OP :DEPTH * ( TAN :I ) / SIN R
END

TO TAN :X
OP ( SIN :X ) / COS :X
END

TO SQ :X
OP :X * :X
END
```

```
TO START
APPARENT.DEPTH 70
END
```

```
TO MAG :HEIGHT :DEPTH :ANG.RAY
HT CS HOME
MAKE "EYE LIST 0 :HEIGHT
DRAW.WATER
MAKE "N 1.33
SETPOS :EYE
PD
DRAW.RAY :ANG.RAY
PU SETPOS :EYE PD
DRAW.RAY - :ANG.RAY
( PRINT [ANGLE IN AIR =] HEADING )
( PRINT [ANGLE IN WATER =] :ANG.RAY )
( PRINT [MAGNIFICATION =] :ANG.RAY / HEADING )
END

TO DRAW.WATER
PD RT 90 FD 100 BK 200 HOME PU
END

TO DRAW.RAY :ANG
SETH ( 180 - :ANG )
FD DIST.TO.WATER
SETH ( 180 - I )
FD DIST.TO.FISH
SETH TOWARD FIRST :EYE LAST :EYE
END

TO DIST.TO.WATER
OP :HEIGHT / COS I
END

TO DIST.TO.FISH
OP :DEPTH / ABS COS I
END

TO I
MAKE "SINE.I ( SIN :ANG ) / :N
OP ATAN :SINE.I SQRT ( 1 - SQ :SINE.I )
END

TO ABS :X
IF :X > 0 THEN OP :X ELSE OP - :X
END
```

```
TO START
MAG 100 60 20
END
```

```
TO RAINBOW :D :DD
CLEARSCREEN FULLSCREEN
MAKE "R 60
MAKE "N 1.33
CIRCLE :R
DRAW.RAY.AT :D
END

TO DRAW.RAY.AT :D
IF ( ABS :D ) > :R STOP
START.RAY.AT :D
REFRACTION :N
CROSS.DROP
REFLECTION
CROSS.DROP
REFRACTION 1 / :N
EXIT.RAY
PRINT HEADING - 90
DRAW.RAY.AT :D + :DD
END

TO START.RAY.AT :D
PU
SETXY ( 40 + SQRT ( ( SQ :R ) - SQ :D ) ) :D
SETH 270
PD
FD 40
END

TO REFRACTION :N
SNELL :N
END

TO SNELL :N
FIND.ANGLE
MAKE "ANG ARCSIN ( ( SIN :ANG ) / :N )
SETH ( HEADING + :ANG )
END

TO FIND.ANGLE
MAKE "HEAD HEADING
SETH TOWARDS 0 0
IF ( ABS ( :HEAD - HEADING ) ) > 90 THEN RT 180
```

```
MAKE "ANG :HEAD - HEADING
END

TO REFLECTION
LT 180
MAKE "HEAD HEADING
SETH TOWARDS 0 0
RT ( HEADING - :HEAD )
END

TO CROSS.DROP
MAKE "L 2 * :R * COS :ANG
FD :L
END

TO CIRCLE :RAD
MAKE "PI 3.14159
HT PU HOME
FD :RAD RT 90 PD LT 15
REPEAT 12 [RT 30 FD 2 * :PI * :RAD / 12]
LT 75
END

TO START
RAINBOW 30 2
END
```

```
TO ELEC.FIELD :Q1 :X1 :Q2 :X2
HT FULLSCREEN CS
MAKE "POS.Q1 LIST :X1 0
MAKE "POS.Q2 LIST :X2 0
PU CHOOSE.LINE 0 330 :POS.Q1 10
IF ( :Q1 * :Q2 > 0 ) THEN CHOOSE.LINE 0 330 :POS.Q2
 10 ELSE CHOOSE.LINE ( 360 - :ANGLE + 20 ) ( - 180
 + :ANGLE - 30 ) :POS.Q2 ( - 10 )
END

TO CHOOSE.LINE :ANG :FINAL.ANG :Q.POS :LENGTH
PU
SETPOS :Q.POS
SETH :ANG
PD FD 10
STEP 0
IF :ANG > :FINAL.ANG STOP
CHOOSE.LINE :ANG + 20 :FINAL.ANG :Q.POS :LENGTH
END
```

```
TO STEP :N
FIND.FIELD.DIREC
FORWARD :LENGTH
IF ANYOF ( ABS XCOR ) > 110 ( ABS YCOR ) > 110 THEN
 STOP
IF ANYOF ( ABS :SQ.DIST.1 ) < 90 ( ABS :SQ.DIST.2 )
 < 90 THEN MAKE "ANGLE HEADING STOP
STEP :N + 1
END

TO FIND.FIELD.DIREC
MAKE "OLD.POS POS
PU SETH TOWARDS FIRST :POS.Q1 LAST :POS.Q1
MAKE "HEAD.1 HEADING
MAKE "SQ.DIST.1 SQ.DIST :POS.Q1
SETH TOWARDS FIRST :POS.Q2 LAST :POS.Q2
MAKE "SQ.DIST.2 SQ.DIST :POS.Q2
BACK :Q2 / :SQ.DIST.1
SETH TOWARDS FIRST :OLD.POS LAST :OLD.POS
SETPOS :OLD.POS
PD RIGHT 180
END

TO SQ.DIST :POSITION
OP ( ( SQ ( ( FIRST POS ) - FIRST :POSITION ) ) +
 SQ ( LAST POS ) - LAST :POSITION )
END

TO START
ELEC.FIELD 20 ( - 50 ) ( - 10 ) 50
END
```

Index

Also available from Holt Rinehart and Winston:

- Allen et al THINKING ABOUT TLC LOGO (1984)
- Bull et al NUDGES: APPLE LOGO PROJECTS (1985)
- Burrowes et al EXPLORING IBM LOGO: A GUIDE FOR ADULTS (1985)
- Tipps et al NUDGES: IBM LOGO PROJECTS (1984)
- Tobias et al BEYOND MINDSTORMS: TEACHING WITH IBM LOGO (1985)

ISBN 0-03-002913-9