MIT/LCS/TR-347

A Relative-Motion Microworld

by

Linda E. Morecroft

September 1985

Copyright (C) Massachusetts Institute of Technology 1985

This research was partially supported by the Defense Advanced Research Projects Agency of the Department of Defense monitored by the Office of Naval Research under contract number NOOO1483-K0125.

This blank page was inserted to preserve pagination.

A Relative-Motion Microworld

by

Linda Elizabeth Morecroft
B.Sc. (Hons.) University of Bristol, England (1972)

Submitted in partial fulfillment
of the requirements for the
degree of

Master of Science

at the

Massachusetts Institute of Technology

September 1985

Copyright (C) Massachusetts Institute of Technology 1985

Signature of Author . . Linda E. Morecroft . . . . . .
Department of Electrical Engineering and Computer Science
September 12, 1985

Certified by . . . Harold Abelson . . . . . .
Professor Harold Abelson
Thesis Supervisor

Accepted by . . . . . .
Professor Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

A Relative-Motion Microworld

by

Linda Elizabeth Morecroft

Submitted to the
Department of Electrical Engineering and Computer Science
on September 12, 1985 in partial fulfillment of the requirements
for the Degree of Master of Science

Abstract

A relative-motion microworld has been designed to aid high-school students in understanding the concepts of relative motion and frames of reference.

Relative motion and frames of reference are usually introduced in a high-school physics or mathematics course. Most students, and many teachers too, have difficulty understanding the concepts and applying them to solve problems. The traditional approach to relative motion uses vector algebra. However, vector terminology is complex and it does not allow a mental picture of what is happening to be easily built up. Students do not understand what it means to be in a different frame of reference and how moving objects appear within this reference frame. Most people have a much more intuitive approach to motion problems.

The relative-motion microworld uses a new representation for thinking about motion, based on the intrinsic, local and procedural characteristics of turtle geometry. This representation provides the student with a simple framework for understanding motion and relative motion, that overcomes the problems associated with the traditional approach. The microworld allows students to set up their own motion problems, in order to view the motion from different reference frames. The new representation, together with the dynamic simulation, provide a very powerful tool for thinking about motion and understanding the motion seen from a chosen reference frame.

Four key primitives have been designed: these make an object, give an object a motion, select a reference frame, and move all the objects that have been set up. These four primitives are the building blocks that can be used to set up complex problems. For example, how does a stationary observer see a penny fall to the ground when dropped by a running person? How does the running person see the penny fall to the ground? In a few lines of code students implement this problem and use the simulation and new

representation to analyze the motion.

Problems from a variety of motion situations can be explored using the relative-motion microworld. These include rectilinear motion, non-parallel rectilinear motion, motion under gravity, and rotational motion.

A student text and teacher's reference manual accompany the microworld. The text leads the student through a discussion of relative motion and frames of reference. It introduces the new representation for motion and describes how the microworld models relative motion. The text outlines various areas of investigation: parallel and non-parallel rectilinear motion, motion under gravity and circular motion. Each section includes examples and exercises. The exercises range in complexity and include ones to implement with the microworld and ones to think through. The teacher's reference manual gives hints on solving the exercises and solutions to the exercises.

Thesis Supervisor: Professor Harold Abelson
Title: Professor of Electrical Engineering and Computer Science
Laboratory for Computer Science
Massachusetts Institute of Technology

Keywords: Computer-based learning, Logo, microworld, physics, reference frames, relative motion.

Acknowledgments

I would like to take this opportunity to thank those people who have been closely involved in both my education at M.I.T. and with the preparation of this thesis.

I extend my sincere thanks to my advisor, Harold Abelson, who during the past two years has given freely of his time and has provided an encouraging, challenging environment in which to work. His prompt feedback is greatly appreciated. I also thank Andrea diSessa and Gerald Sussman for their advice and enthusiasm.

Thanks also to David Gifford for the loan of computer equipment; to Robert I. Hulsizer and Irwin A. Pless, from the Department of Physics, for their guidance; to fellow graduate students Dennis Lee and Jim Miller for providing motivation and support; and to my officemates Michael Eisenberg and Henry Lieberman for their good humor, friendship and stimulating conversation.

Besides my M.I.T. colleagues I thank Joyce Tobias, from the Brookline Public Schools, for coordinating my field work; Virginia Flook, from Brookline High School, for allowing me to spend time with her senior physics class and for her encouragement, enthusiasm and comments on my work. I thank the students from the Brookline High School who provided me with valuable feedback on the computer software and student text. In addition, I owe thanks to Sarah McCann and Julie McClennen, from the Belmont High School, who gave freely of their time to work with the software and student text during the initial phases of this project.

I thank Leigh Klotz, from Terrapin, Inc., for taking time from his heavy workload to rewrite a section of my coding in assembler language. The speed with which he accomplished this task is truly amazing.

Finally, my gratitude and love to my family. They have provided me with warmth, guidance, support, motivation and most importantly, love. Thanks to my parents, Bill and

Margaret Greenwood, for numerous trips to Boston to help out; to my sister, Jennifer Ristok, for her quiet motivation and friendship; to my daughters, Katherine and Jennifer, for providing a link with reality; and to my husband, John, for his constant support, encouragement, advice and sympathetic ear.

Table of Contents

Chapter One: Introduction 10

1.1 Overview of the Concepts 11
1.2 Objectives of the Thesis 12
1.3 Overview of the Thesis 13

Chapter Two: Logo and Turtle Geometry 15

2.1 Turtle Geometry 19
2.2 Representing Motion 20
2.3 Representing Relative Motion 22
2.4 Extending Logo to Model Relative Motion 26

Chapter Three: Relative Motion 27

3.1 Current Teaching Methods 27
3.1.1 A Traditional Presentation 28
3.1.2 Problems with the Traditional Method 29
3.2 Common Misconceptions in Relative Motion 31
3.2.1 Understanding Wrong Intuitions 34
3.3 Overcoming the Problems of the Traditional Method 37

Chapter Four: The Relative-Motion Microworld 47

4.1 Implementation of the Microworld 47
4.1.1 Primitives 48
4.1.2 System Implementation 53
4.2 Explorations Using the Microworld 56
4.2.1 Linear Motion 57
4.2.2 Non-parallel Motion 58
4.2.3 Motion Under Gravity 59
4.2.4 Circular Motion 62
4.2.4.1 The Solar System 64
4.2.4.2 Examining Circular Motion 66
4.2.4.3 Rotating Reference Frames 67
4.3 The Student Text 69
4.4 The Teacher's Reference Manual 72

Chapter Five: Using the Microworld 73

5.1 Working With Students 74
5.1.1 Tutorial Groups I 74

5.1.2 Tutorial Group II 82
5.1.3 The High-School Class 85
5.2 The Screening Test 87
5.3 The Student Questionnaire 89
5.4 Summary 90

Chapter Six: Conclusion **95**

Appendix A: The Student Text **99**

Appendix B: The Teacher's Reference Manual **178**

Appendix C: The Microworld Coding **214**

C.1 The Apple II Logo Implementation 215
C.2 The Apple II Assembler Code 231
C.3 The Assembler Implementation 237

References **241**

Table of Figures

Figure 2-1: Drawing with the turtle 18
Figure 2-2: A and B after one interval of time 23
Figure 2-3: A and B after two intervals of time 24
Figure 2-4: The motion of B as seen by A 24
Figure 3-1: The velocity of P relative to Q 28
Figure 3-2: A simple relative motion problem 29
Figure 3-3: The method of vector difference 30
Figure 3-4: Suggested paths of a penny in falling to the ground 32
Figure 3-5: A penny dropped by a moving person 33
Figure 3-6: Motion in terms of velocity vectors 38
Figure 3-7: Motion in terms of paths 39
Figure 3-8: Some positions of CIRCLE1 and CIRCLE2 40
Figure 3-9: Building the path of CIRCLE2 as seen by CIRCLE1 41
Figure 3-10: The path of CIRCLE2 as seen by CIRCLE1 42
Figure 3-11: Velocity vectors for CIRCLE1 and CIRCLE2 43
Figure 3-12: Relative velocity diagrams for CIRCLE2 44
Figure 3-13: The path of the relative velocity vectors 45
Figure 3-14: Adding in positional vectors 46
Figure 4-1: A procedure to simulate A and B moving 51
Figure 4-2: Setting up an object to move in a circle 63
Figure 4-3: Retrograde motion 65
Figure 4-4: A track with one epicycle loop 67
Figure 4-5: A one cusp track 68

Table of Tables

Table 4-1: The microworld primitives 49
Table 5-1: Percentage of questions answered correctly 91

Chapter One

Introduction

Relative motion and frames of reference are usually included in a high-school physics or mathematics course. Many students have difficulty in understanding the concepts of relative motion and applying them to solve problems. Teachers too, often experience difficulty with these topics. In everyday occurrences of motion, students are often not aware that motion viewed from a moving frame of reference, can look completely different. Many students are unable to visualize what form the motion takes. In addition, students often incorrectly interpret motion due to being in a moving reference frame.

This thesis explores how relative motion can be taught in a more intuitive way. It takes advantage of currently available computer technology in designing a learning environment to enhance the teaching of relative motion. The learning environment is embedded within the Logo programming language. Logo is a unique learning environment itself, designed to allow exploration. The work described here illustrates how we can incorporate the principles of physics into Logo to create a novel and dynamic learning environment. The relative-motion system uses the principles of turtle geometry as a foundation for describing motion and relative motion. Turtle geometry, a feature of Logo, is a new representation for geometry based on an intrinsic, local and procedural method for describing geometric figures. Logo is extended to provide a set of primitive commands for setting up motion and for specifying a reference frame. The student uses these commands to set up and view a variety of different motion scenarios.

A student text accompanies the relative-motion system. This text is a stand-alone text that leads the student through discussions and examples of frames of reference and relative motion. The text introduces the new representation for motion, based on turtle geometry, and it shows how we can use this representation to model relative motion. In addition, the text gives numerous examples and suggestions for independent exploration. A teacher's

reference manual gives solutions to the exercises.

1.1 Overview of the Concepts

The development of a comprehensive learning environment is central to this work. To help students reason about motion the environment must provide an understandable representation for motion. The computer simulation must reinforce this representation. In addition, the computer system must be flexible, easy to operate and provide students with a vehicle for learning.

The following concepts are of paramount importance in developing the relative-motion learning environment:

* Turtle geometry can be used as a foundation for describing a new representation for motion. This motion is based on the intrinsic, local and procedural characteristics of turtle geometry.

* An object can be given a procedural motion, which is performed continuously over discrete intervals of time. The object's overall motion is the path it moves on over several intervals of time.

* An object's relative motion is its apparent path seen from a chosen object over discrete intervals of time. This apparent path is the path relative to a reference frame object. Reference is only made to the intrinsic coordinate system of the reference frame object, not to a globally defined coordinate system.

* A frame of reference is a *stationary* object with all other objects moving *relative* to it.

* An object is a unique entity with a set of properties describing it. These properties include ones found directly in turtle geometry, such as heading and position.

* A set of primitive commands can be designed to incorporate these concepts into Logo. Logo is both procedural and extensible, and it therefore provides a solid base for extending the language to include new concepts. This provides the student with a unique and flexible environment for initiating and studying motion.

1.2 Objectives of the Thesis

The objectives of this work are:

* To introduce students to the concept of relative motion and to provide them with a robust representation for motion that can be used over a range of scenarios.

* To make students aware that motion seen from different reference frames can look entirely different.

* To debug students' intuitive notions about motion.

* To help students gain a better understanding and recognition of observable phenomena.

In a traditional treatment of physics, students usually become competent in a subject area by rigorously solving sets of problems. This method aspires to give the student a "feel" for the subject. The traditional curriculum usually includes problems that are known to be solvable, rather than those that involve independent inquiry to determine if a solution can be realized, or a method viable. The computer-based microworld facilitates an open-ended "learning by doing" approach to the learning environment. Students often best learn technical concepts by using them to explore scientific models and to formulate and test their ideas. The microworld developed here provides a framework within which the student can explore the relevant concepts of relative motion, together with sets of examples to initiate the student's exploration. A student text accompanies the microworld. As the student

works through the text the following questions are answered:

* What is a frame of reference?

* What are the general principles involved in changing from one reference frame to another?

* How do you determine what frame of reference you are in?

* What is meant by the term "relative velocity"?

* How do moving objects look when viewed from different reference frames, including moving reference frames?

In addition to answering the above questions, the student text outlines several areas for the student to explore: parallel rectilinear motion, non-parallel motion, motion under gravity and circular motion. The section on circular motion also includes the effects due to the rotating earth. The earlier exercises allow the student to become competent using the computer system and to obtain a thorough grasp of the concepts. The later sections challenge the student to use these concepts in exploring motion under gravity and rotational motion.

1.3 Overview of the Thesis

Logo is used as a framework for implementing a learning environment for relative motion. Chapter Two discusses why Logo is suited to this task and the unique characteristics of turtle geometry that provide a good representation for motion and relative motion.

Chapter Three outlines a traditional presentation of relative motion and the problems that students face using this representation. The traditional representation uses vector algebra. This static representation does not provide the student with an intuitive notion

about motion. Students build up knowledge over time from experience and formal learning. Chapter Three discusses why some students have difficulty using their knowledge in dealing with motion problems and where their wrong intuitions originate from. A final section of Chapter Three shows how the new representation for motion, based on turtle geometry, can give the student an intuitive representation for motion to solve relative motion problems.

Chapter Four describes the relative-motion microworld. This chapter outlines in detail the primitive commands we need to initiate motion on the computer and how the system is implemented. In addition, Chapter Four describes the student text and teacher's reference manual, which accompany the relative-motion microworld, and it gives some suggestions for possible explorations using the microworld.

The relative-motion microworld has been used in small tutorial groups and in a classroom setting of twenty-two students. Chapter Five describes the students' work using the microworld and monitoring the students' performance.

Chapter Six summarizes the work and suggests possible extensions and modifications to the relative-motion microworld.

The appendices contain the student text, the teacher's reference manual and the coding of the microworld.

Chapter Two

Logo and Turtle Geometry

The relative-motion microworld is embedded within the Logo programming language. Logo was developed during the 1970's at the Logo Laboratory, a joint endeavor of M.I.T.'s Artificial Intelligence Laboratory and the Division for Study and Research in Education. A student does not need to know how to program in Logo to use the relative-motion system. Very few commands are needed to set up motion simulations. Logo is procedural, recursive, interactive, extensible and user-friendly. These characteristics provide an ideal framework for designing and implementing the relative-motion microworld.

Logo is a *procedural* language. When you undertake a Logo project you usually do not write it as one long program. Instead, you break it down into small pieces and write a separate *procedure* for each piece. The notion of procedures is a powerful problem-solving strategy. You break the problem down into small pieces, each of which has a well-defined solution. In this modular way you can attempt and solve very complex problems. A procedure may use another procedure, a *subprocedure*, to do part of its work. Consider the following example; we define a procedure, SQUARE, to draw a square and then use SQUARE as a subprocedure in our definition of WINDOW.

```
TO SQUARE
  REPEAT 4 [FORWARD 60  RIGHT 90]
END

TO WINDOW
  REPEAT 4 [SQUARE  RIGHT 90]
END
```

In addition to being procedural, Logo is *recursive*. This means that a procedure can

call *itself* as a subprocedure. *Recursion* allows you to state complicated problems in a compact form. For example, finding the factorial of a number, N. The solution to this problem is:

```
N!  =  N * (N - 1) * (N - 2) * (N - 3)   ...   * 3 * 2 * 1
```

We can express this solution *recursively* as:

```
N!  =  N * (N - 1)!
```

and give as a terminating condition, N! = 1, when N = 0. This can be written in Logo using only two lines of code:

```
TO FACTORIAL  :N
  IF :N = 0 OUTPUT 1
  OUTPUT :N * FACTORIAL (:N - 1)
END
```

Logo is an *interactive language.* This means that you can type in a command to be carried out *immediately.* For example,

```
PRINT FACTORIAL 3
```

will immediately return 6. There is no need to make a separate file containing your command, compile the program, then run it to receive your answer. Interactive languages are very important in education as students receive an immediate response to their commands.

All computer languages have built-in operations or primitives. Many have the arithmetic operations: +, -, * and /. Logo is a procedural language which is also *extensible*; it allows you to create new operations defined in terms of its built-in primitives and user-defined procedures. You can give these new primitives names and then use them as commands, in effect extending the language. In this way you can incorporate new ideas into Logo. The relative-motion system uses this feature; four basic commands have been

implemented to set up and simulate motion. This straightforward way of extending the language allows you to concentrate on the embedded principles and not on programming itself. For example, diSessa [diSessa 80] in his work with the Dynaturtle, extended Logo to embody the principles of Newtonian mechanics. The Dynaturtle is a turtle that obeys Newton's laws of motion. Using the Dynaturtle you can experiment in a world governed by Newtonian mechanics yet free from the usual restraints of friction and air resistance.

Students who are not familiar with programming are often intimidated by the responses they receive from the computer system. Logo is *user-friendly* in that it responds with helpful messages. For example, if you type:

```
PRINT FACTORIAL
```

Logo responds with a message similar to:

```
not enough inputs to FACTORIAL
```

This means that the procedure FACTORIAL needs an input. If you type:

```
PRINT FACTORILA 3
```

Logo responds with:

```
I don't know how to FACTORILA
```

which means that you are trying to use a procedure, FACTORILA, which you have not defined.

Turtle Graphics is one feature of Logo. The *turtle* is a graphics cursor that responds to a few simple commands. For example, the command **FORWARD** moves the turtle in the direction it is facing, by a given number of units. If you type **FORWARD 100** the turtle moves forwards 100 units. **RIGHT** turns the turtle clockwise by a given number of degrees. **RIGHT 90** will turn the turtle through 90 degrees to the right. A turtle can leave a trace of

where it has been. If a turtle changes its position it can leave a trace as it moves to this new position. This is controlled by the commands **PENUP** and **PENDOWN**. When the turtle's pen is down the turtle draws lines as it moves. For example, Figure 2-1 shows drawing on the computer screen with the turtle.

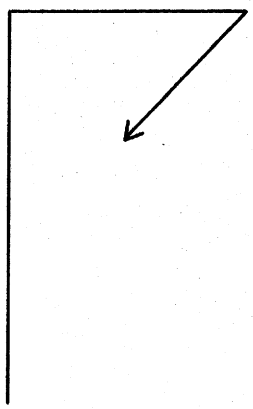

Figure 2-1: Drawing with the turtle

The commands to produce Figure 2-1 are:

```
PENDOWN
FORWARD 50
RIGHT 90
FORWARD 30
RIGHT 135
FORWARD 20
```

Turtle graphics with its visual display has great appeal among children. Children can immediately see a response to their commands. Logo is now widely used as a language to introduce the computer to children and is used from kindergarten to university level [Abelson 80]. It is a language for a broad range of ages and abilities. For example, very young children have great difficulty in typing correctly and quickly. To overcome this problem teachers developed a "one finger" Logo called *Instant Turtle*, which allows a child

to move the turtle using one letter input commands, e.g. **F** for **FORWARD** and **B** for **BACK**. At the other end of the educational spectrum Abelson and diSessa [Abelson 80] use the turtle to explore curved space and time, which leads to a discussion of the General Theory of Relativity.

2.1 Turtle Geometry

Turtle geometry is a new way of thinking about geometry. You can use turtle commands to draw geometric figures on the computer screen. A square of side 50 units is made using the command:

```
REPEAT 4 [FORWARD 50  RIGHT 90]
```

Turtle geometry is the geometry of using turtle commands such as **FORWARD** and **RIGHT** to describe geometric shapes. Coordinate geometry is another representation. Coordinate geometry uses the Cartesian coordinate system to define figures. There are some fundamental differences between coordinate geometry and turtle geometry. Turtle geometry is intrinsic, local and procedural. Coordinate geometry is extrinsic and global and it uses equations to describe figures. This thesis uses the properties of turtle geometry as a foundation for describing motion.

Turtle geometry is intrinsic. This means that the figures we draw do not depend on any external properties. For example, the command to draw a square, will draw the square, with four equal, perpendicular sides. The orientation of the square depends only on the turtle's initial heading. Drawing a square using the Cartesian coordinate system means referring to an external reference frame to define the vertical and horizontal directions. Coordinate geometry is therefore extrinsic.

Turtle geometry is local. To draw a circle we must use a command similar to:

```
REPEAT 360 [FORWARD 1  RIGHT 1]
```

We can choose different values as input to **FORWARD** and **RIGHT** and still draw a circle; for example, [FORWARD 5 RIGHT 10]. This description is local, as the turtle only concentrates on drawing a small piece of the figure at a time. Each piece is drawn in an identical way.

Coordinate geometry is global. To draw a circle we must use the equation:

$x^2 + y^2 = r^2$,

which defines the circle as being equidistant from a central point. In addition, if we change the coefficients of this equation, we will get a different figure.

$x^2 + 2y^2 = r^2$,

will not draw a circle, but will draw an ellipse.

Another important property of turtle geometry is that it is procedural. Turtle geometry defines figures in terms of procedures. We can define the procedure for a circle as:

```
TO CIRCLE  :SIDE  :ANGLE
  REPEAT  360/:ANGLE  [FORWARD :SIDE  RIGHT :ANGLE]
END
```

The procedure, CIRCLE, takes two inputs: the amount to move forward on each step and the angle to turn through on each step. Coordinate geometry uses equations to define figures. A procedural definition is more flexible as it is easily modified and it can take full advantage of the procedural capabilities of the Logo language.

2.2 Representing Motion

A clear method of representation is necessary if students are to understand concepts and be able to use them in dealing with a range of problems. This is particularly important when students are faced with problems that do not precisely fit their textbook method of solution. In situations such as these, students often use their intuitive notions, which may be incorrect. A clear representation is therefore vital for students to understand motion

problems across a broad spectrum of situations.

The traditional method of solving relative motion problems is using vector algebra. This representation is sufficient to solve all problems. However, most people have problems with this representation and cannot deal effectively with relative motion. We will discuss the traditional approach and outline some of the problems presented by this approach in Chapter Three. In this section we will see how turtle geometry can be used to represent motion and in the following section, detail how we can extend this to model relative motion.

A turtle has a state given by its position and heading. We use state-change operators to change the turtle's state. If we give the turtle the command FORWARD 3, the turtle moves forwards 3 units, on its original heading. The turtle's position is changed. The turtle has a new state given by its new position and its original heading. RIGHT is a state-change operator that effects the turtle's heading. RIGHT 60 turns the turtle clockwise by 60 degrees.

If we give the turtle the command FORWARD 3 and then repeat the command, the turtle moves forwards 3 units and then moves forwards an additional 3 units. If we constantly repeat the command FORWARD 3, the turtle keeps moving forwards in steps of 3 units. When we view this on the computer screen, the turtle appears to be moving constantly. We can think of the turtle as moving with a speed of 3 units. Similarly, if we repeatedly give the turtle the command FORWARD 5, the turtle moves forwards in steps of 5 units. The turtle appears to be moving with a speed of 5 units.

When we represent motion using turtle geometry we give the turtle a motion to be performed during a discrete interval of time, such as FORWARD 3. This motion is constantly repeated to show how an object moves. This representation of motion allows us to take advantage of the intrinsic, local and procedural characteristics of turtle geometry.

The representation is intrinsic. We tell the turtle how to move during each interval of time. The turtle's path depends only on the specified motion, and the turtle's original position and heading. We do not need to refer to an external, global frame of reference.

We are now defining motion as motion in discrete intervals of time. This representation is local as we build up the motion by considering what happens during small time intervals. In each time interval the motion is identical. The turtle is only concerned about its current state during each time interval and how this state is modified by the state-change operators which make up the motion.

Defining motion in this way allows us to take advantage of the procedural nature of turtle geometry. In the examples outlined above the procedural motion we gave to a turtle was in the form of a simple, primitive command. Instead, we could define a procedure, and give this to the turtle to be repeated during each interval of time. For example, if we represent an object falling under gravity, we can give it a motion of FALL. In this case, FALL is a procedure which increments the amount to move forwards, on each interval of time, by a constant amount. In this way we represent the acceleration due to gravity and we see the turtle move faster and faster over time.

2.3 Representing Relative Motion

The previous section outlined how we can conceptualize motion in discrete amounts and use turtle geometry to represent this motion. We will now extend this representation to model relative motion.

Relative motion is motion seen from a specified reference frame. An object may appear to move in very different ways seen from different reference frames. In one reference frame the object may have a circular motion and in another, a straight-line motion. An important principle of relative motion is that the reference frame considers itself as stationary, with everything else doing the moving. We may be in a moving frame of reference and when we see objects move we see their apparent path. The apparent path is the path with respect to the reference frame. We can build up a picture of how an object appears to move: we let each object move during a given amount of time, and then determine each object's position *relative* to the reference frame. If we repeat this over successive intervals of time we gradually build up the path of an object seen from the

reference frame.

Let us consider an example to illustrate this. Suppose two people, A and B, are moving at right angles to one another. A is moving North away from a chosen point at 3 m.p.h. and B is moving East, away from the same point at 4 m.p.h. How does A see B move? In other words, what is the *relative motion* of B from A? To answer this, we look at the motion of each person over consecutive intervals of time. During one time interval A moves 3 units to position X, as shown in Figure 2-2. B moves 4 units to position Y. At the end of this time interval, if A looks towards B, A sees B at position Y. If the distance XY is measured, B is a distance of 5 units away from A, on a heading of 127 degrees.

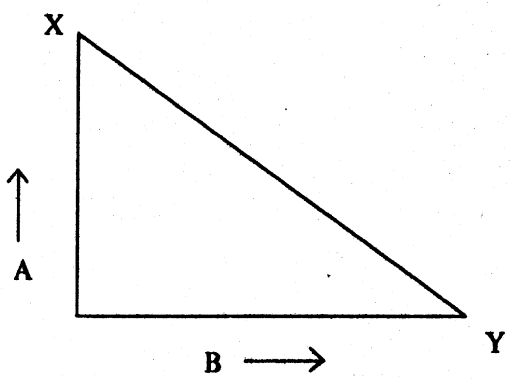

Figure 2-2: A and B after one interval of time

What happens after another interval of time? A moves a further 3 units North to position R and B moves a further 4 units East to position S, as shown in Figure 2-3. From A, B now appears to be 10 units away, on a heading of 127 degrees. This method can be repeated many times to build up a picture of how A sees B move. In this example, for every interval of time, B becomes 5 units further away from A. Therefore, B is moving *relative* to A at 5 m.p.h., on a heading of 127 degrees, as shown in Figure 2-4.

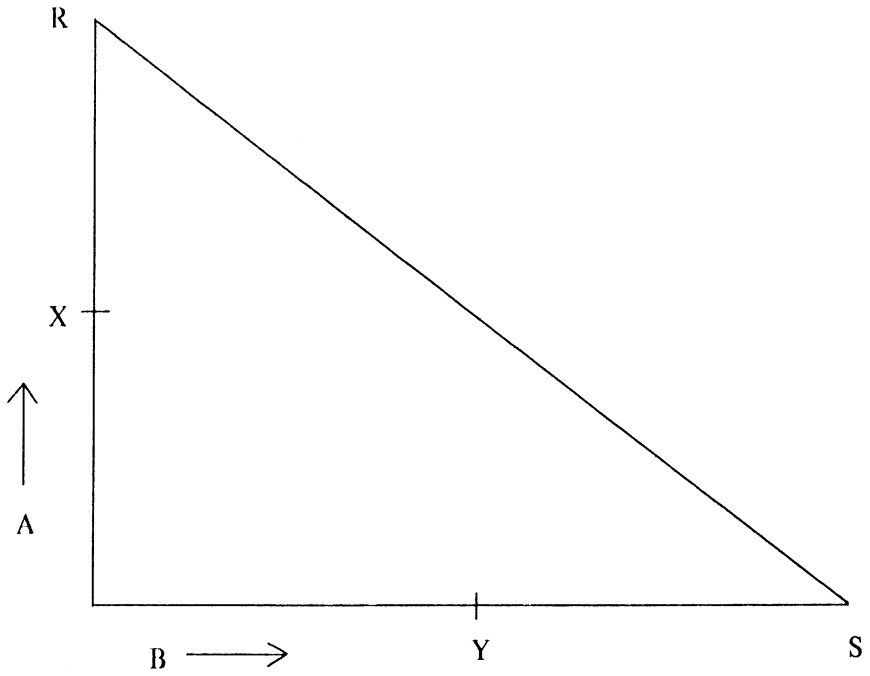

Figure 2-3: A and B after two intervals of time

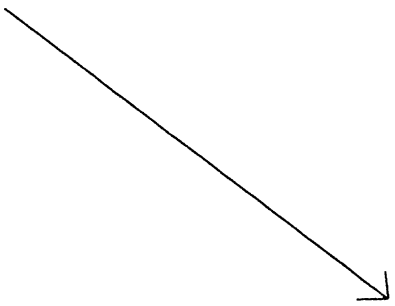

Figure 2-4: The motion of B as seen by A

Using turtle geometry we describe A as having a motion of **FORWARD 3** on a

heading of 0 degrees. A will move forwards 3 units during each interval of time. Similarly, B has a motion of **FORWARD 4**, on a heading of 90 degrees. B will move forwards 4 units during each interval of time.

We can use this same method to determine the motion of objects moving non-linearly, for example, objects moving with circular motion. The first step is to consider how each object moves in one interval of time and then to determine the state of each object from the reference frame object. This method is repeated for the next interval of time, and each object's state is determined with respect to the reference frame object. Gradually, over time, we build up a picture of how an object appears to move, as seen by a chosen reference frame.

This representation uses the intrinsic, local and procedural properties of turtle geometry. In our example, A moves 3 units and B moves 4 units, for each discrete interval of time. Their new state depends only on their original state and the effect of their motion for this time interval. No reference is made to any external frame of reference. The path we see for an object is the path with respect to the intrinsic coordinate system of the reference frame. Each object is given a motion, which is local and procedural, and this is repeated for each interval of time.

Relative motion problems can now be described as moving an object in a specified way during a given amount of time, and then finding the position of all the other objects from the reference frame object. We repeat this process to build up the path we see over several time intervals. We do not make any reference to a fixed, external coordinate system. The problem now involves position, not velocity, with respect to the reference frame object. We can think of velocity as a quantity that directly effects position. The higher the velocity, the greater the distance interval traveled in one unit of time.

2.4 Extending Logo to Model Relative Motion

The relative-motion system is an extension of Logo. Logo is both procedural and extensible. This means that we can design procedures, give them names and use them in a similar way to primitive commands. To simulate relative motion a small group of key primitives must be designed. To do this we must first focus on what steps we need to perform in setting up and simulating motion on the computer screen.

The previous section described a representation for motion based on the intrinsic, local and procedural characteristics of turtle geometry. Our representation involves giving an object a motion to be performed during a discrete interval of time. We must be able to give this motion to an object.

Initially we need to create the objects that we will give motions to. We need to distinguish each object by name, so that we can address commands to it, such as giving it a motion. An object's state, after each interval of time, depends on its initial state and the effect of the state-change operators making-up its motion. We must initially position each object on the computer screen and give each object an orientation. To distinguish each object's path as it moves we need to choose a pencolor or a penwidth, depending on the Logo implementation. We need to perform these initializations once for each object.

We want to see how objects appear to move as seen by a chosen object. We need to create our objects, give them motions and then view their motion from a chosen reference frame. We must be able to specify the frame of reference.

Now we have isolated three key components in setting up a motion simulation: create each object, give each object a motion and choose a frame of reference. We must design primitives to efficiently accomplish each task. We will need an additional command to start the simulation, once we have set up our objects and chosen a reference frame. These commands will be described in detail in Chapter Four, along with examples to illustrate a variety of different motion scenarios.

Chapter Three

Relative Motion

Students and teachers find relative motion problems difficult to solve. The approach used is abstract and the problems are usually biased towards giving an analytic solution that reflects this approach. In this chapter we will discuss the traditional method of solving relative motion problems and outline some of the problems presented by this approach.

Motion problems occurring in daily life are often very different from the "fake" problems we find in textbooks. Many students show poor performance when dealing with problems which do not precisely fit a standard textbook solution. In this class of problem, students work from an intuitive framework. We will examine some common misconceptions students have about motion and suggest why students adopt wrong intuitions.

To conclude this chapter we will use the new representation for motion, developed from turtle geometry, to illustrate how this representation can provide a superior framework for solving motion problems and for intuitively understanding motion.

3.1 Current Teaching Methods

Traditional physics courses usually approach relative motion in a static way, through worked examples. There is no "hands-on" experience to allow the student to get a "feel" for the subject. It is not possible, in a laboratory situation, for the student to set up moving objects and view how the objects appear to move as seen by one of the objects. The teacher may show a film to illustrate how objects move when viewed from different reference frames. However, this is a passive introduction to the concepts of motion. The student cannot explore and experiment on his/her own or verify solutions to analytic problems.

3.1.1 A Traditional Presentation

Relative motion is usually taught using the concepts of vectors and vector algebra. Many relative motion problems use the term *velocity*. For example,

> A train is moving due East with a velocity of 40 m.p.h., and another train is moving south with a velocity of 30 m.p.h; find the velocity of the second train relative to the first.

Vectors are used to represent velocities and vector algebra is used to manipulate the vectors. The laws of vector algebra outline the addition and subtraction of vectors. These methods provide a statement for the principle of relative motion as outlined in the following standard textbook definition:

> If the velocity of a particle P relative to some assigned point O is U and the velocity of Q relative to O is V, then the velocity of P relative to Q is the difference of $U - V$.

This is illustrated in Figure 3-1. The vector $U - V$ represents the velocity of P relative to Q.

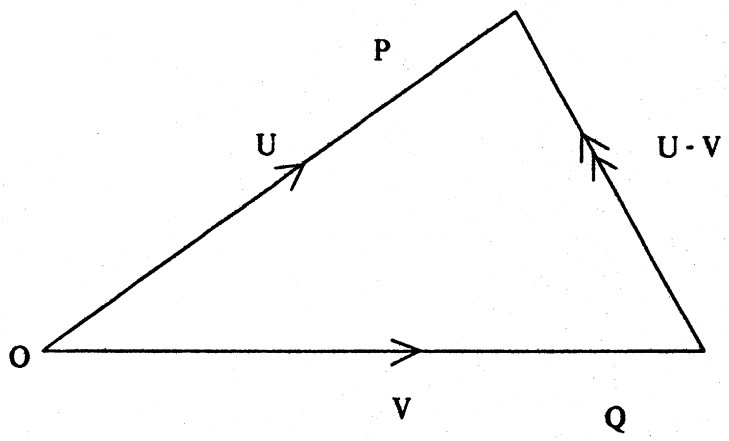

Figure 3-1: The velocity of P relative to Q

For example, two trains P and Q are traveling on parallel rails at 40 m.p.h. and 30

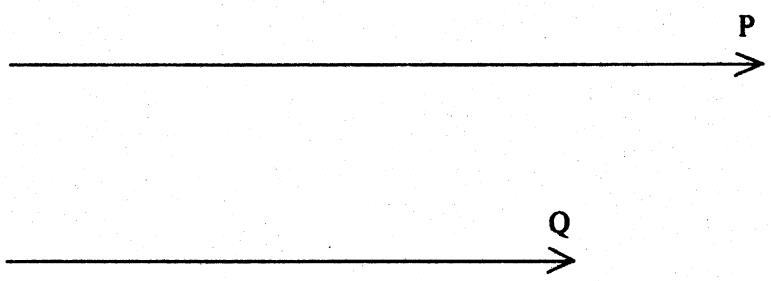

Figure 3-2: A simple relative motion problem

m.p.h., respectively, as shown in Figure 3-2. Because the trains are moving parallel to each other, the relative velocity of P from Q is the algebraic difference of the two velocities, 40 - 30 = 10 m.p.h.

If P and Q are not moving parallel to one another, we must use the method of vector difference to solve the problem. For example, P is moving East at 40 m.p.h. and Q is moving South at 30 m.p.h., then vector ***BA*** represents the velocity of P with respect to Q. This is shown in Figure 3-3.

3.1.2 Problems with the Traditional Method

The previous section gave a brief outline of a traditional approach to relative motion. It takes a very enthusiastic teacher to put this material across in a meaningful way, using the current representation of vector algebra alone. Several problems emerge from this formulation of relative motion:

* Vectors are sufficient, but usually people do not think in terms of vectors; they have a much more intuitive approach to motion problems.

* The principles are written in the terminology of vector algebra. The statement

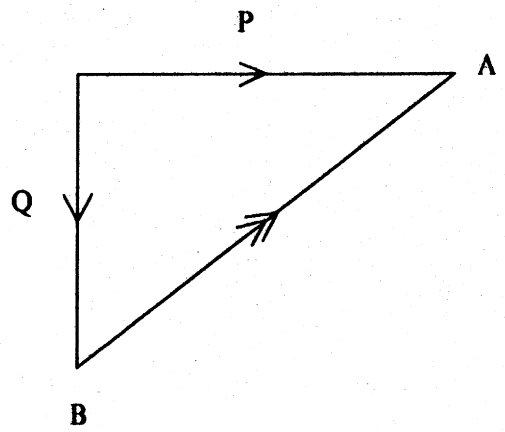

Figure 3-3: The method of vector difference

of these laws does not convey a mental picture of what is actually happening and how you should think about relative motion problems. Students do not remember the statements of the basic principles when they are written in such a fashion. In addition, this representation is abstract; it does not give the student a comprehensive notion of what we mean by a frame of reference, what it means when we are in a given frame of reference, or why a frame of reference is important.

* The formulation is a static representation describing motion in terms of an object's velocity and an object's relative velocity rather than the *path* on which an object moves or appears to move.

* The formulation is restricted to cases of rectilinear motion. It does not outline an approach for discussing relative motion when the objects are moving non-linearly. In fact, in this case, the motion must be discretized, and an object's relative velocity found for each discrete interval of time. However, combining

these resultant relative velocity vectors for each interval of time, results in a velocity vector diagram, not a positional vector diagram. Translating from a velocity vector diagram to a positional vector diagram is a non-trivial task.

* The student uses a static environment to set up and solve problems. These problems are usually solvable. The student cannot experiment, explore or verify his/her solutions to problems formulated independently.

3.2 Common Misconceptions in Relative Motion

Students build up knowledge from formal learning, experimentation and contact with everyday phenomena. Studies by several investigators show that many students use this knowledge as a framework for interpreting new course material. A student's knowledge also consists of intuitive ideas. These ideas are later extremely difficult to modify. Students are very reluctant to cast out their intuitive ideas and often, as a result, view information incorrectly. They may use their formal knowledge in dealing with a specific type of problem, but will revert to their intuitive ideas when something of a slightly different nature is posed. For example, students use this formal knowledge to solve written problems in a given subject area. However, these problems are often very different to the ones students meet in daily life. One objective of this thesis is to help students link their intuitions to formal methods. diSessa [diSessa 83] claims that intuition plays an important part in the way a student learns; not only can it affect the efficiency of learning, but also what a student learns.

In dealing with relative motion the discrepancy between intuition and physical reality can be illustrated using the following example. A person walking briskly drops a penny. What is the path of the penny as it falls to the ground, as seen by a stationary observer? Figure 3-4 shows the responses to this question. The answers given are: the penny falls straight down; the penny falls in an arc for a short way, then falls down; the penny falls forwards in a curve; the penny falls backwards in a curve. The correct answer is that the penny falls forwards in a curve.

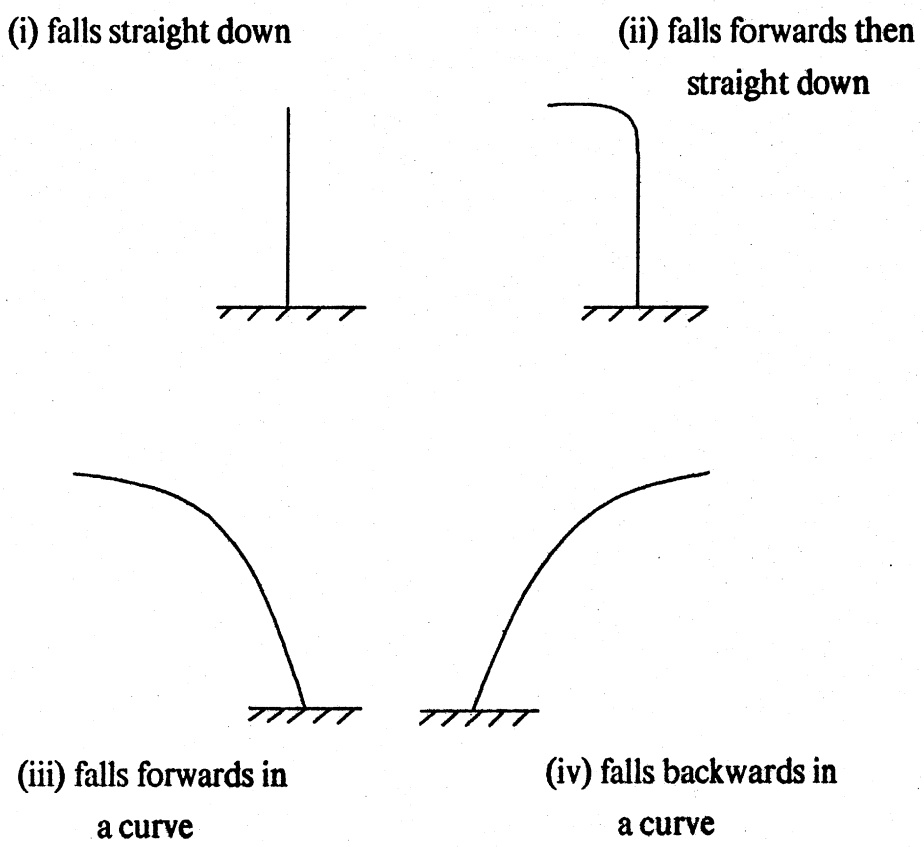

Figure 3-4: Suggested paths of a penny in falling to the ground

"The penny falls straight down" is a popular choice, often given by over 50% of all students. Why is it that so many students answer this problem incorrectly? To a person standing still and dropping a penny, the penny falls at his/her feet. To a person walking forwards and dropping a penny, the penny falls at his/her feet (see Figure 3-5). The incorrect response is given because students put themselves into the incorrect frame of

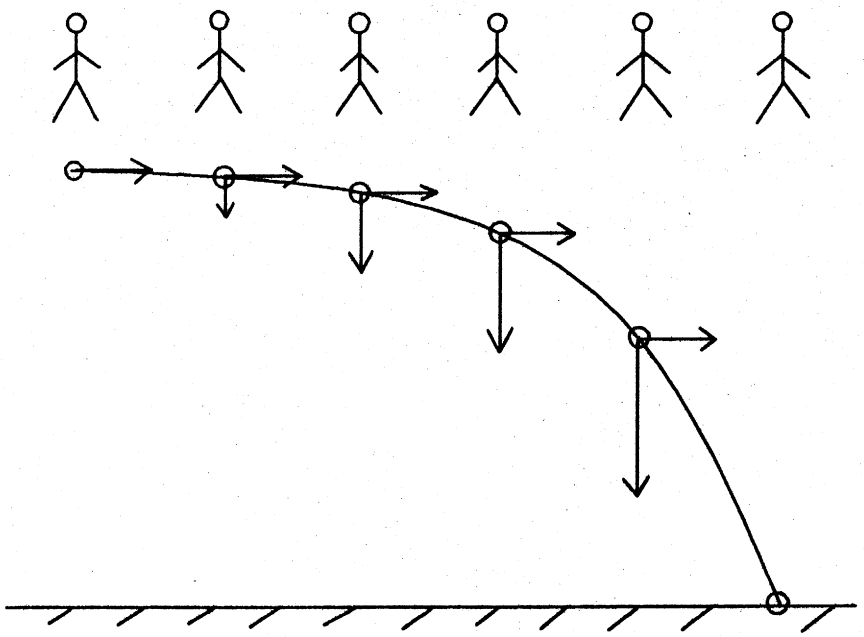

Figure 3-5: A penny dropped by a moving person

reference. They interpret the question to coincide with how *they* themselves see the penny, rather than how an independent observer sees the penny.

Some responses given indicate that the penny continues to move forwards a short way, then falls under gravity. In other words, the penny initially acquires the motion of the person, then gravity takes over. This is similar to the pre-Newtonian theory of impetus [McCloskey 83]. This theory states that when a person sets an object in motion, the person implants into it a certain impetus. Impetus is described as a force which enables the object to move in the direction in which the person starts it. The suggestion that the penny moves forwards, then falls under gravity, is explained by the impetus theory as: "the initial impetus gradually dissipates, leaving the penny to fall under the impetus imparted by its

own weight".

Some people believe the penny falls backwards. This is a misperception caused by a combination of factors. It starts from misinterpreting the penny's initial velocity as zero; then, misinterpreting where the penny is viewed from. If the penny's initial velocity is zero, a person walking forwards sees the penny falling backwards. The student imagines the penny starts from rest, and then describes the penny's motion from a moving reference frame. Where the penny is viewed from is a very important consideration in discussing problems of this nature.

The above discussion shows that even in simple motion examples, such as a penny being dropped by a moving person, the majority of students do not work from a correct intuitive base. Many students who have taken physics courses, and who have studied Newton's laws of motion, fail to answer motion questions correctly.

3.2.1 Understanding Wrong Intuitions

Three distinct areas emerge to suggest why students often answer motion questions, similar to the one detailed above, incorrectly:

* Many students have the notion that objects fall straight down when dropped.

* Students often do not understand the concept of a frame of reference.

* Many students do not understand that an object carried by another moving object is itself "moving" and when released its motion is initially identical to the motion of the object originally holding it.

The notion that objects fall straight down when dropped is very common. Intuitive notions are built up gradually over time. People experiment with dropping from a very early age. Babies nine months of age will drop rattles and toys from a high chair. Babies one year of age will perform the same experiments of dropping, then look over the edge of the high chair to see the rattle or toy on the floor. Psychologists describe this as a child

relating to the concept of *object permanence.* At the same time, however, the child is building up his/her intuitions about dropping; the baby drops the toy then sees it on the floor below himself/herself.

I performed an experiment with a four-year old child using "The Rain Game" supplied with LCSI's Sprite Logo. The purpose of this computer game is to send a rain shower from a moving cloud onto a seed. If the rain hits the seed a tree grows. The seed is positioned randomly across a horizontal plane, near to the bottom of the computer screen, and the cloud moves horizontally across the top of the screen. When the space-bar is pressed a shower of rain is released from the cloud. I watched the four-year old play with this game. Every time the child hit the space-bar to send down a shower of rain, the cloud was directly over the seed. After a few tries, the child realized that this was the incorrect strategy to adopt, and experimented by releasing the shower before the cloud passed over the seed. Success! This shows that even in very young children the intuitive notion that objects fall straight down is present.

Misperceptions can cause us to build up incorrect intuitive notions about motion. McCloskey outlines work which describes illusions in the perception of motion when viewed against a moving background [McCloskey 83]. McCloskey uses a computer simulation experiment, with a dot falling down a screen against a moving rectangular background. The dot represents an object being dropped by a moving person. When the background moves, the dot appears to move either straight down or in a slight curve. In this case the student uses the moving background as the frame of reference. In the absence of the moving background, the dot is seen to move in a parabolic curve, its actual motion when dropped by a moving carrier.

I set up a similar experiment to McCloskey using LCSI's Sprite Logo to implement the simulation. I used a falling dot and a background of similar dots. When the background moved the falling dot appeared to move straight down. Several graduate students responded that the dot was falling straight down. The moving background in this case acted as a reference frame against which the students described the falling dot.

Many students have no comprehension of a frame of reference and consequently do not realize that motion viewed from different perspectives can look totally different. For example, consider the following questions:

1. A relief package is dropped from a plane. When will the package have to be dropped in order to reach its target?

   (a) Before the target. (b) Over the target. (c) After passing the target.

2. If the pilot wants to keep an eye on the package where will the pilot have to look?

   (a) Straight down. (b) Behind the plane. (c) In front of the plane.

(a) is the correct answer to both of these questions. When I gave these questions to a high-school class many students answered the first question correctly. The class had previously studied Newton's laws of motion and dropping objects. Despite giving a correct response to the first question, many students answered the second incorrectly. A popular answer for the second question was (c). The answers given to the first question show that most students know that the package falls in a parabolic curve, so that it will reach the target. Many answers given to the second question indicate that students believe the pilot sees the package fall in a parabolic curve too. Here we see that some students cannot make the connection that the plane is moving too, with the same horizontal speed, so that the pilot sees the package fall straight down. Some students cannot imagine themselves being in a different reference frame to "see" the motion.

The concept that a carried object inherently has the motion of its carrier is often not obvious to students until they study Newton's laws of motion. For example, when a penny is dropped by a moving person, at the instant it is let go, it moves exactly as the person moves; it has an identical speed in the same direction as the person moves.

3.3 Overcoming the Problems of the Traditional Method

In the previous sections we looked at the traditional vector algebra method used to teach relative motion, some of the problems associated with this method, and an example of a relative motion problem dealing with dropping an object.

We will now explore how the new representation for motion, as outlined in Section 2.3, is pedagogically superior to the traditional presentation, in both overcoming the problems of the traditional method and by helping students modify their own intuitive frameworks.

The traditional method represents motion as a static velocity vector. We now use a procedural definition of motion. This is a more intuitive approach as we tell an object how to move during a discrete interval of time. This motion is continuously repeated to build up a complete picture of the motion over time. For example, the traditional method of describing relative motion represents a car traveling North at 10 m.p.h., by a vector of magnitude 10 units, pointing North. This same example is represented procedurally by giving the car a motion of FORWARD 10. This means "move forwards 10 units during each interval of time". The motion performed in this way is local as the turtle only needs to consider its initial and final states.

The new representation is intrinsic. This means we do not concern ourselves with an external frame of reference. For example, let us discuss a relative motion problem with two people, A and B, moving so that A travels North at 3 m.p.h. and B travels East at 4 m.p.h. If we wish to find the relative motion of B from A using the traditional approach, we must first define a coordinate system, draw the vectors, find the vector difference and then describe this difference with respect to our chosen coordinate system. In our new approach, we do not concern ourselves with an external reference frame. We choose an object to view the motion from, we move all the objects as described by their procedural definitions and then we determine the relative position of all the objects from our chosen reference frame object. We do not refer to an external reference frame, reference is *only* made to the intrinsic coordinate system of the reference frame object.

The traditional approach makes no explicit reference to a frame of reference. The new representation involves specifying a frame of reference as the perspective from which to view the motion. We do this whether we use the computer simulation to show the motion, or whether we perform the simulation by hand. In addition, using the computer simulation, we have the added flexibility of quickly being able to move into a different reference frame to view the motion. The concept of a reference frame becomes a viable entity using the new representation for motion.

When we look at relative motion using the new representation for motion we determine the *path* on which the object appears to move. The path is what we actually see traced out if we watch an object move. The traditional method looks at motion in terms of velocity only. Let us consider an example, of two ships, moving as shown in Figure 3-6, to illustrate this basic difference between the two approaches.

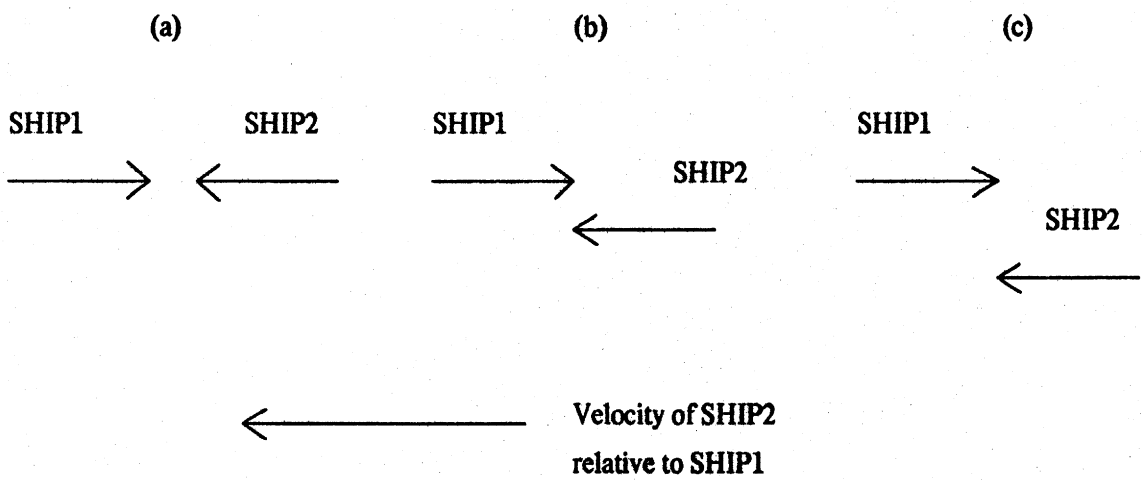

Figure 3-6: Motion in terms of velocity vectors

The ships, SHIP1 and SHIP2, have equal, but opposite velocities. In the first diagram the ships are moving on a collision course, and in the remaining examples they are separated by a short distance. If we determine the relative velocity of SHIP2 from SHIP1 using the

traditional approach, we find in each case the relative velocity is the same. When we use the relative velocity vector alone we lose valuable positional information; we do not see the spatial relationship of SHIP2 from SHIP1. The new representation shows the path SHIP2 appears to move on, and clearly indicates that in the first illustration SHIP2 is on a collision course with SHIP1. This is shown in Figure 3-7.

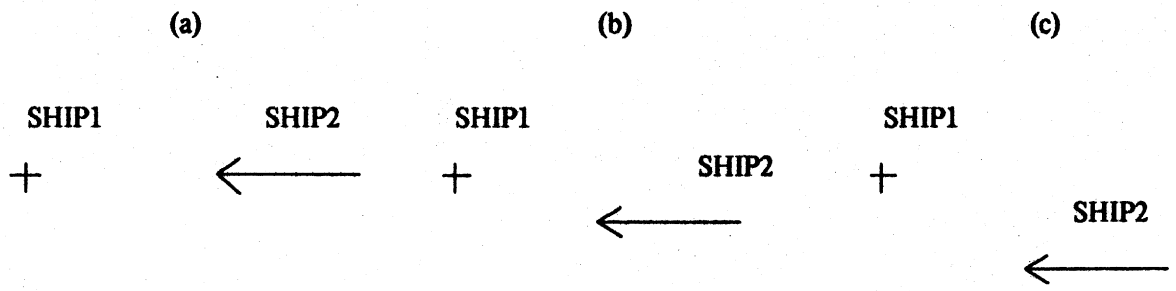

Figure 3-7: Motion in terms of paths

The new representation is far superior in its treatment of non-linear motion; for example, circular motion. We use exactly the same method for relative rotational motion as relative linear motion; we let each object move with its specified motion and then take each object's position relative to the reference frame object. We continue repeating this to build up a path for each object. These paths illustrate how the objects appear to move seen from the chosen reference frame object.

Let us consider the example of two circles, CIRCLE1 and CIRCLE2, which have procedural definitions of motions given by:

```
SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT 5]
SETMOTION  "CIRCLE2  [FORWARD 5  RIGHT 10]
```

This means that CIRCLE1 is drawn by repeating the compound command [FORWARD 5 RIGHT 5] and CIRCLE2 by repeating [FORWARD 5 RIGHT 10]. The radius of each circle is proportional to the ratio of the FORWARD *:amount* to the RIGHT *:amount.* Therefore, CIRCLE1 has a radius of twice that of CIRCLE2. CIRCLE2 turns through twice the angle CIRCLE1 turns through on each step, so that when CIRCLE2 has completed a revolution, CIRCLE1 has completed half a revolution. If we want to find how CIRCLE1 sees CIRCLE2 move, we must move both objects, then find the position of CIRCLE2 from CIRCLE1, and keep repeating this to build up the path CIRCLE2 appears to move on.

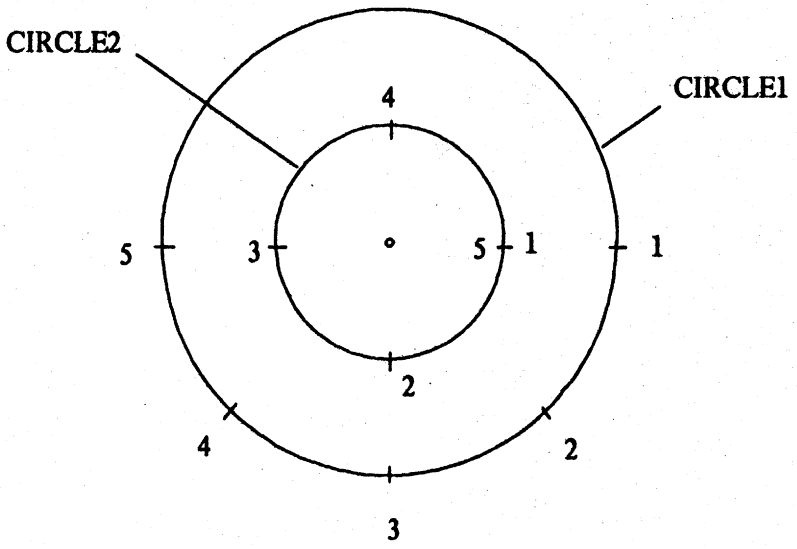

Figure 3-8: Some positions of CIRCLE1 and CIRCLE2

Figure 3-8 shows some positions of CIRCLE1 and the positions of CIRCLE2 at the same instances of time. At the first instant, we stand on CIRCLE1 and note where CIRCLE2 is. Figure 3-9 shows this. Then we repeat this for the next interval of time, and the next, and in this way we gradually build up the path CIRCLE2 appears to move on. This is shown in Figure 3-9. When CIRCLE1 has completed one revolution CIRCLE2 has

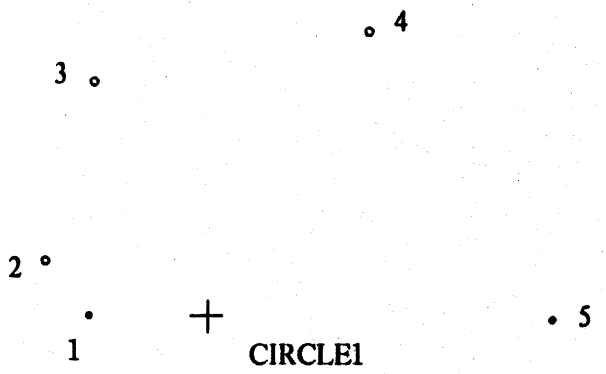

Figure 3-9: Building the path of CIRCLE2 as seen by CIRCLE1

completed two revolutions. Figure 3-10 shows the complete path of CIRCLE2 as seen by CIRCLE1, while CIRCLE1 makes one revolution. This example illustrates how simple this method is. The final result is the actual motion seen by our chosen object.

Now let us illustrate how we can approach this same problem using the traditional representation for relative motion. The traditional approach, however, does not outline a method to deal with non-linear motion. To attempt the problem we have to look at the motion in discrete intervals of time. This involves looking at the velocity vector representation of the motion during each interval of time. When an object moves with circular motion, at a chosen instant in time its velocity is tangential to its circular path. To determine the relative motion of one object moving in a circle, from another moving in a circle, we need to know each object's instantaneous velocity vector, then combine these velocities to form a vector difference diagram to give us the instantaneous relative velocity vector. Then we must repeat this for the next interval of time. To illustrate this let us consider our previous example of two objects moving with circular motions given by:

```
SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT 5]
SETMOTION  "CIRCLE2  [FORWARD 5  RIGHT 10]
```

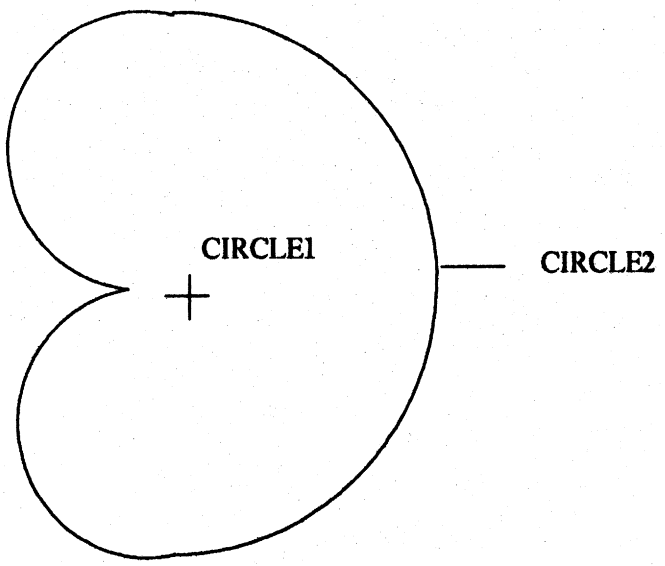

Figure 3-10: The path of CIRCLE2 as seen by CIRCLE1

Using these definitions we can specify the magnitude of each object's velocity vector. The radius of each circle is proportional to the ratio of its FORWARD *:amount* to its RIGHT *:amount.* The amount given to the RIGHT command is the amount to turn through during each interval of time. This amount is therefore proportional to the angular velocity, ω, of an object moving on a circular path. We can use the mathematical equation for the velocity of an object moving in a circle, of radius r,

$$v = r\,\omega,$$

and substitute into this equation the turtle geometry values for r and ω. This shows us that:

v is proportional to the FORWARD *:amount.*

Therefore, an object's velocity is proportional to the FORWARD *:amount.* This means in our example that *both* objects have velocity vectors of the same magnitude, as they both have the same numerical amount given to their respective FORWARD commands. Figure 3-11 shows the velocity vectors at chosen intervals of time.

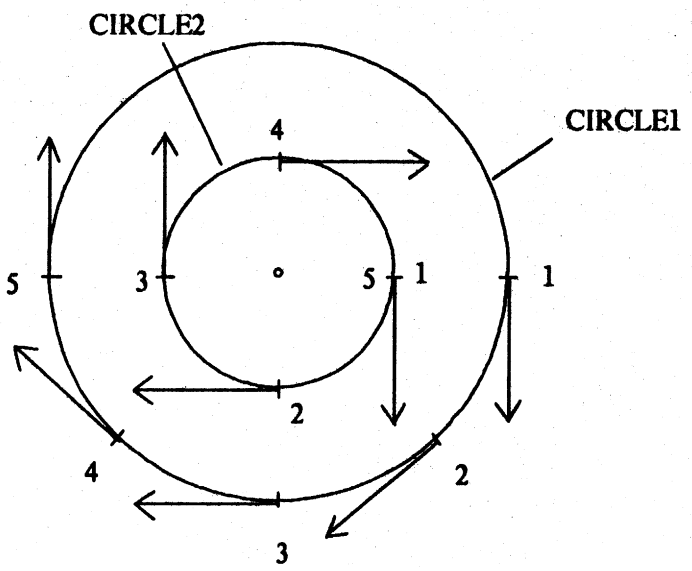

Figure 3-11: Velocity vectors for CIRCLE1 and CIRCLE2

For each interval of time we must combine CIRCLE1 and CIRCLE2's velocity vectors to form a vector difference diagram. Figure 3-12 shows the resulting diagrams for five intervals of time as shown in Figure 3-11. Combining these five relative velocity vectors gives us the path of the relative velocity vector as shown in Figure 3-13.

This solution does *not* tell us how CIRCLE2 appears to move. The problem has been entirely represented in terms of velocity and has given us a solution in terms of CIRCLE2's relative velocity. To go from the velocity vector path to the positional vector path we must construct orthogonal vectors, as in Figure 3-14, to result in a path similar to the one shown in Figure 3-9. However, there is a problem at the initial instant in time. We lose information as we cannot determine the positional vector because the velocity vector is zero.

As illustrated here the traditional approach is long and laborious in dealing with non-rectilinear motion. Unless a student is extremely competent at vector manipulation this

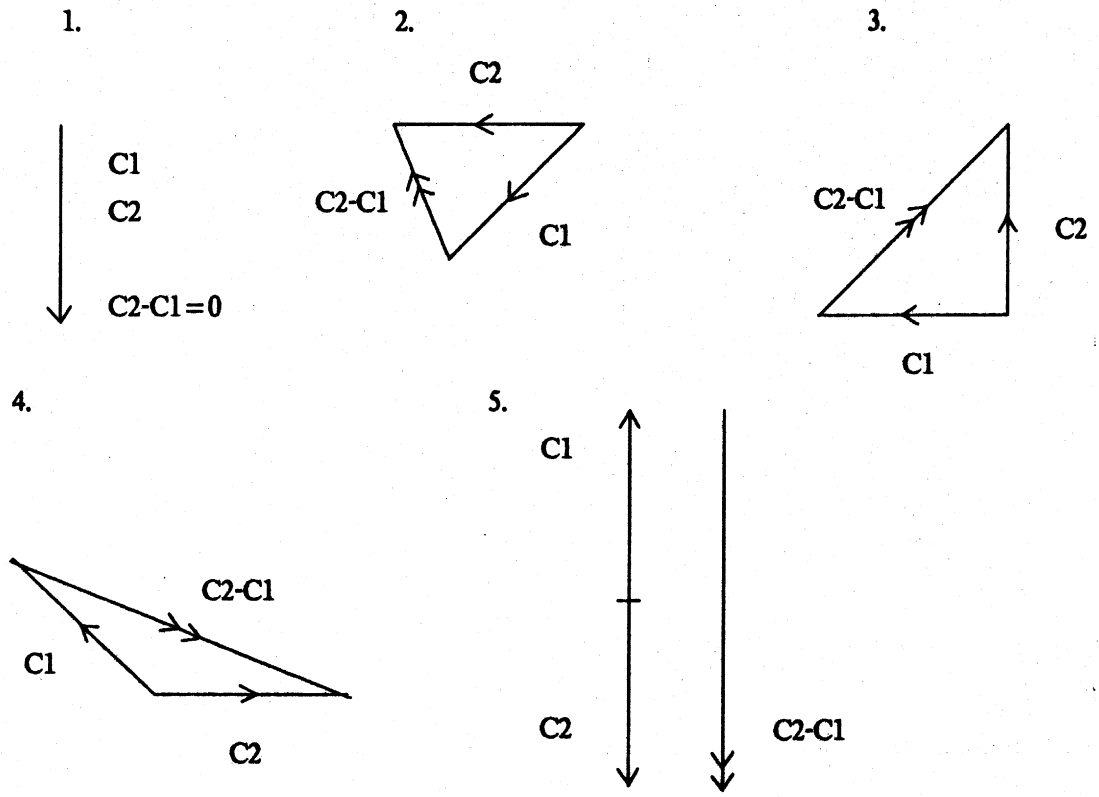

Figure 3-12: Relative velocity diagrams for CIRCLE2

could be an overwhelming task.

The new representation provides students with a comprehensive framework for conceptualizing a frame of reference and illustrates a simple method for determining motion. A student can set up a motion simulation and view the motion from different reference frames. The motion seen is the actual path on which an object appears to move. The student has a visual, dynamic environment to test out his/her theories about motion. The motion, from a chosen reference frame, may take an unusual path, one which the student may not have expected to see. The student can use the new method to analyze the

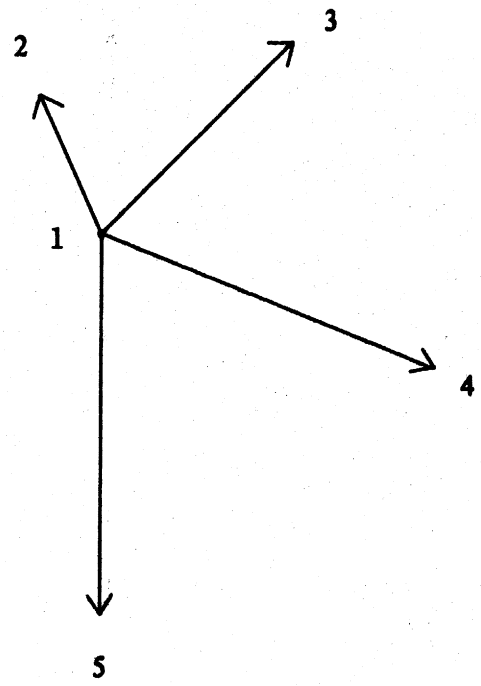

Figure 3-13: The path of the relative velocity vectors

motion and use the simulation to show the motion from different reference frames so that s/he comes to a good understanding of why this motion is occurring.

Students build up notions about motion through experience. These notions are formulated at an early age. However, it is very hard for students to "debug" their intuitions about motion using the traditional representation of motion. It is impossible to set up experiments to move between different reference frames using the traditional approach. We can perform an experiment of a runner dropping a penny. However, it is difficult to describe exactly the path the penny takes in falling to the ground. The motion happens very quickly, external factors, such as air currents, may effect the penny's path, and the moving runner may inadvertently cause the student to use the runner as the frame of reference.

A further problem in dealing with motion is due to a carried object inherently

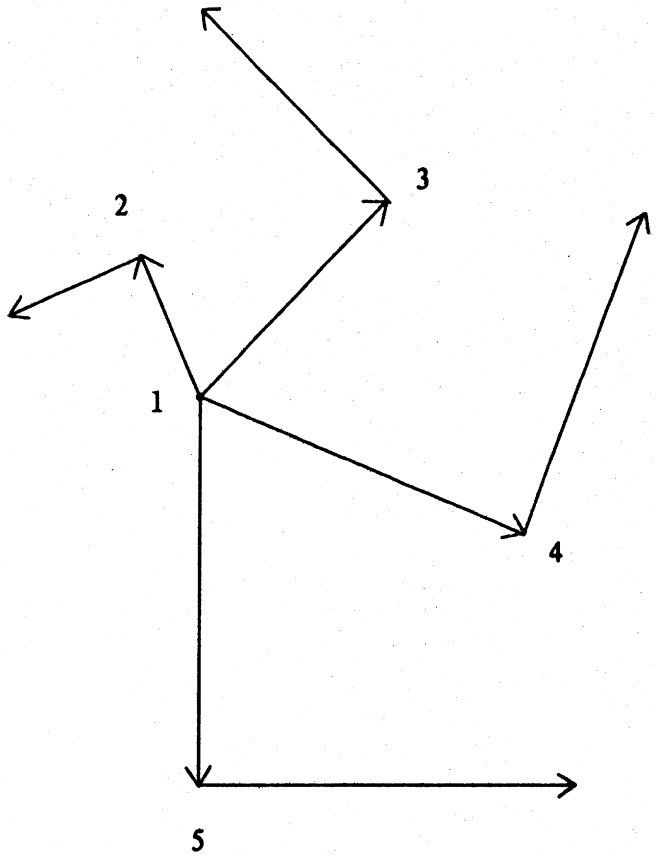

Figure 3-14: Adding in positional vectors

possessing the motion of its carrier, so that when an object is released it initially has the instantaneous motion of the carrier. The relative-motion system provides primitives for falling, tossing and pushing. When a object is given one of these motions, such as falling, its motion must first be initialized. This entails declaring that the named object is being dropped by another named object. The microworld provides a framework for exploring the path that objects take when they are dropped, thrown or pushed from moving objects, by allowing students to set up a variety of motion scenarios to illustrate this.

Chapter Four

The Relative-Motion Microworld

The relative-motion microworld provides a framework within which the student can explore the concepts of relative motion and frames of reference. A stand-alone student text introduces the student to the computer-based microworld and its embedded concepts. This chapter describes the implementation of the microworld. It outlines the primitive commands that we use to set up and view motion on the computer screen and it describes additional primitives that have been designed for use under specific conditions, such as falling under gravity.

The computer-based microworld allows the student to set up a wide variety of different motion problems. These problems include examples of rectilinear motion, non-parallel rectilinear motion, motion under gravity and circular motion. This chapter shows some of the possible explorations that a student can perform when using the microworld. The student uses the student text as a guide. The final sections of this chapter describe the text and the teacher's reference manual.

4.1 Implementation of the Microworld

The microworld is implemented using Logo. Logo and turtle geometry provide a framework for incorporating the principles of relative motion and frames of reference. Four key primitives have been designed, which are included as simple extensions to the turtle's command set. The microworld has been implemented using Terrapin Logo on the Apple II 64K personal computer and using LCSI Logo on the Macintosh 512K personal computer.

4.1.1 Primitives

There are four basic steps in setting up a computer simulation to model relative motion. The first step is to create independent objects that are to move on the computer screen. Each object must then be told how to move, one must be chosen as the reference frame object, and finally all the objects must be set moving. These four steps are the building blocks to simulate motion.

Let us consider setting up an example to introduce the key primitive commands. Table 4-1 gives a complete list of all of the microworld primitives. Suppose two people, A and B, are moving so that A moves North at 3 m.p.h. and B moves East at 4 m.p.h. The first step in simulating motion is to create the objects, A and B, using the command MAKETURTLE:

```
MAKETURTLE  :name  :x.position  :y.position  :heading  :color
```

MAKETURTLE makes an object, with a chosen name, at a given initial position on the computer screen, on a given heading and with a color specified for its drawing pen. For example, A and B are created with:

```
MAKETURTLE  "A  0  0  0  :RED
MAKETURTLE  "B  0  0  90  :GREEN
```

A is at position [0,0] on a heading of 0 degrees. As A moves its path is drawn with a red pen. B is at position [0,0] on a heading of 90 degrees. Its path is drawn with a green pen. The pen color is used to distinguish each object. The Macintosh does not have a color monitor. In order to distinguish objects using the Macintosh implementation each object is made with a drawing pen of a different width. For example, the MAKETURTLE command is:

```
MAKETURTLE  "A  0  0  0  1
```

When A moves across the computer screen its path is drawn with a pen of one pixel width.

Each object must be told how to move using the SETMOTION or the

```
FALL
INIT.FALL    :name    :from
INIT.PUSH    :name    :from
INIT.TOSS    :name    :from
MAKEFRAME   :frame.of.reference
MAKETURTLE   :name   :x.position   :y.position   :heading   :color
MOVE    :turtles
PUSH    :speed
SETMOTION    :name    :motion
SETRELATIVEMOTION   :name   :relative.to    :motion
TOSS    :speed
```

Table 4-1: The microworld primitives

SETRELATIVEMOTION command:

```
SETMOTION    :name    :motion
SETRELATIVEMOTION    :name    :relative.to    :motion
```

The motion given to each object is procedural and it is the motion to be performed during each discrete interval of time. For example, A and B are given the motions:

```
SETMOTION  "A  [FORWARD  3]
SETMOTION  "B  [FORWARD  4]
```

A will move forwards in steps of 3 units and B will move forwards in steps of 4 units.

The SETRELATIVEMOTION command is used when an object is set moving *relative* to another object. For example, if B is moving *relative* to A, the command to give this motion to B is:

```
SETRELATIVEMOTION  "B  "A  [FORWARD  4]
```

In this case, B will move forwards 4 units on its given heading, and in addition, will move

forwards 3 units in A's direction.

An object must be chosen as the frame of reference. The simulation shown on the screen is the path of the objects as seen by the object chosen as the reference frame. MAKEFRAME is used to do this:

```
MAKEFRAME   :frame.of.reference
```

If we want to simulate how A sees B move, then A is the reference frame:

```
MAKEFRAME  "A
```

The final command necessary in setting up a motion simulation on the computer screen is MOVE:

```
MOVE   :turtles
```

MOVE takes as input a list of the objects that are to be moved and shown on the computer screen. MOVE calls the microworld procedures to begin the simulation. In this example, A and B are to be moved, and the following command starts the system:

```
MOVE  [A  B]
```

Figure 4-1 shows the complete set of commands, combined to form a procedure, to illustrate this example. The procedure is called **DEMO.ANGLE.**

DEMO.ANGLE requires an *input parameter,* **:viewfrom,** the frame of reference. For example, if A is chosen as the reference frame, to run the procedure, A must be specified as the input parameter:

```
DEMO.ANGLE  "A
```

Input parameters make a procedure more versatile, as the procedure does not need to be repeatedly edited to change a particular quantity.

```
TO DEMO.ANGLE  :viewfrom
  MAKETURTLE  "A  0  0  0  :RED
  MAKETURTLE  "B  0  0  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [FORWARD  3]
  SETMOTION  "B  [FORWARD  4]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :viewfrom
  MOVE  [A  B  EARTH]
END
```

Figure 4-1: A procedure to simulate A and B moving

A stationary EARTH turtle is included in this example in order that we can display the motions of A and B at the same time on the computer screen.

In addition to the primitives that make up the four basic steps to simulate motion, other primitives have been designed for use under specific conditions: objects falling under gravity, objects being thrown under gravity, and objects being pushed across horizontal surfaces. These three specific conditions all involve giving an object a motion and initializing this motion before the system is started. It is necessary to initialize the object's motion as it may have an inherent motion, due to being carried by a moving object. For example, when a penny is dropped by a running person, at the instant the penny is released it has a motion identical to that of the moving person.

If an object falls under gravity, we give it this motion using the procedure FALL and we initialize the motion, before we start the system, using the procedure INIT.FALL:

```
INIT.FALL   :name   :from
```

The procedure INIT.FALL requires two input parameters: the name of the falling object and the name of the object it is falling from. For example, if an apple falls from a tree, the following commands must be included in a procedure to illustrate the motion:

```
      .
      .
SETMOTION  "APPLE  [FALL]
      .
      .
INIT.FALL  "APPLE  "TREE
```

When an object is thrown into the air the procedure TOSS is used:

```
TOSS    :speed
```

TOSS requires one input parameter, the speed with which the object is thrown. The motion must be initialized using INIT.TOSS:

```
INIT.TOSS    :name    :from
```

For example, if a person throws a ball vertically into the air with a speed of 5 units, the following commands must be included in a procedure to show this:

```
MAKETURTLE  "BALL  0  0  0  :RED
      .
      .
SETMOTION  "BALL  [TOSS  5]
      .
      .
INIT.TOSS  "BALL  "PERSON
```

The procedure PUSH is used to illustrate an object pushed across a horizontal surface:

```
PUSH    :speed
```

PUSH requires one input parameter, the speed with which the object is pushed. The motion is initialized using INIT.PUSH:

```
INIT.PUSH  :name  :from
```

For example, if a person holds a piece of ice and gives it a push of speed 5 units across the surface of a table, the following commands are used to show this:

```
MAKETURTLE  "ICE  0  0  0  :BLUE
     .
     .
SETMOTION  "ICE  [PUSH  5]
     .
     .
INIT.PUSH  "ICE  "PERSON
```

4.1.2 System Implementation

A simulation is set up using the relative-motion microworld by following the four basic steps: create each object, give each object a motion, choose a reference frame, and move all the objects.

The system has been designed for a Logo implementation with *one* turtle. This means that in order to show several objects moving on the screen at the same time, information concerning each object's position, heading, color and motion must be stored and retrieved when necessary. This information is stored in *property lists* associated with each object. The following is an outline of the tasks of each key primitive.

1. MAKETURTLE **:name :x.position :y.position :heading :color**

   * Sets up the following property lists for this named object:

     - **Frame**: a list of objects that move relative to this object, i.e. are dependent on this object.

     - **Base**: the named object is dependent on the object specified here.

     - **X**: the object's x position.

- **Y**: the object's y position.

- **Heading**: the object's heading.

- **Color**: the color of the object's drawing pen.

- **Update**: the x and y increments resulting from the object's motion in one time interval.

2. SETMOTION **:name :motion**

* Sets up a **motion** property list for this named object.

3. SETRELATIVEMOTION **:name :relative.to :motion**

* Sets up a **motion** property list for this named object.

* Adds the named object to **:relative.to**'s list of dependencies, contained in **:relative.to**'s **frame** property list.

* Updates the object's **base** property to be **:relative.to**.

4. MAKEFRAME **:frame.of.reference**

* Stores the frame of reference in a global variable, and indicates the reference frame as a small cross, in its chosen color, at the center of the computer screen.

5. MOVE **:turtles**

* Translates each object's position with respect to the reference frame object at the center of the screen.

* Starts the system, by calling the procedure **MOVELOOP**.

These commands form the building blocks of all motion simulations that can be set up using the microworld. The additional primitives available to users of the system are used to give a specific type of motion to an object: **FALL, PUSH** and **TOSS**. To initialize these motions **INIT.FALL, INIT.PUSH** and **INIT.TOSS** must be used. **FALL, PUSH** and **TOSS** store the object's speed, resolved into vertical and horizontal components, in a word associated with the object. **FALL** and **TOSS** make corrections to the object's speed for the acceleration due to gravity. The initializing procedures initialize the object's starting speed to the speed of the object doing the dropping, pushing or throwing. For example, when a penny is dropped by a moving person, at the instant the penny is let go, it has the same velocity as the moving person.

The remaining procedures which make up the microworld are transparent to the user. The procedure **MOVE** calls **MOVELOOP**, which starts the simulation. The simulation runs until stopped by the user.

MOVELOOP calls the procedure **CALCULATEMOVE** which determines each object's new position on the screen. For each object in turn, the turtle is set at [0,0], given the object's heading, and the object's motion is run. The motion given to each object, is the motion to be repeated in each time interval. The microworld repeats this motion on every iteration of the procedure **MOVELOOP**. Each iteration is equivalent to a time interval. Once the motion is run, the turtle's new heading is stored in the object's **heading** property, and the turtle's position is stored in the object's **update** property. This is the amount to increment the object's path by for this time interval. In addition, this increment is added onto the **update** property of all objects dependent on this object. In this way the total increment is built up for each object, during a given time interval.

When each object's increment has been calculated, **MOVELOOP** calls procedure **MOVEREST** to move all the objects on the computer screen, with the exception of the reference frame object. The reference frame object remains stationary, at the center of the computer screen, with all the other objects doing the moving.

MOVEREST moves each of the objects in turn. For each object the turtle's drawing

pen is raised and the turtle is set at the object's initial position, or last calculated position. The color of the turtle's drawing pen is set and the pen is put down. The turtle is moved to its new calculated position, drawing its path as it moves. The new position is the original position, plus the increment stored in the object's **update** property, minus the increment of the reference frame object. The object's new position is stored for use in the next iteration of the program loop.

The complete coding of the microworld is included as Appendix C.

Part of the system, dealing with moving each object, was recoded in Apple 6502 assembler language to try to speed up the microworld when using the Apple II 64K personal computer. The speed up was not significant. However, the coding is included in Appendix C. The system requires the Terrapin Logo Version 3.00, as the addresses used by the assembler code are the ones specified for this implementation. The assembler code is too large to be assembled using the assembler included with Logo. Instead, the code was implemented on a DECSYSTEM-20, compiled and downloaded onto the Apple II 64K personal computer. The assembler treats each object as a list of state variables. The microworld has been re-written to reflect these changes. This coding is also included in Appendix C.

4.2 Explorations Using the Microworld

The relative motion microworld is designed for students to have a hands-on experience with relative motion problems. Detailed here are some of the explorations possible using the microworld and an outline of areas covered in the student text. Some of the areas of investigation include:

- Parallel rectilinear motion.

- Non-parallel rectilinear motion.

- Motion under gravity, including objects being dropped and thrown.

- Uniform relative rotational motion, including the solar system, the retrograde motion of the planets, and the rotating earth.

4.2.1 Linear Motion

Examples describing linear motion are used to introduce the student to the concepts of relative motion and frames of reference. Often students are familiar with examples in this category, such as viewing moving vehicles traveling on a highway. This section serves to familiarize the student with the microworld: introducing the key primitives, using the primitives to make a computer simulation, specifying a frame of reference and interpreting the simulation motion.

The following demonstration is used:

```
TO DEMO  :VIEWFROM  :R  :W
  MAKETURTLE  "RUNNER  0  60  90  :RED
  MAKETURTLE  "WALKER  0  40  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "RUNNER  [FORWARD  :R]
  SETMOTION  "WALKER  [FORWARD  :W]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [RUNNER  WALKER  EARTH]
END
```

This example sets up two people, a runner and a walker, to move horizontally across the computer screen. The procedure requires three input parameters: the frame of reference, the runner's speed and the walker's speed. The student chooses a reference frame to view the motion and selects speeds for the runner and walker.

A set of exercises allows the student to explore the motion seen from different reference frames:

- What happens to the stationary earth when it is viewed from a moving reference frame?

- What happens when the runner and walker are moving at the same speed?

- What happens when one person moves backwards?

- Under what conditions does a person appear to move backwards, when it has been given a forward motion?

We can give an object a motion with respect to another object. We use the SETRELATIVEMOTION command to do this. A demonstration to illustrate this is:

```
TO DEMO.RELATIVE  :VIEWFROM
  MAKETURTLE  "A  0  40  90  :RED
  MAKETURTLE  "B  0  20  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [ ]
  SETRELATIVEMOTION  "B  "A  [FORWARD  10]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [A  B  EARTH]
END
```

In this example, object B is moving *relative* to object A at a speed of 10 units. Both are initialized to move horizontally across the screen. Exercises allow the student to investigate what happens to object B, when object A's speed is varied.

4.2.2 Non-parallel Motion

Many standard relative motion problems involve objects moving linearly at angles to one another. The four basic steps are again used in setting up demonstrations to illustrate this. For example, two people, A and B are moving at right angles to one another:

```
TO DEMO.ANGLE  :VIEWFROM
  MAKETURTLE  "A  0  0  0  :RED
  MAKETURTLE  "B  0  0  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [FORWARD  3]
  SETMOTION  "B  [FORWARD  4]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [A  B  EARTH]
END
```

Here A is moving North, with a speed of 3 units, and B is moving East, with a speed of 4 units. The computer simulation allows the student to see the motion of B as seen by A, and vice versa. The simulations can be superimposed on the screen allowing the student to determine that the motion of B seen from A, is opposite to the motion of A seen from B.

A set of exercises asks the student to look at this analytically and to verify the results using the computer simulation. Additional exercises are included to give the student practice with the new representation.

4.2.3 Motion Under Gravity

Students often misinterpret the motion of falling objects, especially when an object is dropped by a moving carrier. This is a common problem, even among students who have taken physics courses, including Newton's laws of motion. This section is designed to allow students to investigate falling objects and throwing objects. The microworld allows the student to explore the motion from different reference frames, and to investigate the difference between objects falling from stationary and moving carriers. For example,

- How does a person see a penny fall when s/he drops it?

- How does an independent observer view the scene?

A demonstration to illustrate dropping is:

```
TO DEMO.DROP  :VIEWFROM
  MAKETURTLE  "PENNY  0  0  0  :GREEN
  MAKETURTLE  "PERSON  0  0  90  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "PENNY  [FALL]
  SETMOTION  "PERSON  [FORWARD  5]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.FALL  "PENNY  "PERSON
  MOVE  [PENNY  PERSON  EARTH]
END
```

Other suggested investigations include:

- An apple falls from a tree. How does a person have to move to see the apple move towards himself/herself?

- If two apples fall simultaneously from a tree, how does one of the apples see the other move?

- What is the path of a ball when dropped by a person moving up an inclined plane? How does the person see the ball move?

Objects thrown into the air are treated in a similar way to dropped objects. For example, a ball is tossed on a heading of 30 degrees with speed 5 units:

```
TO DEMO.TOSS :VIEWFROM
  MAKETURTLE  "BALL  0  0  30  :RED
  MAKETURTLE  "PERSON  0  0  0  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "BALL  [TOSS  5]
  SETMOTION  "PERSON  [ ]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.TOSS  "BALL  "PERSON
  MOVE  [BALL  PERSON  EARTH]
END
```

Many explorations are possible using the outline for tossing objects:

- How does the tossed ball see the person move? The student can use the computer simulation to verify his/her analytic description.

- Compare the distance traveled horizontally and vertically, by two balls, one thrown by a stationary person, and the other thrown with the same speed and heading by a moving person.

- Compare simultaneously dropping a ball and projecting a similar ball horizontally from a tower.

- Investigate throwing from moving vehicles.

- Determine the path of water coming out of a hose as it falls to the ground, when the hose is stationary and when the hose is moving.

- Project balls simultaneously and look at the motion of one seen by the other.

4.2.4 Circular Motion

Very interesting examples of relative motion occur when objects move with circular motion. This is a new domain for many students, as the natural occurrences of relative rotational motion, such as the retrograde motion of the planets, is only visible over a long period of time, and the effects due to the rotating earth are negligible for most everyday occurrences of motion. The microworld provides a framework for a wide variety of exploration using circular motion.

Circular motion can be accomplished by giving an object a **FORWARD** and a **RIGHT** motion to be repeated during each time interval. For example, **FORWARD 5 RIGHT 5.** The object will move on a circular path when repeatedly given this motion.

A basic demonstration to illustrate setting up two objects moving on circular paths, and to view these objects from a specified frame of reference is:

```
TO DEMO.CIRCLE :VIEWFROM
  MAKETURTLE  "CIRCLE1  (RADIUS 5 5)  0  180  :GREEN
  MAKETURTLE  "CIRCLE2  (RADIUS 5 10)  0  180  :RED
  MAKETURTLE  "ORIGIN  0  0  0  :YELLOW
  SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT 5]
  SETMOTION  "CIRCLE2  [FORWARD 5  RIGHT 10]
  SETMOTION  "ORIGIN  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [CIRCLE1  CIRCLE2  EARTH]
END
```

Before setting up objects moving in circles the student is asked to investigate how altering the amount given to the **FORWARD** command affects the size of the circle drawn, and then how altering the amount given to the **RIGHT** command affects the size of the circle drawn. The radius is directly proportional to the forward amount and inversely proportional to the right amount.

To initially set up circles to move concentrically, the most convenient position to

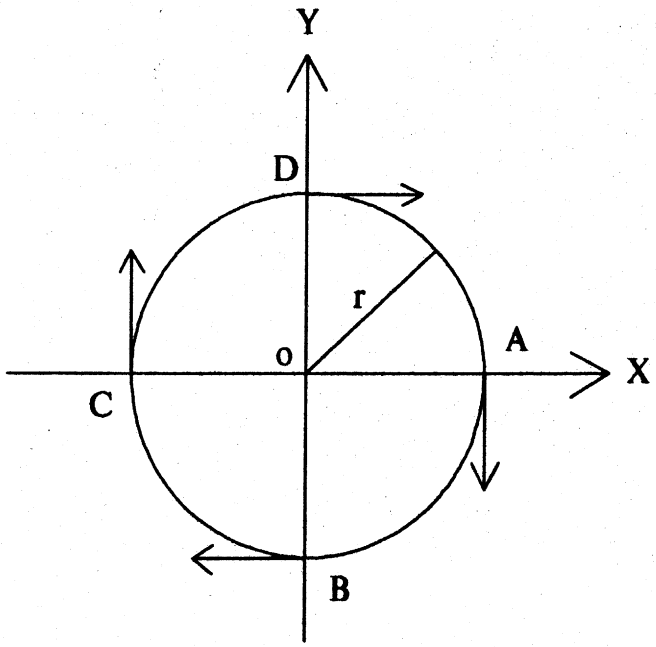

Figure 4-2: Setting up an object to move in a circle

choose is where the circle crosses either the x-axis or the y-axis. This is shown in Figure 4-2.

Position A in Figure 4-2 has coordinates [*r*,0], where *r* is the radius of the circle. A turtle moving in a circle, given by the state-change operators **FORWARD** and **RIGHT**, has a heading of 180 degrees at position A. The student is asked to verify this heading by drawing a circle with the turtle showing on the screen. For example:

```
REPEAT 360 [FORWARD 1 RIGHT 1]
```

This command draws a circle. The circle is drawn slowly, as the command has to be repeated 360 times to complete the circle. The heading of the turtle can easily be verified by watching the moving turtle.

If position B is chosen as an object's initial position, its state is given by: position [0,-*r*]

and heading 270 degrees.

In this demonstration, **DEMO.CIRCLE**, each object is set up at its equivalent position to position A as shown in Figure 4-2. The radius of each circle is determined by a procedure **RADIUS**, which calculates the value of the radius formed by the motion **[FORWARD 5 RIGHT 5]** for CIRCLE1, and **[FORWARD 5 RIGHT 10]** for CIRCLE2. **RADIUS** requires two input parameters: the amount given to the forward command and the amount given to the right command.

This basic demonstration is used with slight variations for most of the investigations of circular motion.

Viewing moving objects from other moving objects often produces surprising results. The beginning exercises in this section allow the student to use the new representation to analytically understand the motion seen on the screen, and to investigate the motion from different reference frames.

4.2.4.1 The Solar System

The solar system provides a natural example of objects moving in concentric circles at different rates. One of the most interesting examples of this is the retrograde motion of the planets, as seen by the Earth. The retrograde motion of a planet is when the planet appears to reverse its direction of motion, then slow down, stop and then continue in its original direction of motion. Loops are formed on the planet's path, as shown in Figure 4-3. These loops are called epicycles.

A procedure **SCALE.SOLAR** allows the student to set up the planets in the solar system, so that their distance from the Sun is scaled to fit on the computer screen, and they move with the correct period of revolution. For example,

```
SCALE.SOLAR [EARTH MARS]
```

returns the following information:

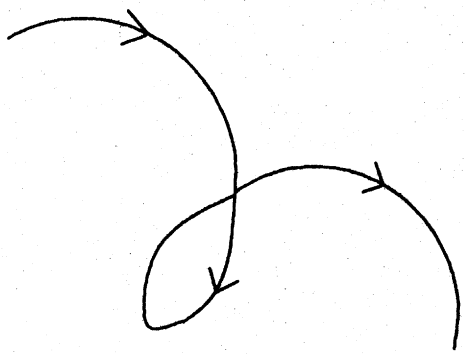

Figure 4-3: Retrograde motion

```
EARTH:  radius 59  forward 6  right 6
MARS:  radius 90  forward 5  right 3
```

This means that the Earth must have an orbit of radius 59 units, and its circular path about the Sun must be made using **[FORWARD 6 RIGHT 6]**. Similarly, Mars must be set up with radius 90, and have a motion given by **[FORWARD 5 RIGHT 3]**.

```
TO DEMO.SOLAR  :VIEWFROM
  MAKETURTLE  "EARTH  0  59  90  :GREEN
  MAKETURTLE  "MARS  0  90  90  :RED
  MAKETURTLE  "SUN  0  0  0  :YELLOW
  SETMOTION  "EARTH  [FORWARD 6  RIGHT 6]
  SETMOTION  "MARS  [FORWARD 5  RIGHT 3]
  SETMOTION  "SUN  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [EARTH  MARS  SUN]
END
```

Some suggested explorations include:

- What is the path of the Sun seen from the Earth?

- Do all the planets show retrograde motion as seen from the Earth?

- Does the Sun move with retrograde motion as seen from the Earth?

- What position does a planet have to be in to produce retrograde motion?

- How is the retrograde motion of an outer planet from the Earth different from the retrograde motion of an inner planet?

4.2.4.2 Examining Circular Motion

Exercises have been designed to enable the student to investigate various aspects of circular motion. For example:

- Given circles made from motions of **FORWARD** and **RIGHT** develop a theory to decide which is the largest circle.

- If two circles are set up, and one circle's path is drawn in twice as fast as the other, why is only one epicycle loop seen, and not two, when the motion of one circle is seen from the other?

- Not all circles produce epicycles when the motion of one is viewed from the other. Some produce epicycles as shown in Figure 4-4 and others produce cusps as shown in Figure 4-5. Determine what conditions are necessary to produce epicycles rather than cusps.

- Verify that the theory developed for epicycles and cusps holds for the solar system.

- Set up a system of orbiting planets which have moons. (The SETRELATIVEMOTION command can be used to set up a planet's moon.) What is the path of a moon as seen by a sun at the center of the

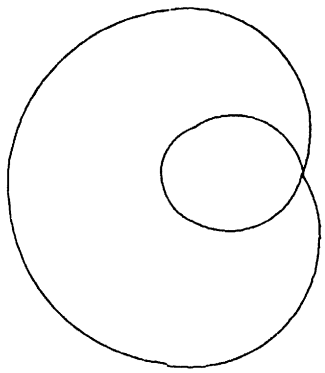

Figure 4-4: A track with one epicycle loop

system?

- What is the path of our Moon, as seen by the Sun?

4.2.4.3 Rotating Reference Frames

The final section takes a closer look at the consequences of being in a rotating frame of reference. Our example is one of a person standing on a rotating merry-go-round and throwing a ball to a partner at the center of the merry-go-round. A demonstration to illustrate this is:

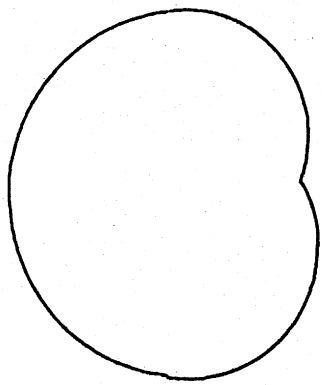

Figure 4-5: A one cusp track

```
TO DEMO.PUSH :VIEWFROM
  MAKETURTLE  "PERSON (RADIUS 5 5)  0  180  :GREEN
  MAKETURTLE  "BALL  (RADIUS 5 5)  0  270  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "PERSON  [FORWARD 5  RIGHT 5]
  SETMOTION  "BALL  [PUSH 5]
  SETMOTION  "EARTH  [ ]
  INIT.PUSH  "BALL  "PERSON
  MOVE  [PERSON  BALL  EARTH]
END
```

A merry-go-round is a good illustration of the rotating Earth. The discussion is extended to include the rotating Earth and how corrections must be made for being in a rotating reference frame. Some investigations using **DEMO.PUSH** include:

- Throwing balls to a partner on a merry-go-round and investigating how the balls should be thrown so they can be caught by another person.

- Throwing a ball so that it can be caught by the person throwing it.

- Letting a piece of ice move on a rotating surface.

- Throwing two balls simultaneously, and looking at the motion of one from the other.

4.3 The Student Text

The student text is a comprehensive workbook that introduces the student to the concepts of relative motion and frames of reference. The text assumes that the student has previously used Logo. However, the programming involved is minimal, so it is not necessary that a student invest a great deal of time in learning Logo before using the microworld. The student only has to use four basic commands to set up a motion simulation on the computer.

The student text is divided into three sections each covering a main area of investigation. These areas are rectilinear motion, motion under gravity and circular motion. The material is presented so that there is a natural progression in the level of complexity as the text is worked through.

The previous section outlined some of the possible investigations using the microworld. The sample programs are taken directly from the text. In each main area the text gives sample programs so that students with little or no programming experience will not be overwhelmed and will have a base to work from.

The first section of the text leads the student through a discussion of frames of reference. It uses as an example the motion of two cars traveling on a highway, and how we can describe their motion seen from the side of the road and from both of the cars. Students are usually familiar with examples similar to this. It is important at this stage for students to grasp the concept of a frame of reference, and how the motion of a moving object may look entirely different when seen from different perspectives.

In this first section the text outlines how we can use turtle geometry to look at motion in discrete intervals of time and how we can build up the path an object moves on by continuously repeating the given motion. This discussion is extended to show how we can use this representation to model relative motion. The computer simulation involves setting up objects on the screen, giving each object a motion, choosing a reference frame and moving the objects. The text uses an example to introduce each of these key primitives.

The exercises are straightforward in this first section. They are designed to allow the student to become familiar with the computer-based microworld, the steps necessary to initiate motion and the motion as seen on the computer screen. The questions may appear to be simple, but it is necessary that students understand the representation for motion, the motion shown on the computer screen and how we can choose different reference frames to view the motion.

As a final example in the first section the text covers non-parallel rectilinear motion. Most traditional relative motion courses ask students to solve problems dealing with this type of motion. One of the objectives of this thesis is to enhance students' reasoning capabilities so that they are better able to solve traditional problems. It is important to forge a link between a student's formal training and this more intuitive approach to motion. In addition, non-parallel rectilinear motion provides an excellent framework for illustrating how we model relative motion using the new representation. It is particularly important for students to feel confident with this representation before attempting the later sections of the text dealing with non-uniform motion.

The second section of the text discusses motion under gravity. A large percentage of students incorrectly answer questions dealing with problems of dropping. Students often cannot fit these questions into the traditional framework used to solve relative motion problems and they revert to an intuitive solution. Unfortunately, many students have incorrect notions about motion, with the result that they answer these questions incorrectly. This section attempts to help students form correct intuitive notions about motion, by allowing them to experiment with different scenarios of dropping and throwing objects.

The new representation for motion provides a simple method for analyzing this motion.

Circular motion is an extremely rich domain for discussing relative motion. The solar system and the rotating earth both provide natural examples of relative rotational motion. The representation for motion allows us to very easily determine the relative rotational motion of one object from another. The text discusses using turtle geometry to draw circles and how the state-change operators, FORWARD and RIGHT, have an effect on the size of the circle drawn.

Exercises lead the student through setting up the planets in the solar system, and understanding how the phenomenon of the retrograde motion of the planets occurs, as seen from the Earth. The microworld includes procedures to scale the planets so that a representation can be accurately simulated on the computer screen. Additional exercises allow the student to discover a theory for predicting whether the relative motion of an orbiting planet seen from another will produce a track with epicycle loops, as seen in retrograde motion, or will produce a cusp shaped path. The text extends this theory to look at the Earth-Sun-Moon system, and to predict the path of the Moon as seen by the Sun.

To complete this section on circular motion the text looks at what happens when we stand in a rotating frame of reference and try to perform some actions. A merry-go-round is used as our rotating reference frame. The text discusses what happens when we release a piece of ice from a position on the rotating merry-go-round and how we see the ice move when we view it from the merry-go-round and from a stationary position above the merry-go-round. This provides a good example of a non-inertial reference frame and how we must "invent" a force to make Newton's laws consistent.

Problems dealing with circular motion are often not attempted in a more traditional relative motion course and when they are, students are usually thoroughly confused by the methods used to solve the problems. Circular motion provides us with extremely interesting, open-ended investigations. More advanced students will find this a fascinating and challenging area to explore. The representation used in this treatment of relative motion provides us with a straightforward way of specifying how each object moves and

how we can determine the path of one object seen by another. This section gives the student a thorough exposure to rotational motion and why we see objects move as they do.

The complete student text is given in Appendix A.

4.4 The Teacher's Reference Manual

The teacher's reference manual is a guide to help teachers use the microworld. It contains hints for solving the problems and complete solutions to all the problems. Some problems require writing Logo procedures to implement the simulation on the computer. In these cases the coding for the computer simulation is included.

The teacher's reference manual is given in Appendix B.

Chapter Five

Using the Microworld

I tested the relative-motion microworld in small tutorial groups and in a large classroom setting. The purpose of working with students is to check on the content and quality of the microworld and student text. Is the microworld easy to use? Is the software reliable? What are the difficulties encountered with the microworld? What improvements can be made to the student text? Do the exercises help the student come to an understanding of the concepts of relative motion and frames of reference? What areas do students find confusing? In addition, I hope to show that the new representation for motion, which has its roots in turtle geometry, is a general, robust and easily understandable representation.

In evaluating the microworld I used four methods: personal observation of the students, discussion with the students, administration of a screening test before and after working with the microworld, and conducting a student survey. The screening test covers sample questions in each of the key areas of study from the student text: linear motion, non-parallel linear motion, dropping and throwing objects under gravity, and circular motion. This chapter includes these questions in Section 5.2. The student survey consists of questions pertaining to a student's overall impression of the microworld. Did the student do the reading assignments? Were any parts of the text confusing? Was the text useful in the course? Which exercises did the student find the easiest? Which exercises did the student find the hardest? What did the student most like about the course? The complete student survey is listed in Section 5.3.

I conducted the microworld evaluation in three phases. In the first phase I used the microworld with a tutorial group of two students. The results of this phase highlighted several problem areas. An extensive revision of the text followed. The second phase in the evaluation process used a high-school physics class to work with the microworld over a

period of one week during the regularly scheduled class time. A small tutorial group worked with the material in the final phase of testing. Each phase allowed the material to be used from a slightly different perspective. I made revisions to the student text after each phase, the most extensive ones taking place after working with the first tutorial group.

5.1 Working With Students

Two tutorial groups used the relative-motion microworld: one group of two students from the Belmont High School and a second group of three students from the Brookline High School. A class of twenty-two students, mainly seniors from the Brookline High School, used the microworld during their regular physics classes.

5.1.1 Tutorial Groups I

The first tutorial group consisted of two students, a ninth grader and a tenth grader, from the Belmont High School. Neither student had taken any physics courses. Both students had previously used personal computers, at home and at school, and one of the students had worked a little with Logo. The students were given the microworld screening test before the microworld was introduced. Both students answered all of the questions incorrectly.

Seven tutorial sessions were conducted, totalling nine hours of microworld use and physics instruction. The Macintosh 512K personal computer, using LCSI Logo, was used for the sessions. These sessions formed the most comprehensive use of the microworld and they provided much valuable information concerning problems with the student text and the exercises. Most of the exercises were attempted, although some were omitted due to the students' limited exposure to physics. This tutorial group was invaluable in providing feedback on the microworld before it was used in a more formal school setting. The first session covered the Macintosh system, using Logo and an introduction to frames of reference. The other sessions began with a review of the concepts previously studied, followed by a discussion of the new concepts being introduced during the current session. The students then performed independent explorations using selected exercises as

guidelines. We discussed their work at the completion of each set of exercises. The students prepared for each session by reading ahead in the student text. Problems encountered were discussed at an appropriate point during the next session. We worked through the entire module in a systematic way, but excluded some of the more advanced exercises.

I will now outline some of the students' work. The first session included an introduction to Logo and turtle graphics on the Macintosh. I introduced the concept of procedures, using as an example a procedure to draw a square on the computer screen. I extended this procedure definition to include input parameters to make the procedure more general. Both students quickly became proficient using the Macintosh system: moving between windows, using turtle graphics commands and using the editor to define and redefine procedures.

We initially set up the procedure DEMO. This procedure creates two objects, a walker and a runner, and moves them horizontally across the computer screen. The grade 10 student wrote her own procedure for this example using the primitives as described in the student text. The procedure was correct. Both students understood the four basic building blocks to simulate motion and how we could include all of the primitives in a procedure. The grade 9 student did not understand running the procedure DEMO with *three* input parameters. The input parameters were the frame of reference, the runner's speed and the walker's speed. I later changed the text to step through this initial example, first without input parameters, then with one input parameter, the frame of reference, and finally with three input parameters. This more gradual introduction helps a student who has little or no exposure to Logo understand the important concept of input parameters. Most of the procedures used throughout the text require one or more input parameters.

The first set of questions, Exercises 2.2.2, allow the student to become familiar with the system and the visual representation of the motion on the computer screen. The students typed in and defined the procedure DEMO to do this. These exercises underline the fundamental concepts that a frame of reference is stationary and the movement of all other objects is the motion as seen by the reference frame object.

The students used the command DEMO "EARTH 10 5, to show how the runner, moving with a speed of 10 units, and the walker, moving with a speed of 5 units, appear from the earth as the reference frame. They made numerous spelling mistakes. The students copied the procedure DEMO from the student text. The input variables for the speeds of the runner and walker were represented by the variables *:runner.speed* and *:walker.speed*. These long names resulted in several spelling mistakes. I used shorter variable names after this.

Finally, the command DEMO "EARTH 10 5 produced some results: a cross appeared in the center of the screen and two horizontal lines were gradually drawn across the screen. I asked the students to verbalize what they saw on the screen. One student pointed to the screen "this is the RUNNER and this the WALKER". I asked them what the cross at the center of the screen represented. They had to be helped through an explanation that the cross represented the reference frame they had chosen to view the motion from.

The students then completed the exercises in this first section without any assistance. They understood each simulation they saw on the computer screen. For example, DEMO "RUNNER 10 5, shows how the RUNNER sees the WALKER and the EARTH move. The WALKER moves backwards with a speed of 5 units and the EARTH moves backwards with a speed of 10 units.

> Why are two lines being drawn on the screen?...Look, there's the RUNNER (pointing to the cross at the center of the screen)...There's the WALKER...What's this line?...Oh, it's the EARTH (checking the pen width definition given in DEMO)...Why is the EARTH moving?...That's because when you drive in a car you can think of the earth moving backwards passed you.

I learned from this first exercise that procedures to outline demonstrations should keep variable names as short as possible, to minimize errors. The procedure DEMO took a while to get going. This did not discourage the students. It gave them good practice in in interpreting error messages (in Logo they are fairly straightforward) and using the editor. The students definitely had a sense of accomplishment when the demonstration worked. I decide to cut down on the amount of typing the students had to do in the future, by

providing the demonstrations outlined in the text on a disk file. Too many simple spelling mistakes could waste time, discourage the students and detract from the main purpose of using the system to study relative motion. Both students said they found it helpful to have the reference frame indicated on the screen so that they could imagine themselves "standing" on the cross to view the motion.

The third session introduced non-parallel motion. Both students understood the concept of building up the apparent path of an object by looking at what happens during discrete intervals of time. I used the example, as illustrated in DEMO.ANGLE, of two people moving at right angles to one another. However, both students were skeptical that we would produce the suggested path when viewing from one object. The students set up this example on the computer, first viewing the motion from the Earth as reference frame and then from a chosen object. The motion was along the path we had originally suggested. Being able to see the motion in this way made the representation a concrete one for the students. They were immediately able to answer that the movement of object A, seen from object B, is opposite to the movement of B, seen from A. They proved this by superimposing the motion seen from A and from B on the computer screen.

The exercises on non-parallel motion include an example of two objects moving in zig-zags. This example is one to work through without using a simulation. I initially had to remind the students to discretize the motion. They quickly completed the question, and then extended it to have one object move at twice the speed of the other object. They obtained the correct solution.

We spent the fourth session looking at motion under gravity. I asked both students the following question:

> If I stand still and watch you drop a penny while you are running, what path do I see the penny take in falling to the ground?

One student drew a backwards parabolic curve and the other a line straight down. Neither student could verbalize why she thought the penny moved as she had shown on her diagram.

We spent some time talking about gravity, Newton's laws of motion and splitting motion into perpendicular components. Neither student had taken a physics course at school, or had been exposed to resolution in mathematics classes. I then ran the demonstration using DEMO.DROP "EARTH, to illustrate these concepts and to show the path of the penny in falling to the ground. The grade 9 student then asked "how does the person dropping the penny see it move?" I replied "can you work it out?" The simulation on the screen, from the earth as the reference frame, showed the penny falling in a parabolic curve and the person moving horizontally. The student's immediate response was "the penny falls straight down". Here we see that the student is developing the ability to switch reference frames. The student was able to watch the simulation shown from the earth's frame of reference and imagine herself as the person dropping the penny. I then suggested that she verify this for herself by running DEMO.DROP "PERSON.

The students worked through several of the questions in the gravity section. They needed some help in interpreting the motion of a falling apple, when the scene was viewed by the apple. Apart from this, some problems occurred because of their limited experience in physics, and others because of the questions themselves. If they needed help in interpreting a question then I would later work on rewording that question.

The final three sessions were spent working with problems of circular motion. This is the most challenging area of the relative-motion module.

The questions included in Exercises 4.4.1 deal with the effect of changing the input to the FORWARD command and then changing the input to the RIGHT command. Circular motion is formed by repeating a compound command [FORWARD *:f* RIGHT *:r*]. The exercises suggest making procedures, CIRCLEA and CIRCLEB, to save on typing. CIRCLEA takes one input, the amount given to the FORWARD command. CIRCLEB takes two inputs, the number of times it is necessary to repeat the compound command to turn through 360 degrees and the amount to turn through on each step. The students ran both procedures:

CIRCLEA 1...CIRCLEA 2...CIRCLEA 5...CIRCLEA 10... We get bigger circles made.

CIRCLEB 72 5...CIRCLEB 36 10...CIRCLEB 18 20...CIRCLEB 9 40... We get bigger circles drawn...No...Smaller circles for bigger values given to the RIGHT command.

Neither student could verbalize this to say "the radius of the circle is *proportional* to the amount given to the FORWARD command and is *inversely proportional* to the amount given to the RIGHT command". I later modified the questions to specifically ask for the relationship between the radii of the circles drawn. Again, if a question needed clarifying I used my explanation as a guide for rewording that question.

Both students had some difficulty in setting up concentric circular motion. They understood where to initially position the objects and the heading they should choose for each object. Difficulty arose when I showed them the coding for DEMO.CIRCLE, which includes lines of code similar to:

```
MAKETURTLE  "CIRCLE1  (RADIUS  5 5)  0  180  1
    .
    .
SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT 5]
```

Here the turtle is set up at x = radius, y = 0. The motion to draw this circle is [FORWARD 5 RIGHT 5]. Both the FORWARD and the RIGHT commands have an effect on the radius of the circle. RADIUS *:f :r* returns the radius of the circle made from a motion of [FORWARD *:f* RIGHT *:r*]. Using a procedure directly as an input parameter was a concept neither student initially understood. To understand this they checked the value of each circle's radius by drawing out several circles and then printing the value returned by the RADIUS procedure:

```
PRINT  RADIUS 5 5
PRINT  RADIUS 5 10
```

Exercise 4.2.3 used the procedure DEMO.CIRCLE as a base for exploration.

DEMO.CIRCLE "ORIGIN... This is CIRCLE1 (pointing to the outer circle), and this is CIRCLE2 (pointing to the inner circle).

DEMO.CIRCLE "CIRCLE1... Look, this is the same as the diagram in the text (pointing to CIRCLE2)... What's this other circle?.. It's the ORIGIN (checking the penwidth against the coding of DEMO.CIRCLE)... Why is the ORIGIN moving in a circle?

The students cleared the graphics screen and repeated the demonstration. I suggested they think about the Earth orbiting the Sun and how we see the Sun move from the Earth, to see if they could find any similarities.

That's right... We have the Sun with the Earth moving around it... But we see the Sun going around us... I see.

I then asked the students to draw CIRCLE1 orbiting the ORIGIN, and using this diagram determine how CIRCLE1 sees the ORIGIN appear to move over small intervals of time. Both students could use this representation correctly to map out the ORIGIN's path seen from CIRCLE1.

The section dealing with the retrograde motion of the planets was a popular unit. Both students were fascinated to see the motion of various planets viewed from the earth. While we were discussing the motion and setting up the simulation, the grade 10 student pointed out that the loops would be made when the planets passed each other. Both students could verbalize why the Sun did not move with retrograde motion as seen by the Earth, and outlined the work illustrated above to show this.

Exercises 4.4.1 begin a section on developing a theory for the shape of an object's apparent path. When we set two objects moving in circles and view the motion of one object from the other, under certain conditions we see a path with epicycle loops made, under other conditions a path with cusps. The questions in this exercise deal with the relative sizes of the circles and the number of loops we see formed.

CIRCLE1 (made from FORWARD 5 RIGHT 5) is bigger than CIRCLE2 (made from FORWARD 10 RIGHT 20) because CIRCLE1 has a ratio of 5:5 and CIRCLE2 a ratio of 10:20.

CIRCLE1 (made from FORWARD 10 RIGHT 10) has a ratio of 1:1 and CIRCLE2 (made from FORWARD 5 RIGHT 5) has a ratio of 1:1... Are they the same size? (The students used the turtle and drew out the circles to verify this.)

One question deals with two circles, one being drawn four times quicker than the other. The question asks for an explanation of why three epicycle loops are formed and not four. The students edited DEMO.CIRCLE to give the circles the motions as specified in the question.

> DEMO.CIRCLE "ORIGIN... Here's CIRCLE1 (FORWARD 10 RIGHT 20) (pointing to the outer circle)... No... That's CIRCLE2 (FORWARD 5 RIGHT 5)... Why?.. Because CIRCLE2's ratio is 5:5 and CIRCLE1's is 10:20... CIRCLE1 is drawn in four times faster... Why?.. It's turning through 20 degrees on each step, four times more than CIRCLE2...
> DEMO.CIRCLE "CIRCLE1... Three loops are formed... That's strange... Why?

The students could not resolve this question. So I gave them a hint to think about the positions that the objects are in when the loops are formed. The students drew the circles out on paper and indicated where the objects would be at given intervals of time. They soon realized that the objects only passed each other three times. When the faster one has completed one revolution, the slower one has moved a quarter revolution. By the time the faster one catches up with the slower one again, the slower one has moved a short distance.

The final section of the text deals with rotating frames of reference and uses a merry-go-round as an example of such a reference frame. Both students believed that when an object is moving in a circle and is released it moves on a curved path. I asked them to perform DEMO.PUSH "EARTH. This demonstration releases a piece of ice from the surface of a rotating merry-go-round. "The ice moves in a straight line." I asked them to draw a circle on the computer screen with the turtle showing, and then to draw, on paper, a diagram of the turtle's heading as it moves around the circle. "Which way is the turtle heading when you release the ice?" I reminded them of Newton's First Law, to illustrate that the ice continues to move in the direction it is heading when released, so its path will be a straight line. The students experimented with rotating reference frames using Exercises 4.5.2.

One question in Exercises 4.5.2 deals with throwing a ball so that it remains stationary. The students set up the ball giving it a "push" in the direction opposite to the ball's motion inherited from the merry-go-round. How can you throw the ball so that it can be caught by

your partner? This was answered by trial and error. The students clearly understood that the resultant motion of the ball was the combination of the "push" and the motion from the merry-go-round.

Using the microworld with this first tutorial group was very successful. The students gave many suggestions for improvements, and in addition, gave me a different perspective on the material being presented. Both students learned a lot of physics using the microworld. They worked extremely hard, both during the tutorial sessions, and at home reading the student text. Listed here is only a fraction of their work. The students were highly motivated by the computational environment. Neither student had used a computer in this way before. The students completed the screening test one week after they finished using the microworld. Both students answered all but the last question correctly. Previously all had been answered incorrectly. The last question, dealing with the path of a point on a bicycle wheel, as the bicycle moves in a straight line, is a difficult one. Many graduates in physics answer this incorrectly! The dramatic increase in the number of correct solutions given to the screening test is a clear indication of the effectiveness of using a well designed computational environment as a tool to think with.

5.1.2 Tutorial Group II

The second tutorial group consisted of three eleventh graders from the Brookline High School. These students initially participated in the large classroom testing of the microworld. The Macintosh 512K personal computer, using LCSI Logo, was used during the tutorial groups. Four fifty-minute lessons were given. The first session was an overview of the Macintosh system and a review of Logo. The format followed was similar to the sessions conducted with the first tutorial group. A short introduction was given, followed by independent exploration using the exercises outlined in the text. The students looked at a selection of the material on circular motion. The students were monitored by personal observation.

The three students worked in two groups, each group with its own computer. The student working by himself was not proficient in English. I decided to give him a machine

on his own so that he could understand and use the material at his own pace. He worked slowly and steadily at the suggested exercises, and in some cases extended them. For example, in initially using Logo on the Macintosh system he wrote his own program to draw a triangle, rather than a square. With assistance, he came to understand retrograde motion and why we see the planets move with this motion from the Earth. He set up the solar system with different planets in turn and looked at their epicycle motion.

The second group consisted of two of the brightest pupils in the high-school class. During the introductory session on Logo they discovered recursion. They used the procedure to draw a square of variable size and then extended it to draw a window pane with four squares. When I looked at their work they had run SQUARE 10, SQUARE 20, SQUARE 30 without clearing the screen. They said "there must be a way of doing this automatically". I asked them to verbalize what they were doing each time. "Making the sides 10 units bigger." I suggested putting this command into SQUARE. They did this and described it as a "neat" concept. Just before the lesson ended they were busy at work with another recursive procedure to change both the angle and the side. Their teacher worked for four hours after class experimenting with this concept too!

The final sessions all ran smoothly with few problems. Both students could articulate that the radius of a circle is proportional to the FORWARD *:amount* and is proportional to 1/RIGHT *:amount*. They understood how to set up objects to move on concentric circles and how to view the objects from different reference frames. I reminded them of the representation we use to work out how one object appears to move as seen by another object.

For example, one question uses the equations:

```
SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT 10]
SETMOTION  "CIRCLE2  [FORWARD 10  RIGHT 10]
```

Here we are giving a circular motion to two objects, CIRCLE1 and CIRCLE2. The radius of CIRCLE1 has half the magnitude of the radius of CIRCLE2. When we view the motion from CIRCLE1, CIRCLE2 and the ORIGIN appear to be moving in the same orbit. From

this reference frame is there a way to show that CIRCLE2 and the ORIGIN are not actually in the same orbit?

> This is CIRCLE1 (pointing to the inner circle)... This is CIRCLE2 (pointing to the outer circle)... When viewed from the ORIGIN the circles complete their orbits at the same rate... (Draws out diagram.) If we view from CIRCLE1, we see the ORIGIN orbiting in a circle, and we see CIRCLE2 as a circle too, with exactly the same radius as the circle formed by the ORIGIN... Therefore, there is absolutely no way, without some outside knowledge, to prove that CIRCLE2 and the ORIGIN are not moving on the same path.

The retrograde motion of the planets was again a popular unit. The students set up several planets to check on this phenomenon. They used the procedure SCALE.SOLAR to determine the motion they should give to each planet to correctly represent the planet in the solar system. One student pointed out that SCALE.SOLAR was returning "crazy" numbers:

```
SCALE.SOLAR  [JUPITER EARTH]
```

returned

```
Jupiter : radius 467 forward 0 right 0
Earth : radius 90 forward 5 right 3
```

Here there is a problem with SCALE.SOLAR. The procedure assumes the list of planets as input is ordered from the inner planet to the outer planet. I later corrected this to accept the input list of the planets in any order.

Both students understood retrograde motion; for example, "how is the retrograde motion of an inner planet, seen from the Earth, different from that of an outer planet?"

> There are less loops in the path. This is because there is only one certain position at which the loop occurs in. In the motion of the Earth and an outer planet, the Earth is moving faster, and passes the planet several times, causing several loops. The Earth, for example, passes Jupiter about eleven times, while Jupiter makes one revolution about the Sun.

The students used their knowledge about epicycle loops to answer the question of why three epicycle loops are made when one circle is being drawn in four times faster than the

other. The circles have the following motions:

```
SETMOTION  "CIRCLE1  [FORWARD 10  RIGHT 20]
SETMOTION  "CIRCLE2  [FORWARD 6  RIGHT 6]
```

It makes sense that there are only three loops. This is because during their cycles they only pass each other three times. Therefore, only three loops occur. The reason that there are only three loops is because we are dealing with objects that are in motion.

This section completed the time spent with the second tutorial group. I made some clarifications to the questions and some coding changes, as detailed earlier. The students' teacher continued working through the student text. She was extremely motivated by the work that the students did and she wanted to continue further on her own. Although I do not have her work documented here, her feedback on using the microworld was extremely useful. Her main criticism, dealing with the later circular motion sections, was that the material could possibly be too challenging for the average student, with the result that the student could soon loose sight of where s/he is going. I rewrote the section, Epicycles and Cusps, to take her comments into account. I gave more hints, broke the questions down into smaller units, and clarified the questions further.

5.1.3 The High-School Class

I used the relative-motion microworld in a large classroom situation with a group of twenty-two students at the Brookline High School. The students were enrolled in the standard physics course, a course in physics for students of moderate mathematical and scientific ability. The class itself spanned a broad range of ability. The students consisted of seniors, with the exception of three eleventh graders.

I organized the class on a more formal basis than the tutorial groups because of its size. The teacher in charge of the class assigned a grade to each student and required that each student hand in completed exercises. I used the microworld over the course of one week, during the regularly scheduled physics classes. There were three fifty-minute sessions and a final session of one hundred minutes. The Apple II 64K personal computer, using

Terrapin Logo, was used to conduct these sessions. The students worked in groups of two students to one machine.

The first session served as an introduction to Logo and the Apple II system. Most students had never used Logo before, although many had used a personal computer. Each session was conducted in a similar way to the tutorial group sessions: an introductory discussion followed by a period of independent working. The students' teacher and myself helped the students when they required assistance, and in addition, we tried to spend time individually with each group.

I had to overcome many logistical problems to run these sessions. Several classes were rescheduled so that the physics class could use the computer room. The computer room was small for twenty-two students. In order that students could work two to a machine, I brought extra machines into the room, crowding it further. The students had little room to maneuver. To facilitate the students completing the exercises, I made lesson plans for each day. The lesson plans outlined the necessary steps to install the microworld on the Apple II and listed the exercises for the day, with space for the answers. This reduced the amount of paper that each student had to refer to while working with the microworld.

Due to time constraints, the students did not work through all of the student text. They covered linear motion, non-parallel linear motion and motion under gravity. Although we kept discussion to a minimum, to allow more time for individual exploration and learning, the class time soon passed. The final session of one hundred minutes gave the students a better opportunity of concentrating on their explorations.

The students spent most of the first two classes learning the computer system and exploring the capabilities of Logo. The students used a procedure to draw a square on the computer screen to become familiar with the editor and the steps in defining procedures. They also used the editor to define the first demonstration, DEMO. By the end of the session all of the students were proficient at using the computer system and the editor. About two-thirds of the class ran DEMO successfully. I gave the students the remaining demonstrations on a disk file.

Unfortunately, most of the computer monitors had monochromatic screens. This made it much harder for a student to identify each object. Using this implementation of the relative-motion system each object is created with a given pen color. To overcome this problem I suggested first running the simulation from the stationary earth's frame of reference and identifying each object by noting where each object was initially positioned on the screen. Secondly, to choose an object as the reference frame and note the positions of all of the other objects in relation to this object. The objects have the same spatial relationship when the procedure is run to illustrate the motion from the chosen reference frame. This technique worked well.

I gave the students the screening test the week before they used the relative-motion microworld and the week after they used the relative-motion microworld. Eighteen students completed the pre-microworld test and nineteen the post-microworld test. The tests were given anonymously. The screening test is listed in the next section, Section 5.2. In addition, I gave a questionnaire to the students. Twenty students completed this questionnaire. Section 5.3 lists these questions.

5.2 The Screening Test

The following questions make up the screening test I gave to the students before and after they used the relative-motion microworld:

1. When you are in a car driving along the highway at 40 m.p.h. another car overtakes you traveling at 50 m.p.h. How do you see the other car move?

2. Suppose an oncoming car is driving towards you at 30 m.p.h. How do you see the oncoming car move?

3. You are in a car moving North and a friend is in another car moving East, at the same speed. Draw a diagram of how you see your friend's car move.

4. You are running and drop a quarter, draw a diagram of how the quarter moves

in falling to the ground. Indicate on the diagram where you will be when the quarter reaches the ground.

5. A relief package is dropped from a plane, when will the package have to be dropped in order to reach its target?

   (a) Before the target. (b) Over the target. (c) After passing the target.

6. If the pilot wants to keep an eye on the package where will the pilot have to look?

   (a) Straight down. (b) Behind the plane. (c) In front of the plane.

7. If someone on the ground is directly below the plane when the package is dropped, where will the person have to look to keep an eye on the package?

   (a) Straight up. (b) In the direction the plane is moving. (c) In the direction opposite to that in which the plane is moving.

8. Two similar balls are at the top of a tower. Ball X is dropped from the top of the tower and simultaneously ball Y is projected horizontally. If we neglect air resistance, which of the following statements is true?

   (a) Ball X will reach the ground first. (b) Ball Y will reach the ground first. (c) X and Y reach the ground at the same time.

9. A ball is on a flat, horizontal table and is being swung around in a circle by a piece of string attached to it, so that it always remains in contact with the table. The string suddenly breaks. Draw a diagram of how the ball moves across the surface of the table.

10. In our solar system the Earth orbits the Sun and the Moon orbits the Earth. If we could stand on the Sun and watch the Earth moving, draw the path that the Earth traces out.

11. Draw the path that the Moon traces out.

12. If you are watching a friend ride a bicycle at a constant speed, draw a diagram of the path made by a point, such as the valve on the bicycle wheel, as the bicycle moves along.

5.3 The Student Questionnaire

The student questionnaire consists of the following questions:

1. Have you used a personal computer before? If so what for?

2. Have you used Logo before?

3. Did you like this opportunity of using a computer in your physics class?

4. Did you do the reading assignments for homework?

5. Was the text useful to you in this course?

6. Could you understand the material in the text? Were any parts confusing?

7. Have you any suggestions for improving the text?

8. Did you find the exercises useful?

9. Were the exercises at an appropriate level for you? Too hard? Too easy?

10. Which exercises did you find the easiest?

11. Which exercises did you find the hardest?

12. Was the computer software easy to use?

13. Did you find anything confusing about the software?

14. Have you any suggestions for improving the software?

15. Do you think that this short course has improved your understanding of relative motion?

16. What did you most enjoy about the course?

17. What did you not like about the course?

18. Any other comments?

5.4 Summary

The two tutorial groups provided me with the main feedback on the content and quality of the relative-motion system. In particular, I found my work with the first tutorial group to be beneficial. I was able to make substantial modifications to the student text. The students were very generous with their time, allowing me to work through the entire text with them. The students were not working under any fixed time constraints; they could work for as long, or as little, as they felt inclined to. The two students from the first tutorial group achieved excellent scores on the post-microworld test, indicating that they had really come to understand motion, relative motion and frames of reference.

Testing the relative-motion microworld with a large group of students gave me the

opportunity of monitoring how students perform with the material, without them having received a great deal of personal tuition. The logistics of organizing the sessions took away from some of the available class time. For example, getting the class established in the computer room with sufficient machines took a few minutes each day. Lack of space was a constant problem. I produced work sheets so that the students did not need to constantly refer to different papers. The class time was short. Often students were just getting involved in a project when they had to leave for their next lesson.

In a large classroom situation, I feel the microworld could be better utilized as a tool at the time relative motion is covered. For example, the first section of the text dealing with linear motion and non-parallel linear motion could be used as a framework for representing and illustrating relative motion. The section dealing with motion under gravity could be used at a later time, to illustrate Newton's laws of motion and to allow the student to experiment with dropping and throwing objects. A student could use the section on circular motion for a long-term project during the semester.

Table 5-1 lists the results of the pre-microworld test and the post-microworld test given to the high-school students.

Question	Pre-microworld	Post-microworld
1	88.9	84.2
2	44.4	57.9
3	38.9	68.4
4	50.0	73.7
5	77.8	78.9
6	38.9	47.4
7	72.2	78.9
8	100.0	94.7

Table 5-1: Percentage of questions answered correctly

I have listed the results of the first eight questions only, as these questions correspond to topics covered in the class sessions. All but Questions 1 and 8 show some improvement.

I was looking for an answer in Question 1 similar to "the car appears to be moving slowly" or "the car appears to be moving at 10 m.p.h." Some incorrect responses had a horizontal line drawn on the questionnaire. In this case, the student was probably indicating what s/he would see using the simulation. This explanation could account for the lower percentage of correct responses to Question 1 in the post-microworld screening.

Question 2 shows some improvement. The students had experimented with cars moving towards one another. More students gave correct numerical responses after using the relative-motion microworld.

The students answered Question 3 very poorly before using the microworld. This question is fairly typical of the standard textbook relative motion questions. The correct number of responses to this question almost doubled after using the microworld.

Questions 4 - 7 consider dropping objects. The students' responses to Questions 4 and 5 showed that the students know that an object dropped from a moving carrier falls on a parabolic path. Many students showed the path of the falling quarter, in Question 4, correctly. However, they failed to show the correct position of where they would be when the quarter reached the ground. The increase in the number of correct responses to Question 4 on the post-microworld test shows that many students acquire a better notion of dropping from using the microworld.

Questions 6 and 7 look at the motion of a dropped package from different reference frames. There is some improvement in the number of correct responses to both of these questions, showing that some students better understand the notion of a reference frame and how to visualize motion from a chosen reference frame.

Question 8 shows a slight decrease in the number of correct responses after using the relative-motion microworld. One student failed to answer the question. This could have

been because the question was at the bottom of a page and the multiple choice answers were at the top of the next page.

The microworld survey was completed by twenty students anonymously. The survey is listed in the previous section. Seventeen students appreciated the opportunity of using a computer in their physics class, one adamantly disliked computers, and two students were indifferent.

Most students did the reading assignments prior to coming to class and found the text useful. The students suggested the following improvements to the text: more diagrams, make the text more concise, and explain editing. I changed the order of the beginning sections of the text and included diagrams to illustrate the meaning of a frame of reference. I also included a series of diagrams for each demonstration outlined in the text to show the motion, as seen by each object, in the given demonstration.

The majority of students found the exercises useful and at an appropriate level for them. Some found the introductory exercises, which enabled them to gain experience with the system, too easy. However, some students cited they were initially hampered by lack of experience with computers. The hardest exercises were the ones dealing with motion under gravity, especially the exercises to throw objects above and below the horizontal.

All the students found the software easy to use. Two students mentioned problems associated with Logo: editing and putting " before a name. Specific Logo details such as the difference between "EARTH, :EARTH and EARTH were not discussed with the students. The students followed the outlines for the procedures as given in the student text.

A range of responses was received as to what the students did not like about the course. Two students did not like the text, or having to do the reading assignments. Other responses included: relative motion had previously been studied in class, some exercises were too drawn out, the course was too easy, the course was too short. One student complained that s/he had to be absent and miss part of the course!

Most students agreed that the course had improved their understanding of relative motion. An overwhelming number enjoyed using the computer in their physics class, seeing the motion and working with the software. They appreciated having a different experience in their physics class. Overall, the relative-motion microworld was very well received.

Chapter Six

Conclusion

The relative-motion microworld is a computer environment within which the student can explore the concepts of relative motion and frames of reference. The complete module consists of the computer-based learning environment and a student text. The student text is a comprehensive guide that outlines the concepts of relative motion and frames of reference. In addition, it provides exercises for the student to explore these concepts. The student cannot easily set up laboratory experiments to demonstrate relative motion. Computer simulation provides a method of giving the student a tangible experience with relative motion. The computer simulation described here is not only a simulation, it is an open-ended learning environment. The student is in direct control of his/her learning; by experimenting, by exploring and by deciding the direction in which to pursue the investigations.

The overall objective of this work is to give the student a better notion of motion, in particular relative motion, and how moving objects appear to move in very different ways seen from different positions. Students build up intuitions about motion from daily experience. However, they cannot physically view motion simultaneously from different reference frames in order to provide a foundation for the concept that all motion is measured *relative* to a reference frame. This work allows the student to experiment and explore motion seen from different perspectives.

I believe that this microworld is successful in its objectives. The students who used the relative-motion microworld did appear to have a better grasp of motion and the concept of a frame of reference after working with the material. The students used the simulation, shown on the computer screen, together with the new representation for motion, as introduced by this work, to come to an understanding of relative motion and how objects appear to move from different reference frames. The new representation for motion is a

simple and effective one that can be used in exactly the same way whether uniform or non-uniform motion is being considered.

A computer-based learning environment supports a pupil-centered mode of learning rather than a teacher-centered mode as found in the typical classroom. This pupil-centered mode of learning allows a student to attain a much deeper understanding of the subject material. The student learns by discovery. However, learning-by-discovery can take different students or groups of students varying amounts of time to come to an understanding of the subject being investigated. The teacher must act as a consultant. This puts a burden on the teacher; s/he must know when to give hints, know when to hold back from making suggestions and be able to monitor each group of students closely. Time constraints can be a problem. A pupil-centered mode of learning needs time for the students to get involved with their work. The current segmentation of the school day into small units of time is often not conducive to this method of learning.

I hope that this microworld will serve as a model for teachers to develop their own computer-based learning environments. In designing such an environment it is important to reduce the task to a set of fundamental building blocks and once these are identified, to understand how a student can use these to explore the subject area. This can be a time consuming task; for example, the system described here took six months of effort. Creating examples and projects for the average high-school student is non-trivial. Whether the project work is designed for a few minutes or an hour it entails the designer investing a substantial amount of time.

I implemented the relative-motion system on two personal computers: the Apple II 64K and the Macintosh 512K. The Macintosh implementation was much superior. The graphics have a much higher resolution and the simulations run on average six times faster than on the Apple II. The additional speed of the Macintosh and the ease of using the Macintosh system provide a powerful framework for illustrating the concepts of relative motion and frames of reference. If it is necessary to implement the relative-motion system on a machine of similar speed to the Apple II, sections of the code could be monitored and

then, if necessary, rewritten with more efficient primitives.

Further improvements can be made to the Macintosh implementation. For example, the relative-motion system can be made user interactive. One key input commands can be given to the microworld as it is running. These can include:

- **S**: To stop the microworld simulation.

- **C**: To clear the graphics screen.

- **R**: To specify a new reference frame.

- **?**: To return the frame of reference.

The relative-motion primitive commands can be made into short-forms. For example, **MT** is the short-form for **MAKETURTLE** and **SM** is the short-form for **SETMOTION**. This would decrease the number of typing errors the students make and allow them to quickly set up their own demonstrations. I did not implement this with the high-school class because of time constraints and the Apple II 64K's memory constraints.

A further addition to the system is one to show an object's resultant relative velocity vector over time. For example, the system could draw a small set of axes, either in one corner of the graphics screen or in an additional window, and draw in the relative velocity vector with its correct magnitude on its correct heading. This could help the student merge his/her traditional vector algebra representation for relative motion with the new representation presented in this thesis. In addition, it allows the student to see the change in relative speed that can take place. For example, when two objects are moving in concentric circles and we view the path of one object from the other, the path may epicycle. In this case the object appears to slow down, then stop and then reverse its direction of motion. This slowdown is hard to see from watching the path of the apparent motion being drawn in.

An additional extension to the Macintosh implementation of the relative-motion system is one to take further advantage of the Macintosh capabilities. For example, each object could create its own graphics window, to show the motion of all the other objects. In this way the student would have a complete picture, presented on the computer screen, of the motion seen from each reference frame. The student would no longer need to clear the graphics screen and rerun the demonstration to see the motion from a different reference frame.

The relative-motion microworld is a contribution of bringing together computer technology and a scientific discipline in a meaningful way. The student is not a passive observer, but initiates his/her own explorations within the framework provided. The majority of the students I worked with found using the microworld to be a unique learning experience. Not only did they benefit from exploring an area of physics, but they had the opportunity of using current technology. As demonstrated here, the computer provides a powerful framework for developing a stimulating learning environment.

Appendix A

The Student Text

A Relative-Motion Microworld

Linda E. Morecroft
Educational Computing Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts

September 1985

Copyright (C) Massachusetts Institute of Technology 1985

Table of Contents

1 Introduction 104
2 Frames of Reference 105
2.1 A Closer Look at Motion 109
2.2 Modeling Relative Motion 111
2.2.1 A Beginning Example 115
2.2.2 Exercises 121
2.2.3 Setting Motion Relative to another Turtle 122
2.2.4 Exercises 123
2.3 Non-parallel Motion 125
2.3.1 Exercises 127
2.3.2 More on the SETRELATIVEMOTION Command 129
2.3.3 Exercises 129
3 Motion Under Gravity 131
3.1 Dropping Objects 132
3.1.1 Analyzing the Motion 134
3.1.2 Dropping from Moving Objects 135
3.1.3 Exercises 139
3.2 Throwing Objects 140
3.2.1 Exercises 142
4 Circular Motion 145
4.1 Turtles and Circular Motion 145
4.1.1 Exercises 146
4.2 Uniform Relative Circular Motion 148
4.2.1 Setting Up Circular Motion 148
4.2.2 An Example of Circular Motion 150
4.2.3 Exercises 151
4.3 Retrograde Motion 153
4.3.1 Exercises 157
4.4 Investigations of Uniform Circular Motion 157
4.4.1 Exercises 158
4.4.2 Epicycles and Cusps 160
4.4.3 Exercises 164
4.5 Rotating Frames of Reference 170
4.5.1 The Rotating Earth 172
4.5.2 Exercises 176

Table of Figures

Figure 1: Point O as the reference frame 107
Figure 2: Point B as the reference frame 108
Figure 3: The path of a point on a moving bicycle wheel 108
Figure 4: The positions of RUNNER and WALKER after intervals of time 110
Figure 5: Drawing with the turtle 112
Figure 6: Setting up turtles on the computer screen 116
Figure 7: RUNNER and WALKER as seen by the EARTH 117
Figure 8: The RUNNER and EARTH as seen by the WALKER 119
Figure 9: The WALKER and EARTH as seen by the RUNNER 120
Figure 10: Moving a circuit in a magnetic field 124
Figure 11: A and B after one interval of time 125
Figure 12: A and B after two intervals of time 126
Figure 13: The motion of B as seen by A 127
Figure 14: P and Q moving in zig-zags 128
Figure 15: The motion of the APPLE and PERSON seen by the TREE 133
Figure 16: The TREE and APPLE seen by the PERSON 134
Figure 17: Analysis of falling 135
Figure 18: Suggested paths of a penny in falling to the ground 136
Figure 19: A penny dropped by a moving person 138
Figure 20: Circles drawn by the turtle 146
Figure 21: Setting up a turtle to move in a circle 149
Figure 22: CIRCLE1 and CIRCLE2 as seen by the ORIGIN 152
Figure 23: Motion seen by CIRCLE1 153
Figure 24: Retrograde motion 154
Figure 25: An ancient model to explain retrograde motion 155
Figure 26: A track with three epicycle loops 159
Figure 27: A track with one epicycle loop 160
Figure 28: A one cusp track 161
Figure 29: A three cusp track 162
Figure 30: Orbiting Planets 167
Figure 31: Initial positions of the Earth and the Moon 168
Figure 32: A ball thrown on a moving merry-go-round seen from the Earth 172
Figure 33: The moving ball as seen by a person on the merry-go-round 173
Figure 34: The ice as seen from a stationary reference frame 174
Figure 35: The ice seen from the rotating reference frame 175

Table of Tables

Table 1: The new primitives introduced by the microworld 113
Table 2: Distances and periods of the planets 155

1 Introduction

This student workbook introduces relative motion and frames of reference. What is "relative motion"? What is a "frame of reference"? We will discuss these questions in the text and exercises. The workbook is used with a computer-based microworld which allows you to implement the exercises on a personal computer. The computer-based microworld is programmed in Logo. Students of all ages, from nursery to college level, have successfully used Logo. Do not worry if you have not used Logo or a personal computer before. You will need very few commands to implement examples on the computer. You will find numerous examples throughout the text to help you become familiar with the system and how to perform your own explorations.

This text is divided into several sections. We will first look at frames of reference and discuss relative motion. Another section covers how the computer system models relative motion and shows you the commands necessary to use the system.

You will look at different forms of motion: objects moving in straight lines, dropping and throwing objects under gravity, and objects moving in a circle. These forms of motion are very interesting, especially when you view a moving object while you are moving too. You will find in many of the suggested exercises some surprising results, which you will not have expected to happen. The text will show you a simple method for thinking about motion, so that you will understand why objects move as they do.

First let us discuss what we mean by a "frame of reference".

2 Frames of Reference

What is a frame of reference and why are frames of reference important? We will look at these questions in this section. First we will look at what a frame of reference is.

Suppose we watch two people running a race. We see the path each person takes and we see who wins the race. We describe the race as we see it. We are a *frame of reference.* If we now ride on bicycles and ride next to the faster runner, our description of the scene will be quite different. The faster runner is alongside us and the slower one a short way behind. Again we describe the scene as we see it. We are the frame of reference. The two descriptions of the race are different, but both are accurate and correct. We describe the scene as *we* see it. In the first case we are stationary and in the second case we are moving. In the following sections we will take a look at moving objects and describe their motions from different reference frames.

Let us look at an example which you are probably somewhat familiar with, that of driving along the highway, in order to further illustrate our discussion of frames of reference.

Imagine two cars traveling in the same direction down a highway, one at 30 m.p.h. and the other at 40 m.p.h. If you stand at the side of the road how do you see these two cars move? The cars travel with their given speeds of 30 m.p.h. and 40 m.p.h. If you now ride in the car traveling at 30 m.p.h. how do you see the faster car move? The faster car slowly passes you by, and very gradually it moves further and further in front of you. Because you are now moving in the slower car, the faster car does not appear to be moving at 40 m.p.h., but much slower. For every forward movement the faster car does, the slower one also moves forwards. After one hour the faster car has traveled 40 miles, and the slower one 30 miles. If you travel in the slower car, at the end of one hour, the faster car is only 10 miles away from you, not 40!

If you travel in the faster car, you see the slower car get left further and further behind. At the end of one hour the slower car will be 10 miles behind you.

Only one thing happens in this scenario, two cars are moving on a highway, yet here we have three very different descriptions. Each description is described from a different point of view, a different reference frame. First you stand at the side of a highway, then you travel in the slower car, then in the faster one to describe the scene. In the first description you are a stationary frame of reference at the side of the road. You describe the scene as you see it.

When you travel in the slower car, you are in the reference frame of the slower car, and the motion of the faster car is described as as seen from this reference frame. At the end of one hour you see the faster car 10 miles in front of you. You can say that the faster car has a speed of 10 m.p.h. *relative* to you. When you travel in the faster car, at the end of one hour the slower car is 10 miles behind you. From this reference frame the slower car is seen to move *backwards* at a speed of 10 m.p.h. *relative* to you.

Speeds are always measured *relative* to a reference frame. When we talk of a car moving at 40 m.p.h., we really mean that the car is moving at 40 m.p.h. relative to the Earth. This is not the "true" speed of the car, as the Earth itself is moving (spinning on its axis and orbiting the sun).

When you are in a moving car how do you know that you are moving? You look out of the window and see the edges of the highway moving. From the knowledge you already have about the Earth you know that the trees and road do not move, therefore you must be moving. Now suppose you are inside a smoothly moving van, with no windows, so you cannot see outside. How do you know you are moving? There is no way to do this! You cannot tell whether you are moving, or try to find out the speed at which you are moving. You see the sides of the van and they appear to be stationary. *A reference frame always appears stationary to itself.* You are in the reference frame of the van and to you it is stationary.

These examples show us that observations of moving objects are made with respect to a reference frame. In the same way, measurements are taken with respect to a reference frame. For example, in coordinate geometry, points are shown with respect to a set of axes,

and on the Earth, each city has a latitude-longitude position.

Consider setting up two particles A and B at positions [1,2] and [3,1] respectively. Figure 1 shows how we describe these coordinates with respect to the center of the coordinate system, point O. If we now describe each point from B, we stand at B and measure the displacement in X and Y of each point. One way to do this is to imagine a new set of axes passing through B, as shown in Figure 2. Using B, as the center of the coordinate system, the positions are now given by: A is at [-2,1], B is at [0,0] and O is at [-3,-1].

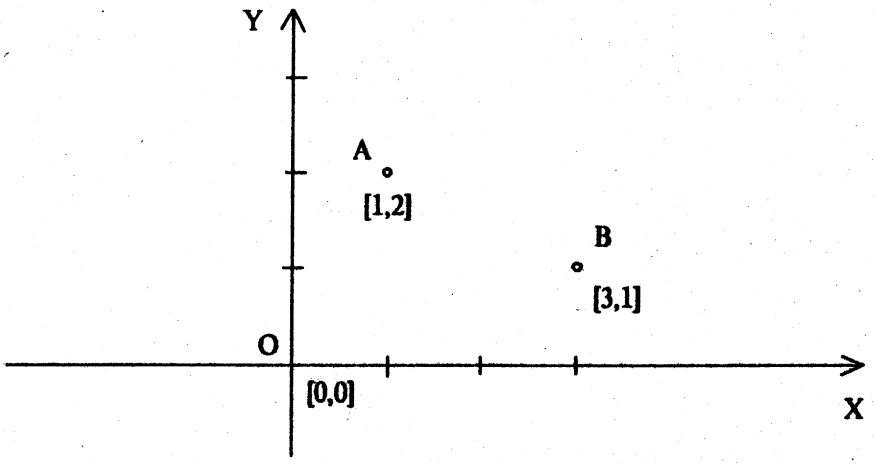

Figure 1: Point O as the reference frame

Here we see that all measurements are taken with respect to a defined coordinate system. The coordinate system is the *frame of reference.* It is *fixed* in space and all measurements are made *relative* to it.

The notion of being in a reference frame is very important in physics. In many cases problems with the real world can be simplified by choosing an appropriate frame of reference to work in. For example, consider a point on the edge of a slowly moving bicycle wheel. If you ride on the bicycle, and look down at the wheel, you see the chosen point

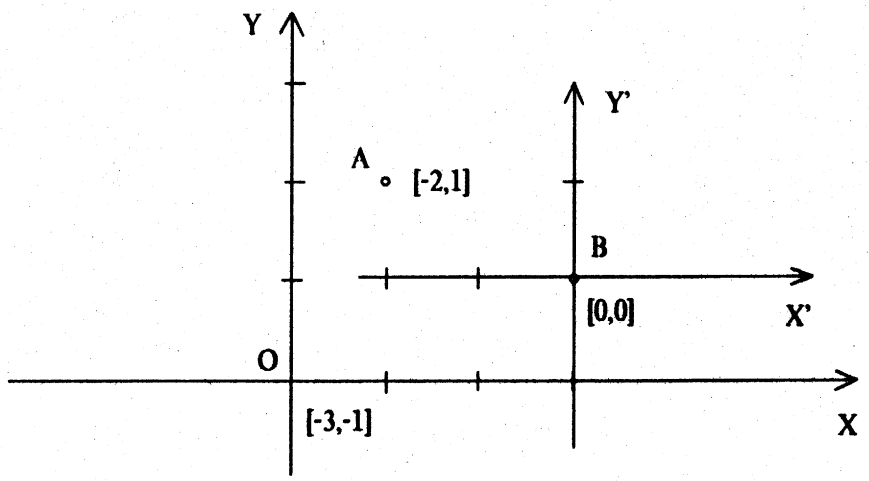

Figure 2: Point B as the reference frame

moving on a circular path. If you watch someone ride a bicycle a point on the edge of the wheel moves through a complicated curve, called a cycloid, as shown in Figure 3. Your description of the motion is simpler from the reference frame of the bicycle, where you see the point moving in a circle.

Figure 3: The path of a point on a moving bicycle wheel

We will meet an example of choosing an appropriate reference frame when we discuss the

orbiting solar system.

In addition, the concept of being in a frame of reference is important when we discuss Newton's Laws of Motion. The three laws of motion state:

* Every object continues in its state of rest or of uniform straight-line motion unless acted on by an external force.

* The acceleration of an object is directly proportional, and in the same direction, as the force acting on it, and it is inversely proportional to the mass of the object.

* To every action there is an equal and opposite reaction.

Are these laws valid in whatever frame of reference we choose, or do we have to have a different set of laws to match the observations seen in different reference frames? Newton's laws are made under the assumption that all measurements and observations are taken with respect to a frame of reference that is *fixed* in space. The frame of reference is at rest. If Newton's laws are valid for one reference frame they are also valid for a reference frame moving relative to it with a constant velocity. These reference frames are called *inertial reference frames* or *Newtonian reference frames.* A *non-inertial* reference frame is one which is accelerating with respect to an inertial reference frame. We will take another look at inertial and non-inertial reference frames later when we discuss rotational motion.

2.1 A Closer Look at Motion

Let us now take a closer look at motion. Imagine two people, a RUNNER and a WALKER, moving in the same direction, RUNNER with a speed of 10 units and WALKER with a speed of 5 units. Let us consider where these two people will be after consecutive intervals of time. RUNNER moves at a speed of 10 units, so RUNNER moves forward a distance of 10 units in one interval of time, to point R, as shown in Figure 4. The WALKER moves at a rate of 5 units and moves forward 5 units from its starting point, in this same interval of time, to point S. This is shown in Figure 4. After a second interval of time RUNNER moves a further 10 units to point X and WALKER moves a further 5 units

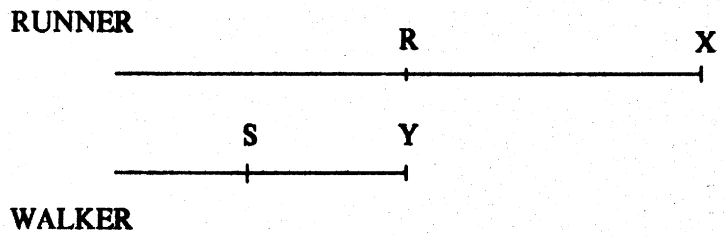

Figure 4: The positions of RUNNER and WALKER after intervals of time

to point Y. This too is shown in Figure 4.

Now let us consider what happens if we want to determine how the WALKER sees the RUNNER move. We have to imagine being the WALKER and watching what happens to the RUNNER. In this case the WALKER is the frame of reference. After one interval of time the RUNNER is at position R, and the WALKER is at position S. The WALKER sees the RUNNER 5 units in front of itself. After two intervals of time the RUNNER is at position X and the WALKER is at position Y. The WALKER sees the RUNNER 10 units in front of itself. So for each interval of time that passes the RUNNER moves 5 units further away from the WALKER. We can think of the RUNNER moving with a speed of 5 units *relative* to the WALKER.

We can use Figure 4 to find out how the RUNNER sees the WALKER move. In other words the RUNNER becomes the frame of reference. We use the same principle: consider what happens after intervals of time and find the position of the WALKER *relative* to the RUNNER. Let us do this.

After one interval of time the RUNNER moves to point R and the WALKER moves to point S. If we imagine being the RUNNER, we see the WALKER 5 units behind. After another interval of time we see the WALKER 10 units behind. The WALKER appears to be moving *backwards* with a speed of 5 units *relative* to the RUNNER.

In this example we look at how one object appears to move as seen by another object. The general principle is to let each object move for a small amount of time (a time interval) and then imagine standing on a chosen object to view the position of the other objects. If we do this for several intervals of time we build up a picture of how an object appears to move as seen by another object. This method of "time intervals" will be especially useful when we consider motion that is not parallel.

2.2 Modeling Relative Motion

Before we discuss how relative motion can be modeled, let us first discuss how motion can be simulated on the computer screen.

Some of the commands available in Logo are used to move a *turtle* around on the computer screen. A turtle is a triangular shaped object in some implementations of Logo. In others, you choose the turtle's shape from a set of shapes. The turtle can leave a trace of its path as it moves around on the computer screen. For example:

```
PENDOWN
FORWARD  50
RIGHT  90
FORWARD  30
RIGHT  135
FORWARD  20
```

puts the turtle's drawing pen down, moves the turtle 50 units drawing a line as it goes, turns the turtle to the right by 90 degrees, moves the turtle forwards by 30 units, turns the turtle a further 135 degrees to the right, and moves the turtle forwards 20 units on this new heading. This is shown in Figure 5.

The command FORWARD *:amount* moves the turtle forwards by a certain amount in the direction the turtle is heading. If you give this command repeatedly to a turtle you see the turtle move continuously across the computer screen. For example, if you type:

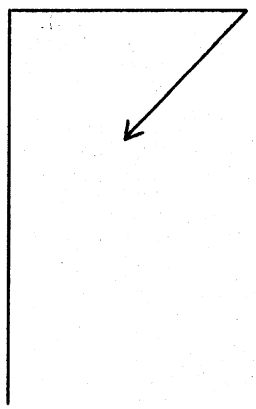

Figure 5: Drawing with the turtle

```
REPEAT 100 [FORWARD 1]
```

you see the turtle move across the computer screen with a "speed" of 1 unit. Similarly, if you type the command:

```
REPEAT 100 [FORWARD 2]
```

you see the turtle move across the screen with a "speed" of 2 units.

Now let us look at how the system models relative motion. What we want to do is to move objects and ride with a chosen object to view how everything looks from that object. We did this when we considered the example of the RUNNER and the WALKER. We imagined ourselves being the RUNNER and watched how the WALKER moved.

There are four basic steps to setting up a simulation on the computer screen: make the objects that will move around, tell each object how to move, choose which object you want as the reference frame, and start the objects moving. The simulation on the screen shows how all the objects appear to move as seen by the object you chose as the reference frame.

There are several new primitives introduced by the microworld. These are listed in Table 1. We will discuss each primitive as it is introduced in the text. First let us consider the four basic steps in setting up an example on the computer.

```
FALL
INIT.FALL    :name  :from
INIT.PUSH    :name  :from
INIT.TOSS    :name  :from
MAKEFRAME    :frame.of.reference
MAKETURTLE   :name  :x.position  :y.position  :heading  :color
MOVE    :turtles
PUSH    :speed
SETMOTION    :name  :motion
SETRELATIVEMOTION    :name  :relative.to  :motion
TOSS    :speed
```

Table 1: The new primitives introduced by the microworld

First we must make the objects. The command MAKETURTLE is used to do this:

```
MAKETURTLE :name :x.position :y.position :heading :color
```

This sets up an object with a chosen name, at a given initial position on the computer screen, with a heading and color of drawing pen. For example, you may want a turtle called RUNNER to be at the position $x = 0$ and $y = 20$, on a heading of 90 degrees, and to draw its path as it moves with a red pen. The command to do this is:

```
MAKETURTLE  "RUNNER  0  20  90  :RED
```

After making the turtle you need to tell the turtle how to move, using the SETMOTION command:

```
SETMOTION  :name  :motion
```

This command requires you to specify a motion for the turtle. This motion will be continuously repeated. If you give your turtle the motion [FORWARD 5], then this motion is repeated again and again, and the turtle appears to move forwards on the computer screen with a speed of 5 units. The command to give this motion to a turtle named RUNNER is:

```
SETMOTION "RUNNER [FORWARD 6]
```

You must choose a frame of reference to view the motion, using the command MAKEFRAME:

```
MAKEFRAME :frame.of.reference
```

MAKEFRAME takes as input the name of the turtle you choose as the reference frame. Suppose you make three turtles: RUNNER, WALKER and EARTH. If you want to see how the RUNNER and WALKER move from the EARTH, you make the EARTH the reference frame.

```
MAKEFRAME "EARTH
```

The final command you must give to set up a demonstration is the MOVE command. MOVE takes as input a list of the turtles to be moved and shown on the computer screen:

```
MOVE :turtles
```

For example, if you make three turtles RUNNER, WALKER and EARTH and give them motions then:

```
MOVE [RUNNER WALKER EARTH]
```

will repeatedly perform each turtle's motion. If you choose the EARTH as the reference frame you will see on the screen the paths of the RUNNER and WALKER as seen from the EARTH.

These commands are the basic commands necessary to set up turtles and move them

around on the computer screen. Additional commands have been designed for use under specific conditions, see Table 1. These will be described later.

As you work through the text you will find examples of short programs to write which will help you become familiar with the microworld and setting up turtles with different forms of motion.

2.2.1 A Beginning Example

Let us now consider a specific example. We will set up the example of the RUNNER and the WALKER on the computer. How do we do this? We must use the four steps outlined previously:

1. Make the turtles.

2. Tell the turtles how to move.

3. Choose which frame of reference we want to view the turtles from.

4. Move the turtles.

The following short program outlines how you can set up this problem. It is simpler to use the Logo editor to make a short program which includes all the commands you need, rather than type in the commands each time you want to run a demonstration. You type the program or *procedure* into the editor. The procedure has a *title* line, which consists of the word TO followed by the name chosen for the procedure, a sequence of commands, and the word END to indicate the end of the procedure. Here we call the program DEMO, and once we define it, all we need to do is type DEMO to run the program.

```
TO DEMO
  MAKETURTLE  "RUNNER  0  60  90  :RED
  MAKETURTLE  "WALKER  0  40  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "RUNNER  [FORWARD  10]
  SETMOTION  "WALKER  [FORWARD  5]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  "EARTH
  MOVE  [RUNNER  WALKER  EARTH]
END
```

In this example we set up three turtles, using MAKETURTLE. A turtle named RUNNER starts at $x = 0, y = 60$ on the computer screen, with a heading of 90 degrees. A turtle named WALKER starts from position $x = 0, y = 40$, with a heading of 90 degrees. Figure 6 shows this.

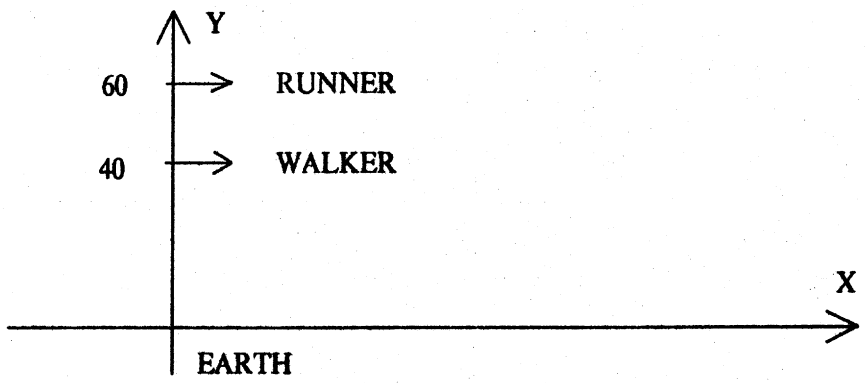

Figure 6: Setting up turtles on the computer screen

The third turtle, the EARTH, is necessary so that we can use it as the reference frame to see both RUNNER and WALKER moving on the screen at the same time. It is positioned at $x = 0, y = 0$. We use a different color for each turtle so that we can distinguish each turtle's

track.

We give each turtle a motion, RUNNER [FORWARD 10] and WALKER [FORWARD 5]. This means that RUNNER repeatedly moves forwards 10 units and WALKER repeatedly moves forwards 5 units. The EARTH turtle does not move at all. However, it is still necessary to give it a motion of "do nothing".

MAKEFRAME specifies where you view the motion from. In this example the EARTH is the reference frame. Finally the MOVE command specifies which turtles are to be moved on the computer screen.

The command DEMO starts the demonstration. We see the RUNNER and WALKER move on the computer screen, as seen by the stationary EARTH. The EARTH is the reference frame and is indicated by a small cross at the center of the screen. RUNNER and WALKER leave a trace of their paths as they move across the screen. Figure 7 shows this.

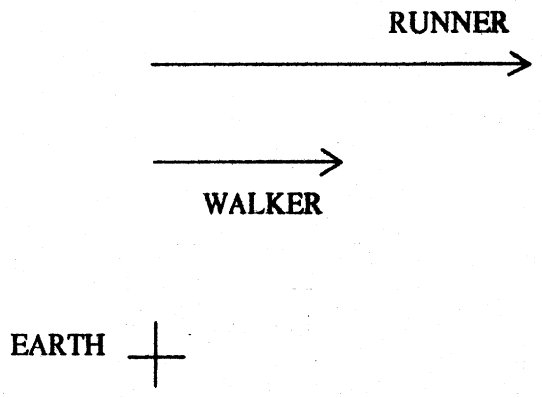

Figure 7: RUNNER and WALKER as seen by the EARTH

This program is not very versatile. Every time we run DEMO we see the RUNNER and WALKER as seen by the EARTH. If we want to see how the RUNNER sees the WALKER move, we need RUNNER as the reference frame. To do this we need to enter the Logo editor and change the command MAKEFRAME "EARTH to MAKEFRAME "RUNNER. Every time we want to look from a different frame of reference we need to change the program.

It is much more efficient to write the program so that all the quantities that we want to change are passed into the program as *input variables.* Then each time we run DEMO we tell the program what values we want the variables to have. If we call the frame of reference the input variable *:VIEWFROM* we write the program as:

```
TO DEMO :VIEWFROM
  MAKETURTLE "RUNNER 0 60 90 :RED
  MAKETURTLE "WALKER 0 40 90 :GREEN
  MAKETURTLE "EARTH 0 0 0 :YELLOW
  SETMOTION "RUNNER [FORWARD 10]
  SETMOTION "WALKER [FORWARD 5]
  SETMOTION "EARTH [ ]
  MAKEFRAME :VIEWFROM
  MOVE [RUNNER WALKER EARTH]
END
```

Now in order to see the RUNNER and the WALKER from the reference frame of the EARTH we start the program with the command:

```
DEMO "EARTH
```

If we choose the WALKER as the reference frame then we use the command:

```
DEMO "WALKER
```

to start the program. In this case we see the RUNNER and the EARTH as seen by the WALKER. This is shown in Figure 8.

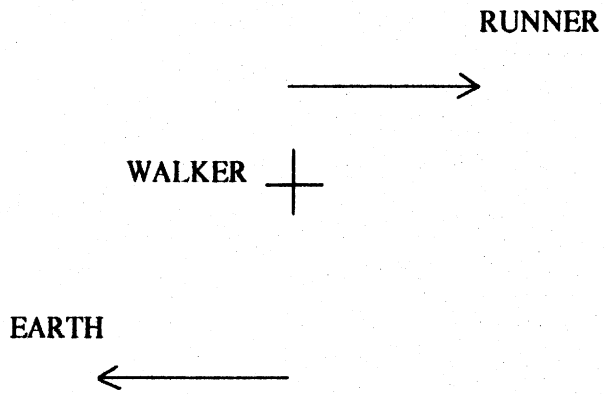

Figure 8: The RUNNER and EARTH as seen by the WALKER

The WALKER is the reference frame and it believes itself as standing still, with all the other objects doing the moving. It is shown as a small cross at the center of the computer screen. The RUNNER is seen moving forwards at a speed of 5 units with respect to the WALKER. The EARTH is seen to move backwards at a speed of 5 units relative to the WALKER. Note the positions of the RUNNER and the EARTH. When we consider the initial positions of the turtles, see Figure 7, the RUNNER is above the WALKER and the EARTH below. The turtles still have these relative positions when the WALKER is the reference frame. The RUNNER is above the center cross, and the EARTH below it.

The RUNNER can be chosen as the frame of reference using:

```
DEMO  "RUNNER
```

Figure 9 shows the WALKER and the EARTH as seen by the RUNNER. The WALKER appears to move backwards at a speed of 5 units relative to the RUNNER and the EARTH appears to move backwards at a speed of 10 units.

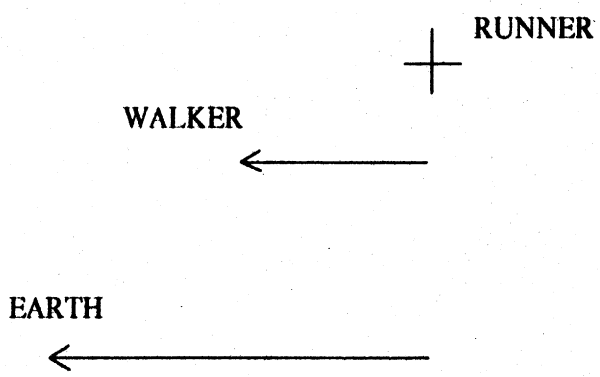

Figure 9: The WALKER and EARTH as seen by the RUNNER

The following exercises will give you practice in setting up and moving turtles. The first question asks you to try the demonstration with different values for the speed of the RUNNER and for the speed of the WALKER. Instead of editing the program each time you want different speeds you can specify the speed of both the RUNNER and the WALKER as input parameters to the program DEMO.

```
TO DEMO :VIEWFROM :R :W
  MAKETURTLE "RUNNER 0 60 90 :RED
  MAKETURTLE "WALKER 0 40 90 :GREEN
  MAKETURTLE "EARTH 0 0 0 :YELLOW
  SETMOTION "RUNNER [FORWARD :R]
  SETMOTION "WALKER [FORWARD :W]
  SETMOTION "EARTH [ ]
  MAKEFRAME :VIEWFROM
  MOVE [RUNNER WALKER EARTH]
END
```

The program DEMO now takes three input parameters: the frame of reference, the

RUNNER's speed and the WALKER's speed. To give the RUNNER a speed of 10 units and the WALKER a speed of 5 units and to view their motion from the EARTH you type the command:

```
DEMO  "EARTH  10  5
```

2.2.2 Exercises

1. Practice using DEMO with different values for *:R* and *:W*. E.g. 10 and 5, 5 and 10, 20 and 10, 10 and 20. View the scene from the EARTH, then from the RUNNER, and then the WALKER.

   In each case what do you see?

   Can you explain your findings?

2. When the RUNNER or the WALKER is the reference frame, you see the EARTH move on the screen. What is the motion of the EARTH? Is this what you expect?

3. What happens if RUNNER and WALKER are moving at the same speed? What do you see when the EARTH is the frame of reference?

   What do you see when the RUNNER is the reference frame? Explain the motion you see.

4. If a turtle is moving in the forwards direction under what conditions do you see it move backwards?

5. What happens when the RUNNER and WALKER are set moving in opposite directions? Suppose you set RUNNER moving forwards at a speed of 10 units

and WALKER backwards at a speed of 5 units. For example, DEMO "EARTH 10 (-5).

What do you see when you view from each reference frame?

What happens to the RUNNER's speed when you view from the WALKER?

6. Set both the RUNNER and the WALKER moving backwards using negative values for *:R* and *:W*. For example, DEMO "RUNNER (-10) (-5). What happens when you view the scene from either the RUNNER or the WALKER? Is this what you expect? What happens to the EARTH in this case?

2.2.3 Setting Motion Relative to another Turtle

The SETRELATIVEMOTION command gives a turtle a motion with respect to another turtle:

```
SETRELATIVEMOTION  :name  :relative.to  :motion
```

When we use the SETMOTION command, as in the previous exercises, we can think of the motion as being with respect to the stationary EARTH. Suppose you are standing on a moving sidewalk, similar to the ones found at large airports. The sidewalk is moving forwards at 3 m.p.h., then to a person standing on the stationary ground, you appear to move at 3 m.p.h. If you now walk forwards on the sidewalk at 2 m.p.h., the person on the ground sees you moving forwards at 5 m.p.h. Your walking motion is *relative* to the sidewalk. The SETRELATIVEMOTION command sets up this motion:

```
SETRELATIVEMOTION  "PERSON  "SIDEWALK  [FORWARD 2]
```

Let us consider an example on the computer of using the SETRELATIVEMOTION command:

```
TO DEMO.RELATIVE  :VIEWFROM
  MAKETURTLE  "A  0  40  90  :RED
  MAKETURTLE  "B  0  20  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [ ]
  SETRELATIVEMOTION  "B  "A  [FORWARD  10]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [A B EARTH]
END
```

DEMO.RELATIVE illustrates setting up two turtles, A and B. Turtle A is a stationary turtle. The SETRELATIVEMOTION command sets turtle B moving with a speed of 10 units *relative* to A. DEMO.RELATIVE takes one input variable, the frame of reference. The following exercises use this program.

2.2.4 Exercises

1. Use DEMO.RELATIVE to show that B moves with a speed of 10 units when viewed from either the EARTH as reference frame or A as reference frame.

   Why is the demonstration the same from either reference frame?

2. If you set A moving, at a speed of 20 units, show that from turtle A, turtle B still moves at a speed of 10 units.

   How does B appear to move from the EARTH turtle? Is this what you expect?

3. What happens when you give A a motion backwards? You can do this by giving a negative value to the FORWARD command. Try values of -5, -10, and -20. How does turtle B move when viewed from the EARTH turtle and from turtle

A?

4. This is a question to think about. The principles of relative motion can be used in the area of electromagnetic induction (the study of producing a magnetic field using an electric current and vice versa). Suppose you place a circuit, consisting of a loop of wire and a meter to measure the current, inside a magnetic field, as shown in Figure 10. If you move the wire loop through the magnetic field a current is produced. If you move the wire twice as fast, the current is doubled. If the loop is moved to the right instead of to the left, the current is reversed.

   What do you think happens if you move the wire loop and the magnet together at the same speed? Will you induce a current in the loop?

   What do you think happens if you hold the loop at rest and move the magnet? How can you get a maximum current induced in the loop? What can you conclude about the motion necessary to produce a current?

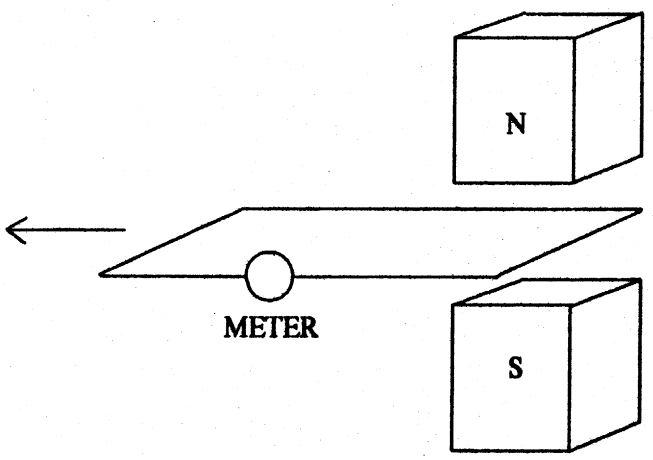

Figure 10: Moving a circuit in a magnetic field

2.3 Non-parallel Motion

In this section we will consider objects that move in straight lines, but are not parallel to one another. As an introduction let us discuss two turtles, A and B, moving at right angles to one another. A travels North at 3 m.p.h. and B travels East at 4 m.p.h. How does A see B move? To solve this problem we can use the principles outlined earlier. First consider what happens to each turtle during a small interval of time, then imagine standing on A and looking to see where B is positioned with respect to A. Then consider what happens during the next interval of time, and continue like this building up a picture of the motion. Let us do this.

Suppose A and B move from the same point. After one interval of time A is at position X and B is at position Y, as shown in Figure 11.

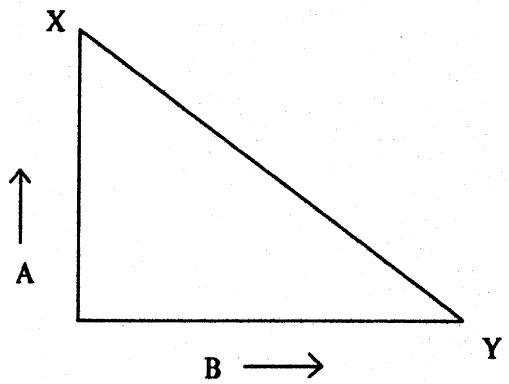

Figure 11: A and B after one interval of time

Imagine standing on turtle A, at point X, and viewing turtle B. How far away is B and what direction is it in? B is 5 units from A and is on a heading of 127 degrees. Figure 11 is drawn to scale, so you can check these measurements.

What does the scene look like after another interval of time? A travels a further 3 units North to point R and B travels a further distance of 4 units East to point S, as shown in Figure 12.

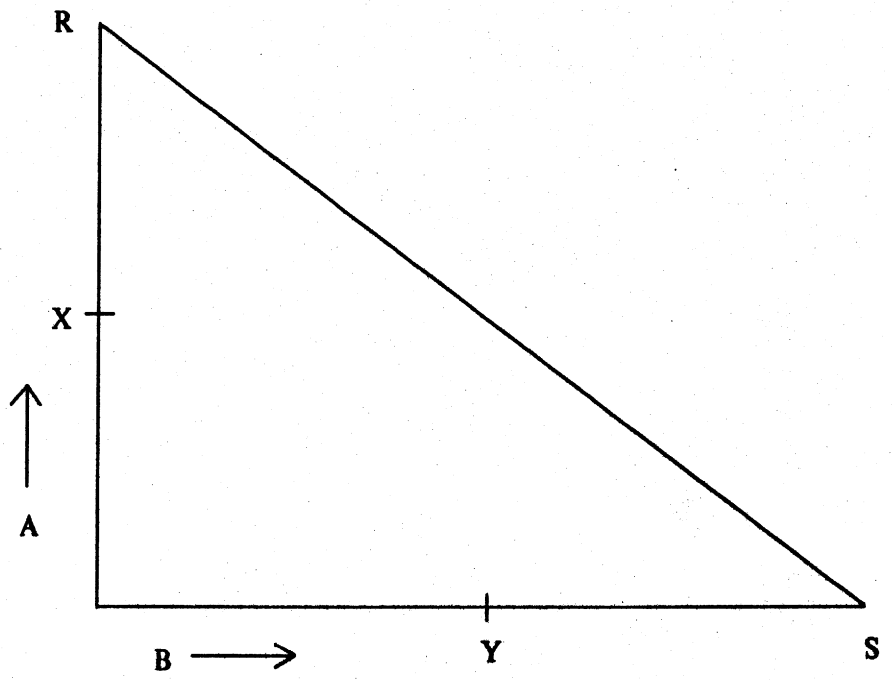

Figure 12: A and B after two intervals of time

If you stand on A you see B 10 units away on a heading of 127 degrees.

We can continue building up a picture of the motion in this way. For every interval of time, B becomes 5 units further away from A. Therefore, B is moving *relative* to A at 5 m.p.h., on a heading of 127 degrees, as shown in Figure 13.

When you stand on A and view what is happening to the other turtles as they move around, you are in A's frame of reference. You see the world as seen by turtle A. Turtle A considers itself to be stationary, with everyone else doing the moving.

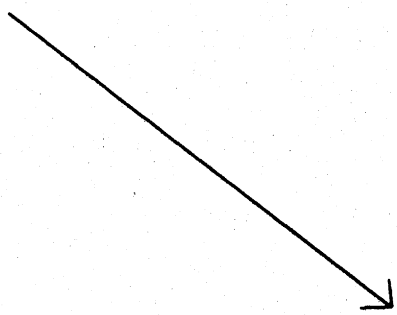

Figure 13: The motion of B as seen by A

A short program to illustrate the motion of turtles A and B on the computer is:

```
TO DEMO.ANGLE  :VIEWFROM
  MAKETURTLE  "A  0  0  0  :RED
  MAKETURTLE  "B  0  0  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [FORWARD  3]
  SETMOTION  "B  [FORWARD  4]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [A B EARTH]
END
```

The following exercises use this program so that you can investigate non-parallel motion.

2.3.1 Exercises

1. Use the procedure DEMO.ANGLE to look at A and B moving from the EARTH as reference frame. A heads North and B moves East.

   Now repeat the demonstration with A as reference frame. See that B moves as shown in Figure 13.

2. Use Figure 12 to imagine yourself in B's reference frame. After each interval of time how does A appear to be moving?

   Use the procedure DEMO.ANGLE with B as the reference frame to verify this. How is this different from Figure 13? [Hint: A good way to check this is to run DEMO.ANGLE with A as the reference frame, then with B as the reference frame, without clearing the screen in between.]

3. P and Q move in zig-zags as shown in Figure 14. They move at the same rate and so reach the apex of their paths at the same time. Use the method of time intervals (take a step and look at the distance and direction of the other object from you) to determine how P sees Q move.

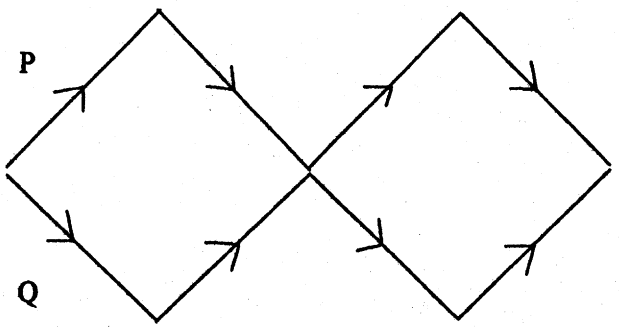

Figure 14: P and Q moving in zig-zags

4. Suppose turtle A starts from a position $x = -50$, $y = 0$ on a heading of 90 degrees. Turtle B starts from a position $x = 0$, $y = -50$ on a heading of 0 degrees. A has a motion of [FORWARD 4] and B has a motion of [FORWARD 6].

Edit DEMO.ANGLE to put in these new values and run the procedure to show that the two turtles do not collide.

What can you do to determine their closest distance apart? [Hint: Consider viewing the scene from one of the moving turtles.]

2.3.2 More on the SETRELATIVEMOTION Command

You will need to use the SETRELATIVEMOTION command many times as you work through the exercises in this text. You use the command when you set a turtle moving relative to another turtle. Let us consider what happens when we row a boat on a river. If we sit in a boat on a river, without rowing, what happens? The boat floats downstream. The river is flowing at a certain speed, and the boat floats along with the river. How do we know we are moving? The banks of the river appear to be moving backwards. We can think of the boat as absorbing the speed of the river. Relative to the river the boat is not moving. Both are moving along at the same speed. If we now row the boat, we give the boat a speed *relative* to the river. The SETRELATIVEMOTION command is used to do this:

```
SETRELATIVEMOTION "BOAT "RIVER [FORWARD 2]
```

Here we row the boat at 2 m.p.h. relative to the river.

The following exercises use the SETRELATIVEMOTION command. You can edit DEMO.ANGLE to include the necessary changes for each exercise.

2.3.3 Exercises

1. Investigate rowing on a river which flows at 4 m.p.h. when you can row at 6 m.p.h. Set up a demonstration with a river flowing horizontally across the screen. Try rowing the boat on different headings.

   What happens when you head slightly upstream? Downstream?

2. How do you row to a point directly opposite on the other bank of the river? In which frame of reference do you have to be to see the boat move straight across the river?

3. Two boats are sailing on the ocean. Boat A is at position [0,0] and boat B at position [50,0]. Boat A moves with a speed of 5 units and sees boat B heading straight towards itself with a speed of 10 units. Boat A knows that it is on a heading of 45 degrees. Set up a demonstration to determine the actual motion of boat B with respect to the ocean.

3 Motion Under Gravity

In this section we will look at objects moving under the influence of gravity. What happens when you throw a ball vertically into the air? The ball moves up, stops, and then moves down again, so that you can catch it. If you take a series of photographs of the movement at regular short intervals of time, the ball moves a smaller amount during each successive interval of time, finally stopping. The ball then falls from this position, and during each successive interval of time falls a greater distance. Gravity is responsible for this motion. When a ball is thrown vertically upwards gravity acts to slow the ball down, until it comes to rest. Then the ball falls from this position, moving faster and faster under gravity on its descent.

Near the Earth's surface all objects fall with a constant acceleration, providing air resistance is negligible. This constant acceleration is denoted by **g** and is called the *acceleration due to gravity.* **g** has an approximate value of 9.8 m/sec/sec. When a ball falls from rest, at the end of one second it has a speed of 9.8 m/sec, at the end of two seconds a speed of 19.6 m/sec, and at the end of three seconds a speed of 29.4 m/sec. For every second, the ball's speed increases by 9.8 m/sec.

The relative-motion system simulates gravity. Previously we gave an object a speed by repeatedly moving the object forwards by a given amount. Now when a ball moves with gravity we move the ball forwards by a certain amount, and for the next interval of time, we move the ball forwards by a slightly larger amount. When the ball moves against gravity we move the ball forwards a slightly smaller amount for each interval of time. The amount added or subtracted from the forward motion is a constant amount which represents the acceleration due to gravity.

This section explores motion under gravity and how objects appear to move as they are dropped or thrown. The frame of reference from which you view the motion becomes very important in this section. Remember the example of two cars moving along a highway; an observer standing at the side of the highway described the scene in a very different way from a person riding in of one of the cars. We will find the same phenomenon here; the

description of the scene depends very much on where the scene is being viewed from, in other words the frame of reference.

3.1 Dropping Objects

An apple falls from a tree. How does a person walking along at 5 m.p.h. see the apple fall? What does the apple see? This example can be demonstrated in the following way:

```
TO DEMO.FALL  :VIEWFROM
  MAKETURTLE  "PERSON  (-50)  (-20)  90  :YELLOW
  MAKETURTLE  "TREE  0  0  0  :GREEN
  MAKETURTLE  "APPLE  0  0  0  :RED
  SETMOTION  "PERSON  [FORWARD  5]
  SETMOTION  "TREE  [ ]
  SETMOTION  "APPLE  [FALL]
  MAKEFRAME  :VIEWFROM
  INIT.FALL  "APPLE  "TREE
  MOVE  [TREE  APPLE  PERSON]
END
```

The example follows the steps outlined previously for setting up demonstrations: make the turtles, specify their motion, choose a reference frame and move the turtles.

First we make the turtles. Three turtles are set up: a PERSON, a TREE and an APPLE. A PERSON starts from position $x = -50, y = -20$ and moves horizontally across the screen with a speed of 5 units. The TREE is stationary and is at $x = 0, y = 0$. The APPLE starts at $x = 0, y = 0$ and is given a falling motion using:

```
SETMOTION  "APPLE  [FALL]
```

The procedure FALL is a microworld primitive that makes a turtle move as though it falls under gravity. Before you set the turtles moving with the MOVE command, you must initialize the motion of the falling turtle with INIT.FALL. INIT.FALL requires two input parameters: the name of the falling turtle and where it is falling from:

```
INIT.FALL  :name  :from
```

In the above example the APPLE falling from the tree is initialized with:

```
INIT.FALL  "APPLE  "TREE
```

Try this demonstration. Look at the motion of the turtles viewed from each turtle's frame of reference. Figure 15 shows the motion you see from the stationary TREE. The APPLE falls straight down and the PERSON moves horizontally across the screen.

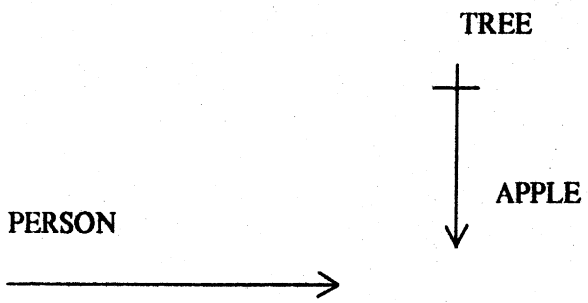

Figure 15: The motion of the APPLE and PERSON seen by the TREE

Figure 16 shows the motion seen by the walking PERSON. The TREE moves in a straight line towards the PERSON and the APPLE falls towards the PERSON in a curve. We will discuss how this motion occurs in the next section.

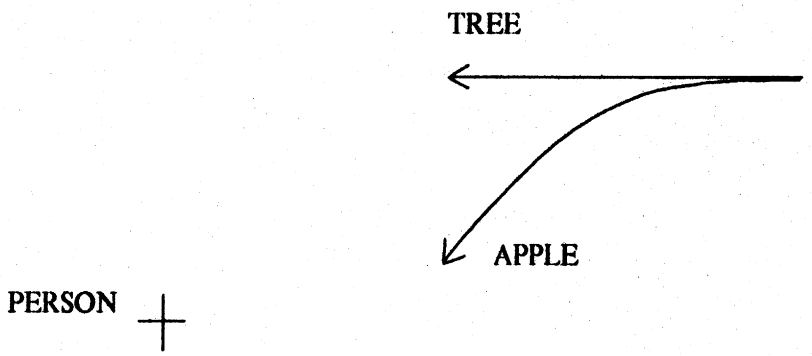

Figure 16: The TREE and APPLE seen by the PERSON

3.1.1 Analyzing the Motion

We can look at this example and analyze it, as we have done with previous examples, in terms of what is happening after each interval of time. Figure 17 shows the positions of the moving PERSON and the falling APPLE over successive intervals of time.

What happens during one interval of time? The PERSON moves forwards 5 units, the TREE remains stationary, and the APPLE falls a short distance. The falling APPLE starts with a speed of zero and gradually falls faster and faster due to gravity. Let us assume for this demonstration that the acceleration due to gravity is represented by one unit. This means that the APPLE falls an extra one unit during each interval of time. So after one interval of time the APPLE has a speed of 1 unit, after two intervals a speed of 2 units, and after another interval a speed of 3 units.

During the second time interval the PERSON moves forwards a further 5 units and the APPLE falls an additional 2 units. During the third interval of time the person moves forwards by another 5 units and the APPLE falls 3 units.

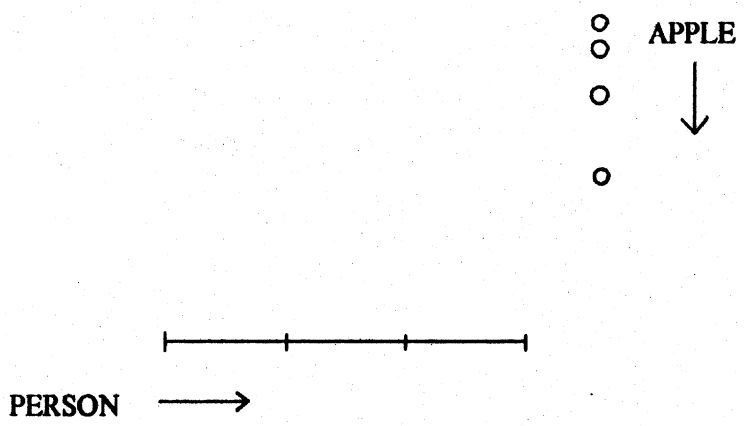

Figure 17: Analysis of falling

If we stand on the stationary TREE turtle then we see the PERSON moving horizontally and the APPLE falling straight down, as shown in Figure 15. If we want to view the scene from a different reference frame we stand on the turtle chosen as the reference frame and determine, time interval by time interval, the positions of the other turtles from us, to build up the path of each turtle. If we stand on the PERSON we see the APPLE moving towards us in a curve, as shown in Figure 16. If we stand on the APPLE and view the PERSON we see the PERSON moving upwards in a curve! Try it!

3.1.2 Dropping from Moving Objects

The previous example showed an apple dropping from a stationary tree. Now let us consider what happens when something falls from a moving object. A penny falls from a runner's hand. What path does the penny take in falling to the ground? Figure 18 shows some suggested answers to this problem. These are: (i) the penny falls straight down; (ii) the penny falls in an arc for a short way, then falls down; (iii) the penny falls in a curve; (iv) the penny falls backwards in a curve. What do you think happens? Before you read any

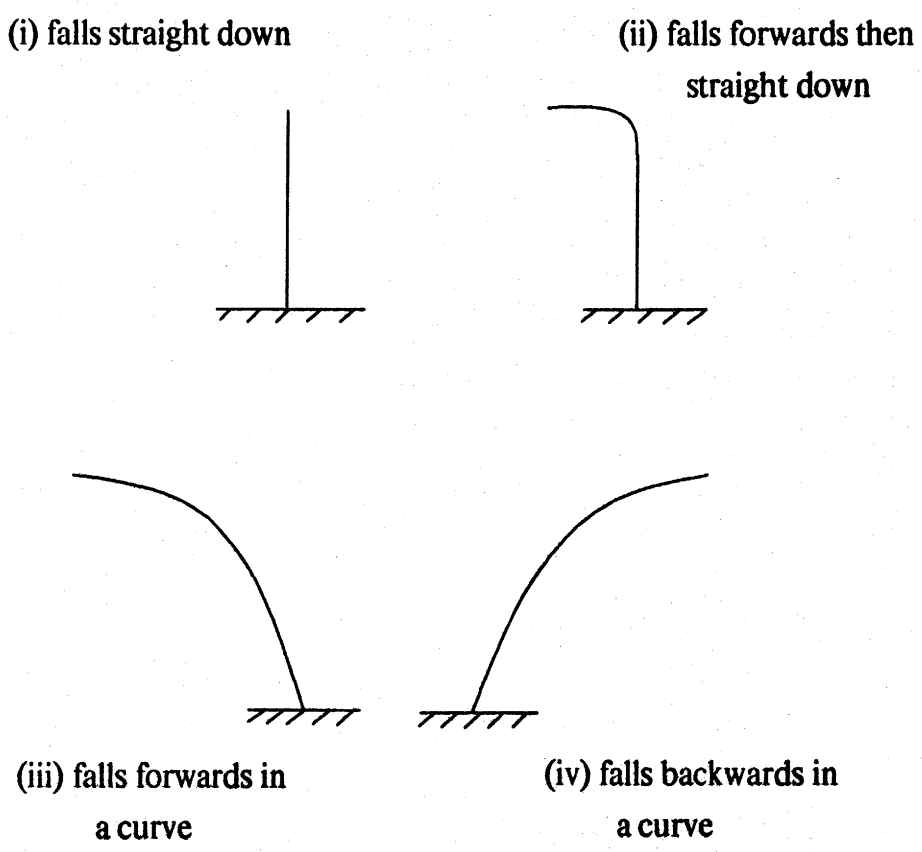

Figure 18: Suggested paths of a penny in falling to the ground

further write down what you think the path of the penny is, and the reasons for your choice.

In order to solve this problem we must think about how the penny moves as it leaves the person's hand. Most students answer this problem with "the penny falls straight down". They choose this solution because when you run and drop something, it falls at your feet. Try it! You move and continue to move after you drop the penny. This means that the

penny moves forwards too, in order to land at your feet. In fact, at the point the penny leaves your hand it is moving exactly as you are moving. Therefore, in this example the penny is moving horizontally forwards as it leaves the person's hand. This means that answers (ii) or (iii) must be correct, which both show the penny moving forwards.

To determine which of these two answers is the correct one we can think about this problem in terms of the laws of physics, namely Newton's First Law of Motion. This law states that an object remains in a state of rest or of uniform straight-line motion unless acted on by a force. What forces are acting on the penny? There is only gravity acting vertically. There is no horizontal force acting on the penny, which is required to give the penny an abrupt change in its direction as shown in (ii). Gravity acts vertically, it does not effect the horizontal motion of the penny. The penny continues to move forwards horizontally at a constant speed. Gravity acts vertically and changes the penny's vertical speed. The penny moves faster and faster vertically. The penny's motion is a combination of a constant horizontal speed and an increasing vertical speed. Therefore, the penny moves in a curve as shown in (iii).

Both the penny and the person move forwards with the same horizontal speed. This speed remains constant for both of them. This means that they will be at the same horizontal position at the same time, as shown in Figure 19. How does the person see the penny? We use the method described earlier. We imagine standing on the person and looking at the penny. For each interval of time the person sees the penny vertically below himself/herself. Therefore, the person sees the penny falling straight down. Now we can see why so many people often answer problems of dropping incorrectly. They answer with how *they* see something that *they* have dropped.

How can we set up this problem using the computer? We need a PERSON, a PENNY and an EARTH turtle. DEMO.DROP illustrates this:

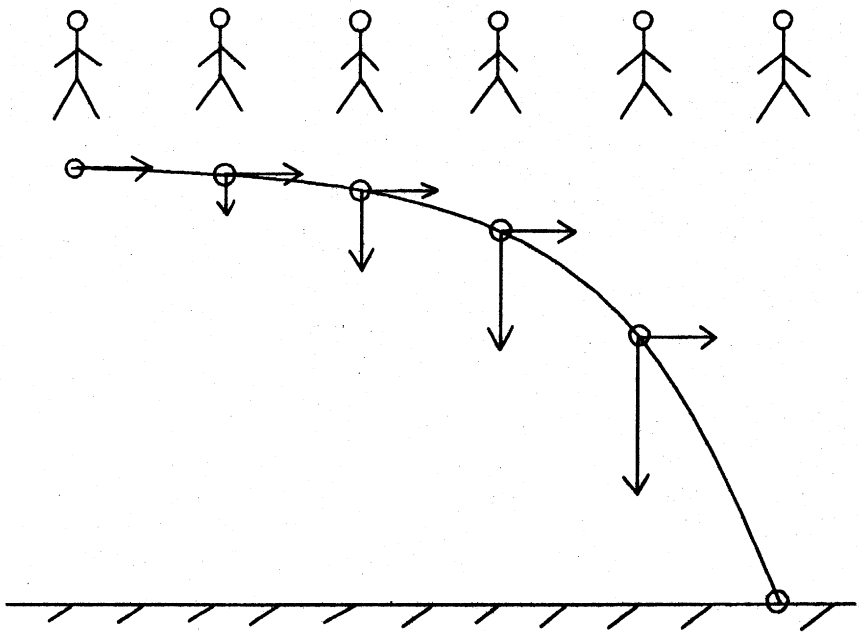

Figure 19: A penny dropped by a moving person

```
TO DEMO.DROP  :VIEWFROM
  MAKETURTLE  "PENNY  0  0  0  :GREEN
  MAKETURTLE  "PERSON  0  0  90  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "PENNY  [FALL]
  SETMOTION  "PERSON  [FORWARD  5]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.FALL  "PENNY  "PERSON
  MOVE  [PENNY PERSON EARTH]
END
```

The PERSON moves forwards at a constant rate, the EARTH remains stationary and the PENNY has a falling motion given by:

```
SETMOTION  "PENNY  [FALL]
INIT.FALL  "PENNY  "PERSON
```

INIT.FALL initializes the PENNY falling *from* the PERSON.

The following exercises give you practice setting up turtles to move under gravity and to view their motion from a variety of reference frames.

3.1.3 Exercises

1. A person standing still sees an apple fall straight to the ground. Using the procedure DEMO.FALL how must the person move in order to see the apple fall towards himself/herself? Fall away from himself/herself?

   Is it possible to see the apple move horizontally? [Hint: Vary the speed of the PERSON and do not clear the screen between running DEMO.FALL.]

2. In the previous example, what does the APPLE see when the PERSON stands still? When the PERSON moves forwards? Moves backwards?

   Explain this motion. To help you do this consider the motion time interval by time interval, and imagine yourself standing on the APPLE and viewing the position of the PERSON.

3. Two apples fall from a tree at the same time. Imagine falling with one of the apples. How do you see the other apple move? Set up this problem on the computer and then use the method of time intervals to explain what you see.

4. A PERSON walking briskly drops a PENNY. Using DEMO.DROP show that

when the PENNY reaches the ground the PERSON can just bend straight down to pick up the PENNY.

What does the PERSON see?

If a RUNNER moving in the same direction as the PERSON watches this scene, how fast does the RUNNER have to move to see the PENNY fall straight down? Move horizontally? [Hint: include a RUNNER turtle in the procedure DEMO.DROP.]

What happens if the RUNNER goes in the opposite direction?

5. In an aircraft moving at a constant speed the steward pours coffee. Why does the coffee fall into the cup and not onto the passenger's lap?

6. A person moves up an incline of 45 degrees to the horizontal, at a constant speed of 5 units, and drops a ball. What does the path of the ball look like to someone standing on the ground? How does the person see the ball move?

How fast does a car have to drive horizontally to keep up with the person traveling on the incline? In this case how does the car see the ball move?

3.2 Throwing Objects

When you throw something into the air it moves under gravity. However, when you throw an object it has an initial speed due to being thrown. Gravity acts vertically and has an effect on the object's vertical speed only. If you throw an object at an angle above the horizontal the object moves against gravity, comes to rest, and then falls with gravity. Meanwhile the object continues to move horizontally at a constant rate. Let us consider how we can set up a demonstration to illustrate this.

Suppose a person stands still and throws a ball into the air at 60 degrees to the horizontal, with a speed of 5 units. The procedure DEMO.TOSS illustrates this.

```
TO DEMO.TOSS  :VIEWFROM
  MAKETURTLE  "BALL  0  0  30  :RED
  MAKETURTLE  "PERSON  0  0  0  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "BALL  [TOSS  5]
  SETMOTION  "PERSON  [ ]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.TOSS  "BALL  "PERSON
  MOVE  [BALL  PERSON  EARTH]
END
```

Here we set up a BALL on a heading of 30 degrees (60 degrees to the horizontal). We give the BALL a motion of [TOSS 5] using the command:

```
SETMOTION  "BALL  [TOSS  5]
```

The procedure TOSS is a microworld primitive that requires one input parameter, the speed of the throw:

```
TOSS  :speed
```

TOSS makes a turtle move under gravity. Before you move the turtles, you must initialize the turtle being thrown using the procedure INIT.TOSS. This procedure requires two input parameters, the name of the turtle being thrown and the name of the turtle doing the throwing:

```
INIT.TOSS  :name  :from
```

In the example DEMO.TOSS the BALL's motion is initialized using:

```
INIT.TOSS "BALL "PERSON
```

The following exercises use DEMO.TOSS to look at a variety of different examples of objects being thrown.

3.2.1 Exercises

1. Set up the demonstration using DEMO.TOSS. If you view the scene from either the EARTH or PERSON what do you see? Why is the motion the same? Can you explain the motion you see?

   What happens when you view the scene from the BALL? Draw out a diagram of what you see. Use the method of time intervals to consider what happens during each interval of time in order to explain this motion.

2. If the PERSON begins to walk forwards at a constant speed, what now happens to the BALL when it is tossed into the air?

3. A stationary person throws a ball into the air at a chosen speed and elevation. A walking person performs the same motion with a similar ball. Which ball travels furthest vertically? Horizontally? Can you explain your findings?

4. Can you think of a way to prove that when you toss a ball into the air its horizontal speed remains constant?

5. Two similar balls are placed in a tower. One ball is dropped and at the same time the other is projected horizontally. Which ball reaches the ground first?

   What happens if the second ball is projected at 30 degrees above the horizontal? At 30 degrees below the horizontal?

Set up a demonstration of this problem to check your answers.

6. Perform the same experiment of dropping a ball and projecting a ball simultaneously from a tower moving horizontally.

From a tower moving constantly up an incline at 45 degrees to the horizontal.

7. An archer fires an arrow from the back of a moving chariot. The arrow's speed is the same as the chariot's speed, only in the opposite direction.

Write a procedure DEMO.ARROW to illustrate this. What is the path of the arrow? Does it produce the results you expect?

What do you conclude from this about the firing speed of arrows?

8. A gun is sighted on a target at the top of a tower. The target is dropped from the tower at precisely the same time as a bullet is fired from the gun. Show that whatever the initial speed of the bullet, it always hits the falling target.

Can you explain this phenomenon?

9. Water is coming out of a hose with speed V. The hose is inclined at 45 degrees above the horizontal. What is the path of the water in falling to the ground?

What path will the water follow if the hose is moved with speed V in a direction opposite to that in which the hose is inclined?

When these demonstrations are shown on the computer screen only the first drop of water coming out of the hose is shown. What do you see in each case as the water constantly comes out of the hose?

10. A child sitting in a school bus, moving at a constant speed, throws a ball straight up into the air. Will the ball fall behind the child, in front of the child, or will the child catch the ball?

    What happens if the bus accelerates while the ball is in the air?

    What happens if the bus goes round a curve while the ball is in the air?

11. Two balls are thrown simultaneously into the air. One is thrown with twice the speed of the other, on the same heading. How does the slower ball see the faster one move?

    Show that the second ball viewed from the first, and vice versa, always moves in the same way regardless of what speed you project the balls with and what elevation you project them at.

4 Circular Motion

In section 2.2 we considered straight line motion and how we move a turtle across the computer screen by repeatedly giving it the command FORWARD *:amount*. For example, if we give a turtle the command FORWARD 10 the turtle moves forwards 10 units in the direction in which it is heading. If we give this command repeatedly the turtle moves in a straight line as though it has a constant speed of 10 units. FORWARD 20 moves the turtle a distance of 20 units. If we give this command repeatedly to the turtle it moves forwards at a constant speed of 20 units. Not only can we move turtles in straight lines we can move them in circles too. Let us consider how we do this.

4.1 Turtles and Circular Motion

Imagine yourself walking in a circle. What actions do you perform to move in a curve? You take a step and turn a small amount, take another step, and turn a small amount, and continue like this until you complete a circle. Try it! You can give this same motion to a turtle: take a step, turn, take another step, turn, and repeat this until a circle is drawn. You tell a turtle to move forwards using the command FORWARD and to turn using the command RIGHT. You can combine these two commands and repeating them to make the turtle move in a circle. For example, the following two commands both draw circles:

```
REPEAT 360  [FORWARD 1  RIGHT 1]
REPEAT 36  [FORWARD 10  RIGHT 10]
```

To complete a circle the turtle must turn through 360 degrees. Try drawing the circle:

```
REPEAT 360  [FORWARD 1  RIGHT 1]
```

The circle takes some time to draw because the commands are repeated 360 times before the circle is complete. Now try:

```
REPEAT 36  [FORWARD 10  RIGHT 10]
```

This circle draws 10 times faster than the first circle. The command is repeated 36 times to

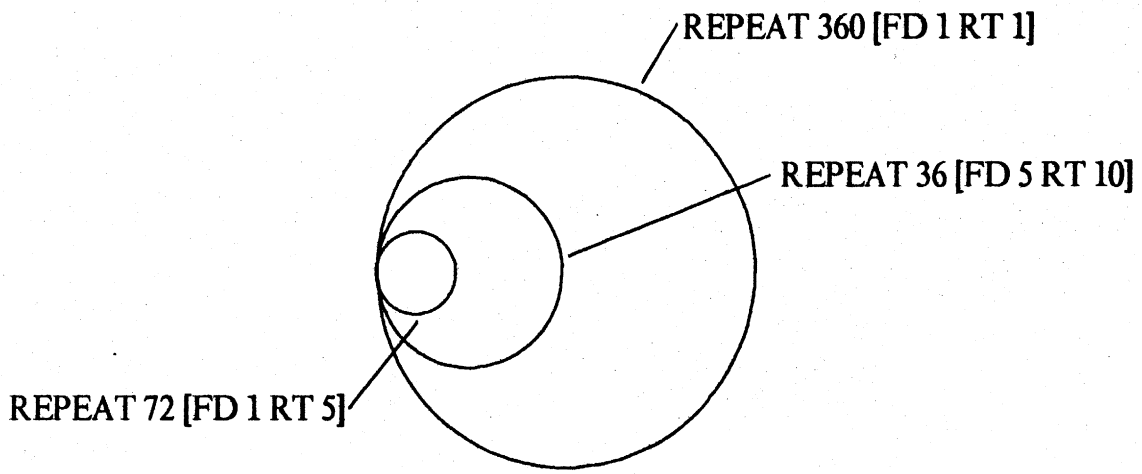

Figure 20: Circles drawn by the turtle

form a circle. Figure 20 shows some circles drawn by the turtle. What happens if you change the input to the FORWARD command to 5 units? The command is:

```
REPEAT 36 [FORWARD 5 RIGHT 10]
```

The following exercises give you practice in drawing circles and allow you to experiment giving different values to the FORWARD and RIGHT commands.

4.1.1 Exercises

1. Try an experiment drawing circles with different inputs to the FORWARD command. Try values 1, 2, 5, 10, and 20. How does this effect the radius of the circle that is drawn? (You can use the procedure CIRCLEA shown below to save on typing.)

   What is the relationship between the radius of the circle drawn with FORWARD 1 and that made with FORWARD 2?

The procedure CIRCLEA requires one input parameter, the amount to move forwards. Once the procedure is defined, type CIRCLEA 2 to draw a circle that will be made from [FORWARD 2 RIGHT 10].

```
TO CIRCLEA  :X
  REPEAT 36  [FORWARD :X  RIGHT 10]
END
```

2. What happens if you vary the angle to turn through? You can use procedure CIRCLEB shown below to do this. Try the following values for the RIGHT command: 5, 10, 20 and 40.

   What is the relationship between the radius of the circle made with RIGHT 20 and that made with RIGHT 40?

   The turtle must turn through 360 degrees to draw a complete circle. If on each step the turtle turns through 10 degrees, the turtle must repeat this 36 times to complete the circle. CIRCLEB takes as input the number of times it is necessary to repeat the motion to make a complete circle and the value of the angle.

```
TO CIRCLEB  :REPNUM  :ANGLE
  REPEAT  :REPNUM  [FORWARD 5  RIGHT :ANGLE]
END
```

   For example, CIRCLEB 36 10 draws a circle made repeating [FORWARD 5 RIGHT 10] 36 times.

4.2 Uniform Relative Circular Motion

Now that we know how to draw circles we can make two turtles move on circular paths and look how one of them moves from the other one. We will find some interesting forms of motion occur when we look at a moving turtle from another moving turtle. We use exactly the same method as described in earlier exercises: make the turtles, give them a motion, choose a reference frame, and set the turtles moving.

4.2.1 Setting Up Circular Motion

First we have to make the turtles using MAKETURTLE. We need to choose coordinates to start the turtles at and give each turtle an initial heading. When we used CIRCLEA or CIRCLEB to make circles, the circles did not have the same center. If the turtle starts at its HOME position on the screen, each turtle has this point on its circumference, see Figure 20.

How can we set up turtles so that they will have the same center? The easiest position to choose for each circle is where the circle crosses the x-axis or the y-axis, as shown in Figure 21. If the radius of the circle is r units, then the coordinates of point A are $x = r$, $y = 0$. The coordinates of point D are $x = 0$, $y = r$.

If we know the value of the radius of the circle we want to draw, then we can use this in specifying the initial position of the turtle. We also need to specify the heading of the turtle. When a turtle is moving in a circle what is its heading as it crosses the x-axis or the y-axis? Figure 21 shows this. You can verify these headings by drawing a circle with the turtle showing on the screen. Choose a circle that draws fairly slowly so that you can see the turtle at the points it crosses the axes. Notice that the turtle is pointing down at position A. This is equivalent to a heading of 180 degrees. The turtle is pointing up at position C, on a heading of 0 degrees.

To move a turtle in a circle we must repeatedly give the turtle a FORWARD and a RIGHT motion. The exercises in the previous section showed you that both the amount given to the FORWARD command and the amount given to the RIGHT command have an

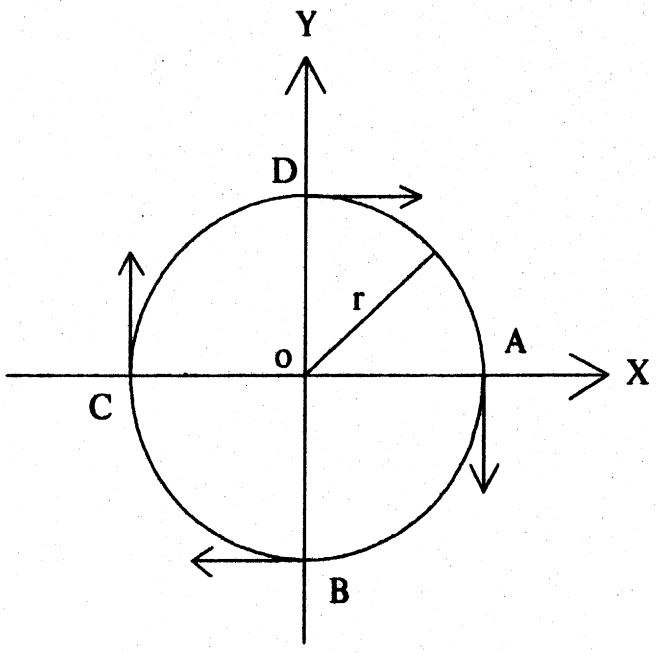

Figure 21: Setting up a turtle to move in a circle

effect on the radius of the circle drawn. A *larger* amount given to FORWARD produces a circle with a *larger* radius. A *larger* amount given to RIGHT produces a circle with a *smaller* radius.

When we set up a turtle to move in a circle we need to know its radius in order to initially position the turtle. We give the turtle a motion in terms of FORWARD and RIGHT, which both have an effect on the radius of the circle drawn. We can use a procedure RADIUS to calculate the value of the circle's radius:

```
RADIUS   :amount.forward  :amount.right
```

RADIUS has two input parameters, the amount we give to the FORWARD command and to the RIGHT command to make the circle. For example,

```
RADIUS   5   10
```

calculates the value of the radius of a circle made from a motion of [FORWARD 5 RIGHT 10]. The value returned is 28.64.

```
RADIUS   10   10
```

calculates the value of the radius of a circle made from [FORWARD 10 RIGHT 10]. The value returned is 57.28.

We can use the procedure RADIUS directly in MAKETURTLE commands. MAKETURTLE needs the initial position of a turtle as input. If we set up our turtle at point A, as shown in Figure 21, we need $x = radius$, $y = 0$. If a turtle, CIRCLE1, moves in a circle made from repeating [FORWARD 5 RIGHT 10], then the MAKETURTLE command using RADIUS directly is:

```
MAKETURTLE   "CIRCLE1  (RADIUS  5  10)  0  180  :RED
```

This sets up CIRCLE1 at position x = 28.64, y = 0, on a heading of 180 degrees, with a red drawing pen.

4.2.2 An Example of Circular Motion

Let us consider an example of setting up two turtles moving in circles about the center of the screen, with motions given by:

```
[FORWARD 5  RIGHT 5]
[FORWARD 5  RIGHT 10]
```

The procedure DEMO.CIRCLE illustrates this:

```
TO DEMO.CIRCLE  :VIEWFROM
  MAKETURTLE  "CIRCLE1  (RADIUS  5 5)  0  180  :GREEN
  MAKETURTLE  "CIRCLE2  (RADIUS  5 10)  0  180  :RED
  MAKETURTLE  "ORIGIN  0  0  0  :YELLOW
  SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT 5]
  SETMOTION  "CIRCLE2  [FORWARD 5  RIGHT 10]
  SETMOTION  "ORIGIN [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [CIRCLE1  CIRCLE2  ORIGIN]
END
```

We position CIRCLE1 initially at x = *radius of CIRCLE1*, $y = 0$, on a heading of 180 degrees and CIRCLE2 at x = *radius of CIRCLE2*, $y = 0$, on a heading of 180 degrees. We include a stationary ORIGIN so we can use it as a reference frame to view both CIRCLE1 and CIRCLE2. Once we define the procedure DEMO.CIRCLE then

```
DEMO.CIRCLE "ORIGIN
```

shows the motion of both circles. This is shown in Figure 22.

```
DEMO.CIRCLE "CIRCLE1
```

shows the motion of CIRCLE2 and the ORIGIN as seen by CIRCLE1 (see Figure 23).

The following exercises use DEMO.CIRCLE. Use this procedure to set up turtles moving in circles and then view their motion from different reference frames.

4.2.3 Exercises

1. Setup the circles described in DEMO.CIRCLE. Look at the scene using the ORIGIN as the reference frame, then CIRCLE1 as the reference frame, and finally CIRCLE2 as the reference frame.

2. Figure 23 shows DEMO.CIRCLE from CIRCLE1 as the reference frame. The

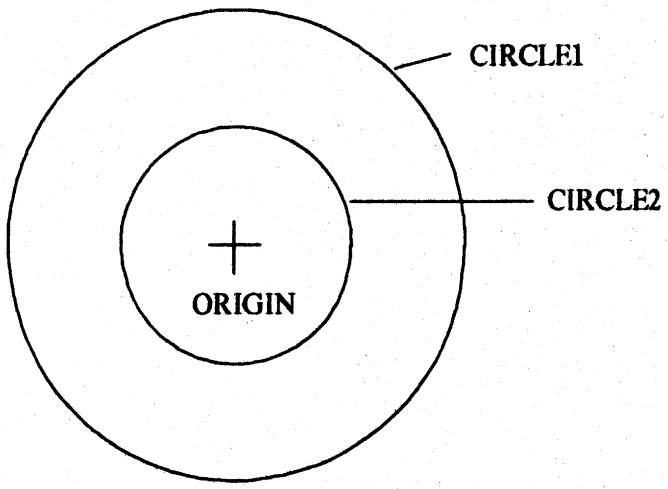

Figure 22: CIRCLE1 and CIRCLE2 as seen by the ORIGIN

central ORIGIN is seen to move in a circle. Imagine yourself standing on CIRCLE1 and consider the ORIGIN, time interval by time interval, to explain why you see the ORIGIN moving in a circle from CIRCLE1.

3. Consider the following two circles formed by:

```
SETMOTION  "CIRCLE1  [FORWARD 6  RIGHT 10]
SETMOTION  "CIRCLE2  [FORWARD 10  RIGHT 10]
```

Edit the procedure DEMO.CIRCLE to give these motions to the two turtles. (You will need to change the inputs to the procedure RADIUS, and also the SETMOTION command.)

When you view from the stationary ORIGIN as reference frame you see the circles being drawn at the same rate. (This is because each turtle turns through the same angle, RIGHT 10, in drawing the circles.)

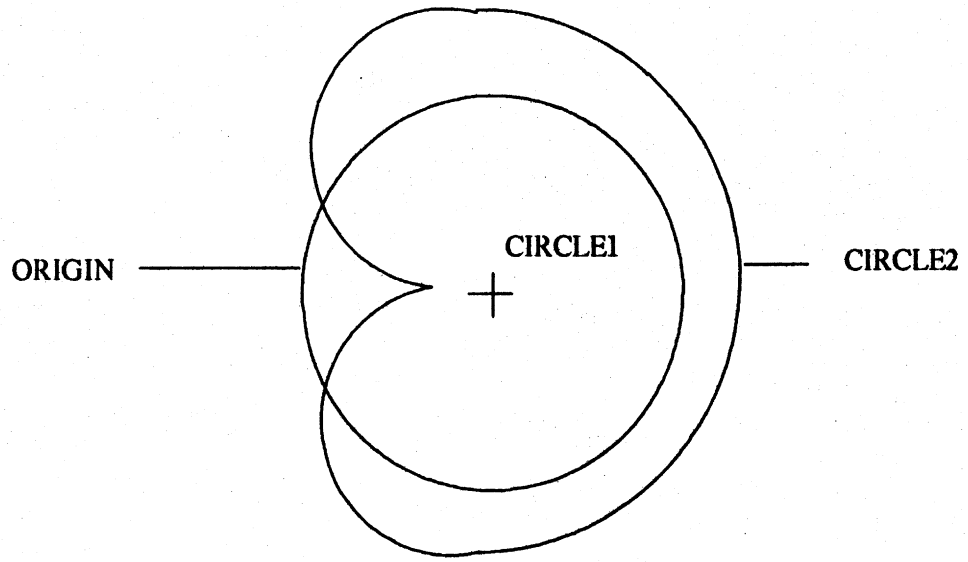

Figure 23: Motion seen by CIRCLE1

If you view from CIRCLE1, why do you see the ORIGIN and CIRCLE2 in the same orbit? From this frame of reference can you prove that the ORIGIN and CIRCLE2 do not have the same orbit?

When you view the system from CIRCLE2, the motion appears the same as that viewed from the ORIGIN. Why is this?

4.3 Retrograde Motion

The retrograde motion of the planets is a very interesting example of viewing circular motion from a reference frame that is itself moving in a circle. Retrograde motion is when a planet appears to reverse its direction of motion, then slow down, stop and then continue in its original direction of motion. Loops are formed on the planet's path (see Figure 24). These loops are called epicycles. If we wish to see a planet move with retrograde motion we

must observe the planet over the course of several weeks. The planet appears to move very slowly due to its great distance from the Earth.

Ancient astronomers were startled by this retrograde motion of the planets and produced many elaborate theories to try to explain it.

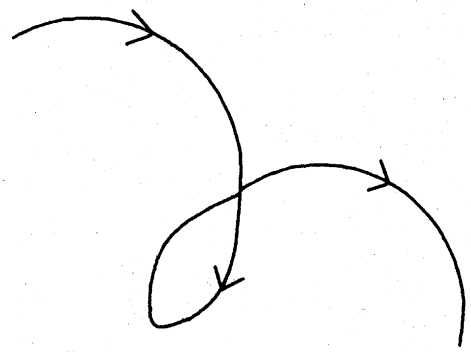

Figure 24: Retrograde motion

They considered the Earth to be stationary at the center of the solar system, with the planets orbiting the Earth on circular paths. One model explained the looping motion using a planet moving in a small circular orbit. The center of this orbit moved on a circular path about the Earth. Figure 25 shows this. When the planet moves from point A to point B, the planet appears to reverse its direction of motion.

The Earth is not at the center of the solar system, the Sun is, and all the planets, including the Earth, orbit the Sun. We can now explain the retrograde motion of the planets in terms of the relative motion of each planet with respect to the Earth. The Earth is orbiting the Sun and we must consider it a moving reference frame.

Let us look at some examples to illustrate the retrograde motion of the planets as seen from the Earth. We can set up turtles to represent the planets moving with circular motion as described earlier. We must choose values for each planet's FORWARD and RIGHT

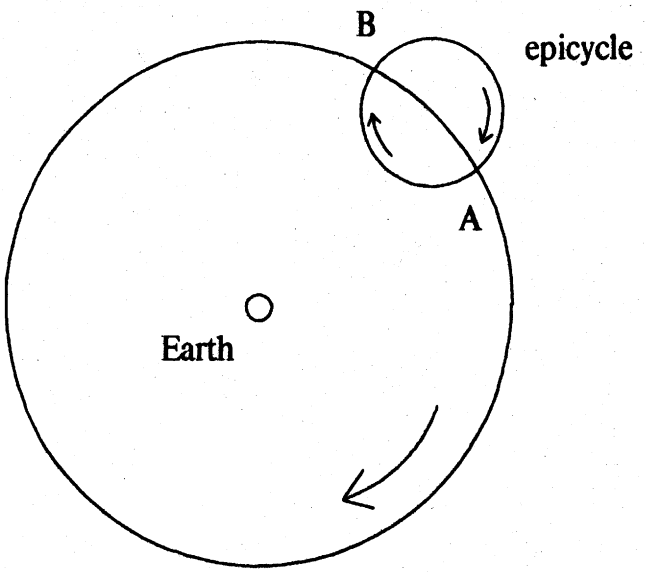

Figure 25: An ancient model to explain retrograde motion

commands to accurately represent its orbit about the Sun. The RIGHT command effects the period of revolution and both commands effect the radius of the orbit. Table 2 lists the distance of each planet in the solar system from the Sun and its period of revolution.

Planet	Distance from Sun (x 10^6 Km)	Period of revolution (Years)
Mercury	58	.24
Venus	108	.62
Earth	150	1.00
Mars	228	1.88
Jupiter	778	11.86
Saturn	1427	29.46
Uranus	2869	84.10
Neptune	4498	164.80
Pluto	5900	248.40

Table 2: Distances and periods of the planets

We must scale the planets' orbits to fit on the computer screen. We can use a procedure SCALE.SOLAR to do this:

```
SCALE.SOLAR  :list.of.planets
```

SCALE.SOLAR takes as input a list of the planets to be shown on the screen. It returns the values that we must use for each planet's radius, FORWARD *:amount* and RIGHT *:amount*, to accurately draw the motion of the planet. For example:

```
SCALE.SOLAR  [EARTH MARS]
```

returns the following information:

```
EARTH:  radius 59  forward 6  right 6
MARS:  radius 90  forward 5  right 3
```

If we set up a demonstration of the Earth and Mars orbiting the Sun, we must move the Earth on a circle of radius 59 units, with a motion [FORWARD 6 RIGHT 6] and move Mars on a circle of radius 90 units, with a motion [FORWARD 5 RIGHT 3]. This accurately represents both the Earth and Mars in the solar system.

The following procedure DEMO.SOLAR illustrates the motion of the Earth and Mars about the Sun.

```
TO DEMO.SOLAR  :VIEWFROM
  MAKETURTLE  "EARTH  0  59  90  :GREEN
  MAKETURTLE  "MARS  0  90  90  :RED
  MAKETURTLE  "SUN  0  0  0  :YELLOW
  SETMOTION  "EARTH  [FORWARD 6  RIGHT 6]
  SETMOTION  "MARS  [FORWARD 5  RIGHT 3]
  SETMOTION  "SUN [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [EARTH MARS SUN]
END
```

In this example we set up the EARTH at $x = 0, y = 59$, on a heading of 90 degrees (see Figure 21) and MARS at $x = 0, y = 90$, on a heading of 90 degrees. The following exercises use DEMO.SOLAR.

4.3.1 Exercises

1. Use the procedure DEMO.SOLAR to show the epicycle nature of Mars as seen from the Earth. Compare this with Figure 25 to see how this fits in with the ancient model of the solar system.

2. Do all the planets show retrograde motion as seen from the Earth? Use DEMO.SOLAR to look at some of the planets from the Earth, such as Mercury and Jupiter.

   Is the Sun seen to move with retrograde motion from the Earth? Can you explain why its motion is different from the other planets' motion?

3. When you see a planet, such as Mercury, from the Earth move with retrograde motion, what position is Mercury in with respect to the Earth?

4. How is the retrograde motion of an inner planet, such as Mercury, seen from the Earth, different from that of an outer planet, such as Jupiter?

4.4 Investigations of Uniform Circular Motion

The orbiting solar system produced some good examples of retrograde motion when viewed from the Earth. Not all orbiting objects produce retrograde motion. For example, our initial example DEMO.CIRCLE, seen from CIRCLE1, did not produce retrograde motion; CIRCLE2 appeared to move on a cusp shaped path (see Figure 23). We will consider the characteristics necessary to produce this phenomenon in this section. The section is in two parts. The first part, Exercises 4.4.1, considers how we can determine the size of a circle in terms of how we constructed it using the FORWARD and RIGHT

commands. In addition, the exercises look at the number of epicycle loops produced when we view one circle from the other. In the second set of exercises, Exercises 4.4.3, we will take a much closer look at why epicycles occur in some cases but not in others.

4.4.1 Exercises

1. Consider the two circles produced using the following motions:

```
[FORWARD 5  RIGHT 5]
[FORWARD 10  RIGHT 20]
```

   Which of the two circles is the largest one? Can you think of a way to determine this other than drawing the circles on the computer screen? [Hint: Remember what happens when we separately change the input to the FORWARD command, and to the RIGHT command.]

2. Test out your theory with the following two circles:

```
[FORWARD 10  RIGHT 10]
[FORWARD 5  RIGHT 40]
```

3. What happens in the case of the following circles? Does your theory predict this correctly?

```
[FORWARD 10  RIGHT 10]
[FORWARD 5  RIGHT 5]
```

4. The following circles produce epicycles when the motion of one circle is viewed from the other circle:

```
[FORWARD 10  RIGHT 20]
[FORWARD 5  RIGHT 5]
```

Edit DEMO.CIRCLE to show the motion of these circles. (Remember to edit the inputs to the RADIUS procedure as well as the SETMOTION command.)

The first circle is drawn 4 times faster than the second because the first circle's angle to turn through on each step is 4 times larger than the second circle's angle to turn through. You can see this using DEMO.CIRCLE "ORIGIN. When CIRCLE1 completes a revolution only 1/4 of CIRCLE2 is drawn.

When CIRCLE1 is the reference frame the motion of CIRCLE2 is seen. The motion is shown in Figure 26. Why are three epicycle loops produced instead of four, as CIRCLE1 is moving 4 times faster than CIRCLE2? Use diagrams and the time interval method to help you explain your reasons. [Hint: Consider the actual positions of each turtle when the motion of one turtle from the other shows a loop being formed.]

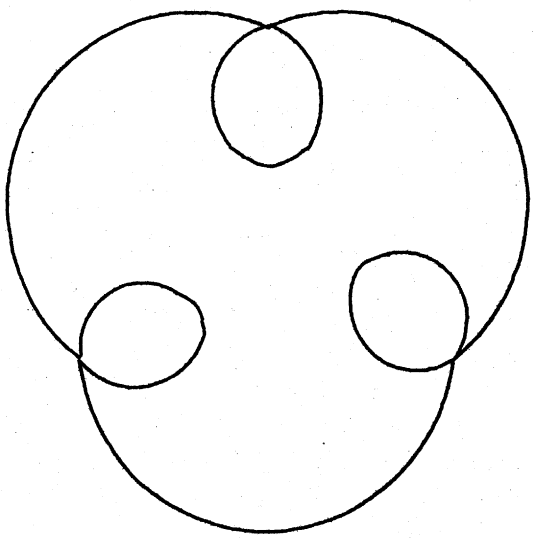

Figure 26: A track with three epicycle loops

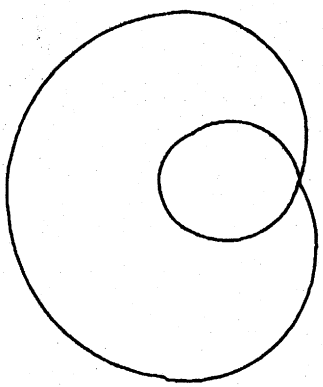

Figure 27: A track with one epicycle loop

5. The circles made with the following motions produce an epicycle when the motion of one circle is viewed from the other:

```
[FORWARD 15  RIGHT 20]
[FORWARD 10  RIGHT 10]
```

The motion of CIRCLE2 seen from CIRCLE1 produces the track shown in Figure 27. CIRCLE1 is moving twice as fast as CIRCLE2, so why is only one loop drawn instead of two? Again use the time interval method to help you understand this motion.

4.4.2 Epicycles and Cusps

We will now take a look at the conditions necessary to produce epicycles when we view an object moving in a circle from a reference frame moving in a circle. For example:

```
SETMOTION  "CIRCLE1  [FORWARD 10  RIGHT 20]
SETMOTION  "CIRCLE2  [FORWARD 6  RIGHT 6]
```

produce a track with three epicycle loops, as shown in Figure 26, when CIRCLE2 is viewed from CIRCLE1. However, the motions:

```
SETMOTION  "CIRCLE1  [FORWARD 6  RIGHT 10]
SETMOTION  "CIRCLE2  [FORWARD 6  RIGHT 6]
```

produce a figure with one cusp, as shown in Figure 28. The motions:

```
SETMOTION  "CIRCLE1  [FORWARD 6  RIGHT 20]
SETMOTION  "CIRCLE2  [FORWARD 6  RIGHT 6]
```

produce a figure with three cusps, as shown in Figure 29.

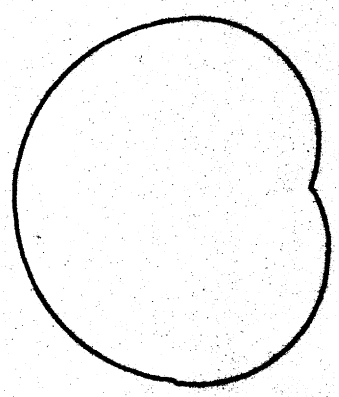

Figure 28: A one cusp track

In these cases, the properties associated with each circle determine whether epicycles or cusps are seen.

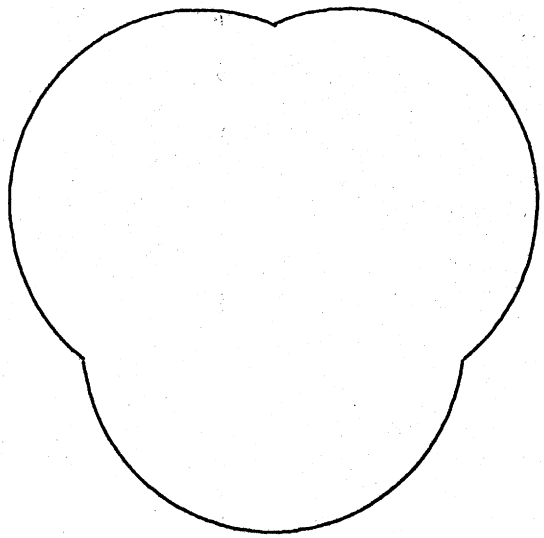

Figure 29: A three cusp track

In the previous set of exercises we looked at the size of each circle in terms of the amounts we give to the FORWARD and RIGHT commands. Both commands have an effect on the size of the circle drawn. The radius is *proportional* to the amount we give to the FORWARD command and *inversely proportional* to the amount we give to the RIGHT command. (Inversely proportional to RIGHT *:amount* means proportional to 1/RIGHT *:amount.*) This means we can determine which circle is the larger by looking at the ratio of the FORWARD *:amount* to the RIGHT *:amount.* For example:

```
SETMOTION  "CIRCLE1  [FORWARD 5  RIGHT  10]
SETMOTION  "CIRCLE2  [FORWARD 10  RIGHT 10]
```

The ratio of the FORWARD *:amount* to the RIGHT *:amount* for CIRCLE1 is 1/2 (5/10) and for CIRCLE2 is 1 (10/10). CIRCLE2 is the larger of the two circles, with a radius twice that of CIRCLE1.

The amount we give to the RIGHT command determines how quickly the circle is drawn in on the computer screen. When a circle made using RIGHT 10 completes one revolution a circle made using RIGHT 20 has completed two revolutions. The second circle draws twice as fast as the first. The amount we give to the RIGHT command is proportional to ω, the *angular velocity.* The angular velocity, ω, is how much the angle changes at the center of the circle. The time to complete one revolution, the period of revolution, is *inversely proportional* to the RIGHT *:amount.* For example, the second circle takes half the time of the first to complete a revolution.

Mathematically we can define the linear speed v of an object moving in a circle by:

$$v = r * \omega,$$

where r is the radius of the circle and ω the angular velocity.

We set up our circles using the command FORWARD and RIGHT. How do these commands effect the linear speed, v, of an object? The radius, r, is proportional to the ratio of FORWARD/RIGHT and the angular velocity, ω, is proportional to the RIGHT *:amount.* If we combine these two we have v is proportional to:

(FORWARD / RIGHT) * RIGHT.

Therefore, v is proportional to the FORWARD *:amount.*

When we considered straight line motion, repeatedly giving a turtle the command FORWARD 10, made the turtle move forwards constantly with a speed of 10 units. Repeatedly giving a turtle the motion FORWARD 20, made the turtle move twice as fast. The motion was in a straight line in the direction that the turtle was originally heading.

When we draw a circle, after each FORWARD command, the turtle turns. However, we still think of the speed as being proportional to the *:amount* given to the FORWARD command. We verify this using the mathematical equation,

$$v = r * \omega.$$

Now we will look at the conditions necessary to produce epicycles and cusps. In the following set of exercises, remember that the FORWARD *:amount* is proportional to the linear speed of the object, the RIGHT *:amount* is proportional to the object's angular velocity and the object's period of revolution is *inversely* proportional to the RIGHT *:amount.*

4.4.3 Exercises

1. Set up DEMO.CIRCLE with the equations which produced the three cusp figure:

```
[FORWARD 5  RIGHT 20]
[FORWARD 5  RIGHT 5]
```

Investigate how the motion of CIRCLE2 changes, when you view from CIRCLE1, as you use different values for CIRCLE1's FORWARD command. Use values over the range 2 - 10. To do this you can edit the procedure DEMO.CIRCLE to have the *:amount* given to FORWARD as an input parameter. You always want to view the motion from CIRCLE1, so CIRCLE1 can be made the reference frame in the procedure. The modified procedure is shown below:

```
TO DEMO.CIRCLE  :amount
  MAKETURTLE  "CIRCLE1  (RADIUS  :amount 20)  0  180  :RED
  MAKETURTLE  "CIRCLE2  (RADIUS  5 5)  0  180  :GREEN
  MAKETURTLE  "ORIGIN  0  0  0  :YELLOW
  SETMOTION  "CIRCLE1  [FORWARD  :amount  RIGHT 20]
  SETMOTION  "CIRCLE2  [FORWARD 5  RIGHT 5]
  SETMOTION  "ORIGIN [ ]
  MAKEFRAME  "CIRCLE1
  MOVE  [CIRCLE1  CIRCLE2  ORIGIN]
END
```

2. What is the value given to CIRCLE1's FORWARD command when epicycles start to appear? How does this compare with the value of CIRCLE2's FORWARD *:amount*?

3. Let *:amount* have a value of 10. What do you see when you run DEMO.CIRCLE? Now switch the values given to the RIGHT commands so that CIRCLE1 has RIGHT 5 and CIRCLE2, RIGHT 20. What happens now when you run DEMO.CIRCLE? (Remember to edit both the RADIUS and SETMOTION commands.)

4. Do the same exercise of varying the *:amount* to CIRCLE1's FORWARD command using the following motions:

```
[FORWARD :amount RIGHT 40]
[FORWARD 4 RIGHT 4]
```

(Remember to edit both the RADIUS and SETMOTION commands.)

5. Let *:amount* have a value of 7. What do you see when you run DEMO.CIRCLE? Now switch the values that you give to the RIGHT commands, so that CIRCLE1 has RIGHT 4 and CIRCLE2 has RIGHT 40. What happens now when you run DEMO.CIRCLE?

6. Questions 1 - 5 show that epicycles are not necessarily produced, when we view an object moving in a circular orbit, from another moving in a circular orbit. Under what conditions are epicycles produced? [Hint: Compare the values of both the FORWARD *:amount* and the RIGHT *:amount* given to each circle?] What do your results mean in terms of the object's linear speed and period of revolution?

Test out your theory with the equations:

```
[FORWARD 15  RIGHT 30]
[FORWARD 5  RIGHT 5]
```

What about the equations:

```
[FORWARD 5  RIGHT 30]
[FORWARD 10  RIGHT 10]
```

7. If your theory holds up when you test it out on the equations given in Question 6, you may think that it is incorrect when you try the following equations:

```
[FORWARD 5  RIGHT 30]
[FORWARD 10  RIGHT 20]
```

Your theory should predict that epicycles are not seen. Using DEMO.CIRCLE and viewing from CIRCLE1 with these values, you see a loop made! Why is the loop produced *not* an epicycle?

8. Question 6 deals with the characteristics necessary to produce epicycles in terms of FORWARD *:amount* and RIGHT *:amount.* The object's angular velocity is proportional to the RIGHT *:amount* and the linear speed of an orbiting object is proportional to the FORWARD *:amount.* Show that the planets in the solar system have these same characteristics, in that retrograde motion is seen when the planet with the larger angular velocity has the larger linear speed. [Hint: Use the equation $v = r * \omega$ and the information from Table 2 to calculate the speed of each planet. As a check for your calculation the Earth's linear speed is 29.89 Km/sec.]

9. Figure 30 shows three orbiting planets lined up along a radius from the center of the system. A has a forward motion of 4 units, B a forward motion of 10 units, and C a forward motion of 8 units. The circles were drawn by giving turtles the

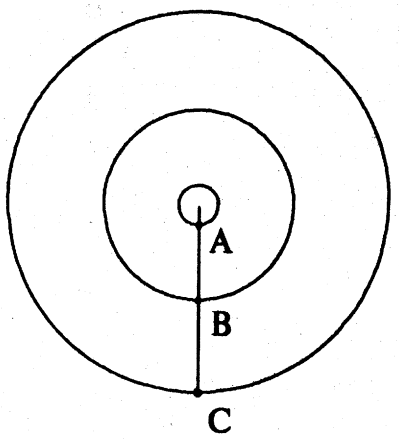

Figure 30: Orbiting Planets

following motions:

```
SETMOTION  "A  [FORWARD 4  RIGHT 40]
SETMOTION  "B  [FORWARD 10  RIGHT 20]
SETMOTION  "C  [FORWARD 8  RIGHT 10]
```

Explain, using diagrams, why A sees C forming a cusp, yet B sees C forming an epicycle. [Hint: Consider what happens on the steps before the planets line up, as they line up, and after they line up.]

10. All the previous exercises consider orbiting planets orbiting about the same point. The orbits are concentric. In our solar system some of the planets have moons. For example, the Earth has the Moon and Jupiter has at least thirteen moons, some of the more familiar ones being Io, Ganymede and Europa. We can set up orbiting planets with moons using the SETRELATIVEMOTION command. The Moon orbits the Earth, while the Earth orbits the Sun. The Moon is moving relative to the Earth, as the Earth moves on its own orbit. The procedure DEMO.MOON shows this:

```
TO DEMO.MOON  :VIEWFROM
  MAKETURTLE  "EARTH  (RADIUS  5 5)  0  180  :GREEN
  MAKETURTLE  "MOON  ((RADIUS  5 5) + (RADIUS  5 20))
                                     0  180  :RED
  MAKETURTLE  "SUN  0  0  0  :YELLOW
  SETMOTION  "EARTH  [FORWARD 5  RIGHT 5]
  SETRELATIVEMOTION  "MOON  "EARTH  [FORWARD 5  RIGHT 20]
  SETMOTION  "SUN  [ ]
  MAKEFRAME  :VIEWFROM
  MOVE  [SUN EARTH MOON]
END
```

We position the Moon with x-coordinate as the sum of the radii of the Earth's orbit about the Sun and the Moon's orbit about the Earth, see Figure 31. We give the Earth a circular motion and the Moon a circular motion with respect to the Earth. We use the SETRELATIVEMOTION command to do this.

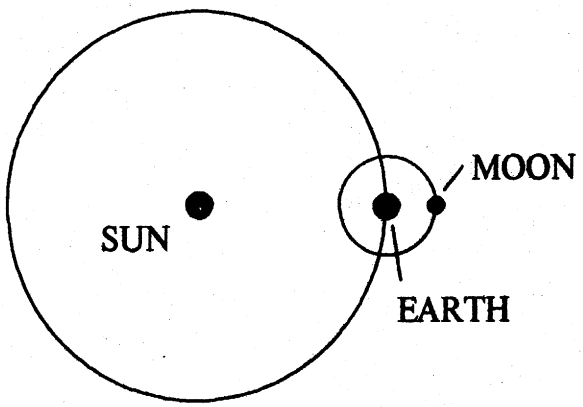

Figure 31: Initial positions of the Earth and the Moon

Set up a similar demonstration to the one outlined in DEMO.MOON. What do you see when the reference frame is the Sun? The Earth? The Moon? Is this what you expect?

11. The previous exercise used the SETRELATIVEMOTION command to show how a moon of an orbiting planet can be set up. We cannot set up a scaled demonstration of our Sun-Earth-Moon system. The Earth - Sun distance is 150×10^6 Km. and the Earth - Moon distance is on average 384,400 Km. The ratio of these two is 1 : 0.0026. However, it is possible to get a feel for how these orbiting planets move.

    Using DEMO.MOON with different values of *:amount* given to the Moon's FORWARD command, what happens to the path of the Moon as seen by the Sun? Does this remind you of a previous exercise?

    The Moon takes approximately 28 days to orbit the Earth. It performs 13 revolutions about the Earth, while the Earth does 1 revolution about the Sun. What is the path of our Moon as seen by the Sun? Are epicycles, cusps or waves (elongated cusps) seen? [Hint: Find the linear speed of the Moon and compare to the Earth's.]

12. Show that the path of a point on a bicycle wheel, seen by a stationary person, always moves in a cycloid, as shown in Figure 3.

    In addition, show that a person riding on the bicycle sees the point move on a circular path.

    Explain the values you give to both the wheel and the bicycle in setting up their motions.

4.5 Rotating Frames of Reference

As a final section we will take a closer look at what it means to be in a rotating frame of reference and the consequences of being in such a reference frame. In section 4.3 we saw how the planets move with retrograde motion as seen from the Earth. The planets are seen to move in epicycles because we are viewing their orbits about the Sun from the Earth, a reference frame that is itself moving.

Let us now consider standing on something that is turning around in a circle, such as a merry-go-round, and try to perform some actions. Suppose you stand on the rotating merry-go-round and throw a ball to a partner, also on the merry-go-round. How do you throw the ball so that your partner can catch it? In addition, how do you see the ball move?

These are not easy questions to answer. In fact it is extremely difficult to play a game of "catch" on a rotating merry-go-round. You will need several tries and a good supply of balls before you throw a ball so that your partner can catch it. You will see the balls move in a very strange way.

How can we set up a demonstration to show this motion? Again we use the four basic steps: make the turtles, give the turtles a motion, choose a frame of reference and finally move the turtles. The procedure DEMO.PUSH illustrates these steps:

```
TO DEMO.PUSH  :VIEWFROM
  MAKETURTLE  "PERSON  (RADIUS  5 5)  0  180  :GREEN
  MAKETURTLE  "BALL  (RADIUS  5 5)  0  270  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "PERSON  [FORWARD  5  RIGHT  5]
  SETMOTION  "BALL  [PUSH  5]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.PUSH  "BALL  "PERSON
  MOVE  [PERSON  BALL  EARTH]
END
```

A person standing on a rotating merry-go-round moves in a circle. To represent this we use a turtle called PERSON, and give it a circular motion. The person throws a ball. Initially the BALL is at the same position as the PERSON. The BALL is thrown towards the center of the merry-go-round (on a heading of 270 degrees from its initial position), and is given a motion using:

```
SETMOTION "BALL [PUSH 6]
```

The procedure PUSH is a microworld primitive that gives an object a push across a surface. It requires one input parameter, the speed of the push. Before the turtles are set moving with the MOVE command the motion of the pushed turtle must be initialized with INIT.PUSH. The procedure INIT.PUSH requires two input parameters, the name of the turtle being pushed and the name of the turtle doing the pushing:

```
INIT.PUSH :name :from
```

In the above example the BALL is thrown by the PERSON and its motion is initialized with:

```
INIT.PUSH "BALL "PERSON
```

Gravity will not be considered in this example and in the exercises along with this section. We have to imagine ourselves as looking at the merry-go-round and looking straight down onto the merry-go-round.

Try this example. A person looking down on the merry-go-round, from a reference frame of the stationary EARTH, will see the ball move in a straight line, see Figure 32. The ball is thrown towards the center of the merry-go-round, but it does not move in that direction. This is because as the ball leaves the person hand it is moving as the person is moving. At this particular instant in time the person is moving tangent to the circle, on a heading of 180 degrees. The actual direction the ball moves is the result of the ball being pushed in a given direction (270 degrees in this example) and its initial motion due to being held by a moving person.

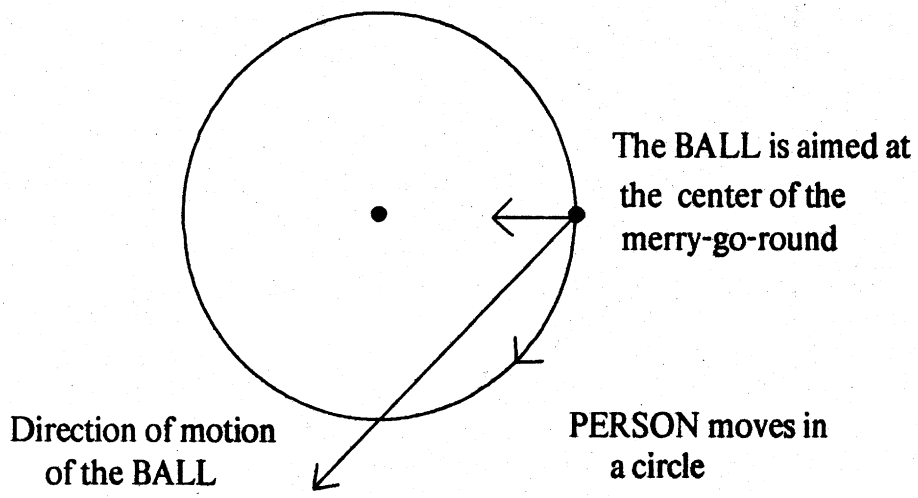

Figure 32: A ball thrown on a moving merry-go-round seen from the Earth

To the person standing on the moving merry-go-round the ball appears to move in a curve as shown in Figure 33. We use the method of time intervals to understand how this motion is formed. Figure 32 shows the actual positions of both the PERSON and the BALL, as seen from the stationary EARTH. Now we must imagine being the PERSON and for several intervals of time move with the PERSON. For each interval of time we determine the position of the BALL from the PERSON. During the first few intervals of time the BALL appears almost on a heading of 270 degrees from the PERSON. Then as the motion continues and the BALL crosses the circumference of the circle made by the PERSON, the BALL is almost on a heading of 180 degrees, regardless of where the PERSON is now positioned.

4.5.1 The Rotating Earth

The example above shows that when we view a moving object from a rotating frame of reference the motion is not what we expect. Figure 33 shows that when we stand on a rotating reference frame such as a merry-go-round and throw a ball straight towards another

Figure 33: The moving ball as seen by a person on the merry-go-round

point, the ball appears to move in a curve rather than in a straight line. This has very important consequences when we discuss Newton's laws of motion and whether they are valid in all reference frames. To illustrate this discussion let us first consider another example.

Suppose you hold a glass of ice on a flat, horizontal, stationary surface and let it go. What happens? Nothing! The ice remains where it is. What happens if you do the same experiment while you are standing on a horizontal, rotating merry-go-round? We use DEMO.PUSH to illustrate this and we represent losing go of the ice by giving the ice a motion of [PUSH 0].

```
TO DEMO.PUSH :VIEWFROM
  MAKETURTLE  "PERSON  (RADIUS 5 5)  0  180  :GREEN
  MAKETURTLE  "ICE  (RADIUS 5 5)  0  0  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "PERSON  [FORWARD 5  RIGHT 5]
  SETMOTION  "ICE  [PUSH 0]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.PUSH  "ICE  "PERSON
  MOVE  [PERSON  ICE  EARTH]
END
```

When we use DEMO.PUSH with the EARTH as the reference frame the ice moves in a straight line, on a tangent to the circle, see Figure 34.

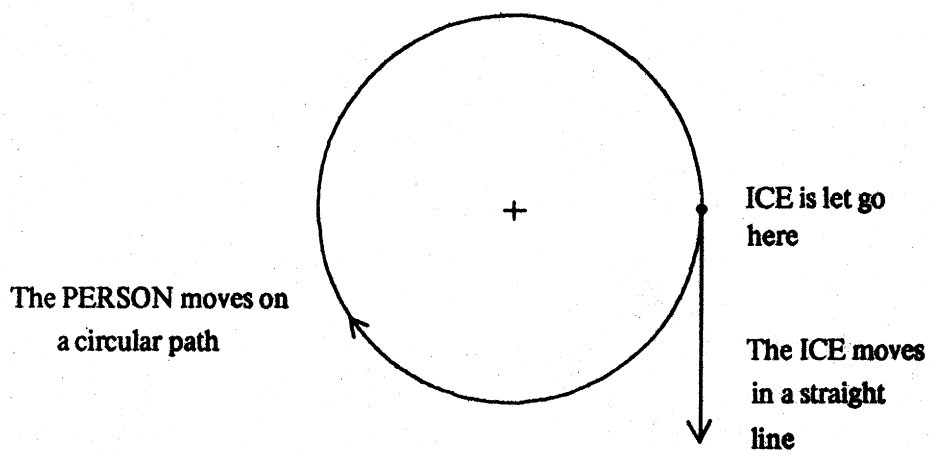

Figure 34: The ice as seen from a stationary reference frame

The ice is originally held by a person moving in a circle. At the time it is let go it has the instantaneous motion due to moving in a circle. Newton's First Law of Motion states that every object continues in its state of rest or of uniform straight-line motion unless acted on

by an external force. If we apply this law to our example we see that no force acts on the ice, as it continues to move at a constant rate in a straight line. This is shown by the demonstration:

```
DEMO.PUSH  "EARTH
```

Now let us look at the demonstration from the reference frame of the PERSON. The ice moves in a curve, as shown in Figure 35.

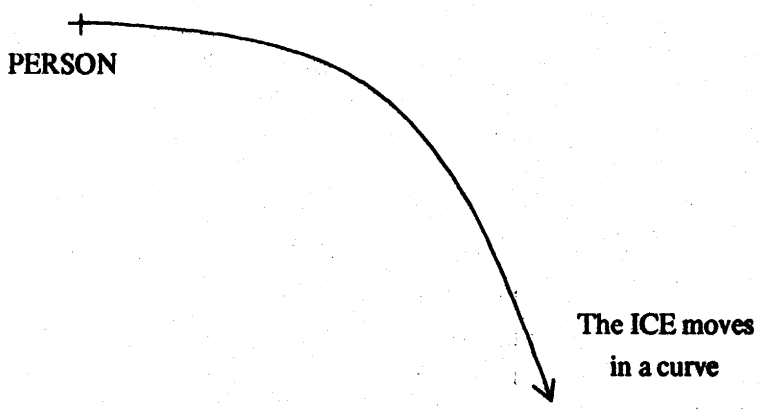

Figure 35: The ice seen from the rotating reference frame

Now suppose the person on the merry-go-round cannot see outside of the merry-go-round. The person believes s/he is standing still. So when s/he lets a piece of ice go, it should remain stationary. However, the person sees the ice move in a curve. It appears as though a force is now present to cause the ice to move in this way. Actually, *no* force is present, but because the person is in a rotating frame of reference, a force has to be invented to make Newton's laws consistent.

The rotating reference frame is an example of a *non-inertial* reference frame. Newton's laws of motion are stated using the assumption that we make all measurements

with respect to a fixed system (inertial system). If a force acts on a particle, all observers in inertial systems see the same force; the force is *invariant.* In this example, there is no force acting on the ice seen from a stationary reference frame, as the ice moves at a constant rate when let go, and yet one appears to be acting seen by the person, as the ice appears to move in a curve. The rotating frame is therefore a non-inertial system as it does not see the same force as the inertial system.

The rotating Earth is not really an inertial system. It is rotating in two ways: it orbits the Sun and it spins on its axis. A merry-go-round is often used as a comparison for the spinning Earth. When we consider the Earth and put ourselves into the reference frame of the Earth, we have to invent two forces to make Newton's laws consistent with the motion as it actually happens in a fixed reference frame. The first is the centripetal force, due to the Earth's orbit about the Sun, and the second the coriolis force, due to the Earth spinning on its axis. In most cases these forces are extremely small and we usually consider the Earth as being an inertial system. However, the Earth is like a spherical merry-go-round in that its coordinate system, the latitude-longitude grid, is always spinning, and objects do not end up where they are aimed unless corrections are made for the Earth's motion. For example, if guns are trained on a target several miles away and fired, the ammunition will miss its target by a considerable distance, if the motion of the Earth, while the ammunition is in the air, is not taken into account.

Use DEMO.PUSH in the following exercises to illustrate throwing objects while you are standing on a rotating merry-go-round.

4.5.2 Exercises

1. Two people A and B stand on a rotating merry-go-round which is rotating at a constant rate. A is at the center and B on the circumference. Assume A is very small and it is stationary at the center.

   To a person looking down on the merry-go-round from a stationary position, how does B have to throw a ball so that it appears to remain stationary?

How does B have to throw the ball so that A can catch it? Can you suggest speeds and angles that work?

How does A have to throw the ball so that B can catch it? Why does A not throw the ball in the opposite direction to B throwing the ball to A?

2. Use DEMO.PUSH to set up a PERSON with a motion of [FORWARD 5 RIGHT 5]. The PERSON throws a ball with speed 5 units on a heading of 315 degrees. To the PERSON's amazement s/he catches the ball!

   In this case how does the PERSON see the ball move? If we only see this demonstration in the rotating reference frame how do we know that the PERSON catches the ball?

   If you initially set up a PERSON at $x = 0, y = (-\ RADIUS\ 5\ 5)$, on a heading of 270 degrees, how does s/he throw the ball so that s/he can catch it?

3. If a person throws two balls simultaneously when standing on a rotating merry-go-round, what form of motion does one ball have as seen by the other ball?

Appendix B

The Teacher's Reference Manual

A Relative-Motion Microworld Teacher's Reference Manual

Linda E. Morecroft
Educational Computing Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts

September 1985

Copyright (C) Massachusetts Institute of Technology 1985

Table of Contents

Introduction	181
Exercise 2.2.2	182
Exercise 2.2.4	184
Exercise 2.3.1	186
Exercise 2.3.3	187
Exercise 3.1.3	188
Exercise 3.2.1	191
Exercise 4.1.1	200
Exercise 4.2.3	201
Exercise 4.3.1	202
Exercise 4.4.1	203
Exercise 4.4.3	205
Exercise 4.5.2	212

Introduction

This text accompanies the Relative-Motion Microworld text. It contains solutions and hints for solving the exercises listed in the student text. In addition, where appropriate, sample programs have been given.

Exercise 2.2.2

These exercises give the student practice in setting up demonstrations and getting turtles moving across the screen. The motion discussed is parallel, straight line motion. Many students do not have problems in discussing this type of motion--as often they have experienced some form of it when traveling on the highway. For example, seeing a car overtake them and travel slowly away from them, or seeing oncoming cars rushing towards them.

Question 1 Try different values of *:R* and *:W*. E.g. 10 and 5, 5 and 10, 20 and 10, 10 and 20. When you view from the EARTH as reference frame both turtles move across the screen. In the first case when $:R = 10$ and $:W = 5$, the RUNNER moves twice as fast as the WALKER.

If the RUNNER is the reference frame, then the RUNNER is at the center of the screen. The other turtles move on the screen according to how RUNNER sees them move. When the RUNNER is moving faster than the WALKER, the WALKER gets left further and further behind. RUNNER sees the WALKER moving backwards. The RUNNER also sees the EARTH traveling backwards.

When the WALKER is the reference frame, it sees the EARTH moving backwards. If the RUNNER is moving faster than the WALKER, the WALKER will see the RUNNER moving slowly ahead.

Question 2 When you are in the reference frame of a turtle that is moving forwards you see the EARTH turtle moving backwards. Imagine being in a moving car, the outside scenery appears to be moving backwards.

Question 3 RUNNER and WALKER move along at the same speed, in the same direction. From the EARTH's reference frame both move along together at the same rate.

From the RUNNER's perspective WALKER is not moving. WALKER is always alongside RUNNER. Both appear stationary on the screen. If you ride in a car and another car comes alongside you and travels at the same speed, the car appears to be stationary next to you. When you travel in a car, how can you tell you are moving? You can look out of the car window and see the trees moving backwards. On the screen the EARTH is seen to move backwards.

Question 4 A turtle appears to move backwards if you view it from a turtle that is moving forwards with a faster speed.

Question 5 When the RUNNER is set moving forwards with a speed of 10 units and the WALKER backwards with a speed of 5 units, we see the turtles move in opposite directions, from the EARTH as reference frame.

When the RUNNER is the reference frame, the WALKER appears to be quickly left behind, and it moves at a speed of 15 units relative to the RUNNER. The EARTH appears to move backwards at 10 units.

When the WALKER is the reference frame, the RUNNER appears to move forwards faster than its true speed. The RUNNER moves forwards at a speed of 15 units relative to the WALKER.

Question 6 If both RUNNER and WALKER are set moving backwards using the command DEMO "EARTH (-10) (-5), both turtles move to the LEFT across the screen. When viewed from RUNNER, WALKER appears to get left further and further behind (moving "backwards"). In this case WALKER is seen to move to the RIGHT across the computer screen. The EARTH moves to the RIGHT too, opposite to the RUNNER.

Exercise 2.2.4

Question 1 From the EARTH or A as reference frame B is seen to move with a speed of 10 units. This can be proved by measuring the distance moved by B, in a given interval of time, when A is the reference frame and when the EARTH is the reference frame. The demonstration is the same from both reference frames, even though B is moving relative to A, because both A and the EARTH are stationary.

Question 2 Turtle A is moving with a speed of 20 units and turtle B with a speed of 10 units relative to A. Turtle B therefore moves at 30 units with respect to the EARTH. From the EARTH, B is seen to move ahead of A. If the program is stopped after a short interval of time A has moved 2/3 of the distance B has moved. From turtle A, B still appears to move at a rate of 10 units.

Question 3 When A is given a backwards motion B still continues to move forwards with speed of 10 units seen by A. From the EARTH different scenes are produced depending on the value of A's speed:

Speed A = -20, B moves to the LEFT.

Speed A = -10, B remains stationary.

Speed A = 0, B moves to the RIGHT.

Speed A = 5, B moves to the RIGHT.

Question 4 This example discusses the principle of electromagnetic induction. When a wire loop is moved through a magnetic field an electric current is produced. If the wire loop is moved twice as fast, the current is doubled. If the loop is moved to the left, instead of the right, the current is reversed. This indicates that movement is important in order to induce a current.

If the magnet is moved as well as the loop in the same direction at the same rate then no current is produced.

If the wire loop is held at rest and the magnet moved a current is produced.

It is important for *relative* movement to be present. The magnet *must* see the loop moving. To get a maximum current you need a maximum *relative* motion. This can be produced by moving the magnet at its maximum speed in one direction and moving the loop at its maximum speed in the opposite direction.

Exercise 2.3.1

Question 2 If you stand on B and view A, A appears to be moving with a speed of 5 units on a heading of 307 degrees. This is opposite to the direction in which B appears to move when viewed from A. This motion can be seen if DEMO.ANGLE is run with A as the reference frame and then with B as the reference frame, without clearing the screen in between.

Question 3 P and Q are moving in zig-zags at a constant rate. To see how Q is moving from P, use the method of time intervals. Move a step, stand on P and view where Q is. Q is directly below P after one interval of time. Keep doing this for several intervals of time. P always sees Q directly below, and Q appears to move up and down a vertical line.

Question 4 These turtles do not collide, as they do not reach the same point at the same time. Looking at the motion of A and B from the EARTH's reference frame the turtles pass very close to one another. In order to determine their closest distance apart consider how one turtle moves from the reference frame of the other. When we consider both turtles moving it is impossible to determine which position of turtle A corresponds to which position of turtle B. When we stand in A's reference frame we see B moving. B leaves a trail across the screen. A is at the center of the screen at position [0,0]. The closest distance that B gets to A is the perpendicular drawn from A to B's trail.

Exercise 2.3.3

Question 1 The river flows at 4 m.p.h. If you row at 6 m.p.h. this is 6 m.p.h. relative to the river. The SETRELATIVEMOTION is used to set up this motion:

```
SETRELATIVEMOTION  "BOAT  "RIVER  [FORWARD  6]
```

Set the river moving horizontally across the screen and experiment with different headings for the boat. From the reference frame of the EARTH see how the boat is always moved downstream by the river regardless of how the boat is initially headed.

Question 2 In order to row across the river, the boat must be headed upstream, as seen by the EARTH. A heading of -42 degrees works. From the reference frame of the boat you row directly across the river.

Question 3 You can set up this problem on the computer using the information given. A knows that it is on a heading of 45 degrees at position [0,0], and is moving with a speed of 5 units. B is at position [50,0] and is heading straight for A. Therefore, its heading is 270 degrees. B is moving relative to A at a speed of 10 units. The demonstration becomes:

```
TO DEMO.BOAT  :VIEWFROM
  MAKETURTLE  "A  0  0  45  :RED
  MAKETURTLE  "B  50  0  270  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [FORWARD  5]
  SETRELATIVEMOTION  "B  "A  [FORWARD  10]
  MAKEFRAME  :VIEWFROM
  MOVE  [A  B  EARTH]
END
```

From the reference frame of A, B moves straight for A on a heading of 270 degrees. B will eventually collide with A. From the EARTH, A and B move in straight lines and collide. Their paths form an inverted V.

Exercise 3.1.3

Question 1 The PERSON standing still sees the APPLE fall straight to the ground. If the PERSON is moving forwards, the APPLE appears to move in a curve towards the PERSON, the APPLE getting closer and closer to the PERSON during each interval of time. In order to see the APPLE fall towards himself/herself the PERSON must move towards the APPLE (in a direction to the RIGHT across the screen). To see the APPLE move away from himself/herself the PERSON must move away from the APPLE (in a direction to the LEFT across the screen).

As you vary the speed of the PERSON, the faster the PERSON moves, the bigger the distances covered in each interval of time. The PERSON sees the APPLE moving almost horizontally when the PERSON is moving with a speed of approximately 40 units.

Question 2 When the PERSON stands still and the APPLE falls, the APPLE gets closer to the ground during each interval of time, and therefore closer to the PERSON. The APPLE sees the PERSON moving vertically upwards.

When the PERSON moves forwards and the APPLE falls to the ground, the APPLE sees the PERSON moving upwards in a curve, getting closer to the APPLE during each interval of time.

When the PERSON moves backwards and the APPLE falls, the APPLE sees the PERSON moving upwards in a curve, but getting further away during each time interval.

Question 3 When two apples fall simultaneously from a tree, they both move in an identical way, neglecting friction and air currents, and reach the ground at the same time. After each successive time interval they both will have fallen the same amount. If you move into the reference frame of one of the apples, you always see the other one in the same position relative to yourself. The second apple never appears to move away from the first one, and appears stationary.

Question 4 From a stationary EARTH frame of reference the PENNY moves in a curve, and the PERSON moves horizontally across the screen. The PERSON and PENNY are at the same position horizontally after each interval of time. This is because the PENNY and the PERSON both have the same horizontal speed. After each interval of time the PENNY is vertically below the PERSON. The PERSON sees the PENNY fall straight down.

If a RUNNER is introduced into the scene and travels in the same direction, at the same speed as the PERSON, then the RUNNER sees the PENNY fall straight down too.

If the RUNNER moves in the same direction with a faster speed, the PERSON is seen to move backwards and the PENNY fall backwards in a curve. The PENNY moves horizontally when the RUNNER's speed is high compared to the PERSON's, about 50 units.

When the RUNNER changes direction, the PERSON and PENNY appear to move in the opposite direction, as seen by the RUNNER.

Question 5 When the coffee leaves the spout of the coffee pot it is moving with the same speed as the plane. This example is just like a person dropping a penny.

Question 6 The demonstration can be set up as follows:

```
TO DEMO.INCLINE  :VIEWFROM
  MAKETURTLE  "PERSON  (-50)  0  45  :RED
  MAKETURTLE  "PENNY  (-50)  0  180  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "PERSON  [FORWARD  5]
  SETMOTION  "PENNY  [FALL]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.FALL  "PENNY  "PERSON
  MOVE  [PERSON  PENNY  EARTH]
END
```

To someone standing on the ground (the EARTH's reference frame) the PENNY moves upwards in a curve, then falls in a curve. When the PENNY is dropped by the PERSON moving up the incline it absorbs the initial motion of the PERSON. Initially the PENNY moves with a speed of 5 units at 45 degrees to the horizontal. The PENNY's speed can be split into two perpendicular components, vertical and horizontal. The initial vertical speed is (5 * sin 45), and the initial horizontal speed is (5 * cos 45). The horizontal speed remains constant as there is no force acting on it to change it. The vertical speed will be affected by gravity. The PENNY gradually slows down to rest as it moves against gravity, then falls under gravity. The combination of the constant horizontal speed and changing vertical speed results in the PENNY moving in a curve. The horizontal speed of both the PERSON and the PENNY remain constant, and the PERSON sees the PENNY fall straight down.

A car has to drive with a speed of (5 * cos 45) to match the horizontal speed of the PENNY.

```
SETMOTION "CAR [FORWARD (5 * COS 45)]
```

The car sees the PENNY move vertically upwards, then fall vertically.

Exercise 3.2.1

Question 1 The ball moves upwards in a curve, comes to rest, and falls in a curve. Both the PERSON and the EARTH turtles are stationary and see the ball move in the same way. To explain the motion we must use Newton's laws. We can think of the ball's speed as having two components, a vertical one and a horizontal one. Gravity acts vertically and slows the ball's vertical speed until the ball stops. Then the ball falls under gravity, and its vertical speed becomes larger and larger. Horizontally no force acts on the ball to slow it down, and its speed remains constant. Its motion is therefore a combination of a changing vertical speed and a constant horizontal speed.

If the ball is thrown at an angle θ to the horizontal with speed V, then the ball's initial vertical speed is V * sin θ, and the ball's horizontal speed is V * cos θ.

When the scene is viewed from the BALL the PERSON and EARTH both appear to move away from the BALL, downwards in a curve, getting further away from the BALL during each interval of time.

Question 2 When a ball is tossed by a person moving horizontally, the ball absorbs the horizontal motion of the person. If a ball is thrown at a speed V, at an angle of θ to the horizontal, by a person moving with a speed S, the ball's horizontal speed will be S + V * cos θ. The ball's initial vertical speed will be V * sin θ. The vertical speed is not affected by the person's horizontal motion.

The ball's increased horizontal speed causes the ball to move ahead of the person. The ball moves in a shallower curve to the horizontal due to the increase in the ball's horizontal speed.

Question 3 If we run DEMO.TOSS with two people tossing balls, the first person being stationary and the second moving at a constant horizontal speed, we see both balls travel the same distance vertically. This is because when the second person is moving

horizontally his/her speed does not contribute to the ball's vertical speed. Both balls start with the same vertical speed and move against gravity in exactly the same way.

The ball thrown by the person moving horizontally will travel furthest, due to the increased horizontal speed of the ball.

Question 4 To prove that the horizontal speed of a ball tossed into the air remains constant, set up a constantly moving turtle, CONSTANT with the same horizontal speed as the BALL. If the ball is thrown at a speed of 10 units at 60 degrees to the horizontal, the BALL's horizontal speed is 10 * cos 60 . Give this speed to the turtle CONSTANT.

```
SETMOTION "CONSTANT [FORWARD (10 * COS 60)]
```

Viewing from the EARTH as reference frame, the constantly moving horizontal turtle, CONSTANT, will be seen to keep up with the BALL.

Question 5 Both the balls reach the ground at the same time. If the second ball is projected horizontally its vertical speed is zero. It falls under gravity just like the first ball.

```
TO DEMO.TOWER
  MAKETURTLE "BALL1 (-50) 0 0 :RED
  MAKETURTLE "BALL2 (-50) 0 90 :GREEN
  MAKETURTLE "TOWER (-50) 0 90 :BLUE
  MAKETURTLE "EARTH 0 0 0 :YELLOW
  SETMOTION "BALL1 [FALL]
  SETMOTION "BALL2 [TOSS 5]
  SETMOTION "TOWER [ ]
  SETMOTION "EARTH [ ]
  MAKEFRAME "EARTH
  INIT.FALL "BALL1 "TOWER
  INIT.TOSS "BALL2 "TOWER
  MOVE [BALL1 BALL2 TOWER EARTH]
END
```

If the second ball is projected at 30 degrees above the horizontal (heading 60 degrees) it will have an initial vertical velocity upwards. The ball will move upwards in a curve until

it begins to fall under gravity. The ball falls in a curve due to its constant horizontal velocity and its increasing vertical velocity. The ball dropped from the tower will reach the ground first.

If the second ball is projected at 30 degrees below the horizontal (heading 120 degrees) it will initially have a vertical velocity due to the projection. It will reach the ground before the first ball, and will fall in a steeper curve than the ball projected horizontally.

Question 6 When the tower is moving horizontally at a constant speed of 5 units the tower has a motion of [FORWARD 5].

```
SETMOTION "TOWER [FORWARD 5]
```

When we view the scene from the EARTH we see that the two balls reach the ground at the same time. Both fall in a curve. The ball which falls from the tower falls in a steeper curve to that being projected horizontally. Both balls reach the ground at the same time because both have an initial vertical speed of zero. Horizontally BALL2 travels further, as it has its own horizontal speed of projection (from TOSS 5) and it absorbs the towers horizontal speed (5 units). BALL1 just absorbs the tower's horizontal speed.

When BALL2 is projected below the horizontal it reaches the ground first, as it has an initial vertical speed due to the throwing. If it is projected above the horizontal, it reaches the ground last, as it initially moves upwards against gravity.

When the tower is moving constantly at 45 degrees to the horizontal the MAKETURTLE command becomes:

```
MAKETURTLE "TOWER (-50) 0 45 :BLUE
```

Both balls inherit a vertical and a horizontal motion from the tower. They both move upwards in a curve under gravity, and then fall in a curve reaching the ground at the same time. They both reach the same height vertically, as they both have the same initial vertical

velocity. BALL2 travels further horizontally as its horizontal speed is the sum of its projected speed plus the horizontal speed of the tower.

When BALL2 is projected below the horizontal it reaches the ground first, due to its initial vertical speed from being projected. It reaches the ground last when it is projected above the horizontal, as initially it must move against gravity.

Question 7 The arrow's speed is the same as the chariot's speed in the opposite direction. The arrow "absorbs" the forward direction of the chariot. This cancels out the speed that the arrow is fired with horizontally. The arrow falls straight down to the ground. A demonstration to show this is:

```
TO DEMO.ARROW :VIEWFROM
  MAKETURTLE  "ARROW  0  0  (-90)  :RED
  MAKETURTLE  "CHARIOT  0  0  90  :GREEN
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "ARROW  [TOSS  5]
  SETMOTION  "CHARIOT  [FORWARD  5]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.TOSS  "ARROW  "CHARIOT
  MOVE  [ARROW  CHARIOT  EARTH]
END
```

When you fire arrows backwards from a moving chariot you must project them at a high enough speed to overcome the forward motion they inherit from the chariot.

Question 8 A target is dropped from a tower, and a bullet is fired aimed at the top of the tower. We need to choose the initial coordinates for the target. This will allow us to choose the heading on which to fire the bullet, using the primitive ATAN. The demonstration can be set up using:

```
TO DEMO.COLLIDE  :HORIZ  :VERT
  MAKETURTLE  "BULLET  0  0  (ATAN :HORIZ :VERT) :RED
  MAKETURTLE  "TARGET  :HORIZ  :VERT  180  :GREEN
  MAKETURTLE  "TOWER  :HORIZ  :VERT  0  :YELLOW
  SETMOTION  "BULLET  [TOSS  5]
  SETMOTION  "TARGET  [FALL]
  SETMOTION  "TOWER  [ ]
  MAKEFRAME  "TOWER
  INIT.TOSS  "BULLET  "TOWER
  INIT.FALL  "TARGET  "TOWER
  MOVE  [BULLET  TARGET  TOWER]
END
```

The following commands both run the demonstration:

```
DEMO.COLLIDE  40  40
DEMO.COLLIDE  20  (20 * SQRT 3)
```

You can put in the speed that the bullet is fired with as an input variable:

```
TO DEMO.COLLIDE  :HORIZ  :VERT  :BULLET.SPEED
      .
      .
  SETMOTION  "BULLET  (LIST  "TOSS  :BULLET.SPEED)
      .
      .
END
```

You must give the bullet a motion in this way, so that the variable *:bullet.speed* is assigned the value you give it. [TOSS :BULLET.SPEED] does not substitute in the value of *:bullet.speed*. In Logo the contents of a list are not evaluated. You will get an error message similar to " * doesn't like *:bullet.speed* as input in XMOTION".

Some implementations of Logo do not have the ATAN primitive. This means that the

corresponding heading must be included for the bullet to be fired on. The heading is dependent on the values chosen for the coordinates of the top of the tower. We can choose an angle of elevation which allows us easily to determine the coordinates of the target at the top of the tower.

If the angle of elevation of the top of the tower is 45 degrees (heading 45 degrees) the horizontal and vertical distances to the top of the tower are equal. If the angle of elevation is 30 degrees (heading 60 degrees) the horizontal and vertical distances are in the ratio of sqrt 3 : 1. If the angle of elevation is 60 degrees (heading 30 degrees) the horizontal and vertical distances are in the ratio of 1 : sqrt 3.

For example, we could choose:

* Heading 45 degrees and coordinates [50,50].

* Heading 45 degrees and coordinates [40,40].

* Heading 30 degrees and coordinates [20,(20 * sqrt 3)].

* Heading 60 degrees and coordinates [(20 * sqrt 3),20].

In this case DEMO.COLLIDE can have the bullet's heading as an input parameter:

```
TO DEMO.COLLIDE  :HORIZ  :VERT  :BULLET.ANGLE
  MAKETURTLE  "BULLET  0  0  :BULLET.ANGLE  :RED
   .
   .
END
```

Experimenting with different values of *:bullet.speed* and headings shows that the target and bullet always collide. Why is this?

If we have a situation where there is no acceleration due to gravity, then the target will not fall when released, and the bullet will move along the line of sight of the gun, directly

into the target. Gravity acts constantly and its effect is to cause both the bullet and target to accelerate downwards from the position they should have had. After a given amount of time, both the bullet and target will be the same distance vertically from where they should have been. When the bullet crosses the line of fall of the target, the target is a certain distance below where it is dropped from. If there is no gravity the bullet will reach the top of the tower, instead it will fall the same distance as the target from the top of the tower. Both fall the same distance vertically. Hence when the bullet crosses the line of fall of the target they collide!

Question 9 This problem is similar to ones of throwing at a given speed and angle. The water has both a vertical and a horizontal speed. The horizontal speed remains constant, and the vertical speed decreases under gravity. The water moves upwards in a curve, then falls in a curve.

```
TO DEMO.WATER
  MAKETURTLE  "WATER  0  0  45  :GREEN
  MAKETURTLE  "HOSE  0  0  90  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "WATER  [TOSS  5]
  SETMOTION  "HOSE  [ ]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  "EARTH
  INIT.TOSS  "WATER  "HOSE
  MOVE  [EARTH  WATER  HOSE]
END
```

If the hose moves in a direction opposite to that of the water, it moves on a heading of 225 degrees. In this case the water falls straight down. The vertical speed of the water is directly canceled by the vertical speed of the hose. The horizontal speed is canceled too. The water falls from rest out of the hose. As water is constantly coming out of the hose a falling curtain of water is seen, with a parabolic outline.

Question 10 The child will catch the ball, as both child and ball "absorb" the initial

motion of the bus.

If the bus accelerates while the ball is in the air, the child accelerates too. However, the ball does not experience this acceleration, and will fall behind the child.

If the bus goes round a curve while the ball is in the air, the bus and child experience an acceleration, and consequently their speed is changed. The ball does not experience this, and the ball does not return to the child.

Question 11

```
TO DEMO.BALL  :VIEWFROM
  MAKETURTLE  "BALL1  0  0  45  :RED
  MAKETURTLE  "BALL2  0  0  45  :GREEN
  MAKETURTLE  "PERSON  0  0  0  :BLUE
  SETMOTION  "BALL1  [TOSS  5]
  SETMOTION  "BALL2  [TOSS  10]
  SETMOTION  "PERSON  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.TOSS  "BALL1  "PERSON
  INIT.TOSS  "BALL2  "PERSON
  MOVE  [BALL1  BALL2  PERSON]
END
```

Both balls move in a curve under gravity. The slower one sees the faster one moving in a straight line. The faster one sees the slower one moving in a straight line.

The motion of one ball from the other is always a straight line, regardless of the angle of projection or the speed of projection. Let us verify this. Let us consider the motion vertically and horizontally and see what happens during short intervals of time. Horizontally both balls move constantly, so standing on one and viewing the other's horizontal motion, that ball is seen to move at a constant rate in the horizontal direction.

Vertically both balls are affected by gravity and will be slowed down by the same

amount in the same time interval. Therefore, when we view one ball from the other it appears to be moving at a constant rate in the vertical direction.

Standing on one ball and viewing the other, we see a constant vertical motion and a constant horizontal motion. The combined motion will be constant and therefore will be in a straight line.

Exercise 4.1.1

Question 1 Different inputs to CIRCLEA produce circles of different radii.

CIRCLEA 2 produces a circle of twice the radius of the circle drawn by CIRCLEA 1. The radius is *proportional* to the amount given to the FORWARD command.

Question 2 Different inputs to CIRCLEB produce circles of different radii. CIRCLEB takes as input the number of times it is necessary to repeat the RIGHT command to get a closed circle, and the value given to the RIGHT command. CIRCLEB 18 20 produces a circle of twice the radius of CIRCLEB 9 40.

The *larger* the angle the *smaller* the circle. Here the radius of the circle drawn is *inversely proportional* to the amount given to the RIGHT command.

Exercise 4.2.3

Question 3

```
CIRCLE1:  [FORWARD 6  RIGHT 10]
CIRCLE2:  [FORWARD 10  RIGHT 10]
```

CIRCLE2 is the larger of the two circles. It has a radius twice that of CIRCLE1.

Both circles are drawn in at the same rate, because both circles turn through the same amount on each step of the circle.

Draw out a diagram of CIRCLE1 and CIRCLE2 orbiting the ORIGIN. Stand on CIRCLE1. CIRCLE2 and the ORIGIN are always an equal distance away from CIRCLE1. They start from opposite points on the circumference of the orbit, as they are in opposite positions relative to CIRCLE1.

With the information that we have here, there is no way from the reference frame of CIRCLE1, to show that ORIGIN and CIRCLE2 do not have the same orbit.

When we view from CIRCLE2 the motion appears the same as that viewed from the ORIGIN. From the ORIGIN we see two circles, one twice the size of the other. From CIRCLE2 we see CIRCLE1 and the ORIGIN on the same heading, the ORIGIN twice the distance away from CIRCLE2 as CIRCLE1. Therefore, the motion produced is the same as seen from the ORIGIN.

Exercise 4.3.1

Question 2 All the planets show retrograde motion when viewed from the Earth.

The sun does not show retrograde motion. It is stationary at the center of the solar system. When viewed from the Earth, it appears to orbit the Earth on a circular path. You can use the method of time intervals to verify this.

Question 3 When retrograde motion occurs the Earth and a planet, such as Mercury, pass each other. At this particular position in their orbits they are the closest distance to each other. In astronomical terms this is called *opposition.* As the Earth passes a planet, the line of sight of the planet seen from the Earth changes direction.

Question 4 When the outer planets are viewed from the Earth, retrograde motion occurs when the Earth passes by the planet, as the Earth performs several revolutions while the planet performs one. For the inner planets of Venus and Mercury, retrograde motion occurs when these planets pass by the Earth on their orbit. Their period of revolution is smaller than that of the Earth's and they perform several revolutions about the Sun while the Earth performs one.

Exercise 4.4.1

Question 1

```
CIRCLE1:  [FORWARD 5  RIGHT 5]
CIRCLE2:  [FORWARD 10  RIGHT  20]
```

CIRCLE1 is the larger of the two circles. A *larger* value given to the FORWARD command produces a circle with a *larger* radius. The radius is *directly proportional* to the *forward* motion. A *larger* value given to the RIGHT command produces a circle with a *smaller* radius. The radius is *inversely proportional* to the *right* motion. We can combine these two to say:

The radius is *proportional* to (FORWARD *:amount*) / (RIGHT *:amount*)

The radius of CIRCLE1 is proportional to 5/5 or 1. The radius of CIRCLE2 is proportional to 10/20 or 1/2. Therefore CIRCLE1 is the larger of the two circles.

Question 2

```
CIRCLE1:  [FORWARD 10  RIGHT 10]
CIRCLE2:  [FORWARD 5  RIGHT 40]
```

The radius of CIRCLE1 is proportional to 10/10 or 1. The radius of CIRCLE2 is proportional to 5/40 or 1/8. CIRCLE1 is the larger circle.

Question 3

```
CIRCLE1:  [FORWARD 10  RIGHT 10]
CIRCLE2:  [FORWARD 5  RIGHT  5]
```

The radius of CIRCLE1 is proportional to 10/10 or 1. The radius of CIRCLE2 is proportional to 5/5 or 1. The circles have the same size.

Question 4

```
CIRCLE1:  [FORWARD 10  RIGHT 20]
CIRCLE2:  [FORWARD 5  RIGHT 5]
```

CIRCLE1 is being drawn in 4 times faster than CIRCLE2, because it has a turning angle of 4 times the magnitude of CIRCLE2's turning angle.

Epicycles are produced when the circles pass each other by. When CIRCLE1 has made one revolution CIRCLE2 has completed 1/4 revolution. By the time CIRCLE1 catches up with CIRCLE2, CIRCLE2 has moved round some more. As CIRCLE1 and CIRCLE2 continue their revolutions, CIRCLE1 passes by CIRCLE2 three times, while CIRCLE2 makes one revolution, so three epicycles are formed.

Question 5

```
CIRCLE1:  [FORWARD 15  RIGHT  20]
CIRCLE2:  [FORWARD 10  RIGHT 10]
```

CIRCLE1 is moving twice as fast as CIRCLE2. When CIRCLE1 has completed a revolution, CIRCLE2 has completed only 1/2 a revolution. By the time CIRCLE1 catches up with CIRCLE2, CIRCLE2 has moved some more. CIRCLE1 only passes CIRCLE2 once while CIRCLE2 make a revolution, so one epicycle loop is formed.

Exercise 4.4.3

Question 1

```
CIRCLE1:  [FORWARD 6  RIGHT 20]
CIRCLE2:  [FORWARD 6  RIGHT 6]
```

The motion of CIRCLE2 as seen from CIRCLE1 is a three cusp figure.

When the value given to CIRCLE1's forward motion is varied, the figure takes on different forms. The figure always has three waves, cusps or epicycles. The smaller amounts given to FORWARD produce waves and the larger amounts produce epicycles. The *:amount* given to the FORWARD command is proportional to the linear speed of the object. Epicycles or cusps are produced at the time CIRCLE1 passes CIRCLE2, or vice versa. They are closest at this point.

Question 2 Epicycles start to appear when the *:amount* given to CIRCLE1's FORWARD is greater than CIRCLE2's FORWARD *:amount.* In this example epicycles start to appear when CIRCLE1 has the motion of FORWARD 6. FORWARD 10 will produce epicycles with large loops. This means that epicycles appear when CIRCLE1's linear speed is greater than CIRCLE2's.

Question 3 Here we choose *:amount* = 10 and epicycles are produced with large loops. When we interchange the values given to the RIGHT commands so that now the motions are:

```
SETMOTION  "CIRCLE1  [FORWARD 10  RIGHT 6]
SETMOTION  "CIRCLE2  [FORWARD 6  RIGHT 20]
```

elongated waves are formed. CIRCLE1's linear speed is still greater than CIRCLE2's, but now CIRCLE1 has a smaller angular velocity, given by RIGHT 5 instead of RIGHT 20.

Question 4

```
CIRCLE1:  [FORWARD :amount  RIGHT 40]
CIRCLE2:  [FORWARD 4  RIGHT 4]
```

A figure with nine waves, cusps or epicycles is produced when the motion of CIRCLE2 is seen from CIRCLE1. Epicycles start to appear when *:amount* = 5. Again we see epicycles only when CIRCLE1 has a larger linear speed than CIRCLE2.

Question 5 Here we choose *:amount* = 7. A figure with nine epicycles is produced. When we interchange the value given to the RIGHT commands so that the motions are:

```
SETMOTION  "CIRCLE1  [FORWARD 7  RIGHT 4]
SETMOTION  "CIRCLE2  [FORWARD 4  RIGHT 40]
```

elongated waves are formed. CIRCLE1's angular velocity is now less than CIRCLE2's, even though its linear speed is greater.

Question 6 Questions 1 - 5 show that for epicycles to be produced CIRCLE1 must have a larger value for its FORWARD command and a larger value for its RIGHT command. This means that for epicycles to be produced, we have to have a *larger* value of FORWARD (larger linear speed) together with a *larger* value of RIGHT (larger angular velocity). A larger angular velocity means a smaller period of revolution. Circles:

```
CIRCLE1:  [FORWARD 15  RIGHT 30]
CIRCLE2:  [FORWARD 6  RIGHT 6]
```

will produce epicycles, whereas circles:

```
CIRCLE1:  [FORWARD 6  RIGHT 30]
CIRCLE2:  [FORWARD 10  RIGHT 10]
```

will not.

The answer to Question 9 discusses this phenomenon further.

Question 7 The circles made with:

```
CIRCLE1:  [FORWARD 5  RIGHT 30]
CIRCLE2:  [FORWARD 10  RIGHT 10]
```

should not produce an epicycle, as CIRCLE1 with the larger value for the RIGHT command does not have a larger value for the FORWARD command. However, when the motion is viewed from CIRCLE1 a loop is seen. Notice that this is *not* an epicycle. Imagine standing on the reference frame and watching CIRCLE2 being formed. As CIRCLE2's path is drawn in you never see CIRCLE2 reverse its direction of motion, it continues in the same direction. So from the reference frame you do not see a loop being formed.

Question 8 For epicycles to be produced we need a larger value of FORWARD *:amount* together with a larger value of RIGHT *:amount.* The speed is proportional to FORWARD *:amount.* The solar system produces epicycles when we stand on the Earth and view the other planets. To prove that our theory is correct we must show that the planets with the largest angular velocity (those that take the shortest time to make one revolution) have the highest linear speed.

To calculate the speed of a planet, we use the equation: $v = r * \omega$. ω is measured in radians per unit of time.

The Earth completes one revolution (2π radians) in one year. This means that ω has a value:

$\omega = 2\pi$ radians/year
$\omega = 2\pi / (365 * 24 * 60 * 60)$ radians/sec.

You can write a procedure PLANET.SPEED to return the value of the planet's speed. It takes as input the radius of the orbit (in Km. * 10^6) and the period of revolution (in years) as given in Table 2.

```
TO PLANET.SPEED  :RADIUS  :REVOLUTION
  OUTPUT ((:RADIUS * 1000000 * 2 * PI)/
          (:REVOLUTION * 365 * 24 * 60 * 60))
END
```

The speed of each planet in Km/sec is:

Mercury	48.16
Venus	34.71
Earth	29.89
Mars	24.16
Jupiter	13.07
Saturn	9.66
Uranus	6.80
Neptune	5.44
Pluto	4.73

This proves that each planet not only has a higher angular velocity, but also a higher linear speed. Epicycles are therefore produced when a planet is viewed from another one in the solar system.

Question 9

```
A:  [FORWARD 4  RIGHT 40]
B:  [FORWARD 10  RIGHT 20]
C:  [FORWARD 8  RIGHT 10]
```

Planet A has a higher angular velocity than planet C, as it has a larger value for its RIGHT command. However, planet A's linear speed is smaller than C's, as its amount for its FORWARD command is less than C's. Therefore, A will see C moving in a cusp pattern.

B has a higher angular velocity than C, as its value for its RIGHT command is larger than C's. B has a higher linear speed, as its amount for its FORWARD command is larger than C's. Therefore, B will see C moving in epicycles.

Epicycles or cusps appear when the objects pass each other. This is exactly the same phenomenon as the retrograde motion of the planets. In order to determine whether epicycles or cusps are produced when viewing one object from the other, we must consider the positions of the objects before and after they line up. If one object from the other always appears to be moving in the same direction then cusps or waves are made. The condition for epicycles to be produced is when the object appears to reverse its direction of motion.

On the step before line up C is to the right of A. C is 8 units away from the radius drawn on the diagram, on a heading of (270 - 10) = 260 degrees. A is 4 units away from the radius on a heading of (270 - 40) = 230 degrees. When the planets line up as shown they all have a heading of 270 degrees. On the step previous to this, C is a further 8 units away on a heading of 250 degrees. A is a further 4 units away on a heading of 190 degrees. On the step after line up, C is to the left of A, 8 units away from the line up position on a heading of 280 degrees. A is 4 units away from the line up position on a heading of 310 degrees. From A, C appears to move in the same direction, therefore the path does not epicycle.

We can again look at what happens on either side of the line up position in order to determine the path of C from B. On the step before line up C is 8 units away from the radius drawn in the diagram, on a heading of 260 degrees. B is 10 units away, on a heading of 250 degrees. B sees C in front of itself. On the step before this, C is a further 8 units away on a heading of 250 degrees and B a further 10 units away on a heading of 230 degrees. B sees C in front of itself. On the step after line up, C is 8 units in front of the radius, on a heading of 280 degrees and B 10 units in front on a heading of 290 degrees. B sees C behind itself. In this situation C is originally in front of B, then lined up and finally behind B. To B, C appears to reverse its direction of motion. Epicycles are therefore produced.

The motion seen depends on how the planets catch up with one another. This can be seen by looking at the position of each turtle for several steps around the passing position.

Question 10 When the reference frame is the SUN, you see the EARTH on a circular orbit about the SUN. The MOON moves in a circular orbit about the EARTH. From the reference frame of the SUN we do not see a circular path for the MOON. The MOON traces out a path consisting of a series of cusps. Notice that the MOON is always a fixed distance from the EARTH.

When the reference frame is the EARTH, the MOON orbits the EARTH and the SUN orbits the EARTH. This is what we expect to see from the EARTH!

When the MOON is the reference frame, the EARTH has a circular orbit about the

MOON, and the SUN moves on a cusp shaped orbit.

Question 11 In order to vary the *:amount* given to the MOON's FORWARD command you can edit the procedure DEMO.MOON to have this as an input parameter. The frame of reference is the SUN.

```
TO DEMO.MOON :F
  MAKETURTLE "EARTH (RADIUS 5 5) 0 180 :GREEN
  MAKETURTLE "MOON ((RADIUS 5 5) + (RADIUS :F 20))
                                   0 180 :RED
  MAKETURTLE "SUN 0 0 0 :YELLOW
  SETMOTION "EARTH [FORWARD 5 RIGHT 5]
  SETRELATIVEMOTION "MOON "EARTH [FORWARD :F RIGHT 20]
  SETMOTION "SUN [ ]
  MAKEFRAME "SUN
  MOVE [SUN EARTH MOON]
END
```

When *:F* is varied from 1 - 10, the path of the MOON as seen by the SUN varies. When *:F* = 1 elongated waves are formed, *:F* = 5 produces cusps, and *:F* = 6 epicycles start to appear.

The exact same phenomenon is seen here as was seen in Questions 1 - 5. Epicycles start to appear when the planet with the larger angular velocity (larger RIGHT *:amount*) has a larger value of linear speed (larger FORWARD *:amount*).

When we consider our Sun-Earth-Moon system, the Moon has a higher angular velocity than that of the Earth, as it performs 13 revolutions in one year, whereas the Earth performs one. The Earth's linear speed is given by:

$$\frac{150 * 1000000 * 2 * \pi}{365 * 24 * 60 * 60} = 29.89 \text{ Km/sec}$$

The Moon's linear speed is given by:

```
( 384400 * 2 * π) /          = 1.00 Km/sec
   ( 28 * 24 * 60 * 60 )
```

The Moon's linear speed is not higher than the Earth's, so epicycles will *not* be formed.

Question 12

```
TO DEMO.BIKE :VIEWFROM
  MAKETURTLE "WHEEL (RADIUS 2 10) 0 180 :GREEN
  MAKETURTLE "BIKE 0 0 90 :RED
  MAKETURTLE "EARTH 0 0 0 :YELLOW
  SETRELATIVEMOTION "WHEEL "BIKE [FORWARD 2 RIGHT 10]
  SETMOTION "BIKE [FORWARD 2]
  SETMOTION "EARTH [ ]
  MAKEFRAME :VIEWFROM
  MOVE [WHEEL BIKE EARTH]
END
```

The procedure DEMO.BIKE illustrates setting up a demonstration of a bicycle moving along. Note that the WHEEL is given a circular motion relative to the BIKE. As seen by the BIKE a point on the wheel moves in a circular path.

In addition, both the BIKE and WHEEL must have the same linear speed. An amount given to the FORWARD command is proportional to the linear speed. The linear speed for both is the same. In one revolution both the center of the wheel and a point on the rim will move through the same distance, therefore they must have the same speed. Try rolling a quarter to prove that the center and a point on the quarter's rim move through the same distance.

Exercise 4.5.2

Question 1 A is the center of the merry-go-round and is assumed stationary. B rotates around with the merry-go-round.

From the EARTH's frame of reference, when the ball is let go, it absorbs the speed of the person, and moves in a straight line.

From the previous section we see that the linear speed of an orbiting object is proportional to its amount given to the FORWARD command. In order for the ball to appear to be stationary this forward motion must be exactly balanced.

```
TO DEMO.PUSH  :VIEWFROM
  MAKETURTLE  "A  0  0  0  :GREEN
  MAKETURTLE  "B  (RADIUS  5 5)  0  180  :BLUE
  MAKETURTLE  "BALL  (RADIUS  5 5)  0  0  :RED
  MAKETURTLE  "EARTH  0  0  0  :YELLOW
  SETMOTION  "A  [ ]
  SETMOTION  "B  [FORWARD 5  RIGHT 5]
  SETMOTION  "BALL  [PUSH  5]
  SETMOTION  "EARTH  [ ]
  MAKEFRAME  :VIEWFROM
  INIT.PUSH  "BALL  "B
  MOVE  [A  B  BALL EARTH]
END
```

To balance the forward motion of the ball, which it absorbs from the person, the ball must be thrown with the same speed, in an opposite direction. The ball is set at a heading of 0 degrees, and is given a motion of [PUSH 5].

In order for the ball to pass through the center of the merry-go-round, B has to push the ball with a speed on a heading that exactly balances the forward motion (5 units) that the ball absorbs from the rotating person. This can be done with [PUSH 10] on a heading of

-60 degrees.

A now throws the ball. In this case the ball is thrown by a stationary person, and it does not absorb any motion. A must anticipate where B will move to during the time of flight of the ball.

When A throws to B, the direction of throw is not opposite to B throwing to A. This is because when B throws to A, the ball moves on a path which combines the throwing motion and the initial motion due to being on the rotating merry-go-round.

Question 2 From the reference frame of the EARTH, the ball moves in a straight line and the person moves round on the merry-go-round. The ball and person intersect, so the ball can be caught. In the reference frame of the person, the ball starts out at the same position as the person, loops around and comes back to the person.

When the person and ball are initially set up at $x = 0$, $y = (-\ RADIUS\ 5\ 5)$, the ball must be thrown on a heading of 45 degrees to be caught again by the person moving on the merry-go-round.

Question 3 When two balls are thrown simultaneously by a person on a merry-go-round, a stationary observer sees both balls move at a constant rate in straight lines. The motion of one ball from the other is a straight line. If both balls are thrown on the same heading, with the same speed then the second ball appears stationary to the first.

Appendix C

The Microworld Coding

C.1 The Apple II Logo Implementation

Documented here is the relative-motion microworld source code, written in Terrapin Logo for the Apple II 64K personal computer. This code provides a guide for implementing the system in other versions of Logo, such as LCSI Logo for the Macintosh.

Four basic commands form the building blocks of setting up objects on the computer screen and moving them around. The commands are: **MAKETURTLE, SETMOTION** or **SETRELATIVEMOTION, MAKEFRAME,** and **MOVE.**

* **MAKETURTLE.** This command makes an object. The procedure requires five inputs: the name of the object, its initial X position, its initial Y position, its heading, and a pen color to draw with on the screen.

```
TO MAKETURTLE :TURT :X :Y :H :COLOR
  PPROP "BASE :TURT [ ]
  PPROP "FRAME :TURT [ ]
  PPROP "COLOR :TURT :COLOR
  ZEROONE :TURT
  INIT.PAIR :TURT :X :Y
  RSETH :H
END
```

MAKETURTLE initializes the property lists for each object.

- **Base:** the named object is dependent on the object specified here.

- **Frame:** a list of objects that are dependent on this named object.

- **Color:** the color of the object's drawing pen.

The following procedures are called:

- **ZEROONE:** to initialize the object's **update** property.

- INIT.PAIR: to store the object's X and Y positions.

- RSETH: to store the object's heading.

* SETMOTION and SETRELATIVEMOTION. These commands are used to give an object a motion. The motion is to be repeated on every iteration of the program loop.

```
TO SETMOTION :TURT :PROC
  PPROP "MOTION :TURT :PROC
END

TO SETRELATIVEMOTION :TURT :BTURT :PROC
  PPROP "BASE :TURT :BTURT
  PPROP "MOTION :TURT :PROC
  DEPEND :TURT :BTURT
END
```

Both commands set up a **motion** property list for the named object. In addition, SETRELATIVEMOTION:

- Updates the object's **base** property list to contain **:BTURT.**

- Adds the named object, **:TURT**, to **:BTURT**'s list of dependencies, using the procedure **DEPEND.**

* MAKEFRAME. This procedure defines the frame of reference. The reference frame is stored in a global variable, **FRAME.** The procedure **REF.FRAME** is used to indicate the reference frame's position at the center of the computer screen.

```
TO MAKEFRAME :VIEWFROM
  MAKE "FRAME :VIEWFROM
  REF.FRAME
END
```

* MOVE. This procedure takes as input a list of the objects that are to be moved and shown on the computer screen.

```
TO MOVE :TURTLES
  MAKE "RESTTURTLES ALLBUT :FRAME :TURTLES
  SHIFT :RESTTURTLES
  INIT.PAIR :FRAME 0 0
  MOVELOOP
END
```

The procedure:

- Calls SHIFT to translate each object's position with respect to the reference frame object.

- Calls INIT.PAIR to store the reference frame's position at [0,0], the center of the screen.

- Starts the system by calling MOVELOOP.

The following routines are used by the four basic primitives of the system. These routines are transparent to the user.

* ZEROONE. Initializes an object's **update** property to zero. The **update** property accumulates the increment in X and the increment in Y for each iteration of the program loop.

```
TO ZEROONE :TURT
  PPROP "UPDATE :TURT [0 0]
END
```

* INIT.PAIR. This stores the X and Y positions of each named object.

```
TO INIT.PAIR :TURT :X :Y
  MAKE (WORD "X :TURT) :X
  MAKE (WORD "Y :TURT) :Y
END
```

* RSETH. This stores the heading of a named object. The value of :TURT is available via dynamic scoping.

```
TO RSETH :NEWHEAD
  PPROP "HEADING :TURT :NEWHEAD
END
```

* DEPEND. Updates :BTURT's list of dependencies by adding in the named object :TURT. All the objects contained in this list are moving *relative* to :BTURT.

```
TO DEPEND :TURT :BTURT
  PPROP "FRAME :BTURT (FPUT :TURT (GPROP "FRAME :BTURT))
END
```

* REF.FRAME. This procedure indicates the reference frame object as a cross at the center of the computer screen.

```
TO REF.FRAME
  PU HT HOME
  PENCOLOR GPROP "COLOR :FRAME
  PD FD 5
  PU HOME PD BK 5
  PU HOME PD RT 90 FD 5
  PU HOME PD LT 90 FD 5
  PU HOME
END
```

* SHIFT. Translates each object's position with respect to the reference frame object. It requires a list of the objects as input. SHIFT calls CORRECT.FOR.FRAME to perform the translation.

```
TO SHIFT :TURTLES
  IF :TURTLES = [ ] STOP
  CORRECT.FOR.FRAME FIRST :TURTLES
  SHIFT BF :TURTLES
END
```

* CORRECT.FOR.FRAME. Calculates an object's X and Y positions with respect to the frame of reference object.

```
TO CORRECT.FOR.FRAME :TURT
  MAKE (WORD "X :TURT)   ((THING (WORD "X :TURT))
                           - (THING (WORD "X :FRAME)))
  MAKE (WORD "Y :TURT)   ((THING (WORD "Y :TURT))
                           - (THING (WORD "Y :FRAME)))
END
```

* MOVELOOP. A recursive procedure, called by MOVE to start the system. Each iteration of MOVELOOP is equivalent to a time interval. The following procedures are called:

- ZEROUPDATE: to zero the update property of each object.

- CALCULATEMOVE: to run each object's motion and calculate its new position.

- MOVEREST: to move all the objects, except the reference frame object.

```
TO MOVELOOP
  PU
  ZEROUPDATE :TURTLES
  CALCULATEMOVE :TURTLES
  MOVEREST :RESTTURTLES
  MOVELOOP
END
```

* ZEROUPDATE. Calls ZEROONE for each object in turn to initialize its update property.

```
TO ZEROUPDATE :TURTLES
  IF :TURTLES = [ ] STOP
  ZEROONE FIRST :TURTLES
  ZEROUPDATE BF :TURTLES
END
```

* CALCULATEMOVE. This procedure calls CALCULATEONE for each object in turn.

```
TO CALCULATEMOVE :TURTLES
  IF :TURTLES = [ ] STOP
  CALCULATEONE FIRST :TURTLES
  CALCULATEMOVE BF :TURTLES
END
```

* CALCULATEONE. This procedure uses the turtle to compute an object's increment in a given time interval. The turtle is set at [0,0], given the object's heading, and the object's motion is run. Its new heading, increment in X

position, and increment in Y position is stored. In addition, the increment in X and Y is given to each of the objects moving relative to this object.

```
TO CALCULATEONE :TURT
  SETH RHEADING
  SETXY 0 0
  RUN RMOTION
  RSETH HEADING
  UPDATE.PAIR :TURT
  UPDATE GPROP "FRAME :TURT
END
```

* MOVEREST. Calls MOVEONE to move each object.

```
TO MOVEREST :TURTLES
  IF :TURTLES = [ ] STOP
  MOVEONE FIRST :TURTLES
  MOVEREST  BF :TURTLES
END
```

* MOVEONE. Moves an object from its original position to its new position, drawing as the object moves.

```
TO MOVEONE :TURT
  PU
  SETXY ( THING ( WORD "X :TURT))
                ( THING ( WORD "Y :TURT))
  PD
  PENCOLOR GPROP "COLOR :TURT
  UPDATE.POSITION
  SETXY ( THING ( WORD "X :TURT))
                (THING ( WORD "Y :TURT))
END
```

* UPDATE.POSITION. Calculates and stores an object's position, with respect

to the reference frame's new position, for this iteration of the program loop.

```
TO UPDATE.POSITION
  MAKE ( WORD :X :TURT)  (( THING ( WORD "X :TURT)) +
                         (FIRST GPROP "UPDATE :TURT) -
                         (FIRST GPROP "UPDATE :FRAME))
  MAKE ( WORD :Y :TURT)  (( THING ( WORD "Y :TURT)) +
                         (LAST GPROP "UPDATE :TURT) -
                         (LAST GPROP "UPDATE :FRAME))
END
```

* RHEADING. Outputs an object's heading.

```
TO RHEADING
  OUTPUT GPROP "HEADING :TURT
END
```

* RMOTION. Outputs an object's motion.

```
TO RMOTION
  OUTPUT GPROP "MOTION :TURT
END
```

* UPDATE.PAIR. Adds the turtle's X and Y positions onto the update property of an object.

```
TO UPDATE.PAIR :TURT
  PPROP "UPDATE :TURT (LIST (XCOR +
                             FIRST (GPROP "UPDATE :TURT))
                            (YCOR +
                             LAST  (GPROP "UPDATE :TURT)))
END
```

* UPDATE. Calls UPDATE.PAIR to increment dependent objects. Nested dependencies are also considered.

```
TO UPDATE :TURTLES
  IF :TURTLES = [ ] STOP
  UPDATE.PAIR FIRST :TURTLES
  UPDATE (GPROP "FRAME FIRST :TURTLES)
  UPDATE BF :TURTLES
END
```

* ALLBUT. Removes a member :X from a list :L.

```
TO ALLBUT :X :L
  IF :L = [ ] OUTPUT [ ]
  IF :X = FIRST :L OUTPUT BF :L
  OUTPUT FPUT FIRST :L ALLBUT :X BF :L
END
```

* PPROP. Puts :VALUE on :SYMBOL's property list, :PROPERTY.

```
TO PPROP :SYMBOL :PROPERTY :VALUE
  MAKE (WORD :SYMBOL :PROPERTY) :VALUE
END
```

* GPROP. Returns the value of SYMBOL's property, :PROPERTY.

```
TO GPROP :SYMBOL :PROPERTY
  OUTPUT THING (WORD :SYMBOL :PROPERTY)
END
```

* INIT.MOTION. Initializes the motion of object :TURT, with respect to the object :BTURT.

```
TO INIT.MOTION :TURT :BTURT
  MAKE (WORD "SPEEDX :TURT) ((XMOTION :BTURT) +
                             (XMOTION :TURT) +
                             (XMOTION (GPROP "BASE :BTURT)))
  MAKE (WORD "SPEEDY :TURT) ((YMOTION :BTURT) +
                             (YMOTION :TURT) +
                             (YMOTION (GPROP "BASE :BTURT)))
END
```

* XMOTION. Finds the horizontal component of object :TURT's motion.

```
TO XMOTION :TURT
  IF :TURT = [ ] OUTPUT 0
  OUTPUT ((EXTRACT.MOTION (GPROP "MOTION :TURT)) *
                          COS (90 - (GPROP "HEADING :TURT)))
END
```

* YMOTION. Finds the vertical component of object :TURT's motion.

```
TO YMOTION :TURT
  IF :TURT = [ ] OUTPUT 0
  OUTPUT ((EXTRACT.MOTION (GPROP "MOTION :TURT)) *
                          SIN (90 - (GPROP "HEADING :TURT)))
END
```

* EXTRACT.MOTION. Determines the value given to an object's motion.

```
TO EXTRACT.MOTION :L
  IF :L = [ ] THEN OUTPUT 0
  IF FIRST :L = "FORWARD THEN OUTPUT FIRST BF :L
  IF FIRST :L = "FD THEN OUTPUT FIRST BF :L
  IF FIRST :L = "BACK THEN OUTPUT FIRST BF :L
  IF FIRST :L = "BK THEN OUTPUT FIRST BF :L
  IF FIRST :L = "FALL THEN OUTPUT 0
  IF FIRST :L = "TOSS THEN OUTPUT FIRST BF :L
  IF FIRST :L = "PUSH THEN OUTPUT FIRST BF :L
  EXTRACT.MOTION BF :L
END
```

* SCALE. Calls SCALEONE for each object in turn.

```
TO SCALE :TURTLES
  IF :TURTLES = [ ] STOP
  SCALEONE FIRST :TURTLES
  SCALE BF :TURTLES
END
```

* SCALEONE. Scales a given object and returns its radius, forward and right motions in order to initialize the object as a planet in the solar system.

```
TO SCALEONE :TURT
  MAKE "SCALEANGLE (INTEGER (ROUND (3 *
                            (THING ( WORD (LAST :TURTLES)
                                          "PERIOD)) /
                            (THING ( WORD :TURT "PERIOD)))))
  MAKE "SCALERADIUS (INTEGER (ROUND (90 *
                             (THING ( WORD :TURT "RADIUS)) /
                             (THING ( WORD (LAST :TURTLES)
                                           "RADIUS)))))
  MAKE "SCALEFORWARD (INTEGER (ROUND
                              (2 * 3.14156 * :SCALEANGLE *
                                  :SCALERADIUS / 360)))
  IF :SCALEANGLE > 720 THEN
                   MAKE "SCALEANGLE (:SCALEANGLE - 540)
  IF :SCALEANGLE > 360 THEN
                   MAKE :SCALEANGLE (:SCALEANGLE - 360)
  PRINT (SE :TURT [RADIUS] :SCALERADIUS
                  [FORWARD] :SCALEFORWARD
                  [ANGLE] :SCALEANGLE)
END
```

* ORDER. Orders the list of planets, from inner planet to outer planet.

```
TO ORDER :TURTLES
  IF :TURTLES = [ ] OUTPUT [ ]
  MAKE "TEMP SMALLEST :TURTLES
  OUTPUT FPUT :TEMP (ORDER (ALLBUT :TEMP :TURTLES))
END
```

* SMALLEST. Determines the planet with the smallest radius of orbit.

```
TO SMALLEST :L
  IF :L = [ ] OUTPUT [ ]
  IF BF :L = [ ] OUTPUT FIRST :L
  IF ( THING ( WORD FIRST :L "RADIUS )) <
             ( THING ( WORD (FIRST BF :L) "RADIUS ))
               THEN OUTPUT (SMALLEST ALLBUT (FIRST BF :L) :L)
               ELSE OUTPUT (SMALLEST ALLBUT FIRST :L :L)
END
```

* SETUP. Sets up the screen so that the turtle does not wrap around. Initializes the pen colors.

```
TO SETUP
  NOWRAP
  MAKE "WHITE 6
  MAKE "GREEN 5
  MAKE "RED 4
  MAKE "BLUE 3
  MAKE "YELLOW 2
  HT CS
END
```

The following routines are special purpose primitives used under specific conditions by the user, such as falling under gravity.

* CLEAR. Used to set up the screen initially and in between demonstrations.

```
TO CLEAR
  ERASE NAMES
  HOME CS HT
  SETUP
END
```

* RADIUS. Returns the value of the radius of a circle made with FORWARD :S and RIGHT :A.

```
TO RADIUS :S :A
  OUTPUT (:S * 360 / (2 * 3.1415 * :A))
END
```

* INIT.TOSS. Initializes an object :TURT tossed from object :FROMTURT.

```
TO INIT.TOSS :TURT :FROMTURT
  INIT.MOTION :TURT :FROMTURT
END
```

* INIT.FALL. Initializes object :TURT falling from object :FROMTURT.

```
TO INIT.FALL :TURT :FROMTURT
  INIT.MOTION :TURT :FROMTURT
END
```

* INIT.PUSH. Initializes object :TURT pushed by :FROMTURT.

```
TO INIT.PUSH :TURT :FROMTURT
  INIT.MOTION  :TURT :FROMTURT
END
```

* TOSS. To toss an object with a given speed, INIT.SPEED. The procedure makes a correction in the vertical speed to compensate for the acceleration due to gravity.

```
TO TOSS :INIT.SPEED
  SETX (THING ( WORD "SPEEDX :TURT))
  SETY (THING ( WORD "SPEEDY :TURT))
  MAKE ( WORD "SPEEDY :TURT)
              ((THING ( WORD "SPEEDY :TURT)) - 1)
END
```

* FALL. To give an object the motion of falling.

```
TO FALL
  TOSS 0
END
```

* **PUSH.** To give an object a push of speed **:INIT.SPEED** across a horizontal surface.

```
TO PUSH :INIT.SPEED
  SETX (THING ( WORD "SPEEDX :TURT))
  SETY (THING ( WORD "SPEEDY :TURT))
END
```

* **SCALE.SOLAR.** Returns the value of the radius and the forward and right amounts that an object must be initialized with to simulate a planet in the solar system. The procedure takes a list of the planets to be set up as input.

```
TO SCALE.SOLAR :TURTLES
  MAKE "MERCURYRADIUS     58
  MAKE "VENUSRADIUS      108
  MAKE "EARTHRADIUS      150
  MAKE "MARSRADIUS       228
  MAKE "JUPITERRADIUS    778
  MAKE "SATURNRADIUS    1427
  MAKE "URANUSRADIUS    2869
  MAKE "NEPTUNERDAIUS   4498
  MAKE "PLUTORADIUS     5900
  MAKE "MERCURYPERIOD    0.24
  MAKE "VENUSPERIOD      0.62
  MAKE "EARTHPERIOD      1.00
  MAKE "MARSPERIOD       1.88
  MAKE "JUPITERPERIOD   11.86
  MAKE "SATURNPERIOD    29.46
  MAKE "URANUSPERIOD    84.1
  MAKE "NEPTUNEPERIOD 164.8
  MAKE "PLUTOPERIOD   248.4
  SCALE ORDER :TURTLES
  ERASE NAMES
END
```

C.2 The Apple II Assembler Code

Part of the microworld coding, dealing with moving each object, was written in Apple 6502 assembler language to speed up the system. This assembler code was written and implemented by Leigh Klotz from Terrapin, Inc. The addresses used in the assembler version refer to the ones specified in the Terrapin Logo Version 3.00. The assembler code is listed below. The assembler code is too large to be assembled using the assembler included with Terrapin Logo. The code was implemented on a DECSYSTEM-20, compiled and downloaded onto the Apple II 64K personal computer.

```
;-*-MIDAS-*-
.ENABL LC
.LIST ME

;A turtle is an object with the following state variables:
;[X Y DX DY HEADING MOTIONLIST DEPENDENTS COLOR BASE]
;X and Y are the turtle's current position on the screen.
;DX and DY are the changes accumulated during the processing
;of MOVEONE called on this turtle and also on MOVEONE called on
;turtles on which this turtle depends.
;DEPENDENTS is a list of turtles whose DX and DY should be updated
;by the change produced by running this turtle's MOTIONLIST starting
;at [0,0] on an initial heading, HEADING.
;COLOR is the desired color.
;BASE defines the turtle on which this turtle is dependent.

;TO MOVEONE :TURTLE
; PU
; SETXY :X :Y
; PD
; PC :COLOR
; MAKE "X :X + :DX - FRAMEDX
; MAKE "Y :Y + :DY - FRAMEDY
; SETXY :X :Y
;END
.SBTTL  Memory
START   =$99A0
;Hardware control addresses
GETRM1  =$C08B          ;Read twice to enable nodespace

;Addresses of primitives in Logo 3.00 64K
SSETXY  =$611E          ;SETXY
SPENU   =$6052          ;PU
SPENDN  =$6059          ;PD
SPENC   =$61CF          ;PC
SSUM    =$8905          ;+
SDIF    =$8937          ;-
;We don't really need thing--the value of an SATOM is the value cell.
```

```
;;;STHING        =$9CB7          ;THING


;Addresses of Logo variables
IHOME    =$D21C          ;SATOM object for HOME

;Addresses of other routines in Logo 3.00 64K.
ITEMI    =$9727
VPOP     =$5256
VPUSHP   =$5228
PUSH     =$5208
POPJ     =$40B7

;Page zero interface variables.
OTPUTN   =$07     ;Number of outputs from primitive
;;NARGS  =$05     ;Number of inputs to primitive
USERPZ   =$FC     ;Beginning of space available for us always
TEMPPZ   =$A2     ;Beginning of temp pz space
ARG1     =$A2     ;Logo ARG1
NARG1    =ARG1    ;for numeric references
ARG2     =$9E     ;Logo ARG2
NARG2    =ARG2    ;for numeric references

;Page zero safe through primitive invocations
TURTLE   =USERPZ                 ;Pointer to the turtle list
FRAME    =USERPZ+2               ;Pointer to the global frame turtle
;That's all we have for our own safe PZ variables.

;Page zero temporaries -- not preserved through primitives.
TMP      =TEMPPZ+2
TMP1     =TEMPPZ+4
TEMPX    =TEMPPZ+6
TEMPY    =TEMPPZ+7

;Addresses of primitive objects for passing extra variables around

;NEGATIVE FRAME DX variable (negative of real frame DX)
ISETX    =$D260
;NEGATIVE FRAME DY variable (negative of real frame DY)
ISETY    =$D268

.SBTTL   Macros
.MACRO   MOVI DEST,VAL
         LDA #VAL\
         STA DEST
         LDA #VAL↑
         STA DEST+1
.ENDM

.MACRO   MOV DEST,SRC
         LDA SRC
         STA DEST
```

```
        LDA SRC+1
        STA DEST+1
.ENDM

;Vpush element INDEX of the turtle list.
.MACRO  VPUSHI INDEX
        LDX #INDEX
        JSR VPUSHI
.ENDM

;VPOPs to the INDEXth element of TURTLE.
.MACRO  VPOPI INDEX
        LDX #INDEX
        JSR VPOPI
.ENDM

.MACRO  LDIXY ADDR
        LDX #ADDR\
        LDY #ADDR↑
.ENDM

;Push the address on the PDL
.MACRO  PUSHA ADDR
        LDIXY ADDR
        JSR PUSH
.ENDM

.MACRO  RPLACA LIST, NEWCAR
        LDA NEWCAR
        LDY #$00
        STA (LIST),Y
        INY
        LDA NEWCAR+1
        STA (LIST),Y
.ENDM

;Gets the value of the INDEXth item to DEST.
.MACRO  GTITEM DEST,INDEX
        LDY #DEST
        LDX #INDEX
        JSR GTITEM
.ENDM


.SBTTL  Turtle Index Variables
;Turtle state variable indices.  Call GETITM with this index in X
;to access the corresponding element of the list stored in TURTLE.
TRX=1
TRY=2
TRDX=3
TRDY=4
TRHEAD=5
```

```
TRMLST=6
TRDPND=7
TRCOLR=8

.SBTTL  Entry point for MOVEONE primitive
.=START

;Expect the turtle descriptor in NARG1 -- the calling sequence
;from Logo is
;.CALL :MOVEONE .ADDR :TURTLE.

MOVE1:  BIT GETRM1      ;Turn nodespace back on
        BIT GETRM1
        MOV TURTLE,NARG2        ;Get our argument -- a .ADDR.
;Get the FRAME turtle from Logo :HOME
        MOV TMP, IHOME  ;VPUSH THING "HOME -> VPUSH CAR "HOME
        JSR VPUTMP
        LDX #FRAME
        JSR VPOP
        JSR PSXYP       ;PU SETXY :x :y PD
        VPUSHI TRCOLR
        JSR JSPC        ;PC :color
;MAKE "X :X + :DX + FRAME(DX)
        LDX #TRX        ;VPUSHI TRX
        JSR VPUSHI
        LDX #TRDX
        JSR JSPLUX      ;vpush (:x + :dx)
        JSR VPUFDX      ;vpush FRAME's DX
        JSR JSDIF       ;subtract it to get :x + :dx - :framedx
        LDX #TRX        ;MAKE "x vpop
        JSR VPOPI
;MAKE "y :y +  :dy - frame(DY)
        LDX #TRY        ;VPUSHI TRY
        JSR VPUSHI
        LDX #TRDY       ;VPUSHI TRDY
        JSR JSPLUX      ;vpush (:y + :dy)
        JSR VPUFDY      ;vpush frame's dy
        JSR JSDIF       ;subtract it to get :y + :dy - :framedy
        LDX #TRY
        JSR VPOPI       ;MAKE "y vpop
;SETXY :x :y
        VPUSHI TRX
        VPUSHI TRY
        JMP SSETXY      ;And POPJ back to Logo
.SBTTL  Calls to primitives
;If the primitive takes multiple args you must set NARGS.
;If it outputs you must decrement OTPUTN afterwards.
JSPLUX: JSR VPUSHI      ;vpush :xreg
        ;...
JSPLUS: JSR PSHRDC      ;vpush vpop + vpop
        JMP SSUM
JSPC:   JSR PSHRDC
```

```
        JMP SPENC
JSDIF:  JSR PSHRDC
        JMP SDIF
;Routines which push pointers to routines popj'd to.
PSHRDC: PUSHA CLRRTS    ;Push exit point below
        RTS
        ;...
;Entry routines which are POPJ'd to. -- also just an RTS for above.
CLRRTS: LDA #$00
        STA OTPUTN
        RTS
.PAGE
;Pseudo-code routines
;Vpush the X'th element of the TURTLE list.
VPUSHI: JSR MVA2TR      ;MOV ARG2,TURTLE
;VPUSH Xth element of ARG2.
VPUSHA: JSR ITEMXA      ;MAKE "tmp <- ITEM xregister :arg2
VPUTMP: LDX #TMP
        JMP VPUSHP

;RPLACA (NTHCDR :N TURTLE) VPOP
;The :N index we get is an ITEM index.
;If :N=1 we do not do any CDRs.
VPOPI:  STX TEMPX
        LDX #TMP1
        JSR VPOP
        JSR MVA2TR      ;MOV ARG2,TURTLE
        LDX TEMPX
        CPX #$01
        BEQ VPOPIR      ;If it's the car we're setting skip the item.
        JSR ITEMXA      ;MAKE "arg1 ITEM :xregister :arg2
VPOPIR: RPLACA ARG2,TMP1
        RTS

.IFNE 0
;This is here in case you need it.  These days everything is done
;with stack operations.
;Sets (Y) to the X'th element of the TURTLE list.
GTITEM: STY TEMPY
        JSR ITEMX       ;MAKE "tmp item :xregister :turtle
        LDY TEMPY
        LDA TMP
        STA $00,Y
        LDA TMP+1
        STA $01,Y
        RTS
.ENDC

;The FRAME turtle's object lives in the Logo variable :HOME.
;It has as its DX and DY values that must be subtracted
;from the rest of the update information.
;VPUSHes the DX entry from the frame.
```

```
VPUFDX: MOV ARG2,FRAME
        LDX #TRDX
        JMP VPUSHA

;VPUSHes the DY entry from the frame.
VPUFDY: MOV ARG2,FRAME
        LDX #TRDY
        JMP VPUSHA
.SBTTL  Lower-level routines
;Interface to Logo's ITEM internal routine.  Gets ITEM X of TURTLE
;ITEMXA gets ITEM :xreg ;arg2
;MAKE "tmp ITEM :xregister :arg2
ITEMX:  JSR MVA2TR      ;MOV ARG2,TURTLE
ITEMXA: STX ARG1
        LDA #$00
        STA ARG1+1
        JSR ITEMI       ;MAKE "arg1 ITEM :arg1 :arg2
        MOV TMP,ARG1    ;arg2 is the list whose car was arg1, usually
        RTS

;PU
;SETXY :x :y (:x and :y from current turtle object)
;PD
;The logic in this is a little strange to save bytes.
;We PUSH a return pointer to an RTS
;   PUSH a return pointer to PENDOWN
;   PUSH a return pointer to SETXY
;   and jump to SPENU.
PSXYP:  VPUSHI TRX
        VPUSHI TRY
        JSR PSHRDC
        PUSHA SPENDN
        PUSHA SSETXY
        JMP SPENU

;MOV ARG2,TURTLE
MVA2TR: MOV ARG2,TURTLE
        RTS

ZZZZZZ  =.
.PRINT  ZZZZZZ          ;Final address plus 1
.IIF GT,ZZZZZZ-$9AA6,.ERROR ZZZZZZ      ;Too big
.END
```

C.3 The Assembler Implementation

Listed here are the modified routines to implement the assembler version of the microworld. An object is a list of state variables:

```
[X Y DX DY HEADING MOTIONLIST DEPENDENTS COLOR BASE]
```

X and Y are the object's current position on the screen. DX and DY are the object's accumulated increments for the current time interval. DEPENDENTS is a list of turtles whose DX and DY should be updated by the change produced by running this object's MOTIONLIST starting at [0,0], on a heading, HEADING. COLOR is the color of the object's drawing pen. BASE defines the object on which this object is dependent.

The major changes involve passing the object values. The procedure TVALS is used to build a list of the objects' values. SETITEM is used to set an item in an object's list. The procedure START loads a file, FRAME, which was created on the disk when the assembler code was downloaded to the Apple II.

The following routines have been modified to reflect the necessary changes to install the assembler version:

```
TO MAKETURTLE :TURT :X :Y :H :COLOR
  MAKE :TURT ( LIST :X :Y 0 0 [ ] [ ] :COLOR [ ] )
END

TO SETMOTION :TURT :PROC
  SETITEM 6 THING :TURT :PROC
END

TO SETRELATIVEMOTION :TURT :BTURT :PROC
  SETITEM 9 THING :TURT FPUT :BTURT ITEM 9 THING :TURT
  SETITEM 6 THING :TURT :PROC
  SETITEM 7 THING :BTURT FPUT :TURT ITEM 7 THING :BTURT
END

TO MAKEFRAME :VIEWFROM
  MAKE "FRAME :VIEWFROM
  MAKE "HOME THING :FRAME
  REF.FRAME
END
```

```
TO MOVE :TURTLES
  MAKE "RESTTURTLES ALLBUT :FRAME :TURTLES
  MAKE "RESTTURTLES.VALS TVALS :RESTTURTLES
  MAKE "TURTLES.VALS TVALS :TURTLES
  MAKE "TURTLES :TURTLES
  SHIFT :RESTTURTLES.VALS
  SETITEM 1 :HOME 0
  SETITEM 2 :HOME 0
  MOVELOOP
END

TO SHIFT :TVALS
  IF :TVALS = [ ] STOP
  CORRECT.FOR.FRAME FIRST :TVALS
  SHIFT BF :TVALS
END

TO CORRECT.FOR.FRAME :TURT
  SETITEM 1 :TURT (ITEM 1 :TURT) - ITEM 1 :HOME
  SETITEM 2 :TURT (ITEM 2 :TURT) - ITEM 2 :HOME
END

TO MOVELOOP
  PU
  ZEROUPDATE :TURTLES.VALS
  CALCULATEMOVE :TURTLES.VALS :TURTLES
  MOVEREST :RESTTURTLES.VALS
  MOVELOOP
END

TO ZEROUPDATE :TURTLES
  IF :TURTLES = [ ] STOP
  SETITEM 3 FIRST :TURTLES 0
  SETITEM 4 FIRST :TURTLES 0
  ZEROUPDATE BF :TURTLES
END

TO CALCULATEMOVE :TURTLES.VALS :TURTLES
  IF :TURTLES.VALS = [ ] STOP
  LOCAL "TURTV MAKE "TURTV FIRST :TURTLES.VALS
  LOCAL "TURT MAKE 'TURT FIRST :TURTLES
  SETH ITEM 5 :TURTV
  SETXY 0 0
  RUN RMOTION :TURTV
  SETITEM 5 :TURTV HEADING
  UPDATE.PAIR :TURTV
  UPDATE ITEM 7 :TURTV
  CALCULATEMOVE BF :TURTLES.VALS BF :TURTLES
END

TO UPDATE :TURTLES
  IF :TURTLES = [ ] STOP
```

```
  UPDATE.PAIR THING FIRST :TURTLES
  UPDATE ITEM 7 THING FIRST :TURTLES
  UPDATE BF ;TURTLES
END

TO UPDATE.PAIR :TURT
  SETITEM 3 :TURT XCOR + ITEM 3 :TURT
  SETITEM 4 :TURT YCOR + ITEM 4 :TURT
END

TO MOVEREST :TURTLES
  IF :TURTLES = [ ] STOP
  .CALL 39328 .ADDR FIRST :TURTLES
  MOVEREST BF :TURTLES
END

TO RMOTION :TURT
  OUTPUT ITEM 6 :TURT
END

TO TVALS :TURTLES
  IF EMPTY? :TURTLES OP [ ]
  OP FPUT THING FIRST :TURTLES TVALS BF :TURTLES
END

TO START
  DOS [BLOAD FRAME]
  SET.CL
END

TO XMOTION :TURT
  IF :TURT = [ ] OUTPUT 0
  OUTPUT ((EXTRACT.MOTION (ITEM 6 THING :TURT)) *
          COS (90 - (ITEM 5 THING :TURT)))
END

TO YMOTION :TURT
  IF :TURT = [ ] OUTPUT 0
  OUTPUT ((EXTRACT.MOTION (ITEM 6 THING :TURT)) *
          SIN (90 - (ITEM 5 THING :TURT)))
END

TO INIT.MOTION :TURT :BTURT
  MAKE ( WORD "SPEEDX :TURT)  (XMOTION :BTURT +
                               XMOTION :TURT +
                               XMOTION ITEM 9 THING :BTURT)
  MAKE ( WORD "SPEEDY :TURT)  (YMOTION :BTURT +
                               YMOTION :TURT +
                               YMOTION ITEM 9 THING :BTURT)
END

TO SETITEM :ITEM :LIST :THING
```

```
  IF :ITEM > 1 SETITEM :ITEM - 1 BF :LIST :THING STOP
  MAKE "LIST .ADDR :LIST
  MAKE "THING .ADDR :THING
  LOCAL "HIGH MAKE "HIGH QUOTIENT :THING 256
  .DEPOSIT :LIST REMAINDER :THING 256
  .DEPOSIT :LIST + 1 :HIGH
END

TO CLEAR
  ERASE NAMES
  HOME CS HT
  SET.CL
END

TO SET.CL
  NOWRAP
  MAKE "WHITE 6
  MAKE "GREEN 5
  MAKE "RED 4
  MAKE "BLUE 3
  MAKE "YELLOW 2
END
```

To implement the system the following procedures must be included. These are the same as listed in section C.1.

```
ALLBUT              INIT.TOSS          SCALE
EXTRACT.MOTION      ORDER              SCALEONE
FALL                PUSH               SCALE.SOLAR
INIT.FALL           RADIUS             SMALLEST
INIT.PUSH           REF.FRAME          TOSS
```

References

[Abelson 80]
Abelson, H. and diSessa, A.A.
Turtle geometry: The Computer as a Medium for Exploring Mathematics.
MIT Press, 1980.

[Abelson 85]
Abelson, H. and Sussman, G.J.
Structure and Interpretation of Computer Programs.
MIT Press, 1985.

[Alonso 67]
Alonso, M. and Finn, E.J.
Fundamental University Physics, Volume 1.
Addison-Wesley, 1967.

[Clement 82]
Clement, J.
Students' Preconceptions in Introductory Mechanics.
American Journal of Physics 50(1), January, 1982.

[diSessa 75]
diSessa, A.A.
ORBIT: A Mini-Environment for Exploring Orbital Mechanics.
Computers and Education :359, 1975.

[diSessa 80]
diSessa, A.A.
Computation as a Physical and Intellectual Environment for Learning Physics.
Computers and Education 4:67-75, 1980.

[diSessa 83]
diSessa, A.A.
Phenomenology and the Evolution of Intuition.
In Gentner, D. and Stevens, A., editor, *Mental Models.* Erlbaum Press, 1983.

[diSessa 85]
diSessa, A.A.
Learning about Knowing.
In E.L.Klein, editor, *Computers and Children.* San Fransisco: Jossey-Bass, 1985.

[Feynman 63]
Feynman, R.P., Leighton, R.B. and Sands, M.L.
The Feynman Lectures on Physics, Volume 1.
Addison-Wesley, 1963.

[Gilbert 82]
Gilbert, J.K., Watts, D.M. and Osborne, R.J.
Students' Conceptions of Ideas in Mechanics.
Physics Education 17(2):62, 1982.

[Groen 84]
Groen, Guy.
Theories of Logo.
In *Pre-Proceedings of the 1984 National Logo Conference*, pages 49. McGill University, June, 1984.

[Lawler 82]
Lawler, R.W.
Designing Computer-Based Microworlds.
Byte :138, August, 1982.

[McCloskey 83]
McCloskey, M.
Intuitive Physics.
Scientific American :122, April, 1983.

[McDonald 52]
McDonald, J.
The Coriolis Effect.
Scientific American :72, May, 1952.

[Papert 80]
Papert, S.
Mindstorms: Children, Computers, and Powerful Ideas.
Basic Books, 1980.

[Resnick 77]
Resnick, R. and Halliday, D.
Physics, Part I.
John Wiley and Sons, 1977.

[Shanon 76]
Shanon, B.
Aristotelianism, Newtonianism and the Physics of the Layman.
Perception 5:241-243, 1976.

[Tobias 84]
Tobias, Joyce.
From the Classroom to the Laboratory.
In *Pre-Proceedings of the 1984 National Logo Conference*, pages 93. Public Schools of Brookline, Mass., June, 1984.

[Zeilik 79]
Zeilik, M.
Astronomy: The Evolving Universe.
Harper and Row, 1979.

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 4/22/96

Report # LCS-TR-347

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 243 (251-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form (2) ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP! (1-243)	1-243
(244-251) SCANCONTROL, COVER, SPINE, DOD(2), TRGT'S(5)	

Scanning Agent Signoff:
Date Received: 4/22/96 Date Scanned: 4/22/96 Date Returned: 4/25/96

Scanning Agent Signature: Michael W. Cook

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MIT/LCS/TR-347	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A Relative-Motion Microworld		5. TYPE OF REPORT & PERIOD COVERED Masters Thesis 1984-85 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Linda E. Morecroft		8. CONTRACT OR GRANT NUMBER*(s)* DARPA/DOD N0001483-K0I25
9. PERFORMING ORGANIZATION NAME AND ADDRESS MIT Laboratory for Computer Science 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA/DOD 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE September 1985 13. NUMBER OF PAGES 243
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* ONR/Department of the Navy Information Systems Program Arlington, VA 22217		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release, distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Computer-based learning, Logo, Microworld, Physics, Reference Frames, Relative Motion

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A relative-motion microworld has been designed to aid high-school students in understanding the concepts of relative motion and frames of reference.

Relative motion and frames of reference are usually introduced in a high-school physics or mathematics course. Most students, and many teachers too, have difficulty understanding the concepts and applying them to solve problems. The traditional approach to relative motion uses vector algebra.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

However, vector terminology is complex and it does not allow a mental picture of what is happening to be easily built up. Students do not understand what it means to be in a different frame of reference and how moving objects objects appear within this reference frame. Most people have a much more intuitive approach to motion problems.

The relative-motion microworld uses a new representation for thinking about motion, based on the intrinsic, local and procedural characteristics of turtle geometry. This representation provides the student with a simple framework for understanding motion and relative motion, that overcomes the problems associated with the traditional approach. The microworld allows students to set up their own motion problems, in order to view the motion from different reference frames. The new representation, together with the dynamic simulation, provide a very powerful tool for thinking about motion and understanding the motion seen from a chosen reference frame.

Four key primitives have been designed: these make an object, give an object a motion, select a reference frame, and move all the objects that have been set up. These four primitives are the building blocks that can be used to set up complex problems. For example, how does a stationary observer see a penny fall to the ground when dropped by a running person? How does the running person see the penny fall to the ground? In a few lines of code students implement this problem and use the simulation and new representation to analyze the motion.

Problems from a variety of motion situations can be explored using the relative-motion microworld. These include rectilinear motion, non-parallel rectilinear motion, motion under gravity, and rotational motion.

A student text and teacher's reference manual accompany the microworld. The text leads the student through a discussion of relative motion and frames of reference. It introduces the new representation for motion and describes how the microworld models relative motion. The text outlines various areas of investigation: parallel and non-parallel rectilinear motion, motion under gravity and circular motion. Each section includes examples and exercises. The exercises range in complexity and include ones to implement with the microworld and ones to think through. The teacher's reference manual gives hints on solving the exercises and solutions to the exercises.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned
Date: 04/22/1996
M.I.T. Libraries
Document Services