MASSACHUSETTS INSTITUTE OF TECHNOLOGY

A.I. LABORATORY

Artificial Intelligence
Memo No. 432

April 1977
LOGO Memo No. 46

Teacher's Guide for
Computational Models of Animal Behavior

Hal Abelson and Paul Goldenberg

A Computer-Based Curriculum Unit

To Accompany the Elementary Science Study Guide

Behavior of Mealworms

This is an experimental curriculum unit which suggests how the computational perspective can be intergrated into a subject such as elementary school biology. In order to illustrate the interplay of computer and non-computer activities, we have prepared the unit as a companion to the Elementary School Science Study "Teacher's Guide to Behavior of Mealworms." This material is based on use of the Logo computer language.

The work reported in this paper was supported by the National Science Foundation under grant number EC40708X and conducted at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Preface

The computer presence in the classroom opens the way for dramatic innovations in instructional content. The educational potential of a technology that the individual student can learn to manipulate, to extend and to apply to projects goes beyond merely providing teaching machines and computer-managed instruction. Much more than the machines themselves, it is the *computational perspective* which can enrich and transform traditional curricula, act as a focus for integrating insights from diverse disciplines and enable learning to become more active and project-oriented.

The following experimental curriculum unit suggests how this might happen in a subject such as elementary school biology. In order to better illustrate the interplay of computer and non-computer activities we have prepared the unit as a companion to the *Elementary Science Study* "Teacher's Guide for Behavior of Mealworms."[1] The Mealworm Guide represents, in our opinion, an excellent introduction to the observation of animal behavior. It leads students to devise and carry out experiments centered around such questions as "Can mealworms see?" "How do mealworms follow walls?" and "How do mealworms find food?" The value of this material lies not so much in the knowledge gained about mealworms as in the opportunity it presents for students to conduct meaningful scientific investigations at the elementary school level.

To these activities the computer adds the dimension of *investigating a phenomenon through making a mathematical model of it*. So, for example, the student who hypothesizes a mechanism that a mealworm might use to follow along a wall can embody this mechanism as a computer program for a robot turtle and see if it actually works. The student who is investigating whether mealworms use smell in locating food can endow the turtle with a simulated sense of smell and compare the turtle's resulting motion with that of the mealworm. Conversely, designing such a simulation requires choosing among alternative representations for smell information and alternative ways of using this information to locate the food. This, in turn, suggests further experiments with the mealworm. Such use of *computation as a descriptive language* illustrates how observing a real animal (the mealworm) and programming simulated animals (such as the robot turtle) can be complementary and mutually enhancing activities.

We feel compelled here to emphasize the distinction between this use of computation and the pre-programmed biological simulation games of the "insert parameter -- see result" type. For the major educational value of the activities described below is that students are given the opportunity to design *their own* models, to decide for themselves which features to include in a simulation and to practice formulating their hypotheses and observations in the mathematical language of computation. This flexibility also pays off in the wealth of insights that grow from these simple investigations. For example, a project which focusses on making the turtle simulate a mealworm's random path can easily serve as a springboard for an introduction to the theory of random walk; and attempts to have the turtle navigate

by means of sensory information can develop into graphic illustrations of the power of feedback control.

Students who are to explore in this way must be provided not only with a computer but also with a rich computational environment. The material below is based upon using Logo, a computer language which is simple enough for elementary students to use in self-directed ways and yet powerful enough to avoid casting all interactions with the computer into a rigid numerical processing mold. The "turtles" discussed in this guide are of two species -- floor-turtle: a robot equipped with a primitive sense of touch; and TV-turtle: a simulated creature on a computer-generated graphics display. We envision this material as a *second* exposure to Logo, and assume that students will have met both Logo and turtles in a previous unit on turtle geometry.[2] So we do not discuss here issues of introducing students to the basics of writing procedures and controlling turtles. We have concentrated instead on illustrating how these facilities, once accessible, can be integrated into the classroom environment.

Introduction

The study of animal behavior offers so many fascinating avenues for investigation and so many side trips along the way that it is difficult to restrict oneself to any area small enough to report on responsibly. Early in the planning of this guide, we had thought of including a much wider range of animal behaviors, and thus the story about Bal Boa is truly an appendix to this guide in the sense that though vitally connected, it is only the vestige of a much larger previous organ. That irresistable temptation to share something interesting and informative might serve as a pedagogical guiding principle for the use of this unit. There really *is* a lot to study and the side trips are not obviously less valuable than the highway they branch off of. Thus we have suggested areas of inquiry which seem to us particularly rich in opportunities to get side-tracked, raising questions about the mechanisms of a behavior, the usefulness of particular programs for the behavior, bugs in the behavior, and the effects of certain impairments on the behavior.

These studies are designed to complement the work the students are doing in the **Behavior of Mealworms** unit and are intended as an accompaniment rather than as an independent and separate study. Five major topics are included. The first two, **Observing Turtle**, and **Random Motion** are probably the most appropriate introductions to the turtle, but otherwise there is no essential sequence to the activities described in the guide. Within each section, we have included suggestions for initial experiments, questions that may lead to further experimentation and ideas about possible strategies for conducting investigations. We have shown examples of some of the kinds of procedures that students might write and have provided notes on each section which contain, among other things, the text of all of the more complex service procedures that neither you nor your students need bother about.

The first section, **Observing Turtle**, asks the student to use some of the thinking styles of the ethologist and naturalist. The questions do not deal so much with mechanisms as with patterns of behavior. The student's efforts are directed toward the clarification of certain issues such as what features to consider relevant when comparing Turtle with Mealworm or the meaning of "smarter than" in the same context.

The second section, **Random Motion**, explores ways in which randomness may appear in behavior and ways in which its effects may be biased. The skills of examining a behavior in a variety of environments (section 1) and introducing unpredictability into a behavior model (the current section) are basic to all of the remaining projects.

Section 3 **Orientation by Touch**, suggests experiments which may show the surprisingly great power of a seemingly very limited sense. Touch does not give information at a distance (as do smell and sight) nor does the turtle's sense of touch even give any definition to the object touched (intensity, shape, size, etc.) as might another sense. Yet it seems quite capable of modeling a number of fairly complex behaviors.

The fourth and fifth sections deal with modelling smell and sight respectively. For these experiments, rather primitive versions of the senses are suggested. For example, the smell procedure provided in the notes for section four gives information about a smell using the assumption that the farther one is from the stimulus, the weaker the smell will be. It indicates neither the direction from which the smell is coming nor the nature of the smell. It does assume that the creature can smell the object no matter how distant it is. The question *When, if ever, is the turtle getting information from its environment?* in sections 1-3, may stimulate discussions about what limitations the turtle lives with.

The sight model provides a two-eyed creature information about the direction from which the stimulus comes by giving different intensities of a light source to each eye. It does not afford any other information about the stimulus such as size, shape, or color. It might be interesting for some students to explore more complex variations of these senses.

Notes for Preface and Introduction

1. *Teacher's Guide for Behavior of Mealworms*, Elementary Science Study, Educational Services Incorporated, McGraw-Hill, 1966.

2. See, for example, Seymour Papert, "Teaching Children to Be Mathematicians vs. Teaching about Mathematics," *International Journal of Mathematics Education*, vol. 3, 1972. Also available as Memo No. 249, Artificial Intelligence Laboratory, MIT.

1: Observing Turtle

As with the study of mealworms, it is important to allow time for students to observe the turtle in an informal way. The similarities between mealworm and robot-turtle are as important to an understanding of each of them as are their differences and so comparisons of their behaviors will probably be easier to make if, from the start, both the turtle and the mealworm are thought of and referred to as *creatures*.

A First Activity

Prepare a few simple turtle behaviors for the class to observe.[1] To help insure that the observations are not biased by *a priori* anticipated behaviors, it is best if the procedures not have mnemonic names. The students should be encouraged to try several of the procedures without looking at the code and observe closely. If the students are not initially aware that the turtle has touch sensors, their first experiences with turtle behaviors should afford them that information in a relatively clear way. Compare, for example, **BEH1**, **BEH4** and **BEH7** in the notes at the end of this section. **BEH1** responds to an obstacle in front of it. Indeed, the response depends to some extent on the nature of the obstacle and the direction of impact -- the turtle may or may not continue to strain. **BEH4** does *nothing* until it is deliberately stimulated, a situation which is less likely to occur by accident than **BEH1**'s crashing into a wall. **BEH7** does not use the touch sensors at all, and therefore gives no information about their existence. **BEH3** may be for some students a clearer demonstration of the turtle's sense of touch than **BEH1**, but the latter's virtue as a first experience lies in the very fact that it does afford a somewhat greater variety of conjectures about why the turtle stops at an obstacle. Close observation and several trials will indicate that the turtle can know when it touches something.

In addition to the floor-turtle activities, you may wish to have some students observing the behavior of a TV-turtle. Some behaviors for the TV-turtle are suggested in the notes.[2] The TV-turtle's world is quite different from the floor-turtle's world and presents some special difficulties for observation. Unlike the floor-turtle, the TV-turtle's anatomy does not help the student find conjectures to test. A child cannot physically enter the TV-turtle's world and cannot so easily differentiate between the TV-turtle and the procedure that it follows.

Children may be encouraged, as with the mealworms, to keep a chart with the headings "What I Did" and "What The Turtle Did" and they may be helped to identify and keep a record of their conjectures as they go along. Sometimes exploratory curiosity seems intractible, but it is important to remember that even the most random-looking experiments are usually based on some notions about the beast being studied. (Putting a drop of vinegar on or near the tail of a mealworm is a more likely experiment than doing the same with an elephant or with a robot-turtle. There is a built-in implicit theory about what can be sensed by mealworms, elephants and robots.) Becoming consciously aware that his

curiosities are not always as "idle" as they may seem can enhance the student's self-image as a scientist. In addition, it will provide a record that can both afford insights about the history of science and lead to new and more specifically directed experiments.

The usual limits of students' theories about the turtle constrain them to experiments which are actually safe. In addition to petting it and putting obstacles in front of it, you might expect to see experiments like unplugging the turtle, picking it up to see what it does, pushing it a distance, setting it on its back. In unusual circumstances, a student might try something less safe.

Each student might be encouraged to make a variety of observations on the turtle, including observations (for both video- and land-tortoises) of each of Turtles' behaviors, Turtle anatomy, Turtle environments. An alternative, and generally easier plan would involve students specializing in a particular feature or group of phenomena. All students might observe the same behaviors, but pay greater attention to their own area. In particular, observations of the TV-turtle might be kept separately because of the greater abstraction involved.

Suggestions for Discussion

One or more class sessions should be taken to allow students to share their observations of Turtle and Mealworm. The Mealworm Guide makes a number of useful suggestions for conducting the discussion. In addition, these questions may be considered.

Do you think Turtle can see?

How are mealworms different from Turtle?

Is Turtle's behavior as predictable as a mealworm's?

Which creature is smarter: Turtle or Mealworm?

Which creature is smarter: Floor-turtle or TV-turtle?

In comparing the smartness of Turtle and Mealworm, the class may need to decide *which* of Turtle's behaviors (programs) will compete with Mealworm's only behavior. Or the class may decide that Turtle's ability to accept any program from us (or inability to do anything but sit *unless* it receives a program from us) should be considered an important part of the answer. In any case, comparisons of the two creatures becomes somewhat easier when it is explicitly acknowledged that the turtle's program is not part of the turtle in quite the way that the mealworm's program is part of the mealworm. For one, the program can be changed by us; for the other, it is built-in characteristic of the creature being studied. Thus, observations on "the" behavior of the turtle must take some account of *which*

program the turtle was using.

Notes for Section 1

1. Here are several floor-turtle procedures you may wish to have ready.

```
TO BEH1
10 FORWARD 50 UNTIL FTOUCH
END

TO BEH2
10 FORWARD 50 UNTIL FTOUCH
20 TOOT 1
END

TO BEH3
10 FORWARD 30 UNTIL FTOUCH
20 BACK 10
30 LEFT 180
40 BEH3
END

TO BEH4
10 IF FTOUCH BACK 50 B4SUB1
20 IF BTOUCH FORWARD 50 B4SUB1
30 IF RTOUCH RIGHT 90 BACK 50 B4SUB1
40 IF LTOUCH LEFT 90 BACK 50 B4SUB1
50 BEH4
END

TO B4SUB1
10 IF RANDOM < 3 TOOT 1
END

TO BEH5
10 IF FTOUCH LAMPON
20 IF RTOUCH TOOT 1
30 IF LTOUCH TOOT 20
40 IF BTOUCH LAMPOFF
50 BEH5
END
```

```
TO BEH6
10 B6SUB1 UNTIL B6SUB2
20 WAIT 300
30 BEH6
END
```

```
TO B6SUB1
10 FORWARD 10 RIGHT 10
20 BACK 10 LEFT 10
30 BACK 10 RIGHT 10
40 FORWARD 10 LEFT 10
END

TO B6SUB2
10 IF EITHER FTOUCH BTOUCH OUTPUT "TRUE
20 IF EITHER RTOUCH LTOUCH OUTPUT "TRUE
30 OUTPUT "FALSE
END
```

```
TO BEH7
10 B6SUB1
20 BEH7
END
```

2. Here are some TV-turtle procedures you may wish to have ready:

```
TO TV1
10 FORWARD 20
20 IF RANDOM < 3 RIGHT 90
30 TV1
END
```

```
TO TV2
10 PENDOWN
20 TV2SUB1
30 PENUP
40 TV2SUB1
50 TV2
END

TO TV3
10 TV2SUB1 UNTIL (RANDOM < 3)
20 HOME
30 TV3
END
```

```
TO TV2SUB1
10 FORWARD 10 * RANDOM
20 RIGHT 45 * RANDOM
END
```

2: Random Motion

The Mealworm Guide (Sections 3, 4 and 7) suggests a number of projects observing the paths of mealworms moving in a box. These provide a natural context for writing simple turtle programs to *model* the mealworms' observed behavior. With TV-turtle, students can test out different procedures and then get paper copies of the screen to compare with real mealworm tracks.

Most likely, you will see considerable variety in the features different students select to model. Some students will focus on the *irregularity* of the mealworm's movements. Others may try to duplicate certain regularities that they have noticed, and may even devise sophisticated measures of the shape of the path, the frequency of turning, and so on. Class discussions should respect these different approaches.

What is important is that the students *themselves* design the experimental models, and not merely supply parameters to a pre-programmed simulation. Because a variety of different programs may be found which "make the turtle draw tracks like the mealworms do," the students may have a unique opportunity to discover the sometimes surprising truth that models with basically different underlying theories can sometimes produce indistinguishable results.

Bear in mind also that a modeling activity like this tends to be very open-ended. Often, the insights to be gained from working on a particular project are not built in from the start, but develop as students modify and elaborate their initial approaches. This is illustrated in the two sample projects discussed below.

Sample Project 1: Paths

This project, like so many valuable programming activities, grows initially out of a *bug*, the unexpected "wrong" outcome of a program. In this case the bug arises in deciding that, since the mealworm moves "at random" its behavior can be approximated by

```
TO WORM1
10 FORWARD RANDOM
20 RIGHT RANDOM
30 WORM1
END
```

But see what happens! Instead of randomly meandering, the turtle travels mostly in circles. Do you see why? The bug is that the turtle only turns toward the *right*. This can be remedied by including some left turns:

```
TO WORM2
10 FORWARD RANDOM
20 RIGHT RANDOM
25 LEFT RANDOM
30 WORM2
END
```

Comparing the procedures **WORM1** and **WORM2** leads to the idea of generating behaviors that are intermediate between the two. You can do this by supplying a biasing factor for one of the turns:

```
TO WORM3 :BIAS
10 FORWARD RANDOM
20 RIGHT RANDOM
25 LEFT RANDOM * :BIAS
30 WORM3 :BIAS
END
```

Or one can bias both of the turns:

```
TO WORM4 :LEFTBIAS :RIGHTBIAS
10 FORWARD RANDOM
20 RIGHT RANDOM * :RIGHTBIAS
25 LEFT RANDOM * :LEFTBIAS
30 WORM4 :LEFTBIAS :RIGHTBIAS
END
```

If the turns are small, the resulting paths tend to look like circles or be mostly straight. As the biasing factors become larger, the paths get more jagged and random looking.

Which biasing factors give the best mealworm simulations?

Is the worm's motion constant or intermittent? How can the above procedures be modified to take this into account?

Is the mealworm's turning biased to the left or right?

How about consecutive turns? Are they related randomly or does the worm keep some "preferred direction" from one step to the next?

Maybe the worm's turns are biased, not towards right or left, but rather towards or away from the edges of the box. How could we model this?

Sample Project 2: Edges

Another bug is inherent in all of the above **WORM** procedures -- the turtle keeps running off the edge of the screen. This bug is also rich in ideas to investigate and thus suggests a new project in modeling the behavior of a mealworm crawling near the edge of the box. First of all, how does the mealworm know when it's at the edge? One way could be by touch -- the worm senses the edge when he runs into it. TV-turtle can be provided with a similar "sense" through a **STUCK** operation which indicates whether the previous **FORWARD** command tried to move the turtle out of bounds.[1]

By adding a line at the beginning of any of the above **WORM** procedures we can, for example, have the turtle reverse direction when he runs into the edge:

```
5 IF STUCK RIGHT 180
```

or, not reverse direction completely, but still make a fairly large turn:

```
5 IF STUCK RIGHT 30
```

or, turn a little at a time until he can go forward again:

```
5 IF STUCK (RIGHT 1 FORWARD 1) UNTIL (NOT STUCK)
```

This last variation yields an unexpected dividend. When combined with a random **WORM** procedure it causes the turtle to spend most of the time wandering near the edge of the screen. This provides one possible explanation for something the students may have already noticed while observing the real mealworm -- the worm's preference for remaining near the sides of the box.

The class may question in this context the validity of anthropomorphising the mealworm's actions. If the worm's edge behavior can be accounted for by such a simple mechanism, then are we really justified in saying that the worm "*prefers* to stay near the sides" or "*dislikes* remaining in the middle of the box?" Can we legitimately make these same statements about the turtle?

Suggestions for Discussion

A valuable issue for discussion is what makes a simulation "good." One way to check out the authenticity of a TV-turtle mealworm simulator is to show a friend two pencil tracings of tracks, one from TV-turtle and one from Mealworm, and have him try to guess which is which. But depending on the goals of a simulation, it may be decided that "looking right" is not a sufficient criterion for the goodness of a simulation. To highlight the distinction

between a program which only reproduces an particular path and one which provides a plausible mechanism for behavior, you might prepare for the class a *fixed instruction* procedure that retraces a particular (actual) mealworm path.[2]

Another way to check out the authenticity of a simulation is to compare turtle and mealworm behaviors other than path-shape. Section 7 of the Mealworm Guide suggests the compiling of statistics which deal with how much time the worm spends in various parts of the box. If the students do not think of it themselves, you may wish to suggest that they make similar observations on the behavior of their programs and compare these with the mealworm's statistics.

Of course, with the turtle, we can just as easily make other kinds of measurements -- the total amount turned right versus the total amount turned left, the number of moves before running into the edge of the screen, the distance from the starting point after a given number of moves, and so on.[3] These can also provide ways of comparing different programs.

Notes for Section 2

1. The **STUCK** operation can be conveniently supplied to the students via Logo's **ERRORSET** capability. Redefine the normal **FORWARD** command to clear a "failure signal". Then enable the **OUT OF BOUNDS** error to set the flag:

```
TO STUCK
10 OUTPUT :FORWARD.FAILED
END
```

```
TO FORWARD :N
10 MAKE "FORWARD.FAILED
       "FALSE
20 ERRORSET BOUNDS.ERROR
30 OLD FORWARD :N
END
```

```
TO BOUNDS.ERROR
10 IF ERRORNAME = [OUT OF BOUNDS] MAKE "FORWARD.FAILED "TRUE
20 ERROR.RETURN
END
```

2. A fixed instruction program is one whose behavior is identical each time it is run and is not influenced by outside conditions (the turtle's touch sensor) or by random events (e.g., **FORWARD RANDOM**).

3. The simplest way to record these statistics is to print them by adding lines to the **WORM** procedures. Alternatively, a **RECORD** subprocedure could automatically compile a table

to be printed after a large number of trials. This could be readily modified to compute average values or apply other sampling techniques.

3: Orientation by Touch

Programming the floor-turtle to feel its way along a wall by sense of touch is a natural adjunct to the experiments suggested in Section 3 of the Mealworm Guide on how mealworms follow walls[1]. We have already seen that a worm's wall-following behavior could arise merely as the consequence of a tendency to keep moving with as little change in direction as possible. (See sample Project 2 in Section 2 of this guide.) Using this theory of the behavior of a mealworm, we have no difficulty explaining why a mealworm turns when he reaches the inside of a corner, a wall that turns toward him. His turn may reflect simply a tendency to move in as straight a line as possible. But a real mealworm turns even when the wall bends away from the mealworm and our theory of minimum-turn would not predict that. Students should perform experiments, comparing mealworms and with any wall-following procedures they devise for the turtle.

You may wish to suggest some introductory work with touch sensors before students embark on a wall-following project. Simple procedures can make the turtle bounce back and forth between two obstacles:

```
TO BOUNCE
10 FORWARD 50
20 IF FTOUCH BACK 10 RIGHT 180
30 BOUNCE
END
```

or continually back away from a touch stimulus:

```
TO BACK.AWAY
10 IF FTOUCH BACK 50
20 BACK.AWAY
END
```

Another preparation for wall-following is to program the turtle to work its way along a narrow passageway[2]. This can be accomplished by using a feedback technique: if the turtle touches something on its right, it assumes that it is bumping against the right-hand wall of the passageway and therefore adjusts by turning slightly to the left before proceeding forward. Similarly, a touch on the left causes the turtle to turn right:

```
TO FOLLOW.A.PASSAGEWAY
10 FORWARD 50
20 IF RTOUCH LEFT 10
30 IF LTOUCH RIGHT 10
40 FOLLOW.A.PASSAGEWAY
END
```

When the passageway is much wider than the turtle, the turtle cannot easily use both walls to guide him. Let us assume the turtle has found a wall to his right. When the turtle feels that wall, he should veer away from it (turn left) as before. But when the turtle does *not* feel the wall to his right he knows that he is heading away from the wall and so must turn back a little towards the right to avoid wandering too far from it:

```
TO FOLLOW.A.WALL
10 FORWARD 50
20 IF RTOUCH LEFT 10
30 IF NOT RTOUCH RIGHT 10
40 FOLLOW.A.WALL
END
```

Watch how the turtle can weave its way along the wall in this fashion. Observe closely how it manages to get around corners. This project suggests many variations. For example, program the turtle to get round any obstacle placed in its path by following the obstacle around to the other side. Build an obstacle course and have the turtle work its way from one side of the room to the other.[3]

You should encourage the students to "play turtle," following along a wall by sense of touch and describing their actions in turtle language, to aid in developing these procedures. One difference between the turtle's sense of touch and that of a blindfolded student is that the student can reach out with her arms to feel her way about. How does the task of following a wall while blindfolded change if you must also keep your arms at your sides?

Suggestions for Discussion

These procedures for following walls and passageways are good illustrations of the use of *feedback*: doing something in small steps and making adjustments at each step. It is very common for initial strategies not to make use of this principle. Some of your students may have tried procedures like **FOLLOW1** and **FOLLOW2** (see below) and it is worth comparing them with **FOLLOW.A.WALL**.

A strategy often tried when having the turtle circumnavigate a square object is to measure the side of the square in turtle steps and not use the touch sensors at all:

```
TO FOLLOW1 :SIDE
10 FORWARD :SIDE
20 RIGHT 90
30 FOLLOW1 :SIDE
END
```

A major disadvantage of this method is that the turtle remains very dependent on human help. First, it must be told the size of the object. In addition, it will not successfully round even the first corner unless it is very accurately aimed along the wall. If there is *any* unforseen circumstance (a wheel slips, the aim is not perfect, the object is moved slightly), *the whole program may fail!*

A second strategy eliminates the need for prior knowledge of the object's size:

```
TO FOLLOW2
10 FORWARD 50 UNTIL (NOT RTOUCH)
20 RIGHT 90
30 FOLLOW2
END
```

This approach again relies on the turtle's initial aim. It also has a subtle bug. When the turtle feels no wall to its right, it turns right 90 degrees. But unless we are very fortunate, this turn will not position the right-hand touch sensor exactly along the new wall. So the turtle will go forward, feel no touch, and turn again, this time to face into the wall. Again, no touch is felt on the right, and the turtle turns once more, and so on. The result is that, at the first corner, the turtle will probably get trapped in this way, tracing a small square over and over.

This bug will most likely slip past students who are checking their procedures by playing turtle.[4] Human turtle simulations know enough to be extra careful with their touch sensors when they round a corner. The bug is therefore also a good illustration of the principle that the turtle does exactly what we tell it to do.

In contrast to these less successful attempts, the **FOLLOW.A.WALL** procedure is relatively insensitive to whether the turtle starts out aimed exactly parallel to the wall. It will also work whether or not the corners are 90 degree turns or even if the wall is not straight. That feedback can guide the turtle and continually correct its path as the turtle moves along is well illustrated by these experiments. But the real importance in seeing this point is that feedback is a powerful idea essential to most behavior and all learning.

Notes for Section 3

1. It is of course also possible to provide TV-turtle with a simulated sense of touch. The use of floor-turtle, however, can provide a more concrete experience. In addition, the inevitable innaccuracies of a mechanical device tend to force students to incorporate feedback techniques in their algorithms.

2. Passageways, walls and obstacles are readily constructed by arranging bricks on the classroom floor. The bricks are easy to rearrange and yet heavy enough so that the turtle won't move them when it runs against them.

3. While working on this project, one student developed an algorithm which allows the turtle to get around *any* obstacle at all (a universal maze-solving algorithm). The basic idea is this: the turtle proceeds in a preferred direction, say "northward," until it runs into a wall. It then follows the wall until it is once again facing "north" and its total turning while following the wall (amount turned right minus amount turned left) is zero. Then it continues "northward." For further details see Seymour Papert, "The Uses of Technology to Enhance Education," Memo No. 298, Artificial Intelligence Laboratory, MIT.

4. Because people working with the turtle navigation experiments have often suggested a similarity to the experience of blind persons, we feel a comment on that observation worthwhile. Although there are, indeed, insights about blindness to be gained from the experiments, it important to note that neither the student's personal experience walking blindfolded nor the observed "experiences" of the turtle are complete or accurate representations of the experience of a blind person. A blindfolded person already has a sight-based model of the world around him, even when walking in unfamiliar territory, and any simple turtle program is obviously using much less information than a blind person has. Still, some of the mobility difficulties of blind people may be understood when we consider what thinking *we* must go through to tell the turtle how to get around an obstacle.

4: Sensory Information -- Modeling Smell

If we place some bran in the box with the mealworms, they will crawl around and eventually find it. The experiments in the Mealworm Guide (Sections 3,4 and 8) lead the class to consider what kind of sensory information the mealworms use in locating the bran. Making computer models requires that we be more precise. Not only must we consider the sensory modality (sight, hearing, smell, etc.) but also

exactly what information is being received, and

how does the worm use this information to locate the food?

This is a subtle point, for adults as well as for children. We are so accustomed to the ways in which we use our *own* senses that it is difficult to imagine other possibilities. For instance, an initial discussion of how a mealworm might use *sight* to locate food is likely to get no further than "He sees the food and he goes to it."

It seems better, therefore, to begin by modeling a sense like *smell*. Since we ourselves do not normally navigate by odors, it should be easier for the students to be objective about how a mealworm might do so.

A worm's ability to "smell" could be furnishing him with many different kinds of information. For example:

The "amount of smell" could be a value which depends on how far the worm is from the food. The larger the distance, the weaker the smell.

As above except that there might be only three levels of smell: either the worm doesn't smell the food at all, or he smells it a little, or the smell is very intense when he's right at the food.

The worm might not sense any particular *level* of smell, but each time he moves he is able to tell whether the smell is getting stronger or weaker.[1]

Hot-Cold Game

One way to help get this point across to the students is to organize a "hot-cold" game as a human simulation of "smelling something out." Typically, one person is trying to find some hidden object while the onlookers shout out "hot" or "cold" to indicate how well the detective is doing. Since human detectives, especially young ones, tend to be distracted as often as helped by hunches, it is probably best to blindfold the searcher. Several varieties might be worth trying:

1. The searcher receives information based upon distance from the object. Before beginning this game, the class must agree on "zones" and tell the searcher some number between 1 and 5 (or some more suitable scale) depending on his closeness. Several ideas can be explored here. How accurate need the class be? How accurate need a sense organ be? Is a 1 to 10 scale better than a 1 to 5? Does the direction the person faces influence the score he gets? Different answers to these questions may, themselves, suggest different kinds of senses.

2. The same kind of information with a coarser scale. A scale of 1 to 3, where 3 means "within easy hand-reach", 2 means "within two or three steps and a short handreach", and 1 means "farther than that" might be best.

3. The searcher receives information based upon the *change* in distance from the object. Again, before beginning the game, the class must decide how much of a difference in distance will be considered significant. The searcher is never told how close he is, but gets the same "you are closer" or "you are farther" regardless of his absolute distance. At what times does the searcher receive feedback? After each move, or only when he has moved a significant distance -- enough to change his status? Should "significant distance" depend in any way on absolute distance? (A change in one foot is not as significant when one is 100 feet from the target as it is when one is 2 feet from it.)

4. To help in making the transition to writing turtle programs, you should vary each of the above games so that no information is given to the searcher except when the searcher *specifically asks* for it. The feedback in game (3) will then refer to difference from last position tested, rather than last position visited.

All of these games might be best played on a flat open area, such as a gymnasium or playground, both to allow for less restricted motion and to minimize conjectures based on "knowing the lay of the land." (In a classroom with desks in rows, for example, a student may "search" the class by going back and forth the length of the room, row by row. In an open territory, there are fewer landmarks and the search procedure will be more dependent on the feedback gained.) It may be interesting to keep a record of the paths of the searchers in the different games. If the playing field can be marked off as a grid, perhaps with students as the markers at the edge or with chalk on the ground, the path can be traced easily.

Sample Project

Each of the above variations suggests a computational investigation of how a creature could use a sense like smell in locating food. For example, the "closer-farther" kind of smell in game (3) is reflected in the procedure:[2]

```
TO SMELL
10 IF DISTANCE.TO.FOOD > :DISTANCE.LAST.TIME
      MAKE "RESULT [WEAKER]
      ELSE MAKE "RESULT[STRONGER]
20 MAKE "DISTANCE.LAST.TIME DISTANCE.TO.FOOD
30 OUTPUT :RESULT
END
```

How can the turtle use this information to locate the food? One possibility is this: if the turtle finds that the smell is getting stronger he keeps going in the same direction, otherwise he turns:

```
TO FIND.BY.SMELL1
10 FORWARD 1
20 IF SMELL = [WEAKER] RIGHT 1
30 FIND.BY.SMELL1
END
```

Experimenting further, we can add a parameter to adjust the size of the turtle's turns. This leads to an interesting study of how the geometry of the path varies with the turn angle:

```
TO FIND.BY.SMELL2 :TURN
10 FORWARD 1
20 IF SMELL = [WEAKER] RIGHT :TURN
30 FIND.BY.SMELL2 :TURN
END
```

A more realistic simulation would also include some random motion as developed in the projects in Section 2 of this guide:

```
TO FIND.BY.SMELL3 :TURN
10 FORWARD RANDOM
15 LEFT RANDOM
16 RIGHT RANDOM
20 IF SMELL = [WEAKER] RIGHT :TURN
```

```
30 FIND.BY.SMELL3 :TURN
END
```

Adding a biasing factor to the random turns in lines 15 and 16 of **FIND.BY.SMELL3** suggets another investigation:

```
TO FIND.BY.SMELL4 :TURN :BIAS
10 FORWARD RANDOM
15 LEFT RANDOM * :BIAS
16 RIGHT RANDOM * :BIAS
20 IF SMELL = [WEAKER] RIGHT :TURN
30 FIND.BY.SMELL4 :TURN :BIAS
END
```

In this procedure the turtle's motion is governed by two opposing tendencies -- a "random motion" scaled by **BIAS** and a "directed motion" scaled by **TURN**. This can be highlighted by adjusting the relative sizes of the two parameters. How large, for example, must **BIAS** be with respect to **TURN** before the random motion dominates completely and the turtle makes no discernable progress towards the food?

Finally, consider how the **FIND.BY.SMELL** mechanism would behave if the *food* were also moving, as in the case of a predator trying to catch dinner. There are many possible variations to try. Suppose, for example, that the hunted creature, unaware of the predator's intensions, moves round and round in a circle:

```
TO FOOD.STEP :SPEED
10 FORWARD :SPEED
20 RIGHT 1
END
```

while the predator uses the **FIND.BY.SMELL2** procedure with a **TURN** of 90°:

```
TO CHASE.STEP :SPEED
10 FORWARD :SPEED
20 IF SMELL = [WEAKER] RIGHT 90
END
```

The result is seen by having the predator and prey move simultaneously:[3]

```
RUN.TOGETHER [CHASE.STEP :CHASE.SPEED] [FOOD.STEP :FOOD.SPEED]
```

The path shows that a predator would remain very hungry by trying to catch dinner using smell in this way. The geometry of the path, however, is interesting in its own right. It

seems amazing, for example, that the predator's path is closed. This should be investigated further, using different speeds, different **TURN** angles in the **CHASE.STEP** procedure and different initial positions for predator and prey.

Suggestions for Discussion

The experiences in modeling smell suggest other experiments with both mealworm and turtle:

Does the mealworm sense the direction of the stimulus?

Does it move toward or away from the smell?

Can it distinguish slightly differing intensities of smell?

Does it recognize different kinds of smell?

Does the amount of smell depend upon which way the animal (worm or turtle) is facing?

The work on smell also forms a foundation for comparing sensory modes with respect to the *kind of information* received from a stimulus. Normal human vision, for example, gives information at a distance but only from a limited direction (less than 180°). The information tells direction of stimulus, distance of stimulus, shape, size and color of the stimulus. Most non-human vision does not give this much information. Some gives no more information than that there is or is not a certain amount of light present. Some human vision does not give color information accurately. Some, when unaided by glasses, does not give shape information accurately. Hearing has a different set of parameters. Normal human hearing gives information at a distance and from *any* direction. Still, it tells *which* direction the stimulus is coming from and, in most contexts, gives some information about the distance of the stimulus. Both of these senses tell us *when* the stimulus producing agent is still present.

Human smell is quite different. Information comes from a distance and from all directions, but does not tell us which direction the stimulus is coming from. It gives us little to no information about the distance and does not tell us whether the stimulus producing agent is still present. We can distinguish many different kinds of smells, but they are not as distinct to us as they are, for example, to a bloodhound. We can tell in a crude way if the smell is becoming more or less intense, and can sometimes locate objects by smell, but in a much more try-and-compare sort of way than with hearing or sight. Although there are sights and sounds we like and dislike, they are not generally universally aversive unless they are terribly sudden or intense. Certain smells, however, are highly aversive, even when not considered especially intense, and may be highly aversive not only to people but to other creatures as well.

Children generally have theories about these differences in the senses. While they might test a mealworm to see if it likes or dislikes a particular smell, it would seem less likely that they would test it to see if it liked or disliked a particular picture or popular song. It may help students become aware of their theories and conjectures if you write down the assumptions that underly their statements during discussions.

Notes for Section 4

1. This is an example of a *gradient field* as opposed to an *intensity field*, that is, decisions are based not directly on the observed intensity of a stimulus but rather upon the change in intensity as the observer moves.

2. The **SMELL** procedure makes use of the following subprocedure which outputs the distance from the TV-turtle to a named point:

```
TO DIST :POINT
10 LOCAL "XDIST
20 LOCAL "YDIST
30 MAKE "XDIST XCOR - FIRST :POINT
40 MAKE "YDIST YCOR - FIRST :POINT
50 OUTPUT SQRT ( :XDIST * :XDIST + :YDIST * :YDIST )
END
```

2. The **RUN.TOGETHER** procedure runs two turtle procedures "simultaneously":

```
TO RUN.TOGETHER :STEP1 :STEP2
5  INIT
10 RUNSTEP :STEP1 :PLACE1
20 MAKE "PLACE1 HERE
30 RUNSTEP :STEP2 :PLACE2
40 MAKE "PLACE2 HERE
50 RUN.TOGETHER :STEP1 :STEP2
END

TO RUNSTEP :STEP :PLACE
10 PENUP
20 SETTURTLE :PLACE
30 PENDOWN
40 RUN :STEP
END
```

The procedures referenced by **STEP1** and **STEP2** must be "one step" moves, i.e.,

without the usual recursion line to keep them running over and over. **INIT** is a procedure which must be supplied by you to initialize the starting states **PLACE1** and **PLACE2** for the two **STEP** procedures.

5: Modeling Sight

As with smell, the first step in equipping TV-turtle with a simulated eye is to decide what information the eye should receive from the environment. While we could hardly begin to model the complexity of human vision, a mealworm's sense of sight is a much simpler affair. One plausible model for mealworm vision ignores all the color, shape and texture information that our own eyes perceive and registers merely the intensity of light reaching the eye. We can experience something like this by covering our eyes with a piece of paper. Students may wish to try this and see whether they can locate an object such as a light while relying on "mealworm vision."

This kind of "sight" is not so different from the "smell" discussed in Section 4: each receives some kind of intensity information from the environment. The major difference is that sight is *directional*, it depends on how the turtle is facing with respect to the stimulus. Algorithms for locating an object by sight, even mealworm sight, are therefore different from the "smelling something out" of Section 4. The paragraphs below suggest three ways of furnishing the turtle with mealworm vision of various levels of complexity. You might have the entire class concentrate on one of them, or have groups of students working with each and comparing results.

Facing a Stimulus

The first model assumes that any creature able to see a light is able to turn to face that light. So students can investigate what new things the turtle can do when given the ability to **FACE** a named point.[1] Getting to the point is easy. Simply face the point and go forward. (But how does the turtle know when to stop?) Even this simple scheme has a number of fascinating variations. What happens, for example, when we allow the pursued point to move in a circle (as in Section 4):

```
TO CHASE.STEP :SPEED
10 FACE :FOOD
20 FORWARD :SPEED
END

TO FOOD.STEP :SPEED
10 FORWARD :SPEED
20 RIGHT 1
30 MAKE "FOOD HERE
END

RUN.TOGETHER [CHASE.STEP :CHASE.SPEED] [FOOD.STEP :FOOD.SPEED]
```

Students may wish to study the paths generated by the above sequence as they vary the

speeds of predator and prey. In addition they could develop various "evade strategies" for the pursued creature.[2]

Another use for the FACE command is to have the turtle not face a point directly but rather keep the point at a fixed bearing:

```
TO KEEP.A.BEARING :POINT :ANGLE
10 FACE :POINT
20 RIGHT :ANGLE
END
```

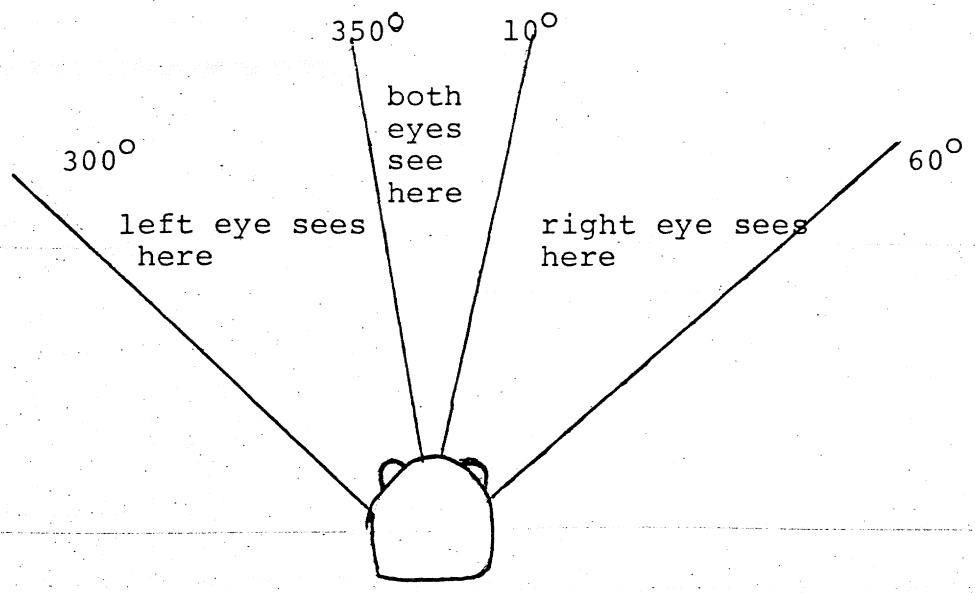
The following procedure has the turtle move while keeping a fixed point at a 90° bearing:

```
TO FIXED.BEARING :POINT
10 KEEP.A.BEARING :POINT 90
20 FORWARD 10
30 FIXED.BEARING :POINT
END
```

If you try this procedure you will find that it causes the turtle to spiral in about the point. Does this remind you of anything? How about a moth getting trapped by a light? But why would a moth be trying to keep a light at a fixed bearing? Some people believe that moths and other night-flying insects have learned to fly along straight paths by keeping the *moon* at a constant bearing as they fly. Keeping a very distant light like the moon at a fixed bearing does indeed make the insects fly straight. But when they confuse the moon with a nearby light, the fixed-bearing mechanism produces a spiral.

A Two Eye Model

The next model focusses on *how* a creature might use vision in order to face a point. Assume that the turtle, like the real mealworm, has two eyes, each with its own field of vision:



We give the turtle the ability to tell whether a point is within each eye's fieled of vision:[3]

```
TO LEFT.EYE.SEES :POINT
10 IF (BEARING :POINT) > 300 THEN OUTPUT "TRUE
20 IF (BEARING :POINT) < 10 THEN OUTPUT "TRUE
30 OUTPUT "FALSE
END

TO RIGHT.EYE.SEES :POINT
10 IF (BEARING :POINT) > 350 THEN OUTPUT "TRUE
20 IF (BEARING :POINT) < 60 THEN OUTPUT "TRUE
30 OUTPUT "FALSE
END
```

The turtle will know that he is facing roughly in the direction of a named point when the point lies in at the field of vision on at least one side. So, as he moves, he should keep checking that he can still see the point. Otherwise he turns:

```
TO HEAD.FOR :POINT
10 IF (LEFT.EYE.SEES :POINT) THEN
      FORWARD 10 UNTIL (NOT LEFT.EYE.SEES :POINT)
20 IF (RIGHT.EYE.SEES :POINT) THEN
      FORWARD 10 UNTIL (NOT RIGHT.EYE.SEES :POINT)
30 SEARCH
40 GO 10
END

TO SEARCH
10 RIGHT 10 UNTIL LEFT.EYE.SEES :POINT
END
```

It may seem amazing that a turtle following this procedure manages to reach the point despite the fact that his way of heading for the point is so innaccurate. Once again, this illustrates the resilience of a feedback mechanism -- constant adjustment can often compensate for lack of accuracy. Students may wish to try some of the "moving food" and "fixed bearing" projects with this model as well.

A Two Eye Model with Intensity

A more elaborate model for vision registers not only the presence of a light source in the visual field, but also the *intensity* that each eye receives from the source. This intensity depends on the strength of the source, the distance of the source from the animal and also the angle at which the light strikes the animal's eye. The intensity is greatest when the light hits the eye "straight on" and tapers off to zero as the light source moves toward the edge of the visual field.

The notes[4] describe how to construct procedures **INTENSITY.LEFT** and **INTENSITY.RIGHT** which output the intensity received from a light source. To get an idea of how these work, students should prepare some graphs of how the intensity varies, perhaps a graph of **INTENSITY** versus **BEARING** for a fixed source.

There is a simple yet effective way to incorporate intensity information in a feedback mechanism to make the turtle approach a light source. The turtle walks forward while trying to keep the amount of light received at both eyes "in balance." So, if he sees more light to his right, he turns slightly to the right. If he sees more light to his left, he turns slightly to the left.

```
TO FIND.BY.SIGHT :SOURCE
10 TEST (INTENSITY.LEFT :SOURCE) > (INTENSITY.RIGHT :SOURCE)
20 IFTRUE LEFT 10
30 IFFALSE RIGHT 10
40 FORWARD 5
50 FIND.BY.SIGHT :SOURCE
END
```

Real animals may actually use this mechanism for approaching light sources. Biologists have obtained experimental evidence for this conclusion by taking animal and masking one of its eyes (say, the left one). What happens when the animal tries to approach the light? You and your students can simulate this experiment by modifying **INTENSITY.LEFT** to always output zero and have the turtle follow the **FIND.BY.SIGHT** procedure. You can see that the **TEST** in line 10 will now always be **FALSE** and so the turtle will always turn right -- therefore travel in a circle. Biologists call this behavior "circus movement." It has been observed in experiments with numerous species of insects.[5]

Students may wish to undertake projects which grow out of varying the **FIND.BY.SIGHT** procedure. You might suggest having the light source move, and see how well the turtle manages to follow it. Does **FIND.BY.SIGHT** work better than **FIND.BY.SMELL** (Section 4) for chasing a moving object?

Another possible project arises from the "circus movement" experiment described above. Modify **INTENSITY.LEFT** to output, not zero, but half its normal value. (This corresponds to an animal with weak vision in one eye.) What kind of path does **FIND.BY.SIGHT** produce now? Does the animal still reach the light? How does the path degenerate to a "circus movement" as the eye becomes weaker and weaker?

Finally, as a very ambitious project, consider what happens when there are two or more light sources. The intensity for each eye is found by adding together the intensities from the individual sources:

```
TO FIND.BY.SIGHT2 :SOURCE1 :SOURCE2
10 MAKE "TOTAL.LEFT
      (INTENSITY.LEFT :SOURCE1) + (INTENSITY.LEFT :SOURCE2)
20 MAKE "TOTAL.RIGHT
      (INTENSITY.RIGHT :SOURCE1) + (INTENSITY.RIGHT :SOURCE2)
30 TEST :TOTAL.LEFT > :TOTAL.RIGHT
40 IFTRUE RIGHT 10
50 IFFALSE LEFT 10
60 FORWARD 10
70 FIND.BY.SIGHT2 :SOURCE1 :SOURCE2
END
```

How does the turtle behave? Does it go to the stronger light? to the closer light? between the lights? Keep records of what happens for different strength sources and different initial positions of turtle and sources. This experiment, called the "two light experiment" is often performed with insects.[6]

Notes for Section 5

1. The **FACE** uses the a subprocedure called **BEARING** which outputs, in effect, how much the turtle needs to turn right in order to face the source. If **BEARING** is not supplied as a primitive to the system, it can be constructed as in Note 3 below.

```
TO FACE :POINT
10 RIGHT BEARING :POINT
END
```

2. One particularly interesting variation is to have the pursuer use the **CHASE.STEP** procedure and the evader keep heading at a relative bearing of 90° to the pursuer:

```
TO FOOD.STEP :SPEED
10 KEEP.A.BEARING :CHASER 90
20 FORWARD :SPEED
30 MAKE "FOOD HERE
END
```

Running **CHASE.STEP** and **FOOD.STEP** together with the food moving faster than the chaser, produces a pattern where both creatures end up travelling in fixed circles. Can you understand why this happens?

3. This note shows how to reconstruct the **BEARING** primitive if it is not supplied with the system. **BEARING** is defined using **TOWARDS**, which outputs the direction of a point with respect, not to the turtle's currentheading, but rather with respect to a heading of zero degrees (straight up).

```
TO BEARING :POINT
10 LOCAL "BEAR
20 MAKE "BEAR TOWARDS (FIRST :POINT) (FIRST BUTFIRST :POINT)
30 MAKE "BEAR :BEAR - HEADING
40 IF :BEAR < 0 OUTPUT :BEAR + 360 ELSE OUTPUT :BEAR
END
```

TOWARDS is essentially an arctangent function determined by the difference between the turtle's position and the named point. To compute it, assume that we have a **TOWARDS.SUB1** procedure which works for *positive* values of **DX** and **DY**. Then we

can get the answer in general by appropriately modifying the output of **TOWARDS.SUB1** by 180 or 360 degrees:

```
TO TOWARDS :X :Y
10 LOCAL "DX LOCAL "DY LOCAL "ANG
20 MAKE "DX :X - XCOR  MAKE "DY :Y - YCOR
30 MAKE "ANG TOWARDS.SUB1 (ABS :DX) (ABS :DY)
40 IF :DY < 0 MAKE "ANG 180 - :ANG
50 IF :DX < 0 MAKE "ANG 360 - :ANG
60 OUTPUT :ANG
END
```

Finally, we must write **BEARING.SUB1**. This checks for the special cases where **X** or **Y** is zero. Otherwise it outputs the arctangent of **X/Y**:

```
TO TOWARDS.SUB1 :X :Y
10 IF :X = 0 OUTPUT 0
20 IF :Y = 0 OUTPUT 90
30 OUTPUT ARCTAN :X/:Y
END
```

4. Following the model given in *The Orientation of Animals*, by G. Franekel and D. Gunn (Dover, 1961) the intensity of light falling on the eye is (S/D^2) *cos* A where S is the strength of the source, D is the distance from the source, and A is the angle at which light from the source strikes the eye. Accordingly we have:

```
TO INTENSITY.LEFT :SOURCE
10 IF NOT ( LEFT.EYE.SEES :SOURCE ) OUTPUT 0
20 LOCAL "FACTOR LOCAL "DIR
30 MAKE "FACTOR :STRENGTH / ( (DIST :SOURCE) * (DIST :SOURCE) )
40 MAKE "DIR BEARING :SOURCE
50 OUTPUT :FACTOR * COS (315 - :DIR)
END


TO INTENSITY.RIGHT :SOURCE
10 IF NOT ( RIGHT.EYE.SEES :SOURCE ) OUTPUT 0
20 LOCAL "FACTOR LOCAL "DIR
30 MAKE "FACTOR :STRENGTH / ( (DIST :SOURCE) * (DIST :SOURCE) )
40 MAKE "DIR BEARING :SOURCE
50 OUTPUT :FACTOR * COS (45 - :DIR)
END
```

STRENGTH is a parameter which you must supply to indicate the intensity of the source.

5. See the book by Fraenkel and Gunn.

6. Ibid.

Appendix: Bugs in Behaviors

Computational descriptions of animal behavior can help explain the potential *bugs* in an animal's behavioral repertoire. Often these bugs surface when a change in the animal's enviornment causes some behavioral mechanism to be "fooled." We have already mentioned one example in Section 5: the moth's evolved mechanism for orienting its night flight by moonlight leads it to become trapped by nearby artificial lights. The following true story gives another example of such a behavioral bug.

The Story of Bal and the Blanket

Consider Bal Boa. As Bal slithers about, he meets up with a variety of objects. Some are rocks and pebbles warmed by the sun or cooled by the shade, some are leaves blowing about in the wind, some are tasty mice, some are beautiful boas, and, depending on Bal's neighborhood, some might even be people. In a natural environment, a snake's decisions about what things to eat are based on a few very reasonable conditions. Bal doesn't eat rocks or vegetables, so he merely ignores them. And as for animals, Bal can't chew them up or rip them apart or spit them out, so, before starting a meal, Bal must be quite sure that he can finish it. That means that Bal's meal must be small enough to swallow whole! And whether an animal is large and ill-flavored or small and delicious, if Bal is still busy digesting last week's meal he'd rather avoid the other creature's company. You can never trust a rat.

Never mind, for the moment, how Bal knows he's hungry. (Maybe his stomach rumbles?) And also never mind how he sizes up an animal. It may sound silly to wonder how he even knows when he has seen an animal, because, for us people, that seems so easy. But Bal has never been a boy-scout, does not recognize animal tracks and can't tell one kind of fur from another. He only knows this: *a possible food animal must be warm enough to be alive and must be moving.* So, to summarize, Bal asks himself these three questions before deciding to attack: *Am I hungry? Is it an animal? Can I swallow it whole?* If all three answers are *YES!*, Bal attacks. A Logo program to describe how Bal comes to his decision might look like this:

```
TO DECIDE.TO.ATTACK :THING
10 IF EITHER ( TOOBIG :THING ) ( NOT HUNGRY ) AVOID :THING STOP
20 IF ANIMAL :THING ATTACK :THING
END

TO ANIMAL :IT
10 IF BOTH ( MOVING :IT ) ( WARM :IT ) OUTPUT "TRUE
20 OUTPUT "FALSE
END
```

```
TO ATTACK :IT
10 GRAB :IT
20 COIL.AROUND :IT
30 SQUEEZE :IT UNTIL JUST.DIED :IT
END
```

Bal coils himself around his food and squishes it. The animal's movements (struggling, heartbeat and breathing) all cause Bal to squish a little harder until the meal can't breathe any more. Bal certainly does not want to eat anything while it is still alive, so Bal must be quite sure the thing is dead, and not just faking it. How does Bal know? He senses the temperature, and if the little animal has cooled down enough, Bal knows it must have died. We can write a Logo program for Bal to do this, too.

```
TO EAT :THING
10 IF NOT COILED.AROUND :THING STOP
20 IF JUST.DIED :THING SWALLOW :THING
END

TO JUST.DIED :IT
10 IF TEMPERATURE.CHANGED.FROM.WARM.TO.COOL.VERY.RECENTLY :IT
      OUTPUT "TRUE
20 OUTPUT "FALSE
END
```

Still, remember that Bal will not eat just any old dead thing. Things that cool down *near* Bal are just not the same as meals Bal has prepared himself, and he will not eat them. But if he is coiled around something and that thing cools down, that generally convinces Bal he killed it. And surely if he killed it, he must have intended to eat it. So....

But, alas, poor Bal did not live in the wild. He lived with my friend Marsha, and she gave Bal a nice heating pad so that Bal could enjoy the comfort of a warm and cozy home.

"And," thought Bal to himself, "what can a nice nine-foot snake like me do with such a small one-foot heating pad? It isn't running around so it can't be an animal so I won't attack it and I won't run away from it. Yet it is nice and warm. It is too small to wrap around me, but I can wrap around it." Marsha was afraid, however, to leave the heating pad connected while she went out shopping and so she unplugged it. And now Bal, coiled snugly around the unplugged pad, began to notice that it was cooling down.

"That's interesting," he thought. "I don't recall having attacked this thing and I'm sure it didn't squirm or struggle or breathe or even have a heartbeat. Yet, there can be no doubt, I am coiled around it and it has certainly died because it is getting cooler. So I guess I must have killed it. And *that* means I must have been planning a meal! I'm getting more absent

minded week by week." When Marsha returned from shopping, there was Bal stretched out in a lazy curve with a plug hanging out of his mouth at the end of a few feet of electric cord.

Despite the bug in Bal's program, the story has a happy ending. Bal *did* show signs of indigestion, but the heating pad was removed in time and a few days later Bal had a much more wholesome meal.

Discussion

One can imagine using this story in a variety of ways depending on the interests of the group. The act of eating is so natural to us and seems so very lacking in cleverness that one goal of this story is to point out the kinds of decisions that must go into even such a primitive behavior as that. After becoming impressed with the complexity of Bal's program, we might turn about face and marvel at its simplicity. Our own human program is, at present, unanalyzably complex. A message to derive from this comparison is that *a creature develops in an environment.* Though it is able to adapt to unexpected variations in its environment and respond appropriately to some novel stimuli, this ability is not unlimited. A creature can in general only learn to cope with those contingencies of its environment that it is likely to meet *in the environment in which it has developed.* Bal's program does not *need* to account for things like heating-pads and it would be wasteful of brain power for it to be such a general program. Students who are interested in this aspect of animal behavior might like to pursue it with other readings in the area.