1

Five centuries ago the printing press sparked a radical reshaping of the nature of education. By bringing a master's words to those who could not hear a master's voice, the technology of printing dissolved the notion that education must be reserved for those with the means to hire personal tutors. Today we are approaching a new technological revolution, one whose impact on education may be as far-reaching as that of the printing press: the emergence of powerful computers that are sufficiently inexpensive to be used by students for learning, play, and exploration. It is our hope that these powerful but simple tools for creating and exploring richly interactive environments will dissolve the barriers to the production of knowledge as the printing press dissolved barriers to its transmission.

This hope is more than our wish for students to experience the joy of discovery and the give and take between investigator and investigation that typifies scientific research. Like Piaget, Dewey, and Montessori, we are convinced that personal involvement and agency are essential to truly effective education. Yet much of almost any mathematics curriculum is devoted to practicing rote algorithms and rehashing ancient theorems. It is the rare student who gets the chance to approach mathematics by doing it rather than only learning about it, by investigating new phenomena, by formulating original hypotheses, or by proving original theorems. Computation—especially the activity of programing—can offer many opportunities for students to engage in such activities without first having to master a formidable apparatus. We hope to demonstrate how a computational approach can change the relation between the students and the mathematical knowledge.

Excerpts from Abelson, Harold. (1979). *Turtle Geometry*.
Cambridge, MA. MIT Press.

2

Experience is an important ingredient in discovery. The abundance of the phenomena students can investigate on their own with the aid of computer models shows that computers can foster a style of education where "learning through discovery" becomes more than just a well-intentioned phrase. Computers can bring to learning the essential element of surprise, for despite the popular belief that a computer can never surprise its programer since it does only what it was programed to do, even very simple algorithms can and often do produce unexpected and striking results. Encountering one of these results, studying it, and understanding how it comes about can be an open-ended adventure very different from most of the "discovery methods" in teaching, in which the teacher knows beforehand precisely what is supposed to be "discovered."

This book is a computer-based introduction to geometry and advanced mathematics at the high school or undergraduate level. Besides altering the form of a student's encounter with mathematics, we wish to emphasize the role of computation in changing the nature of the content that is taught under the rubric of mathematics. We wish to demonstrate a curriculum that shows the computational influence in its choice of ideas as well as in its choice of activities. Some of the major themes of the book illustrate the point: representation, local-global dichotomy, linearity, state and state-change operators. Most important in this endeavor is the expression of mathematical concepts in terms of constructive, process-oriented formulations, which can often be more assimilable and more in tune with intuitive modes of thought than the axiomatic-deductive formalisms in which these concepts are usually couched. As a consequence we are able to help students attain a working knowledge of concepts such as topological invariance and intrinsic curvature, which are usually reserved for much more advanced courses.

Imagine that you have control of a little creature called a turtle that exists in a mathematical plane or, better yet, on a computer display screen. The turtle can respond to a few simple *commands*: `FORWARD` moves the turtle, in the direction it is facing, some number of units. `RIGHT` rotates it in place, clockwise, some number of degrees. `BACK` and `LEFT` cause the opposite movements. The number that goes with a command to specify how much to move is called the command's *input*.

In describing the effects of these operations, we say that `FORWARD` and `BACK` change the turtle's *position* (the point on the plane where the turtle is located); `RIGHT` and `LEFT` change the turtle's *heading* (the direction in which the turtle is facing).

The turtle can leave a trace of the places it has been: The position-changing commands can cause lines to appear on the screen. This is controlled by the commands `PENUP` and `PENDOWN`. When the pen is down, the turtle draws lines. Figure 1.1 illustrates how you can draw on the display screen by steering the turtle with `FORWARD`, `BACK`, `RIGHT`, and `LEFT`.

Excerpts from Abelson, Harold. (1979). *Turtle Geometry*.
Cambridge, MA. MIT Press.

Turtle geometry would be rather dull if it did not allow us to teach the turtle new commands. But luckily all we have to do to teach the turtle a new trick is to give it a list of commands it already knows. For example, here's how to draw a square with sides 100 units long:

```
TO SQUARE
   FORWARD 100
   RIGHT 90
   FORWARD 100
   RIGHT 90
   FORWARD 100
   RIGHT 90
   FORWARD 100
```

This is an example of a *procedure*. (Such definitions are also commonly referred to as programs or functions.) The first line of the procedure (the *title line*) specifies the procedure's name. We've chosen to name this procedure `SQUARE`, but we could have named it anything at all. The rest of the procedure (the *body*) specifies a list of instructions the turtle is to carry out in response to the `SQUARE` command.

There are a few useful tricks for writing procedures. One of them is called *iteration*, meaning repetition—doing something over and over. Here's a more concise way of telling the turtle to draw a square, using iteration:

```
TO SQUARE
   REPEAT 4
      FORWARD 100
      RIGHT 90
```

This procedure will repeat the indented commands `FORWARD 100` and `RIGHT 90` four times.