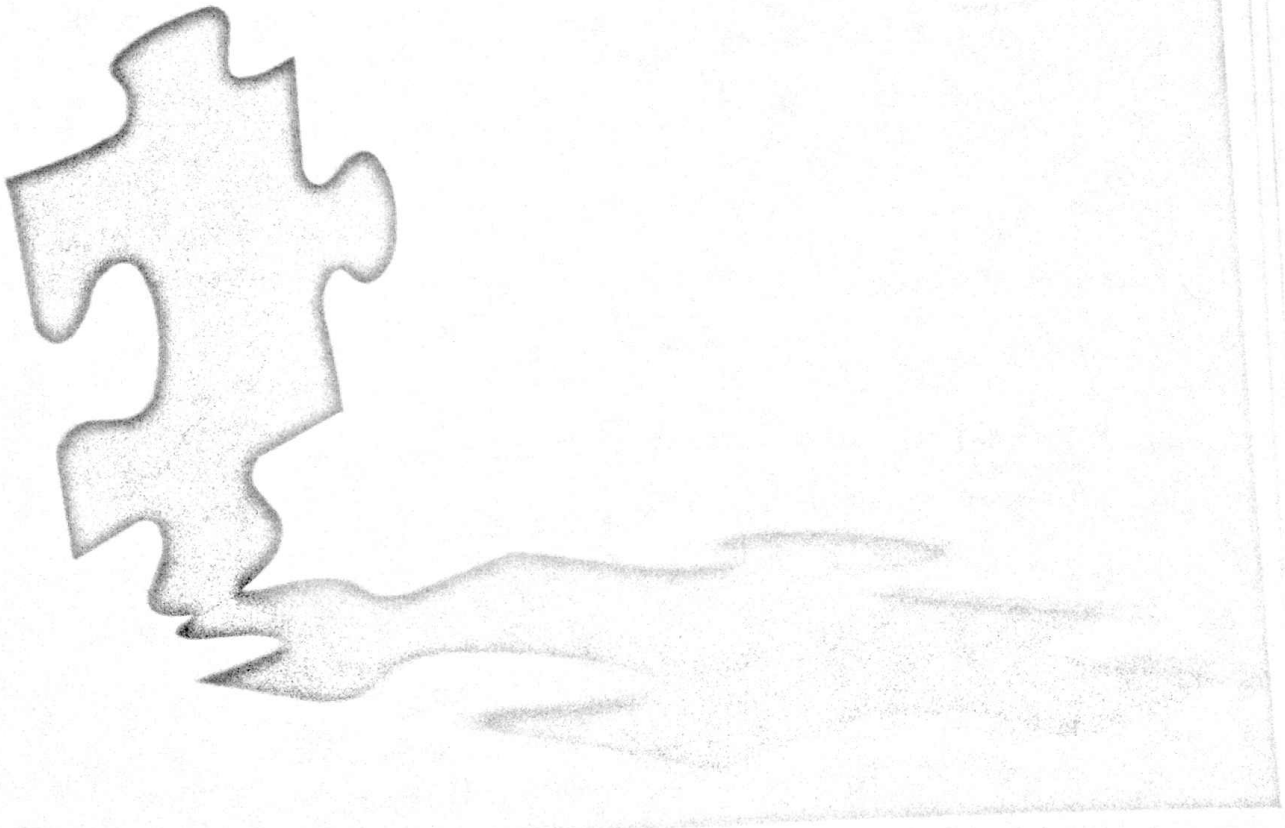
8

Teaching by Providing Concreteness, Activity, and Familiarity

Chapter Outline

This chapter investigates three instructional methods aimed at producing meaningful learning through guided exploration—concrete methods that make the learning task more concrete, discovery methods that make the learning task more active, and inductive methods that make the learning task more familiar. In each case, the learner explores a problem or task and is given various forms of guidance—including concrete materials (concrete methods), hints (discovery methods), and links to prior experience (inductive methods). For each technique, the chapter provides an example, explains the underlying theoretical concepts, describes some representative classic research, suggests some implications for instruction, and provides current applications using computers.



The Parallelogram Problem

Consider the parallelogram problem in Figure 8–1. How would you teach children to solve problems like this one? Let's assume that the children have already learned how to find the area of a rectangle but have not yet learned about the area of a parallelogram.

The Gestalt psychologist Wertheimer (1959) contrasts two distinct methods of instruction for the parallelogram problem. The first method is to teach the child to find the height and the base and to plug them into the formula Area = Height × Base. In the parallelogram problem in Figure 8–1, the child must find that the height is 5, the base is 11, so the area is 5 × 11, or 55. Wertheimer calls this approach the "rote method" because the child learns to mechanically apply a formula. The rote method is summarized in the top of Figure 8–2.

The second method suggested by Wertheimer (1959) is to allow the child to have *structural insight* (i.e., to see how a parallelogram can be changed into a rectangle by moving the triangle on one end to the other end). Once the child sees how to restructure the parts of a parallelogram into a rectangle, the child can go ahead using a previously learned method for finding the area of a rectangle. Wertheimer calls this approach the *meaningful method* because the learner understands how the parts of a parallelogram fit together. The middle of Figure 8–2 summarizes the meaningful method.

Why should we be concerned whether a child learns by rote or by understanding? Isn't it enough to teach the child how to use the formula effectively so that the child can get the right answer on parallelogram problems? Wertheimer's (1959) answer to these questions is that *understanding* is important for some instructional objectives but not for others. For example, according to Wertheimer, both methods of instruction lead to good performance on standard problems like those given as examples during instruction. Thus, if the goal of instruction is efficient application of a rule on standard problems, the meaningful method of instruction is not needed. However, what happens when you present children with unusual problems such as shown in the bottom of Figure 8–2? According to Wertheimer, the children who learned by understanding are able to solve transfer problems, whereas the children who learned by rote say, "We haven't had that yet." Thus, the payoff for meaningful methods of instruction is not in exact retention of the taught material but rather in creative transfer to new situations. If the goal of instruction is that the child be able to creatively apply learning in new situations, then meaningful methods of instruction are important.

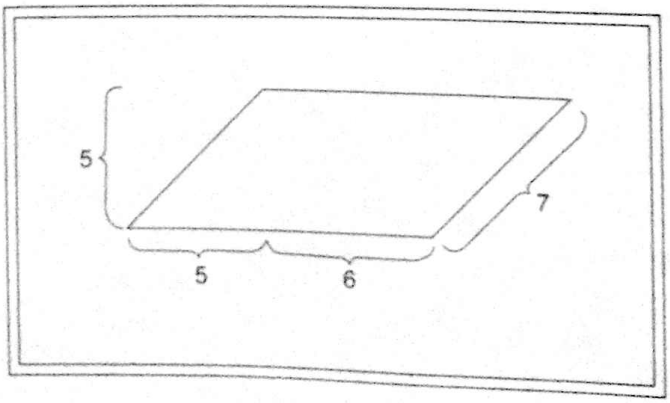
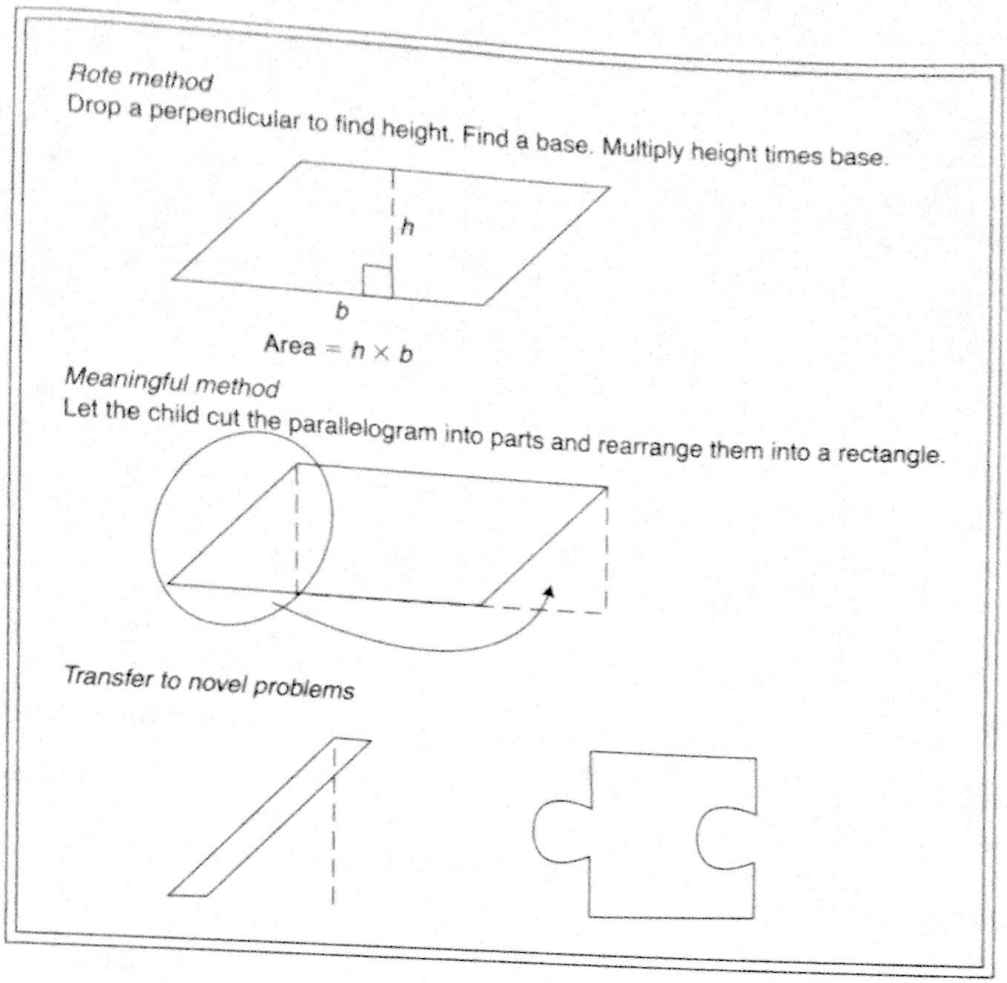


FIGURE 8–1 Find the area of a parallelogram

FIGURE 8–2 Rote and meaningful methods of instruction for the parallelogram problem



Wertheimer and other Gestalt psychologists (Katona, 1940; Kohler, 1925) distinguished between learning by rote and learning by understanding. Even though their work provides many interesting examples of the distinction, the cognitive theory underlying the distinction was not well spelled out. In this chapter, therefore, we investigate several well-known attempts to provide meaningful methods of instruction: concrete methods, discovery methods, and inductive methods. Each represents a form of guided exploration in which a learner is asked to solve a problem and is given some support along the way—including relating the problem to concrete objects (concrete methods), giving hints to keep the learner on track (discovery-oriented methods), and relating the task to something the learner already knows (inductive methods).

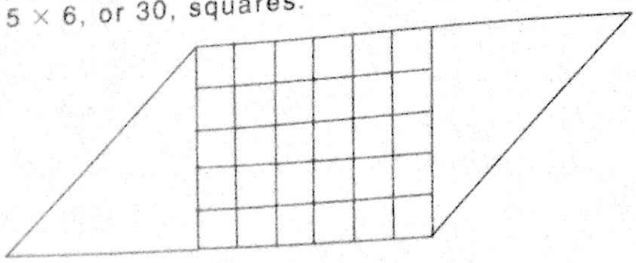
Concrete Methods

EXAMPLE OF A CONCRETE METHOD

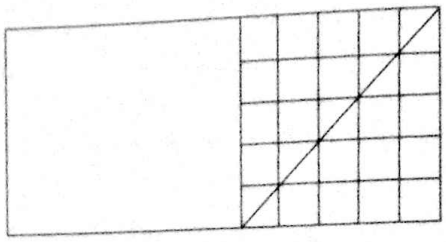
One way to make an idea more meaningful is to make it more concrete. For example, in the parallelogram problem, the teacher may make the concept of area more concrete by using 1 × 1-inch squares. Figure 8–3 shows how using 1 × 1 squares can give a concrete

FIGURE 8–3 How many unit squares are needed to cover the parallelogram?

If we put 131 unit squares over part of the parallelogram, we need 5 × 6, or 30, squares.



If we move the triangle from one end to the other and cover the triangles with 1 × 1 unit squares, we need 5 × 5, or 25, squares.



The total number of squares needed to cover the parallelogram is 11 × 5, or 55.

Source: Reprinted with the permission of Cambridge University Press.

way of representing area. These materials are called *concrete manipulatives* because the student can physically move and rearrange them.

THEORY: MAPPING CONCEPTS TO CONCRETE MODELS

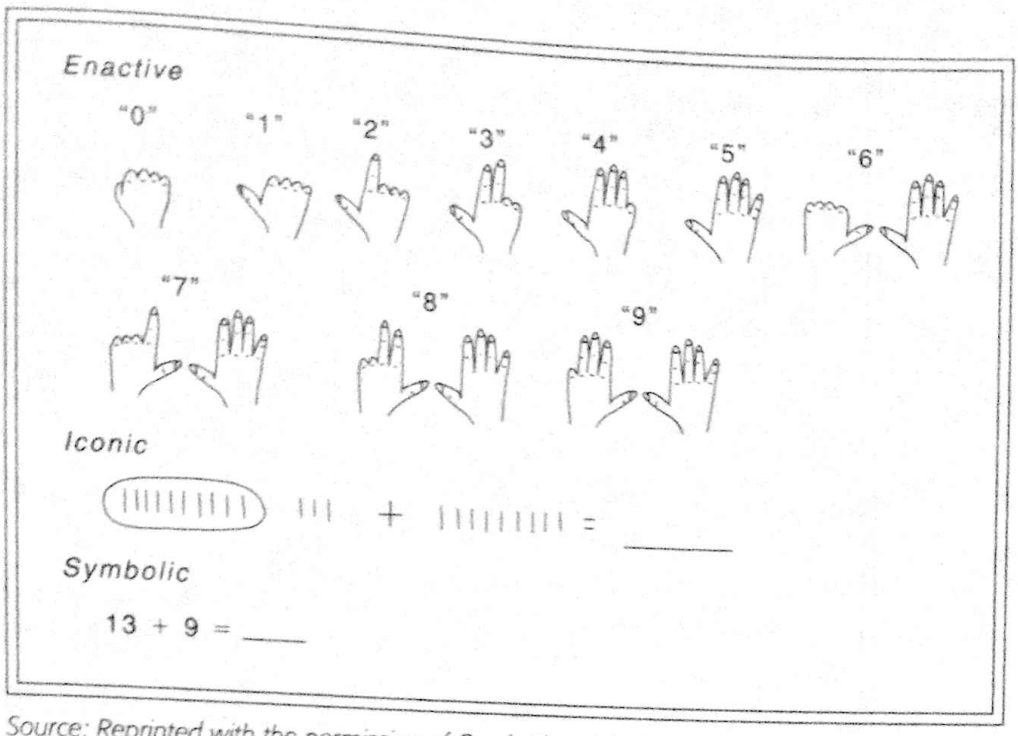
Why does a concrete representation of the material to be learned influence learning? One explanation comes from Bruner's (1964) theory of cognitive development. According to Bruner, children develop modes of representing information in the following order:

Enactive mode—using actions to represent information, such as tying a shoe.
Iconic mode—using visualization to represent information, such as thinking of a friend's face.
Symbolic mode—using language or other symbols to represent information, such as knowing that the area of a circle equals pi times the square of the radius.

In learning a new skill, such as arithmetic, several modes of representation may be involved, as shown in Figure 8–4. The enactive mode involves the physical actions of counting aloud with fingers; the iconic mode involves visualizing bundles of sticks that can be grouped by 10s; the symbolic mode involves numerals.

The development of understanding must progress through the same stages as representation in intellectual development: understanding first by doing, then by visualizing, and eventually by symbolic representation. Bruner and Kenney (1966) state this idea as follows: "We would suggest that learning mathematics may be viewed as a microcosm of

FIGURE 8–4
Three ways of representing a problem



Source: Reprinted with the permission of Cambridge University Press.

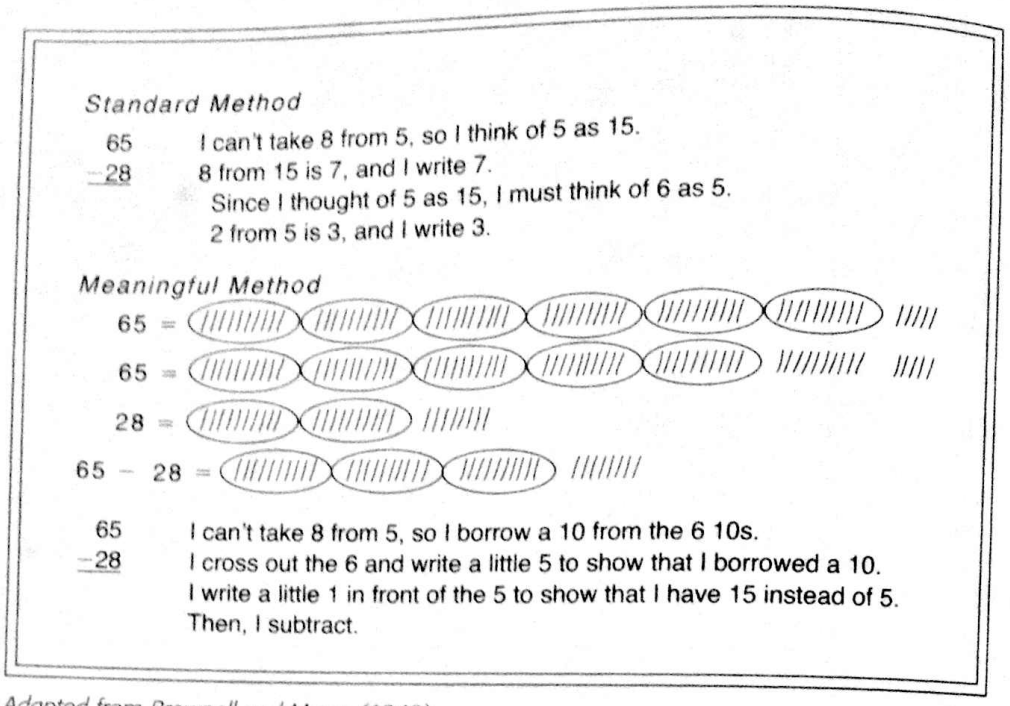
intellectual development. It begins with instrumental activity, a kind of definition of things by doing." Eventually, mathematical operations "become represented . . . in the form of . . . images," and finally, "with the help of symbolic notation, the learner comes to grasp the formal or abstract properties of the things he is dealing with" (p. 436). According to this view, understanding progresses from the level of active manipulation of objects and images and eventually leads to symbolic representation. Therefore, instruction that begins with formal symbolic representations without first allowing the learner to develop an enactive or iconic representation will lead to rote learning. Concrete manipulatives may be useful in connecting one mode of representation to another.

RESEARCH AND DEVELOPMENT: CONCRETE MANIPULATIVES IN MATHEMATICS

Bundles of Sticks Brownell (1935) was one of the first to demonstrate the important pedagogic role of concrete analogies in school learning. For example, Brownell suggested using manipulatives such as bundles of little sticks to concretize the subtraction algorithm. Suppose you wanted to teach children to subtract two-digit numbers such as $65 - 28 =$ _____ . One method of instruction would be to drill the student on the subtraction procedure as shown in the top of Figure 8–5. An alternative, which Brownell called the *meaningful method*, is to show how the problem can be represented as bundles of sticks, as shown in the bottom of Figure 8–5. In this system, place value can be represented by tying sticks together into bundles of 10 each.

In a careful research study, Brownell and Moser (1949) taught two groups of third graders, one by the standard method and the other by the meaningful method. On subsequent tests, both groups of children were able to solve two-digit subtraction problems like those given during instruction; however, the children who learned with the concrete analogy performed better than the standard group in learning to solve different kinds of problems. Apparently,

FIGURE 8–5 How to make arithmetic more concrete



Adapted from Brownell and Moser (1949).

the advantage of meaningful learning comes when the child is asked to transfer to new situations. According to Brownell (1935), a student needs to relate each piece of information together within a meaningful framework: "One needs a fund of meanings, not a myriad of automatic responses" (p. 10). Brownell noted that drill is appropriate only after "ideas and processes already understood are to be practiced to increase efficiency" (p. 19).

Let's suppose that concrete manipulatives such as bundles of sticks can help children understand the concepts of number, place value, sets, and operations on sets. Is it better to avoid teaching computational procedures until students first understand the concepts, or is it better to present the computational procedures first and then show how they relate to concrete manipulatives? As Resnick and Ford (1981) pointed out, "This is the sort of question that research has not yet answered" (p. 110). More recently, English (1997) has shown how concrete manipulatives—such as using bars and blocks to represent a two-digit number—rely on a form of analogical reasoning. For example, 43 can be represented as four 10-unit bars and 3 single-unit blocks. In thinking by analogy, when you are presented with an unfamiliar problem such as $43 - 27 =$ _____ (target problem), you can think of a familiar representation of the problem such as representing 43 as a collection of bars and blocks from which 27 must be taken (base problem). Then you can think through your answer using the base problem—that is, taking away two 10-unit bars, opening one 10-unit bar into 10 single-unit blocks and taking away 7, and then counting the remaining as one 10-unit bar and six 1-unit blocks. Then you can convert your answer from the base problem into the form of the target problem, $43 - 27 = 16$. Yet, like earlier work on concrete manipulatives, English (1997) notes: "Research on analogical reasoning in mathematical problem solving by elementary school children is in its infancy" (p. 199).

Montessori Materials Montessori (1964; Lillard, 2005) developed concrete materials that can be used to teach the structure underlying arithmetic. For example, Figure 8–6 shows some Montessori materials that can be used to teach the concept of place value.

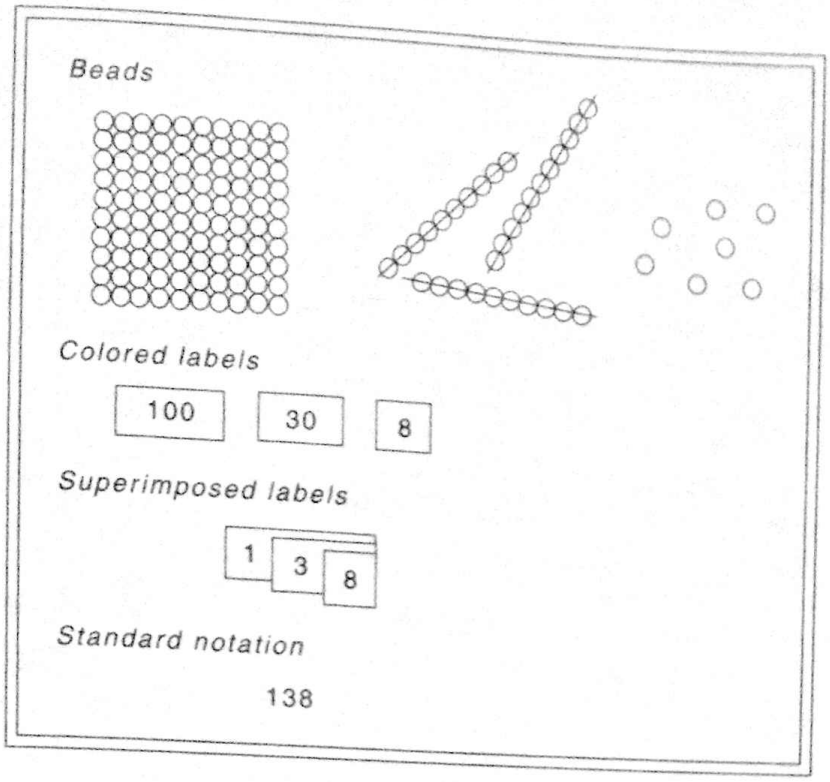


FIGURE 8–6
Some Montessori materials for numbers

These materials allow the child to progress from representing numbers as beads (with units, tens, and hundreds) to expanded notation using the colored labels, to standard notation using superimposed labels. To teach computational algorithms, the Montessori materials include wooden squares with 1 printed in green, 10 printed in blue, or 100 printed in red. A problem can be translated from standard notation into colored labels, such as shown in Figure 8–6. Then a child can learn the procedure of carrying as trading in 10 green unit squares for one blue 10-square, or trading 10 blue 10-squares for one red 100-square. Once the child is proficient at such exchanges, the symbolic notation for carrying can be introduced. For example, the 1 written at the top of the 10s column in the standard algorithm corresponds to exchanging 10 units for one 10 in colored squares.

Dienes Blocks Another set of concrete materials was developed by Dienes (1960, 1967). For example, place value and computation can be represented using multibase arithmetic blocks (MAB), also known as Dienes blocks, such as shown in Figure 8–7. The blocks come in units that are about 1 cubic centimeter; units can be snapped together into lines of 10, called longs; longs can be attached to form 10×10 squares, called flats; flats can be piled together to form $10 \times 10 \times 10$ cubes, called blocks. Bruner and Kenney (1966) showed how materials adapted from Dienes blocks can be used to teach the underlying structure of factoring quadratic equations. Figure 8–8 shows that materials consist of units (i.e., 1 by 1 blocks), longs (i.e., blocks that are 1 by X), and flats (i.e., blocks that are X by X). To make a square that is $(X + 1)$ by $(X + 1)$, you need one flat, two longs, and a unit. To make a square that is $(X + 2)$ by $(X + 2)$ you need one flat, four longs, and four units. To make a square that is $(X + 3)$ by $(X + 3)$ you need one flat, six longs, and nine units, and so on. Once the child can represent squares using blocks such as the $(X + 1)$ square, the formal notation can be given such as $(X + 1)^2 = X^2 + 2X + 1$. Bruner and Kenney suggest that instruction should begin by giving children a chance to actively manipulate actual objects and eventually progress towards the symbolic representation of the problem.

FIGURE 8–7
Dienes blocks for numbers

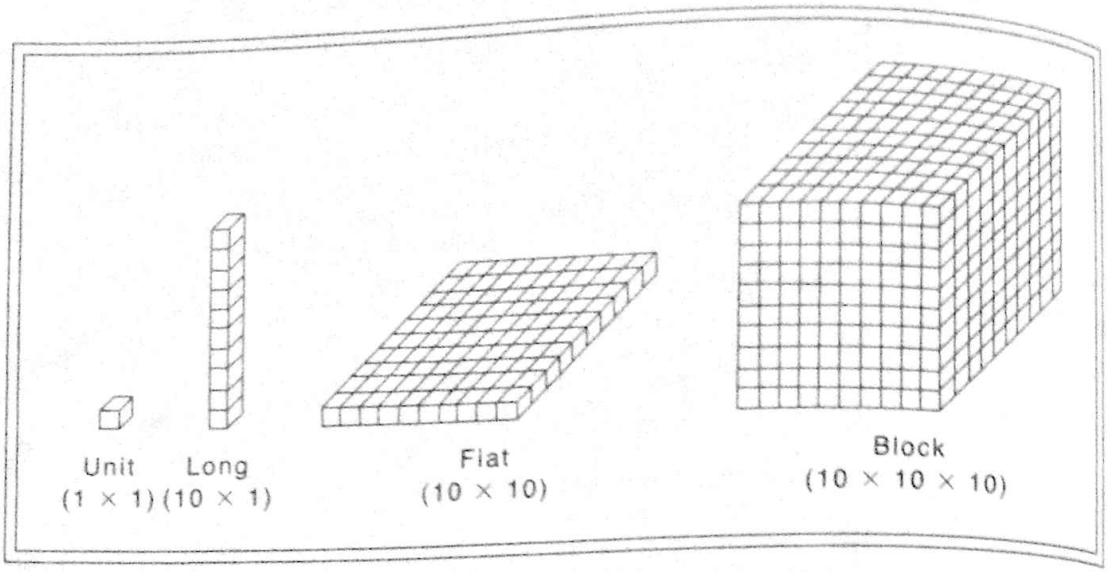
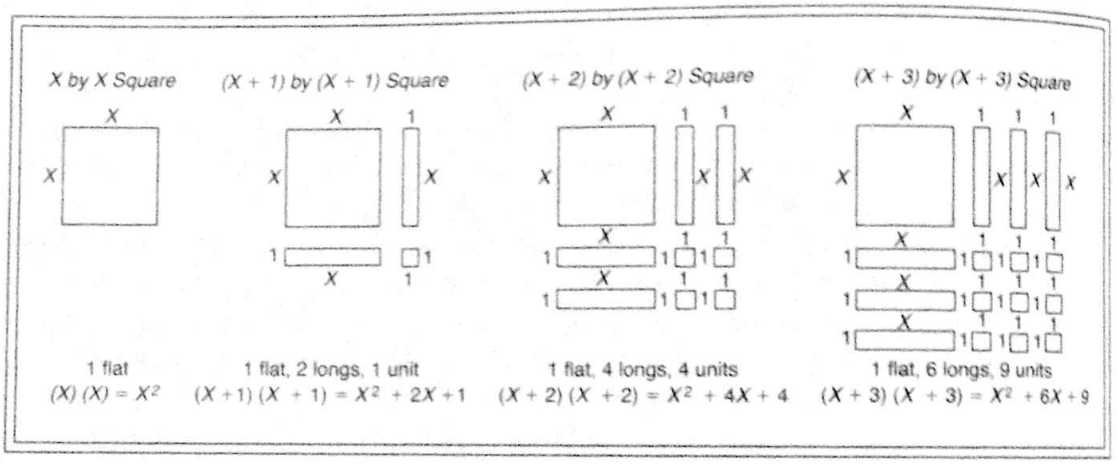


FIGURE 8–8
Using modified Dienes blocks to teach quadratic factoring



Source: Adapted from Bruner, J. S., & Kenney, H. (1966). Multiple ordering. In J. S. Bruner, R. R. Oliver, & P. M. Greenfield (Eds.), Studies in cognitive growth. *New York: Wiley.*

Based on his observations of children, Bruner (1960) argued for the importance of teaching the underlying structure of mathematics and science to children: "Grasping the structure of a subject is understanding it in a way that permits many other things to be related to it meaningfully. To learn structure, in short, is to learn how things are related. . . . The teaching and learning of structure, rather than simply the mastery of facts and techniques, is at the center of the classic problem of transfer" (pp. 3–11). For example, the child who learns through manipulating beads or blocks that 7 + 3 is the same as 6 + 4 or 2 + 8 has learned something about how arithmetic facts are related to one another.

IMPLICATIONS OF CONCRETE METHODS

So far we have sampled some of the commonly used manipulatives in mathematics instruction. Other manipulatives include attribute blocks, Cuisenaire rods, and geoboards. These materials are used in an attempt to present the underlying structures of mathematics

in a simple and concrete way to children. During the 1960s, mathematics curricula were reformed to emphasize this structure-oriented approach and to deemphasize drill and practice. However, in a review of manipulatives in mathematics instruction, Resnick and Ford (1981) pointed out that there has been very little research to identify the psychologically important structures that underlie mathematics: "The structure-oriented methods and materials have not been adequately validated by research, and we know little from school practice about the effects of the curriculum reforms upon the quality of children's mathematical learning" (p. 126). In another review of concrete manipulative, Hiebert and Carpenter (1992) concluded: "Despite the intuitive appeal of using materials, investigations of the effectiveness of concrete materials in classrooms have yielded mixed results" (p. 70). In a more recent review of research on Montessori materials for teaching mathematics, Lillard (2005) concluded that the question of whether children learn better with concrete manipulatives "is ripe for empirical research" (p. 57) and that "more research should be done to investigate which types of manipulatives are beneficial to children learning mathematical concepts" (pp. 69–70). According to these authors, the effectiveness of concrete materials can be enhanced when students have opportunities to reflect on the underlying principles, perhaps through discussions with peers. In sum, concrete methods must take into account the way that the learner tries to make sense of the manipulatives; once a learner understands a concept, drill and practice may be needed to ensure increased efficiency.

COMPUTER APPLICATIONS: COMPUTER SIMULATIONS IN MATHEMATICS

More recently, advances in educational computing technology have made it possible to allow students to interact with computer simulations of real-world objects and events. These computer simulations—called *microworlds*—allow learners to interact with and think about concrete representations of otherwise abstract ideas. Educational computer simulations are not magical toys that guarantee meaningful learning, but when used wisely, they offer a way of bringing the power of concrete manipulatives to a whole new level.

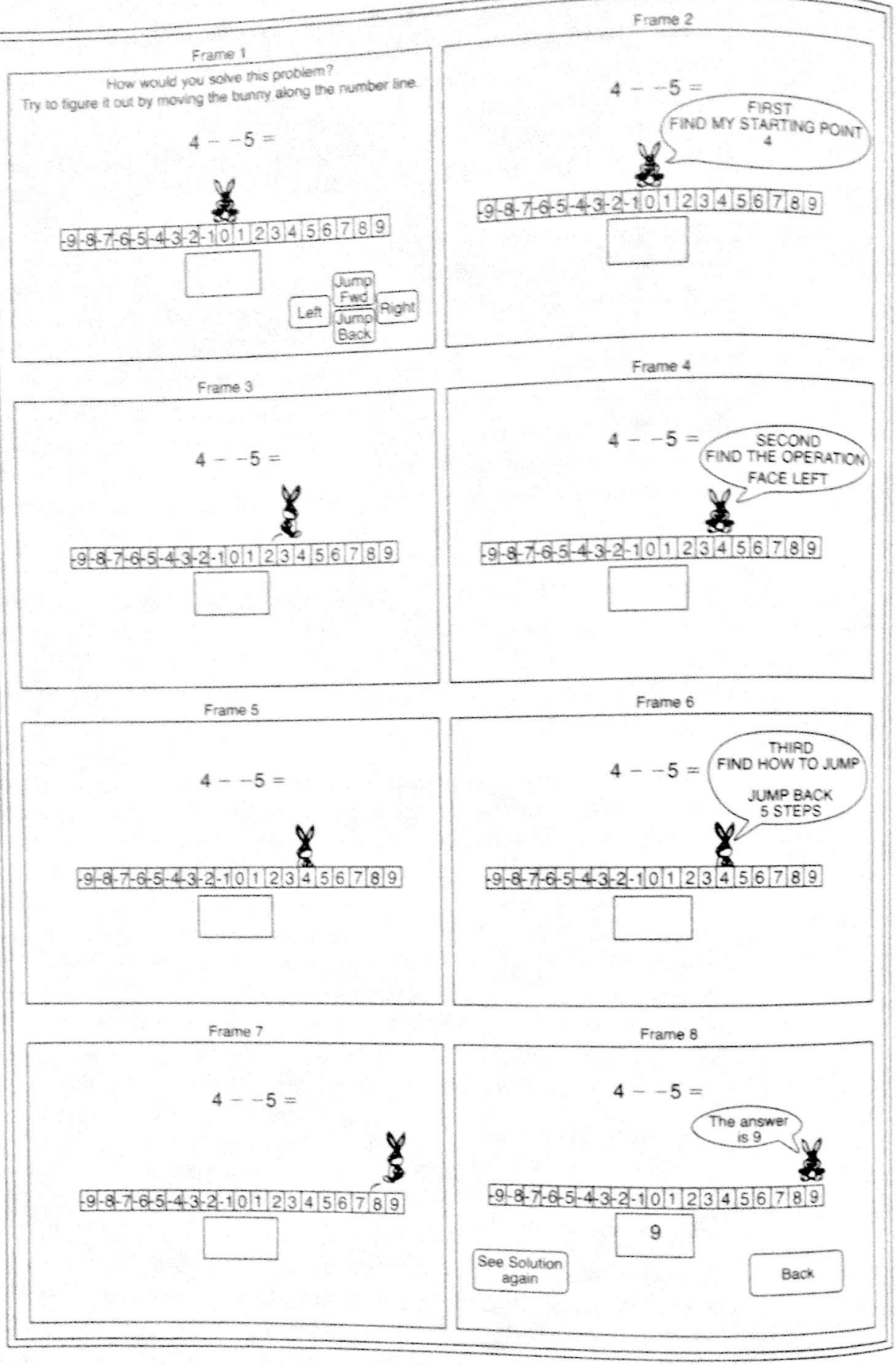
In the top-left frame of Figure 8–9, you see a number line running from -9 to $+9$ with a bunny sitting on the space for 0. Suppose you wanted to have the bunny enact the procedure for the problem, $4 - -5 =$ _____, which you can read as "4 minus negative 5 equals what?" Write down the steps you would ask the bunny to carry out. If you are like most successful problem solvers, your list would consist of the following steps:

1. The bunny moves to 4.
2. The bunny faces left.
3. The bunny jumps backward 5 steps.
4. The bunny lands on 9, which is the answer.

In this example, the symbolic number sentence $4 - -5 = 9$ can be translated into a concrete situation involving a bunny moving along a number line. This concrete representation allows you to make a conceptual distinction between a minus sign (in which case the bunny faces left) and a negative number such as -5 (in which case the bunny jumps backward five steps).

Does practice in relating symbols to concrete situations help students understand how to add and subtract signed numbers? Moreno and Mayer (1999) tested this issue with

FIGURE 8–9
Selected frames from the bunny simulation



Source: From Moreno, R., & Mayer, R. E. (1999). Multimedia-supported metaphors for meaning making in mathematics. Cognition and Instruction, *17, 215–248. Copyright 1999 Lawrence Erlbaum Associates. Reprinted by permission.*

high-achieving sixth-grade students who had no previous experience with adding and subtracting signed numbers. All students received practice on 64 problems spaced across four different sessions. For each problem presented on a computer screen (such as $4 - -5 =$ _____), the control group typed in an answer and then the correct answer appeared on the screen. For each problem presented on the computer screen, the experimental group used a joystick to move the bunny along the number line and typed in an answer based on where the bunny ended up; then the computer presented an annotated animation of the bunny correctly moving along the number line, ending with the final answer (such as shown in Figure 8–9). Thus, both groups solved the same problems, but the experimental group also worked on relating the problem to a concrete context of a bunny moving along a number line. In this study involving high-achieving students, the experimental group showed large pretest-to-posttest gains in correctly solving problems (i.e., up 25%), whereas the control group did not (i.e., up 6%).

What was learned by students in the control and experimental groups? Some students—particularly those in the control group—may tend to form specific S-R associations, such as memorizing that when the question is "$4 - -5 =$ _____" the answer is "9." This kind of very specific learning would not lead to much transfer on a posttest. In contrast, other students—particularly those in the experimental group—may have created more general principles for understanding the problems. Many students enter the learning situation with what can be called a *negative-bias bug*: a misconception in which students fail to recognize that the "−" symbol sometimes is a minus sign and sometimes is a negative sign. Whenever a problem contains two "−" symbols, the students interpret this to mean that they should find the absolute difference between the numbers and then either place a negative sign in front or not (e.g., $-3 - 1 = 2$ or -2; $7 - -2 = 5$ or -5; $-8 + -1 = 7$ or -7). If learning is more general, students may learn to make a distinction between the minus sign (which means to subtract) and the negative sign (which means that the number is negative), thus eliminating the negative-bias bug and increasing the potential for transfer. In Moreno and Mayer's (1999) study, the control group displayed the negative-bias bug 15% of the time on the pretest and 15% of the time on the posttest, suggesting little progress in building a general distinction between minus and negative. The experimental group displayed the negative-bias bug 23% of the time on the pretest and 9% of the time on the posttest, demonstrating a large improvement in building a general principle. Apparently, building a general principle (such as the distinction between minus signs and negative signs) is a major accomplishment that may be an important key to transfer. Somewhat similar results were obtained by Schwartz, Nathan, and Resnick (1996) in which students learned to represent adding and subtracting signed numbers as the building and comparing of trains of various lengths along a number line.

Let's now move from arithmetic to algebra. Consider the following problem:

> A huge ant is terrorizing San Francisco. It travels east toward Detroit, which is twenty four hundred miles away, at four hundred miles per hour. The Army learns of this one hour later and sends a helicopter west from Detroit at six hundred miles per hour to intercept the ant. If the ant left at 2 p.m., what time will the helicopter and ant collide (ignoring any time changes)? (Nathan, Kintsch, & Young, 1992, p. 349)

If you are like most of the college students in the study by Nathan and colleagues, you had some difficulty with this problem. Overall, less than half the students correctly solved problems like this one on a pretest.

To help students improve, you could provide a review of algebra and exposure to three time-rate-distance word problems like the ant problem. However, students who received this kind of instruction (control group) showed little pretest-to-posttest gain in correctly solving other word problems (i.e., up 6%). In contrast, you could help students learn how to relate the problem to a concrete situation by letting them use a computer program (called ANIMATE) designed to allow learners to animate the problem. For example, for the ant problem, students could select an image of an ant from a menu and place it on the left side of the screen and select an image of a helicopter and place it on the right side of the screen. Then, they could create a system of distance-rate-time equations and fill in the relevant numbers (such as typing in rate as 400 for the ant and rate as 600 for the helicopter). They could run the animation at any time and revise it based on what they saw. As you can see, solving problems with the ANIMATE program helps students understand the connection between words in the problem and a concrete visual representation. Students who received practice in using the ANIMATE program to work on three time-rate-distance problems (experimental group) showed large pretest-to-posttest gains in correctly solving other word problems (i.e., up more than 300%).

What is learned by the control and experimental groups? According to Nathan et al. (1992), students in the experimental group are more likely than students in the control group to learn general strategies for how to represent word problems—a skill they refer to as being able to build a situation model of the problem. Consistent with this analysis, they found that errors in representing the problem decreased only slightly from the pretest to the posttest for the control group (i.e., down 10%), whereas such errors dropped greatly for the experimental group (i.e., down 69%). When it comes to meaningful learning, it appears that being able to build situation models using pictures of objects is a general skill that can enable transfer.

My goal is not to review the volumes of research on computer-based tutoring but rather to examine a few exemplary research studies on the use of computerized concrete manipulatives. As you can see, there is encouraging evidence that concretizing an abstract concept or procedure can help students understand and learn in ways that promote transfer. Further, computer-based visualizations seem to work because they help students build general principles or strategies that apply across situations—such as recognizing the difference between minus and negative or being able to construct situation models. In a pioneering set of studies, White and colleagues (Schwartz & White, 2005; White, 1993; White & Frederiksen, 1998) found similar positive effects on understanding when high school students learn and discuss physics principles within the context of a visually concrete computer game.

In computer-based microworlds, the student is able to relate general principles to more familiar objects by manipulating simulated objects on the computer screen. The research results are promising because they show that *concrete manipulatives* can be moved productively to the computer screen. The number of commercially available educational computer simulations is increasing, and research such as highlighted in this section suggests that some simulations may be useful. However, not all computer games are useful instructional tools. What makes a "good" game? The foregoing examples suggest that good games are based on appropriate design principles, are presented at a level that is appropriate for students, and focus on teaching generalizable skills that are fundamental parts of the academic program. Clearly, encouraging students to interact with, think about, and talk about concrete representations of otherwise abstract ideas provides a potentially useful path to meaningful learning.

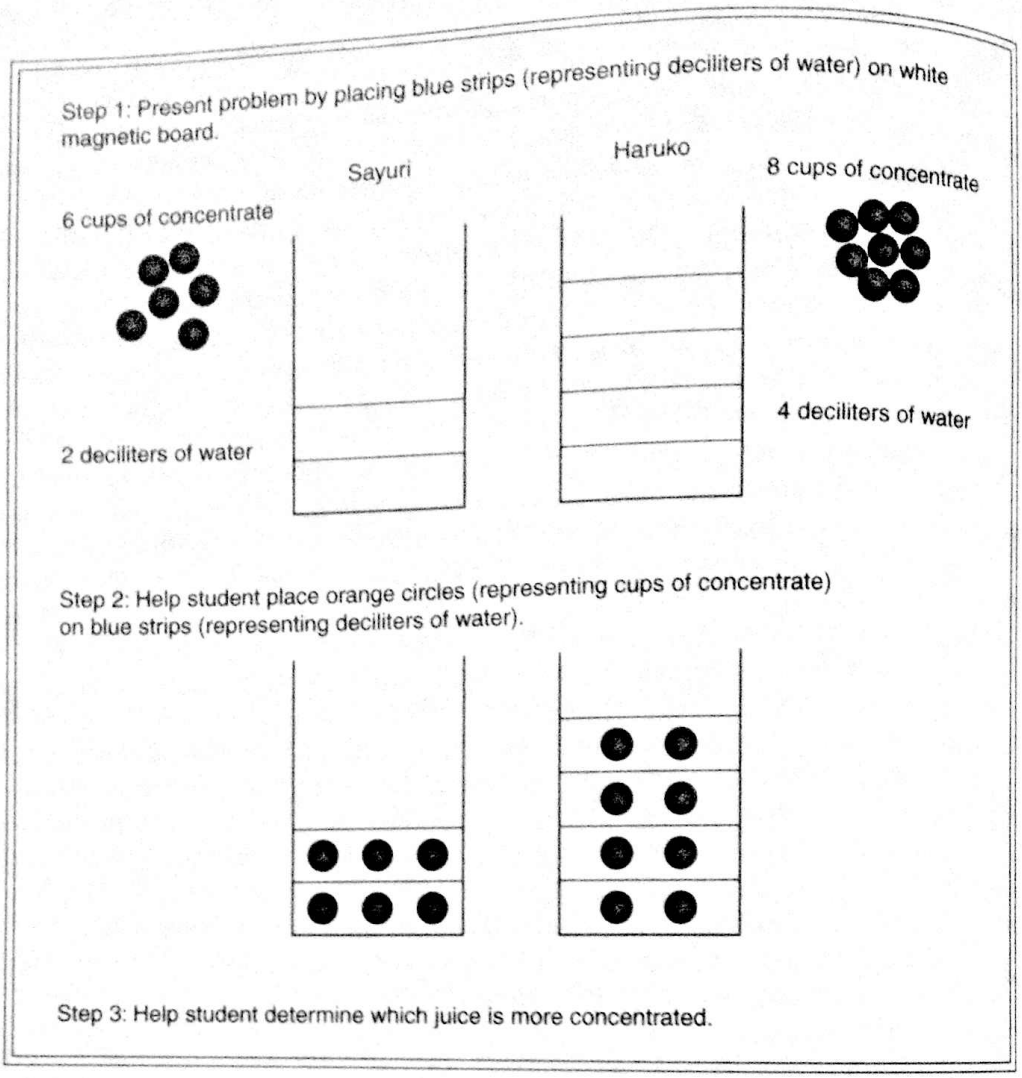
What can go wrong with using manipulatives—either in computer games or as concrete objects? If you intend to use concrete manipulatives, an important consideration is whether the manipulatives will create so much cognitive load in the learner that they interfere with learning. For example, as described previously, Moreno and Mayer (1999) found the use of an on-screen concrete manipulative—a bunny moving along a number line—helped skilled students learn about addition and subtraction of signed numbers in a computer-based educational game, but the manipulative detracted from learning of less skilled learners. Similarly, Campbell and Mayer (2004) reported that concrete manipulatives helped high-skilled third graders learn to solve equivalent fraction problems but hurt the learning of less skilled third graders. To understand why manipulatives might hurt less able learners and help more able learners, consider the role of cognitive load in learning. The more skilled learners have automated their component skills (such as how to count or how to retrieve basic number facts or how to use the manipulatives), so they are able to process the manipulative without overloading their working memory. In contrast, the less skilled learners must devote attention to carrying out needed component skills as well as to learning to use the manipulatives. For example, Uttal, Liu, and DeLoache (1999) point out that "manipulatives are also symbols" because "teachers intend them to stand for . . . a concept or written symbol" (p. 37). Thus, concrete manipulatives can be a double-edged sword when students must struggle to determine what the manipulatives represent.

Further evidence concerning the possible limitations of concrete manipulatives comes from carefully observing how students use them. Based on interviews with 13-year-old students who learned to solve linear equations using concrete manipulatives, Boulton-Lewis et al. (1997) concluded that most students did not successfully use the maniplatives because of "increased processing load caused by concrete representations." Boulton-Lewis and colleagues concluded that students did not use the concrete manipulatives because of the "processing load associated with processing concrete representations" (p. 395), suggesting that manipulatives would be more beneficial for students who had more practice using them. Thus, when students have not yet automated basic skills, "use of concrete representations by students was counterproductive" (p. 395).

Further evidence for individual differences in the effectiveness of concrete manulatives comes from a study by Fujimura (2001) on helping fourth graders in Japan solve concentration-comparison problems. For example, consider the following problem: "Yukio and Masachi each make orange juice by mixing water and concentrate. Yukio has 2 deciliters of water and 8 cups of concentrate. Masashi has 3 deciliters of water and 9 cups of concentrate. Which juice is more concentrated, Yukio's or Masashi's, or is the concentration the same? Why do you think so?" On a pretest with four problems, students solved about 22% correctly.

Some students were given a brief training using concrete manipulatives, consisting of a magnetic board with blue strips to represent deciliters of water and orange dots to represent cups of orange concentrate. The experimenter explained the situation as she placed the names of two children on the board along with a certain amount of water for each, such as 2 deciliters of water for Sayuri and 4 deciliters of water for Haruko. Then the experimenter asked the child to arrange the orange dots (6 for Sayuri and 8 for Haruko) so that the "orange juice" would be "completely mixed" in the "water." Students were also asked to judge which juice was more concentrated based on the manipulatives. Teachers provided hints and guidance. Figure 8–10 shows the anticipated correct answer.

FIGURE 8–10
Using concrete manipulatives to represent a concentration-comparison problem



Adapted from Fujimura (2001).

On a posttest, students were given four new problems, without the concrete manipulatives. Students who received the concrete manipulative training showed a large improvement on the posttest—now solving 54% of the problems correctly—whereas students who did not receive the concrete manipulatives did not show much improvement—now solving 26% of the problems correctly. Clearly, exposure to concrete manipulatives helped students learn to solve concentration-comparison problems. As you can see, the training was done in a way that minimized cognitive load. In subsequent studies, Fujimura found that the concrete manipulative training greatly improved the performance of students who had strong prerequitisite skills (from 25% on the pretest to 77% on the posttest) but not for students who lacked prerequisite skills (from 12% on the pretest to 25% on the posttest). Consistent with other studies, concrete manipulatives helped higher skill students more than lower skill students.

Two major implications of research on concrete manipulatives are that the prerequisite knowledge of the learner can play an important role, and the cognitive load imposed by manipulatives can play an important role. In short, manipulatives work best when the

student has mastered component skills and when manipulatives are used in ways that minimize extraneous cognitive processing. I am not suggesting that concrete manipulatives be banned, but rather that you use them in ways that are sensitive to the cognitive load imposed on each learner. For example, most Montessori materials are designed to be very simple to use, although there is a need for empirical evidence concerning their effectiveness.

Discovery Methods

EXAMPLE OF DISCOVERY METHODS

Let's return to the parallelogram problem shown in Figure 8–1. What else can be done to make the rule for finding area more understandable? One suggestion is to encourage the learner to try to solve problems actively before being presented with the rule to be learned. For example, you could give the student a paper parallelogram and a scissors and ask the learner to cut up the paper and rearrange it as a rectangle. In this case, we want the learner to cut the triangle from one end and place it on the other end (as shown in the middle frame of Figure 8–2). Then, the rule can be given.

THEORY: THE JOY OF DISCOVERY

Bruner (1961) helped instigate modern interest in discovery learning in his famous essay "The Act of Discovery." Bruner's paper distinguished between two modes of instruction: expository mode, in which the teacher controls what is presented and the student listens, and hypothetical mode, in which the student has some control over the pace and content of instruction and may take on an "as if" attitude. The hypothetical mode allows the learner to discover new rules and ideas rather than simply memorize rules and ideas that the teacher presents. According to Bruner, the discovery of rules results in better learning—because the learner has organized the material in a useful way—and results in the student's becoming a better learner and problem solver in general because the student gets practice in processing information.

RESEARCH AND DEVELOPMENT: DISCOVERY OF RULES

Although Bruner is an eloquent proponent of the discovery method and although his suggestions were implemented in some curricular projects (Davis, 1973), you might wonder whether there is any empirical evidence that discovery enhances learning. During the 1960s a flurry of research was concerned with the question of how much guidance a teacher should provide (Shulman & Keisler, 1966). Although the researchers often used terms in different ways, we can define three basic levels of guidance in instruction:

1. *Pure discovery.* The student receives representative problems to solve with minimal teacher guidance.
2. *Guided discovery.* The student receives problems to solve, but the teacher provides hints and directions about how to solve the problem to keep the student on track.
3. *Expository.* The final answer or rule is presented to the student.

Let's look at how these methods can be used to help students learn how to solve logical reasoning problems and mathematical reasoning problems.

Logical Reasoning An early study by Craig (1956) was the forerunner of more recent method-of-instruction studies. Students were given training in "finding the word that doesn't belong" in sets of five words. For example, given

CYCLE SELDOM SAWDUST SAUSAGE CELLAR

the appropriate answer is to mark CYCLE because it does not share the same initial sound (i.e., "sigh") as any of the other words. Items were organized in sets of four, all having the same relational rule (e.g., initial sound), and each training booklet contained several such types of rules.

Two instructional methods were used: A guided discovery group was told the relation (e.g., "look for initial sound") at the beginning of each set of four items but was not told the answer per se; the other group, which could be called "pure discovery," was not given any hints. Results indicated that the group given some guidance learned more efficiently, retained more, and transferred just as well as the pure discovery group. This study calls into doubt the emphasis on extreme classroom freedom and independence; some learners simply may not be able to discover the appropriate concepts and rules without some direction from the teacher.

Kittel (1957) reported a study using material similar to Craig's but that involved all three levels of guidance—pure discovery, guided discovery, and expository. The training, like Craig's, involved giving the learner a set of five words, such as

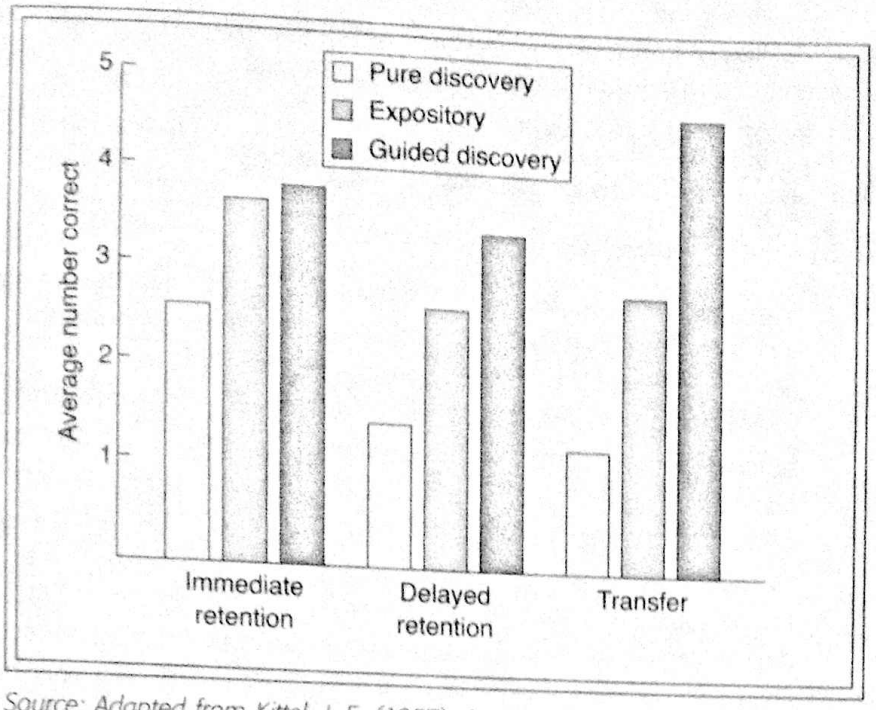
GONE START GO STOP COME

and asking the learner to mark the word that doesn't belong. In this example, the relational principle is "form two pairs of opposites"; hence, the correct answer is "GONE." In the training booklets, each set of three items had the same principle, and there were 15 such principles in all.

Some subjects were not given any direction (pure discovery); some subjects were told the principle (e.g., "form two pairs of opposites") for each set of problems but were not given the answer (guided discovery); some subjects were told both the principle and the correct answer for each problem (expository). Figure 8–11 summarizes some of the major results of the study. As Figure 8–11 shows, the pure discovery group performed worse than the other two groups on immediate retention, suggesting that pure discovery resulted in less initial learning. On tests of transfer and long-term retention, the guided discovery group outperformed both the pure discovery group and the expository group. Apparently, the pure discovery group did not discover many of the principles during learning; in addition, whereas the guided discovery and expository groups seem to have learned equal amounts during initial learning, the extra processing and thinking during learning led the guided discovery group to retain the information and transfer the information better than the expository group.

Mathematical Reasoning The foregoing results suggest that a major drawback of pure discovery methods is that some students may fail to discover the underlying principle. To overcome this problem, Gagné and Brown (1961) conducted a study in which students learned to solve series sums and derive formulas using three different instructional methods. For example, students learned how to compute the sum of "1,3,5,7,9 . . ." and

FIGURE 8–11
How much guidance should be given during learning?



Source: Adapted from Kittel, J. E. (1957). An experimental study of the effect of external direction during learning on transfer and retention of principles. Journal of Educational Psychology, 48, *391–405.*

to write a formula for the series. In the pure discovery method students were given problems to solve; however, if they were unable to solve the problem, hints were provided until the correct principle was found. Thus, the pure discovery method was modified to make it more like guided discovery (i.e., to ensure that the student actually learned). In the guided discovery method, problems were given along with a systematic succession of questions to aid the student, thus, providing more guidance concerning how to solve the problem. The expository group was given problems along with the solution formula already worked out. All students had to continue working until they were able to master four separate series; thus, all students were forced to learn equal amounts.

Table 8–1 shows the amount of time and number of errors in learning under the three methods of instruction and the amount of time and number of errors on a subsequent transfer test for the three treatment groups. As Table 8–1 shows, the guided discovery group took the longest amount of time to learn but performed best on the transfer test. The

TABLE 8–1
Effects of discovery methods on learning and transfer

	Pure Discovery		Guided Discovery		Expository	
	Time (min)	**Errors**	**Time (min)**	**Errors**	**Time (min)**	**Errors**
Learning phase	28	6	46	17	41	9
Transfer phase	20	2	17	1	27	6

Source: Adapted from Gagné, R. M., & Brown, L. T. (1961). Some factors in the programming of conceptual learning. Journal of Experimental Psychology, 62, *313–321.*

pure discovery group also performed well on transfer, presumably because of the procedure ensured that initial learning actually occurred.

IMPLICATIONS OF DISCOVERY METHODS

Our review of research on discovery identifies the following patterns:

1. Pure discovery methods often require excessive amounts of learning time, result in low levels of initial learning, and result in inferior performance on transfer and long-term retention. However, when the principle to be learned is obvious or when a strict criterion of initial learning is enforced, pure discovery learners are likely to behave like guided discovery learners. Apparently, pure discovery encourages learners to get cognitively involved (Anastasiow, Sibley, Leonhardt, & Borish, 1970) but fails to ensure that they will come into contact with the rule or principle to be learned.
2. Guided discovery may require more or less time than expository instruction, depending on the task, but tends to result in better long-term retention and transfer than expository instruction. Apparently, guided discovery both encourages learners to search actively for how to apply rules and makes sure that the learner comes into contact with the rule to be learned.
3. Expository instruction may sometimes result in less learning time than other methods and generally results in equivalent levels of initial learning as compared to guided discovery. However, if the goal of instruction is long-term retention and transfer, expository methods seem inferior to guided discovery. Apparently, in some situations, expository instruction does not encourage the learner to actively think about the rule but does ensure that the rule is learned.

When the goal of instruction is long-term retention and transfer of learned principles, the teacher needs to use enough guidance so that the student finds the to-be-learned principle but not so much guidance that the student is discouraged from working actively on understanding how the principle can be applied. The learner's level of prior knowledge is likely to play an important role because students with low prior knowledge may need more guidance, whereas students with high prior knowledge may need less guidance.

COMPUTER APPLICATIONS: DISCOVERY IN COMPUTER PROGRAMMING

Although research in the 1960s tended to argue against the usefulness of pure discovery, the introduction of computers into schools during the 1980s brought renewed calls for discovery learning. For example, Papert (1980), in his influential book *Mindstorms*, argues eloquently for two important aspects of computing instruction: (1) hands-on discovery in which students are allowed to "learn without being taught" (i.e., students receive unstructured, hands-on computer experience that is not tied to any curricular demands), and (2) LOGO environment, in which students work in a powerful and responsive computer environment that is supposedly provided by the programming language LOGO.

To understand the LOGO environment, let's focus briefly on *turtle graphics*, generally the first aspect of LOGO to be learned by elementary school children. You begin with a "turtle," represented as a triangular cursor on a computer screen, as shown in the top of

FIGURE 8–12
Some LOGO commands

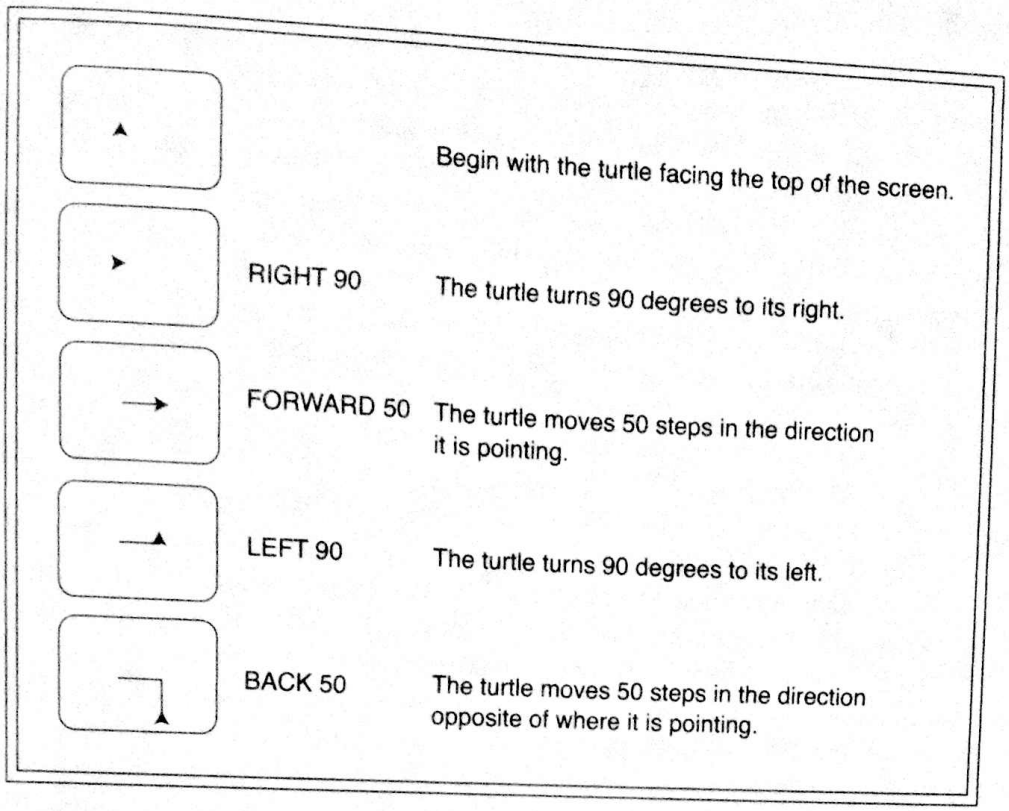


Figure 8–12. You can turn the turtle by issuing commands such as RIGHT _____ or LEFT _____ , where each command is followed by a number that indicates how many degrees the turtle will turn. For example, RIGHT 90 means turn the turtle 90 degrees clockwise from its current position and LEFT 90 means turn the turtle 90 degrees counterclockwise from its current position. You can move the turtle by issuing commands such as FORWARD _____ or BACK _____ , where each command is followed by a number indicating how many steps the turtle will take. Examples of these four commands are given in Figure 8–12.

Suppose you want to draw a square. What commands would you type into the keyboard? One way to correctly solve this problem would be to type

```
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
```

However, the following commands (using abbreviations) were produced by a child who was trying to draw a square (Papert, 1980):

```
FD 100
RT 100
FD 100
ERASE 1 (This undoes the previous command.)
```

```
RT 10
LT 10
LT 10
FD 100
RT 100
LT 10
RT 100
LT 10
FD 100
RT 40
FD 100
RT 90
FD 100
```

As you can see, the child had a difficult time turning the turtle at right angles, presumably because the child did not know that 90 degrees is a right angle. However, once you have developed a program to draw a square you can name and save the program, such as by naming it SQUARE. Then you can "call" this program at any time within another program simply by typing SQUARE as a command. For example, if you wanted to draw a house (i.e., a square with a triangle on top), you could use the SQUARE command within your program because drawing a square is a component in drawing a house. Then you could name that new program HOUSE and use it within a larger program, and so on.

According to Papert (1980), as children explore LOGO, they develop "powerful ideas" concerning how to solve problems procedurally—including how to modularize a program and call each subprogram when needed. What evidence is there to support Papert's demand for unrestricted, hands-on experience in a LOGO environment and his claim that learning LOGO will transfer to other problem-solving domains? Regrettably, the evidence that is available does not support the use of pure discovery as an effective instructional method. Real children in real classrooms generally have difficulty learning even the fundamentals of LOGO programming (Dalbey & Linn, 1985; Kurland & Pea, 1985; Mayer, 1988; Pea & Kurland, 1984; Perkins, 1985). For example, Kurland & Pea (1985) tested seven children, ages 11 to 12, who averaged more than 50 hours of hands-on LOGO programming learning under discovery conditions. Children were given programs and asked to predict the output. The children had little trouble with short, simple programs that involved only commands to move or turn the turtle. However, on transfer problems that involved using fundamental programming concepts, students performed poorly. Through in-depth interviews, Kurland and Pea found that the children had developed incorrect conceptions of how programs operated. Furthermore, even though "none of these sources of confusion will be intractable to instruction," pure-discovery students faced an "absence of instruction" (p. 242). Apparently, hands-on experience does not guarantee productive learning of LOGO, and these results suggest that "discovery needs to be mediated within an instructional context" (p. 242).

Similar difficulties have been observed in students learning BASIC, another "beginner's language" (Bayman & Mayer, 1983; Linn, 1985; Mayer, 1985). For example, in one study (Bayman & Mayer, 1983; Mayer, 1985), low-achieving college students learned BASIC either through hands-on experience only or through hands-on experience supplemented

with direct instruction in the basic programming concepts, such as how data are stored in memory. Students who learned with only hands-on experience exhibited many misconceptions of fundamental programming concepts, such as not knowing how data are stored in memory, not understanding where incoming data come from, and not understanding how the computer determines which command it will follow next. In contrast, as expected, students who were given direct instruction in how these concepts related to each command displayed many fewer misconceptions. On programming tests involving transfer, such as writing or interpreting complex programs, the group given added direct instruction performed better than the hands-on only group. Although hands-on experience was not effective for low-achieving students, complementary studies found that hands-on experience was effective for high-achieving students. Presumably, the high-achieving students came to the learning situation with appropriate knowledge they could use to interpret their programming experiences. These results show that hands-on experience does not always lead to meaningful learning of programming concepts, especially when students lack appropriate prerequisite knowledge.

In summary, there is very little evidence to support eloquent calls for pure discovery as the way of teaching programming to children (Mayer, 1988). Based on the present research, the most productive approach appears to be a mix of teacher-based instruction and student-based exploration. However, students need to be tested frequently to determine whether they are acquiring useful programming concepts. If they are not, direct instruction is warranted.

Guided discovery is helpful because it helps students reflect on their learning so they are more likely to build general principles and strategies that enable transfer. What are some general principles in LOGO? Two useful design principles are modularity (breaking a procedure into parts) and reusability (using the same subprocedure more than once). Regrettably, many students who learn LOGO through hands-on exploration fail to develop basic design principles such as modularity and reusability (Fay & Mayer, 1994).

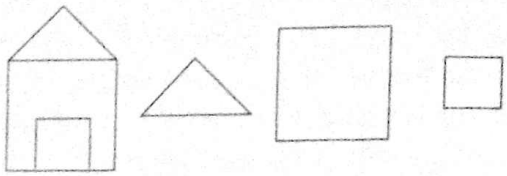
Suppose you wish to provide some guidance to students as they learn to program in LOGO. The first step could be to introduce them to the concepts of modularity and reusability before describing any programming command. For example, Figure 8–13 presents the concept of modularity in concrete, familiar terms without any computer commands. As you can see, a large task (drawing a house) is broken down into distinct subtasks (drawing a roof, frame, and door). To help them apply design principles to LOGO, students can be given some programs to write along with hints, such as shown in the house assignment in the top of Figure 8–14. After they finish their program, they receive feedback that shows step by step how to apply modularity and reusability to the house program, as shown in the bottom of Figure 8–14. The feedback is intended to foster reflection on how the design principles can be used in writing LOGO programs.

Fay and Mayer (1994) taught LOGO programming to computer-naive college students using a discovery or guided discovery method. Some students learned by being given a manual about LOGO commands and then engaging in hands-on writing of programs for programming assignments over four 1-hour sessions (discovery group). They were allowed to run their programs on computers and revise them if they did not work. Others received the same experience along with a bit more guidance about design principles, including receiving general pretraining on the design principles, hints on the program assignments to promote use of design principles, and feedback about how the programs related to design principles (guided discovery).

FIGURE 8–13
Excerpt from design instruction

Modularization

One way to simplify a list of instructions is to break down the list into smaller, more manageable units. Each unit would consist of a shorter set of instructions that would be easier to follow. First, you have to decide how to break a list into parts. With a drawing, one way to divide the list is by the parts of the drawing. In our house example, there are 3 different parts: the roof, the frame, and the door. Each part can be identified as a separate unit, as shown below.



House = Roof + Frame + Door

Second, take each part, one at a time, and write the instructions for that part.

To make the frame
Move forward 1 inch
Turn right 90 degrees
Move forward 1 inch
Turn right 90 degrees
Move forward 1 inch
Turn right 90 degrees
Move forward 1 inch
Turn right 90 degrees
End of frame

To make the door
Move forward 1/2 inch
Turn right 90 degrees
Move forward 1/2 inch
Turn right 90 degrees
Move forward 1/2 inch
Turn right 90 degrees
Move forward 1/2 inch
Turn right 90 degrees
End of door

To make the roof
Move forward 1 inch
Turn right 120 degrees
Move forward 1 inch
Turn right 120 degrees
Move forward 1 inch
Turn right 120 degrees
End of roof

Source: From Fay, A. L., & Mayer, R. E. (1994). Benefits of teaching design skills before teaching LOGO computer programming: Evidence for syntax-independent learning. Journal of Educational Computing Research, *11, 187–210. Copyright 1994 Baywood Publishing Company. Reprinted by permission.*

Does focused guidance affect learning? In writing programs during the learning phase of the experiment, the guided discovery group created modules and reused them often, whereas the discovery group rarely did. This evidence shows that the guided discovery students were better able to develop an understanding of general design principles. On a posttest involving new LOGO programming problems, the guided discovery group achieved a higher percentage correct than did the discovery group. This evidence indicates that learning the design principles helped students transfer their learning of LOGO to new problems. However, the groups did not differ on posttests that were not directly related to the design principles of modularity and reusability, such as tests of spatial reasoning. Taken together, the results are consistent with the theory of specific transfer of general principles. It appears that transfer was enhanced by guiding students in their learning of design principles that are directly related to writing LOGO programs. When students rely solely on free exploration—without any guidance—they learn to write LOGO programs, but the programs are inelegant and transfer to new programming is

FIGURE 8–14
House problem with clues and feedback

Assignment 1: Write a program, named HOUSE, that will draw the house below.

How to Design Your Program

1. Break down the program into smaller parts. (MODULARIZATION)
2. Check to see if there are some parts that are the same. If there are, one procedure may be used for both parts. (REUSABILITY)
3. Check each part to see if there are sequences of actions that are repeated. The repeated sequences will be written using the REPEAT command. (REUSABILITY)
4. Write a procedure for each of the parts, one at a time. (MODULARIZATION)
5. a) Run the procedures, one at a time, and edit any mistakes. (MODULARIZATION)
   b) Note the location of the turtle at the end of each procedure. If necessary, write the commands that will move the turtle to the proper location for it to draw the next procedure.
6. Arrange the order of the procedures and write a procedure named HOUSE that includes the procedures for the parts and the "joining" commands. Run the procedure HOUSE and fix any errors.

Example of How to Design and Write Assignment 1

1. Break down the program into smaller parts. (MODULARIZATION)

 Roof  Frame  Door

2. Check to see if some parts are the same. (REUSABILITY)
   Frame and Door are both squares. Can make one procedure and use a variable input to change the length of the sides.

3. Check each part for repeated sequences of actions. (REUSABILITY)

   Door and frame are squares. A square repeats the sequence "move forward, turn right" 4 times. Can use REPEAT command.

   Roof is an equilateral triangle. It repeats the sequence "move forward, turn right" 3 times. Can use REPEAT command.

4. Write a procedure for each part, one at a time. (MODULARIZATION)

```
TO SQUARE:SIDE
  REPEAT 4[FD:SIDE RT 90]
END
```

This is the procedure to make the frame and the door. SIDE is the length of the sides.

(continued)

FIGURE 8–14

(continued)

```
TO TRIANGLE:SIDE
  REPEAT 3[FD:SIDE RT 120]
END
```

This is the procedure to make the roof. SIDE is the length of the sides.

5. a. Run the procedures, one at a time, and edit mistakes. (MODULARIZATION)

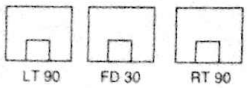
SQUARE 80 

SQUARE 20 

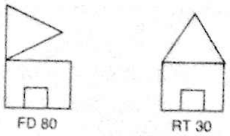
Each procedure started and ended with the turtle facing the top of the screen.

TRIANGLE 80 

b. Write commands to move the turtle from its position at the end of one procedure to the correct position for it to start drawing next procedure.



To move the turtle from end position of the door to correct position to start the frame you need the commands LT 90 FD 30 RT 90.



To move the turtle from the end position of the frame to the correct position to start the roof you need FD 80 RT 30.

Example of Program HOUSE

```
TO HOUSE
  SQUARE 20
  LT 90 FD 30 RT 90
  SQUARE 80
  FD 80 RT 30
  TRIANGLE 80
END
```

Other Procedures

SQUARE:SIDE

```
TO SQUARE:SIDE
  REPEAT 4[FD:SIDE RT 90]
END
```

TRIANGLE:SIDE

```
TO TRIANGLE:SIDE
  REPEAT 3[FD:SIDE RT 120]
END
```

If you followed the steps for designing a program, your program should be similar to this one. If it isn't, review the steps and compare what you did at each step with the example shown on the previous page.

Source: From Fay, A. L., & Mayer, R. E. (1994). Benefits of teaching design skills before teaching LOGO computer programming: Evidence for syntax-independent learning. Journal of Educational Computing Research, 11, *187–210. Copyright 1994 Baywoood Publishing Company. Reprinted by permission.*

limited. In a more recent study, Lehrer, Lee, and Jeong (1999) found that LOGO learning was greatly enhanced when students were encouraged to reflect on program design by taking the role of program designer for peer audiences.

In another LOGO study, Lee and Thompson (1997) provided 16 hours of instruction in LOGO based on a discovery or guided discovery method. In each session, all students received an introduction to new programming commands and sets of LOGO programming problems for in-class student activity. Guided discovery students received a worksheet that

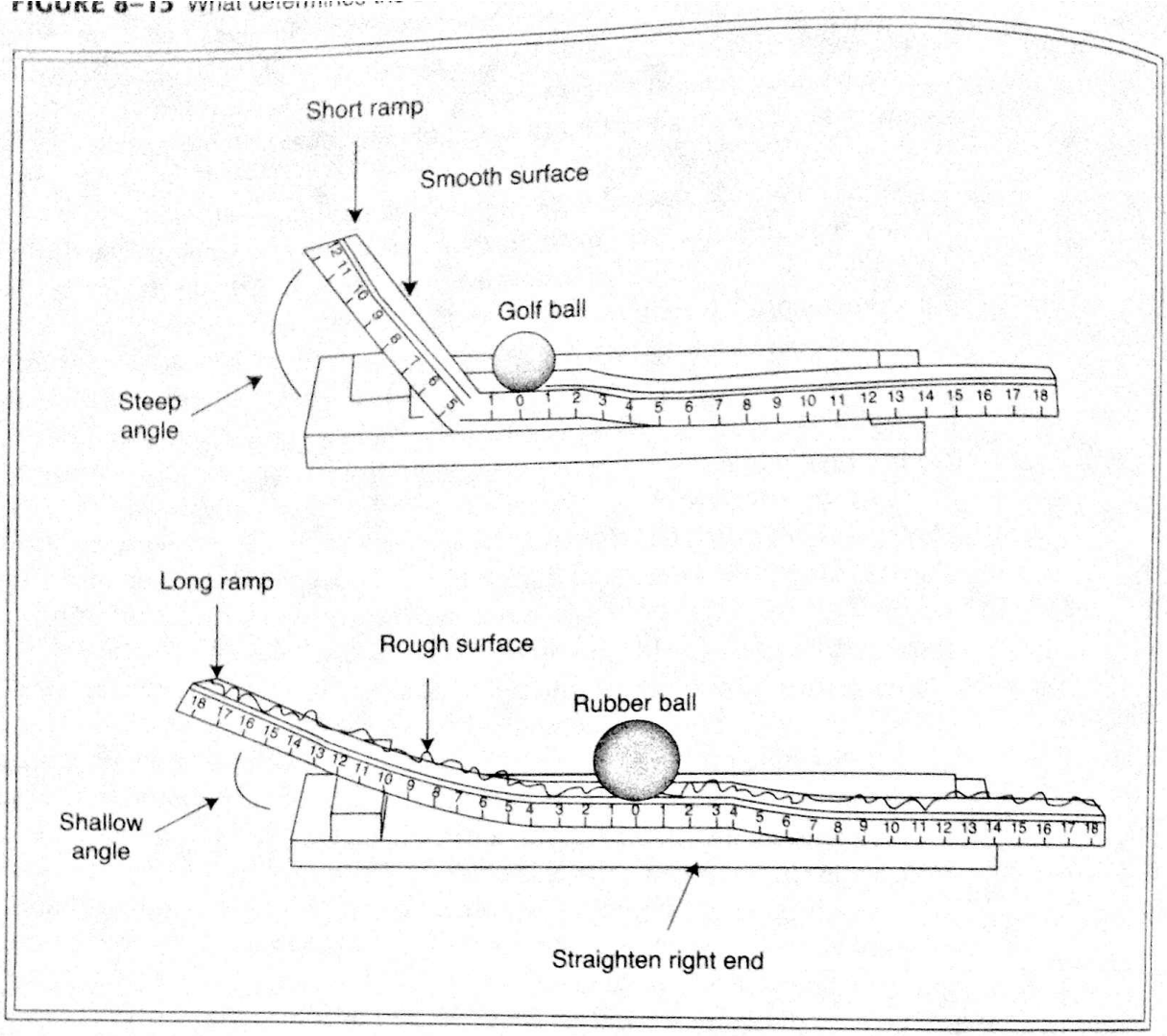
guided them through the processes of decomposing, planning, executing, identifying errors, and debugging errors, and classroom discussion was tied to reflecting on these steps. Discovery students received a less structured worksheet, and classroom discussion was free ranging. Posttest results revealed the guided discovery students outperformed the discovery students on almost every aspect of LOGO programming, including being able to generate and debug programs.

Overall, research on teaching computer programming serves as a case example of the benefits of guided discovery methods over pure discovery methods. Hands-on exploration can result in deep learning or shallow learning, depending on the degree to which students learn general principles and strategies. Pure discovery methods often fail to support transfer because students fail to gain the insights they need; in contrast, appropriate levels of guidance can promote transfer to the extent they enable students to build generalizable principles and strategies. The pattern of results, first popularized in the 1960s, once again has been replicated in the domain of teaching and learning computer programming.

In a recent review of research on discovery methods in three different subject areas, Mayer (2004a) concluded that "there is sufficient evidence to make any reasonable person skeptical about the benefits of discovery learning" (p. 14). In each of the three areas of research, "guided discovery was more effective than pure discovery in helping students learn and transfer" (p. 14) prompting Mayer to ask, "Should there be a three-strikes rule against pure discovery learning?" (p. 14). Similarly, in a recent review of research on scientific discovery learning with computer-based games and simulations, de Jong (2005) concluded that students learn best when they are given substantial guidance rather than left to work through a simulation on their own. The guidance can take the form of *scaffolding*, in which a tutor models how to carry out part of a task, provides hints for how to carry out the task, or fills in important parts of the task for the learner. Although games and simulations have much potential for helping students learn, de Jong notes that "finding out which types of guidance are effective in scientific discovery learning is the main research question for the coming years" (p. 224).

Perhaps discovery works best when the goal is to help the student engage in high-level problem-solving strategies such as learning how to conduct unconfounded scientific experiments (as described in Chapter 6). For example, consider the following situation: You let go of a ball on a downhill ramp to see how far it travels after it leaves the bottom of the ramp. You can choose ramps that are long or short, have rough or smooth surfaces, and have high or low steepness, and you choose a rubber ball or a golf ball. For each ramp you construct, you can let go of the ball and see how far the ball travels after leaving the downhill ramp.

Klahr and Nigam (2004) examined this situation using pure discovery and guided discovery methods. Some third- and fourth-grade students (pure discovery group) were allowed to test the hypothesis that length of the ramp determines how far the ball will travel. For example, to test the hypothesis that the length of the ramp determines how far the ball will travel after leaving the ramp, the student may place a golf ball on a short ramp with a smooth surface and steep decline (shown in the top of Figure 8–15) and compare that to placing a rubber ball on long ramp with a rough surface and shallow decline (shown in the bottom of Figure 8–15). The trouble with this comparison, of course, is that the student has confounded several variables so it is not possible to attribute any differences in the length the ball travels to whether the ramp is long or short. Students in the pure discovery group were free to conduct experiments and interpret the results but they received no instruction from the experimenter.



Adapted from Klahr and Nigam (2004).

For other third- and fourth-grade students (guided discovery group), the experimenter set up an experiment, such as the one shown in Figure 8–15, and asked the students whether this design will allow them "to tell for sure" whether a variable (e.g., length of ramp) had an effect on how far the ball travels. Then the experimenter explained how unconfounded experiments allowed you to tell for sure, whereas confounded experiments did not. Following this instruction, the student was allowed to design his or her own experiments to test new hypotheses.

The results showed that 75% of the guided discovery group learned to reason scientifically (by creating unconfounded experiments to test hypotheses), whereas 25% of the pure discovery group did. The learning paid off: The students who learned to reason scientifically—that is, the majority of the guided discovery learners and the minority of the pure discovery learners—performed well on a transfer test consisting of judging the scientific quality of science fair posters. Overall, this study shows that even when the goal is to teach students how to discover, a guided approach is more effective than a pure discovery approach.

Inductive Methods

EXAMPLE OF INDUCTIVE METHODS

Another issue raised by teaching students to solve parallelogram area problems—such as in Figure 8–1—concerns when to present the formula or rule. We could begin by stating the rule, Area = Height × Base, and then ask students to solve problems. This is a deductive method because the rule is given first. Alternatively, we could begin by asking students to solve problems and only after the students have built up some good intuitions, then present the formula. This is an inductive method, because the rule is given only after the learner has induced the underlying framework for the rule.

THEORY: ASSIMILATION TO EXISTING KNOWLEDGE

Ausubel (1968) and Mayer (1999) have suggested that meaningful learning involves actively connecting new material with existing knowledge. Thus, learners must be challenged into thinking about how new principles or laws relate to other ideas in the learners' memory. In inductive methods of instruction, learners are exposed to long periods of mental searching before they can verbalize the rule, or what Hendrix (1947, 1961) called "nonverbalized awareness." This period of mental searching helps activate more of the learner's prior knowledge and enables the learner to actively encode the strategy or concept to be learned into a wider or more meaningful context. In contrast, deductive methods of instruction do not encourage this search and predispose the learner toward encoding an isolated series of mechanical steps.

RESEARCH AND DEVELOPMENT: INDUCTION OF MATHEMATICAL PRINCIPLES

As early as 1913, Winch presented evidence demonstrating the superiority of deductive methods over inductive methods for short-term retention performance and the superiority of inductive methods for certain types of transfer performance. In a literature review covering the subsequent half century, Hermann (1969) concluded that there still was qualified support for the claim.

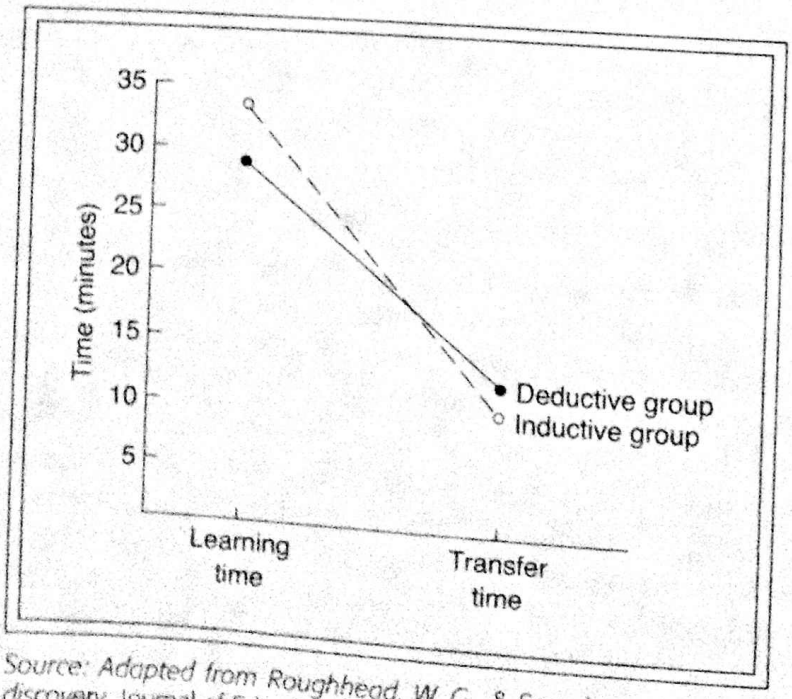
Computational Principles In a well-controlled classroom study, Worthen (1968) used two methods to teach children such concepts as notation, addition, and multiplication of integers, the distributive principle of multiplication over addition, and exponential multiplication and division. One group was given inductive instruction: Examples were presented for the children to solve, followed by verbalization of the required principle or concept. Another group was given deductive instruction: Verbalization of the required concept or principle was followed by examples for the children to solve. Significant effects as a result of instructional method were found in measures of learning ease (inductive inferior to deductive), long-term retention (inductive superior to deductive), and transfer (inductive superior to deductive), with no differences in learner attitude.

As a part of a larger study, Roughhead and Scandura (1968) used inductive and deductive methods to teach children about series summation. In a deductive method, the rule was

given and applied to several problems; then the subjects were asked to solve similar problems that were based on the same rule. In the inductive method, subjects were asked to solve some problems; then the rule was given and applied to similar problems. The deductive group learned faster than the inductive group; however, in a transfer task, the inductive group learned to solve the new problems faster than the deductive group. These results are summarized in Figure 8–16.

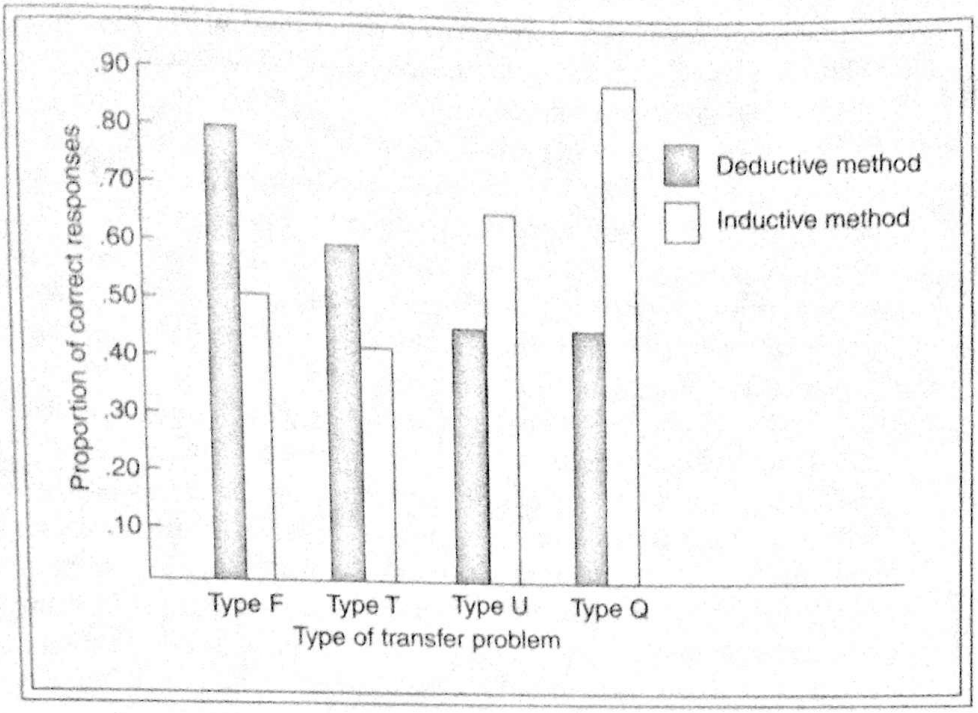
Statistical Principles Another study (Mayer & Greeno, 1972) varied the sequencing in a programmed text for the concept of binomial probability (i.e., the probability of obtaining *R* successes in *N* trials). An inductive booklet began by presenting underlying concepts such as "trial," "success," and "probability of success" and gradually put the parts together into a formula by the end of the booklet. A deductive booklet began by presenting the computational formula in symbolic notation and then gradually showed how the component variables figured in using the formula. Although both booklets presented the same basic information and same computational examples, the inductive booklet was sequenced to move from example to rule, and the deductive booklet was sequenced to move from rule to example. Figure 8–17 summarizes the groups' performance on a subsequent test that included problems just like those given in the booklet (Type F), problems that were slightly modified from those given in the booklet (Type T), problems that were unanswerable (Type U), and questions about when and how to use the formula (Type Q). Emphasis on the formula (i.e., deductive training) resulted in better performance on problems like those that the student was trained to solve, but emphasis on underlying concepts resulted in better performance on recognizing when the formula did not apply and on creative question answering. The Mayer and Greeno study suggests that deductive methods are better for simple retention of a basic rule whereas inductive methods are better for situations in which the rule must be transferred to new situations.

3–16
deductive
ive
n
nd



Source: Adapted from Roughhead, W. G., & Scandura, U. M. (1968). What is learned in mathematical discovery. Journal of Educational Psychology, 59, 283–289.

FIGURE 8–17
Effects of inductive and deductive methods on problem solving



Source: Adapted from Mayer, R. E., & Greeno, J. G. (1972). Structural differences between learning outcomes produced by different instructional methods. Journal of Educational Psychology, 63, *165–173.*

IMPLICATIONS OF INDUCTIVE METHODS

Research on sequencing of instruction suggests that deductive methods lead to superior performance in cases of one single rule to learn or a limited number of problems to be solved. Explicit instruction and practice in applying a specific rule is most effective when the goal of instruction is limited to behaviors that are similar or identical to those being taught. In contrast, the foregoing research demonstrates that inductive methods of instruction are useful when the goal of instruction is the ability to learn how to form rules (rather than learning of a specific rule) or how to transfer to new situations. By being encouraged to think actively about how to solve problems during instruction, the learner develops problem-solving strategies that can be applied in many situations.

COMPUTER APPLICATIONS: LEARNING FROM SIMULATED EXPERIENCES

In the inductive approach, teachers begin with the learner's concrete experiences and carefully relate those experiences to abstract principles. In the deductive approach, teachers present a to-be-learned principle for students to memorize and apply. As you could see in the previous sections, classic laboratory research shows that students are better able to transfer when they learn by an inductive rather than a deductive approach. Do inductive approaches also promote learning of academic material in classrooms?

An example of a popular inductive approach is the predict-observe-explain (POE) method for teaching scientific principles (White & Gunstone, 1992). In the POE method, a teacher presents a familiar concrete situation and asks the student to predict what will

happen and justify that prediction (predict phase); the student observes what happens (observe phase); and the student explains any differences between the predicted and the observed result (explain phase).

For example, suppose I asked you which would be a better way to keep a can of soda cold for a picnic—wrapping it in aluminum foil or wrapping it in wool? Not only must you choose an answer, you must also explain why you gave that answer. This is the predict phase. Then you conduct an experiment—such as watching a computer simulation of heat transfer through the wrapping over time. This is the observe phase. Then you engage in a discussion with peers and the teacher about the observations you just made. This is the explain phase.

What would happen to eighth-grade students who took a semester-long science course based on this kind of inductive teaching approach? To help answer this question, Linn and Hsi (2000) report on a project in which students learn general science principles by carefully examining familiar situations involving heat and temperature, thermal equilibrium, and insulation and conduction. These situations include figuring out whether a metal desk feels cooler than a wooden desk, whether soup stays warmer in a small cup or a large bowl, whether it's better to use a wooden spoon or a metal spoon to stir a pot of boiling noodles, whether it's a good idea to leave a car window slightly open when it will sit in the sun all day, or how to keep some pizzas warm for 30 minutes. The instructional method is inductive to the extent that it begins with the learner's familiar experience—such as experience with warm food getting cold—and carefully moves toward building abstract principles, such as a theory of heat transfer.

Linn and Hsi report that the instruction also relies on technology—such as computers to make visible changes in temperature in objects over time—and on peers, such as group discussions involving what happens in the concrete situations. For example, a useful computer simulation called "Heat Bars" allows the student to select bars of materials such as aluminum, glass, or plastic; place one end of each bar next to a heat source for a selected amount of time; and then watch the heat flow across the bar over time. In this way the student can see, for example, that heat flow is slower for some materials than for others. Using a program called Electronic Laboratory Notebook, students are able to make predictions, run experiments, collect data, and generate principles that are shared with others.

Over many years of refining their instructional program called Computers as Learning Partners (CLP), a multidisciplinary research team produced a series of eight versions—each one intended as an improvement over the previous one. Before taking the course, almost all students were unable to provide scientifically valid answers to questions about temperature and heat, whereas afterward they showed a large improvement. Most important, with each new revision, the percentage of successful students increased. When later tested in high school, students who had taken the CLP course showed better performance on science tests involving high-level thinking than did comparable non-CLP students. Linn and Hsi's (2000) report shows that an inductive approach that builds on students' familiar experiences can be a useful way to teach for transfer, but the road to crafting a useful curriculum requires a long-term, multidisciplinary effort.

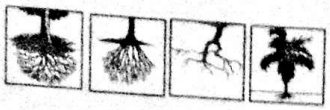
In a smaller scale study, Moreno, Mayer, and Lester (2000) developed an educational computer game aimed at teaching principles of environmental science through concrete experience as simulated on a computer screen. In the program, students meet Herman the Bug, who takes them on a space trip to a new planet that has certain environmental

URE 8–18
ansfer problem
the Design-A-
nt game

Design a plant to live in an environment that has low sunlight.

Circle the type of root (1 or more):

Branching



Deep, Thick | Deep, Thin | Shallow, Thick | Shallow Thin

Nonbranching

Deep, Thick | Deep, Thin | Shallow, Thick | Shallow, Thin

Circle the type of stem (1 or more):

Short

Thick Bark | Thin Bark | Thick No Bark | Thin No Bark

Long

Thick Bark | Thin Bark | Thick No Bark | Thin No Bark

Circle the type of leaf (1 or more):

Thin

Small, Thick skin | Small, Thin skin | Large, Thick skin | Large Thin skin

Thick

Small, Thick skin | Small, Thin skin | Large, Thick skin | Large, Thin skin

Why do you think that the plant you designed will survive in this environment?

Source: From Moreno, R., Mayer, R. E., Spires, H., & Lester, J. C. (2001). The case for social agency in computer-based teaching: Do students learn more deeply when they interact with animated pedagogical agents? Cognition and Instruction, 19, 177–219. *Copyright 2001 Lawrence Erlbaum Associates. Reprinted by permission.*

conditions—such as low rainfall and heavy winds. The student's job is to design a plant that will survive on the planet by selecting appropriate roots, stems, and leaves. For example, in a rainy environment, is it better to have roots that are thick or thin, deep or shallow, branched or nonbranched? Along the way, Herman provides explanations for why the student's choice leads to the plant's death or survival. The program, called Design-A-Plant, is based on an inductive method to the extent that it begins with simulated, familiar experience and builds on that experience to help students understand general principles.

Does the Design-A-Plant experience help students learn generalizable principles about plant growth? To test this question, Moreno, Mayer, Spires, & Lester (2000) asked students to solve new problems presented in paper-and-pencil form. For some problems, students were given new environmental conditions (not encountered during instruction) and asked to select appropriate roots, stems, and leaves and to write a justification; for other problems, they were shown a picture of a plant with certain roots, stems, and leaves and asked to describe an environment in which it would survive and give an explanation for their response. Figure 8–18 shows an example of a transfer problem. Students who learned with the Design-A-Plant game generated more correct solutions on the transfer test than did students who received the identical information presented in traditional textbook format. The improvement on this problem-solving transfer test was 24% for college students and 48% for high-school students over their peers who learned from traditional deductive-based text (Moreno et al., 2000; Moreno, Mayer, Spires, & Lester, 2001). Clearly, an inductive approach of building on the simulated experience of learners can promote meaningful learning.

Chapter Summary

This chapter has explored three representative techniques for providing meaningful methods of instruction—concrete materials, discovery activities, and inductive sequencing. In each case, there is some laboratory-based evidence and classroom-based evidence that meaningful methods of instruction encourage the learner to become more cognitively involved in the learning task, and thus the outcome of learning allows for better problem-solving transfer. Designing classroom programs aimed at promoting transfer may involve aspects of all three techniques—providing concrete materials, opportunities for active problem solving, and links to familiar experiences. For example, Brenner et al. (1997) developed an effective middle school prealgebra unit that used concrete manipulatives to represent functional relations (concrete method), related algebraic problems to familiar situations involving pizzas (inductive method), and required active discussion of problem-solving strategies (discovery method).

The major curriculum development efforts of the 1960s attempted to incorporate aspects of meaningful methods of instruction. Since that time, there has been a reaction against developments such as the "new math." In an effort to reestablish the traditional emphasis on "getting the right answer," the pendulum seems to have swung "back to basics" in the 1970s and 1980s. Since then, the pendulum continues to swing, sometimes in ways that lose the positive features of both meaningful and basic approaches.

If you are concerned with teaching for meaningful learning, you might want to ask, "Will meaningful methods ever again become an acceptable part of school learning?" This chapter provided current computer applications that may help the pendulum swing in the direction of guided exploration. During the 1970s, teaching machines, including classroom computers, were best adapted for drill and practice on basic skills. However, as computers become more powerful, they enable meaningful methods of instruction (e.g., computerized visualization systems, LOGO environments, and computer simulation games) aimed at teaching transferable problem-solving skills. Whether these advances will rekindle a more lasting and productive interest in meaningful learning remains to be seen. What is new, however, is

the growing realization that promoting student exploration is not enough; student exploration must be supplemented with guidance that promotes reflection and helps students find generalizable rules and principles that can live on in new situations. Crafting such lessons involves the skillful use of many techniques, including those based on concreteness, activity, and familiarity.

SUGGESTED READINGS

Bruner, J. S. (1968). *Toward a theory of instruction*. New York: Norton. (A classic call for meaningful methods of instruction.)

Lillard, A. S. (2005). *Montessori: The science behind the genius*. New York: Oxford University Press. (An exploration of Montessori's vision of education along with reviews of modern relevant research.)

Mayer, R. E. (2005). Should there be a three-strikes rule against pure discovery learning? *American Psychologist, 59*, 14–19. (A review of research on discovery learning.)

Wertheimer, M. (1959). *Productive thinking*. New York: Harper and Row. (A classic call for learning by understanding.)