

This is a pre-publication version of this paper. The definitive final version is available via the publisher.

Doering, A., & Veletsianos, G. (2009). Teaching with Instructional Software. In M. D. Roblyer & A. Doering (Eds.), *Integrating Educational Technology into Teaching* (73-108). Upper Saddle River, NJ: Pearson Education.

# Chapter 3

## Teaching with Instructional Software



The fact that individuals bind themselves with strong emotional ties to machines ought not to be surprising. The instruments [we] use become . . . extensions of [our] bodies.

Joseph Weizenbaum, in *Computer Power and Human Reason* (1976, p. 9)

# Technology Integration Example

## The Alien Rescue® Project

**Grade Level:** Middle school • **Content Area/Topic:** Science, solar system • **Length of Time:** Three weeks

### Phases 1 and 2: Assess technological pedagogical content knowledge; Determine relative advantage

Mr. Leroy was a veteran middle school science teacher who had won many awards for his teaching. However, he felt that, despite his efforts to make his class hands-on and project based, his students tended to see science as facts and procedures unrelated to practical problems. He felt most students left middle school without learning why scientific inquiry was useful. One day, he saw a conference presentation on software for middle school students. The software began with a video clip in which a problem scenario was presented as if it were a real live newscast. The video said that an alien spaceship was orbiting Earth and broadcasting a plea for help. Aliens on board the ship were from several planets in a galaxy whose sun had exploded. They escaped but now needed new habitats that would meet the needs of their various species. They asked Earth for help in matching them with planets and moons that each species might find habitable. The software provided an array of information and images on the planets and moons of our solar system, and students could find out about the characteristics of each by doing searches and activities like designing and “sending out” probes. After Mr. Leroy reviewed the

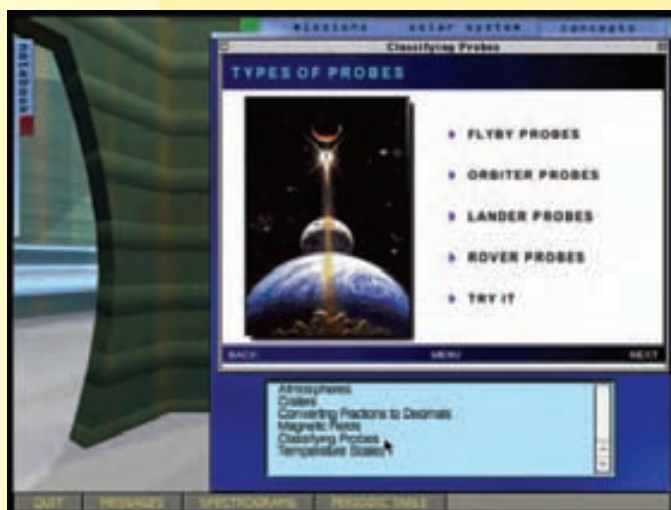
software himself and decided it met all criteria for what he believed to be excellent software, he decided to have his students do this project for his solar system unit instead of building models of planets. He felt it would be a good way to generate enthusiasm for using scientific inquiry to help solve complex problems.

Before jumping immediately into the project, however, Mr. Leroy assessed his technological pedagogical content knowledge to see where he might be deficient and also how he might capitalize on his strengths. He realized that his content and pedagogical knowledge was very strong as compared to his technological knowledge. However, he also believed that the software he would be using guided the teacher enough that he could definitely integrate it within his classroom.

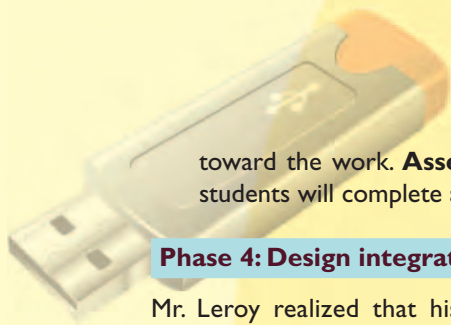
### Phase 3: Decide on objectives and assessments

Mr. Leroy decided that, in addition to passing his usual test on solar system information, he wanted students to accomplish several other outcomes. Also, he wanted to make sure students enjoyed this new approach. He developed the following outcomes, objectives, and assessments for his project:

- **Outcome:** Demonstrate a problem-solving approach to assigned problems. **Objective:** All students, working in small groups, will create well-designed problem statements, solutions steps, and workable solutions for at least one of three assigned problems. **Assessment:** A checklist based on steps and criteria from the *Alien Rescue Teacher Manual (TM)*.
- **Outcome:** Create and present new problems and methods to solve them. **Objective:** All groups will achieve a rubric score of at least 80% on presentations of a new problem and solution related to the alien simulation software. **Assessment:** A rubric to assess the content and quality of the presentation.
- **Outcome:** Use scientific inquiry language outside the unit. **Objective:** At least 80% of students will use the language of scientific inquiry outside of the unit. **Assessment:** An observation checklist.
- **Outcome:** Enjoy using inquiry methods. **Objective:** At least 90% of students will demonstrate positive attitudes



Source: Screen from *Alien Rescue Software* The Alien Rescue Team 2001–2004 (<http://www.alienrescue.com>). Reprinted by permission.



toward the work. **Assessment:** A five-item survey that students will complete anonymously.

#### Phase 4: Design integration strategies

Mr. Leroy realized that his students would learn inquiry-based methods only by seeing them in action and practicing them over time. He would have to facilitate, rather than direct, student learning, which would take more time than usual. He also realized that students had to answer questions about the solar system on the state's required exams, so he would have to provide diagnostic testing and if necessary, remedial practice about these concepts and facts. Luckily, the *TM* provided test items like this to cover background information, which he could supplement with his own questions. With these things in mind, he created the following activity sequence:

- **Day 1:** Show the opening video to the whole class. Assign students to groups, and let them log on to computers. Allow them to explore the software environment on their own.
- **Day 2:** Have the class review the problem presented in the video by asking the probing questions listed in the *Alien Rescue TM*. Let groups explore the software further, and answer any additional questions they might have.
- **Days 3–5:** Hold discussions about how to work in groups, and use inquiry approaches to solve problems. Ask students to develop their own problem statements, and discuss procedures they will use to solve them. Students will demonstrate the software features they will need. Students practice using the software features and define lists of alien needs.
- **Days 6–10:** Students use software features to explore planets and moons and develop hypotheses about the best matches for each alien species. They use word processing software to document their notes so they can update them easily throughout the activity.
- **Days 11–13:** Groups present their findings on the best places for each species and write up final selections in the required format. Individual students complete the fact worksheets and get remedial practice as necessary.
- **Days 14–16:** Final reflections, assessments, and remedial practice where necessary.

#### Phase 5: Prepare the instructional environment

Mr. Leroy decided to take the *TM*'s advice and resist the tendency to overdirect students. He knew he had to keep to a timeline to complete the project on schedule, but he also realized that the software designers were warning that too much structure would work against the purpose and design of the software. Keeping this in mind, he began his planning:

- **Software skills:** The software was quite rich with features, tools, and resources, but it was so complex that

Mr. Leroy knew he couldn't be expert on all of it. He also realized that some of his students would probably become more expert than he in a very short time. He wanted to become as familiar with it as possible, however, so he could answer students' initial questions. So he went through it carefully and took notes.

- **Handouts:** The *TM* provided a chart to help students keep track of each species' needs. Mr. Leroy prepared a group task sheet and inserted this and other helpful notes on it.
- **Lesson planning:** Mr. Leroy was such a veteran that he usually didn't use lesson plans. But this project was so new and different from his usual teaching methods that he felt he needed to prepare a written sequence. He left space for items he would add as the project progressed so that he would be even more prepared when he taught it next time.
- **Software copies:** Mr. Leroy knew the software was available to teachers only from the university where it was developed. It was free, but he had to sign an agreement that only his class would be using it. He completed the paperwork and obtained the software. He made sure each copy worked on the available equipment.
- **Computer scheduling:** The trickiest part would be scheduling time on computers in the computer lab, which he would be using since he had only one computer in his room. Mr. Leroy arranged with the lab coordinator to have the lab on the days he needed it. He also spoke with the principal, who agreed to give him an extra computer for his room so that more students could work there if necessary.

#### Phase 6: Evaluate and revise

Mr. Leroy was impressed with the impact the project had on his students' enthusiasm for inquiry skills. Rubric scores indicated that they did better with each probe they designed and each set of data they analyzed. Perhaps best of all, Mr. Leroy documented that about half of the students were talking like scientists, using "inquiry speak" in other work. Some girls asked about careers in space exploration, so Mr. Leroy sent them to look at the NASA and Space Camp sites on the Internet. The problem-solving checklists reflected good progress, too. Scores on the first ones the students did were a little low, but the last ones were really fine in all groups. Mr. Leroy observed only two real problems. One was that some group members had to leave for special school events, leaving a "hole" in the group, which then had to improvise. The other problem was access to computers. Mr. Leroy resolved to ask the principal for a five-computer workstation and printer like the English teacher had. Students' scores on the solar system test varied a lot, so he interviewed students informally to determine why some did so much better than others. In general, attitudes toward science seemed way up, and students were more on task than he had ever seen them.





## Objectives

After reading this chapter and completing the learning activities for it, you should be able to:

1. Define five software functions (*drill and practice*, *tutorial*, *simulation*, *instructional game*, and *problem solving*) according to their unique features, benefits, and limitations, and identify examples of each.
2. Define *integrated learning systems (ILSs)* according to their unique features, benefits, and limitations, and identify examples of ILS products.
3. Identify one or more types of instructional software functions that could meet classroom needs.
4. Plan lesson activities that integrate instructional software using a directed learning strategy.
5. Plan lesson activities that integrate instructional software using a constructivist or combination learning strategy.

## What Is Instructional Software?

Weizenbaum (1976) called using computers “extensions of [our] bodies.” Reeves and Nass (1996) showed the intimate and inherently social relationships between computers and humans, calling computers “social actors.” Such metaphors have a long history in education. At the same time as people began to realize that computers could help them do many clerical tasks more quickly and accurately, they also began thinking, “If **computer programs** can be created to do essentially anything, why not program computers to teach?” Educators and developers alike began to pursue this idea in the 1960s and 1970s. Some, like William Norris (1977), who developed Control Data’s PLATO teaching system, believed that computer-based education was the only logical alternative to education’s “outdated, labor-intensive ways” (p. 451). He believed that education would be more efficient if computers took over the traditional role of teachers.

Today, after more than 30 years of development and experimentation, there is less talk of computers replacing teachers. Instead, there is more conversation about computer programs helping perform various teaching functions, essentially transforming the teaching process. This chapter shows how software programs empower rather than replace teachers.

**Software** is another word for programs written in programming languages. Software designed and developed to make computers function is called **systems software**, whereas programs written to perform tasks such as word processing or tutoring are called **applications software** or **applications programs**. **Instructional software** is applications software designed specifically to deliver or assist with student instruction on a topic. Although applications software such as word processing and spreadsheets can also enhance instructional activities, this textbook differentiates between applications software and instructional software. Software tools serve many

purposes other than teaching; instructional software packages are developed for the *sole* purpose of supporting instruction and/or learning.

In the early days—when instructional software was used primarily to tutor students—it was called **computer-assisted instruction (CAI)**. The term is still in common use, but some kinds of instructional software are designed with more constructivist purposes in mind; they support, rather than deliver, instruction. Therefore, many people consider the term *CAI* outdated and misleading. Teachers may hear instructional software referred to as *computer-based instruction (CBI)*, *computer-based learning (CBL)*, or *computer-assisted learning (CAL)*, or in more generic terms such as *software learning tools*.

## Instructional Roles for Software: Past and Present

It used to be easy to designate a software package by the type of teaching function it served. It was a **drill-and-practice**, **tutorial**, **simulation**, **instructional game**, or **problem-solving** program. (See descriptions in Table 3.1.) These terms originated because each package had clearly different characteristics and served a different instructional purpose. In contrast, much of today’s software defies easy classification because many software packages contain several different activities, each of which may perform a different function. For example, language-learning software may have a number of straight drill activities along with activities that fulfill problem-solving and game functions. Also, developers use the terms interchangeably; there seems to be no consensus among developers about the terms used to describe various types of programs. Some developers refer to a drill program that gives extensive feedback as a tutorial. Others refer to simulations or problem-solving functions as games.

Software still reflects the same five functions, but in light of current trends toward multiple-function software

**TABLE 3.1** Five Instructional Software Functions

Function/Examples	Description
<b>Drill and Practice</b> <a href="http://www.transparent.com/">http://www.transparent.com/</a>	Allows learners to work problems or answer questions and get feedback on correctness.
<b>Tutorial</b>	Acts like a human tutor by providing all the information and instructional activities a learner needs to master a topic: information summaries, explanation, practice routines, feedback, and assessment.
<b>Simulation</b> <a href="http://www.digitalfrog.com">http://www.digitalfrog.com</a>	Models real or Imaginary systems to show how those systems or similar ones work or to demonstrate underlying concepts.
<b>Instructional Game</b>	Increases motivation by adding game rules to drills or simulations.
<b>Problem Solving</b>	(a) Teachers directly (through explanation and/or practice) the steps involved in solving problems or (b) helps learners acquire problem-solving skills by giving them opportunities to solve problems.

packages, teachers may have to analyze a package to determine which instructional function(s) it serves so as to ensure it supports their specific teaching needs. They may not be able to refer to an entire package as a drill or a simulation, but it is possible and desirable to identify whether it provides, for example, science vocabulary skill practice and/or opportunities for studying plant growth in action. As this chapter will show, each software function serves a different purpose during learning and, consequently, has its own appropriate integration strategies.

### Learning Theory Connections

The first instructional software reflected the behavioral and cognitive learning theories that were popular at the time. Some software functions (e.g., drill and practice, tutorial) remain focused on *directed* strategies that grew out of these

theories, delivering information to help students acquire and retain information and skills. Later instructional software was designed to support the more *constructivist* aims of helping students explore topics and generate their own knowledge. Therefore, some software functions (e.g., simulation, games) can be used in either directed or constructivist ways, depending on how they are designed. Table 3.2 summarizes the strategies underlying each of the five software functions described in this chapter.

Gagné, Wager, and Rojas (1981) suggested a way to look at software that can help educators analyze a given product with respect to its instructional function(s) and design appropriate integration strategies that make use of these functions. Gagné et al. said that drills, tutorials, and simulations each accomplish a different combination of the Events of Instruction. (See the description of Gagné's Events of Instruction in Chapter 2.) The nine events are guidelines identified by Gagné

**TABLE 3.2** Types of Integration Strategies for Each Instructional Software Function

Software Function	Instructional Uses	Strategy	
		Directed	Constructivist
Drill and practice	Skill practice	X	
Tutorial	Information delivery	X	
Simulation	Demonstration	X	
	Exploration		X
Instructional game	Skill practice	X	
	Exploration		X
Problem solving	Skill practice	X	
	Exploration		X



## Technology Integration Lesson 3.1

### Example Strategy for Using Programming Languages

**Title:** A Window on Learning Logo

**Grade Level:** Middle school

**Content Area/Topic:** Logic and analysis skills

**NETS for Students:** Standards 1 (Creativity and Innovation) and 4 (Critical Thinking, Problem Solving, and Decision Making)

**Description of Standards Applications:** This integration idea offers exploration and creativity in the Logo programming environment. Students use creativity, critical thinking, problem solving, and decision making as they create a Gothic rose using the on-screen “turtles” while they are learning how to develop and analyze patterns. Logo’s graphic qualities make it a natural choice to explore the design qualities of symmetry, repetition, and precision. Logo’s powerful language structure allows students to create intricate designs quickly and dramatically. With other methods, modifications would be more time consuming, and students would not be able to pinpoint the exact reasons and ways that designs were made different.

**Instruction:** Begin by reviewing basic Logo commands and exploring the Logo programming environment. Make sure students know that designs are programmed using the “turtle’s” on-screen perspective and not their own. Have them practice developing simple procedures, and review debugging procedures line by line to determine how designs are drawn. Begin with an easy problem, such as analyzing the steps in drawing a simple square, to get children used to the logistics. Give students pictures of the Gothic rose window from Notre Dame Cathedral. Ask them to analyze the window, looking for patterns, shapes, and structures. Help them see that complicated designs are made up of simple geometric shapes. Then assign them the task of drawing their own window using three such shapes. Show them how to adjust basic designs by changing variable numbers. The best moments in the project occur when a student displays a window for the first time and a gasp of delight fills the room.

**Assessment:** Use a rubric on programming language use and creativity.

that can help teachers arrange optimal “conditions for learning” for various types of knowledge and skills. By determining which of the events a software package fulfills, he said, educators can determine the teaching role it serves and where it might fit in the instructional process. However, Gagné’s approach was primarily for directed uses, rather than constructivist ones. This chapter describes both kinds of strategies for instructional software.

### Programming Languages as Instructional Software

This chapter focuses on software designed solely for instructional purposes (see Table 3.1 for descriptions and examples of drill and practice, tutorials, simulations, instructional games, and problem solving), whereas Chapters 4 and 5

address the uses of tool software in education. Unlike instructional software described in this chapter, the uses of tool software (e.g., word-processing, computer-assisted design or CAD software) are not limited to education. However, a few programming languages were designed especially for educational purposes and thus may be considered hybrid software, since they merge the capabilities of instructional and tool software. One of the most widely known of the programming languages used for instruction is **Logo**. Logo used to introduce young children to problem solving through programming and to explore concepts in content areas such as mathematics, science, and language arts (Galas, 1998; Gonsalves & Lopez, 1998; Weinstein, 1999). See Technology Integration Lesson 3.(p. 000) for a sample use of Logo as instructional software.

The work of Seymour Papert (1980) (see Chapter 2) and his colleagues at the Massachusetts Institute of Technology made Logo “widely used throughout the world as an introductory programming language and mathematical learning environment for students in elementary and secondary schools” (Watt, 1992, p. 615). Although not as popular as they were in the 1980s, Logo and some of its derivative materials are still used for instructional purposes.

## Recent Trends in Software Design and Delivery

Although instructional software resources have been around since the 1960s, the following are the most recent developments in their features and uses:

- **Online access and components** — The Internet is playing an increasingly prominent role in software. Much software is now delivered online with students using web-based applications that can be accessed from virtually any internet-enabled device (e.g., a computer or a cell phone).
- **Web 2.0 technologies** — Web 2.0 (pronounced “two point Oh”) refers to the transition of the web from a collection of related websites to a computing platform that emphasizes user collaboration and contribution. Examples of Web 2.0 applications are blogs, wikis, and social networking sites. At the time of writing, Web 2.0 technologies are being widely adopted in educational circles as user-centered and empowering tools. (See Chapter 6 for a complete discussion of Web 2.0 technologies.)
- **Rich user experiences** — Software design and development has advanced from a focus on information dissemination to providing experiences (as opposed to “products”) that are user friendly, engaging, fun, and aesthetically pleasing.
- **Renewed emphasis on directed strategies and networked systems** — The recent emphasis on educational accountability as reflected in the No Child Left Behind (NCLB) Act has breathed new life into strategies that were once considered passé. Even in research circles, some authors have claimed that directed teaching strategies are more effective than minimally guided teaching techniques (Kirshner, Sweller, & Clark, 2006). As constructivist methods became more popular in the 1980s and 1990s, the demand for drill-and-practice and tutorial instructional software waned and use of simulation and problem-solving software increased. Now directed strategies made possible by drills and tutorials—which are ideal for preparing students for tests—are once again on the rise. The same accountability emphasis has

created new demand for networked instructional software products called **integrated learning systems (ILSs)**, networked or online systems that provide both computer-based instruction and summary reports of student progress, described later in this chapter. These systems became popular in the late 1980s and early 1990s as an efficient way for many students to access instructional software from a central source (e.g., a school or district server), and educators often employed them to support directed instruction for remedial programs (e.g., Title III programs that provide special resources for disadvantaged students). Although de-emphasized in the late 1990s, today’s ILSs are becoming valued not only for their centralized access, but also for their ability to track and report on student progress. Data on individual and group progress in a given classroom, school, or district is a central feature of the new NCLB Act accountability requirements.

## Drill-and-Practice Software Functions

Drill-and-practice software provides exercises in which students work example items, usually one at a time, and receive feedback on their correctness. Programs vary considerably in the kind of feedback they provide in response to student input. Feedback can range from a simple display like “OK” or “No, try again” to elaborate animated displays or verbal explanations. Some programs simply present the next item if the student answers correctly.

Types of drill and practice are sometimes distinguished by how the program tailors the practice session to student needs (Merrill & Salisbury, 1984). Types of drill functions, described below, include flash card activities, branching drills, and extensive feedback activities:

- **Flash card activity** — This is the most basic drill-and-practice function, arising from the popularity of real-world flash cards. A student sees a set number of questions or problems, presented one at a time. The student chooses or types an answer, and the program responds with positive or negative feedback depending on whether the student answered correctly.
- **Branching drill** — This is a more sophisticated form of drill and practice. In branching drills, the software moves students on to advanced questions after they get a number of questions correct at some predetermined mastery level; it may also send them back to lower levels if they answer a certain number wrong. Some programs automatically review questions that students get wrong before going on to other levels. Students may not realize that branching is happening, since the program may do it automatically





**A popular use of drill software is preparing for important tests.**

without alerting them to this fact. Sometimes, however, the program may congratulate students on good progress before proceeding to the next level, or it may allow them to choose their next activities. More recently, educational software designers have attempted to use learners' cognitive status to decide what task learners should be presented with next (Salden, Paas, & van Merriënboer, 2006).

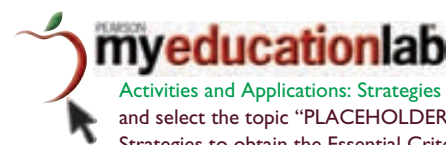
- **Extensive feedback activities** — In these drills, students get more than just correct/incorrect feedback. Some programs give detailed feedback on why the student got a problem wrong. This feedback is sometimes so thorough that the software function is often mistaken for a tutorial. (See the next section for a description of tutorial functions.) However, the function of a drill is not instruction, but rather practice. Consequently, the integration strategies for drill and tutorial functions differ.

## Selecting Good Drill-and-Practice Software

In addition to meeting general criteria for good instructional software, well-designed drill-and-practice programs should also meet specific criteria:

- **Control over the presentation rate** — Unless the questions are part of a timed review, students should have as much time as they wish to answer and examine the feedback before proceeding to later questions. A student usually signals readiness to go to the next question by simply pressing a key.
- **Answer judging** — If programs allow students to enter a short answer rather than simply choosing one, a good drill program must be able to discriminate between correct and incorrect answers.

- **Appropriate feedback for correct and incorrect answers** — If students' responses are timed, or if their session time is limited, they may find it more motivating simply to move quickly to the next question. When drills do give feedback, they must avoid two common errors. First, feedback must be simple and display quickly. Students rapidly tire of elaborate displays, and the feedback ceases to motivate them. Second, some programs inadvertently motivate students to get wrong answers by giving more exciting or interesting feedback for wrong answers than for correct ones. The most famous example of this design error occurred in an early version of a popular microcomputer-based math drill series. Each correct answer got a smiling face, but two or more wrong answers produced a full-screen, animated crying face that students found amusing. Consequently, many students tried to answer incorrectly so they could see it. The company corrected this flaw, but this classic error still exists today in other programs.



Activities and Applications: Strategies Go to MyEducationLab and select the topic "PLACEHOLDER, PLACEHOLDER." Go to Strategies to obtain the Essential Criteria Checklist for

Evaluating Instructional Software.

## Benefits of Drill and Practice

Research has shown that drill-and-practice software activities can allow the effective rehearsal students need to transfer newly learned information into long-term memory (Merrill & Salisbury, 1984; Salisbury, 1990). Many teachers feel that such practice gives students more rapid recall and use of basic skills as prerequisites to advanced concepts. They like students to have what Gagné (1982) and Bloom (1986) call *automaticity*, or automatic recall of these lower order skills, to help them master higher order ones faster and more easily. The usefulness of drill programs in providing this kind of practice has been well documented, but the programs seem especially popular among teachers of students with learning disabilities (Hasselbring, 1988; Higgins & Boone, 1993; Okolo, 1992).

Although curriculum increasingly emphasizes problem solving and higher order skills, teachers still give students on-paper practice (e.g., worksheets or exercises) for many skills to help them learn and remember correct procedures. Drill software provides the following acknowledged benefits as compared to paper exercises (Kahn, 1998–1999):

- **Immediate feedback** — When students practice skills on paper, they frequently do not know until much later whether or not they did their work correctly. To quote a common saying, "Practice does not make perfect; practice makes permanent." As they complete work incorrectly,



## Adapting for Special Needs

Given the diverse level of student abilities in every classroom, teachers need products that anticipate and engage a broad range of skills. Early-learning products like *Bailey's Book House*, *Sammy's Science House*, and *Millie's Math House* (all from Riverdeep) give students opportunities to play, explore, and interact with a variety of pre-academic instructional skills.

Also, many instructional software products support special access devices like switches and alternative keyboards. These are useful to students who have disabilities that prevent them from using instructional software via the standard keyboard.

*Contributed by Dave Edyburn*

students may actually be memorizing the wrong skills. Drill-and-practice software informs them immediately whether their responses are accurate so they can make quick corrections. This helps both “debugging” (identifying errors in their procedures) and retention (placing the skills in long-term memory for future access).

- **Motivation** — Many students refuse to do the practice they need on paper, either because they have failed so much that the whole idea is abhorrent, they have poor handwriting skills, or they simply dislike writing. In these cases, computer-based practice may motivate students to do the practice they need. Computers don't get impatient or give disgusted looks when a student gives a wrong answer.
- **Saving teacher time** — Since teachers do not have to present or grade drill and practice, students can practice on their own while the teacher addresses other student needs. The curriculum has dozens of areas in which the benefits of drill and practice apply. Some of these are:
  - Math facts
  - Typing skills
  - English- and foreign-language vocabulary
  - Countries and capitals
  - SAT and TOEFL skills
  - Musical keys and notations.

## Limitations and Problems Related to Drill and Practice

Although drill and practice can be extremely useful to both students and teachers, it is also the most maligned of the

software activities, sometimes informally referred to among its critics as “drill and kill.” This criticism comes from the following two sources:

- **Perceived misuses** — Some authors have criticized teachers for presenting drills for overly long periods or for teaching functions that drills are ill suited to accomplish. For example, teachers may give students drill-and-practice software as a way of introducing new concepts rather than just for practicing and reinforcing familiar ones.
- **Criticism by constructivists** — Since it is identified so closely with traditional instructional methods, drill-and-practice software has become an icon for what many people consider an outmoded approach to teaching. Critics claim that introducing isolated skills and directing students to practice them contradicts the trend toward restructured curriculum in which students learn and use skills in an integrated way within the context of their own projects that specifically require the skills.

Despite these criticisms, it is likely that some form of drill-and-practice software will be useful in many classrooms for some time to come. Rather than ignoring drill-and-practice software or criticizing it as outmoded, teachers should seek to identify needs that drills can meet and use the software in ways that take advantage of its capabilities. See Figure 3.1 for examples of drill software and a summary of drill features.


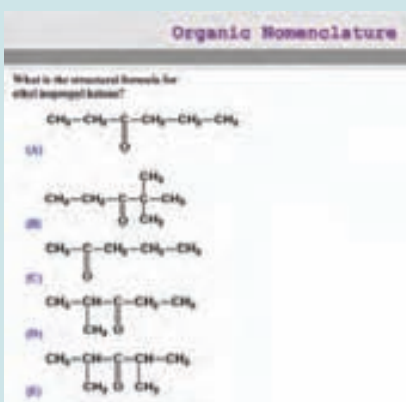

## Using Drill and Practice in Teaching

### *Classroom integration strategies for drill functions.*

Drill-and-practice programs may be used whenever teachers feel the need for on-paper exercises such as worksheets. On some occasions, even the most creative and innovative teacher may take advantage of the benefits of drill-and-practice software to give students practice using isolated skills. Integration strategies include:

- **Supplement or replace worksheets and homework exercises** — Whenever students have difficulty with higher order tasks ranging from reading and writing to mathematics, teachers may have to stop and identify specific prerequisite skills that these students lack and provide the instruction and practice they need to go forward. In these cases, learning may require a rehearsal activity to make sure information is stored in long-term memory so students can retrieve it easily. Drills' motivation, immediate feedback, and self-pacing can make it more productive for students to practice required skills on the computer than on paper.
- **Prepare for tests** — Despite the new emphasis on student portfolios and other authentic assessment measures, students can expect to take several kinds of objec-

**FIGURE 3.1** Drill-and-Practice Summary Information

Description of Drill and Practice		
<b>Characteristics</b> <ul style="list-style-type: none"> <li>• Presents items for students to answer</li> <li>• Gives feedback on correctness</li> <li>• Sometimes gives explanation of why answers are incorrect</li> </ul>	<b>Criteria for Effective Drill Software</b> <ul style="list-style-type: none"> <li>• User control over presentation rate</li> <li>• Good answer judging</li> <li>• Appropriate feedback for correct, incorrect answers</li> </ul>	<b>Benefits</b> <ul style="list-style-type: none"> <li>• Gives immediate, private feedback</li> <li>• Motivates students to practice</li> <li>• Saves teacher time correcting student work</li> </ul>
Sample Software with Drill-and-Practice Functions		
 <p><b>Earobics®</b> by Cognitive Concepts, Inc. <a href="http://www.earobics.com/">http://www.earobics.com/</a></p> <p>Practice in phonological awareness and other reading readiness skills.</p> <p>© 1997–2004 by Cognitive Concepts, Inc. All rights reserved.</p>	 <p><b>Organic Nomenclature®</b> by the <i>Journal of Chemical Education</i> <a href="http://www.jce.divched.org/">http://www.jce.divched.org/</a></p> <p>Practice in naming organic compounds and identifying structural formulas.</p> <p>Used with permission from the <i>Journal of Chemical Education</i>, Vol. 80, No. 10, 2003, pp. 1223–1224; copyright © 2003, Division of Chemical Education, Inc.</p>	 <p><b>MacGAMUT®</b> by Music Software International <a href="http://www.macgamut.com/">http://www.macgamut.com/</a></p> <p>Practice in recognizing intervals, scales, chords, melodic dictation, harmonic dictation, and rhythmic dictation.</p> <p>MacGAMUT 2003 for Mac and Windows. © 2003, MacGAMUT Music Software, Inc. <a href="http://www.macgamut.com/">http://www.macgamut.com/</a></p>

tive examinations in their education careers. When they need to prepare to demonstrate mastery of specific skills in important examinations (e.g., for end-of-year grades or for college entrance), drill-and-practice software can help them focus on their deficiencies and correct them. An example integration strategy for drill functions is shown in Technology Integration Lesson 3.2.

**Guidelines for using drill and practice.** Use the following guidelines to make best use of drill capabilities when designing integration strategies for drill-and-practice functions:

- **Set time limits** — Teachers should limit the time devoted to drill assignments to 10 to 15 minutes per day. This ensures that students will not become bored and that the drill-and-practice strategy will retain its effectiveness. Also, teachers should be sure students

have been introduced previously to the concepts underlying the drills. Drill software should serve mainly to debug and to help students retain their grasp of familiar concepts.

- **Assign individually** — Most drills are designed to allow self-pacing and personalized feedback. Therefore, these activities usually work best for individual computer use. However, some teachers with limited technology resources have found other, ingenious ways to capitalize on the motivational and immediate feedback capabilities of drills. If all students in a class benefit from practice in a skill using a drill program, the teacher may divide them into small groups to compete with each other for the best group scores. The class could even be divided into two groups for a “relay race” competition to see which group can complete the assignment the fastest with the most correct answers.



## Technology Integration Lesson 3.2

### Example Strategy for Drill-and-Practice Software

**Title:** Organic Nomenclature

**Grade Level:** High school

**Content Area/Topic:** Chemistry

**NETS for Students:** Standard 4 (Critical Thinking, Problem Solving, and Decision Making)

**Description of Standards Applications:** This integration idea offers practice in critical thinking and decision making as students strive to understand organic compounds and how they are represented by the formulas that show how their components combine. Using software that presents a name or formula, students need to select correct formulas or names.

**Instruction:** Begin by introducing the topic of organic compounds and how they are represented by the formulas that show how their components combine. Demonstrate to the whole class how to do these formulas for one or two compounds. Then ask students to respond to others as a whole-class exercise. Next, turn the problem around by presenting one or two formulas and asking students to name the compound. Since it is important for each student to be able to identify and do formulas for these compounds quickly as a prerequisite to further work on organic compounds, the software is designed for use by individual students. The software presents a name or a formula, and the student selects the correct formula or name from a list of five options. Students do a set of these, and the software gives them feedback on their correctness. They do as many sets as needed to master the concept. Both students and teachers can use these exercises to gauge their readiness to proceed with other work.

**Assessment:** Use a multiple-choice test formatted similar to items in software.

**Source:** From Shaw, D. B., & Yindra, L. (2003). Organic nomenclature. *Journal of Chemical Education*, 80 (10), 1223–1224. Used with permission from the *Journal of Chemical Education*; copyright © 2003, Division of Chemical Education, Inc. Software available from the *Journal of Chemical Education*, <http://www.jce.divched.org>.

- **Use learning stations** — If not all students need the kind of practice that a drill provides, the teacher may make software one of several learning stations to serve students with identified weaknesses in one or more key skills. Drill-and-practice functions are used best when matched to the specific learning needs of individual students.

## Tutorial Software Functions

Tutorial software is an entire instructional sequence on a topic, similar to a teacher's classroom instruction. This instruction usually is expected to be a self-contained instructional unit rather than a supplement to other instruction. Students should be able to learn the topic without any other help or materials. Unlike other types of instructional software, tutorials are true teaching materials. Gagné et al. (1981) said that good tutorial software should address all

nine instructional events. (See the discussion of Gagné's Events of Instruction in Chapter 2.)

People may confuse tutorial and drill activities for two reasons. First, drill software may provide elaborate feedback that reviewers may mistake for tutorial explanations. Even software developers may claim that a package is a tutorial when it is, in fact, a drill activity with detailed feedback. Second, a good tutorial should include one or more practice sequences to check students' comprehension. Since this is a drill-and-practice function, reviewers can become confused about the primary purpose of the activity.

Tutorials often are categorized as linear or branching tutorials (Alessi & Trollip, 2001), as described below:

- **Linear tutorial** — A simple, linear tutorial gives the same instructional sequence of explanation, practice, and feedback to all learners regardless of differences in their performance.



- **Branching tutorial** — A more sophisticated branching tutorial directs learners along alternate paths depending on how they respond to questions and whether they show mastery of certain parts of the material. Branching tutorials can range in complexity by the number of paths they allow and how fully they diagnose the kinds of instruction a student needs. More complex tutorials may also have computer-management capabilities; teachers can place each student at an appropriate level and get progress reports as each one goes through the instruction.

Tutorials are usually geared toward learners who can read fairly well and who are older students or adults. Since tutorial instruction is expected to stand alone, it is difficult to explain or give appropriate guidance on-screen to a non-reader. However, some tutorials aimed at younger learners have found clever ways to explain and demonstrate concepts with graphics, succinct phrases or sentences, or audio directions coupled with screen displays.

## Selecting Good Tutorial Software

Being a good teacher is a difficult assignment for any human, let alone a computer. However, software must accomplish this task to fulfill tutorial functions. In addition to meeting general criteria for good instructional software, well-designed tutorial programs should also meet the following standards:

- **Extensive interactivity** — Good tutorials, like good teachers, should require students to give frequent and thoughtful responses to questions and problems and should supply appropriate practice and feedback to guide students' learning. The most frequent criticism of tutorials is that they are "page-turners"—that is, they ask students to do very little other than read. Interactive tutorials have been shown to present cognitive benefits for learners (e.g., Schwan & Riempp, 2004).
- **Thorough user control** — User control refers to several aspects of a tutorial program. First, students should always be able to control the rate at which text appears on the screen. The program should not go on to the next information or activity screen until the user has pressed a key or has given some other indication of having completed the necessary reading. Next, the program should offer students the flexibility to review explanations, examples, or sequences of instruction or to move ahead to other instruction. The program should also provide frequent opportunities for students to exit the program if they like.
- **Appropriate pedagogy** — The program's structure should provide a suggested or required sequence of instruction that builds on concepts and covers the content adequately. It should provide sufficient explanation and examples in both original and remedial sequences. In

sum, it should compare favorably to an expert teacher's presentation sequence for the topic.

- **Adequate answer-judging and feedback capabilities** — Whenever possible, programs should allow students to answer in natural language and should accept all correct answers and possible variations of correct answers. They should also give appropriate corrective feedback when needed, supplying this feedback after only one or two tries rather than frustrating students by making them keep trying indefinitely to answer something they may not know.
- **Appropriate graphics** — Although some authors insist that graphics form part of tutorial instruction (Baek & Layne, 1988), others warn that graphics should be used sparingly and not interfere with the purpose of the instruction (Eiser, 1988). Where graphics *are* used, they should fulfill an instructional, aesthetic, or otherwise supportive function.
- **Adequate recordkeeping** — Depending on the purpose of the tutorial, teachers may need to keep track of student progress. If the program keeps records on student work, teachers should be able to get progress summaries quickly and easily.

## Benefits of Tutorials

Since a tutorial includes drill-and-practice activities, helpful features include the same ones as for drills (immediate feedback to learners, motivation, and time savings) plus the additional benefit of offering a self-contained, self-paced unit of instruction. Many successful uses of tutorials have been documented over the years. For examples, see Arnett (2000), CAI in Music (1994), Cann and Seale (1999), Graham (1994, 1998), Kraemer (1990), Murray et al. (1988), and Steinberg and Oberem (2000).

## Limitations and Problems Related to Tutorials

Tutorials can fulfill many much-needed instructional functions, but like drill and practice, they also attract their share of criticism, including:

- **Criticism by constructivists** — Constructivists criticize tutorials because they deliver directed instruction rather than allowing students to generate their own knowledge through hands-on projects. Thus, they feel tutorials are trivial uses of the computer.
- **Lack of good products** — Software publishers describe fewer packages as tutorials than any other kind of microcomputer software. This is partly due to the difficulty and expense of designing and developing tutorial software. A well-designed tutorial sequence emerges from extensive research into how to teach the

topic well. Designers must know what learning tasks the topic requires, the best sequence for students to follow, how best to explain and demonstrate essential concepts, common errors students are likely to make, and how to provide instruction and feedback to correct those errors. Therefore, programming and graphics can become fairly involved.

- **Reflect only one instructional approach** — Tutorial problems become still more difficult because teachers frequently disagree about what should be taught for a given topic, how to teach it most effectively, and in what order to present the learning tasks. A teacher may choose not to purchase a tutorial with a sound instructional sequence, for instance, because it does not cover the topic the way he or she presents it. Not surprisingly, software companies tend to avoid programs that are difficult to develop and market.

Although tutorials have considerable value and are popular in military and industrial training, schools and colleges have never fully tapped their potential as teaching resources. However, recent trends toward combining tutorial software with audiovisual media and distance education initiatives may bring tutorial functions into more common use. See Figure 3.2 for examples of tutorial software and a summary of tutorial features.

## Using Tutorials in Teaching

*Classroom integration strategies for tutorial functions.* Self-instructional tutorials should in no way threaten teachers, since few conceivable situations make a computer preferable to an expert teacher. However, the tutorial's unique capability of presenting an entire interactive instructional sequence can assist in several classroom situations:

- **Self-paced reviews of instruction** — Students often need to repeat instruction on a topic after the teacher's initial presentation. Some students may be slower to understand concepts and need to spend additional time on them. Others may learn better in a self-paced mode without the pressure to move at the same pace as the rest of the class. Still others may need a review before a test. Tutorials can provide self-paced instruction to address all these needs.
- **Alternative learning strategies** — Some students, typically those at advanced levels, prefer to structure their own learning activities and proceed at their own pace. With a good tutorial, advanced students can glean much background material prior to meeting with a teacher or others to do assessment and/or further work assignments.
- **Instruction when teachers are unavailable** — Some students have problems when they surge ahead of their class. The teacher cannot leave the rest of the class to provide the instruction that such an advanced student

needs. It is also true that many schools, especially those in rural areas, may not offer certain courses because they cannot justify the expense of hiring a teacher for the comparatively few students who need physics, German, trigonometry, or other lower demand courses. Well-designed tutorial courses, especially in combination with other methods such as distance learning, can help meet these students' needs.

*Guidelines for using tutorials.* Use the following guidelines to make the best use of tutorial capabilities when designing integration strategies for tutorial functions:

- **Assign individually** — Like drill-and-practice functions, tutorial functions are designed for use by individuals rather than by groups of students.
- **Use learning stations or individual checkout** — Depending on which of the above strategies it promotes, a tutorial may be used in a classroom learning station or may be available for checkout at any time in a library/media center. Sometimes teachers send students to learning stations with tutorials in order to review previously presented material while the teacher works with other students. (See Technology Integration Lesson 3.3.)



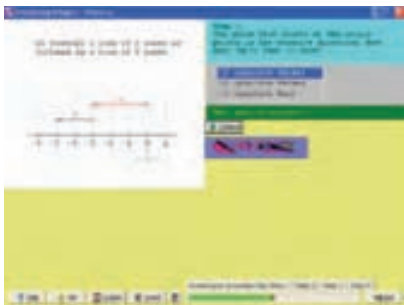
## Simulation Software Functions

A simulation is a computerized model of a real or imagined system that is designed to teach how the system works. Unlike tutorial and drill-and-practice activities, in which the teaching structure is built into the package, learners using simulations usually must choose tasks to do and the order in which to do them. Alessi and Trollip (2001) identify two main types of simulations: those that teach *about* something and those that teach *how to do* something. They further divide the "about" simulations into physical and iterative types and the "how to" simulations into procedural and situational types.



**Simulations can be used instead of or in conjunction with certain science activities such as dissections.**

**FIGURE 3.2** Tutorial Summary Information

Description of Tutorial Software		
<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Presents an entire instructional sequence</li> <li>• Is complete, rather than supplemental, instruction</li> <li>• Includes drill-and-practice functions</li> <li>• Can be either linear or branching</li> </ul>	<p><b>Criteria for Effective Tutorial Software</b></p> <ul style="list-style-type: none"> <li>• Extensive interactivity</li> <li>• Thorough user control</li> <li>• Appropriate pedagogy</li> <li>• Adequate answer judging and feedback</li> <li>• Appropriate graphics</li> <li>• Adequate record keeping</li> </ul>	<p><b>Benefits</b></p> <ul style="list-style-type: none"> <li>• Same as drill and practice (immediate, private feedback, time savings)</li> <li>• Offers instruction that can stand on its own</li> </ul>
Sample Software with Tutorial Functions		
 <p><b>PhysicaElementa®</b> by Intellectum Plus, Inc. <a href="http://www.mathandscience4u.com/">http://www.mathandscience4u.com/</a></p> <p>Instruction in states of matter: A sequence of screens gives explanations, descriptions, and animated examples of solids, liquids, and gases.</p> <p>© Intellectum Plus, Inc. Used with permission.</p>	 <p><b>Congress for Kids®</b> Sponsored by the Dirksen Congressional Center <a href="http://www.congressforkids.net">http://www.congressforkids.net</a></p> <p>Instruction in various aspects of U.S. government: Sequences of screens give explanations of how government works, with assessment items to review concepts and check comprehension.</p> <p>© The Dirksen Congressional Center, Pekin, Illinois. Used with permission.</p>	 <p><b>Basic Algebra Shapeup®</b> by Merit Software, Inc. <a href="http://www.meritsoftware.com/">http://www.meritsoftware.com/</a></p> <p>Instruction in algebra concepts: A series of screens explains a concept, then assessment items check students' knowledge.</p> <p>© Merit Software, Inc. Used with permission.</p>

## Simulations That Teach About Something

- **Physical simulations** — These simulations allow users to manipulate things or processes represented on the screen. For example, students might see selections of chemicals with instructions on how to combine them to see the result, or they might see how various electrical circuits operate. More recent investigations of simulation software include the use of three-dimensional models (e.g., Kim, 2006).
- **Iterative simulations** — These simulations speed up or slow down processes that usually happen either so slowly or so quickly that students cannot see the events unfold. For example, software may show the effects of changes in demographic variables on population growth or the effects of environmental factors on ecosystems. Alessi and Trollip (2001) refer to this type as

“iterative” because students can run it over and over again with different values, observing the results each time. Biological simulations, such as those on genetics, are popular since they help students experiment with natural processes. Genetics simulations let students pair animals with given characteristics and see the resulting offspring.

## Simulations That Teach How to Do Something

- **Procedural simulations** — These activities teach the appropriate sequences of steps to perform certain procedures. They include diagnostic programs, in which students try to identify the sources of medical or mechanical problems, and flight simulators, in which students simulate piloting an airplane or other vehicle.





## Technology Integration Lesson 3.3

### Example Strategy for Tutorial Software

**Title:** The *ActivStats* Statistics Tutorial

**Grade Level:** High school

**Content Area/Topic:** Advanced placement statistics

**NETS for Students:** Standards 3 (Research and Information Fluency) and 4 (Critical Thinking, Problem Solving, and Decision Making)

**Description of Standards Applications:** This integration idea offers the opportunity to research raw data, analyze the data, and then interpret the findings through an interactive statistics program, *ActivStats*. The *ActivStats* tutorial software has 26 lessons that cover the basics of introductory statistics.

**Instruction:** A teacher can use the short animated clips at the beginning of each lesson to introduce the topic to the whole class. The teacher continues the introduction with a review of the lesson's basic concepts by using an animation to illustrate the uses of the statistic and how changing the raw data can alter it. After this demo, students work individually on the remaining lesson activities. For example, most lessons allow the user to import pre-existing data files and provide instructions on how to plot them in a chart or in three dimensions.

**Assessment:** Students use the built-in quizzes to gauge their comprehension and determine if they are ready to take the teacher's final test on the topic.

**Source:** Based on Ashbacher, C. (2003). *ActivStats*. *Mathematics and Computer Education*, 37(2), 254–255. Reprinted with permission of *Mathematics and Computer Education* journal. Software available from Addison-Wesley, <http://www.aw-bc.com/activstats/>.

- **Situational simulations** — These programs give students hypothetical problem situations and ask them to react. Some simulations allow for various successful strategies, such as letting students play the stock market or operate businesses. Others have most desirable and least desirable options, such as choices when encountering a potentially volatile classroom situation.

The preceding descriptions serve to clarify the various forms a simulation might take, but teachers need not feel they should be able to classify a given simulation into one of these categories. Simulations usually emphasize learning about the system itself rather than learning general problem-solving strategies. For example, a program called *The Factory* has students build products by selecting machines and placing them in the correct sequence. Since the program emphasizes solving problems in correct sequence rather than manufacturing in factories, it should probably be called a problem-solving activity rather than a simulation. Programs such as *SimCity*<sup>TM</sup> (Electronic Arts), which let students design their own cities, provide more accurate examples of building-type simulations (Adams, 1998). (See Technology Integration Lesson 3.4 for a classroom example using *SimCity*.)

### Selecting Good Simulation Software

Simulations vary in type and purpose, so a uniform set of criteria is not possible. For some simulations, a realistic and accurate representation of a system is essential (Reigeluth & Schwartz, 1989), but for others, it is important only to know what the screen elements represent. Since the screen often presents no set sequence of steps, simulations need good accompanying documentation—more than most software. A set of clear directions helps the teacher learn how to use the program and show the students how to use it rapidly and easily. See Figure 3.3 for examples of simulation software and a summary of simulation features.

### Benefits of Simulations

The field of science seems to include more simulations than any other area (Andaloro, 1991; Lunce & Bailey, 2007; Mintz, 1993; Richards, 1992; Ronen, 1992; Simmons & Lunetta, 1993; Smith, 1992), but the use of simulations is also popular in teaching social science topics (Adams, 1998; Allen, 1993; Clinton, 1991; Estes, 1994). However, more simulations are currently in development that feature on-line delivery or supplements to combine the control, safety,



## Technology Integration Lesson 3.4

### Example Strategy for Simulation Software

**Title:** Community Planning Projects with *SimCity*™

**Grade Level:** Middle school

**Content Area/Topic:** Social studies—citizenship/group cooperation in social projects

**NETS for Students:** Standards 1 (Creativity and Innovation), 2 (Communication and Collaboration), 3 (Research and Information Fluency), 4 (Critical Thinking, Problem Solving, and Decision Making), and 5 (Digital Citizenship)

**Description of Standards Applications:** This integration idea offers the opportunity for students to work together to develop a city using the *SimCity*™ software. Students will use their creativity as they research the many alternatives for developing a city that has the qualities of an environment where they and others would like to live. This lesson helps raise students' awareness of their responsibility to become informed citizens, shows how they can participate in local decision making, and gives practice in working cooperatively with a group to carry out a social project. The software allows the development of a comprehensive plan for the township.

**Instruction:** A representative from the county planning and zoning office talks with students about factors of concern to community development. The teacher then introduces students to the *SimCity*™ software. They form teams of four to five students each and start their own community planning projects with *SimCity*. Each group meets every week to discuss and develop its plan. As a group, they decide how to select and place features such as roads, homes, and utilities. After recording their decisions on paper, they enter them into the program and observe the results. The groups discuss feedback that the program provides on areas such as taxes, crime rates, and public opinion. After the groups' plans are complete, each group presents its plan to the teacher and explains and defends its choices.

**Assessment:** Papers are graded by a rubric.

**Source:** From Jacobson, P. (1992). Save the cities! *SimCity* in grades 2–5. *The Computing Teacher*, 20 (2), 14–15.

and interactive features of computer simulations with the visual impact of pictures of real-life devices and processes. Depending on the topic, a simulation can provide one or more of the following benefits (Alessi & Trollip, 2001):

- **Compress time** — This feature is important whenever students study the growth or development of living things (e.g., pairing animals to observe the characteristics of their offspring) or other processes that take a long time (e.g., the movement of a glacier). A simulation can make something happen in seconds that normally takes days, months, or longer, so that students can cover more variations of the activity in a shorter time.
- **Slow down processes** — Conversely, a simulation can also model processes normally invisible to the human eye because they happen so quickly. For example, physical education students can study the slowed-down

movement of muscles and limbs as a simulated athlete throws a ball or swings a golf club.

- **Get students involved** — Simulations can capture students' attention by placing them in charge of things and asking, "What would you do?" The results of their choices can be immediate and graphic. Users can also interact with the program instead of just seeing its output.
- **Make experimentation safe** — Whenever learning involves physical danger, simulations are the strategy of choice. This is true when students are learning to drive vehicles, handle volatile substances, or react to potentially dangerous situations. They can experiment with strategies in simulated environments that might result in personal injury to themselves or others in real life. For example, the First Responders Simulation and Training Environment (FiRSTE) allows for the training of civilian

**FIGURE 3.3** Simulation Summary Information

Description of Simulation Software		
<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>Models a real or imaginary system</li> <li>Can model physical phenomena (e.g., growth), procedures (e.g., dissections), and hypothetical situations (e.g., stock market)</li> <li>Users can see the impact of their actions</li> </ul>	<p><b>Criteria for Effective Simulation Software</b></p> <ul style="list-style-type: none"> <li>System fidelity and accuracy (for some simulations)</li> <li>Good documentation to explain system characteristics and uses</li> </ul>	<p><b>Benefits</b></p> <ul style="list-style-type: none"> <li>Compresses time or slows down processes</li> <li>Gets students involved</li> <li>Makes experimentation safe</li> <li>Makes the impossible possible</li> <li>Saves money and other resources</li> <li>Allows repetition with variations</li> <li>Allows observation of complex processes</li> </ul>
Sample Software with Simulation Functions		
 <p><b>Virtual Labs: Electricity®</b> by Riverdeep, Inc. <a href="http://web.riverdeep.net">http://web.riverdeep.net</a></p> <p>Simulated electrical circuit board: Provides a safe means for students to apply electrical concepts to simulated devices.</p> <p>© 2004 Riverdeep Interactive Learning Limited, and its Licensors. All rights reserved.</p>	 <p><b>Oregon Trail®</b> by The Learning Company <a href="http://www.learningcompany.com">http://www.learningcompany.com</a></p> <p>Simulated trip in the Old West of the 1890s: Students learn about pioneer days by taking roles in a simulated wagon train journey.</p> <p>© 2004 Riverdeep Interactive Learning Limited, and its Licensors. All rights reserved.</p>	 <p><b>SimCity 3000®</b> by Electronic Arts, Inc. <a href="http://www.maxis.com/">http://www.maxis.com/</a></p> <p>Simulated city: Lets users build their own cities, create a budget for them, populate them, and run them, including responding to intermittent disasters.</p> <p>SimCity 3000 © 1999 Electronic Arts Inc. SimCity and SimCity 3000 are trademarks or registered trademarks of Electronic Arts Inc. in the U.S. and/or other countries. All rights reserved.</p>

first responders to respond to attacks employing weapons of mass destruction (Tichon et al., 2003).

- **Make the impossible possible** — Very often, teachers simply cannot give students access to the resources or situations that simulations can. Simulations can show students, for example, what it would be like to walk on the moon or to react to emergencies in a nuclear power plant. They can see cells mutating or hold countrywide elections. They can even design new societies or planets and see the results of their choices. For example, one researcher at Davis (UCDavis) re-created a mental health treatment ward in a virtual world and gave each of his

students a taste of what it means to experience schizophrenia in the real world. As students' virtual characters walked the hallways, they were overcome by hallucinations including "the floor disappearing from underfoot, writing on posters that morphs into derogatory words, a pulsating gun that suddenly appears on a table, and menacing voices that laugh" (UCDavis, 2007).

- **Save money and other resources** — Many school systems are finding dissections of animals on a computer screen to be much less expensive and just as instructional as using real frogs or cats. (It is also easier on the animals!) Depending on the subject, a simulated experiment may



be just as effective a learning experience as an actual experiment is, at a fraction of the cost.

- **Allow repetition with variations** — Unlike in real life, simulations let students repeat events as many times as they wish and with unlimited variations. They can pair any number of cats or make endless spaceship landings in a variety of conditions to compare the results of each set of choices.
- **Allow observation of complex processes** — Real-life events often are so complex that they are confusing—especially to those seeing them for the first time. When many things happen at once, students find it difficult to focus on the operation of individual components. Who could understand the operation of a stock market by looking at the real thing without some introduction? Simulations can isolate parts of activities and control background noise. This makes it easier for students to see what is happening when, later, all the parts come together in the actual activity.

## Limitations and Problems Related to Simulations

Most educators acknowledge the instructional usefulness of simulations. There are some concerns, however, which include the following:

- **Accuracy of models** — When students see simplified versions of systems in a controlled situation, they may get inaccurate or imprecise perspectives on the systems' complexity. For example, students may feel they know all about how to react to driving situations because they have experienced simulated versions of them. Many educators feel strongly that situational simulations must be followed at some point by real experiences. In addition, many teachers of very young children feel that learners at early stages of their cognitive development should experience things first with their five senses rather than on computer screens.
- **Misuse of simulations** — Sometimes, simulations are used to teach concepts that could just as easily be demonstrated on paper, with manipulatives, or with real objects. For example, students usually are delighted with the simulation of the food chain called *Odell Lakes*, a program that lets students see which animals prey on other animals in a hypothetical lake. However, some educators wonder whether such a computer simulation is necessary or even desirable for teaching this concept. Hasselbring and Goin (1993) point out that students often can master the activities of a simulation without actually developing effective problem-solving skills; on the contrary, such applications actually can encourage counterproductive behaviors. For

example, some simulations initially provide little information with which to solve problems, and students are reduced to “trial-and-error guessing rather than systematic analysis of available information” (p. 156). Teachers must structure integration strategies carefully so that students will not use simulations in inappropriate ways.

## How to Use Simulations in Teaching: Integration Strategies and Guidelines

Simulations are considered among the most potentially powerful computer software resources; as with most software, however, their usefulness depends largely on the program's purpose and how well it fits in with the purpose of the lesson and student needs. Teachers are responsible for recognizing the unique instructional value of each simulation and for using it to its best advantage.

*Classroom applications of simulation functions.* Real systems are usually preferable to simulations, but a simulation is useful when the real situation is too time consuming, dangerous, expensive, or unrealistic for a classroom presentation. Simulations should be considered in the following situations:

- **In place of or as supplements to lab experiments** — When adequate lab materials are not available, teachers should try to locate computer simulations of the required experiments. Many teachers find that simulations offer effective supplements to real labs, either to prepare students for making good use of the actual labs or as follow-ups with variations of the original experiments without using consumable materials. Some simulations actually allow users to perform experiments that they could not otherwise manage or that would be too dangerous for students.
- **In place of or as supplements to role-playing** — Many students either refuse to role play in front of a class or get too enthusiastic and disrupt the classroom. Computerized simulations can take the personal embarrassment and logistical problems out of the learning experience, make classroom role playing more controllable, and spark students' imagination and interest in the activities.
- **In place of or as supplements to field trips** — Seeing an activity in its real setting can be a valuable experience, especially for young children. Sometimes, however, desired locations are not within reach of the school, and a simulated experience of all or part of the process is the next best thing. As with labs, simulations provide good introductions or follow-ups to field trips.

- **Introducing and/or clarifying a new topic** — Software that allows students to explore the elements of an environment in a hands-on manner frequently provides students' first in-depth contact with a topic. This seems to accomplish several purposes. First, it is a non-threatening way to introduce new terms and unfamiliar settings. Students know they are not being graded, so they feel less pressure than usual to learn everything right away. A simulation can be simply a get-acquainted look at a topic; it can also build students' initial interest in a topic. Highly graphic, hands-on activities draw them in and whet their appetite to learn more. Finally, some software helps students see how certain prerequisite skills relate to the topic; this may motivate students more strongly to learn the skills than if the skills were introduced in isolation from the problems to which they apply. For example, Tom Snyder's *Decisions! Decisions!* software helps students see the relevance of topics such as the U.S. Constitution and elections.
- **Fostering exploration and process learning** — Teachers often use content-free simulation/problem-solving software (e.g., *The Factory*) as motivation for students to explore their own cognitive processes. Since this kind of software requires students to learn no specific content, it is easier to get them to concentrate on problem-solving steps and strategies. However, with content-free products, it is even more important than usual that teachers draw comparisons between the skills used in the software activities and those in the content areas to which they want to transfer the experience. For example, *Virtual Lab* (Riverdeep) presents an implicit emphasis on science-process skills that the teacher may want to point out. It seems best to use these kinds of activities just prior to content-area activities that will require the same processes.
- **Encouraging cooperation and group work** — Sometimes a simulated demonstration can capture students' attention quickly and effectively and interest them in working together on a product. For example, a simulation on immigration or colonization might be the "grabber" a teacher needs to launch a group project in a social studies unit.

**Guidelines for using simulation functions.** Simulations offer more versatile implementation than tutorials or drills do. They usually work equally effectively with a whole class, small groups, or individuals. A teacher may choose to introduce a lesson to the class by displaying a simulation or to divide the class into small groups and let each of them solve problems. Because they instigate discussion and collaborative work so well, simulations usually are considered more

appropriate for pairs and small groups than for individuals. However, individual use certainly is not precluded.

## Instructional Game Software Functions

Technology-based games bridge the worlds of gaming, entertainment, and education in an attempt to deliver fun and effective learning. Simply defined, instructional games add game-like rules and/or competition to learning activities. Even though teachers often use them in the same way as they do drill-and-practice or simulation software, games usually are listed as a separate software activity because their instructional connotation to students is slightly different. When students know they will be playing a game, they expect a fun and entertaining activity because of the challenge of the competition and the potential for winning (Gee, 2004; Raessens & Goldstein, 2005; Randel, Morris, Wetzel, & Whitehill, 1992; Squire, 2005). Teachers frequently intersperse games with other activities to hold students' attention or as a reward for accomplishing other activities, even though games, by themselves, could be powerful teaching tools.

It is important to recognize the common characteristics that set instructional games apart from other types of software: game rules, elements of competition or challenge, and amusing or entertaining formats. These elements generate a set of mental and emotional expectations in students that make game-based instructional activities different from nongame ones.

## Selecting Good Instructional Games

Since instructional games often amount to drills or simulations overlaid with game rules, these three types of software often share many of the same criteria (e.g., better reinforcement for



**Instructional games are one way to engage students in learning.**

correct answers than for incorrect ones). Teachers should use the following criteria in choosing instructional games:

- **Appealing formats and activities** — When Malone (1980) examined the evidence on what makes things fun to learn, he found that the most popular games included elements of adventure and uncertainty and levels of complexity matched to learners' abilities.
- **Instructional value** — Teachers should examine instructional games carefully for their value as both educational and motivational tools. While a number of researchers argue for the benefits of educational games (e.g., Gee, 2004; Squire, 2005), others question their value (e.g., Clark, 2007).
- **Physical dexterity is reasonable** — Teachers should ensure that students will be motivated rather than frustrated by the activities.
- **Social, societal, and cultural considerations** — Games may be inappropriate for children if they are not designed with a respectful outlook. For instance, games that call for violence or combat require careful screening, not only to avoid students' modeling this behavior, but also because girls often perceive the attraction of these activities differently than boys do. In addition, games may present females and various ethnic and cultural groups in stereotypical roles. Ideally, teachers should choose games that do not perpetuate stereotypes, while at the same time highlighting positive messages (e.g., peace and friendship) rather than unnecessary violence (e.g., aggression).

## Benefits of Instructional Games

A classroom without elements of games and fun would be a dry, barren landscape for students to traverse. In their review of the effectiveness of games for educational purposes, Randel et al. (1992) found "[the fact] that games are more interesting than traditional instruction is both a basis for using them as well as a consistent finding" (p. 270). They also observed that retention over time favors the use of simulations/games. Successful uses of games have been reported in many content areas (Flowers, 1993; Muckerheide, Mogill, & Mogill, 1999; Trotter, 1991). The appeal of games seems to center around students' desire to compete and play. Games provide teachers with opportunities for taking advantage of this innate desire to get students to focus on a curriculum topic. See Figure 3.4 for examples of instructional game software and a summary of game features.

## Limitations and Problems Related to Instructional Games

Some teachers believe that any time they can sneak in learning under the guise of a game, it is altogether a good thing

(McGinley, 1991). However, games are also frequently criticized from several standpoints:

- **Learning versus having fun** — Some schools forbid any use of games because they believe games convince students that they are escaping from learning, thus drawing attention away from the intrinsic value and motivation of learning. Critics also feel that winning the game becomes a student's primary focus and that the instructional purpose is lost in the pursuit of this goal. Observers disagree about whether getting lost in the game is a benefit or a problem.
- **Confusion of game rules and real-life rules** — Some teachers have observed that students can become confused about which part of the activity is the game and which part is the skill; they may then have difficulty transferring their skill to later nongame situations. For example, the teacher's manual for Sunburst's *How the West Was One + Three × Four* reminds teachers that some students can confuse the math operations rules with the game rules and that teachers must help them recognize the need to focus on math rules and use them outside the game. Recent studies seem to indicate that instructional games can be useful in fostering higher order skills but that their usefulness hinges on how teachers employ them (DiPietro, Ferdig, Boyer, & Black, 2007; Henderson, Klemes, & Eshet, 2000; Rieber, Smith, & Noah, 1998).
- **Inefficient learning** — Although students obviously find many computer games exciting and stimulating, it is sometimes difficult to pinpoint their educational value. Teachers must try to balance the motivation that instructional games bring to learning against the classroom time they take away from non-game strategies. For example, students may become immersed in the challenge of the *Carmen Sandiego* series, but more efficient ways to teach geography may be just as motivating.

## Using Instructional Games in Teaching

*Classroom applications for instructional games.* Several kinds of instructional opportunities invite teachers to take advantage of the motivational qualities of games. Instructional games should be considered in the following situations:

- **In place of worksheets and exercises** — As with drill-and-practice software, teachers can use games to help students acquire automatic recall of prerequisite skills.
- **To teach cooperative group working skills** — Like simulations, many instructional games serve as the basis for or introduction to group work. In addition, some games can be played collaboratively over the Internet (e.g., via an Internet-enabled game console). A game's competitive qualities can present opportunities for competition among groups. (See Technology Integration Lesson 3.5.)



**FIGURE 3.4** Instructional Game Summary Information

Description of Instructional Game Software		
<p><b>Characteristics</b></p> <ul style="list-style-type: none"> <li>• Opportunities for content skill practice or problem solving in a fun, entertaining environment</li> <li>• Has game rules</li> <li>• Challenges students to compete and win</li> </ul>	<p><b>Criteria for Effective Instructional Game Software</b></p> <ul style="list-style-type: none"> <li>• Appealing formats and activities</li> <li>• Obvious instructional, as opposed to entertainment, value</li> <li>• Reasonable required levels of physical dexterity</li> <li>• Minimal violence/aggression</li> </ul>	<p><b>Benefits</b></p> <ul style="list-style-type: none"> <li>• Fun activities motivate students to spend more time on the topic</li> </ul>
Sample Software with Instructional Game Functions		
 <p><b>Arthur's<sup>®</sup> Math Games<sup>™</sup></b> by The Learning company. <a href="http://www.learningcompany.com">http://www.learningcompany.com</a></p> <p>Game to practice math skills: Students do adventures that require essential math skills in counting, logic, geometry.</p> <p>© 2001 Riverdeep Interactive Learning Limited, and its Licensors. All rights reserved.</p>	 <p><b>Where in the USA Is Carmen Sandiego?<sup>®</sup></b> by The Learning company. <a href="http://www.learningcompany.com">http://www.learningcompany.com</a></p> <p>Mystery game to study geography/history: Students answer geography and history questions that yield clues as to where the mystery character (Carmen) is hiding.</p> <p>© 2001 Riverdeep Interactive Learning Limited, and its Licensors. All rights reserved.</p>	 <p><b>Alice in Vivaldi's Four Seasons</b> by Kids Music Stage <a href="http://www.kidsmusicstage.com/">http://www.kidsmusicstage.com/</a></p> <p>Adventure game to practice music skills: Students must answer puzzles with music questions to free Alice, who is trapped in the magical music clock.</p> <p>© Music Games International. Used with permission.</p>

- **As a reward** — Perhaps the most common use of games is to reward good work. This may be a valid role for instructional software, but teachers should avoid overuse of it. Otherwise, the game can lose its motivational value and become an “electronic babysitter.” In addition, using games as rewards disregards the power of games to be teachable software, limiting them to a behaviorist tool. Some schools actually bar games from classrooms for fear that they overemphasize the need for students to be entertained.

**Guidelines for using instructional games.** Use the following guidelines to make the best use of game capabilities when designing integration strategies for game functions:

- **Use appropriately** — Some educators believe that games—especially computer-based ones—are overused, misused, and used inappropriately. Use games appropriately so that students effectively learn from them while they continue to stay motivated by the game play.
- **Involve all students** — Make sure that girls and boys alike are participating and that all students have a meaningful role.
- **Emphasize the content-area skills** — Before students begin playing, make sure they know the relationship between game rules and content-area (e.g., math) rules. Students should recognize which rules they will be using in their later work and which are merely part of the game environment.



## Technology Integration Lesson 3.5

### Example Strategy for Instructional Game Software

**Title:** Problem Solving and Geography

**Grade Level:** Elementary to high school

**Content Area/Topic:** Geography

**NETS for Students:** Standards 1 (Creativity and Innovation), 2 (Communication and Collaboration), 3 (Research and Information Fluency), and 4 (Critical Thinking, Problem Solving, and Decision Making)

**Description of Standards Applications:** This integration idea offers the opportunity for students to work, compete, and collaborate as they use the software *Carmen Sandiego* to maneuver through towns and track a thief. It is the research with the atlas software and the creativity and decision making of the teams that will illuminate the winners. *Carmen Sandiego* (The Learning Center) can be used as the basis for learning geography and research skills. The game becomes a more powerful activity for improving research and reference skills when it is paired with use of an online atlas.

**Instruction:** The class is divided into teams, and each team member is assigned a role: field operative (runs the game software), researcher (runs the atlas software), or recorder (records information and coordinates team activities). Members switch roles every class period to give everyone a turn at each. A bar graph is posted in the classroom to show how many “cases” each class has solved. Within the class, teams are encouraged to help and coach each other. Thus, competition is fostered between classes, but collaboration is fostered within classes. To guide their work, the students receive data sheets and maps similar to those provided with the game software. The recorder uses the map to circle each city the team “goes through” and draws a line indicating the direction they traveled from one city to the next. This information is recorded on another sheet, which the researcher uses to try to find the next location to track the thief. At the end of each case, information is filled in on a matrix on a final sheet.

**Assessment:** The teacher checks the final sheet to grade each group’s work.

**Source:** Greenman, M. (1991). A computer-based problem solving geography unit. *The Computing Teacher*, 18(2), 22–24.

## Problem-Solving Software Functions

Although simulations and instructional games are often used to help teach problem-solving skills, problem-solving software is designed especially for this purpose (Hmelo-Silver, 2004). Problem-solving software functions may focus on fostering component skills in or approaches to general problem-solving ability, or it may provide opportunities to practice solving various kinds of content-area problems (e.g., Doering & Veletsianos, 2007). However, defining the activity of problem solving seems elusive. One way to think about problem solving is through three of its most important components (Sherman 1987–1988): recognition of a goal (an opportunity for solving a problem), a process (a sequence of physical activities or operations), and mental activity (cognitive operations to pursue a solution).

Problem solving covers a wide variety of desired component behaviors. The literature mentions such varied subskills

for problem solving as metacognition, observing, recalling information, sequencing, analyzing, finding and organizing information, inferring, predicting outcomes, making analogies, and formulating ideas. Although there are many opinions about the proper role of instructional software in fostering these abilities, there seem to be two main approaches:

- **Content-area skills** — Some problem-solving software focuses on teaching content-area skills, primarily in mathematics and science. For example, *The Geometric Supposer* by Sunburst encourages students to learn strategies for solving geometry problems by drawing and manipulating geometric figures. Others, like *Alien Rescue* developed by the University of Texas, are what might be called problem-solving “environments.” These complex, multifaceted packages offer a variety of tools that allow students to create solutions to science-related problems presented by a scenario. Still others provide opportunities to practice solving specific kinds of math or science problems.



## Making the Case for Technology Integration

Use the following questions to reflect on issues in technology integration and guide discussions within your class.

1. Drill-and-practice software is often referred to by the derogatory term “drill and kill.” Do you believe this is because the number of situations is diminishing in which drill-and-practice software would be the strategy of choice or because people fail to recognize the appropriate situations for using it? Explain your reasoning.
2. Some schools, such as those with a college preparatory focus, do not allow the use of instructional games of any kind. Is there a compelling case to be made for allowing the use of instructional game software to achieve specific educational goals? That is, can games do something in an instructional situation that no other strategy is able to do? If so, what? Can you give examples?

- **Content-free skills** — Some educators feel that general problem-solving ability can be taught directly by specific instruction and practice in its component strategies and subskills (e.g., recalling facts, breaking a problem into a sequence of steps, or predicting outcomes). Others suggest placing students in problem-solving environments and, with some coaching and guidance, letting them develop their own heuristics for attacking and solving problems.

Although the purposes of the two views overlap somewhat, the first is directed more toward supplying prerequisite skills for specific kinds of problem solving whereas the second view aims more toward motivating students to attack problems and to recognize problem solving as an integral part of everyday life.

## Selecting Good Problem-Solving Software

Qualities to look for in good problem-solving software depend on the purpose of the software. In general, problem formats should be interesting and challenging, and software should have a clear link to developing a specific problem-solving ability. Software documentation should state clearly which specific problem-solving skills students will learn and how the software fosters them. See Figure 3.5 for examples of problem-solving software and a summary of problem-solving features.

## Benefits of Problem-Solving Software

Problem-solving software can help students in the following ways:


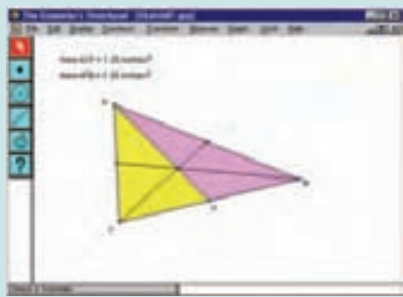
- **Improved interest and motivation** — Students are more likely to practice solving problems in activities they find interesting and motivating. Some educators also feel that students will become more active, spontaneous problem solvers if they experience success in their initial problem-solving efforts.
- **Prevents inert knowledge** — Content-area problem-solving environments can make knowledge and skills more meaningful to students because they illustrate how and where information applies to actual problems. Students learn both the knowledge and its application at the same time. Also, students gain opportunities to discover concepts themselves, which they frequently find more motivating than being told or, as constructivists might say, being programmed with, the information (McCoy, 1990).

## Limitations and Problems Related to Problem-Solving Software

Problem-solving software packages are among the most popular of all software functions; however, the following issues are still of concern to educators:

- **Names versus skills** — Software packages use many terms to describe problem solving, and their exact meanings are not always clear. Terms that appear in software catalogs as synonyms for problem solving include thinking skills, critical thinking, higher level thinking, higher order cognitive outcomes, reasoning, use of logic, and decision making. In light of this diversity of language, teachers must identify the skills that a software package addresses by looking at its activities. For example, for a software package that claims to teach inference skills, one would have to see how it defines *inference* by examining the tasks it presents, which may range from determining the next number in a sequence to using visual clues to predict a pattern.
- **Software claims versus effectiveness** — It would be difficult to find a software catalog that did not claim that its products foster problem solving, yet few publishers of software packages that purport to teach specific problem-solving skills have data to support their claims. When students play a game that requires skills related to problem solving, they do not necessarily learn these skills. They may enjoy the game thoroughly—and even be successful at it—without learning any of the intended skills. Teachers may have to use problem-solving software themselves to confirm that it achieves the results they want.
- **Possible negative effects of directed instruction** — Some researchers believe that direct attempts to teach problem-solving strategies can actually be counterproductive for some students. Mayes (1992) reports on studies that found “teaching sequenced planning to solve prob-

**FIGURE 3.5** Problem-Solving Summary Information

Description of Problem-Solving Software		
<p><b>Characteristics</b></p> <p>Four different types:</p> <ol style="list-style-type: none"> <li>1. Tools to help students solve problems</li> <li>2. Environments that challenge students to create solutions to complex problems</li> <li>3. Problems to help develop component problem-solving skills (e.g., recalling facts, following a sequence)</li> <li>4. Opportunities for practice in solving content-area problems</li> </ol>	<p><b>Criteria for Effective Problem-Solving Software</b></p> <ul style="list-style-type: none"> <li>• Challenging, interesting formats</li> <li>• Clear links to developing specific problem-solving skills or abilities</li> </ul>	<p><b>Benefits</b></p> <ul style="list-style-type: none"> <li>• Challenging activities motivate students to spend more time on the topic</li> <li>• Prevents inert knowledge by illustrating situations in which skills apply</li> </ul>
Sample Software with Problem-Solving Functions		
 <p><b>Alien Rescue®</b> by The Alien Rescue Team <a href="http://www.alienrescue.com">http://www.alienrescue.com</a></p> <p>Challenge activity to teach scientific inquiry: Students must use software tools and information to locate new home planets in the solar system to match the needs of various aliens.</p> <p>Screens from <i>Alien Rescue Software</i> © The Alien Rescue Team 2001–2004</p>	 <p><b>The Geometer's Sketchpad®</b> by Key Curriculum Press, Inc. <a href="http://www.keypress.com/sketchpad">http://www.keypress.com/sketchpad</a></p> <p>A dynamic construction and exploration tool: Students use software features to draw objects, investigate their mathematical properties, analyze problems, propose solutions, and do proofs to test their hypotheses.</p> <p>The Geometer's Sketchpad®, Key Curriculum Press, 1150 65th Street, Emeryville, CA 94608, 1-800-995-MATH, <a href="http://www.keypress.com/sketchpad">www.keypress.com/sketchpad</a></p>	

lems to high-ability learners could interfere with their own effective processing” (p. 243). In a review of research on problem solving in science, Blosser (1988) also found indications that problem-solving instruction may not have the desired results if the instructional strategy does not suit certain kinds of students. For example, students with high math anxiety and low visual preference or proportional reasoning abilities will profit from instruction in problem solving only if it employs visual approaches.

- **Transfer** — Although some educators feel that general problem-solving skills, such as inference and pattern recognition, will transfer to content-area skills, scant evidence supports this view. In the 1970s and 1980s, for example, many schools taught programming in mathe-

matics classes under the hypothesis that the planning and sequencing skills required for programming would transfer to problem-solving skills in math. Research results never supported this hypothesis. In general, research tends to show that skill in one kind of problem solving will transfer primarily to similar kinds of problems that use the same solution strategies. Researchers have identified nothing like “general thinking skills,” except in relation to intelligence (IQ) variables.

## Using Problem-Solving Software in Teaching

*Classroom integration strategies for problem-solving software.* Integration strategies for problem-solving software



**FIGURE 3.6 Problem-Solving Log for the Logical Journey of the Zoombinis**

<i>Role/Date</i>	<i>Problem</i>	<i>Solution Tried</i>	<i>Outcome</i>	<i>Strategy</i>
Monitor	What to do?	Click on screen	Not much happens	Trial and error
12.10.05			When click on dice, Zs appear	
		Quit and start again	Listen to introduction for clues!	Pay attention to detail

vary considerably depending on whether the desired approach is a directed or a constructivist one. The following are guidelines for both kinds of integration strategies. Problem-solving software is appropriate for the following instructional needs:

- **To teach component skills in problem-solving strategies** — Many problem-solving packages provide good, hands-on experience with one or more of the skills required to use a problem-solving approach. These include identifying and following a logical sequence, identifying relevant information to solve problems, not jumping to conclusions too quickly, and remembering relevant information.
- **To provide support in solving problems** — Some software packages are specifically designed to scaffold students as they practice solving complex problems. For example, *Geometer's Sketchpad* helps students draw objects and investigate their mathematical properties.
- **To encourage group problem solving** — Some software provides environments that lend themselves to solving problems in small groups. For example, wiki software provides capabilities for collaborative problem solving.

**Guidelines for directed teaching.** Usually, teachers want to teach clearly defined skills. To teach problem solving, they must decide which particular kind of problem-solving ability students need to acquire and how best to foster it. For example, Stokes (1999) recommends that students use a teacher-designed reflection sheet and keep a log of problem-solving strategies and outcomes. (See Figure 3.6 for Stokes' example log.) With clearly identified skills and a definite teaching strategy, problem solving software has unique abilities to help focus students' attention on required activities. This kind of software can get students to apply and practice desired behaviors specific to a content area or more general abilities in problem solving.

The following six steps can help you integrate software for directed teaching:

1. Identify problem-solving skills or general capabilities to build or foster skills in:
  - solving one or more kinds of content-area problems (e.g., building algebra equations);

- using a scientific approach to problem solving (identifying the problem, posing hypotheses, planning a systematic approach); and
- identifying the components of problem solving such as following a sequence of steps or recalling facts.

2. Decide on an activity or a series of activities that will help teach the desired skills.
3. Examine software to locate materials that closely match the desired abilities, remembering not to judge capabilities on the basis of vendor claims alone.
4. Determine where the software fits into the teaching sequence (for example, to introduce the skill and gain attention, as a practice activity after demonstrating problem solving, or both).
5. Demonstrate the software and the steps to follow in solving problems.
6. Build in transfer activities and make students aware of the skills they are using in the software.

**Guidelines for using constructivist strategies.** Like many technology resources, some software with problem-solving functions is designed for more constructivist approaches to learning. These models give students no direct training in or introduction to solving problems; rather, they place students in highly motivational problem-solving environments and encourage them to work in groups to solve problems. Bearden and Martin (1998) describe such a strategy using problem-solving software combined with a listserv email (an email feature that allows a message to be sent to a group) for students to share their results. (Also see Martin and Bearden, 1998.) The following seven steps can help you integrate problem-solving software according to constructivist models:

1. Allow students sufficient time to explore and interact with the software, but provide some structure in the form of directions, goals, a work schedule, and organized times for sharing and discussing results.
2. Vary the amount of direction and assistance provided, depending on each student's needs.
3. Promote a reflective learning environment; let students talk about the methods they use.



## Technology Integration Lesson 3.6

### Example Strategy for Problem-Solving Software

**Title:** Geometry with *Geometer's Sketchpad*

**Grade Level:** Middle school

**Content Area/Topic:** Beginning geometry

**NETS for Students:** Standards 3 (Research and Information Fluency) and 4 (Critical Thinking, Problem Solving, and Decision Making)

**Description of Standards Applications:** This integration idea involves using software that supports many kinds of learning activities ranging from highly structured student work to free exploration. These activities encourage students to utilize digital tools as they develop critical thinking skills through problem solving.

**Instruction:** An example of a structured activity with a pre-made sketch requires a teacher to demonstrate a *Sketchpad* sketch and lead students to understand the underlying concepts with guided observations and questions. A more open-ended, constructivist approach to the same objective would have students explore the sketch themselves using guidelines the teacher provides (e.g., they make measurements of it and/or add new constructions). *Sketchpad* also supports activities called "black box tasks" for students with more sophisticated knowledge of the software. In these activities, students use the software tools to re-create a given figure or deduce the underlying properties that two or more objects have in common.

**Source:** From Ng, K., & Teong, S. (2003). *The Geometer's Sketchpad for primary geometry: A framework. Micromath 19*(3), 5–9. Used with permission.

4. Stress thinking processes rather than correct answers.
5. Point out the relationship between software activities and other kinds of problem solving.
6. Let students work together in pairs or small groups.
7. For assessments, use alternatives to traditional paper-and-pencil tests.

For an example of a problem-solving software that supports both directed and constructivist applications, see Technology Integration Lesson 3.6.

## Integrated learning systems

Integrated learning systems (ILSs) are systems that offer computer-based instruction and other resources to support instruction, along with summary reports of student progress through the instruction; all are provided through networked or online sources. The most powerful—and the most expensive—of available instructional software products, ILSs were introduced in the early 1970s and, until recently, were delivered via computer network from a central computer (server), usually located in the school. Initially, students and teachers used the

instructional and management systems from terminals or microcomputers connected to the server. More recently, ILS-type capability is provided by offering the curriculum online via the Internet rather than through a local network.

Regardless of the delivery system, an ILS is characterized as a "one-stop shopping" approach to providing software. Each ILS offers a variety of instructional techniques in one place, usually as a curriculum package complete with technical maintenance and teacher training. In addition to providing a combination of drill-and-practice, tutorial, simulation, problem-solving, reference, and tool software, an ILS is capable of maintaining detailed records on individual student assignments and performance and supplying printouts of this information to teachers. Bailey and Lumley (1991, p. 21) include the following as characteristics of an ILS:

- Instructional objectives specified, with each lesson tied to those objectives;
- Lessons integrated into the standard curriculum;
- Software that spans several grade levels in comprehensive fashion; and
- A management system that collects and records results of student performance.



# Top Ten

# Integration Strategies for Instructional Software

The best kinds and uses of instructional software depend on the content area and skills teachers need to address. The ten uses given here are among the most popular described in the literature.

1. Use simulation software such as Tom Snyder's *Operation Frog* or The Digital Frog International's online *Digital Frog* to replace or supplement hands-on dissections to help teach anatomy.
2. Use ExploreLearning's *Mouse Genetics* online simulation software to let students explore basic principles of genetics and heredity.
3. Use Electronic Arts, Inc.'s *SimCity* simulation software to give students hands-on opportunities to see environmental and other consequences of city policy and planning decisions.
4. Use Tom Snyder's *Decisions, Decisions: The Constitution* to let students take a scenario approach to learning how separation and balance of power works in the U.S. federal government.
5. Use Key Curriculum Press's *Geometer's Sketchpad* or the Center for Educational Technology's *Geometric Supposer* problem-solving software to let students explore geometric principles and proofs.
6. Use Sunburst's *The Factory Deluxe* problem-solving software to give young children a visual environment for analyzing processes, predicting outcomes, and learning the importance of sequence in solving complex problems.
7. Use Sunburst's *How the West Was One + Three × Four* instructional game software to give students a highly motivating format for practicing the correct mathematical order of operations when solving math problems.
8. Use GeoThentic, an online scaffolding learning environment from the University of Minnesota, to learn how to use geospatial technologies to learn geography.
9. Use Byki Deluxe, drill-and-practice software available from Transparent Software, to let students practice foreign language vocabulary and phrase translations.
10. Use tutorial instructional sequences such as *Odyssey K-5* from CompassLearning to give students detailed reviews of basic math or science topics or to let advanced students surge ahead.

The teacher usually makes initial assignments for work on the system, monitors student progress by reviewing ILS reports, and provides additional instruction or support where needed. As each student signs on to a microcomputer connected to the server or Internet, the file server sends (or downloads) student assignments and software to the station and proceeds to keep records on what the student does during the time spent on the system.

***The software component of an ILS.*** Instructional activities available on an ILS range from simple drill and practice to extensive tutorials. Many ILSs are moving toward complete tutorial systems intended to replace teachers in delivering entire instructional sequences. An ILS usually includes instruction on the entire scope and sequence of skills in a given content area; for example, it may cover all discrete mathematics skills typically presented in grades 1 through 6.



**Integrated learning systems software is even more accessible with portable computers.**

*The management system component of an ILS.* The capability that differentiates ILSs from other networked systems is the emphasis on individualized instruction tied to records of student progress. A typical ILS gives teachers progress reports across groups of students as well as the following kinds of information on individual performance:

- Lessons and tests completed;
- Questions missed on each lesson by numbers and percentages;
- Numbers of correct and incorrect tries;
- Time spent on each lesson and test; and
- Pretest and post-test data.

## Selecting ILSs

One way to ensure the appropriate use of ILSs is to have a careful, well-planned initial review process that involves both teachers and school administrators. Criteria for the selection process are usually based on the curriculum coverage and pedagogical strategies in the ILS, as well as perceptions about the usefulness of the reports that the system produces and to what extent they meet the needs of the district. See Figure 3.7 for examples of ILSs and a summary of ILS features.

## Benefits of ILSs

Brush (1998) estimated that as of 1998, between 11% and 25% of all U.S. schools owned ILSs. Two recent developments have caused a rapid rise in these numbers (Readers' Choice Awards, 2004):

- **District-and state-adopted academic standards** — An increased emphasis on educational accountability in

the 1990s caused states to set curriculum standards and to create tests to measure students' progress on them. Integrated software and management assessments aligned with these standards helps schools prepare for meeting these standards.

- **Accountability requirements of the No Child Left Behind (NCLB) Act** — The NCLB Act stipulated that schools who fail to meet Adequate Yearly Progress (AYP) for 3 years in a row must use a portion of their Title I funds for out-of-school tutoring, which the NCLB Act refers to as Supplemental Educational Services. ILS materials are ideal for providing these services.

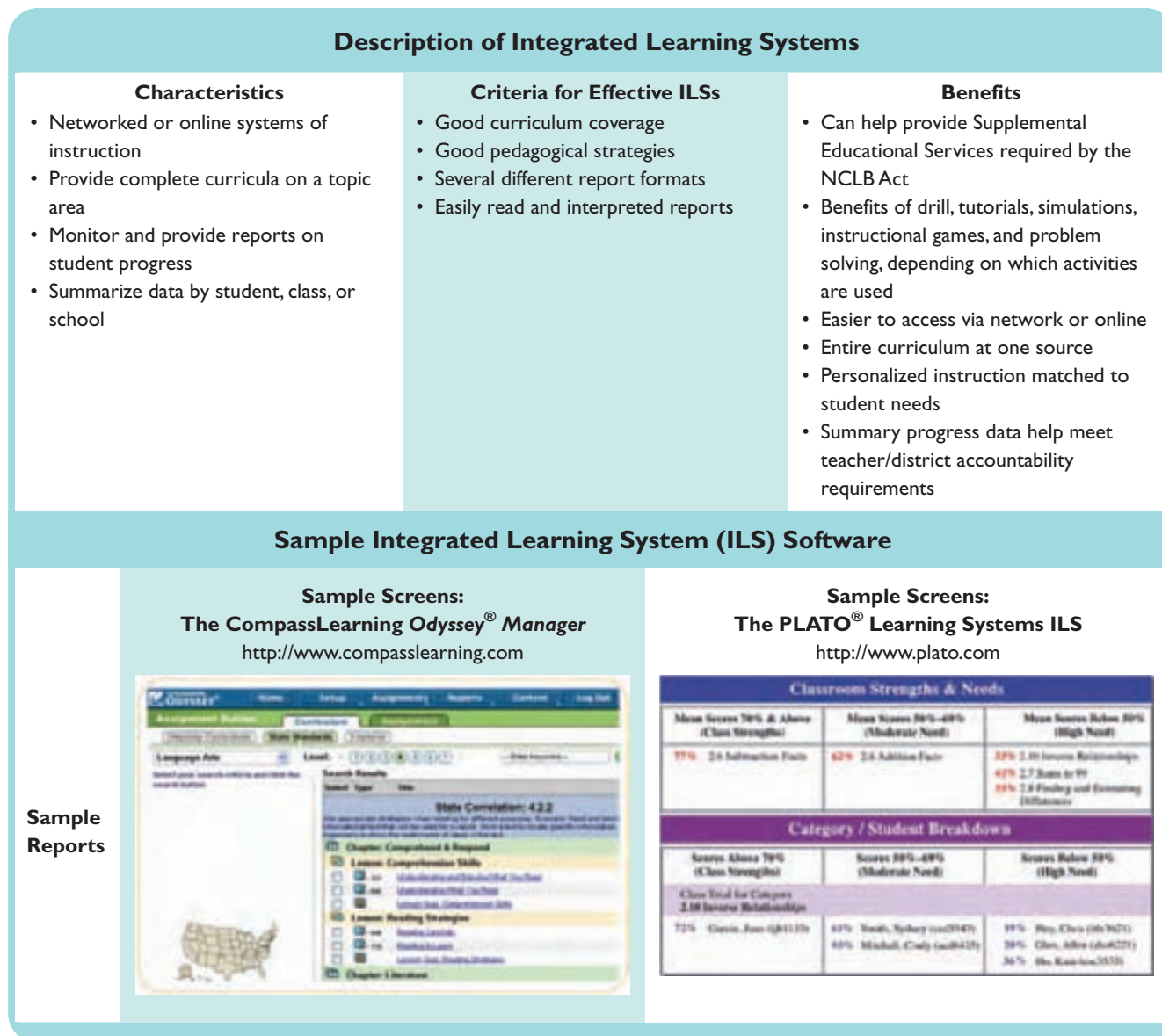
Since an ILS provides a combination of the materials described previously in this chapter, its potential benefits duplicate the benefits of those functions. In addition to those benefits, ILS networked or online materials are easy for teachers and students to access and can provide entire curricula from one location. Prepared curricula and ease of use mean that school personnel need not know a great deal about technology to use an ILS. Consequently, they usually simplify integration decisions by defining schoolwide curriculum rather than individual lessons. When teachers assign some students to use ILS activities, it frees time for them to help other students who need their personal assistance. In addition, teachers can personalize instructional activities for each student by reviewing the extensive information on student and class progress provided by the ILS management system.

## Limitations and Problems Related to ILSs

Teachers should note that although many educators still refer to these multifaceted systems as "integrated learning systems," the vendors of the systems tend to refer to them as "educational solutions," "accountability solutions," or "supplemental educational services" (Readers' Choice Awards, 2004). By whatever name, several items are ongoing concerns:

- **The costs of ILSs** — The primary criticism of ILSs centers on their expense as compared to their impact on improving learning. ILS proponents feel that the students who experience the most success with ILSs are those whose needs are typically most difficult to meet (Becker, 1994; Bender, 1991; Bracy, 1992; Shore & Johnson, 1992). Proponents also say there is value in any system that helps potential dropouts stay in school or helps students with learning disabilities. They point to studies and personal testimony from teachers over the years that ILSs motivate students by allowing them to work at their own pace and to experience success each time they are on the system.



**FIGURE 3.7** Integrated Learning Systems Summary Information

- **Research on ILS impact** — Studies of a variety of ILSs in a number of different locations reached generally the same conclusion as Van Dusen and Worthen (1995) did: The impact of an ILS on student achievement varies greatly with implementation methods. Kulik's (2003) meta-analysis of ILS studies published between 1990 and 1997 shows modest gains for schools using ILSs. Becker's (1992) summary of some 30 studies of ILS effectiveness found wide variation in results with various implementation methods and systems. Students generally tended to do somewhat better with ILSs than with other methods, and results were sometimes substantially superior to non-ILS methods. But Becker found no predictable pattern for successful and unsuccessful ILSs. Subsequent large-scale studies of ILS use in the United

Kingdom (Wood, Underwood, & Avis, 1999), Indiana (Estep, McInerney, & Vockell, 1999–2000), and New York (Miller, 1997; Paterson, Henry, & O'Quin, 2003) duplicated this finding. However, Brush, Armstrong, and Barbro (1999) found that two different resources offered in the same ILS had different impacts on achievement. Individualized software designed to provide foundations instruction had less impact than software that could be selected by teachers to supplement their own instruction.

- **Concerns about the role of ILSs** — In a follow-up to his literature review on ILS uses, Becker (1994) criticized uses of ILSs that encourage “mindless adherence to the principle of individualized instruction” (p. 78). Brush

(1998) agreed with Becker, finding that “. . . lack of teacher involvement (in ILS use) has led to improper coordination between classroom-based and computer-based instructional activities . . . and lack of teacher understanding regarding effective strategies and procedures for using ILSs” (p. 7). An early concern expressed by many educators (White, 1992) that the cost of ILSs combined with the comprehensive nature of their curricula might cause schools to view them as replacements for teachers has not yet proven to be a real problem.

## Using ILSs in Teaching

When used only as a teacher replacement to provide individual student instruction, ILSs seem to be less effective. When viewed as a supplement to other teaching methods and carefully integrated into a total teaching program, they seem more likely to have the desired impact on raising achievement. One way to ensure appropriate and cost-effective uses of ILS products may be through a careful, well-planned purchasing process that involves both teachers and administrators. One such process was developed by the California Department of Education (Armstrong, 1999). This five-stage process (planning, pre-evaluation, evaluation, selection, and implementation/post-evaluation) is designed to “establish selection procedures that ensure that . . . curricular goals remain at the heart of the selection process” (p. 3). Guidelines for potential ILS purchasers based on those offered by Chrisman (1992), Smith and Sclafani (1989), and Vaille and Hall (1998) are summarized here:

- Clearly identify the problem the ILS is supposed to solve, and understand the instructional theory on which the system is based.
- Determine whether the ILS is a closed system (one that provides 80% or more of the instruction for a given course) or an open system (one linked to the school's resources).
- Find out if the system's scope and sequence are matched to that of the school.
- Determine the target population for which the system was designed and whether it closely matches the characteristics of students who will be using the ILS.
- Consider the adequacy of the reporting and management system for the school's needs.
- Consider how much of its resources the school must spend on hardware and software.
- Project the educational benefits to the school from the system, and compare them with the costs.
- Request that vendors inform the school of ILS updates.

- Carefully evaluate the grade-level software, management system, customization, and online tools, and be sure that they match the school's expectations.
- Set up reasonable terms of procurement, and calculate the personnel and fiscal impact of the ILS.

Successful uses of ILSs have been reported for both directed and constructivist teaching approaches.

**Directed applications for ILSs.** In a directed teaching approach, an ILS system can be used for remediation and as a mainstream delivery system. With either of these applications, teachers still have important roles to play. They must assign initial levels of work, follow up on student activities on the system, and give additional personal instruction when needed.

- **For remediation** — Although ILSs are expensive alternatives to other kinds of delivery systems, the requirements of the NCLB Act have provided new motivation—as well as new funding sources—for using them. Even when new funding and the motivation to use ILSs are present, schools must determine how ILS functions coordinate with and complement those of the classroom teacher. ILSs serve target populations that have typically presented the most difficult problems for traditional classroom activities: Title I groups, English for Speakers of Other Languages (ESOL) students, special education students, and at-risk students. Schools have tried and usually failed to reach these students with other methods.
- **As a mainstream delivery system** — Rather than using an ILS only as a backup system to address educational problems, a school may let an ILS do the initial job of teaching whole courses for all students in a grade level. In light of the expense of ILSs, this type of use is more rare. However, some alternative projects, like the Edison Project (Walsh, 1999), predict that the costs of using technology in this way will amount to substantially less over time than teacher salaries. Using ILSs to increase student-to-teacher ratios has stimulated ongoing debate and study.

**Constructivist applications for ILSs.** An ILS can also combine several kinds of technology resources to support constructivist learning approaches. This kind of ILS can provide what Perkins (1991) called a “rich environment” that students can use to construct their own knowledge. ILS products useful for constructivist purposes typically have an information bank (electronic encyclopedias), symbol pads (word processing and/or desktop publishing software), construction kits (Logo or other graphic languages or tools), and phenomenaria (computer simulations and/or problem-solving resources). They also usually have data-

TABLE 3.3 Technology Integration Strategies for Directed, Constructivist, or Either Models

	Directed Models				Constructivist Models				Either Model			
	Remedy Identified weaknesses or skill deficits	Promote skill fluency or automaticity	Provide efficient, self-paced instruction	Support self-paced review of concepts	Foster problem solving and metacognition	Build mental models, increase knowledge transfer	Foster group cooperation	Allow for multiple intelligences	Generate motivation to learn	Optimize scarce personnel and material resources	Remove logistical hurdles to learning	Develop Information and literacy skills
Integration Strategies for Instructional Software	Supplement or replace worksheets, homework exercises	X										
	Prepare for tests	X	X									
Tutorials	Self-Paced reviews		X	X								
	Alternative learning strategies		X	X								
	Instruction when teachers are unavailable		X									
Simulations	Replace or supplement labs									X	X	X
	Replace or supplement role playing							X	X			
	Replace or supplement field trips					X		X	X	X	X	
	Introduce a new topic					X			X			
	Foster exploration, process learning			X		X	X					
	Encourage cooperation and group work						X	X	X			
Instructional Games	Supplement or replace worksheets, homework exercises	X	X									
	Teach cooperative skills						X	X				
Problem Solving	As a reward								X			
	Teach component skills in problem-solving strategies			X		X						
	Provide support in solving problems		X		X							
	Encourage group problem solving			X			X	X				

collection systems to track student usage of the system (Mageau, 1990).

## A Summary of Instructional Software Integration Strategies

Although descriptions of instructional software in the literature are changing, many references to software evaluation criteria and evaluation methods focus on products to be used with directed instruction. While many criteria are appropriate for software designed for both directed and constructivist kinds of uses, additional details are often lacking

on what to look for in software that will be used with constructivist methods.

Constructivist activities tend to emphasize learner-centered, collaborative, and open-ended products rather than drill or tutorial software. For example, Litchfield (1992) lists criteria for “inquiry-based science software and interactive multimedia programs.” Checklists by Hoffman and Lyons (1997) and Vaille and Hall (1998) are among those that include criteria for more open-ended products. Further criteria and methods for evaluating multimedia and online multimedia products are discussed in Chapters 6 through 8. For a summary of all instructional software uses matched to directed and constructivist integration strategies, see the matrix in Table 3.3.

## Interactive Summary

The following is a summary of the main points covered in this chapter. Additional examples and information on these points can be found by visiting recommended websites at MyEducationLab.

1. **Types of instructional software** — Instructional software packages are computer programs designed specifically to deliver or support one or more kinds of learning activities. These programs can serve one or more of the following five functions:
  - **Drill and practice** — Students work example items, usually one at a time, and receive feedback on their correctness.
  - **Tutorial** — These provide an entire instructional sequence similar to a teacher’s classroom instruction on a topic.
  - **Simulation** — These computerized models of real or imagined systems are designed to teach how the system works.
  - **Instructional games** — These activities are designed to increase motivation by adding game rules and/or competition to learning activities. Probably the most famous instructional games are *Math Blaster*® and the *Carmen Sandiego*® series.
  - **Problem solving** — These programs serve one of three purposes: (1) to foster component skills involved in solving problems, (2) to teach or provide practice in general approaches to problem solving, or (3) to teach or provide opportunities to practice solving various problems in specific content areas.
2. **Integrated learning systems** — These products offer computer-based instruction and other resources to support instruction, along with summary reports of student progress through the instruction; all are provided through networked or online sources. Now often referred to as “software solutions” or “technology solutions,” the top software solutions or ILS programs offer a range of resources from assessment system features that help diagnose and remedy student deficits to helping teachers use the products more effectively.
3. **Using instructional software to meet classroom needs** — Instructional software functions can meet each of the following classroom needs:
  - **Drill and practice** — Supplementing or replacing worksheets and homework exercises, preparation for tests
  - **Tutorial** — Self-paced reviews of instruction, alternative learning strategies, and instruction when teachers are unavailable
  - **Simulation** — In place of or as supplements to lab experiments, in place of or as supplements to role playing, in place of or as supplements to field trips, to introduce or clarify a new topic, to foster exploration and problem solving, and to encourage cooperation and group work
  - **Instructional game** — In place of worksheets and exercises, to teach cooperative group working skills, and as a reward
  - **Problem solving** — To teach component skills in problem-solving strategies, to provide practice in solving problems, and to encourage group problem solving