Instructional Design Flaws In Computer-Assisted Instruction

by Julie S. Vargas

Computers have the flexibility to teach effectively. They will do so, however, only if CAI programs adopt those features shown to be necessary for learning. As it is, such programs are filled with serious instructional flaws.

IN THE September 1984 issue of the American Psychologist, B.F. Skinner, the founder of the field of behavior analysis, lamented the lack of concern that the education establishment has shown for effective teaching. Skinner cited the suggestions for improving education given by educators, ranging from vague exhortations, such as the need for a “commitment to excellence,” to more concrete suggestions, such as paying teachers according to merit or lengthening the school day. Skinner pointed out a “curious omission in that list: it contains no suggestion that teaching be improved. There is a conspiracy of silence about teaching as a skill.”

That conspiracy of silence has extended to computer-assisted instruction (CAI). The instructional effectiveness of the techniques used in CAI is rarely addressed, nor are programs often judged on how much students learn from using them. One can read evaluations of CAI programs that do not even mention how well the programs actually work with students.

Perhaps because so little attention has been paid to effectiveness, many CAI programs contain serious instructional flaws. It is not as if there are no principles for what makes an effective self-instructional program. A large body of literature exists in which basic principles of instructional design have been researched and articulated. Since these principles are not widely known, I will summarize them here.

1. A high rate of relevant overt responding. First, effective self-instruction demands a high rate of overt responding by the learner. In a recent analysis of research on instructional programs for inner-city students who are academically delayed, Charles Greenwood and his colleagues concluded that programs that emphasize student reading, writing, and academic discussion produce higher achievement than programs that consist of such “low opportunity instruction” as lectures, media presentations, and teacher/student discussions — all of which ask only for student attention, listening, and relatively passive responding.

2. Earlier research on self-instructional tutorials showed similar results. During the 1960s James Holland and his colleagues compared programmed instruction (in which content is broken into carefully sequenced "frames," each requiring the student to respond by filling in a word or phrase) with the equivalent content presented in the usual text format. Students learned more when they responded overtly throughout.

However, simply leaving out words here and there for students to fill in did not necessarily improve instruction. The effectiveness of overt responding depended on what was left out and on what students were asked to respond to. When irrelevant words or numbers were left blank, the resulting "programmed instruction" taught students no better than simply asking them to read a passage in a straight-text format. The advantages of overt responding appeared only when the responses asked of the student were the terms, definitions, identification of examples, and so on that they were supposed to learn.

In fact, students don't even respond covertly to reading material when it is not necessary for completing a tutorial. Judith Doran and Holland tracked the eye movements of students as they...
worked through two versions of a tutorial. One version omitted trivial words, so that it was possible to fill in the blanks using contextual cues. The second version left out critical terms, so that students could not respond without having read nearly everything in the passage. With the first version, students' eyes flickered back and forth, often beginning at the blank itself. Only when critical responses were required did the students show eye movements characteristic of normal reading. In other words, students simply did not read parts of the text that they did not need for the immediate task at hand.

2. Appropriate stimulus control. To learn effectively, students must not only make relevant responses, but they must also respond to appropriate stimuli. Most educators are sensitive to stimulus control when they construct a test; they avoid cues that might give away correct answers. But in many instructional materials, students can respond "correctly" without learning what the exercises are intended to teach. To take a simple example, a program that routinely highlights words that students are to type teaches students to copy whatever is highlighted rather than to read and think about the content on their screens. Highlighting can be used productively to draw students' attention to examples, key terms, and so on. But in these uses the words highlighted are not the "answers."

In addition to giving away answers by inappropriate cuing, screens that do not ask students to discriminate between alternatives also fail to teach. For example, imagine a program intended to teach comma placement. After a sentence in which the student is told that a comma goes between the name of a city and its state (e.g., Chicago, Illinois), the student is asked to put a comma into a sentence containing the words "Boston Mass." Instead of having the student locate the place where the comma should go, a cute little man with a pointer runs across the screen under the sentence, stopping between "Boston" and "Mass." The student presses the comma key, and another sentence appears on the screen. Again the little man runs across and stops at the appropriate place, and again the student presses the comma key. No alternatives are presented. Students can progress satisfactorily without attending at all to what is on the screen; the response will always be a comma.

3. Immediate feedback. In the early 1920s Sidney Pressey invented a multiple-choice testing machine that graded each item for the student as soon as it was completed. In a series of studies, Pressey and his colleagues demonstrated the power of this immediate feedback. Educators have since applied the term immediate feedback to such procedures as giving students an answer sheet when they have completed a page of problems or returning tests on the very next day. But Pressey's studies showed that delaying feedback only until the end of a series of items inhibits the educational effectiveness of the process. To qualify as immediate, the consequences of one response must precede the next response.

4. Successive approximation. The devices used by Pressey and his students provided drill and practice on skills previously learned — or at best on skills learned by trial and error during the drill sessions. The items were randomly presented and did not systematically build new repertoires of skills.

In the late 1950s, while writing Verbal Behavior, Skinner saw how a carefully constructed sequence of items could teach new behavior by first prompting initial responses and then gradually withdrawing the hints and cues. Skinner did not like multiple-choice responding; instead, he pre-
ferred “constructed responses” in which students were asked to supply terms, write out definitions, give or identify examples, sketch figures, and so on. Successive approximation, as the sequencing techniques came to be called, formed the heart of the “linear programming” with which James Holland, F. D. Kemp, Judith Duran, Louis Eigen, Stuart Margulies, and others studied responding and stimulus control during the 1960s and 1970s.

### INSTRUCTIONAL DESIGN AND EDUCATIONAL SOFTWARE

How well do instructional computer programs incorporate the characteristics that research has shown to be critical for effective self-instruction? The answer depends on the type of software being considered. Educational software falls into four major categories: 1) drill and practice, 2) simulation, 3) tutorial, and 4) tools for writing, designing, or creating (e.g., word processors, music and art composition packages, and computer languages). Because only the first three types of software teach content, I will address only those categories.

### DRILL AND PRACTICE

Drill-and-practice exercises are designed to increase speed and/or accuracy of a skill that has already been learned. The programs present students with a series of problems, usually in random order. The problems may appear one after another, or they may be embedded in some kind of game format in which, for example, by typing correct answers, students shoot down spaceships or solve a mystery or prevent a flood from rising.

Critics have denigrated drill and practice, preferring more analytic and creative educational activities. Certainly, a school that does not go beyond teaching rules, principles, and basic facts fails to serve its students well. But a school that does not provide the repetition needed to master rules, principles, and facts falls down in its job, too.

The drill-and-practice section of a program does not teach new skills. The student doing a drill on estimating mathematical quotients is supposed to have already learned how to estimate. Straight drills, therefore, do not need to incorporate principles of successive approximation. On the remaining instructional principles, most drills score well. They present relevant problems, and few exercises can be answered from irrelevant clues in the questions. Though many drills use graphics to create a game-like atmosphere, the graphics are clearly not part of the instructional content. Graphics do not hurt the instructional mission of a program, unless they hinder students’ progress.

Still, some programs do subvert their own mission. It is surprising to see programs that are specifically designed to increase speed on a given skill actually preventing students from working at top speed. Unfortunately, many drills restrict the speed with which students can complete problems. Some of the forced waiting times, which consist of visual and auditory entertainment intended to appeal to students, are supposed to act as “reinforcement.” However, while the stars twinkle, the trains puff across the screen, or the smiley faces wink and nod, the keyboard is locked, and students are prevented from responding. If such displays last longer than a fraction of the time a student takes to complete a problem, they rob the student of instructional time and artificially limit the student’s speed.

In a study comparing math drills, for example, displays as short as four to 4½ seconds held students to the slow rate of 10 problems a minute over the five-week period during which they used the program. In contrast, students using programs without the intervening feedback completed twice as many problems in the same amount of time. Delays can also be caused by the way responses are entered, though one must be careful to watch students using a program before judging the merits or drawbacks of any particular method. One might think, for instance, that a program such as Alien Addition would slow down students. Alien Addition requires the student to place a cannon under a descending spaceship that contains an addition problem, enter the sum, and press the space bar to “shoot” the spaceship, which explodes instantaneously if the sum is correct. Although this method of answering a problem seems cumbersome, in actual use elementary school students do not respond more slowly to this program than to one that requires only typing in sums, without even requiring them to press the space bar.

Many drill-and-practice programs encourage rapid responding by imposing a time limit for responding. In Alien Addition, the student must respond before the problems in the spaceships descend to the bottom of the screen. The feature of the program that promotes speed is not the graphics but the time limit.

A recent study by John Humphrey compared student progress over time in a multiplication drill-and-practice program with time limits (the problem disappeared from the screen if not answered in time) to their progress in the same program without time limits. Time limits improved progress over time. Humphrey noticed that, even when encouraged to work as fast as possible on the sections with no time limits, the students were easily distracted. They tried to engage him in conversation, or they looked up when someone passed by in the hall. When the problems stayed on the screen for only a second or two, however, the students concentrated intently. Although time limits produced higher error rates in the first few sessions with the program, students quickly learned to answer correctly, eventually answering problems correctly very nearly as fast as they could type (around 50 correct answers in a minute). These students were not math whizzes. Humphrey used students selected by teachers as having problems with multiplication.

Why is speed so important? Certainly speed on problems drilling basic skills may not seem like a very lofty goal. But rapid responding forms the foundation for more advanced behaviors. In reading, a child who struggles to pronounce each word in a sentence faces an almost insurmountable obstacle to reading comprehension. By increasing the speed of pronouncing individual words, teachers free children from effort at that level, so that they can concentrate on understanding, remembering content, reading with expression, and so on. Similarly, in mathematics, where the use of calculators is common, speed in computing seems unnecessary. But calculator users must estimate results in order to catch such common errors as misplacing a
decimal point when entering numbers. Rapid calculation forms the foundation of such estimation skills.

Speed in basic skills supports creative endeavors, as well. Students cannot let ideas flow if they must devote all their efforts to grammar, spelling, finding synonyms, arranging words to make sentences, and so on. Through drills in which students listed descriptive terms for a given object, students in one school were able to write more creative haiku.11 By letting a computer drill students in these component behaviors, a teacher can help students rise above the level of mechanics, so that they can concentrate on the development of ideas. Speed alone is not sufficient, but it is a necessary component of competence in any field.

Because a computer can give immediate feedback, set time limits for students to beat, and repeat problems with which students are having difficulty, it is an ideal medium for providing drill and practice. Many drill programs are fun, so that, unlike worksheets filled with problems, students need little prodding to complete them. It seems suitable for a machine to take over the clerical tasks of giving and grading low-level worksheets, thus freeing teachers for activities more in keeping with their training.

SIMULATIONS

Educational simulations are computer imitations of processes. The computer may, as in a program called CATLAB,14 simulate breeding for a unit on genetics. The screen shows cats of various colorations. The student selects two for breeding, specifies how many offspring are desired, and "breeds" the cats. The computer then presents the resulting litter.

In a simulation called Rocky's Boots,15 the screen depicts various electronic parts that the student can "assemble" on the screen. When the resulting contraption is "turned on" by pressing the appropriate keys, it moves about the screen according to the way the parts were assembled, kicking and sensing the presence of green (a means of detecting an alligator that lives in one "room" of the simulation).

Simulations encourage active responding and provide continual, immediate feedback. In fact, a simulation is nothing but feedback. Simulations, like the real-world interactions they mimic, do not build behavior by providing small steps. Rather, the student is thrown into a situation that requires learning by trial and error. (The designers of Rocky's Boots realized this limitation and added a tutorial at the beginning, so that a first-time user learns to operate each part individually before being turned loose to build a contraption.)

Because most simulations do not operate small, sequenced learning steps, some students experience initial failure with them and do not wish to continue. Others may play around without analyzing what they are doing. A biology teacher, for example, reported that, when she used CATLAB for the first time, her students tended to breed cats without a general plan.16 The resulting litters carried so many mixed characteristics that students could not see the genetic principles at work. One student spent her time producing as many different-looking cats as possible—with impressive success. She did not, however, progress in her understanding of the principles of genetics. The simulation by itself did not do a better job of teaching than leaving a student unsupervised in a lab to learn by trial and error. The students needed the supervision and guidance of a teacher in order to learn the methods of selectively breeding to isolate genetic variables.

Used with planning and supervision, simulations can be very effective teaching tools. Even so, finding a simulation to meet certain instructional needs may prove difficult. Many skills, such as reading critically or writing logical arguments, do not lend themselves to a simulation format.

Most simulations set up a situation and leave the learner to initiate the action, thus encouraging active responding. As we have seen, their strength lies in providing realistic consequences. Other than failing to break down instruction into steps, simulations adhere closely to the four principles of instructional design described above. The quality of simulations centers on how well they actually approximate the situations they imitate. A simulation should require a student to make decisions similar to those required in a real situation, and the consequences of a simulation should be similar, as well.

Obviously, the responses made by students using a microcomputer simulation must be in a form that the computer can accept. In most cases, these responses will differ from those required in the actual situation. Thus breeding cats in a simulation program is accomplished with the touch of a key. To fly an airplane, the microcomputer pilot types in numbers. Such microcomputer simulations require responses differently different from actually operating wheels, throttles, flaps, and pedals or breeding live cats that their usefulness is restricted to teaching general rules and principles.

No one would seriously consider substituting training with existing microcomputer simulations of flying for actual experience in the cockpit. But similar proposals might be made in laboratory sciences, in which the final performance required of students is not so clearly defined. (As one catalogue puts it, "No need for crumbly leaves and microscopes with this program!") Since the

"Good grief! I didn't expect this when I signed up for computer camp!"

JUNE 1986
responses in a simulation rarely duplicate the kinds and complexity of behaviors required in an actual situation, a simulation cannot be a substitute for direct experience.

Some programs classified as simulations imitate very little of a given situation. A widely used program called *Lemonade Stand* is a case in point.\(^{18}\) *Lemonade Stand* is most effective when used by several students together, each operating a separate lemonade stand. The program provides users with a weather report and the cost of producing a cup of lemonade. Then it asks students to type in numbers indicating how much to charge for lemonade, how many cups to prepare, and how many advertising signs to buy. As in real life, the weather does not always follow the forecast, and the weather substantially affects the number of cups of lemonade that are sold.

After the last student has entered the numbers for his or her lemonade stand, a picture of the weather appears, accompanied by sound effects ("Raindrops Keep Falling on My Head," for example). Then the screen displays, for each stand, the cups sold, the profit earned, and the total amount of cash on hand. Students soon learn to price competitively and to advertise. (They also learn not to look at one another's numbers, since that gives an unfair advantage to the student who goes last.) The program omits such considerations as how much lemon and sugar to use per cup (the taste of the brew), where to locate the stand, how to calculate profit (or loss), and the fact that few people would set up a stand on a rainy day. As a simulation, then, *Lemonade Stand* falls short of one claim, that it "involves all the decisions normally found in running a lemonade stand."\(^{19}\) Classifying it as a simulation leads prospective users to expect more than the program delivers.

Not only are the responses in a simulation simplified, but the consequences are usually simplified as well. The designer of a simulation must specify precisely the rules governing the consequences that flow from every possible response. These rules may or may not imitate nature accurately. According to one review of the program *CATLAB*, the litters shown from various combinations of parents are based on actual cats bred in Australia, but not all programmers go to such lengths to insure accuracy. In many simulations, students figure out the rules used. If the rules correspond to the relationships that occur in the real world, fine. If not, the students have learned the idiosyncrasies of a game rather than anything useful for daily life.

Both drill-and-practice programs and simulations usually incorporate the principles of a high rate of responding of the program. They're called tutorials because you act as the tutor.

- Guide the user step by step.
- Point out what the screen shows.
- Anticipate confusion.
- Remind gently.
- Explain the significance of each step.
- Get the user out of trouble.\(^{20}\)

Nowhere is there a mention of the student's responding. Rather, the suggestions describe a lecture. The last step gives away the poor design, because a well-designed, properly field-tested tutorial never lets the student get into trouble in the first place.

Without student responses, a program cannot give feedback. Without feedback, students have no idea how they are doing or whether or not they understand all the content (if they actually read it). Without student responses, the computer cannot adjust the level of the tutorial according to the student's progress, because all students are allowed to do is "press the space bar to continue."

I watched a student work through the first lesson in a commercial algebra tutorial.\(^{21}\) She was impatient to respond. Like most students, she did not pay close attention to the flood of words until the program asked her to solve a problem — after nine screens of rules and examples. She commented on her lack of involvement. When she reached the fourth screen (Figure 1), she said, "I bet I'm going to have to do something after this, and then I'll be sorry." After three more screens of problems worked by the computer, she commented, "I wish they'd give us steps, so we could do them in steps." When she finally was given a problem to solve, in the ninth screen, she solved the problem incorrectly. When the word wrong appeared, she commented, "I guess I get to do it again." A new problem appeared on the screen, prompting the remark, "No, I don't get to do it again." Then she said to me, "I like it when the same problem comes back, because it gives us a chance to do it right."

It is easy to see how this lecture-style tutorial failed to teach. But it is difficult to see how the program got through its initial field-testing, unless its developers ignored another cardinal principle of instructional design: watch individual students going through initial versions of a tutorial, not only paying attention to how they respond, but also trying to determine which features of the content affect their responses.

Unfortunately, this algebra program is typical of tutorials. Few make any use

---

**Most tutorials present several pages of text before asking anything of the student; they become traditional workbooks displayed on a screen.**

and appropriate stimulus control. However, neither drill-and-practice programs nor simulations use the techniques of successive approximation, and that severely limits their effectiveness for the initial teaching of a subject. Tutorials take up the slack in initial teaching — or they would if they were not so poorly designed.

**TUTORIALS**

Tutorials are designed to teach new subject matter. Thus the content of a tutorial should make use of all the techniques and information available in the extensive research on programmed instruction. In particular, tutorials should make use of the results of the research, mentioned above, that shows that students do not learn information presented on a screen unless they are asked to respond to it.

In spite of this research, many authors write tutorial programs that consist of several screens of text (a mini-lecture), followed by a quiz. Indeed, this is the most common format for so-called tutorials, and it is encouraged by the publications that software authors are likely to see. For example, a book on instructional design including the following advice was sent to all developers of Apple software:

*In a tutorial you take your readers step by step through the fundamentals...*
of the techniques of successive approximation — the careful sequencing of responses asked of the student. Rather, they present several pages of text before asking anything of the student; they become traditional workbooks displayed on a screen. The PLATO equivalent of this algebra program, for example, provides another case of a present-and-then-test program. A typical lesson in this algebra “tutorial” presents six to eight screens of text before the first problem requiring a solution comes into view. As a lecture, the program is well-designed. It gives rules, followed by examples (with time delays). But as a tutorial the program fails to incorporate even the most fundamental principles of instructional design.

Not long ago, I had a more personal encounter with a tutorial. I bought a Macintosh personal computer and wanted to learn how to use it. The machine came with some software, including a tutorial that I began immediately. At first the program required me to respond and taught well. I quickly became proficient with the “mouse” that allows the user to manipulate the cursor. Then came a section called “Show me my electronic desk.” The audiotape that came with the tutorial said that the keyboard would be locked during the “tutorial.” With a sinking heart, I realized that I was in for a lecture. Yet it persisted. I sat, watching and listening intently as arrows moved about the screen, menus were “pulled down,” items were “clicked” open and shut, and objects were moved about on the “desktop.” At the end, the tape informed me that I could now practice what I had learned. But I had learned practically nothing. (Evidently, I was not alone, because the tape suggests repeating the tutorial as many times as needed — an admission that the section does not teach effectively in the first place.) I finally learned, inefficiently, how to use the machine through trying things and seeing how they worked, consulting the manuals when I came to an impasse, and feeling much frustration along the way.

Good tutorials do exist, however. For example, Bank Street Writer, a popular word-processing program designed for students, comes with an excellent tutorial. Directions at the top of the screen explain how to do something, such as insert or delete. On the same screen, the student is asked to try the action in a box with a text provided. Soon, with very little effort and without the use of manuals, the novice can operate the word-processing program effectively enough to begin to use it for writing. I suspect that the designers of early versions of this tutorial watched students using their program and revised it accordingly.

**RECOMMENDATIONS**

For those who design programs, I have two recommendations. First, ask for relevant responses in every presentation screen. (Two exceptions are “help” screens that give information requested by users and “initial directions” screens that explain the procedures for operating the program.) Second, try out initial versions with individuals; sit beside them as they go through the program; make notes on what they respond to, as well as on whether or not they respond correctly. When a student fails to respond readily, perhaps because examples or directions are unclear, resist any temptation to jump in and explain. Your screens must stand on their own. If one student has difficulty understanding what to do, others will too. You need to revise.

To those who do not design instructional programs, I offer some suggestions, too. You are consumers. Your buying decisions influence the quality of programs on the market. Most of you already know the advantages of trying out a program with students before buying it, and most of you have encountered difficulties with returning programs you did not want. Some consortia, such as the Educational Computer Consortium of Ohio, will send members copies of educational programs to try out, for a total cost (membership fee plus postage and handling) of less than the cost of one inexpensive program.

Of course, there are reviews of software, but few reviewers try out software with students, and the evaluation checklists that guide their judgments do not require them to watch students use the software. To help evaluate the instructional design of software, I offer the checklist in Figure 2. It assumes that the program will be tried out with students, but it can be used as a rough check on instructional quality in order to eliminate programs that are not worth trying out with students.

Many technological innovations have been touted as aids to teaching and learning. Early in this century, Thomas Edison, excited by his invention of

**FIGURE 1.**

![Diagram of addition and subtraction of monomials](image)

To add monomials that differ only in their numerical coefficients, use the Distributive Axiom of Multiplication. Or, you can find the sum of the numerical coefficients and multiply this sum by the common variable factors.

To subtract one monomial from another like monomial, add the opposite (additive inverse) of the subtrahend to the minuend.

Press + to view the next page. Press + to view the previous page.

---

**Whereas movies, television, and overhead projectors only present information, computers can interact with individual students.**
moving pictures, promised a great revolution in education: "I believe that the motion picture is destined to revolutionize our educational system and that in a few years it will supplant largely, if not entirely, the use of textbooks in our schools."26 Many years later, similar claims were made for instructional television.

Today, many educators predict that the computer will revolutionize education, while others see the computer as just one more in a long line of educational devices that come into fashion for a short time and then recede into the background. But computers differ from other educational innovations in the one capability most critical for effective teaching. Whereas movies, television, and overhead projectors only present information, computers can interact with individual students. Students respond little during movies, television programs, and presentations involving overhead projectors.27 Some even take such presentations as opportunities to catch up on their sleep. A computer program, on the other hand, has the potential to keep students continually responding. Unlike movies or television, a computer program can individualize instruction — letting each student progress at his or her own rate and giving practice on whatever items a student answers incorrectly or slowly. Such individual interaction with students is impossible with movies or television. Even a responsive teacher using an overhead projector would find it difficult to adjust his or her presentation to meet individual needs.

Computers have the flexibility to teach effectively. They will do so, however, only if CAI programs adopt those features shown to be necessary for learning: a high rate of relevant responding, appropriate stimulus control, immediate feedback, and techniques of successive approximation. Until program developers heed the findings of the relevant research in instructional design, the computer revolution in education will remain just one more bright technological future waiting to begin.

4. Ibid.
7. This example is taken from a program I saw on display at a convention. Later in the same program, the student is asked to locate the comma in the correct place. Evidently the designers realized that simply pressing the comma key means nothing, so in the testing section they changed the task. But it should have been changed in the teaching section, as well.
21. Algebra, Volume 3 (Agoura Hills, Calif.: The Science of Learning, Peachtree Software, 1982). This screen is reproduced by permission of the program's new owners, EduWare Services, Inc., a Britannica Learning Company, 185 Berry St., San Francisco, CA 94107. More recent versions of this program may have improved on the version described here.
23. The manual, A Guided Tour of Macintosh, and a disk of the same name come with the purchase of a Macintosh computer.
24. Bank Street Writer Tutorial comes with the Bank Street Writer (San Rafael, Calif.: Broderbund Software, 1982).
27. Greenwood et al., "Opportunity to Respond..."