Using Animation to Help Students Learn Computer Algorithms

Richard Catrambone, Georgia Institute of Technology, Atlanta, Georgia, and **A. Fleming Seay,** Carnegie Mellon University, Pittsburgh, Pennsylvania

This paper compares the effects of graphical study aids and animation on the problem-solving performance of students learning computer algorithms. Prior research has found inconsistent effects of animation on learning, and we believe this is partly attributable to animations not being designed to convey key information to learners. We performed an instructional analysis of the to-be-learned algorithms and designed the teaching materials based on that analysis. Participants studied stronger or weaker text-based information about the algorithm, and then some participants additionally studied still frames or an animation. Across 2 studies, learners who studied materials based on the instructional analysis tended to outperform other participants on both near and far transfer tasks. Animation also aided performance, particularly for participants who initially read the weaker text. These results suggest that animation might be added to curricula as a way of improving learning without needing revisions of existing texts and materials. Actual or potential applications of this research include the development of animations for learning complex systems as well as guidelines for determining when animations can aid learning.

INTRODUCTION

Computer-driven simulations and study aids are increasingly popular components of instructional curricula. Early experiments with such aids described their ability to enhance comprehension of complex subject matter (e.g., Rigney & Lutz, 1976). It has been argued that such instructional systems can reduce training time by an average of 33% and aid the "quality" of learning, as compared with more traditional techniques (Stephenson, 1994). However, there is a need for systematic research into the many components that can make up and influence computer-aided teaching. The aim of the present research is to examine the use of animation to aid learning of computer algorithms.

Foundations in Algorithm Animation Research

A field known as *software visualization* (Price, Baecker, & Small, 1993) is based on the use of graphical representations to help describe the operations of a computer. Algorithm animation is concerned with the use of animation to teach the function and proper implementation of complex computer algorithms (Baecker 1998; Brown, 1988). This is done through the use of graphical depictions of the implied structure and motion of the elements of an algorithm as they work through execution. The first work in the field came in the form of a video presented at SIGGRAPH in 1981 entitled *Sorting out Sorting* (Baecker & Sherman, 1981). Following this, a diverse series of implementations of the algorithm animation concept have come into existence. From multimedia presentations to entire animation development environments such as John Stasko's Tango (Stasko, 1990), XTango (Stasko, 1992), and Samba, the field of algorithm animation is an expanding one (Byrne, Catrambone, & Stasko, 1999).

In spite of its growth, the field has encountered one key problem: It has been difficult to

Address correspondence to Richard Catrambone, School of Psychology, Georgia Institute of Technology, Atlanta, GA 30332-0170; rc7@prism.gatech.edu. **HUMAN FACTORS,** Vol. 44, No. 3, Fall 2002, pp. 495–511. Copyright © 2002, Human Factors and Ergonomics Society. All rights reserved.

empirically demonstrate consistent and significant improvement in student performance as a result of exposure to algorithm animations (Byrne et al., 1999). However, some studies have demonstrated a significant effect of the use of algorithm animations on problem-solving performance (Hansen, Schrimpsher, & Narayanan, 1999). We will outline some possible reasons for the inconsistent findings.

Diagrams, Animations, and Learning

Hays (1996) asserted that animation is most useful when the instructional domain involves dynamic and/or spatial processes as key elements. The results of Hays's studies led him to the conclusion that animation is better than text at communicating concepts involving time and motion. More generally, to the extent that animations are explicit and deterministic in representing the elements and operations of a problem space, they disambiguate that problem space, making misinterpretations less likely. Scaife and Rogers (1996) called this phenomenon "graphical constraining."

Although good diagrams also might reduce misinterpretation, some researchers have suggested that animation might have a privileged position in this regard. For instance, Reiber and Kini (1991) suggested that "animation makes the cognitive task more concrete by providing motion and trajectory attributes directly to the learner, thus reducing the processing demands ... and hopefully increasing the potential for successful and accurate encoding" (p. 86).

Even static diagrams with well-understood, "classical" motion cues (i.e., arrows or flow symbols) suffer from a certain amount of ambiguity that is overcome by an animation that actually performs the suggested motion. With diagrams of dynamic processes such as algorithms, readers must work to connect and "mentally animate" the elements of the display, providing the opportunity for incorrect inferences (Hegarty, 1992; Reiber, 1991).

A large body of work on multimedia learning by Mayer and colleagues (some of which is summarized in Mayer, 2001) indicates that providing learners with information through multiple channels – for example, pictures and text – is often more effective than using a single channel on subsequent problem-solving performance. The benefits might be attributable to reductions in cognitive load as well as to intrinsic differences in media for conveying certain types of information (Mayer 2001, pp. 67–71; see also Mousavi, Low, & Sweller, 1995). Mayer argued that it is not fruitful to compare two media such as pictures and text and ask whether one is better than the other for conveying the same information. The problem with such a comparison is that it is based on the flawed assumption that the "same" information can be conveyed through two different media. Rather, a more fruitful question is to ask whether multiple media help learners to construct better representations of the material. Of course, good and bad instructional material can be developed in multiple media, and therefore it is important to ensure that the materials developed are constructed well. This is the value of a task analysis, which can guide materials development and increase the chances of the instructional material covering the needed information. This issue is discussed in the next section.

METHODOLOGICAL OVERVIEW TO EXPERIMENTS

Two experiments were conducted to examine the effects of animation on learning and transfer in the domain of algorithms. In both experiments participants studied training materials that involved or did not involve animation and that included textual materials that were either based or not based on a task analysis. After the training phase, participants solved problems based on the algorithm they learned (near transfer problems) as well as a related algorithm (far transfer problems). Study time was measured, and learning was assessed by performance (accuracy) on test problems that varied in how similar they were to the training materials. The problem domain and development of the materials are described later in this section.

It is important to note that although the experiments described here include performance comparisons of students studying texts either from a textbook or based on the task analysis, the key comparisons involve students who received additional study aids versus those who did not. Thus in Experiment 1, for example, the key comparison is between the task-analysis-based text condition and the task-analysis-based text plus animation condition. If both task analysis conditions outperform the control text condition, this would suggest that the task analysis text is better in one or more ways than the control text. This result by itself would be of interest in that it would support the value of carefully identifying what learners need to know. However, it would not identify which of the many differences between the control and task analysis texts produced the difference. In contrast, differences in problem-solving performance between the learners who studied only the task-analysis-based text and those who studied that text and also saw the animation could be more reliably attributed to the presence of the animation (particularly if study time is considered in the analyses) and would provide a good test of the value of animation.

The Problem Domain: Arrays, Stacks, and Queues

Arrays. The data structure that forms the basis for the algorithms used in the present research is called an *array.* An array is a group of locations used to store values. Arrays are often given names corresponding to the type of information they store. For example, an array used to store the results from a final exam might be called *scores* (see Figure 1 for an example of a particular type of array called a *stack*). Each individual value stored in an array is called an element. The size of an array determines the number of elements that can be stored, just as the number of dimples in an egg carton determines how many eggs can be held in it. In general, array size describes the form of

1. Consider the following stack:

Scores

6	
5	28
4	19
3	42
2	12
1	17

A. Show the outcome of each of the following operations on this stack using the blank stacks below:

Push (16)
Pop
Pop
Push (48)

6		6		6		6		6	
5	28	5		5		5		5	
4	19	4		4		4		4	
3	42	3		3		3		3	
2	12	2		2		2		2	
1	17	1		1		1		1	

B. What would be the value of top after this series of operations?

Figure 1. Example near transfer problem. In each stack, left column = positions and right column = values.

an array by specifying the number of rows and columns in the array. For example, an array with two rows and three columns could hold a maximum of six elements. One-dimensional arrays have either one row and one or more columns or one column and one or more rows. Because of this one-dimensional property, only one number is needed to describe the size of one-dimensional arrays.

In a one-dimensional array, elements are indexed or addressed using their ordinal (numerically successive) position in the array. By convention, indices are often a series of numbers, starting with *1,* that increment from bottom to top or left to right. Elements are entered into the array in the same order in which the indices are incremented. Each element value is associated with one and only one index. Individual elements can be referred to using the name of the array and the index at which the element in question is stored.

A *pointer* is a variable used to track indices of interest in an array. *Top, head,* and *tail* (covered later) are examples of pointers. Pointers point to indices, never directly to the element values those indices store. However, through the use of correct notation, pointers can be used to indicate element values as follows. The name of the pointer is followed by the name of the array in which it is involved: A pointer top in the array scores would be written "top[scores]." If top[scores] points at the index *1* in the array, it would be written "top[scores] = 1." The value *17* stored in index *1* of the array scores would be indicated by "scores(top[scores]) = 17," which would be equivalent to writing "scores[1] = 17" because top[scores] = 1.

Stacks. A general familiarity with arrays allows understanding of a special kind of array called a *stack.* Stacks adhere to a number of rules and constraints to which not all other arrays adhere. Because of this, stacks are useful for performing some operations for which other types of arrays are less suited. Generally, stacks are referred to as *last-in-first-out* (LIFO) data structures. That is to say, stacks are programmed so that the most recent element entered into the stack (or the last one) will be the first to be removed. It might be useful here to think of the analogy of a stack of plates. The plate on the top of the stack – presumably the one put there last – will be the first to be removed. Essentially, this is how stack data structures operate. Just like a stack of plates, stacks fill with elements from the bottom up.

A stack has two attributes: its size and a pointer called *top.* Stack size defines the number of elements a stack can hold, just like array size. Returning to the stack of plates analogy, a stack of size six would have room enough for storage of a maximum of six plates.

Top is the name given to the pointer that points at the index of the element most recently inserted into the stack. Top stores the value of the index, *not* the value of the element stored at that index. If top = 0, we know that the stack is empty. The initial value of top is always 0 because stacks always start out empty. When top = stack size, then the stack is full because no other available indices would exist to store additional elements.

Push is the command used to insert a new element into the stack. Push is accompanied by an argument or additional information necessary to perform the command. In the case of push, the argument is always the element that is being added to the stack. Though push is a single command, a few subcommands run each time a push is attempted. In particular, the value of top is increased by 1 to point at the next available index in the stack. If there is such an available storage location, then the element value is inserted into the stack at the index to which top points.

Another subcommand of the push command checks for an error condition called *overflow.* The overflow error occurs when one attempts to push a new element onto an already full stack. Because the stack is unable to handle the new element, the "stack overflow" error message is returned.

Pop is the command used to remove the top element from the stack. Pop is not accompanied by any argument because the element corresponding to the index currently stored in top will always be removed by the pop command. A pop command first checks to see whether or not the stack is empty. If it is not empty, the element stored in the index pointed to by top is removed from the stack, and then top is decreased by 1. This reduction in top is necessary to indicate that the next element stored in the array is now the most recently inserted or top element in the stack.

A subcommand of the pop command checks for an error condition called *underflow*. The underflow error occurs when one attempts to pop an element from an already empty stack. Because there are no elements in the stack, the "stack underflow" error message is returned.

Queues. A queue operates in much the same way as a stack but with a few notable distinctions. Whereas the stack operates under a last-in-first-out (LIFO) policy, the queue operates under a first-in-first-out (FIFO) policy. To institute this it is necessary that the queue track two pointers called *head* and *tail*. *Head* points to the index of the oldest element in the queue, whereas *tail* points to the index of the most recently inserted element in the queue. When a *dequeue* command is issued, the element stored at the index pointed to by head is removed from the queue. This is analogous to the pop command in the stack. When an *enqueue* command is issued, the new element is entered into the queue at the rear, tail is incremented by 1, and the tail pointer points to its index. This is analogous to the push command in the stack. Both overflow and underflow errors occur in the same way with queues as they do with stacks.

Development of the Task Analysis and Control Texts

A section of text describing stacks was borrowed from a well-regarded algorithms textbook (Cormen, Lieserson, & Rivest, 1998) to serve as a control text for purposes of comparison with the text generated from a task analysis. This control text, with associated static diagrams, represents what a student might encounter in a contemporary algorithms class.

In order to create the task-analysis-based text, we worked through a large number of sample problems and recorded for each problem what the algorithm would do next and why. The result of this process was a collection of definitions, rules, and examples that seemed to account for the performance of the stack algorithm for all the problems up to that point. These materials were then tested on additional problems in an iterative fashion until it was believed they could be applied to solve all problems dealing with the behavior of stacks as new values were added or old values were removed. These materials were then edited into what became the task-analysis-based text used in the experiments.

An example of how the control and task-analysis-based texts deal with a particular instructional element is presented in order to characterize the type of differences typical between the two text types.

Consider definition three (Def3) of the stack task analysis shown in Figure 2. Mastery of this definition is an important enabling factor for successful problem solving in this domain. To see the differences between the task-analysis-based and control text used in the experiments, consider how each presents the information contained in this definition:

Task analysis-based passage: "*Top* is the name given to the pointer that 'points' at the index of the element most recently inserted into the stack. As such, top stores the value of the index, *not* the value of the element stored at that index."

Control passage (from textbook). "The Stack has an attribute top [Scores] that indexes the most recently inserted element. The Stack consists of elements Scores[1..top[Scores]], where Scores[1] is the element at the bottom of the Stack and Scores[top[Scores]] is the element on top" (Cormen et al., 1998, p. 200).

Figures presented with both texts depict the difference between the value of top and the value of the element stored at top. However, the control text by itself is not as explicit about this point as the task-analysis-based text is. This lack of clear and explicit treatment of such issues characterizes the difference between control and the task-analysis-based texts.

The Development of Test Problems

The creation of the problems used in the posttests was also directly driven by the task analysis. Consider Part B of the question from the near transfer problem shown in Figure 1. In order to correctly answer this question, one would need to understand three definitions (Def1, Def3, and Def4) and one rule (Rule1; see Figure 2). The content of Def3, which indicates the difference between the value of top and the value stored at top, is the key to Part B.

Scores

6	
5	28
4	19
3	42
2	12
1	17

Given: Task analysis of array.

Def1: Stack – a special kind of array with two attributes (size and top) and two operators (push and pop).

Def2: Stack size – the number of elements a stack can hold; equivalent to array size in that an array of size *r* defined as a stack can hold *r* elements.

Def3: Top – pointer representing the index of the element that was most recently inserted into a stack. The value "top" is the index of the array where the most recently inserted element has been stored. Top is not the value of the stored element itself.

Rule1: Top is increased or decreased by one every time an element is inserted or removed from the stack.

Examp1: For the stack scores pictured above, top[scores] = 5 because 28 was the last value inserted into the array. We know that 28 was the last value inserted because arrays are always filled from bottom to top and a stack is a type of array.

Rule2: When top[scores] = 0, then the array is said to be empty. The initial value of top is always 0 because stacks always start out empty.

Rule3: When top[scores] = *r*, the array is said to be full.

Examp2: For the array scores above, if top[scores] = 6, then the array is said to be full.

Def4: Push – the command that inserts a value into the stack.

Examp3: Given the stack above with top[scores] = 5, push(32) would first increase the value of top[scores] by 1 to top[scores] = 6. Then the element value 32 would be inserted into the stack at Index 6. As a result, 32 becomes the most recently inserted element in the stack.

Figure 2. Sample of task analysis for stack data type. In stack, left column = positions and right column = values.

Without a clear understanding of this difference, one is led to report the value stored in top as the answer rather than the true value of top.

Because test questions were generated based on the task analysis, it was possible to determine exactly which elements of the problem domain were necessary to answer each problem. This made it possible to make predictions in regard to the shortcomings that would arise in the performance of individuals exposed to study materials that did not clearly present elements identified by the task analysis. Near transfer problems were those that dealt with the explicitly instructed data structure, stacks, whereas far transfer problems dealt with queues, the data structures that were not taught (other than through an introductory paragraph).

Development of Static and Animated Study Aids

Key frames were defined that represented the important states of the algorithm (i.e., the state of the algorithm after each important action, as identified in the task analysis) as it runs to completion. The computer animation package could then automate the job of creating and inserting the other in-between frames that must exist in order to create an animation showing a smooth transition between the key frames. A frames-based study aid (used in Experiment 2) consisted of only the key frames annotated with pseudocode that described how the algorithm transitioned from key frame *n* to key frame *n* + 1.

The animation presented the general properties of a stack (i.e., size, top, and indexing) as new items were added or removed from the stack. It also depicted the behavior of the stack data structure under several push and pop conditions and, finally, under an overflow condition. Accompanying the animation on the screen was pseudocode associated with the behavior being performed by the animation. This was designed to allow the participant to associate the events on the screen with the execution of the corresponding steps in the algorithm.

EXPERIMENT 1

A between-subjects design was employed to investigate the effect of instructional material type on near and far transfer. There were three levels of the independent variable: control text (control), task analysis-based text (TA), and task-analysis-based text plus animation (TA+anim).

Participants in the two static conditions (control and TA) studied only texts, whereas participants in the TA+anim condition viewed an animation in addition to the text used in the TA condition. After studying the materials, participants in all conditions completed a series of near transfer problems designed to assess their level of understanding of the stack algorithm. Upon completion of these problems, participants attempted far transfer test problems dealing with queues. Some near and far transfer problems required participants to perform a small number of operations on a preexisting data structure, whereas others required participants to draw the data structure and perform a series of operations on it.

The amount of time each participant spent with the materials was measured at two points: after the completion of the near transfer problems (including the time spent with the instructional material) and after completion of the far transfer problems. Time spent viewing the animation was also collected in the TA+anim condition. This allowed analysis of the relationship between study time and performance.

We expected that the task analysis conditions (TA and TA+anim) would perform significantly better than the control condition on both near and far transfer problems. Further, we expected that participants in the TA+anim condition would outperform the TA condition on far transfer problems because of a better understanding of the stack algorithm, which serves as the building block for understanding queues.

The argument could be made that any effect for the animation could be attributable to the fact that participants in the TA+anim condition simply received more materials than those in the other conditions. However, this study was aimed at assessing the utility of algorithm animation as a worthy *addition* to text and static graphics. It was not designed to assess the standalone strength of the animation as a substitute for text-based treatments of this material.

If the TA+anim condition produces significantly better near or far transfer performance relative to the TA condition, this would support the potential value of animation. If participants in the TA+anim condition do not spend significantly more time (in statistical and pragmatic senses) on the materials relative to participants in the TA condition, this would suggest that animation can aid learning without much cost in additional learning time. Of course, differences of opinion may exist among researchers, educators, and administrators about what constitutes a pragmatic and/or practical increase in study time.

Method

Participants. Participants were 99 undergraduate students at the Georgia Institute of Technology who participated to receive credit in their psychology course. No participants were computer science majors, and they did not have programming experience or computer science courses at the high school or college level as indicated by their questionnaire responses.

Apparatus. A PC running the Microsoft Windows operating system and Macromedia Director (Macromedia, San Francisco, CA) was used to create the study aids employed in this study. The animation was displayed on PCs running the Microsoft Windows operating system.

Design and procedure. Participants were randomly assigned to three separate groups: control text (control; $n = 32$), task analysis text

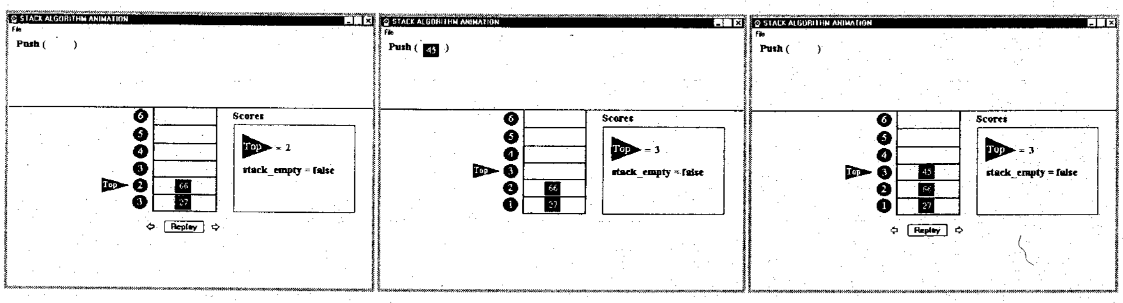

Figure 3. Example of a three-key-frame sequence.

(TA; $n = 31$), and task analysis text plus animation (TA+anim; $n = 36$). Participants in the control and TA conditions were run in a classroom setting, whereas those in the TA+anim condition were run in groups of 4 to 10 in a computer lab. General cognitive ability was assessed through the use of self-report measures of grade point average (GPA) and Scholastic Aptitude Test (SAT) scores.

Participants in all conditions read a brief text containing a general introduction to algorithms followed by a task-analysis-based coverage of the array data structure. Following this, all participants read a text covering the stack algorithm. TA and TA+anim participants read the task-analysis-based text, and participants in the control text condition read the text from the algorithms textbook.

After reading the text, participants in the TA+anim condition made use of an animation that demonstrated the operation of the stack algorithm. See Figure 3 for screen shots from the animation. The participant was able to review each sequence of the animation as desired, replaying small sequences or the entirety.

Next, all participants completed the near transfer items. The near transfer tasks were a series of 10 problems based on the stack algorithm. These exercises contained six completion-type problems and four from-scratch problems. The latter four problems required the participant to draw a stack and perform a series of operations upon it. See Figure 4 for an example problem.

After completing the near transfer problems, participants read a transitional paragraph that introduced the concept of the queue data structure and briefly described its relationship to the stack. This paragraph was the same for all conditions. After reading the queue text participants began the far transfer tasks, which were a series of five completion and six from-scratch problems concerning the operation of queues. These tasks included two short-answer problems requiring participants to apply their knowledge of both the stack and queue data

Draw a stack of Size 5 and write out a series of commands that would cause an overflow error.

5	
4	
3	
2	
1	

Scoring: Correct size of stack (1 point), proper indices (1 point), appropriate commands to produce overflow (e.g., 6 push commands; 1 point) = total of 3 possible points.

Figure 4. Scoring approach for a near transfer problem. In stack, left column = positions and right column = values.

TABLE 1: Mean SAT Verbal Score, SAT Quantitative Score, and GPA as a Function of Condition (Experiment 1)

	Condition			
	Control Text	TA-Based Text	TA + Animation	Average
SAT verbal	610.77 (*n* = 26)	601.67 (*n* = 24)	612.14 (*n* = 28)	608.46
SAT quantitative	629.62 (*n* = 26)	654.17 (*n* = 24)	670.00 (*n* = 28)	651.67
GPA	2.83 (*n* = 30)	2.88 (*n* = 30)	2.68 (*n* = 34)	2.79

structures to a "real world" situation. See Appendix A for two of the far transfer problems.

While working on the near and far problem sets, participants were allowed to refer back to the training materials (including the animation for those in the TA+anim condition). Study and performance times were recorded.

Two raters scored participants' answers to the problem sets. Because each problem required several independent responses, a binary scoring system was used to allow partial credit for each problem. Each problem was assigned a point value based on the number of individual responses it required. For each correct response a 1 was scored; for each incorrect or missing response a 0 was scored. The near transfer problem set contained 53 individual responses, and the far transfer set contained 139.

To illustrate the scoring system, consider the scoring for one of the near transfer problems illustrated in Figure 4. The scoring rationale was that 1 point was awarded for correct size of the stack, 1 point was awarded for the proper indices, and 1 point was awarded for an appropriate set of commands, for a total of 3 available points.

Results

Cognitive abilities. Cognitive ability data are displayed in Table 1. Not all participants provided information on each measure; the number of participants in each condition who contributed to each mean is indicated in the table. An analysis of variance (ANOVA) performed on these data indicated that there were no significant differences in cognitive ability among the groups for SAT verbal, $F(2, 75) = 0.16$, $p = .86$, $MSE = 5215.86$; SAT quantitative, $F(2, 75) = 1.87$, $p = .16$, $MSE = 5929.06$; or GPA $F(2, 91) = 0.92$, $p = .40$, $MSE = 0.36$.

Near and far transfer: Study and performance time. Data on time to completion are displayed in Table 2. Not all participants recorded their start and finish times on all parts

TABLE 2: Time (minutes) Spent on Near Transfer and Far Transfer Tasks (Experiment 1)

	Condition			
	Control Text	TA-Based Text	TA + Animation	Average
Near transfer	27.86 (*n* = 28)	28.60 (*n* = 20)	32.65 (*n* = 17)	29.34
Far transfer	16.00 (*n* = 25)	17.43 (*n* = 21)	20.81 (*n* = 16)	17.73

Note: Near transfer time included time spent reading the text, watching the animation (if available), and solving problems. Far transfer time was spent only solving problems.

TABLE 3: Problem-Solving Scores for Near and Far Transfer Tasks (Experiment 1)

	Condition			
	Control Text (n = 32)	TA-Based Text (n = 31)	TA + Animation (n = 36)	Average
Near transfer	37.66	45.87	49.33	44.48
Far transfer	91.63	91.36	120.14	101.91

Note: Maximum possible score was 53 for near transfer and 139 for far transfer.

of the experimental booklets; the number of participants in each condition who contributed to each mean is indicated in the table.

A statistically significant difference among the three conditions was detected in the combined time to study the training material and solve the near transfer problems, $F(2, 62) = 3.20$, $p = .047$, $MSE = 40.32$. An examination of the means in Table 2 suggest that the TA+anim condition took longer than the other two conditions. Pair-wise comparisons showed that the control group spent less time reading and solving the first problem set than did the TA+anim group, $p = .017$; required $p = .05$ using Shaffer (1986) sequential Bonferroni pair-wise comparisons for providing a family-wise $\propto$ of .05 for multiple comparisons; see also Seaman, Levin, and Serlin (1991). There was no significant difference between the control and TA conditions ($p = .69$) or between the TA and TA+anim conditions ($p = .06$), although there was a trend toward the TA condition being faster than the TA+anim condition.

A statistically significant difference was detected in time to completion of the far transfer problem set among the three conditions, $F(2, 59) = 8.79$, $p = .0005$, $MSE = 13.01$. Pairwise comparisons showed that the TA+anim condition took longer than did the control ($p = .0001$; required $p = .05$) and TA conditions ($p = .006$; required $p = .05$). There was no significant difference between the control and TA conditions ($p = .18$).

Near and far transfer: Problem-solving accuracy. Performance on near and far transfer problems is displayed in Table 3 (all participants provided data on transfer performance). There was a significant difference among the three groups on near transfer performance, $F(2, 96) = 14.88$, $p < .0001$, $MSE = 80.59$. The control group was outperformed by both the TA+anim group ($p < .0001$; required $p = .05$) and the TA group ($p = .0005$; required $p = .05$). There was no significant difference between the TA and TA+anim conditions ($p = .12$).

There was a significant difference among the three groups on far transfer performance, $F(2, 96) = 7.56$, $p = .0009$, $MSE = 1243.95$. The TA+anim group outperformed both the control group ($p = .0012$; required $p = .05$) and the TA group ($p = .0012$; required $p = .05$). There was no significant difference between the TA and control conditions ($p = .98$).

Relation between time to completion and problem solving accuracy. With time spent studying and solving the first problem set entered as a covariate, an analysis of covariance of near transfer performance continued to show an effect of condition, $F(2, 61) = 5.25$, $p = .008$, $MSE = 91.79$. With time spent studying and solving the first problem set and time spent solving the second problem set entered as covariates, an analysis of covariance of far transfer performance no longer showed an effect of condition, $F(2, 50) = 0.13$, $p = .88$, $MSE = 1065.05$. The covariance analyses must be interpreted with particular caution because 44 of the 99 participants failed to record either study or performance times; thus the covariance analyses excluded almost half of the participants.

Discussion

The animation seemed responsible for the superior performance of the TA+anim group on far transfer problems, compared with the other groups. The task-analysis-based text did not seem to play a role in far transfer performance.

The TA+anim condition had a mean performance advantage of roughly 29 points over the other two groups, about a 32% difference. These results represent strong support for the utility of animation in aiding generalization to novel problems. It is important to remember that the performance levels observed on the far transfer problems are measures of knowledge that was not explicitly instructed. This knowledge had to be inferred and generalized from the problem-solving knowledge created during instruction and practice on the near transfer items, with only minimal guidance from the transitional paragraph that preceded the far transfer problems. When considered in this manner, the exhibited level of far transfer performance seems significant from a practical point of view.

The control and TA groups spent roughly 45 min studying and solving problems, whereas the TA+anim group spent about 53 min. Is this 18% increase in time "worth" the 32% improvement in transfer score? Ultimately this is a pragmatic judgment call that must be made by students, teachers, and administrators.

As predicted, the TA and TA+anim groups performed better than did the control group on the near transfer problems. Accuracy differences on the near transfer problems can be largely attributed to the use of task-analysis-based text materials in the training phase. Use of the animation had little effect on near transfer problem-solving performance. However, as mentioned earlier, this finding indicates primarily that carefully constructed training text that is designed to address the information needs of learners can improve performance.

EXPERIMENT 2

Experiment 1 found a significant effect of an animation-based study aid on problem-solving performance. What remained unclear was the aspect or aspects of the animation that fostered the observed performance advantage. Was it the extra study time, the motion in the animation, or, perhaps, the depiction of certain key "events" in the animation? The fact that the group effect on near transfer performance remained with time added as a covariate suggests that the task-analysis-based text aided near transfer performance. The fact that the group effect on far transfer performance disappeared with time added as a covariate suggests that the animation may have helped performance primarily because it led to longer study time. As mentioned earlier, though, it is difficult to draw firm conclusions given the relatively large number of participants who failed to record study and performance times in Experiment 1. One goal of Experiment 2 was to replicate Experiment 1 but to obtain more complete study and performance time information.

A second goal of Experiment 2 was to examine whether any benefit of animation would be attributable to the animation per se or merely the depiction of certain key events in an algorithm's behavior. Therefore, in Experiment 2 a *frames study aid* manipulation was added. The frames study aid was a set of static screen shots from the animation that depicted what was judged (by the experimenters) to be critical points in the animation of the algorithm (see Figure 3 for sample key frames).

A third goal was to examine the benefit that an animation or frames might add to the control text. In Experiment 1 the animation was used in conjunction with only the task-analysis-based text. Pragmatically, if animation or key frames can be shown to aid performance when used in conjunction with a "weaker" text, such as the control text, then they might be adopted by teachers and designers of course materials. Conversely, it is less likely that reliable effects of task-analysis-based text will soon lead to changes in textbooks and lecture notes.

Why would animation aid learning, beyond any benefit obtained from carefully chosen static images of key events? One possibility lies in the notion of "element interactivity" proposed by Sweller and his colleagues (e.g., Tindall-Ford, Chandler, & Sweller, 1997). They argued that when to-be-learned material has multiple components that need to be understood simultaneously (high element interactivity) in order to really grasp the material, then learning and transfer are more difficult. For instance, in one of their experiments they examined students learning to conduct electrical tests of appliances. To understand how to do these tests properly one must simultaneously know, among other things, the settings on the volt meter, where to place the leads of the meter, what the desired reading should be, and where to move the leads as a function of the readings. Tindall-Ford et al. found that training manipulations that functionally increased working memory capacity aided learning.

With respect to the current work, learners studying algorithms must also simultaneously understand multiple aspects of the algorithm, such as the status of the pointer, the value of the top of the stack, and the current operation (e.g., push or pop) so that the learner can grasp the ramifications of a particular operation. An animation may be better able to make the fluid and interactive relationship among these aspects clearer and more salient to the learner, compared with static materials that would encourage a more serial approach.

A static and/or textual presentation can provide a knowledge foundation, whereas an animation is hypothesized to be more effective for helping a learner to understand the interaction of the components. Consistent with the findings of Tindall-Ford et al. (1997), we expected that animations would show their benefits on the more challenging far transfer problems.

For example, for participants in the frames conditions in this experiment, the beginning state of the algorithm was statically represented. After a short period elapsed, a line of pseudocode was displayed above the beginning key frame. When the learner hit the "forward" key, the beginning key frame was replaced by another key frame statically showing the state of the algorithm resulting from the execution of the displayed line of pseudocode.

Users of the frames-based aid were left to infer two things: (a) that movement of the problem elements had occurred and (b) the way in which this movement might have occurred. Users of the animation-based aid were not required to make such inferences, because the movement of the problem elements was clearly displayed to them. If explicit and unambiguous depiction of the movement of the problem elements about the problem space make any significant contribution to performance, then differences would be apparent between those participants using the frames-based study aid and those using the animation study aid. Such differences could then be attributed to the reduction in ambiguity provided by the clear depiction of the dynamics of the algorithm. If no differences were to arise between users of the study aids but both conditions outperformed a no-aid condition, then this might indicate that the actual motion is not a central factor contributing to the performance advantage rendered by graphical study aids.

Interestingly, performance differences in the opposite direction might be linked to the relative difficulty of "filling in the gaps" left in the frames-based aid when the dynamic motion is removed. If the veridical motion of the problem elements is sufficiently implied by the frames-based aid, then participants would be able to mentally animate the elements and override any advantage that an animation might provide. Thus, in contrast to our expectations, the effort to mentally animate the frame-by-frame display might lead to deeper processing and aid subsequent performance.

In this experiment, text type (control versus TA) was fully crossed with study aid type (none, frames, animation). Performance was analyzed as a function of text type and study aid type. However, given that the control text and task-analysis-based text differ in a variety of ways, analyses of times and accuracy as a function of study-aid type were also done separately for the control text conditions and the task-analysis-based conditions.

Expectations

Based on the results from Experiment 1, we expected that the task-analysis-based text conditions would outperform the control text conditions on the near transfer problems. If the use of a study aid can help near transfer performance when one learns from a weaker text, then, for the control conditions, participants receiving the study aids (frames or animation) would outperform the no-aid condition. This difference should hold up even when study time is considered.

With regard to the far transfer problems, we predicted, again based on results from Experiment 1, that there would be no overall effect of text type but that there would be an effect of study aid. It is unclear whether there might be an interaction. For instance, the control text conditions might benefit from the study aids

TABLE 4: Time (minutes) Spent on Near Transfer and Far Transfer Tasks (Experiment 2)

	Condition					
	Control Text			TA-Based Text		
	No Aid (n = 46)	Frames (n = 48)	Animation (n = 46)	No Aid (n = 44)	Frames (n = 49)	Animation (n = 45)
Near transfer	26.26	26.29	27.35	27.91	31.78	29.67
Far transfer	17.96	18.60	17.98	18.16	20.06	18.89

Note: Near transfer time included time spent reading the text, viewing the frames or watching the animation (if available), and solving problems. Far transfer time was spent only solving problems.

more than those in the task-analysis-based text conditions because the former groups would need more support to produce reasonable far transfer. Conversely, it might be the case that only those learners who had the benefit of a stronger text for initial learning would be able to take advantage of the study aids when attempting far transfer problems. Finally, it is not clear whether any of the possible far transfer results mentioned would hold up after time is added as a covariate.

Method

Participants. Participants were 300 undergraduates at the Georgia Institute of Technology who participated to receive credit in their psychology course. No participants were computer science majors, and they did not have programming experience or computer science courses at the high school or college level as indicated by their questionnaire responses. Of the initial 300 participants, data from 22 were removed because they failed to record study and problem solving times; thus data from a total of 278 participants were used in the analyses.

Apparatus. The same apparatus used in Experiment 1 was used here.

Design and procedure. Participants were randomly assigned in approximately equal numbers to six separate groups formed by crossing text type (control, task analysis) and aid type (none, frames, animation). See Table 4 for exact numbers. Training sessions for all six conditions took place in a large computer lab. Between 1 and 16 individuals participated during any one session.

The overall procedure was the same as that used in Experiment 1. General cognitive ability was assessed as in Experiment 1. For participants in the animation and frames conditions, the difference in the study aids was that the transitional motion occurring between states in the algorithm was not displayed in the key frame study aid. In the frames conditions, various key frames of the full animation were displayed. Specifically, the frames study aid statically presented a graphical representation of a beginning state of a stack, displayed a line of pseudocode, displayed the resulting state of the stack, then displayed another line of pseudocode, and so on.

The same near and far transfer problems used in Experiment 1 were used here and were scored in the same way.

Results

Cognitive abilities. The group means for the cognitive ability measures are shown in Table 5. Of the 278 participants, 29 did not provide GPA or SAT scores, so the means in Table 5 are based on 249 participants (the number of participants in each condition who provided data on these measures is shown in the table). There were no significant differences among the groups for GPA or SAT quantitative. However, participants assigned to the different study aid conditions differed on SAT verbal scores, $F(2, 243) = 7.01$, $p = .001$, $MSE = 6385.80$ (no aid = 576.30, frames = 595.82, animation = 623.25). However, when SAT verbal scores were used as a covariate in the analyses of times and accuracy, they did not have a significant effect and thus will not be discussed further.

TABLE 5: Mean SAT Verbal Score, SAT Quantitative Score, and GPA as a Function of Condition (Experiment 2)

	Condition					
	Control Text			TA-Based Text		
	No Aid (*n* = 41)	Frames (*n* = 42)	Animation (*n* = 39)	No Aid (*n* = 40)	Frames (*n* = 46)	Animation (*n* = 41)
SAT verbal	576.10	606.43	633.08	576.50	585.22	613.41
SAT quantitative	645.12	653.57	637.94	613.75	644.57	653.41
GPA	2.88	2.87	2.93	2.85	2.87	2.83

Near and far transfer: Study and performance time. There was a significant effect of type of text on the combined time to study the training material and solve the near transfer problems, $F(1, 272) = 13.45$, $p = .0003$, $MSE = 51.23$ (control text = 26.63 min, task analysis text = 29.78 min.), but there was no effect attributable to study aid type, $F(2, 272) = 1.83$, $p = .16$ (see Table 4). The interaction was also not significant, $F(2, 272) = 1.94$, $p = .15$. Because there was no overall effect for aid type, there was no reason to analyze the effect of aid type separately for each text type condition. These results contrast with those from Experiment 1, in which the study aid seemed to be the biggest factor influencing study time but text type did not seem to matter.

There were no significant differences on the time to complete the far transfer problems attributable to text type, $F(1, 272) = 2.66$, $p = .10$, $MSE = 19.14$, or aid type, $F(2, 272) = 2.11$, $p = .12$. The interaction was also not significant, $F(2, 272) = 0.48$, $p = .62$. Because there was no overall effect for aid type, there was no reason to analyze the effect of aid type separately for each text-type condition. These results contrast with those from Experiment 1, in which participants who had the animation took longer to complete the far transfer problems.

Near and far transfer: Problem solving accuracy. There were significant differences on near transfer scores attributable to text type, $F(1, 272) = 15.03$, $p = .0001$, $MSE = 46.42$ (control text = 43.03, task analysis text = 46.20), and aid type, $F(2, 272) = 7.74$, $p = .0005$ (no aid = 42.38, frames = 46.13, animation = 45.34; see Table 6). The interaction was not significant, $F(2, 272) = 0.12$, $p = .89$. The beneficial effect of the task-analysis-based text is consistent with the results from Experiment 1.

Because there was a main effect of aid type on near transfer scores, it was appropriate to examine this effect separately for the two text-type conditions. If only control text participants are considered, then there is an effect

TABLE 6: Scores on Near and Far Transfer Tasks (Experiment 2)

	Condition					
	Control Text			TA-Based Text		
	No Aid (*n* = 46)	Frames (*n* = 48)	Animation (*n* = 46)	No Aid (*n* = 44)	Frames (*n* = 49)	Animation (*n* = 45)
Near transfer	40.74	44.81	43.54	44.02	47.45	47.13
Far transfer	105.98	103.81	119.46	112.91	121.92	118.69

Note: Maximum possible score was 53 for near transfer and 139 for far transfer.

attributable to aid type on near transfer performance, $F(2, 137) = 4.00$, $p = .021$, $MSE = 50.78$. Pairwise comparisons among the aid conditions for control participants showed that the frames condition outperformed the control condition ($p = .006$; required $p = .05$), but no other differences were significant, although the animation condition marginally outperformed the no-aid condition ($p = .06$).

If only task analysis participants are considered, then there is an effect of aid type on near transfer performance, $F(2, 135) = 3.86$, $p = .024$, $MSE = 41.99$. Pairwise comparisons among the aid conditions for task analysis participants showed that both the frames and animation conditions outperformed the control condition ($ps = .012$ and $.025$, respectively) but did not differ significantly from each other.

There was a significant difference on far transfer performance attributable to text type, $F(1, 272) = 6.09$, $p = .014$, $MSE = 746.05$ (control text = 109.75 earned points, task analysis text = 117.84 earned points). There were marginal effects attributable to aid type, $F(2, 272) = 2.90$, $p = .057$ (no aid = 109.44, frames = 112.87, animation = 119.07), and the interaction of text type and aid type, $F(2, 272) = 2.84$, $p = .06$. Although the effect of aid type was only marginal, this effect is consistent with the result from Experiment 1 in which the group who saw the animation outperformed the other groups.

Because there was a (marginal) main effect of aid type, it was appropriate to examine this effect separately for the two text type conditions. If only control text participants are considered, then there is an effect attributable to aid type on far transfer performance, $F(2, 137) = 4.30$, $p = .016$, $MSE = 777.27$. Pairwise comparisons among the aid conditions for control participants showed that the animation condition outperformed the control and frames conditions ($ps = .022$ and $.007$, respectively) but that the control and frames conditions did not differ significantly.

If only task analysis participants are considered, then there is no effect of aid type on far transfer performance, $F(2, 135) = 1.34$, $p = .27$, $MSE = 714.38$.

Relation between time to completion and problem-solving accuracy. With time spent studying and solving the first problem set entered as covariate, an analysis of covariance of near transfer performance continued to show an effect of type of text, $F(1, 271) = 9.47$, $p < .002$, $MSE = 44.42$, and aid type, $F(2, 271) = 6.40$, $p = .002$. The interaction remained nonsignificant, $F(2, 271) = 0.41$, $p = .66$.

Because there was a main effect of aid type, it was appropriate to examine this effect separately for the two text type conditions. If only control text participants are considered, then there is an effect attributable to aid type on near transfer performance when study time and performance time are taken into account, $F(2, 136) = 4.04$, $p = .02$, $MSE = 49.07$. If only task analysis participants are considered, then there is a marginal effect of aid type on near transfer performance, $F(2, 134) = 2.55$, $p = .081$, $MSE = 40.01$.

With time spent studying and solving the first problem set and time spent solving the second problem set entered as covariates, an analysis of covariance of far transfer performance continued to show an effect of type of text, $F(1, 270) = 5.07$, $p < .025$, $MSE = 695.66$. The effect of aid type moved from marginal significance to conventional significance, $F(2, 270) = 3.23$, $p = .041$. The interaction continued to be marginally significant, $F(2, 270) = 2.92$, $p = .056$.

Because there was a main effect of aid type, it was appropriate to examine this effect separately for the two text type conditions. If only control text participants are considered, then there is an effect attributable to aid type on far transfer performance when study time and performance time are taken into account, $F(2, 135) = 5.31$, $p = .006$, $MSE = 717.55$. If only task analysis participants are considered, then there is no effect of aid type on far transfer performance, $F(2, 133) = 0.78$, $p = .46$, $MSE = 681.98$.

Discussion

Performance on near transfer problems was better when learners either worked from an improved text or used a study aid. The study aids helped learners who studied from either the control or task-analysis-based texts. Far transfer performance also benefitted from the task-analysis-based text and the presence of study aids. For control text participants, the animation study aid benefitted far transfer performance more than did either the static frames aid or no aid. However, for task-analysis-based text participants, there did not appear to be any effect of study aid on far transfer performance.

If study time is taken into account when considering near transfer performance, the effects of text type and aid type remain. The benefit of aid type in the covariance analysis was present for both control and task analysis participants when they were analyzed separately. If study time and problem solving time is taken into account when considering far transfer performance, there also continues to be effects of text type and aid type. The benefit of aid type on far transfer in the covariance analysis was present for control participants but not for task analysis participants when they were analyzed separately.

The overall pattern of results suggests that an improved text aided both near and far transfer performance. The study aids seemed most useful for control text participants, and the animation appeared more successful than the static frames in supporting far transfer performance for these learners.

GENERAL DISCUSSION

A task analysis was used to guide the creation of the task-analysis-based text and the frames and animated study aids. The task analysis was an attempt to ensure that the critical elements to be learned had been identified so that they could be included in the task-analysis-based text and the study aids. The fact that the task analysis text and study aids helped learning, as compared with the control text, is an indicator that the task analysis was useful. Of greater importance to the present paper, however, is that the frames and animated study aids boosted performance for learners who used the control text, even when factoring in the time to use these study aids. This suggests that study aids that illustrate critical elements as identified by the task analysis can improve learning when students study weaker texts. The fact that the animation seemed even more effective than static frames for learners using the weaker control text suggests that animation can add something unique to learning, beyond what is contributed by the static depiction of key elements.

When Can Animation Have the Biggest Impact?

The animation used in this study depicted what might be called *first order* motion – that is, simple "Point A to Point B" motion. Therefore, the dynamic motion that is displayed in the animation could be inferred, thus allowing users of the frames-based aid to accurately mentally animate the scene. Researchers have argued that actively generated information is more memorable than information passively received (McNamara, Kintsch, Songer, & Kintsch, 1996). It may be that the active, bridging inference made necessary by a frames-based study aid can furnish a learning experience that is slightly more robust than passively consuming the full motion animation. However, if the motion of the problem elements is more complex, perhaps including an intervening point, then viewers of the frames-based study aid might not be as successful in inferring the correct pattern of motion and mentally animating the scene in the way made explicit by the animation. Future work in this area should address the relative effect of dynamic motion in situations where more complex motion must be described or inferred. Such a research focus would help explain why some studies, such as the present one, find a benefit of animation over static study aids whereas other studies have not (e.g., Mayer, 1997).

The study aids used here were modest in that the operations performed were predetermined and did not permit user-defined parameters or inputs. Future studies can examine whether permitting learners to determine the inputs to the animations/key frames would cause study aids to produce still greater learning benefits, even in cases in which the text is strong.

From a practical point of view, if the use of frames or animation can improve performance for learners studying weaker texts (presumably many textbooks would represent weaker texts), then simply adding them to existing courses may improve learning without the more drastic solution – which is less likely to be implemented – of rewriting textbooks based on task analyses.

APPENDIX A

Sample Far Transfer Problems

1. Consider an empty queue of Size 7. Perform the following operations on it, being sure to show the outcome of each operation as well as momentary values for head and tail.

   Enqueue(13)
   Enqueue(40)
   Enqueue(75)
   Enqueue(80)
   Dequeue
   Enqueue(23)
   Enqueue(65)
   Enqueue(31)
   Dequeue
   Enqueue(63)
   Enqueue(87)
   Dequeue

2. Of the two algorithms you have dealt with today (stack and queue), which would be most appropriate for reading in an entire list of numbers and then outputting them in reverse order? Why is the one you chose most appropriate?

ACKNOWLEDGMENTS

This research was supported by Air Force Office of Scientific Research Grant F49620-98-1-0362.

REFERENCES

Baecker, R. M., & Sherman, D. (1981). *Sorting out sorting* [30-min. color sound film, presented at ACM SIGGRAPH, 1981, excerpted and reprinted in *SIGGRAPH video review 7,* 1983]. San Francisco: Morgan Kaufmann.

Baecker, R. (1998). Sorting out sorting: A case study of software visualization for teaching computer science. In J. Stasko, J. Domingue, M. H. Brown, & B. A. Price (Eds.), *Software visualization: Programming as a multimedia experience* (pp. 369–381). Cambridge, MA: MIT Press.

Brown, M. H. (1988). Perspectives on algorithm animation. In *Proceedings of the Association for Computing Machinery Special Interest Group on Computer Human Interaction Conference on Human Factors* (pp. 33–38). New York: ACM.

Byrne, M. D., Catrambone, R., & Stasko, J. T. (1999). Examining the effects of animation and prediction in student learning of computer algorithms. *Computers and Education, 33,* 253–278.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1998). *Introduction to algorithms.* New York: McGraw-Hill.

Hansen S., Schrimpsher, D., & Narayanan, N. H. (1999). From algorithm animations to animation-embedded hypermedia visualizations. In *Proceedings ED-MEDIA 1999: World Conference on Educational Multimedia, Hypermedia & Telecommunications* [CD-ROM]. Norfolk, VA: Association for the Advancement of Computing in Education.

Hays, T. (1996). Spatial abilities and the effects of computer animation on short term and long term comprehension. *Journal of Educational Computing Research, 14,* 139–155.

Hegarty, M. (1992). Mental animation: Inferring motion from static displays of mechanical systems. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 18,* 1084–1102.

Mayer, R. E. (1997). Multimedia learning: Are we asking the right questions? *Educational Psychologist, 32,* 1–19.

Mayer, R. E. (2001). *Multimedia learning.* Cambridge, UK: Cambridge University Press.

McNamara, D. S., Kintsch, E., Songer, N. B., & Kintsch, W. (1996). Are good texts always better? Interactions of text coherence, background knowledge, and levels of understanding in learning from text. *Cognition and Instruction, 14,* 1–43.

Mousavi, S. Y., Low, R., & Sweller, J. (1995). Reducing cognitive load by mixing auditory and visual presentation modes. *Journal of Educational Psychology, 87,* 319–334.

Price, B. A. , Baecker, R. M. , & Small, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages and Computing, 4,* 211–266.

Reiber, L. P. (1991). Animation, incidental learning, and continuing motivation. *Journal of Educational Psychology, 83,* 318–328.

Reiber, L. P., & Kini, A. S. (1991). Theoretical foundations of instructional applications of computer-generated animated visuals. *Journal of Computer-Based Instruction, 18*(3), 83–88.

Rigney, J., & Lutz, K. A. (1976). Effect of graphic analogies of concepts in chemistry on learning and attitude. *Journal of Educational Psychology, 68,* 305–311.

Scaife, M., & Rogers, Y. (1996). External cognition: How do graphical representations work? *International Journal of Human-Computer Studies, 45,* 185–213.

Seaman, M. A., Levin, J. R., & Serlin, R. C. (1991). New developments in pairwise multiple comparisons: Some powerful and practical procedures. *Psychological Bulletin, 110,* 577–586.

Shaffer, J. P. (1986). Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association, 81,* 826–831.

Stasko, J. T. (1990). TANGO: A framework and system for algorithm animation. *Computer, 23*(9), 27–39.

Stasko, J. T. (1992). Animating algorithms with XTango. *SIGACT News, 23*(2), 67–71.

Stephenson, S. D. (1994). The use of small groups in computer-based training: A review of recent literature. *Computers in Human Behavior, 10,* 243–259.

Tindall-Ford, S., Chandler, P., & Sweller, J. (1997). When two sensory modes are better than one. *Journal of Experimental Psychology: Applied, 3,* 257–287.

Richard Catrambone is an associate professor of psychology at the Georgia Institute of Technology. He received his Ph.D. in experimental psychology in 1988 from the University of Michigan.

A. Fleming Seay is a Ph.D. student of human-computer interaction at Carnegie Mellon University. He received his M.S. in experimental psychology in 2000 from the Georgia Institute of Technology.

Date received: October 4, 2000
Date accepted: May 7, 2002