Evaluating Animations as Student Aids in Learning Computer Algorithms

Michael D. Byrne
Psychology Department
Rice University
Houston, TX 77005-1892

Richard Catrambone
School of Psychology
Georgia Institute of Technology
Atlanta, GA 30332-0170

John T. Stasko*
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

Abstract

We conducted two experiments designed to examine whether animations of algorithms would help students learn the algorithms more effectively. Across the two studies we used two different algorithms--depth-first search and binomial heaps--and used two different subject populations--students with little or no computer science background and students who were computer science majors--and examined whether animations helped students acquire procedural and conceptual knowledge about the algorithms. The results suggest that one way animations may aid learning of procedural knowledge is by encouraging learners to predict the algorithm's behavior. However, such a learning improvement was also found when learners made predictions of an algorithm's behavior from static diagrams. This suggests that prediction, rather than animation per se, may have been the key factor in aiding learning in the present studies. These initial experiments served to highlight a number of methodological issues that need to be systematically addressed in future experiments in order to fully test the relationship between animation and prediction as well as to examine other possible benefits of animations on learning.

*Author for correspondence.

Key Words: media in education, pedagogical issues, evaluation of CAL systems, applications in subject areas, interactive learning environments.

Educators constantly are seeking new ways to improve instruction, to facilitate learning, and to hold the attention of their students. The power of computers to store vast quantities of information and to simulate environments and conditions that would otherwise be unavailable makes them an intriguing possibility as an instructional aid.

Our work examines whether multimedia capabilities, particularly animation, can facilitate learning about computer algorithms and programs. The use of pictures and visualizations as educational aids is accepted practice; textbooks are filled with pictures and instructors often diagram concepts on the blackboard to assist an explanation.

Animation goes one step further. While static visualizations can provide people with the essence of how something looks, is laid out, or is constituted, animation appears better able to explain a dynamic, evolving process. Animations may aid learners in constructing a mental model (a mentally-runnable simulation) of various "processes" such as the movements of components of a mechanical system. Various studies have suggested that mental models help learners more accurately predict the behaviors of various processes or systems such as a system made up of a set of hypothetical components for a "phaser bank" (Kieras & Bovair, 1984) or a set of interconnected pulleys (Hegarty & Just, 1993).

In general, animation helps viewers track patterns and observe relationships in a display (Robertson, Card, & Mackinlay, 1993). For instance, animation has been used to provide on-line help with interface tasks (Sukaviriya, 1990), and to help users follow interface operations (Baecker & Small, 1990).

1 Background

Our research focus is the use of animation to help teach programming and to help teach students how algorithms work. This is called *algorithm animation* (Brown, 1988a), and is one particular instance of *software visualization* (Stasko, Domingue, Brown, & Price, 1998; Price, Baecker, & Small, 1993) which is the use of images, graphics, and animation to illustrate computer algorithms, programs, and processes. The video *Sorting Out Sorting*, presented at

SIGGRAPH '81, is generally credited with initiating the field of algorithm animation (Baecker, 1998). It showed views of data being sorted by different algorithms to help students understand how the algorithms work and how they compare. Since then, many algorithm animation systems have been built (Brown, 1988b, 1991; Brown & Najork, 1997; Gloor, 1992; Naps, 1990; Roman, Cox, Wilcox, & Plun, 1992; Stasko, 1990; Stasko & Kraemer 1993) and a number are publicly available. Many universities use the systems to help teach algorithms to students. This type of instruction is facilitated by the availability of an “electronic” classroom to conduct lectures and by the availability of powerful computers with multimedia capabilities on which students can work out of the classroom (Bazik, Tamassia, Reiss, & van Dam, 1998). Animations are now relatively easy to design and produce with computers.

Much more research has been performed on algorithm animation system technologies than on the pedagogical effects of such systems. The intuition of computer scientists has led them to believe that the animations must provide a learning benefit, but prior experimental studies of the influence of algorithm animation on student understanding have provided mixed results. Some studies have found benefits, but not at the levels hoped for by system developers, while others did not uncover benefits.

One of the first studies in this area examined students learning about the pairing heap data structure and algorithm (Stasko, Badre, & Lewis, 1993). The study included two test conditions: students learning about the algorithm by reading only a textual explanation, and students learning about the algorithm using the text and interacting with an animation of the algorithm. Each group had an identical amount of time to study the algorithm, which was followed by a post-test including a variety of questions about the algorithm. The post-test was mostly questions about the procedural, methodological operations of the pairing heap, but it included a few concept-oriented questions as well. There was no significant difference in the two groups' performances on the post-test, but the trend favored the animation group.

Lawrence's doctoral dissertation studied a variety of introductory computer science algorithms such as sorting and graph algorithms, and examined the addition of animation to help

teach them (Lawrence, 1993). In one experiment she found a significant learning benefit, as measured on a post-test, when students used an algorithm animation in a laboratory session outside of a class (Lawrence, Badre, & Stasko, 1994). The learning benefit involved the ability to understand and perform algorithm procedures and to answer conceptual questions about the algorithm. The students who benefited were the ones who were able to enter trial data and values to the algorithm and then watch the resulting animation. The learning benefit was relative to two other groups: those students who did not attend the after-class laboratory and those who did attend a lab session, but viewed animations of the algorithm on predefined data sets.

Hansen, Schrimpsher, and Narayanan (1998) did find significant learning benefits via using a hypermedia system that included animation to help teach students about algorithms. In their study, however, the learning benefit may have resulted from any number of attributes of the system, so it is inappropriate to attribute the improvements solely to the animations.

Jarc's doctoral dissertation examined algorithm visualizations deployed over the World Wide Web (Jarc, 1999). He examined students learning about an algorithm by viewing a visualization as compared to students who viewed the animation and answered interactive questions about the algorithm. He hypothesized that the interactive questions would aid learning, but no significant difference was found, and the trend surprisingly even favored the passive viewing group.

Other studies have focused on more qualitative issues concerning student learning from algorithm visualizations. Douglas, Hundhausen, and McKeown (1996) found that the visual depictions commonly used in algorithm visualizations do not accord well with student-generated conceptualizations of the algorithms. This non-congruence may help to explain some of the mixed results in the earlier empirical studies. They argue for a more ethnomethodological approach to evaluating the pedagogical benefits of animations.

Gurka and Citrin (1996), by reviewing earlier empirical studies and carrying out experiments of their own, identified seven factors to consider in studying pedagogical benefits of algorithm animations: animation system usability; animation system availability; training on use of the

animation system; algorithm complexity; animation quality, including graphic design; how, where and when the animation is used; and individual differences among learners.

Because these results have been so uneven, it makes sense to examine a broader context of related work involving the use of animations to assist learning in domains other than algorithms and programming. Much prior work exists (Mayer, 1997; Rieber, 1991), but again the results are perhaps best characterized as mixed.

Rieber and his colleagues found that the addition of animation to help teach science concepts such as Newtonian mechanics had no effect on learning by adults as measured by a multiple choice post-test (Rieber, Boyce & Assad, 1990), but it did have a positive effect on children (Rieber, 1990). They speculate that the animations helped the children form internal mental images of the processes, but this was not necessary for the adults. A later study examined an animated simulation's role in helping students to understand velocity and acceleration (Rieber, 1996). Graphical feedback did not influence explicit understanding as measured by a post-test, but it did benefit tacit knowledge of the concepts as measured by student performance on a game involving the concepts.

Palmiter and Elkerton (1993) studied the use of animation to aid computer authoring tasks (see also Palmiter, 1993). Animation initially assisted both accuracy and speed, but after one week had elapsed, the subjects exposed to animations actually had regressed behind the non-animation subjects.

Park and Gittelman (1992) compared animated versus static visual displays in helping students troubleshoot electronic circuits. They did find a significant benefit of animated visuals, that is, the animated visuals supported repairs with fewer trials. The investigators argued that a key factor in the learning benefits of such displays is an appropriate match to the specific learning requirements of the particular tasks being examined.

Studies by Mayer and Anderson (1991, 1992) and Mayer and Sims (1994) found that animation did help college students learn about mechanical concepts when measured by a creative problem solving post-test. The benefits from the animations occurred when the animations were

accompanied by verbal explanations of the concepts. Mayer (1997) argues that these results support the dual-coding theory advocated by Paivio (1990), which articulates how multiple representations of a problem help learners understand new concepts and build connections among the representations.

If we shift our focus from animation to multimedia in general, a number of prior studies are pertinent. Pane, Corbett, and John (1996) found that a simulation (biology demonstration) helped learners because each time it was run it probabilistically varied the developmental sequence being shown. Thus, learners received more information than those who just saw a movie multiple times. Such a simulation is similar to the types of animations we are studying in that the initial conditions can be varied each time an animation is run. Faraday and Sutcliffe (1997) studied multimedia presentations and suggested that for animation to be an effective tool, it must be used to draw learners' attention to appropriate features of the presentation.

Although the above studies of animation clearly are pertinent to our work, we are reluctant to take too many implications from them with respect to algorithm animation. Algorithm animations are fundamentally different than the types of animated simulations described above. In those studies, the animations usually portrayed physical processes, thus the animations represent real-world phenomena. Algorithm animations, conversely, are symbolic, graphical representations of computer algorithms that have no physical manifestation. Providing a learner with a visual representation for something abstract could be just as important to the learning process as the fact that the representation is an animation.

Advocates of algorithm animation identify the concrete visualization of abstract concepts as one of the central reasons for their belief in the technology's value. Abstract, conceptual phenomena are notoriously difficult for students to understand. Algorithm animations help make concepts concrete. The second reason for the belief that algorithm animations should aid learning involves the animated visuals. Computer algorithms are dynamic processes that evolve over time. Static images presumably do not convey this time evolution as well as animated visuals.

These two reasons fuel computer scientists' belief in the pedagogical value of algorithm animations and they form the basis of our "theory" of why algorithm animations benefit student learning. Clearly, other processes may be involved such as benefits from dual coding or dual representations (visual and textual) and the stimulation of mental models in learners. However, our experiences in teaching algorithms to students for a number of years underline the difficulties that students have with these abstract concepts and the value in providing concrete, dynamic representations to the concepts. Thus, an examination of the importance of dynamic versus static presentations seems an appropriate place to begin an empirical investigation.

2 Study Overview

This article describes experiments involving two different algorithms: depth-first search and binomial heap. Our primary objective was to identify how the presence or absence of an algorithm animation in the instructional environment influenced learning in students.

In the experiments we utilized two already existing algorithm animations rather than building new ones. This was because we have used these animations in classes and they have been widely distributed along with accompanying animation systems, thus we sought to validate this use in a pragmatic manner. The animations originally were constructed to illustrate the important steps of the two algorithms. The depth-first search animation, for instance, shows the classical vertex and edge representation of a graph. A token (represented as a circle) moves about the graph during the search, vertices change color as they are visited, and the order of visitation is labeled. Similarly, the binomial heap animation illustrates binomial heaps just as they are shown in most algorithms textbooks. Additionally, operations on the heaps are illustrated by smooth animated movements of the nodes.

Because of our desire to make our initial experiments have some ecological validity to classroom settings, we presented the animations in the context of a larger learning situation. Students in our studies watched a brief video-lecture on the algorithm and also read a short text on the algorithm before watching an animation. As mentioned earlier, the animations had visual

representations as shown in current textbook presentations in order to make the animations consistent with present classroom learning.

Following the learning sessions, students completed a post-test designed to measure their understanding of the algorithms. Again, this overall methodology was tailored to mimic a traditional university course "teach-then-test" learning scenario. Clearly, many people today question this traditional scenario as being the best scenario for learning, but that debate is outside the scope of our study. We simply mimicked what occurs in typical classrooms.

The post-test was designed to assess students' understanding of the algorithm just taught. Our definition of "understanding" an algorithm directly influences the form of the exercises/questions on the post-test. Here again, we chose an approach consistent with traditional testing instruments in college courses. Our post-test questions, in general, evaluated the students' procedural and conceptual knowledge of the algorithms (Experiment 1 examined only procedural knowledge; Experiment 2 examined procedural and conceptual knowledge). The questions involved short answers whose correctness could clearly be determined. By "procedural" we mean questions relating to the methodological, step-by-step operations of the algorithms on different input data sets. For example, given a binomial heap in a particular configuration, could a student perform an insert operation and illustrate the heap's configuration after the new element was inserted? By "conceptual" questions we mean the type of declarative, fact-based questions common on exams. These questions typically involve *properties* of an algorithm. For instance, conceptual questions may ask students what the running time of an algorithm is or whether a certain size tree data structure could exist as the algorithm runs.

Clearly, these two classes of questions are interrelated. Conceptual knowledge about the properties of an algorithm can help a learner to carry out the algorithm's steps. Similarly, being able to perform the step-by-step operations of an algorithm may assist a learner in determining the veracity of a conceptual question about it. Nonetheless, we characterize post-test questions as fitting into these two broad classes for the experiments described here.

Thus, our operational definition of understanding includes a learner's ability to predict algorithm performance on novel problems and to answer certain conceptual questions. Obviously, understanding may involve other measures such as the ability to implement an algorithm and the ability of a student to learn a new algorithm as a function of having studied a prior algorithm. While these measures are pertinent, they were outside the scope of our initial focus.

Given this operationalization of understanding, it is reasonable to ask how learners acquire such an understanding and how animations might aid the acquisition process. As discussed earlier, prior research has hypothesized that animations and simulations help learners to build mental models of processes. Such models help learners to reconstruct, run, and simulate the processes. Furthermore, in carrying out earlier experiments, we noticed how the students often would anticipate the next movement, flash, or color change of an animation reflecting the next algorithm operation. Prior research has suggested that when learners acquire a mental model for a system or process, one of the benefits is that they are better able to make predictions about the behavior of the system (e.g., Kieras & Bovair, 1984).

However, it is possible to turn this logic on its head. That is, if learners are explicitly led to make predictions about the animation during training, this might help the development of a mental model of the algorithm. Thus, if the benefit of seeing an animation is that it leads a learner to make predictions, then learners who see an animation, or learners who see static images but are asked to make predictions based on the images, might be more likely to develop a mental model which aids them when they attempt to answer post-test questions about the algorithm.

Therefore, we hypothesize that animations might encourage the learner to make and test predictions of what is going to happen at each step of the algorithm--perhaps by making it relatively easy for the learner to make and test the predictions. For example, viewing an animation of a graph traversal may communicate the pattern of operations to the viewer and facilitate anticipation of its next step. This prediction element could help the learner understand the algorithm better than a learner who simply examines static images and thus, presumably is led to be more passive in how he or she interacts with the material. A learner could certainly make

predictions from a static textual/graphic presentation of an algorithm, but perhaps the advantage of an animation is that it will encourage a learner to *spontaneously* make the predictions without being prompted and will provide the learner with rapid feedback about the accuracy of his or her predictions.

Because feedback is a critical element in various psychological theories of how people acquire procedural knowledge (e.g. Anderson, 1993), we decided to concentrate on the "make and test predictions" aspect of algorithm animation. Both of the experiments presented here manipulate whether learners see animations and whether they are asked to make explicit predictions about algorithm behavior (and receive feedback from their predictions). If animations provide advantages to a learner above and beyond those conferred by simply testing predictions, learners exposed to the animations should outperform those making explicit predictions from static images. On the other hand, if animations primarily encourage learners to make predictions--either explicitly or implicitly--and to examine the accuracy of their predictions, then learners in the "making predictions" conditions of the experiments should perform equally well regardless of whether they learn from animations or static images.

3 Experiment 1

Experiment 1 involved students learning about depth-first search. In a depth-first search, a graph is searched from vertex to vertex by exploring a path from a given vertex as far as that path will go. The path is then backtracked until a vertex is reached that has a path (or "edge") going from it that has not yet been explored. By convention, the vertices are assigned letters and the search starts at the vertex named with the first letter occurring alphabetically. For instance, in Figure 1 the search would start at vertex "a." If more than one edge can be followed from this vertex, the edge connected to the vertex with the lowest letter is chosen first. In Figure 1, the possible choices would be to go to vertices "b," "c," "d," "e," or "f." The edge to vertex "b" is chosen since "b" is the lowest letter of the possible choices. When that vertex is reached, the search continues to the next unvisited vertex connected to the just-reached vertex. This process

continues until a vertex is reached that contains no unexplored edges. At this point, the search backs up to the "parent" of the present vertex and an unexplored edge (if there is one) from that vertex is taken. In Figure 1 this means that after vertex "b" is visited, the search would back up to vertex "a" and then one of the remaining unexplored vertices connected to "a" would be visited (i.e., one of the edges leading to vertex "c," "d," "e," or "f" would be taken; the edge to vertex "c" would be chosen since "c" is the lowest letter of the possible choices). If no unexplored edges exist, the search backs up again. This process continues until all vertices have been visited and the search has backed up to the starting vertex.[1]

The animation of the depth-first search algorithm that we utilized in the experiment was built with the Polka animation system (Stasko & Kraemer, 1993) and utilizes the traditional view of a graph as a set of vertices and edges. A frame from an animation on an example graph is shown in Figure 1. The vertices are represented by black squares and identified by a letter label. The edges are simply lines drawn between the vertices. The animation utilizes a moving circle (seen between vertices J and O in Figure 1) to represent the progression of the search. The circle moves throughout the graph incrementally and smoothly, traversing edges and encountering vertices. When the search is moving to encounter a new vertex, the circle is solid black. When the search is backtracking from a vertex, the circle is a black outline. When a vertex is first discovered in the search, the vertex becomes solid green and a number representing the order of discovery is placed beside the vertex. When the search moves back past a vertex (i.e., all of its descendants in the depth-first search tree have been found) then the vertex changes to solid blue.

Insert Figure 1 about here

3.1 Method

Participants. Participants were 88 undergraduate, non-computer science majors at the Georgia Institute of Technology who participated for extra credit in a psychology course.

Participants were screened to ensure they had no previous experience with the depth-first search algorithm. The participants were generally high-ability students with a mean self-reported SAT-Quantitative score of 645 and SAT-Verbal score of 535.

Materials and Procedure. Participants were run individually. Participants first filled out the consent form and a screening questionnaire that included self-reports of SAT scores and grade-point average. Participants were then shown a six-minute videotaped lecture given by one of the authors (JS) that covered depth-first search. After watching the videotape, participants were given a three-page text describing the depth-first search algorithm. This text was adapted from the chapter covering depth-first search in a standard algorithms textbook (Cormen, Leiserson, & Rivest, 1990) and included pictures of graphs similar to those that would be seen on the post-test.

Participants were divided into four groups: no-animation/no-prediction, animation/no-prediction, no-animation/prediction, animation/prediction. Participants in the no-animation/no-prediction condition were given 10 minutes to read the text, all others were given five minutes. The reason the former group was given additional time to study the text was to attempt to roughly equate the time spent in the training phase. Those who received only five minutes with the text then either watched the algorithm animation (animation/no-prediction condition), made explicit predictions while watching the animation (animation/prediction condition), or made explicit predictions from static printed graphs (no-animation/prediction condition). This took approximately five minutes. All participants then received the post-test (on paper), on which they had unlimited time to work.

Participants in the no-animation/prediction condition were given two graphs on paper and asked to predict the order in which the vertices would be visited. An example of such a task is presented in Appendix A. If they made an error on any prediction, the error was immediately corrected by the experimenter. The other half of the participants who made explicit predictions made those predictions using the animation. The animation was halted immediately after a vertex was visited and the participant asked to state which vertex they thought would be visited next. Errors made by these participants were not pointed out by the experimenter since the animation

displayed the correct answer. Thus, the feedback in both the static and animation cases merely indicated to the participant the next vertex to be visited.

The primary dependent variable in this experiment was the score the participants earned on a post-test. Questions on the post-test were divided into two categories: "basic" and "challenging." Basic questions were those that required the participant to determine a single next step of a search or that required determination of a complete search on a graph very similar to an example participants had already seen. Challenging questions were those that involved complete searches of novel graphs. Examples of questions of both types are presented in Appendix B. Questions on the post-test were scored stringently as being either completely correct or incorrect. In addition, the number of training errors made by participants in the prediction condition were also recorded.

The videotaped lecture was displayed on a 19" color television and participants were free to adjust the sound volume to suit their preference. The animations were presented on a Sun SPARCStation2 with a 19" color monitor at a preset speed determined by pilot testing.

Design. The independent variable was training condition and there were 22 participants per condition. Although animation was crossed with prediction, the resulting design was not strictly a 2 x 2 design since the two animation groups and the no-animation/prediction group all had the opportunity to work with graphically-oriented materials (either the animation or the graphs used for making predictions) after reading the introductory text while the no-animation/no-prediction group did not. Thus, the initial analyses were done with one-way analyses of variance using condition as the single independent variable with four levels.

There were two hypotheses concerning animation and prediction that we wished to evaluate:

[1] Animation aids learners because it encourages them to test predictions. If this is the case, then the animation/prediction, animation/no-prediction, and no-animation/prediction conditions should all be equivalent to each other and superior to the no-animation/no-prediction. Evaluating this hypothesis requires a planned comparison between the three former groups and the latter group.

[2] Animation benefits learners in ways that are distinct and independent from the benefits (if any) of prediction testing. This can be tested by a planned comparison between the animation/prediction and no-animation/prediction conditions.

3.2 Results

Our first concern was whether or not the participants, who were bright but not sophisticated with respect to computer algorithms, could understand even the basics of the depth-first search algorithm on the basis of the materials they had seen. Participants performed very well on the basic post-test questions, scoring an average of 7.58 out of 8, which is about 95% correct. There were no group differences on the basic post-test questions, $F(1, 3) = 1.61$, $p = .19$, $MSE = 3.24$, so participants most likely grasped the fundamentals of the algorithm and could correctly search the post-test isomorphs of the graphs they had seen during training. Groups means and standard deviations for the basic questions are presented in Table 1.

Insert Tables 1 and 2 about here

It seems, then, that participants did not have trouble understanding the procedural fundamentals of depth-first search. We can, then, assess the effects of animation on learning the more complex and subtle aspects of the algorithm by comparing performance on the more challenging questions on the post-test. For the challenging portion of the post-test, each problem was scored as either being correct (the participant gave the complete ordered list of visited vertices correctly) or incorrect (the participant made one or more errors). There were seven such problems on the post-test, and the overall mean was 5.18 correct out of 7, or about 74% correct. The means (with standard deviations in parenthesis) are presented in Table 2.

The first hypothesis, that animation confers benefit and that benefit is equivalent to that of prediction (because the benefits of animation are based on prediction-testing), received some support. First, several participants in the animation/no-prediction condition verbally generated

predictions while viewing the animation even though they were not instructed to do so. Second, if performance of the no-animation/no-prediction group is compared to the averaged performance of the other three groups (M = 5.41), an overall difference is found, $F(1, 86) = 6.13$, $p = .015$, $MSE = 2.22$. Further, as Table 2 suggests, there were no meaningful performance differences among the first three groups in the table. All of this is consistent with the interpretation that the two interventions, animation and prediction, were of equivalent effectiveness.

As discussed earlier, this was not strictly a 2 x 2 design; however, it can be conceptualized that way. Treating animation/no-animation as one factor and prediction/no-prediction as a second factor, the 2 x 2 ANOVA shows a reliable main effect for animation, $F(1, 84) = 3.98$, $p = 0.049$, $MSE = 2.24$, though not for prediction, $F(1, 84) = 3.43$, $p = 0.068$. While only the animation effect is reliable at an a criterion of 0.05, the observed effect sizes in standard deviation units (Cohen, 1988) were, in fact, quite similar: the effect size for animation was 0.43 standard deviations and the effect size for prediction was 0.39 standard deviation units. This would be classified by Cohen as a medium to small effect size for both factors.

Our second hypothesis--that animation has benefits distinct from prediction--was not supported. The animation/prediction group did not outperform the no-animation/prediction group ($p = .27$; required $p = .017$ using Shaffer (1986) sequential Bonferroni pairwise comparisons for providing a familywise a of .05 for multiple comparisons; see also Seaman, Levin, & Serlin, 1991).

During the training phase, participants in the prediction conditions had to make a total of 42 predictions across two graphs. The average number of errors made by those who saw the animation was 0.46, while it was 1.19 for participants in the no-animation condition. This difference is statistically reliable ($F(1, 42) = 7.47$, $p < 0.01$) but, considering the low overall error rate, it is not particularly large in a practical sense. In addition, number of prediction errors made during training was not reliably correlated with post-test performance ($r(43) = 0.26$, $p = 0.096$).

In sum, the hypothesis with the strongest support is our first hypothesis, that the benefits conferred by animation center around encouraging learners to make and test predictions, as animation and prediction have indistinguishable effects on post-test performance in this study.

3.3 Discussion

Experiment 1 provides some support for the hypothesis that the ability to test predictions and get feedback is what confers any advantage associated with animation, since the participants who made predictions on paper did just as well as those who saw animations. However, there was some indirect evidence that those in the animation condition did acquire the knowledge faster, since they made fewer errors based on a fixed number of training trials. There was also some indirect evidence that animations may be motivating for learners. Participants who saw the animation without being required to make explicit predictions did about as well as those participants who saw the animation and were required to make predictions. Whether the former group did as well because they were making predictions on their own or whether it was some other factor is unclear.

Experiment 1 was limited in its ability to discriminate between the various possibilities for the results. First, the algorithm may have been too simple, limiting the amount of assistance that could be provided by either the animation or the prediction task. That is, the simplicity of the depth-first search algorithm involved may not play to the strengths of animation, such as the ability of the animations to make complex abstractions concrete. Second, the participants used in Experiment 1 were not the typical target population for algorithm animations as they were not computer science students. Perhaps computer science students would be more likely to reap the purported benefits of animations because they have more practice working with various sorts of algorithms and visualizations. On the other hand, there may be less of an advantage for animations for this population since computer science students may be better at spontaneously generating their own visualizations. Finally, the post-test examined only one aspect of the students' understanding of the algorithm: their ability to execute the algorithm. Typical college-level courses on algorithms require knowledge about more than just how to execute an algorithm; they require students to

demonstrate knowledge about the theoretical properties of the algorithm and the data structures used in the algorithm.

In order to address some of these concerns, we conducted a second experiment using a significantly more complex algorithm that may benefit from animation, advanced computer science students, and a more difficult post-test that included conceptual questions as well as procedural ones. It was hoped that these changes would allow us to detect possible advantages of animations in an ecologically valid situation.

4 Experiment 2

In Experiment 2 we wished to compare the results we found with relatively naive students and a simple algorithm to those based on a more sophisticated audience and a more complex algorithm. Thus, we tested upper-level computer science students on the use of a relatively advanced data structure, the binomial heap.

A second key extension from Experiment 1 to Experiment 2 was the structure of the post-test. In Experiment 1, even the "challenging" post-test questions directly related to the strengths of the animation and the prediction task, since the test questions were concerned exclusively with the execution of the algorithm. Animations and predictions are directly tied to the step-by-step execution of an algorithm, and thus may be particularly useful in helping students learn how the algorithm executes on the micro level. However, the analysis of algorithms, and thus typical computer science courses on algorithms, are also concerned with the macro-level behavior of the algorithms, such as running time or memory use. Using advanced computer science majors allowed us to test these more analytical aspects of algorithm understanding along with performance on the step to step execution. It is possible that instead of aiding performance on conceptual questions, the time spent viewing an animation or making explicit predictions, while perhaps helping learners to understand the execution of the algorithm, might take their focus off the conceptual aspects and hurt their performance on post-test questions about these aspects.

4.1 Binomial Heaps

Binomial heap data structures are used as an implementation for an abstract data type called a priority queue. Priority queues are utilized in numerous computer science algorithms. They operate on nodes with key values (for simplicity, we can safely assume the key is an integer). The most basic version of a priority queue involves two operations, insert and extract-minimum. Insert simply adds a new key-valued node to the priority queue, and extract-minimum removes and returns the node with least key value.

Note that a basic sorted list can be used to implement a priority queue. Under this scheme, the extract-minimum operation is fast and efficient because the smallest key is at the head of the list. The insert operation can be very inefficient, however, because the entire list may need to be examined to find the proper insertion point. In general, computer scientists seek data structures without operations that are proportional to the entire size of the data structure. Binomial heaps are overall more efficient than using a simple list in this way.

A binomial heap is a collection (forest) of binomial trees (see Figure 2). A binomial tree is a heap-ordered tree data structure with smaller key values toward the root. Binomial trees always have a size that is a power of two. Larger binomial trees are made by joining two smaller trees of the same size. To join two binomial trees, their root values are compared. The root with higher key value is then linked in as a new child of the root with lower value, thus preserving the heap property.

The insert operation on a binomial heap simply adds a new binomial tree of size 1. If a tree of size 1 already exists in the heap, the two are joined to make a tree of size 2. If there are now two trees each of size two, they are joined to create a tree of size four. This proceeds analogously until no more joining occurs.

On an extract-minimum operation, the roots of all the binomial trees in the heap are searched and the smallest is found. It is removed and all of its children are elevated to be trees in the binomial heap. This may introduce trees of like sizes, so joining may need to be done. Joining begins by checking for trees of size 1 and proceeds upwards. Both the joining and extract-

minimum operations may require examining the root of all the trees in the binomial heap. But because of the unique way that trees are combined into power of two sizes, only a logarithm of the total size of the heap number of steps is involved. For more details on binomial heaps and their operations, consult any comprehensive computer science algorithms text such as Cormen et al. (1990).

Insert Figure 2 about here

The animation of the binomial heap data structure and algorithm was built using the XTango algorithm animation system (Stasko, 1992) and utilizes the representation of a binomial heap common in algorithms textbooks. The animation uses three work areas, each of which is indicated by a horizontal line as illustrated in Figure 2. The top work area is where stable binomial heaps, that is, heaps after operations, are drawn. All the binomial trees comprising the binomial heap are drawn in increasing tree size from left to right. The lowest work area is where nodes from a new insert operation are initially placed and where "new" trees from an extract-minimum operation (i.e., the children of the root) are originally placed (see Figure 2b). The middle work area represents an intermediate state. Trees reside here in the midst of an insert or extract-minimum operation; trees of the same size are joined, linked, and made into larger trees (see Figure 2c). All of these linking operations are illustrated using smooth, gradual animations that allow the viewer to track the context (e.g., heap location, node position) of the operation. A label at the bottom of the animation window lists the name of the operation currently taking place.

As described above, the binomial heap algorithm that we examined involves two operations: insert and extract-minimum. During an insert operation, the new tree appears at the lowest work area (Figure 2b) and is then moved to the middle work area. Next, all of the binomial trees in the heap above are brought down to the middle work area and the appropriate union and link operations are illustrated (Figure 2c). Finally, the resulting trees are elevated to the top work area again (Figure 2d).

During an extract-minimum operation, the smallest key-valued node in the binomial heap (one of the roots at the top) is flashed in red and removed. All of the resulting trees (child subtrees of the root) are moved to the lowest work area. At this point, all the trees at the lower work area and at the top work area are moved into the middle work level. The trees are moved in increasing size so that the middle work area contains all the intermediate trees in growing size from left to right. The linking operations then commence and the resulting binomial trees move up to the top stable work area.

4.2 Method

Participants. The participants in Experiment 2 were 62 undergraduate computer science majors (primarily juniors and seniors) at the Georgia Institute of Technology who participated for extra credit in an upper-level computer science course. Participants were screened to ensure they had no previous experience with binomial heaps, though they were required to have some background in algorithm analysis. The participants were again high-ability students with a mean self-reported SAT-Quantitative score of 672 and SAT-Verbal score of 567.

Materials and Procedure. The procedure was similar to that in Experiment 1. Participants first filled out the consent form and background questionnaire then watched a videotape of an 11-minute lecture on binomial heaps. After watching the videotape, participants were given a 12-page text describing binomial heaps that was adapted from a chapter in Cormen et al. (1990) and included pictures of data structures similar to those that would be seen on the post-test. Participants in the no-animation/no-prediction condition were given 35 minutes to read the text, all others were given 20 minutes. Once again, the former group was given additional time with the text in order to roughly equate the amount of training time among conditions since the other conditions would be receiving additional materials. Those who received only 20 minutes with the text then either simply watched the algorithm animation (animation/no-prediction condition), made explicit predictions while watching the animation (animation/prediction condition), or made explicit predictions from printed graphs (no-animation/prediction condition). All participants then received the post-test, on which they had 25 minutes to work.[2]

Participants in the prediction condition had to predict the state of a binomial heap after a specified operation was performed. A sample of such task is shown in Appendix C. They made a total of 12 predictions during this phase. Participants were shown a binomial heap--either on the computer screen or on paper--and were asked to draw the resulting binomial heap after a particular insert or extract-min operation was completed. Those watching the animation were shown the correct answer by the animation after making each prediction, while those making predictions on paper were shown a static picture of the correct heap. Participants were not explicitly told to compare their answers to the feedback, but in practice, that is what occurred.

As in Experiment 1, there was a difference in the feedback in the animation and no-animation conditions. Participants in the animation condition watched the animation proceed after they drew their prediction of the resulting heap. Thus they were able to observe the intermediate states. Participants in the no-animation condition were simply shown a picture of the resulting heap after they made their prediction. Thus, they did not see intermediate states.[3]

The number of errors during prediction was again recorded for those participants who made explicit predictions, thus the range of errors could be from 0-12. The questions on the post-test could be divided into two categories, procedural and conceptual. Procedural questions focused on the execution of the algorithm and required participants to perform operations on binomial heaps. Most conceptual questions, on the other hand, required knowledge about the abstract properties of binomial heaps. Examples of each of these two types of questions are presented in Appendix D. Questions on the post-test were again scored stringently as being either completely correct or incorrect.

The apparatus was identical to what was used in Experiment 1.

Design. The independent variable was training condition. There were 15 participants in both the animation/prediction and animation/no-prediction conditions and 16 participants in both the no-animation/prediction and no-animation/no-prediction conditions. As in Experiment 1, the initial analyses were done with one-way analyses of variance using condition as the single independent variable with four levels.

4.3 Results

We expected to see differential performance among the various groups on the procedural questions. Mean number correct out of a possible seven are shown in Table 3. The relative ordering of the means mirrors the results of Experiment 1, though the differences are not as robust as the differences found in Experiment 1. This might be due partly to the relatively high variability in the conditions.

Insert Table 3 about here

We wanted to evaluate the same hypotheses examined for Experiment 1. The first hypothesis, that animation confers benefit and that benefit is equivalent to that of prediction, does have some statistical support. Again, several participants in the animation/no-prediction condition verbally generated predictions while viewing the animation even though they were not instructed to do so. Second, if performance of the no-animation/no-prediction group is compared to the averaged performance of the other three groups (M = 5.02), an overall difference is found, $F(1, 60) = 3.81$, $p = .055$, $MSE = 3.25$. Further, as Table 3 suggests, there were no meaningful performance differences among the first three groups in the table. All of this suggests that, as in Experiment 1, the two interventions, animation and prediction, were of equivalent effectiveness.

As with Experiment 1, this was also not strictly a 2 x 2 design; however, it can be conceptualized that way. Animation/no-animation can be treated as one factor and prediction/no-prediction as a second factor. As in Experiment 1, the observed effect sizes were very similar: 0.33 standard deviations for animation and 0.30 standard deviations for prediction. The 2 x 2 ANOVA reveals that neither of these effects are reliable, animation: $F(1, 58) = 1.65$, $p = 0.204$, $MSE = 3.36$, and prediction: $F(1, 58) = 1.32$, $p = 0.255$. This again suggests that there is no difference in overall impact between animation and prediction, and both of these effect sizes would be classified by Cohen (1988) as small effects.

As in Experiment 1, the second hypothesis, that animation has benefits distinct from prediction, was not supported by the data. The animation/prediction group did not outperform the no-animation/prediction group ($p = .77$; required $p = .017$).

In contrast to one of the results found in Experiment 1, there was no advantage on the accuracy of predictions during training for those who saw the animation compared to those who saw the graphs on paper. Participants who saw the animation averaged 2.77 errors (out of a possible 12) during training and those who worked from the static images averaged 2.54 errors. Also unlike Experiment 1, the number of errors in the prediction phase was correlated with performance on the procedural post-test questions, $r(30) = 0.70$, $p < 0.01$. That is, students who made fewer prediction errors also made fewer post-test errors.

Insert Table 4 about here

No differences were found among any of the groups on the conceptual post-test questions. Mean number correct out of a possible 17 conceptual questions are shown in Table 4, with standard deviations in parentheses. Performance on the conceptual questions was correlated with performance on the procedural questions, $r(61) = 0.51$, $p < 0.01$.

4.4 Discussion

While Experiment 2 provided some evidence for benefits of animation and prediction similar to those found in Experiment 1, this evidence is weakened by the generally high variability in performance. It may be the case that the animation and predictions were useful only to those participants who already had some above-threshold understanding of the algorithm by the time they reached that phase of the experiment. The animations and predictions may simply not have been comprehensible to those participants who were still struggling with the basics of the algorithm. Some aspects of the data suggest this, as one of the more common errors found in the answers to the procedural questions among the weak performers was the creation of data structures that did

not even meet the definition of a binomial heap. This suggests that the students did not fully understand the foundational material presented in the videotape and the text. This claim is further supported by the fact that performance on the prediction task was strongly correlated with performance on the procedural part of the post-test. That is, participants who made a lot of errors in the prediction task did not seem to learn enough from this experience to improve their later performance. In general, good performers did well in every phase of evaluation and poor performers did poorly on all phases. Thus, it is not clear that viewing the animation or making predictions provides any benefit if the learner has not acquired some fundamentals.[4]

5 General Discussion

The results from the two experiments show a trend towards a benefit of animations and predictions on students' ability to solve procedural problems about algorithms. It seems clear, however, that the sheer use of algorithm animation does not automatically and significantly enhance learning as some instructors and researchers hope--there are a host of unresolved issues to consider.

While we have discussed the present results in terms of how animations may lead learners to produce predictions and that predictions are actually what aided (procedural) learning, it is also possible that the mere presence of “good” visual representations of the algorithms is what really mattered. That is, the two animation groups and the no-animation/prediction group all saw a similar, enhanced graphic depiction of the algorithm beyond that of the figures accompanying the text. The no-animation/no-prediction group only saw the figures included in the textbook materials, which, while not too different from those used in the animations or static images, were perhaps not as compelling. Thus, to really test the claim that prediction affected our results, a future experiment would need to include a condition in which learners see the static graphical material used in the no-animation/prediction condition, but are not asked explicitly to make predictions. If this group were to do as well as the animation groups and the no-animation/prediction group, this would suggest that the graphical materials themselves were the

source of improved learning. If, however, this group were not to do as well as the three mentioned above, this would allow us to even more confidently conclude that making predictions is what primarily aided performance in the present study.

One possible explanation for the lack of a strong animation effect in the present study is that prior knowledge of learners, particularly in Experiment 2, might have interacted with the animations and predictions in unanticipated ways. That is, it is not obvious how the information conveyed in the animation or provided by making predictions interacts with the knowledge that the learners bring to bear when they initially engage in these activities. Animations and predictions with complex algorithms may help only some learners, those with just exactly the right amount of knowledge to make use of the information provided, but not so much so that the information provided is redundant with what they already know.

Another interesting issue is that animations might not aid overall learning or performance, but may aid how *quickly* a student learns. We made sure that students in the different training conditions spent roughly the same amount of time with the training materials, but perhaps that control wiped out the benefit that an animation might provide. A related issue is whether animations play a motivational role for learners. There is some evidence from observational research that many students prefer to make use of animations in the learning process (Kehoe & Stasko, 1996). An idea to consider for future research is to allow learners in the different conditions to spend as little or as much time with the different materials as they wish and then to examine performance.

In this same vein, we need to consider what sorts of tests best assess the possible benefits of animations. For instance, tasks that require students to make rapid responses might demonstrate that those who studied animations are able to run their mental models of the algorithm more quickly than students who did not see the animations.

There are, as yet, no clear guidelines for the construction of algorithm animations, not just from a psychological perspective but from an implementational one. What visual elements should be included? How should they look? What should the flow and sequencing be? These sorts of

questions can motivate additional systematic studies into the potential benefits of animations on algorithm learning and understanding.

Two *existing* animations were chosen for use in the present study. The construction of the animations was guided by intuition and thus, broad claims about features of animations that aid learning should not be drawn from the present results. When instructors' preparation time is so critical, the use of existing animations would seem to be the first option. However, if, through theory-guided systematic manipulations of animation features, we can show benefits of animations, we can also attempt to determine as precisely as possible which features actively aided learning. From such results we can begin to develop useful guidelines about how to construct and use animations to aid algorithm learning.

An animation might help learning because it displays the features that one should presumably attend to. That is, the items in the animation might, by their presence, provide information to the learner that might not be easily discernible otherwise. An animation might also encourage the learner to self-explain the behavior of the algorithm. Self-explanation can increase the likelihood of a learner integrating the new information into existing knowledge structures, thus making the learner more likely to transfer the information to novel situations (Catrambone, 1996; Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Chi, De Leeuw, Mei-Hung, & Levancher, 1994).

Many animation design issues persist: Should an animation reflect a “novice” or “expert” point of view for how the algorithm operates? Should learners be allowed to choose the inputs into the algorithm? More generally, what is the effect on learning of interactivity of animations? Should the animation be veridical with respect to the algorithm or should it gloss over certain details in order to make the big picture clear? Should the animation be relatively unadorned or should it be annotated with additional features such as pseudocode? Will such annotation require “split-attention” and what effects will that have on learning (Ward & Sweller, 1989)?

The various issues raised above suggest that additional research on algorithm animation can benefit from a careful functional analysis. That is, one must first decide what it is a student should learn about an algorithm and then consider how this information could potentially be conveyed

(text, static images, animations, etc). Alternative presentations of the information can then be systematically compared. Constructing algorithm animations generally requires serious programming effort and it still has to be demonstrated that the benefits justify this cost. If the same pedagogical advantages can be realized with less labor-intensive materials and analyses, then the less labor-intensive methods make more sense. It is incumbent upon educators and animation-builders to carefully examine their assumptions about what students will learn from an animation, and why it is that an animation is best-suited to convey the desired information.

References

Anderson, J.R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Bazik, J., Tamassia, R., Reiss, S.P., & van Dam, A. (1998). Software Visualization in Teaching at Brown University. In J. Stasko, J. Domingue, M.H. Brown, & B.A. Price (Eds.), *Software visualization: Programming as a multimedia experience*. Cambridge, MA: MIT Press, 383-398.

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

Baecker, R., & Small, I. (1990). Animation at the interface. In B. Laurel (Ed.), *The art of human-computer interface design*. Reading, MA: Addison-Wesley, 251-267.

Brown, M.H. (1988a). Perspectives on algorithm animation. *Proceedings of the ACM SIGCHI '88 Conference on Human Factors in Computing Systems*, 33-38.

Brown, M.H. (1988b). Exploring algorithms using Balsa-II. *Computer, 21*(5), 14-36.

Brown, M.H. (1991). ZEUS: A system for algorithm animation and multi-view editing. *Proceedings of the 1991 IEEE Workshop on Visual Languages*, 4-9.

Brown, M.H., & Najork, M.A. (1997). Collaborative active textbooks. *Journal of Visual Languages and Computing, 8*(4), 453-486.

Catrambone, R. (1996). Generalizing solution procedures learned from examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*(4), 1020-1031.

Chi, M.T.H., Bassok, M., Lewis, R., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13,* 145-182.

Chi, M.T.H., De Leeuw, N., Mei-Hung, C., & Levancher, C. (1994). Eliciting self explanations. *Cognitive Science, 18,* 439-477.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences. Hillsdale, NJ: Erlbaum.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms*. Boston, MA: MIT Press.

Douglas, S., Hundhausen, C., & McKeown, D. (1996). Exploring Human Visualization of Algorithms. *Proceedings of Graphics Interface '96*, 9-16.

Faraday, P., & Sutcliffe, A. (1997). Designing Effective Multimedia Presentations. *Proceedings of the ACM SIGCHI 1997 Conference on Human Factors in Computing Systems*, 272-278.

Gloor, P.A. (1992). AACE--Algorithm animation for computer science education. *Proceedings of the 1992 IEEE Workshop on Visual Languages*, 25-31.

Gurka J., & Citrin, W. (1996). Testing Effectiveness of Algorithm Animation. *Proceedings of the 1996 IEEE International Symposium on Visual Languages*, 182-189.

Hansen, S., Schrimpser, D., & Narayanan, N.H. (1998). Learning algorithms by visualization: A novel approach using animation-embedded hypermedia. *Proceedings of the 1998 International Conference of the Learning Sciences*, 125-130.

Hegarty, M., & Just, M.A. (1993). Constructing mental models of machines from text and diagrams. *Journal of Memory and Language, 32,* 717-742.

Jarc, D. (1999). *Assessing the benefits of interactivity and the influence of learning styles on the effectiveness of algorithm animation using web-based data structures courseware*. Unpublished doctoral dissertation, The George Washington University, Washington, D.C.

Kehoe, C. M., & Stasko, J. T. (1996). *Using animations to learn about algorithms: An ethnographic case study*. Technical Report GIT-GVU-96-20, GVU Center, Georgia Institute of Technology.

Kieras, D.E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science, 8,* 255-273.

Lawrence A. (1993). *Empirical studies of the value of algorithm animation in algorithm understanding*. Unpublished doctoral dissertation, Georgia Institute of Technology, Atlanta, GA.

Lawrence, A.W., Badre, A.M., & Stasko, J.T. (1994). Empirically evaluating the use of animations to teach algorithms. *Proceedings of the 1994 IEEE Symposium on Visual Languages*, 48-54.

Mayer, R.E., & Anderson, R.B. (1991). Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of Educational Psychology*, 83(4), 484-490.

Mayer, R.E., & Anderson, R.B. (1992). The instructive animation: Helping students build connections between words and pictures in multimedia learning. *Journal of Educational Psychology, 84*(4), 444-452.

Mayer R.E., & Sims, V.K. (1994). For whom is a picture worth a thousand words? Extensions of a dual-coding theory of multimedia learning. *Journal of Educational Psychology, 86*, 389-401.

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

Naps, T. (1990). Algorithm visualization in computer science laboratories. *Proceedings of the 21st SIGCSE Technical Symposium on Computer Science Education*, 105-110.

Paivio, A. (1990). *Mental representations: A dual coding approach*. New York: Oxford University Press.

Palmiter, S. (1993). The effectiveness of animated demonstrations for computer-based tasks: A summary, model, and future research. *Journal of Visual Languages and Computing, 4*(1), 71-89.

Palmiter, S., & Elkerton, J. (1993). Animated demonstrations for learning procedural computer tasks. *Human-Computer Interaction, 8*, 193-216.

Pane, J.F., Corbett, A.T., & John, B.E. (1996). Assessing dynamics in computer-based instruction. *Proceedings of the 1996 ACM SIGCHI Conference on Human Factors in Computing Systems*, 197-204.

Park, O., & Gittelman, S.S. (1992). Selective use of animation and feedback in computer-based instruction. *Educational Technology Research & Development, 40*(4), 27-38.

Price, B.A., Baecker, R.M., & Small, I.S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages and Computing, 4*(3), 211-266.

Rieber, L.P. (1996). Animation as feedback in a computer-based simulation: Representation matters. *Educational Technology Research & Development, 44*(1), 5-22.

Rieber, L.P. (1991). Animation in computer-based instruction. *Educational Technology Research & Development, 38*(1), 77-86.

Rieber, L.P. (1990). Using computer animated graphics in science instruction with children. *Journal of Educational Psychology, 82,* 135-140.

Rieber, L.P., Boyce, M.J., & Assad, C. (1990). The effects of computer animation on adult learning and retrieval tasks. *Journal of Computer-Based Instruction, 17*(2), 46-52.

Robertson, G.G., Card, S.K., & Mackinlay, J.D. (1993). Information visualization using 3D interactive animation. *Communications of the ACM, 36* (4), 57-71.

Roman G.C., Cox K., Wilcox C., & Plun, J. (1992). Pavane: A system for declarative visualization of concurrent computations. *Journal of Visual Languages and Computing, 3*(1), 161-193.

Seaman, M.A., Levin, J.R., & Serlin, R.C. (1991). New developments in pairwise multiple comparisons: Some powerful and practical procedures. *Psychological Bulletin, 110*(3), 577-586.

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

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

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

Stasko, J.T., Badre, A., & Lewis, C. (1993). Do algorithm animations assist learning? An empirical study and analysis. *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems,* 61-66.

Stasko, J.T., & Kraemer, E. (1993). A methodology for building application-specific visualizations of parallel programs. *Journal of Parallel and Distributed Computing, 18*(2), 258-264.

Stasko, J., Domingue, J., Brown, M.H., & Price, B.A. (Eds.). (1998). *Software visualization: Programming as a multimedia experience*. Cambridge, MA: MIT Press.

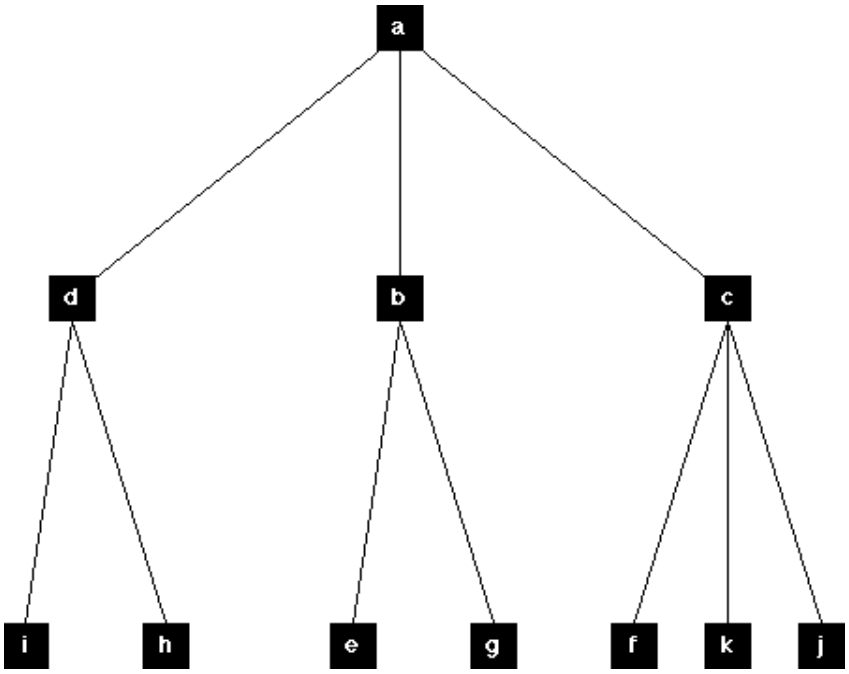
Sukaviriya, P. (1990). Coupling a UI framework with automatic generation of context-sensitive animated help. *Proceedings of the '90 ACM SIGGRAPH Symposium on User Interface Software and Technology*, 152-166.

Ward, M., & Sweller, J. (1989). Structuring effective worked examples. *Cognition and Instruction, 7*(1), 1-39.

Appendix A

Sample of Depth-First Search Prediction Task Materials for No-Animation Condition (Experiment 1)

(Excerpt from instructions read to the participant): You will see a graph and I will ask you at each stage where the depth first search will go next. Take your time and try to answer the questions correctly.

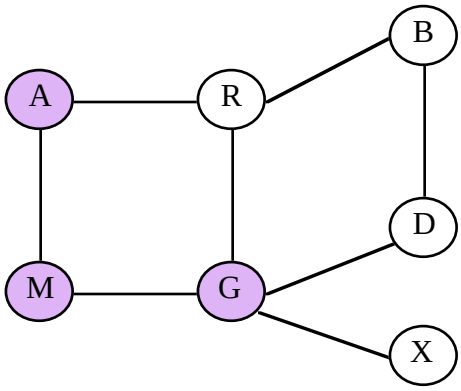


Appendix B

Sample of Post-Test Questions for Depth-First Search (Experiment 1)

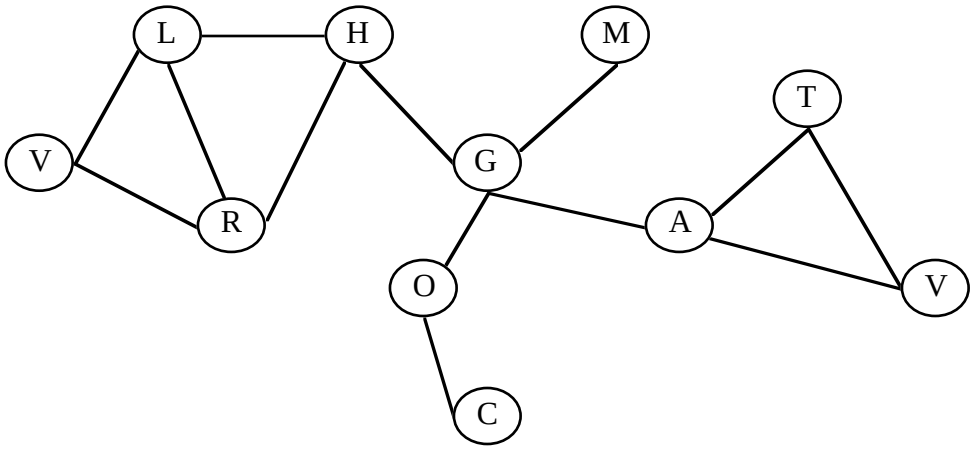
"Basic" Question

In the following graph, assume that all darkened vertices have already been visited in a depth first search. Also assume that we have just visited vertex G. What would be the next vertex that we visit?



"Challenging" Question

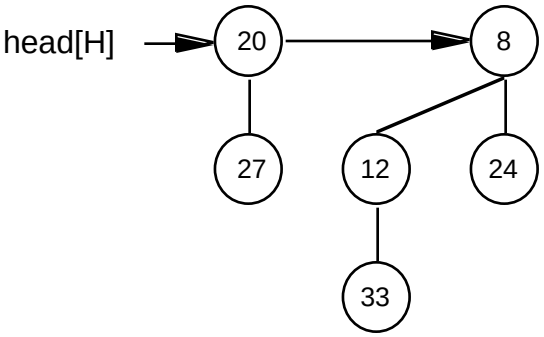
Starting with vertex A, list the order of vertices as they are encountered in a depth first search of the graph below. Also list the order in which vertices are first encountered.



Appendix C

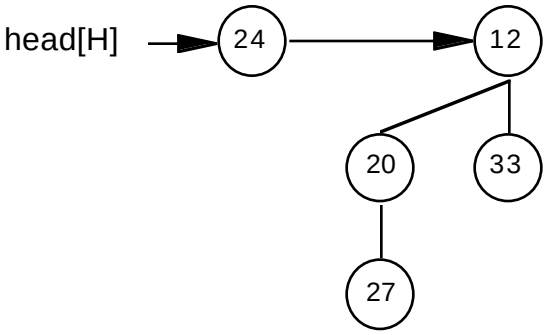
Sample of Binomial Heap Prediction Task Materials for No-Animation Condition (Experiment 2)

Draw the results of doing an EXTRACT-MIN on the binomial heap below.



(the text and figure below would appear on the page following the above text and figure)

The correct answer to the previous problem is the binomial heap shown below.

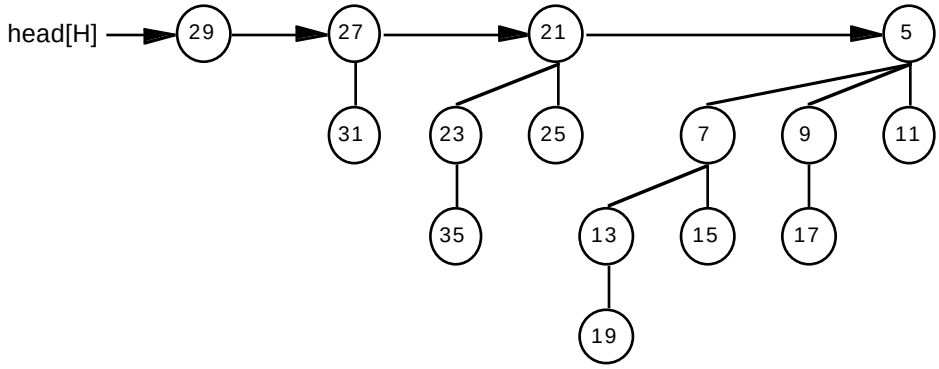


Appendix D

Sample of Post-Test Questions for Binomial Heap (Experiment 2)

Procedural Question

Suppose “33” were inserted into the heap pictured below. Please draw the result.



Conceptual Question

What is the worst case running time for INSERT on a binomial heap?

Author Notes

This research was supported by a grant from the Graphics, Visualization, and Usability (GVU) Laboratory of the Georgia Institute of Technology. Parts of Experiments 1 and 2 were reported at the 7th Annual Winter Text Conference, Jackson Hole, January 1996 and at the Basic Research Symposium at the CHI '96 Conference, Vancouver, April 1996.

Footnotes

[1]The order of vertices visited in Figure 1 (including all backtracking) are: a, b, a, c, g, m, g, n, t, n, u, n, g, c, h, c, a, d, i, d, j, o, v, o, w, o, j, p, j, d, a, e, a, f, k, q, r, x, r, y, r, z, r, k, f, l, s, l, f, a

[2]Note that in Experiment 1 participants had an unlimited amount of time to work on the post-test. However, given the relatively long time we felt participants would require to do Experiment 2, we decided to limit how much time could be spent on the post-test.

[3]Would viewing the intermediate states (either dynamically or through a set of static images) make a difference in performance? This question can not be answered based on the current experiment, however, it can be examined in a future study.

[4]To test the possibility that learners are not much affected by the animation and prediction manipulations once the basic information is presented via a lecture or textbook, a future experiment could replicate Experiment 2, but include assessments after both the lecture/text portion of the experiment and after the animation/prediction portion. The changes, if any, could be compared. This within-subject approach might be more sensitive to the effects of animation/prediction and could shed light on the conjecture that animation/prediction aids learning only if a basic foundation has been previously acquired (e.g., from the lecture/text). One methodological concern of such an approach is that the inclusion of the first post-test might cue learners in the animation/no-prediction condition to make predictions more frequently than they might otherwise.

Table 1

Post-test Means for Basic Problems (Experiment 1)

Condition	
Animation/Prediction	7.59 (0.59)
Animation/No-Prediction	7.45 (0.86)
No-Animation/Prediction	7.63 (0.58)
No-Animation/No-Prediction	7.63 (0.58)

Note: Maximum possible score was 8. Standard deviations are in parentheses.

Table 2

Post-test Means for Challenging Problems (Experiment 1)

Condition	
Animation/Prediction	5.73 (1.20)
Animation/No-Prediction	5.27 (1.72)
No-Animation/Prediction	5.24 (1.64)
No-Animation/No-Prediction	4.50 (1.40)

Note: Maximum possible score was 7. Standard deviations are in parentheses.

Table 3

Post-test Means for Procedural Questions (Experiment 2)

Condition	
Animation/Prediction	5.13 (1.30)
Animation/No-Prediction	5.00 (1.60)
No-Animation/Prediction	4.94 (2.05)
No-Animation/No-Prediction	4.00 (2.19)

Note: Maximum possible score was 7. Standard deviations are in parentheses.

Table 4

Post-test Means for Conceptual Questions (Experiment 2)

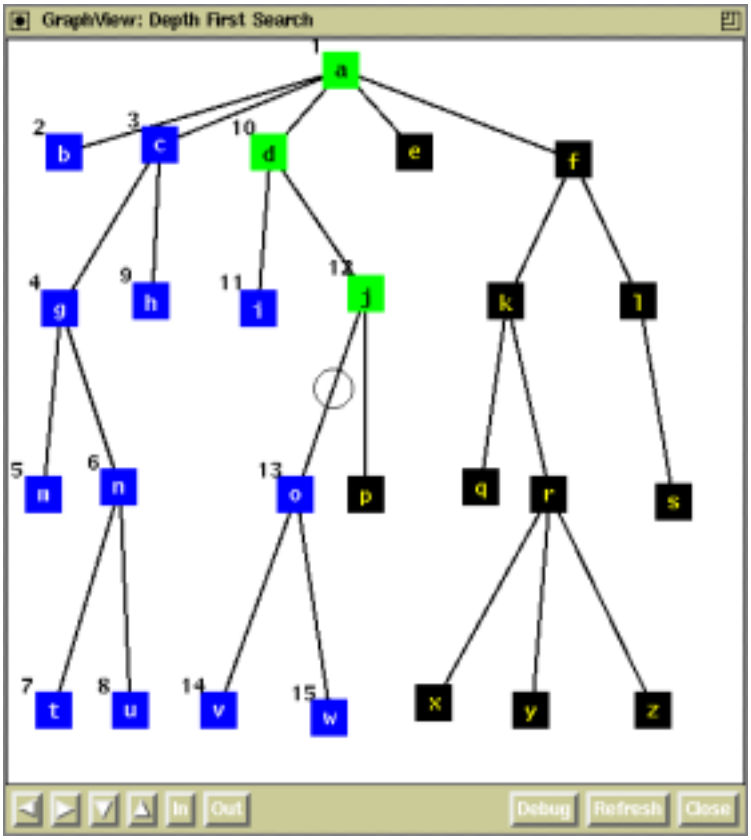
Condition	
Animation/Prediction	9.93 (3.17)
Animation/No-Prediction	10.80 (3.08)
No-Animation/Prediction	10.94 (3.11)
No-Animation/No-Prediction	9.94 (3.11)

Note: Maximum possible score was 17. Standard deviations are in parentheses.

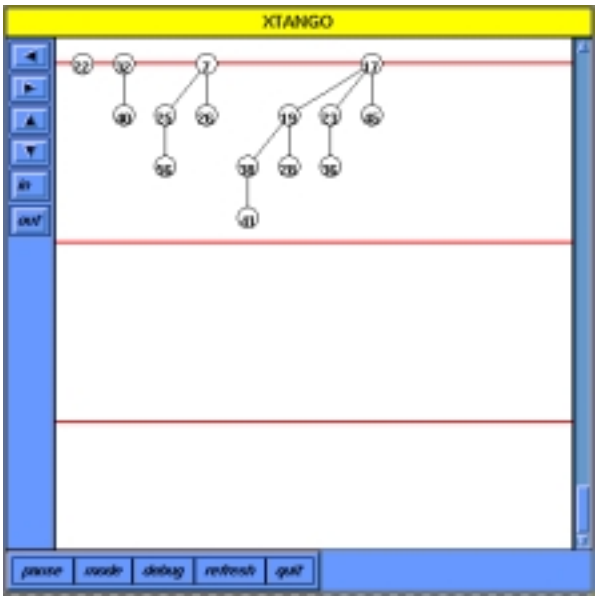
Figure Captions

Figure 1. Screen Shot of Animation for Depth First Search (DFS).

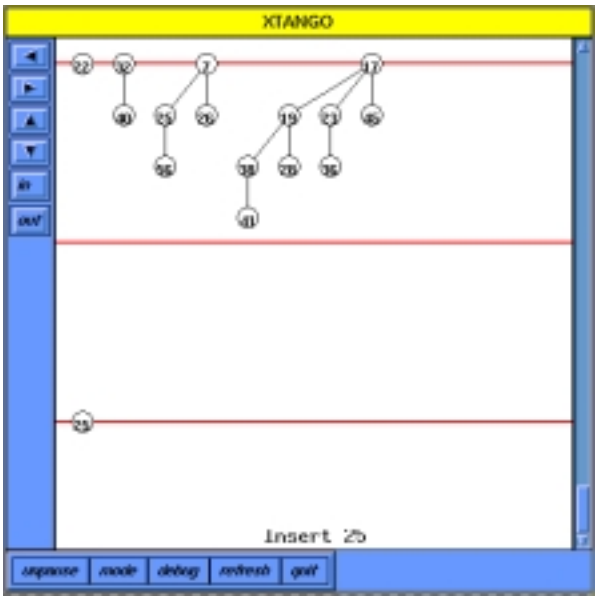
Figure 2. Screen Shots of Animation for Binomial Heap (BH).


GraphView: Depth First Search
1
a
2
b
3
c
10
d
e
f
4
g
9
h
11
i
12
j
k
l
5
m
6
n
13
o
p
q
r
s
7
t
8
u
14
v
15
w
x
y
z
In
Out
Debug
Refresh
Close

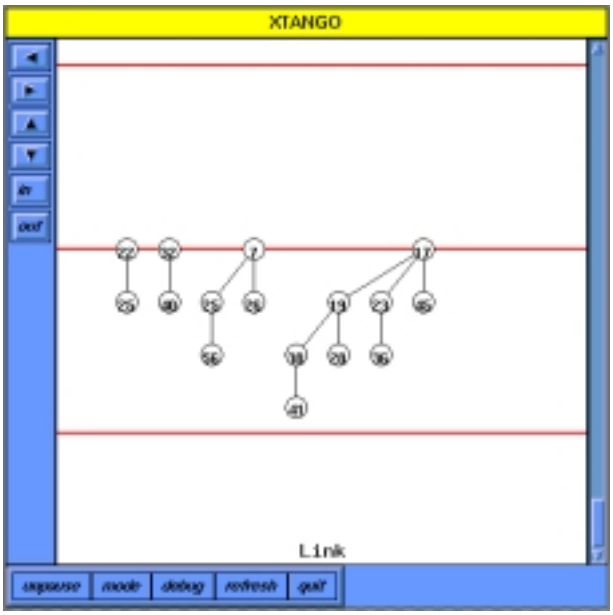
a


XTANGO
pause
mode
debug
refresh
quit

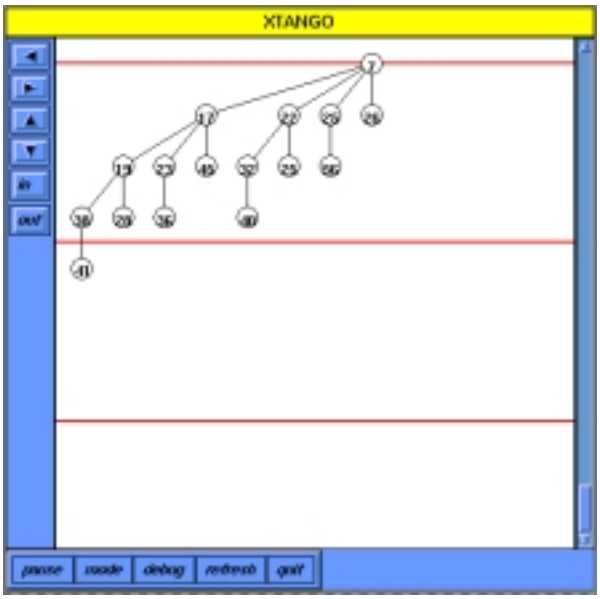
b


XTANGO
Insert 25
unpause
mode
debug
refresh
quit

c


XTANGO
Link
unpause
mode
debug
refresh
quit

d


XTANGO
pause
mode
debug
refresh
quit

Biographies

Michael Byrne is an assistant professor in the Department of Psychology at Rice University. He received his Ph.D. in experimental psychology in 1996 from the Georgia Institute of Technology.

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

John Stasko is an associate professor in the College of Computing at the Georgia Institute of Technology. He received his Ph.D. in computer science in 1989 from Brown University.