A Empirical Study of Web-based Algorithm Animation Courseware in an Ada Data Structure Course

Duane J. Jarc
The George Washington University
Department of Electrical Engineering
and Computer Science
(202) 994-6083
jarc@seas.gwu.edu

Michael B. Feldman
The George Washington University
Department of Electrical Engineering
and Computer Science
(202) 994-5919
mfeldman@seas.gwu.edu

1. ABSTRACT

In this paper, we describe Web-based courseware, which we developed, that contains exercises that use algorithm animation and visualization. It is intended for students in an introductory data structure course in Ada. We also discuss the results of an empirical study that we conducted using this courseware. In the study, we measured whether the interactive feature included in the courseware influenced student performance and whether student learning styles were a factor. No significant differences were found. A redesigned, more tightly controlled experiment is planned to more definitively answer this question.

1.1 Keywords

Ada data structure course, algorithm animation, empirical study, interactive multimedia, learning styles, Web-based courseware

2. INTRODUCTION

Computer-assisted instruction (CAI) has been used for several decades, but the development of the new multimedia technology and the possibility of creating courseware that can be accessed by students worldwide by hosting it on the World Wide Web, presents exciting new opportunities for educators. Before investing considerable time and effort building such multimedia courseware, it is important to determine the effectiveness of such materials and how they can be designed and used most effectively.

The animation of algorithms is one of the most obvious applications of this technology. Intuitively most computer science educators believe that algorithm animation should be helpful in teaching computer science concepts. The research that has attempted to determine whether it improves student learning has surprisingly been rather mixed. Some studies [6] have shown that student involvement improves the benefit of animation. One goal of our study was to determine whether providing a high level of interactivity improves student performance.

A second goal of our study was to determine whether there is a difference in the benefit provided by our courseware between students whose learning style is active compared to those whose style is reflective as measured by the Felder-Silverman learning style inventory [4].

In our study, we are using courseware, which we developed, that is hosted on the World Wide Web. This courseware incorporates algorithm animation and data structure visualizations in a highly interactive multimedia environment. It is designed as laboratory exercises to be used during a four week period of a typical second semester computer science course in elementary data structures taught using Ada 95.

It is our hope that this study adds to the important field of investigation that helps us understand how to effectively incorporate the rapidly changing technology into the field of education—computer science education, in particular.

3. RELATED WORK

First, let us survey the research that has been done to assess whether algorithm animation is really an effective tool in computer science education.

3.1 Empirical Studies of the Effectiveness of Algorithm Animation

Perhaps the earliest study by Badre et al. [1] was an exploratory study that began with a faculty survey to determine the methods most faculty members used to teach algorithms and the type of graphic diagrams they used. They observed several students viewing an animation of the Shell sort algorithm with the intent of gaining insight in how to conduct more formal studies. We find it interesting that among their conclusions they suggest that it is important to consider the spatial abilities of students. They conjecture that poor visualizers might benefit most from animation systems. Despite this observation, we find no subsequent study that has considered this factor.

The first truly empirical study of algorithm animation was conducted by Stasko, Badre and Lewis [9]. Their study used

a priority queue implemented with a pairing heap and involved two groups of students. One group was given a textual description of the algorithm. The second group was given text and the animation. Although their overall results did not suggest that animation helped significantly, they noted that the animation seemed to help more on the declarative and analytical questions, but not on the procedural questions.

In a subsequent study [6] by Lawrence, Badre, and Stasko, a minimum spanning tree algorithm was used. They found that viewing animations in either a classroom or laboratory setting did not significantly improve students' understanding. They did find, however, that there was a significant difference depending upon whether students were an active participant in the laboratory or not. Those participating in an active laboratory setting performed significantly better than those who participated in a passive laboratory. In addition, their results indicated that students who attended a laboratory session involving animation performed significantly better on the free-response test than on the questions that measured factual knowledge.

The animation of sorting and searching algorithms was the subject of a study by Crosby and Stelovsky [3]. They divided questions into two categories that they referred to as textual or graphical. They concluded that the graphical questions better match multimedia instruction than do textual ones. In addition, they divided students into two groups according to their cognitive learning style as measured by the Myers-Briggs Type Indicator. Although this indicator contains four orthogonal dimensions, they considered only one of the four—the one that separates students into sensing or intuitive. Their results indicated that students categorized as sensing were helped significantly by the animation, while the other group of students did not gain nearly the same level of benefit from the animation.

Considering these studies together, we can conclude that it appears that algorithm animation may influence learning certain kinds of knowledge more greatly than learning other kinds or may help certain kinds of students more than others, but given the limited evidence these conclusions are at best tentative. Most of the studies show that algorithm animation has no significant effect when type of knowledge and type of learner are undifferentiated.

3.2 The World Wide Web

Next, we consider the research that has explored staging courseware on the World Wide Web. While the incorporation of multimedia into CAI has the potential to improve the quality of the presentation, staging CAI on the WWW greatly facilitates its distribution. We found no empirical studies that have evaluated the effectiveness of Web-based CAI, although a considerable amount of courseware is currently being developed for the Web.

Like multimedia, the Web has evolved in incremental steps. In its initial stages, only documents containing text and graphics could be hosted on Web sites, using hypertext mark-up language (HTML). The interactivity of such documents was limited to the hypertext links. The creation of dynamic documents required common gateway interface (CGI) programs that reside on the server side. In such a configuration, each interaction requires that a message be sent from the client to the server, the server must then dynamically generate a page that is returned to the client. Clearly, such communication adversely affects the response time. Some computer science educators, such as Marshall and Hurley [7], have noted that using Java can greatly improve the response time of such systems, because the need for client-server communication is eliminated. In their discussion of the use and design of visualization, the working group on visualization at the recent conference on integrating technology into education [8] notes that Java's abstract windowing toolkit is one of its important advantages.

Clearly, the ability to develop CAI that can be hosted on the World Wide Web offers considerable opportunities for educators in general, and computer science educators in particular. The potential of such CAI provides clear reason for its development and evaluation.

4. THE WEB-BASED COURSEWARE

The *Interactive Data Structure Visualization* courseware that we used in this study consists of a collection of web pages, with a Java applet that implements the visualizations. This courseware can be found at: `www.seas.gwu.edu/seas/eecs/research/visualization`

The term visualization is used rather than animation because it includes both animations and nonanimated but highly visual exercises. One example is an exercise designed to assist students in identifying different kinds of binary trees, such as complete trees, binary search trees, heaps and so on. This exercise contains no animation, but is highly visual, nonetheless. For each of the exercises there is a page of explanatory text, a page containing the Ada 95 code associated with that exercise, and the applet containing the visualization.

The adjective *interactive* is included in the title of our courseware because all exercises allow students to be shown solutions with its *Show Me* mode and to interact with the system and try to replicate the steps of an algorithm in the *I'll Try* mode. In a recent study by Byrne et al. [2] the value of students predicting the next step of an algorithm was shown to be of some benefit. Rather than stopping an animation and guessing the next step, our *I'll Try* mode allows students to enter a purely interactive mode. As an example, in the binary tree traversal exercise, they click appropriate tree nodes to indicate the next step of the traversal. A pair of radio buttons allows students to switch between modes at any time.

The subject areas included in our courseware are binary trees, graphs and sorting. Eleven exercises are included in the courseware. They are intended to be used during a four week period.

4.1 Week 1—Undirected Graphs

There are three exercises on undirected graphs, two animations and one visualization. The first exercise is an animation illustrating the adjacency representation of graphs. A graph with exactly twelve nodes is used. Both the graph and the matrix are displayed.

The second exercise is an animation illustrating depth-first and breadth-first graph searches. Graph searches are similar to binary tree traversals, but there are some important differences. Because graphs can contain cycles, it is necessary to mark each node as it is visited to avoid endless loops. We illustrate this marking with a black X. The applet window for this exercise is shown in Figure 1.

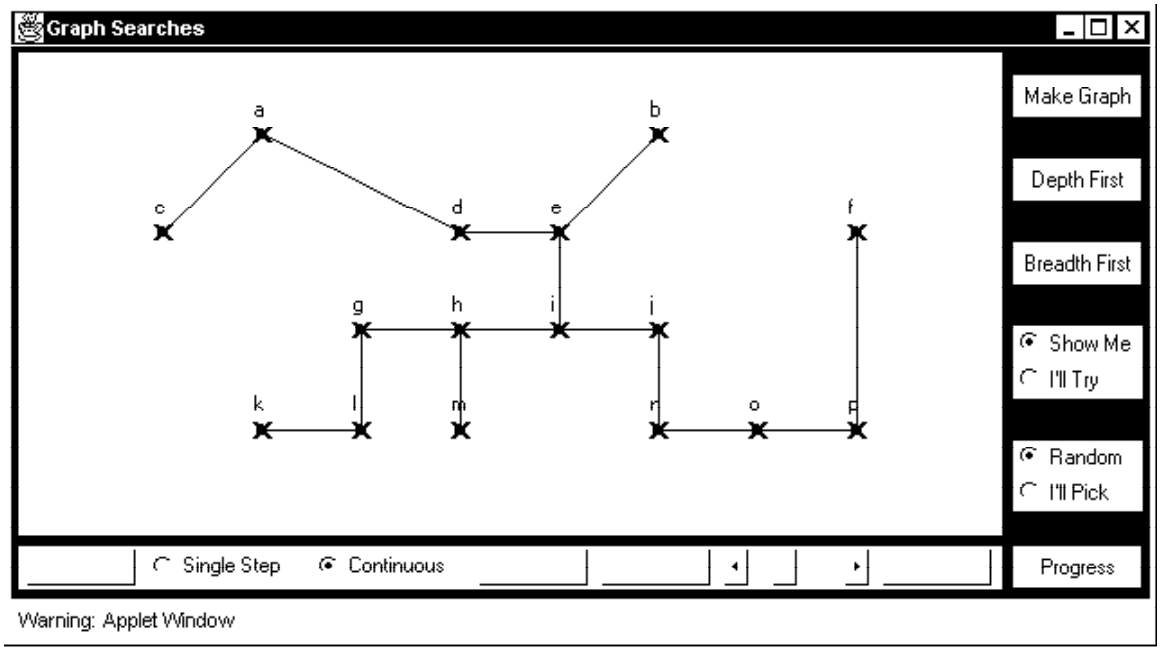

Figure 1 Graph search exercise

The last of the three exercises is a visualization that illustrates three graph properties: whether it is connected, whether it contains cycles and whether it is an undirected tree. Disconnected graphs are indicated by highlighting all nodes that are disconnected to one designated node. Graphs that contain cycles are indicated by highlighting the edges of the cycles in white. Graphs that are not undirected trees are indicated by showing their disconnectedness or cycles as appropriate.

4.2 Week 2—Binary Trees

Binary trees are a visually rich topic. This topic spanned two of the four weeks of our study. We chose two of the binary tree exercises for the second week.

The first exercise illustrates four binary tree traversals. Trees of random shapes containing random values are generated. The traversals include preorder, inorder, postorder, and level order. The traversals are animated by a circle that traces the path of the traversal. When the data value of a node is extracted, its value moves to a list of values at the bottom of the screen. All the recursive traversals follow the same path, so understanding the difference among the three requires understanding that in a recursive traversal each node is visited three times, and in a preorder traversal that value is extracted the first time, for inorder the second, and postorder the third. The applet window for the binary tree traversal exercise, illustrating the preorder traversal, is shown in Figure 2.

The second exercise is an animation of a binary search tree. The operations provided are the most basic ones—insert, delete and find. To allow the student to experiment with various cases, the option to have data values randomly generated or selected by the student is provided for all operations.

4.3 Week 3—More Binary Trees

The first exercise in the third week illustrates a priority queue using a heap. The two operations that are provided, are enqueue and dequeue. The sifting process up or down the tree is illustrated by the swapping of values in both representations when a dequeue or enqueue operation is performed. Random values are chosen, but the option is provided for the student to pick a value for the enqueue operation.

The remaining two binary tree exercises are visualizations rather than animations. The first of these two is designed to assist students in understanding the concepts of tree height or depth, and the concept of balance factors and height-balanced trees—a topic often introduced, but not fully explored in an elementary data structure course. The final binary tree exercise, which is the second of the two visualizations, involves categorizing binary trees. There is

a button to test whether the tree on the screen qualifies for the following binary tree categories: almost complete, complete, binary search tree, heap and AVL tree.

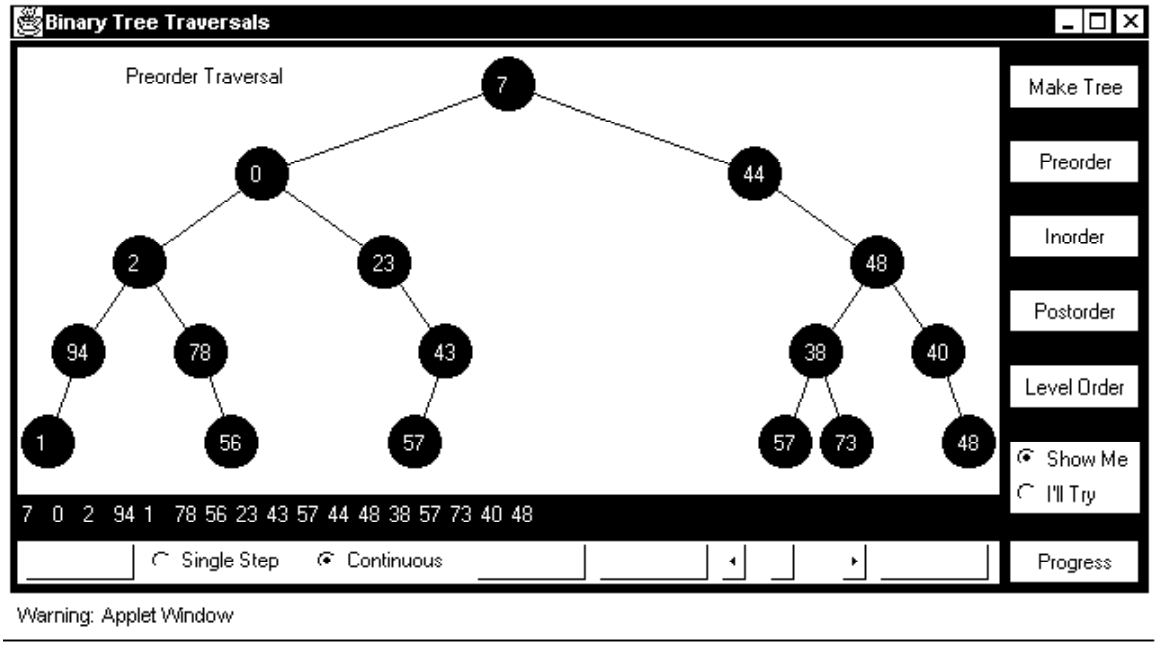

Figure 2 Binary tree traversal exercise

4.4 Week 4—Sorting

Finally there are three animated exercises on sorting. The first animation is the heap sort, which we choose to include among the sorting exercises, although conceptually it fits equally well among the binary tree exercises. As with the heap used for the priority queue exercise, both the tree and array representations are illustrated.

The second exercise is an animation of three quadratic sorts: the insertion sort, the selection sort and the bubble sort. We animate the movement of both the bars and their corresponding numbers to emphasize their connection. Color is used to distinguish sorted values from unsorted and to highlight the values that are being swapped. The applet window for this exercise is shown in Figure 3.

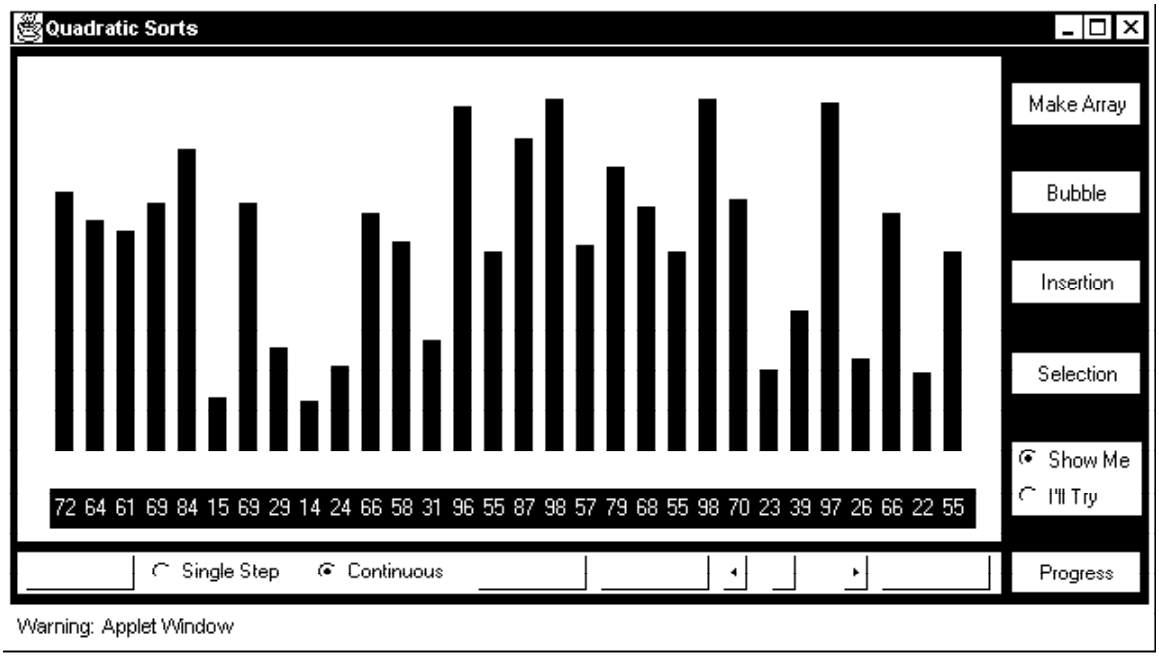

Figure 3 Quadratic sort exercise

The third sorting exercise illustrates two O(n log n) sorting algorithms—the merge sort and the quick sort. The animation of the merge sort is the most unusual. To visualize the merging step, we use a temporary array to copy to while merging the subarrays. Animation of sorting is nothing new, but the *I'll Try* mode, as in all the previous exercises, offers students the opportunity to reproduce the steps of the algorithm as a method of testing their knowledge.

5. EXPERIMENTAL USE OF THE COURSEWARE

This courseware has been used in *CSci 131, Data Structures*, at The George Washington University for the past three semesters. The courseware complements the material in Chapters 10-14 of the textbook used in this course, Feldman's *Software Construction and Data Structures with Ada 95* [5].

5.1 Preliminary Trial and Pilot Study

During the spring 1997 semester, the courseware was used in a preliminary trial and during the fall 1997 semester, a pilot study was conducted to ensure the usability of the courseware and to determine and correct any problems. Students were ask to evaluate the courseware at the end of each of these two semesters. During the preliminary trial, problems with version 3.0 of Netscape, the browser used by most students, and the limited availability of machines equipped with Windows 95, prevented student from using the courseware extensively. The subsequent evaluation by students after the spring 1997 semester contained numerous recommendations that it be incorporated as homework assignments for the course. This approach was used in the pilot study in the fall 1997 semester. The wider availability of newer operating systems and browsers made it possible for students to use the courseware more extensively. The majority of students in both sections of the course completed at least some to the on-line homework assignments. The student evaluations following the pilot study revealed some difficulty in understanding how to use the courseware. These evaluations led to the incorporation of more extensive instructions and on-line help and the decision to use the courseware as laboratory exercises during the final study.

5.2 Experimental Design of the Empirical Study

During the spring 1998 semester the actual study was conducted. In this study the learning style of students was assessed and the *I'll Try* mode was provided to half the students each week. The groups receiving the *I'll Try* mode were rotated during the four week study. Our first hypothesis was that students who received the version that provides both the *I'll Try* and *Show Me* modes would perform better on the posttests than the students who received the version that contains only the *Show Me* mode.

In the study, we used a true experimental design—specifically the posttest-only control group design. The subjects of our study were students enrolled in *CSci 131*, *Data Structures*, at The George Washington University. Our experiment included two independent variables: (1) the software version—either with or without the interactivity provided by the *I'll Try* mode (2) the learning style of the students, either active or reflective. The dependent variables were the scores of students on the posttests.

The instrument that we chose to measure learning styles was the Felder-Silverman learning style inventory. This instrument produces a learning style that has four dimensions: (1) sensing or intuitive learners, (2) visual or verbal learners, (3) active or reflective learners, and (4) sequential or global learners. We chose this learning style model because the first three dimensions are all of interest. Previous studies [3] have demonstrated a difference between the benefit of algorithm animation among sensors compared to intuitors. The second dimension dividing learning styles into visual and verbal learners is pertinent because of the highly visual nature of our courseware. Finally, the third dimension is the focus of our second hypotheses, which is that, active learners, those who learn by trying things out, will benefit more from the interactivity than the reflective learners.

The other instrument that we used was a posttest—specifically, questions on the final examination of the course. The intent of our courseware is to help students either understand the steps of an algorithm, as with the heap sort exercise, or to understand a particular data structure definition, such as the definition of an almost complete binary tree in the binary tree categorization exercise. Our posttests measured precisely the concept that the courseware was designed to convey—can students reproduce the steps of a heap sort given a new heap, for example. It may interesting to determine whether watching an animation better enables students to answer questions about the efficiency of an algorithm, but using such questions introduces potential confounding factors. Consequently, we excluded them from our posttests.

5.3 Experimental Procedure

Our courseware was designed as a set of laboratory exercises. Students used the courseware in the recitation session laboratory before the lecture in which the corresponding material was discussed.

Our procedure was to randomly divide the students into two groups. The two treatments—interactive or noninteractive—were rotated between the two groups so all students received one treatment for two of the weeks and the other treatment for the other two. Our schedule for rotation is shown in Table 1.

Because our courseware is hosted on the World Wide Web, we limited access to it during the period of the study. We included password protection during this time. Furthermore, students were only given access to the courseware during the recitation sessions. When students accessed the courseware, they had to select their name and

provide the password. Depending upon the student's group and the week of the study, the student was automatically given the appropriate version of the courseware.

	Group 1	Group 2
Week 1	Interactive	Noninteractive
Week 2	Noninteractive	Interactive
Week 3	Interactive	Noninteractive
Week 4	Noninteractive	Interactive

Table 1 Rotation of treatments among groups

5.4 Experimental Validity

We have designed our experiment to try to minimize confounding factors. By comparing two versions of Web-based courseware rather than comparing courseware to classroom instruction, as many other studies have done, we have eliminated many potential threats to the validity of our results.

Next, we have intentionally rotated the control group with the treatment group each week. Such a rotation has two benefits. It eliminated the possibility that by chance we had divided the students into groups where one group had, for example, a significantly higher grade point average than the other group. In addition, because we used students in a real classroom setting, we had to be sensitive to fair treatment of all students. Had the treatment proven beneficial, students receiving the treatment would have performed better on the final examination and consequently improved their course grade. The rotation ensured that any benefit provided by the treatment was distributed equally among all students. We restricted the use of the courseware to the laboratory setting to prevent the possibility that someone other than the actual student would do the work and to prevent the students from collaborating.

6. RESULTS AND FUTURE WORK

Our experiment was conducted during the spring 1998 semester. The experiment spanned a seven week period at the end of the semester. The learning style questionnaire was given the first week, the students used the courseware for the next four weeks and they were given the final examination, which was the posttest, in the seventh week. A total of 33 students participated in the study, initially.

6.1 Experimental Results

Of the 33 students who began the study, all but one student took the final examination. We used three of the problems on the final examination as our posttest. Each problem had two parts. The first problem tested the material in two of the graph exercises in the courseware—categorizing graphs and graph searches. The second problem tested material in two of the binary tree exercises—tree traversals and categorizing trees, and the last one tested two of the sorting exercises—the selection sort and the heap sort. Recall that the students were randomly divided into two groups. The results of the individual posttest problems by group are shown in Table 2.

The results indicate that the quick sort was the most difficult and the graph search problem was the second most difficult. These results were not unexpected. What was not expected was the large—almost ten point—difference in performance between the two groups. Although we randomly formed these groups, by chance, one group performed much better than the other. This occurrence underscores why it was important to give both treatments to both groups. In this way we are able to factor out the group differences. The students in group 1 had the *I'll Try* mode for both graph exercises and tree category exercise and did not have it for the other three exercises.

	Group 1 average	Group 2 average	Overall average
Graph categories	74.51	93.33	83.33
Graph searches	48.24	66.67	56.88
Tree traversals	79.78	84.58	82.03
Tree categories	68.24	78.67	73.13
Selection sort	83.53	80.67	82.19
Quick sort	47.65	56.00	51.56
Overall average	66.99	76.65	71.52

Table 2 Posttest results by group

The arrangement was reversed for the students in group 2. In Table 3 we compare the performance for both groups of all exercises on which each group had the *I'll Try* mode with the others.

	Group 1 average	Group 2 average	Overall average
I'll Try exercises	63.66	63.90	64.52
Show Me exercises	70.32	72.98	72.57
Difference	-6.66	-9.08	-8.05

Table 3 Results comparing *I'll Try* and *Show Me*

The results are the opposite of what we would have expected. Both groups performed better on the exercises that did not incorporate the *I'll Try* feature. The difference is not, however, statistically significant. We should note that the above data includes all 32 students who both

completed the learning style questionnaire and took the final examination. Of these 32 students, one did not use the courseware at all and another nine did not use it all four weeks. We examined the same data, eliminating these ten students. It is not appreciably different. There was also no significant difference in performance between the active students and the reflective ones.

6.2 Redesigned Experiment

We believe that several problems in our experimental design may have confounded this result. The most important of these was that the students' performance on the final examination may have been influenced by what they learned from other sources—the classroom lectures and the textbook. Consequently, we plan to perform a second redesigned experiment, in which the posttest will be given immediately following the use of the courseware. This redesigned experiment will be only one week in duration to prevent the possibility that students will participate some weeks, but not others. Because it will be of shorter duration we plan to use only four exercises: the binary tree traversals, the graph searches, the heap and the quick sort. We plan to conduct this revised experiment in the fall 1998 semester.

7. CONCLUSION

The demand for multimedia courseware, particularly courseware accessible from the World Wide Web, seems likely to increase dramatically in the near future. It is important to know how such courseware can be made most effective. Studies have been done to measure the effectiveness of algorithm animation with mixed results. In our study, which incorporates both the visualization of data structures and algorithm animation into a highly interactive courseware environment, we did not detect a significant difference between those who received the interactivity and those who did not. We plan a second, more tightly controlled, experiment to attempt to more definitively answer this question.

8. ACKNOWLEDGMENTS

Our thanks to the students in CSci 131 at The George Washington University during 1997 and 1998 for their participation in this study.

9. REFERENCES

[1] Badre, Andre, Margaret Beranek, J. Morgan Morris and John Stasko. (1991) Assessing Program Visualization Systems as Instructional Aids, *Computer Assisted Learning, ICCAL '92 Proceedings.* Springer-Verlag, pp. 87-99

[2] Byrne, Michael D., Richard Catrambone, and John T. Stasko. (1996) Do Algorithms Aid Learning? Georgia Institute of Technology Technical Report GIT-GVU-96-18

[3] Crosby, Martha E. and Jan Stelovsky. (1995) From Multimedia Instruction to Multimedia Evaluation, *Journal of Educational Multimedia and Hypermedia*, 4(2/3), 147-162

[4] Felder, Richard M. and Linda K. Silverman. (1988) Learning and Teaching Styles in Engineering Education, *Engineering Education.* 78(7), 674-681

[5] Feldman, Michael B. (1997) *Software Construction and Data Structures with Ada 95*. Reading, Mass: Addison-Wesley Publishing Company

[6] Lawrence, Andrea, Albert M. Badre and John T. Stasko. (1994) Empirically Evaluating the Use of Animations to Teach Algorithms, *Proceedings of the 1994 IEEE Symposium on Visual Languages*, St. Louis, MO, October 1994, pp. 48–54

[7] Marshall, A.D. and S. Hurley. (1996) Interactive multimedia courseware for the World Wide Web, *Proceedings of the Conference on Integrating Technology into Computer Science Education*. Barcelona, Spain, June 1996, pp. 1-5

[8] Naps, Thomas L. (1996) Chair, Working Group on Visualization. An overview of visualization: it use and design, *Proceedings of the Conference on Integrating Technology into Computer Science Education.* Barcelona, Spain, June 1996, pp. 192-200

[9] Stasko, John, Albert Badre, and Clayton Lewis. (1993) Do Algorithm Animations Assist Learning? An Empirical Study and Analysis, *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, Netherlands, April 1993, pp. 61–66