Assessing Program Visualization Systems as Instructional Aids

Technical Report GIT-GVU-91-23
October 1991
Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology

Albert Badre[1]
Margaret Beranek[2]
J. Morgan Morris[3]
John Stasko[1]

[1]Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

[2]Computer Information Systems Department
Georgia State University
Atlanta, GA 30303

[3]Department of Mathematics and Computer Science
Georgia State University
Atlanta, GA 30303-3086

Abstract

Recently, program visualization systems have received much attention as learning tools and as software understanding aids. How to evaluate these systems, however, is an open and unexplored area. In order to determine what factors may be important, we conducted an exploratory study using XTango, an algorithm animation system. First, we asked professors to complete surveys intended to solicit information regarding current practices in the teaching of algorithms. Next, we observed two groups of students: one group received a handout and viewed an animation of the Shellsort algorithm, the other received the same handout and listened to a lecture featuring drawings on the blackboard. The students were queried on their understanding of the sort and their impressions of the animation system. Comments indicated a high perceived value for the system, with most students favoring its use as a teaching tool. It was clear from students' responses that an algorithm animation system can be used more effectively as a supplement in the classroom environment than as a substitute for the teacher. The results of this study identified changes to the animation system that will help integrate it into the classroom environment, and provided several important factors to consider in future empirical studies.

Keywords: program visualization, algorithm animation, software understanding, computer aideed instruction, educational software

1 Introduction

Recently, *program visualization* systems[Mye90] have garnered increasing attention as aids for software understanding and for teaching computer science. Program visualization systems provide graphical views of the constituents, methods, and techniques of computer programs. They enable end-viewers to "look inside" the mysterious black-box of a program and visualize its inner workings in a convenient, (hopefully) understandable visual format or metaphor.

Program visualization systems range from showing programs at a low level of detail, such as showing data structure manipulations, to showing programs at a more global level, such as showing the program's purpose and methodologies[SP91]. Systems presenting a more global, high-level abstract view are called *algorithm visualization* systems. Often, these systems present a continuous, smooth view of a program during its execution, this earning the name *algorithm animation.*

Over the past ten years, a number of algorithm visualization systems have been created[LD85, Dui86, Bro88, HHR89, Sta90]. Each concentrates on a slightly different aspect of the problem of viewing program execution, and all have their individual merits. The clear focus to this work, however, has been on improving the systems technology and visual techniques involved. Somewhat lost has been the original motivation for the systems—how to improve software understanding through pictures. Although many systems have been designed, no systematic evaluation of the systems' utility has been conducted. No formal end-user testing has been performed. Essentially no follow-up to the claim that these systems improve software understanding has been checked. The small amount of system evaluation done to date is largely anecdotal.

Perhaps the most extensive application of algorithm visualization technology was the use of the Balsa system[BS85, Bro88] as an aid for teaching algorithms at Brown University in the mid-1980s. Classes were taught in a large auditorium with Apollo, and subsequently Sun, workstations. Each lecture was accompanied by visualizations of the algorithm(s) being taught that day. Informal, anecdotal results of this application were good; students frequently mentioned how the visualizations helped them understand how the algorithms worked. No formal study or attempt to quantify how much the visualizations helped was conducted, however.

Instructors in a class on analysis of algorithms at San Diego State University utilized the MacBalsa[Bro88] system (a descendant of Balsa) and a local system called Algorithms Lab to help teach algorithms[WU90]. The instructors noted that the better students seemed to derive more and learn more from the addition of the computer visualizations. The authors believed that the minimal computer science background of the weaker students somewhat lessened the cognitive gains made by these students. The students' impressions of the visualizations were difficult to quantify precisely. The instructors noted, "Motivation and enthusiasm defy simple measurement." The visualization systems, however, did receive favorable feedback from the students; most felt the visualizations added to the course.

One reason for the lack of user testing in algorithm visualization is the sheer difficulty involved. What are the appropriate visualizations to utilize? How do we measure program understanding? What criteria are used for evaluation? How is the evaluation properly and fairly conducted? We believe that these difficulties are the reasons for the lack of significant system testing, and this has motivated the exploratory study we report in this paper. Rather than performing an evaluation of an animation system immediately, our goal was to learn how to evaluate an algorithm animation system.

We wanted to learn how experienced teachers might utilize this technology as an aid in the classroom. We wanted to learn how to evaluate a student's understanding of a program, and we wanted to learn what problems might arise in performing such testing.

We also wanted to learn what constitutes a good algorithm animation. How should a student

be able to interact with the system? What further capabilities would be beneficial? For example, adding sound and help facilities would make a system more of a multimedia tool, and might significantly improve usefulness.

Existing work in algorithm visualization has given us intuition about the answers to these questions. But as has frequently been shown, intuition is not always correct. Consequently, we set out on this study to gather important evidence for confirming or disputing our beliefs. We focused on identifying the factors that would be important in conducting a more formal, empirical study. In a subsequent project, we will use the information we gathered to perform an exhaustive testing of the impact of an algorithm animation system on instruction.

2 Exploratory Study

The exploratory study consisted of two parts: a survey of faculty for current teaching practices, and an observational study of students interacting with an algorithm animation system. The faculty survey on teaching practices strived to identify current teaching methods for algorithms, specifically, uses of drawings, pictures and conceptualizations of algorithms. The observational study investigated the effectiveness of computer animation to enhance algorithm comprehension in a classroom environment. The computer animation was implemented on top of the algorithm animation system, XTango[SH90] that helps to design graphical representations of data structures and algorithms. The animations were presented to students as a supplemental learning tool, and the students were tested to determine their knowledge and ability to conceptualize the algorithm presented to them. The algorithm was also presented to a second group in a regular classroom environment using blackboard lecture style.

2.1 Faculty Survey

An important part of this study was to identify current methods of teaching algorithms and to identify appropriate conceptualizations of the algorithms to be used in the computer animation. These conceptualizations, or mental models, indicate how the professor visualizes the algorithms and how they transfer their vision of the algorithms to their students. To acquire this information we surveyed professors who have taught algorithms courses at two different universities. The survey consisted of two parts. Part one asked for information about the professors' background in teaching algorithms, the teaching methods used, their use of teaching aids such as textbooks, drawings and diagrams, overhead transparencies, and the use of supplemental notes.

Part two consisted of a list of four algorithms. The professors were requested to draw a diagram to emulate their conceptualization of the algorithms. The drawing could be accompanied by verbal explanations and/or color. The drawing could be static, or it could be a dynamic conceptualization, with motion indicated by the use of multiple drawings, arrows, etc. This helped us identify appropriate conceptualizations of algorithms and data structures to be used in computer animations.

Results and Discussion

Eleven professors responded to the questionnaire. In terms of current practices, the most common teaching method (36%) was a combination of lectures, textbook and use of drawings and diagrams. 82% of the professors used a textbook to supplement lectures, 81% also used drawings and diagrams as a supplement, while only 26% used overheads and only 27% used other supplementary materials. From this, along with the results from part two of the survey, we can conclude that most of the

professors attempt to show the dynamic nature of the algorithms through the use of a series of drawings. Only one professor indicated that he had used any form of computer animation or graphics package in the classroom or laboratory to supplement his lectures. This would indicate that in the teaching of algorithms it is necessary to somehow show the dynamic nature of the operations of these algorithms, but very few, if any, professors are currently using any form of computer graphics package as an aid.

With regard to general conceptualizations, all of the respondents used separate phases or steps in their drawings to indicate dynamic representations for at least one algorithm. 86% of the total conceptualizations described did contain dynamic components. Four professors responded with conceptualizations of the Shellsort, which we chose as the algorithm to represent in our study. Other sorting methods, including bubble sort, selection sort, and insertion sort, are frequently taught in introductory programming courses. Since the Shellsort is usually taught at the data structures level, it was less likely that the Shellsort would be familiar to the students. The concepts involved in the Shellsort may be learned quickly, requiring knowledge of either a bubble sort or an insertion sort. In the Shellsort, subarrays are sorted in each pass, until the entire array is sorted in the final pass. The subarrays are formed using elements a specified distance apart. A sequence of distances are chosen, with the last distance having a value of 1 so that the entire array is sorted. For example, the sequence of distance values could be 40, 13, 4, 1. Each subarray is sorted using either bubble sort or insertion sort.

Of the four Shellsort conceptualizations that the professors described, two used bars of various sizes lined up on a horizontal axis. The height of the bars represented the values to be sorted. Exchange of the bars, or data values, during the sort was indicated by a change in color or circling the bars to be exchanged. A third representation lined up the numbers of the data values on a vertical axis and indicated an exchange using arrows. The fourth representation was from Knuth[Knu73]. We chose the first representation as the one used in our animation.

2.2 Observational Study

The case study group consisted of students from three separate undergraduate data structures classes. They are an appropriate audience for this study, since students usually get their initial exposure to many algorithms in a data structures course. Eleven students volunteered to participate. Approximately 64% of the students were in a Computer Science degree program and 36% were in a Computer Information Systems degree program.

The algorithm animation system we used in our study is the XTango system[SH90]. XTango supports color, two-dimensional views of programs in a workstation-based windowing environment. In our experiments, end-users were viewing animation previously created and designed to help explain different algorithms. XTango is a descendant of the original Tango system[Sta90], and it is a simpler and more portable system. XTango functions directly on top of the X11 Window System, and it generates color or black-and-white animations without any intermediate software. XTango is available via anonymous ftp, and over 200 installations worldwide have acquired the system. It has been used for teaching computer science courses, illustrating the workings of an operating system, illustrating robot planning, doing VLSI chip drawings, etc. As more institutions acquire it, we anticipate further new and interesting usages.

Figure 1 shows a frame from the XTango animation of the Shellsort. In it, the array of values to be sorted is illustrated by a row of rectangles in the top half of the window. Each rectangle's height has been scaled to correspond to the value it represents. The animation illustrates the Shellsort by lowering all the elements in an equidistant subarray, keeping their horizontal orientation constant, and then exchanging and sorting those elements below. When the subarray has been sorted, it

is raised back into the main array, and the next set of elements descends. This continues, using subarrays of decreasing spacing size, until the entire array is sorted. The frame we show has caught a lowered subarray of elements being sorted.

In our study, five of the students received a handout describing the Shellsort plus a blackboard lecture depicting the effects of the sort for a specific data set. The other six students received the same handout, but instead viewed an animation of the Shellsort using XTango. Following the presentations, all of the students were asked to answer three questions. The first question asked that the students perform a Shellsort with a different data set and a particular subarray distance sequence by rearranging the numerical values, and showing the results of each step on a sheet of paper. In the second question, the students were presented with a set of rectangular bars representing data values, and asked to show the results of the sort for those values pictorially. The third question asked for the students to describe why the distances chosen for the Shellsort should not be relatively prime. After answering these questions, the students were given a questionnaire to gather subjective information about animations. They were also asked for their comments regarding the use of animations in a classroom setting.

Results

Performance results varied depending on individual background. Most students reported their expected grade to be at least a B. These good students were observed to perform well regardless of whether they used the animation. Further, the technically-oriented students also performed well irrespective of their use of animation.

A primary focus of this observational study was to determine typical attitudes toward animation systems. The blackboard students were presented with a demonstration of XTango following completion of the experiment, so that they also could comment on the system. The results showed that the students enthusiastically support the animations when asked to rank perceived quality. When asked to rank the quality of XTango on a scale from poor to excellent, no student ranked the system toward the poor end of the scale, with most assigning a rank of very good or excellent. Similarly, most ranked the system as being understandable and efficient. One student who ranked the quality of the system as fair and who found the system confusing made the following statement: "From the beginning of the graphic program execution I was unsure of what was what; more labeling of visual elements would help. If text was included with the animation, it would be appropriate for the classroom. I didn't completely understand the animation – I had more questions about it than it could answer." Other students indicated that the use of color was confusing at first, and others indicated problems with the system response time.

Students were also asked to comment on the appropriate role of an animation system in the classroom. All indicated that they would like to see animations become part of a data structures course. One stated, "I thought it was helpful. I couldn't remember rules, but I could visualize what was going on." When asked whether they should be a part of a separate laboratory or integrated into the traditional classroom setting, most were in favor of integrating them into the classroom. One student differed, stating, "I think they would be helpful in the classroom. Personally, I could use a separate animation system with textual explanations added, but I would like to have someone around to ask a question." Most students preferred the typical group interaction that takes place in the classroom, with one student saying, "We still need a teacher and group to benefit from discussion from others."

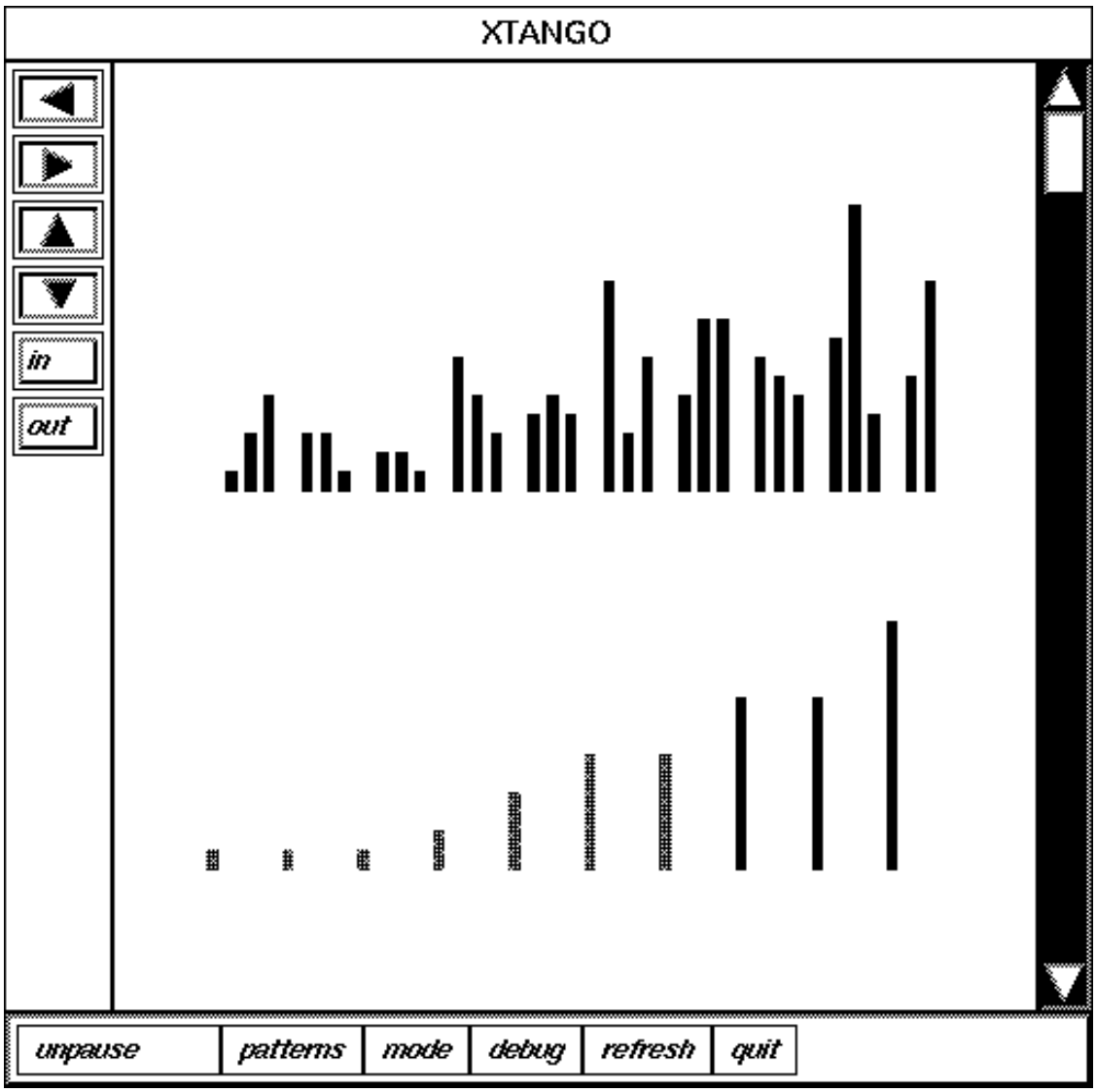

Figure 1: Sample animation frame from the XTango Shellsort animation. A subarray of elements being sorted has been lowered. When this set is in order, it will rise back in place.

3 Conclusion

3.1 Lessons Learned

Discussion

The faculty survey results suggest that the current practice for teaching algorithms is a combination of lectures, textbooks, and drawings. This information, although anticipated, helps define the instructional environment expected by a student learning computer algorithms. Any instructional strategies incorporating animations should follow this model in order to match students' expectations.

The use of drawings in the classroom seems necessary to impart the dynamic qualities of an algorithm in execution. These dynamics are the basis for the existence and use of algorithm animation systems such as XTango. For any given algorithm there may be multiple representations of the objects and actions during execution. The survey results indicated no definitive representation of any of the algorithms. Most of the replies were influenced by depictions provided in textbooks or through prior experience with animation systems. Although none of the responses contradicted the views frequently used in XTango, more research is needed to explore the qualities that comprise an effective view for an algorithm.

The results of the observational study identify a high perceived value for XTango, with many students stating that they would like to see similar animations integrated into data structures courses. Such reports are crucial to the use of animation systems, and enthusiasm could translate into long-term performance gains during a data structures or algorithms course.

Little effort has been directed toward evolving animation systems into effective teaching tools. The responses from students using such systems are the ultimate test of their value. Although the responses about XTango were highly favorable, many suggestions were made that could enhance the effectiveness of XTango as an educational tool. Some suggestions for improvement include better response time, consistent use of labels to accompany states and actions during an animation, and textual explanations of the algorithms. Such reports are consistent with usability principles and with current classroom practices. Adding value labels to the Shellsort animation would be beneficial, according to student comments. Value labels are appropriate for small data sets intended to demonstrate the mechanics of an algorithm. Value labels are impractical, however, with large data sets, particularly when the animation compares several algorithms presented simultaneously on the screen.

The observational study provided valuable insights into formal studies intended to gauge the effects of animation on performance. Important variables to control include academic and technical background, since they will obscure any other variables which may affect performance. Data about spatial abilities should be collected and analyzed. It is possible that poor visualizers would benefit most from the explicit visual representations made possible by animation systems. Data regarding prior experiences with visual technologies, including game-playing and educational systems, should be collected from the participants.

Many algorithms may have inherent visual qualities for which animations can enhance learning and performance. For example, a common view of the heap sort uses a visual representation of the heap, with elements gradually migrating to the sorted array. One student who took part in the observational study who was only moderately enthusiastic about the Shellsort animation was highly impressed with the heap sort animation used in XTango. It is difficult to determine whether this difference may be attributed to the relative quality of the views for Shellsort and heap sort, or if the heap sort algorithm is more suited to a visual representation. A formal study should examine several algorithms in order to gain more insight into this issue. Additionally, the experimental task

must be constructed carefully in order to identify any performance benefits of animations.

Summary

The following is a summary of what we have learned in this initial attempt to evaluate animation systems.

- Many instructors report the use of lectures, textbooks, and drawings to aid the teaching of algorithmic concepts.
- The views reported by instructors report no definitive views and do not contradict those currently used in XTango.
- Students were receptive and enthusiastic towards animations, and would like to see them included in the classroom.
- Performance results are difficult to relate to the effects of an animation, since many factors may influence learning. The factors that must be considered in an empirical study include include:
  1. Academic and technical background.
  2. Spatial abilities.
  3. Prior experience with visual technologies.
- Some algorithms may have more of an inherent visual quality than others. This implies differences in performance benefits by students for animations of different algorithms.
- Generating an experimental task that evokes differences in visual representations may prove difficult or artificial.
- Several features of XTango animations may be improved or manipulated to determine their effects.
  1. The addition of labels to images.
  2. The addition of textual explanations of the algorithms.
  3. Improvements in system response time.
  4. The adjustment of execution time for the algorithm.
  5. The relationship between large and small data sets and the implications for labeling.
- Students prefer that animation tools such as XTango augment the current instructional environment and do not want them to replace the classroom setting. Students seem to prefer the benefits of group interaction for answering questions.

3.2 Future Work

The observational study cited here is the initial step in our project to examine the use of animation to promote the comprehension of algorithms and, more encompassing, the use of computer animation to enhance learning in a classroom environment. The major objectives of the study are to determine if learning is increased, if learning is increased over time, if there is a difference in learning between different algorithms, what types of views better promote comprehension, and

ultimately, whether animation should occupy an important place in a computer science classroom environment.

The next step in this line of research is developing a longitudinal study in a classroom environment, similar to the study of animations and physics in [RBA90]. We will make appropriate changes to the animations as suggested by this study, to make the graphical images easier to interpret and provide for a more responsive system. These changes include the addition of more text and legends to the animations. An implementation in a classroom environment will provide us with a sizable user community and the ability to conduct a longitudinal study.

We will increase the generalizability of the software as a learning tool for algorithms and, to control for learning differences between different algorithms, use it to teach several different algorithms. A second change we may explore is the use of a videotape of the animations for use in a classroom environment. The videotape will become part of the instruction received in classes, and it would allow us to reach a larger audience. We still plan to have students view animations on-line as well, because of the interactive nature of some of the animations. In addition to being tested on comprehension of the algorithms, the students will also be tested at a later dates to determine whether animations lead to changes in retention over time.

This observational study has also suggested future improvements to the animation system itself. We plan to explore the inclusion of sound, voice, textual information, and on-line help to make the system more of a multimedia tool. Such improvements could make the animation system a valuable instructional unit, which a student could independently progress through at an appropriate pace.

Acknowledgements

Reid Turner implemented the XTango Shellsort animation. We thank him for his contribution.

References

[Bro88] Marc H. Brown. Exploring algorithms using Balsa-II. *Computer*, 21(5):14–36, May 1988.

[BS85] Marc H. Brown and Robert Sedgewick. Techniques for algorithm animation. *IEEE Software*, 2(1):28–39, January 1985.

[Dui86] Robert A. Duisberg. Animated graphical interfaces using temporal constraints. In *Proceedings of the ACM SIGCHI '86 Conference on Human Factors in Computing Systems*, pages 131–136, Boston, MA, April 1986.

[HHR89] Esa Helttula, Aulikki Hyrskykari, and Kari-Jouko Räihä. Graphical specification of algorithm animations with Aladdin. In *Proceedings of the 22nd Hawaii International Conference on System Sciences*, pages 892–901, Kailua-Kona, HI, January 1989.

[Knu73] Donald E. Knuth. *Sorting and Searching*. Addison-Wesley, Reading, MA, 1973.

[LD85] Ralph L. London and Robert A. Duisberg. Animating programs using Smalltalk. *Computer*, 18(8):61–71, August 1985.

[Mye90] Brad A. Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing*, 1(1):97–123, March 1990.

[RBA90] Lloyd P. Rieber, Mary J. Boyce, and Chahriar Assad. The effects of computer animation on adult learning and retrieval tasks. *Journal of Computer-Based Instruction*, 17(2):46–52, Spring 1990.

[SH90] John Stasko and J. Douglas Hayes. *The XTANGO Algorithm Animation System, User Documentation.* GVU Center, College of Computing, Georgia Tech, Atlanta, GA, December 1990.

[SP91] John T. Stasko and Charles Patterson. Understanding and characterizing program visualization systems. Technical Report GIT-GVU-91/17, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, September 1991.

[Sta90] John T. Stasko. TANGO: A framework and system for algorithm animation. *Computer*, 23(9):27–39, September 1990.

[WU90] Roger E. Whitney and N. Scott Urquhart. Microcomputers in the mathematical sciences: Effects of courses, students, and instructors. *Academic Computing*, 4(6):14–18,49–53, March 1990.