Review of animation systems for algorithm understanding

Judith Wilson and Robert Aiken
Temple University
Irvin Katz
Educational Testing Service

Abstract

We survey how several algorithm animation systems are used in Computer Science instruction. Reported student reactions to the use of these systems is favorable, but little information is available on their effectiveness for learning. We examine several formal studies that have implications for how animation systems can most effectively be used to teach algorithms.

1 Introduction

Recently, a great deal of time and resources have been devoted to developing animation systems for teaching Computer Science (CS) algorithms [13, 3]. Animations of an algorithm's execution typically include dynamic visualizations of its changing data structures during execution or of its behavior in a microworld (e.g., the progression of a search algorithm as it solves a search problem defined on a map of cities). These systems provide students with concrete, often interactive, environments in which they can explore the behavior of algorithms in specific domains and learn abstract concepts underlying these behaviors. It is assumed that such systems help students learn algorithms better than they could otherwise. It has also been suggested that animation systems make possible the teaching of subjects that have been very difficult, or impossible, to teach without such systems. For example, visual animations are among the best tools '.. for conceptualizing the computational process..' of parallel algorithms [7].

We describe how several algorithm animation systems have been used in CS instruction and note informal reports of their educational value. We then review several formal studies of the effectiveness of such systems for learning algorithms.

2 Instructional use of algorithm animation systems

Visual animation systems are being used with increasing frequency in classroom instruction. Naps reports in 1994 that over 60 institutions are using GAIGS [12], and, in early 1995, Stasko reports that more than 300 sites have accessed the Xtango system since 1991 [3]. In addition, a number of animation systems that have been developed by instructors for use in their classes are described in the literature. However, detailed information about how these visualization systems are used in instruction is difficult to find. Following are several examples.

Integrating Tech. into C.S.E. 6/96 Barcelona, Spain
© 1996 ACM 0-89791-844-4/96/0009...$3.50

2.1 Balsa

Balsa, developed by Mark Brown in the early 1980s, is commonly recognized as the first major algorithm animation system. Balsa supports multiple simultaneous views of an algorithm's data structures and can display multiple algorithms executing simultaneously. Brown describes Balsa's use in an introductory course, and in an algorithms and data structures course [2]. The system was used as a program visualizer in the introductory programming course and as a high-level algorithm animator in the advanced course. Scripts of keystrokes, which were designed by instructors to use in lectures as demos, could be replayed later by students. Brown reports that the use of animation scripts to supplement lectures led to 'demonstrable gains in speed of comprehension ' over the traditional lecture. In both courses, student response to the animated demos was reported as positive.

2.2 Xtango

Xtango (X windows version of Tango [15]) uses a path-transition paradigm to achieve smooth animations. Animation events are not tightly coupled with program events permitting animation of algorithms running on different systems and implemented in different languages. Xtango supports animation of program source code along with high-level visualization of algorithm data structures. Users can select from a large library of animated algorithms or design their own.

The flexibility of Xtango permits its use in a variety of instructional contexts. Rodger, for example, describes 'interactive lectures' in which the instructor can step through an algorithm animation generated by Xtango pausing at key points to allow students to predict what will happen next, which in turn influences subsequent discussion. Rodger reports that students have been positive about the animation tools [14]. Hartley has used Xtango to animate synchronization algorithms (written in SR) for an operating systems course [4] and reports (privately) that students were enthusiastic about the animations.

2.3 GAIGS

GAIGS (Generalized Algorithm Illustration through Graphical Software) [11, 12] is not a true animator but generates a sequence of discrete snapshots of an algorithm's data structures during execution with the advantage that the student can backstep through the algorithm. GAIGS supports different simultaneous representations of the same data structure. GAIGS

was designed to support a 'visualization laboratory' that builds conceptual understanding through experimentation without programming.

Naps, a principal designer of GAIGS, describes several different types of labs that structure experimental activities around use of the animator. Each type of lab features a set of learning tasks designed to achieve a specific kind of learning objective. In discovery labs, for example, students view animations of an algorithm before it is presented in lecture and then deduce the algorithm with the help of lab sheet questions provided by the instructor. In comparison labs, students view different implementations of a data structure and discover through observation the strengths and weaknesses of each. Naps reports that the visualization laboratory has been 'received favorably' [11] and that improved programming skills and significantly better test scores have been observed [3].

2.4 FLAIR

Whereas Balsa, Xtango and GAIGS are systems for animating algorithms, FLAIR is a system that includes algorithm animations as components [6]. FLAIR is a repository of instructional materials for the undergraduate AI course that runs on Sun workstations. These materials include laboratory-based learning environments (called 'modules') in which students learn through experimentation. The Search Module, for example, animates the execution of standard search algorithms on a map of cities. Students can select the algorithm to animate, the heuristic, the start and goal cities, adjust speed, step and pause, and create new city maps. In addition, students can activate a concurrent detailed animation of the underlying list data structures. Multiple windows permit students to directly compare the performance of different algorithms running concurrently on the same search problem. Guided by instructor-provided learning tasks, students conduct experiments on the algorithms and investigate, for example, how different algorithms 'behave' under different circumstances and the circumstances under which one algorithm performs better than another.

Students clearly enjoyed the Search Module and were highly motivated by focused task assignments in the animated environment. But was there a measurable benefit for learning?

3 Formal Studies

Anecdotal evidence suggests that the use of animation systems to teach algorithms has won the approval of students and has increased student interest and motivation. Yet little or no research has been conducted to test the effectiveness for learning of animation systems used in specific instructional contexts.

Several experimental studies have been conducted by Stasko and his associates at Georgia Tech. An early exploratory study of the educational benefits of Xtango found that, although there was a high perceived value for the system, students favored use of the system as a supplement to classroom learning rather than as a substitute for the instructor [1]. Lawrence found that student preferences did not predict performance on post tests to measure learning, and that adding a textual description of an algorithm increased accuracy on conceptual questions [8].

Other studies indicate that there are learning gains when algorithm animations are used to supplement lectures, but that the greatest gains are realized by students who actively engage the animation system. A study of the effects of the use of dynamic graphical displays for solving array manipulation problems suggests that although students benefit from viewing dynamic displays of algorithm performance, they benefit more when allowed to interact with the display by manipulating data elements [5]. These findings were supported by an experimental study at Georgia Tech. In this study, all student subjects were given a lecture on an algorithm. Students who had access to a laboratory in which they interacted with the animation by defining their own data sets demonstrated more accurate performance on a post test than did students who had the lecture only or who used instructor supplied data sets in the laboratory [8, 9].

A pilot study has been conducted to evaluate the effectiveness of FLAIR's Search Module environment for learning AI search algorithms [17]. Two students with no background in heuristic search algorithms and no experience with the FLAIR system worked collaboratively on a task that required comparing the A* and Best First search algorithms on a variety of search problems in order to explain the behavioral differences between the algorithms. Before starting, the students were given a brief overview of heuristic search and of the module environment. Though the students had access to a detailed display of data structures, they tended to rely on the high-level map animation along with pseudocode specifications of the algorithms. A key discovery was made during the second hour when the students observed that A* sometimes 'jumps around' on the map (whereas Best First never does). This visual discovery facilitated the activities that led the students to develop a reasonably good explanation of the observed differences between the two algorithms within two hours. Results from the study suggest that focused tasks in an animated environment that provides access to a range of materials (including text) can facilitate learning. The study also suggests that different types of algorithm animations may be suitable for different types of learning tasks. The students may have favored the high-level map display over the detailed list display because the map display provides insight into the reasons behind changes in the list data structures which the students needed in order to explain the differences between the algorithms.

4 Implications

The classroom experiences and formal studies discussed above have several implications for how animation systems can be used most effectively to teach algorithms. First, students must be actively engaged with the animation tools. The greatest improvements in performance have been observed when students interact with an animation system by designing their own datasets or by interactively modifying runtime data structures. Second, learning is facilitated when animations are included as part of an instructional context which also includes textual components and which features carefully focused task assignments designed to achieve specific learning objectives. Performance improvements observed when using the GAIGS laboratories may be due largely to the structured environment in which the animations played a part. Finally, different types of animations may be suitable for different types of learning tasks. If so, then achieving specific learning objectives may require the development and use of specific types of animations.

5 Conclusion

Computer Science instructors need guidelines for designing and using animations in ways that help students learn algorithms and not just entertain. Developing useful guidelines will require more formal investigation into the types of animations and instructional environments that are most effective in achieving the learning objectives of Computer Science education.

References

1 Badre, A., Beranek, M., Morris, J.M., and Stasko, J. (1992) Assessing Program Visualization Systems as Instructional Aids. *Proceedings of the fourth International Conference on Computer-Assisted Learning* (Wolfville, Canada), 1992.

2 Brown, M., and Sedgwick, R. Progress Report: Brown University Instructional Computing Laboratory, *Proceedings of the ACM SIGCSE Technical Symposium* (1984) 91-101.

3 Grissom, S., Hunkins, D., Naps, T., Rodger, S., Ross, R., and Schweitzer D. Using Visual Demonstrations for Computer Science Education, A Tutorial given at the *ACM SIGCSE Technical Symposium on Computer Science Education* (Nashville, TN), March 1995.

4 Hartley, S. Animating Operating Systems Algorithms with XTANGO. *Proceedings of the ACM SIGCSE Technical Symposium* (1994) 344-345.

5 Hays, H. D. Interactive Graphics: A Tool for Beginning Programming Students in Discovering Solutions to Novel Problems. *Proceedings of the ACM SIGCSE Technical Symposium* (1988) 137-141.

6 Ingargiola, G., Hoskin, N., Aiken, R., Dubey, R., Wilson, J., Papalaskari, M., Christensen, M., Webster, R. (1994) A Repository that Supports Teaching and Cooperation in the Introductory AI Course, *Proceedings of the ACM SIGCSE Technical Symposium* (1994).

7 Johnson, D., Kotz, D., and Makedon, F. Teaching Parallel Computing to Freshmen. *Conference on Parallel Computing for Undergraduates* (1994). ftp://ftp.cs.dartmouth.edu/pub/CS-papers/Kotz/Johnson: freshmen.ps.Z

8 Lawrence, A. Empirical studies of the value of algorithm animation in algorithm understanding. Unpublished doctoral dissertation, Georgia Institute of Technology, 1993.

9 Lawrence, A.W., Badre, A.N., and Stasko, J.T. Empirically evaluating the use of animations to teach algorithms (Technical Report GIT-GVU-94-07). Atlanta, GA: Graphics Visualization, and Usability Center, Georgia Institute of Technology, 1994.

10 Makedon, F., Rebelsky, S.A., Cheyney, M., Owen, C., and Gloor, P. Issues and Obstacles with Muiltimedia Authoring, Invited Presentation, *EdMedia '94* (Vancouver, Canada), 1994. http://www.cs.dartmouth.edu/reports/abstracts/PCS- TR95-256.html.

11 Naps, T.L. Algorithm Visualization in Computer Science Laboratories. *Proceedings of the ACM SIGCSE Technical Symposium* (1990) 105-110.

12 Naps, T.L., and Swander, B. An Object-Oriented Approach to Algorithm Visualization - Easy, Extensible, and Dynamic. *Proceedings of the ACM SIGCSE Technical Symposium* (1994) 46-50.

13 Price, B.A., Baecker, R.M., and Small, I.S. A Principled Taxonomy of Software Visualization. *Journal of Visual Languages and Computing* 4,3 (1994) 211-266. http://hcrl.open.ac.uk/jvlc/JVLC-Body.html.

14 Rodger, S.H. An Interactive Lecture Approach to Teaching Computer Science, *Proceedings of the ACM SIGCSE Technical Symposium* (1995) 278-282.

15 Stasko, J.T. Tango: A Framework and System for Algorithm Animation, *IEEE Computer* 23,9 (1990) 27-39.

16 Stasko, J., Badre, A., and Lewis, C. Do algorithm animations assist learning? An empirical study and analysis. Human Factors in Computing Systems: *InterCHI '93 Conference Proceedings* (1993) 61-66. New York: Addison-Wesley.

17 Wilson, J., Katz, I., Ingargiola, G., Aiken, R.M., and Hoskin, N. Student's Use of Animations for Algorithm Understanding. Short Paper in *CHI95 Companion to the Proceedings* (1995).