Animation and Visualization in the Curriculum: Opportunities, Challenges, and Successes

Thomas Naps
Computer Science Dept
U Wisconsin Oshkosh
naps@uwosh.edu

Susan Rodger
Computer Science Dept
Duke University
rodger@cs.duke.edu

Guido Rößling
Computer Science Dept
Darmstadt U of Technology, Germany
roessling@acm.org

Rockford Ross
(moderator)
Computer Science Dept
Montana State University
ross@cs.montana.edu

SUMMARY

This panel is intended for all instructors who have had a desire to incorporate animation and visualization tools into their courses, for those who may not even be aware of such tools and of their potential benefits for instruction and student learning but want to investigate them, and for those who may want to become involved in the design and implementation of effective animations or visualizations. First, some background.

Software systems for animating or visualizing important computer science concepts have long held strong allure for educators. It is sometimes difficult to convey the often complex and dynamic nature of computer science in the classroom satisfactorily; algorithms, models of computation, machine execution, and a virtually uncountable number of other topics beg to be animated. In most cases, an instructor is the animating agent, using a whiteboard, various colored pens, an eraser, and sometimes inscrutable body gyrations to convey the dynamics of the process being described. As talented as an instructor might be at this task, students invariably leave the classroom with little more than static notes and questionable impressions of the presentation.

The promise of animation and visualization is that software can serve as the animating agent for such dynamic processes. A student could interact with such software as often and in as much depth as desired until the topic was learned—algorithms would become clear, models of computation would be understood, and machine execution processes assimilated. An instructor could run the same software in the classroom to convey the dynamics of the topics under consideration in repeatable, error-free fashion.

That has always been the promise. However, a number of very good software systems (and many more "toy" systems) have been developed over the years that never did see widespread use in the classroom. What happened? In retrospect, a number of issues are evident. Early systems were inherently platform dependent and would not run at many sites other than the creator's. Most of these systems died as new computing platforms replaced the ones on which the systems were developed. More recent systems were designed to be platform independent, but still suffered from lack of widespread use. The main issues were time and support. An instructor who would have liked to use animation and visualization software in a course had to (1) be made aware of its existence (2) locate it (usually through a web search or published URL), (3) probably download and install it, (4) learn to use it, (5) possibly create some animations with it, (6) integrate it into an existing course that likely used different notation than the software, (7) teach its use to students, and (8) assign and grade exercises using it. For instructors who had no time or interest in developing custom animations for their own classroom, the situation was often hopeless. For those who did want to become involved in the design of animations or visualizations using existing systems, the outlook wasn't much better, as extant systems often were poorly documented and unsupported.

If that sounds daunting, there is good news! Recent advancements in the understanding and development of animation and visualization systems have addressed (or are addressing) many of these issues. Thanks to the concerted efforts of a number of researchers, often with generous support from the National Science Foundation and other granting agencies, there has never been a better time to consider integrating animation and visualization tools into the curriculum, or to become involved in the design and development of such tools.

SIGCSE'06, March 1-5, 2006, Houston, Texas, USA.

ACM 1-59593-259-3/06/0003.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education.

General Terms

Algorithms, Design, Human Factors, Theory.

Keywords

Algorithm animation, algorithm visualization, visualization, hypertextbooks, active learning, interactive learning applets.

1. Tom Naps

Tom Naps is Professor of Computer Science at the University of Wisconsin Oshkosh. He has published in the area of animation and visualization since the late 1980s and is well-known internationally for his algorithm visualization systems. His most recent system is known as JHAVÉ. Tom is a regular participant in the SIGCSE and ITiCSE communities. His work can be found at http://jhave.org/. His most recent work is supported through NSF-CCLI grant DUE-0126494, with Scott Grissom and Myles McNally.

There are many reasons why algorithm visualization (AV) has not been effective as an educational resource despite the overwhelming "belief" among CS educators that it should be.

Among these reasons are students being too passive in their interaction with such systems and the lack of convenience and reliability that such systems offer instructors. Consequently our work in trying to make AV more effective should be aimed at correcting these deficiencies rather than adding more graphical flourishes to AV systems. To more actively engage students with visualizations, AV systems need to include "engagement hooks" such as (1) allowing students to provide meaningful input sets to control the direction of algorithms they are watching, (2) asking students non-trivial questions about the visualizations they are watching, and (3) incorporating the delivery of the visualization into a course management system so that instructors can monitor how their students are using the visualization. If such systems are to be provided for instructors in ways that are reliable and convenient, it will be important for these systems and visualizations produced by them to have an existence beyond an amorphous collection of applets that are found by Google when an instructor teaches one iteration of a course but then are gone six months later. One way of ensuring this reliability is to develop effective deployment methods for keeping instructors and their students in touch with the most recent versions of systems. Java Web Start methodology is one way to solve the deployment problem, and CVS repositories can be used to build collections of visualizations that many visualization designers and users will have a vested interest in reviewing and maintaining. JHAVÉ *serves as a successful example of this approach.*

2. Susan Rodger

Susan Rodger is Associate Professor of the Practice in the Department of Computer Science at Duke University. She has been teaching computer science with visualizations for sixteen years. She has developed software tools for animation and computer science concepts, including the well-known JFLAP animation software for teaching the automata theory course, and JAWAA, software for creating animations on the Web. She is a co-author of the book *JFLAP—An Interactive Formal Languages and Automata Package,* to be published in 2006. She is currently the PI of the NSF DUE CCLI grant 0442513 entitled "An Interactive Approach to Formal Languages and Automata with JFLAP." This project can be found at www.jflap.org.

To be effective, animations must be integrated into the fabric of a course. One successful approach to teaching computer science courses is to incorporate animation software into an interactive lecture format in which students interact with other students and the instructor during lecture. In one example, such a lecture consists of a series of mini-lectures. Each mini-lecture has three phases. First, the instructor introduces a new concept with the animation software; second, students work on a related problem in small groups either with the animation software or on paper; and third, the problem is discussed and solved by the class as a whole with the instructor using the animation software. This approach has been used in a variety of courses from introductory programming courses to the automata theory course. In integrating animation software into a course, supporting materials—such as tutorials and a user manual with a variety of exercises—and possibly even modifications to the software may be needed. For problem solving it is helpful to have examples already built for the animation software that students can load and experiment with, before they attempt to build their own animations. The software should also have capabilities for saving files for electronic submission and support for grading. JFLAP serves as an example of this approach.

3. Guido Rößling

Guido Rößling is a faculty member at Darmstadt University of Technology, Germany. A relative newcomer to the field, Guido has quickly established an international reputation as a result of his tireless work with the animation and visualization community. His most notable contribution, called ANIMAL, can be found at http://www.animal.ahrgr.de/index.php3,

Instructors and students who wish to learn better by designing effective animations—in contrast to just viewing them—need to be encouraged to create animations of concepts, as opposed to just interacting with already-created instances. Powerful tools are needed to make the animation or visualization creation task as easy as possible. Such tools must be well-documented, well-supported, well-grounded in known animation and visualization techniques, and be straightforward to learn and use. ANIMAL is one such system that continues to evolve as more is learned about the effectiveness of animations and visualizations on student learning; it will serve as an example.

4. Rocky Ross

Rocky Ross is Professor of Computer Science at Montana State University. He has been working in the area of computer science animation and visualization since the late 1970s. His current work is in producing hypertextbooks for the Web. Check www.cs.montana.edu/webworks/projects/theoryportal for more information. Some of his most recent work was funded by NSF grants 0088728 and 0089397.

One way to ensure that instructors are encouraged to use animation and visualization software in their courses, and also to ensure that students actively engage with this software, is to incorporate it into the primary teaching and learning resource for a course. Called hypertextbooks, *such resources interweave standard textual presentations with arbitrarily many images, audio explanations, slide shows with narration, narrated video clips, and, most importantly, active learning applets (animations and visualizations) of the primary concepts of the course. Hypertextbooks can be constructed with different paths through the material for students at different academic levels and learning styles. They are also intended to be minimally dependent on plug-ins and to be accessible through standard web browsers, making them available to virtually anyone virtually anywhere at minimal cost. Hypertextbooks are particularly well-suited for instructors who have no time to develop animations or visualizations on their own. Hypertextbooks are under construction for the theory of computing and for the non-computing topic of biofilms. The tools used in their construction will be presented as examples of this approach to incorporating animation and visualization into the curriculum*