Engaging Students with Active Learning Resources: Hypertextbooks for the Web[1]

Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, Rockford J. Ross
CS Department, Montana State University, Bozeman, MT 59717
ross@cs.montana.edu

The World Wide Web has mesmerized educators for over a decade now with its tacit promise of platform independent, universal educational resource delivery. Indeed, many useful and exciting educational tools and distance learning courseware have been developed for the Web. On the other hand, some of the really fascinating educational resources—envisioned some years ago—remain elusive. One of these is the active learning hypertextbook. In this paper we discuss the concept of an active learning hypertextbook, pointing out the state of the art, the technical problems to be surmounted, and our own research and development leading to hypertextbook resources in the Webworks Laboratory at Montana State University.

What is an Active Learning Hypertextbook?

A *hypertextbook* is a computer-based teaching and learning resource that either augments or takes the place of a traditional textbook in a course. The "hyper" aspect of such textbooks means that they extend "above" or "beyond" traditional textbooks in their ability to help a student learn. In their most rudimentary incarnations, hypertextbooks may simply make effective use of hyperlinks that allow the reader to branch to related portions of the text by clicking a mouse button on a section of highlighted text—an action now second nature to all users of the Web. More advanced applications may include sound, pictures, video, and/or animated presentations of concepts (e.g., algorithm animations).

An *active learning hypertextbook* will include most of the features mentioned in the previous paragraph, and will additionally incorporate interactive software modules—usually in the form of applets—that actively engage students in the learning experience.

[1] Support for some of the work described here has come in part from the National Science Foundation, grant numbers DUE-9752673 and DUE-0088728.

State of the Art

The use of hypertext links to tie learning material together in innovative ways is old hat by now, and even predates the Web. Over time, however, hypertext learning materials have become more and more sophisticated. Recent works have gone beyond mere hypertext linking of the material to include animations of key concepts to help students learn. One of the first large-scale examples of this kind of resource was the CD that was designed to accompany the *Algorithms* book by Cormen, Leiserson, and Rivest [6]. This system included animations of some of the algorithms in the textbook, but was only available for Macintosh computers. More recently, an entire textbook has been published on CD and is being disseminated without a companion traditional hardcopy book[2] [7]. This book-on-a-CD is a very interesting experiment in hypertextbook development that includes voice descriptions as well as text and provides dynamic views of program construction. It is not platform independent, and it does not incorporate elaborate active learning modules as an integral part of the learning environment.

A distinction should be made here between *hypertextbook projects* and *visualization* or *animation projects*. Visualization and animation projects are generally standalone efforts intended to produce software systems that allow students to learn by viewing models of concepts on a computer screen. A visualization of a concept may be static—for example, a single picture of a finite state automaton. When animation is added to a visualization the student can watch the concept in action. For example, a finite state automaton may be shown changing states as it reads individual symbols from the input tape, until the input is consumed and the automaton ends in either an accept or reject state. Further, when active learning is included, a student can interact with the animation. For instance, when working with an animation of a finite state automaton, a student may be able to change inputs to the finite state automaton, modify the finite state automaton, or

[2] It is interesting to notice the progression from traditional textbooks, to textbooks that were incomplete without the inclusion of a supplemental CD attached to the back cover, to a standalone CD that is the book itself, with no hardcopy as a crutch.

create entirely new automata from within the animation system.

Most visualization projects in computer science education involve animation. A great number of animation systems for individual topics have been created, with most of the activity centered around algorithm animation. Among those are some developed by faculty whose long-time research interests have been in visualization and animation for computer science education—Tom Naps, Susan Rodger, John Stasko, and others. Space constraints rule out citations here, but links to these projects can be found at the Webworks Web site [8].

Hypertextbook projects are the next logical extension of animation and visualization projects. The concept is simple to explain (but a challenge to implement): Create hypetextbooks that incorporate standard text accompanied by sound, active learning animations of key concepts, and hyperlinked presentations of the material catering to different learning needs into a single, self-sufficient resource.

There are few hypertextbook projects of this scope underway, although more are sure to start. The fourth author of this paper was a member of the program committee for the first Program Visualization Workshop[3] held in the summer of 2000 in Porvoo, Finland [2]. There, the work of a number of researchers from different countries was presented, most of which represented standalone visualization and animation systems.

In the concluding presentation of the conference this author encouraged the participants to begin the effort of refocusing research and development efforts from standalone animation systems to complete, integrated learning resources. The results of this workshop were summarized and presented in a panel discussion at the ITiCSE 2000 conference in Helsinki [1].

Points to Ponder

The Program Visualization Workshop opened discussion to a number of issues regarding the use of visualization and animation software in the classroom. The following main points were noted:

- Many very good animation and visualization systems for computer science education have been developed.
- Most of these systems languish, being used in few institutions beyond the ones where the systems were developed.

This lack of use has been a puzzling question to many of the researchers and developers in the field. Our own view of the matter in relationship to our research efforts at MSU are stated in the following observations and conclusions, which we believe anyone working in this field would be wise to take to heart:

[3] There is a move afoot to change the name of future workshops, as "Program Visualization" does not seem to capture the subject of the workshop, which is the use of visualization tools in active learning computer science education.

Observation 1 — *downloading, installation, and maintenance issues:* Regardless of how good an educational software system is, if it requires any download, installation, and maintenance efforts, it will see only minimal use. Computer science instructors almost everywhere are overworked because of a shortage of faculty and large enrollments, and they simply do not have the time—even when they have the desire—for any added effort.

Conclusion 1: For computer-based educational materials to receive widespread use, they must be turnkey systems that require no (or very minimal) startup efforts to use.

Observation 2 — *platform dependence:* Platform dependence has always been a deterrent to the widespread adoption of visualization systems for computer science education. Some early systems worked only with certain computer types, operating systems, graphics packages (e.g., X-windows), and monitors making them virtually unusable in most settings outside their home departments. Even later systems that have worked on PCs but not Macs, or vice versa, have not been well received. This problem is exacerbated by the fact that most students today have their own (wide variety) of computers and operating systems, and should be able to run animation software used in class at home on their own as they study.

Conclusion 2: If a visualization or animation system for education is to be widely used, it must, as far as possible, be platform independent.

Observation 3 — *single concept animations versus integrated learning resources:* If animations of concepts can only be included in a course by incorporating them in bits and pieces from a wide variety of sources, they are less likely to be used. This again goes back to the problem of overworked faculty, who would be called upon to find the sources of such animations, be sure that links are still valid (for Web-based resources), and introduce these animation systems at appropriate times in the course.

Conclusion 3: To ensure widespread use of animation systems for education, the animations must be part of a comprehensive and cohesive package that becomes the primary teaching and learning resource for a class.

Web based hypertextbooks address all of these concerns. The use of the Web is second nature to virtually all instructors and all students entering higher education today (if it isn't, it is a skill that needs to be acquired in any case). Thus, there are virtually no startup costs to using a well-designed Web-based hypertextbook. It is either available on the Web, or, more likely, on a CD. The student just needs to pop the CD into the drive on the computer, and access the hypertextbook on the CD through a standard browser. Questions about platform dependence disappear. And hypertextbooks can be the major supporting educational resource for an entire class, not just a resource that is used for a few demonstrations along the way.

A Theory of Computing Hypertextbook for the Web

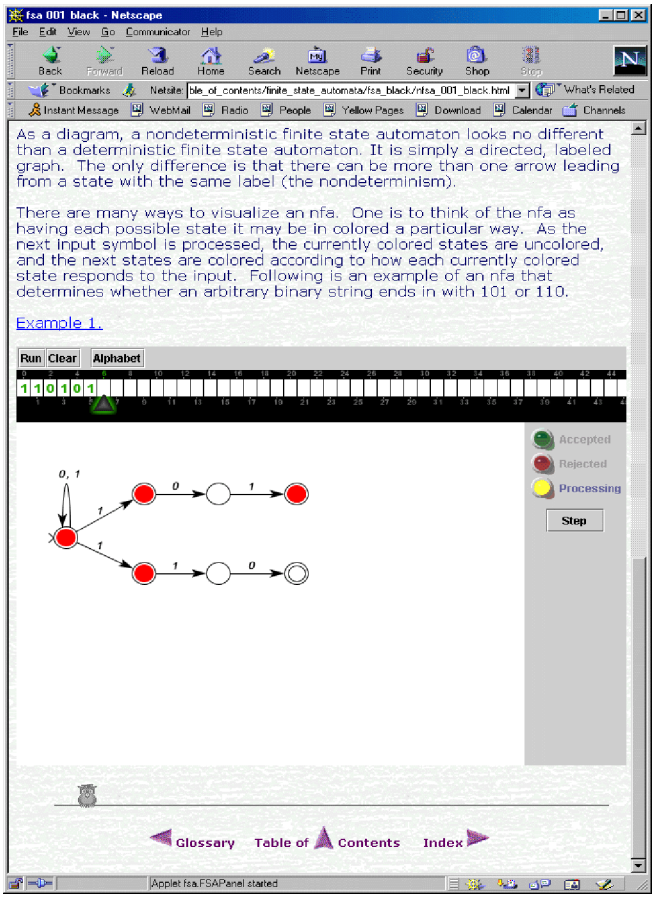
Our longtime research in animation for computer science education at MSU has led us to begin construction of a hypertextbook of the kind discussed above and first reported in [3]. The topic we chose was the theory of computing. There were many compelling reasons for this choice.

- The theory of computing is perhaps the most challenging topic computer science students need to learn, and therefore can perhaps benefit most from well-designed educational resources.
- The theory is replete with concepts that demand visualization. The various models of computation—from finite state automata to Turing machines—which are normally presented at a whiteboard through multiple diagrams, are natural candidates for software animation, for example. The same is true for derivations in a grammar, various conversion algorithms (e.g., from a nondeterministic finite state automaton to an equivalent deterministic finite state automaton), problem reductions required in NP-completeness presentations, and many other aspects of the theory.
- The theory compartmentalizes nicely, allowing hypertextbook modules to be constructed and used incrementally. For instance, a module on finite state automata would be of use in many different courses, including an introduction to computer science, the theory of computing, and compilers.
- It sounded like a fun topic to tackle!

Known as *Snapshots of the Theory of Computing*,[4] the hypertextbook includes a number of the features described earlier. Most notable are active learning animation modules that are tightly integrated with the text. The accompanying figure illustrates this, albeit in rather unsatisfactory form. Without the color, dynamics, sound, and interactivity of the animation module much is left to the imagination (readers are encouraged to try the hypertextbook themselves at www.cs.montana.edu/webworks). Still, the idea can be described.

The following picture shows a standard Netscape browser window. One of the key objectives of the *Snapshots* project was that it be platform independent and universally available. Thus, it works under all browsers and on all platforms that include the most up-to-date releases of the Java environment. The top part of the window includes text describing features of nondeterministic finite state automata. At the bottom of the window is an example of a nondeterministic finite state automaton. Here, similarities with standard textbooks ends. The example nondeterministic finite state automaton is not a simple illustration of a finite state automaton, but is really an applet (the work of the third author) that can be operated by the reader.

[4] So named because it is being constructed incrementally, with various snapshots of the theory, such as finite state automata, tackled one at a time.



This snapshot shows the overall configuration of the computation of the nondeterministic finite state automaton after the input string 110101 has been completely processed. The shaded states are those that the automaton is in (nondeterministically) at this point. The input string is accepted, because one of those states is an accepting state.

This finite state automaton applet has a number of features.

- An arbitrary input string can be typed by the student on the input tape at the top of the applet.
- Execution can be set to run a step at a time under student control, or it can be allowed to progress automatically without student intervention.
- Input symbol consumption is shown by an input head that moves across the tape, one cell at a time.
- State changes are shown by colored disks that move from current states to the next states as each input symbol is consumed; nondeterminism is illustrated in that the disks in the currently colored states all move simultaneously to the next states. If a particular (input, state) pair does not lead to a next state, the color of the disk in that state is changed and then the disk disappears.
- The initial finite state automaton presented in an instance of the applet can be modified by the student:

states and transitions can be added and deleted at will. Indeed the entire automaton can be cleared and a new one constructed by way of simple mouse clicks and drags.

- Sound effects are used to highlight various actions of the automaton during execution.
- Since the applet can be inserted at arbitrary points in the hypertextbook, any desired number of examples of finite state automata can be presented with little added effort. In the spirit of the hypertextbook, a student can voluntarily choose whether to experiment with more exercises or not by following (or not following) special links.
- Behind the applet are a number of supporting routines, such as one which can determine whether a student-constructed finite state automaton is correct or not. This allows the applet to be used in exercises that are critiqued automatically as a further aid to student learning.

The Webworks research team has created and tested a number of other active learning animation applets for integration into the *Snapshots* hypertextbook.

A Program Animation Applet. A program animation applet for presenting virtually all important aspects of a program in execution has been adapted for the Web from earlier work by the Webworks team (primarily the work of the second author). The applet window has panes for:

- The program code.
- Program variables.
- Input and output.
- Execution cost (for time complexity analysis).
- Animation controls.

Supporting the applet is a virtual machine[5] that is capable of executing its virtual machine code both forwards and in reverse. Under student control, the program can be executed a step at a time (or left to run automatically) as relevant portions of code are highlighted, variable value changes are noted, input and output recorded, and the underlying cost of executing the program is updated. The animator supports active learning by allowing students to input values to the program on the fly. At puzzling junctures the student can reverse execution arbitrarily far and start through a section of code again (thanks to the reversing capabilities of the underlying virtual machine).

The programming language supported by the animator is Pascal[6]. Although Pascal is dated as an introductory teaching language, it now serves marvelously as a pseudolanguage for teaching and learning about algorithms. A version of this animator for Java is under development by the first author.

Examples abound where this animation applet will prove useful in the *Snapshots* hypertextbook. It can be used to show how a finite state automaton can be implemented as a program, to explain how certain algorithms (such as the conversion of a finite state automaton to its minimal equivalent) work, and for demonstrating through experimentation what the time complexity of various machine models is.

A Context-Free Grammar Animation Applet. An applet that allows a student to expand the rules of a supplied context free grammar as a parse tree is constructed or to build and experiment with new grammars is available. The animator provides switches to enforce leftmost or rightmost derivations if so desired. This animator has been used in introductory computer science courses, compiler courses, and theory courses at MSU.

An LL(1) Table Construction Applet. An applet that demonstrates the construction of the First sets, the Follow sets, the Predict sets, and the LL(1) table for arbitrary context free grammars is nearly finished. As with all of the other applets so far developed in the Webworks Laboratory, this applet has also been designed for active learning. Students can supply a grammar, modify an existing grammar, and construct First, Follow, and Predict sets, as well as an LL(1) table, to be checked for accuracy by the applet against the given grammar. The applet is designed specifically to help students learn the rather complex task of LL(1) table construction for parsing.

Many more applets than the ones described above are planned. Indeed, it is unlikely that the *Snapshots* hypertextbook project will ever truly be finished. There will always be enhancements and new active learning animation modules that will make *Snapshots* even more useful with each new release (which can be made quite frequently).

A number of other features are included in or planned for *Snapshots*. At present, different learning needs are addressed through an organization that resembles a Montana ski resort. There are ways through the material aimed at beginners (the "easy way down") and marked with green circles. These ways through include lots of examples and a more detailed, intuitive introduction to each new topic. Blue squares mark the way for intermediate learners who already have some background in the material and more mathematical maturity, and who need fewer examples and less in the way of intuitive introduction. For experts, the way through the material is marked with black diamonds. Only a few examples are given, and new topics are introduced in the traditional "definition, theorem, proof" style of advanced textbooks. *Snapshots* can thus be used for classes at all levels. It is anticipated, too, that after beginning students have been through the material once, they will find that reviewing the material is more easily done at one of the more challenging levels.

Planned enhancements include the incorporation of voice versions of the material. Students will be able to listen to an explanation as well as read it.

[5] It is easy to forget that the concept of a virtual machine is as old as the discipline of computing; this one predates the Java Virtual Machine by many years.

[6] This shows the age of the project; Pascal was the teaching language of choice when we first started working on animation for education.

What Have we Learned?

We have learned many things as we have continued to work on the *Snapshots* project. The most important, perhaps, is that it requires a lot of careful work. The design and development of useful, active learning applets for animating interesting computer science concepts is laborious, albeit quite fun. A good animation together with its testing, evaluation, and publication is certainly worth a Master's thesis. A PhD thesis can be gleaned from an animation system that encompasses an entire subarea of computer science in which new ground is broken in the development of the methods (e.g., compiling for reverse execution and animation) or in a careful evaluation of the effects of the project on student learning.

We have also learned by trial and error which things work and which don't in the construction of a hypertextbook. A simple example is avoiding the use of pop up windows wherever possible. Animations are more effective when included directly in line with the rest of the presentation. Pop up windows are distracting, and students can easily lose track of where they are in the ensuing confusion. A very helpful talk discussing the many nuances of effective computer presentation for learning was given by cognitive scientist Pertti Saariluoma [9] at the aforementioned Program Visualization Workshop.

Hurdles

All is not perfect on the World Wide Web. The technology still promises more than it provides. The two most pressing obstacles to the immediate completion of a work like the *Snapshots* hypertextbook are (1) the lack of a mathematical presentation markup language and (2) the capability to easily save files from an applet. That both of these problems are still unsolved we find to be somewhat surprising, because both problems have been worked on for a number of years.

The problem of saving files has to do, of course, with security. One doesn't want a rogue applet to be able to destroy a hard disk. On the other hand, there have been promises made for years that this security issue would soon be solved. Unfortunately, the problem affects a hypertextbook like *Snapshots* in that it precludes students from saving objects (e.g., a finite state automaton) that they have constructed from within an applet for later modification or for electronic submission to an instructor. At the time of this writing, the only way to save files from an applet is to set special security waivers in the browsers, which hardly makes for a turnkey system.

The lack of a markup language for mathematics is also particularly troublesome in a hypertextbook like *Snapshots*, which involves mathematical notation at every turn. MathML [5] is the obvious solution, but browsers are not quite ready to display MathML, and good editors are only now being readied for writing in MathML. (In both cases we blame the market for the current situation. There is little money to be made in either case; otherwise both problems would long since have been solved.)

Summary

The discussion of hypertextbooks could go on forever (and, indeed, probably will). There are so many avenues to explore to enhance student learning that simply are not possible with traditional textbooks. Perhaps the most intriguing at present is the possibility that a textbook could be made to adapt to a student's learning needs and styles as the student uses the textbook (see, for example, the work of Peter Brusilovsky [4]). There is also a need to conduct formal evaluations of the effectiveness of hypertextbook use in learning environments.

In spite of the arduous labor and the problems yet to be solved, hypertextbook development is an exciting and fun challenge that engages both students and faculty in many very interesting projects. The work we do in hypertextbook development in computer science will form the basis for similar work in other disciplines.

References

[1] ITiCSE 2000. *ITiCSE 2000 Conference*. http://www.cs.helsinki.fi/events/iticse/.

[2] PVM 2000. *Program Visualization Workshop*. http://cs.joensuu.fi/pages/pvw/workshop.htm.

[3] Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, Jessica L. Lambert, and Rockford J. Ross. Tying it All Together Creating Self-Contained, Animated Interactive, Web-Based Resources for Computer Science Education. In *Thirtieth SIGCSE Technical Symposium on Computer Science Education (SIGCSE Bulletin)*, volume 31, number 1, pages 7–11, March 1999.

[4] Peter Brusilovsky. *Brusilovsky's home page*. http://www2.sis.pitt.edu/∼peterb/.

[5] World Wide Web Consortium. *MathML*. http://www.w3c.org.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990. An accompanying hypertextbook CD is available.

[7] David Gries and Paul Gries. *ProgramLive: A Multimedia Java Learning Resource*. Data Description, Inc., 2000. Java hypertextbook on CD.

[8] Rockford J. Ross. *Webworks Laboratory Web Site*. http://www.cs.montana.edu/webworks.

[9] Pertti Saariluoma. *Image and interface: Some psychological aspects of visualisation*. http://cs.joensuu.fi/pages/pvw/saariluoma.htm.