Truly Interactive Textbooks for Computer Science Education

Clifford A. Shaffer*, Thomas L. Naps**, and Eric Fouh*
*Department of Computer Science
Virginia Tech

**Department of Computer Science
University of Wisconsin, Oshkosh

shaffer@cs.vt.edu, naps@uwosh.edu, efouh@vt.edu

1 Introduction

The dream of an electronic textbook has been actively pursued for at least two decades. Goals include (i) improving exposition through a richer collection of technologies than are available through print textbooks, and (ii) increase student engagement with the material, in order to get them to learn at a higher level in Bloom's taxonomy (Naps et al., 2002). Instead of merely viewing material, we can hope to use frequent assessment (by asking questions) to get them responding, and through interactive activities get them to change and construct virtual artifacts. We will use the term *hypertextbook* to refer to an electronic textbook that integrates interactive exercises and assessment. See (Rössling et al., 2006; Ross and Grinder, 2002) for background on efforts to define and implement the hypertextbook.

In this paper, we discuss our plans to create a hypertextbook for a complete semester course in Data Structures at the Sophomore level. Data Structures and Algorithms as a topic can particularly benefit from the use of advanced technology to aid explanation of the dynamic processes that make up the essence of an algorithm, and which can be difficult to convey using words and images. Therefore, our particular focus is on the use of algorithm visualization (Shaffer et al., 2010; Naps et al., 2002) as a means both to deliver the necessary dynamic exposition, and to increase student interaction with the material. We will discuss how research results indicating the value of AVs combined with a lack of progress in uptake of AVs in actual courses leads us to the conclusion that a complete semester-long course package is the right way to go. We describe our plans for implementation, including a discussion of relevant technology for the project.

2 Motivation

Periodic surveys on both instructor interest in AVs and their level of use have been conducted among attendees of education conferences and listservs for more than a decade. Naps et al. (2002) report on three surveys conducted in 2000. Collectively, they indicate a strong positive view in favor of AV use (over 90%). However, only about 10% of respondents at that time indicated frequent use of AVs in data structures courses at their institutions, while about half to three quarters indicate occasional use of AVs in such classes.

At SIGCSE'10 (held in Milwaukee during March 2010) we conducted a new survey to determine instructor attitudes toward AVs and their use in the classroom. For details, see Shaffer et al. (2011). The survey was designed in the spirit of the 2000 surveys, and the findings are consistent with those earlier results. In 2010, 41 of 43 respondents agreed or strongly agreed that AVs can help learners learn computing science, with two neutral. However, just over half had used AV in a class within the past two years. It is not clear that all of the "yes" answers from the present survey refer to what we might consider to be "AVs." But they probably all refer to some sort of dynamic software visualization run on a computer, as was the case in the 2000 survey. We did not ask about frequency of use of AVs in class.

Our third survey question asked what respondents see as the greatest impediments to using AVs in courses. Roughly half the responses relate to finding suitable AVs to use. We can

hope that the presence of online resources such as the AlgoViz Portal (`http://algoviz.org`) will reduce this information gap, by making it easy for instructors to locate quality AVs. However, about half of the responses relate to issues with integrating AVs into courses. These results are roughly the same as reported in Naps et al. (2002, 2003) and Rössling et al. (2006). Such issues are much harder to deal with, and are representative of well-known problems for adoption of educational technology in general (Hew and Brush, 2007). While it is easy to give students pointers to AVs as supplemental material, it is much harder to integrate AVs into lectures, labs, and assignments. Respondants cite lack of time to make course changes, lack of time within the course to spend additional time on a given topic, and inconsistencies between the course textbook and the AV.

We hope that community support provided by the AlgoViz portal can help. The model embraced by AlgoViz is to allow the community to add value beyond information embodied in a simple catalog of AVs. This value comes as a byproduct of direct communication between community members. While forums are one obvious method for this, there is a deeper communication involving community ratings and recommendations for content entries and sharing experiences on how to make the best use of content. A major concern for instructors is deciding whether a given educational resource is of good quality, and how to use it. Therefore, user feedback on resources is as important as the resources themselves. AlgoViz provides the *field report*, which gives a convenient way for instructors who have used an AV in a class to share their experiences. Field reports are designed to supplement the evaluation data in the AV Catalog, since AV ratings in the catalog are not necessarily based on real-world classroom experiences. A field report can thus provide empirical evidence to strengthen or qualify a recommendation.

The survey results lead us to the conclusion that it is easier to integrate a complete block of instruction (either a complete topic or even a semester course) than it is to fit a piece of instruction such as an AV into existing presentation on that topic. Instructors are used to the concept of adopting a new textbook for new courses that they teach, and often welcome lecture notes and other class support artifacts. Even for courses taught previously, instructors will adopt new textbooks and new presentations for various topics in the course. A key aspect is that adopting a new chunk of content allows the instructor to completely replace or populate a block of course time, as opposed to squeezing new content or presentation techniques on top of existing material. In the past, most AVs developers have implicitly taken the approach that their AV will be integrated into existing presentations. A typical example might be an AV presenting a Quicksort algorithm, with no tutorial explanation of the algorithm. The idea seems to be that this visualization can be slipped into lecture or used by students to supplement the textbook for self-study. But this approach has led to the problems noted above. In contrast, a complete unit of instruction (including AVs) can more easily replace an unsatisfactory existing presentation of the topic.

Important technological factors contribute to making AVs easier to use in the classroom than in prior years. More AVs are available than ever before (Shaffer et al., 2010), backed up by improved research studies and improved access both through general Internet search and at the AlgoViz Portal. Increased access to the Internet by students and instructors, both in and out of the classroom, makes AV use more practical. For example, at Virginia Tech, while it has been "possible" to project Internet material from a computer in class for over a decade, it has only been in the past couple of years that such access has become both ubiquitous and reliable in all of our classrooms. This makes a huge difference in the confidence of mainstream instructors for using such technology, in contrast to early adopters (Hew and Brush, 2007).

Another potential factor in favor of the use of AVs and hypertextbooks is ubiquitous availability of laptops and mobile devices. For example, all Engineering majors at Virginia Tech are required to own a tablet PC, and most also have mobile devices including smartphones, ebook readers, or iPads. However, there is a downside to the non-PC devices, in that they have various technology limits on how eTextbook content can be provided. So while there

is ubiquitous access to the Internet among our target audience, there are still limits. For example, Java Applets cannot be displayed on most such devices.

3 Prior Work

The bulk of effort by CS educators involved with interactive AVs has focused on development of the AV technologies themselves and at best small, focused hypertextbook modules that incorporate AVs into a particular topic or small set of topics within a course. For example, Grinder et al. (2002) report on a set of web pages integrated with applets that were used in parts of a theory of computation course. Many instructors who teach theory now use Susan Rodger's JFLAP (Gramond and Rodger, 1999) throughout much of their course and Rodger has published a hard copy guide to using JFLAP (Rodger and Finley, 2006). In the preface Rodger warns the reader "our book assumes that the reader has read briefly about these topics first in an automata theory textbook or a compiler textbook".

Ross (2008) describes efforts at building a Perl-based infrastructure for creating hypertextbooks. The infrastructure relies on the Dreamweaver toolkit. Only a few chapters (three out of seven) of a theory of computing and a biology book has been produced using this technology. The system requirements for these hypertextbooks also hinder their wide adoption due to browser restrictions, use of Java applets, and specific screen resolutions.

Rößling and Vellaramkalayil (2009) report on a technology that integrates AV into Moodle-based lessons, but the emphasis of that report was again on a technology that would support visualization-based hypertextbooks in Moodle, and to our knowledge little progress has been made on the actual writing of such a textbook.

Titterton et al. (2010) have used a lab-centric mode of instruction for introductory CS courses at UC-Berkeley. Their current work is being done in Moodle and uses a small amount of AV along with many other "check point" exercises that students must complete as they progress through a set of material presented in Moodle. It is built upon an earlier technology called UC-WISE that was developed and used exclusively at UC-Berkeley. Their materials are designed for introductory CS courses that aims mostly at developing students' programming skills. Hence they make only limited use of AVs. Alharbi et al. (2010) are using eXe, an open source authoring system for academic-related web content using XML and HTML (`http://www.exelearning.org/`). Their efforts intend to help teaching Operating Systems using visualization. Learners can interact with the AVs, and can take quizzes and tests online. They used the Sharable Content Object Reference Model (SCORM: `http://www.adlnet.gov/`) to support integration of digital teaching materials with a CMS (Moodle in their case).

Another effort to integrate AVs into hypertext is being performed by Karavirta (2009). His solution is based on HTML and JavaScript, allowing the hypertextbook to be viewed in any browser without additional plug ins. The learner can interact with the animation and draw annotations on it, but the current system does not store the annotation, nor does it support quizzes and tests. Their earlier work on TRAKLA2 (Malmi and Korhonen, 2008) includes a tutorial on heaps that is integrated with AVs. Animal (Rößling et al., 2000) and JHAVÉ (Naps, 2005) are AV development systems that include tutorial descriptions for some single AVs. Crescenzi and Nocentini (2007) takes the novel approach of a traditional textbook whose examples and illustrations are closely tied to the AlViE AV system.

Largescale use of educational hypermedia in South Korea. provides interesting feedback about the challenges of hypertextbooks. Kim and Jung (2010) identify usability, portability, interactivity, and feedback as major elements to consider while designing such systems. Learners should be able to ask questions and receive help, as well as control, manipulate, search and browse hypertextbook content. They also advocate for the development of models to support group collaboration.

There exists no complete electronic textbook, tightly integrated with AVs, that could be used as the primary learning resource in a semester-long computer science course. This is

perhaps surprising because Marc Brown's groundbreaking dissertation on AV ((Brown, 1988)) issued the following caution:

> Much of the success of the BALSA system at Brown [at the time Brown's thesis was written] is due to the tight integration of its development with the development of a textbook and curriculum for a particular course. BALSA was more than a resource for that course – the course was rendered in software in the BALSA system.

Why have CS educators not heeded Brown and authored hypertextbooks? We suspect the answer is something known to anyone who has written a textbook or an AV: it consumes huge amounts of time. While writing a textbook is a big job, writing and associated set of AVs and the assessment support is a far bigger job yet.

4 A Case Study

In this section we briefly describe an online hashing tutorial and a study conducted to determine its pedagogical efficacy. The results have helped us to make progress toward defining the requirements for a more comprehensive electronic textbook. Note that this approach is similar in spirit to the TRAKLA2 heaps tutorial (`http://svg.cs.hut.fi/heaptutorial`).

In 2008 and repeated again in 2009, students in separate sections of a sophomore level course on data structures and algorithms at Virginia Tech were taught about hashing. One section was given standard lecture and textbook for one week, equivalent to what had been done previously in the class. The other section spent the class time working through an online tutorial combined with algorithm visualizations to present the same material. The tutorial used text content taken from the course textbook, so that it was an exact match to the material being presented in the control section. However, the online tutorial heavily supplemented this text with algorithm visualizations. The tutorial can be found online at `http://research.cs.vt.edu/AVresearch/hashing`.

In each of the trials, the two sections were given a quiz on hashing at the conclusion of the week of instruction. The results were positive: An analysis of variance shows a significant difference between the two treatment groups ($F(1, 118) = 4.37$, $p < 0.05$), with students completing the tutorial averaging higher quiz scores than those who were given the lecture and textbook content. However, the difference in the means is 6.2 percentage points, or approximately one third of one standard deviation. These results indicate that not only can an online tutorial be as effective as lecture (which has important implications for distance learning), but that providing proper interactivity allows computerized instruction to be better than lecture-based (passive) instruction. However, there is at least one major consideration that might influence the results of this particular study: How much impact did the controlling structure of coming to class and doing the tutorial in "lab" setting have on the results? The outcome could be quite different for a student just reading the material and working through the visualizations on their own, where self-discipline might well not be sufficient to provide the necessary amount of time and attention. Likewise, the controlled environment of attending lecture before reading the textbook on one's own is also likely to have a major difference compared to just reading the book on one's own.

We hypothesize that, if we want to have a viable self-contained electronic textbook that supports self-study of the material, the system needs to build in some equivalent to the controlled pacing and feedback that is encountered when attending classes or labs. This means that assessment and an objective measure of "progress" through the material needs to be an integral part of the system (whether purely self assessment or with results submitted to an instructor). This means a much tighter integration of material, interactive exercises, and assessment than is provided by a system as simple as the Hashing Tutorial.

5 Implementation Principles for an Electronic Textbook

Based on our interpretation of the results of the 2010 survey (Shaffer et al., 2011), our experiences with the Virginia Tech Hashing Tutorial, and the continuous improvements in online technologies that we observe, we think that the time is ripe to create an entire online textbook for an undergraduate course on Data Structures and Algorithms. Depending on the exact topics to be covered, this could be beneficial at any level from CS2 through senior algorithms.

When considering the full breadth of content that would be contained in a complete course textbook on the subject, we conclude that three distinct forms of content presentation are desirable. First, no matter how dynamic and interactive the topic, text and images as are now found in typical textbooks continue to have their place as part of the exposition. Some content simply is not visual or dynamic and so is efficiently transferred via words and images.

Second, some content is essentially expository (i.e., at that point in the presentation there is no need for student constructive interaction), but the content is about dynamic processes or conducive to visual presentation. This includes most algorithm descriptions, such as how a particular sorting algorithm works. Since initial presentation does not involve exploration or decision making, or demonstration of proficiency, the prime concern is what techniques provide the clearest explanation. This could best be handled by a presentation that relies heavily on diagrams and simple animation, with pacing controlled by the the reader. In essence, an animated slide show is adequate for such presentations.

Third, there many instances that from student interaction. This includes things as simple as probing a calculation, (e.g., trying different inputs to a simple simulation or calculation). An example is the famous Birthday Problem: How many people need to be in a room before the odds are greater than even that two share a birthday? Another need for interaction comes with demonstrating proficiency with an algorithm, such as the interactive exercises for tree insertions that are part of TRAKLA2 (Malmi and Korhonen, 2008; Malmi et al., 2004). These are best handled by something like a Java Applet to support user interaction and dynamic on-the-fly calculations/processing of the algorithm. Performance comparisons are also of this nature, where the student is invited to define and run built-in simulations of multiple data structures or variants to see how they perform. Binding all of this together should be a steady stream of assessment activities to make sure that students stay "on track" and to keep them engaged even during otherwise passive exposition. This can be done with simple pop-up questions (that might or might not require a successful response to continue) and end-of-section quizzes whose success might be required to demonstrate competence needed to continue to the next section.

As we progress from the first to the third of these presentation approaches, the development cost goes up greatly. Text and images can be developed relatively quickly. In contrast, it took several student-years of effort (and over two actual years) to develop the Hashing Tutorial, mainly due to the effort involved in developing a handful of Java applets. A properly animated slideshow takes longer to create than equivalent text and images, but should be significantly faster to implement than a fully interactive exercise.

6 Technical Considerations

A number of technologies are available for developing the three presentation types for the electronic textbook as was envisioned in the previous section. Text and images can certainly be done with any traditional web development tool. The second component, which we characterize as an animated slide show, can be done (as the characterization suggests) using a variety of presentation tools such as MicroSoft Powerpoint, LaTeX's Beamer package, OpenOffice Impress, or Apple Keynote. However, while presentations can be created in these tools, the resulting presentations cannot so easily be integrated with the rest of the electronic textbook. Browsers can support some of these tools as plugins, but not universally across a range of

devices. The presentations can generally be converted to PDF format, but only Adobe's Reader can actually display the animations (at least, evince and xpdf, popular alternative PDF readers do not). Nor do the PDFs well integrate with non-PDF portions of the whole.

Flash is another popular tool for developing animations, and is rich enough to support the interactive aspects of the third component of presentation as well. However, Flash requires a plugin in most browsers, and so is not compatible with devices such as the iPad.

One technology that appears to be robust enough to implement all desired dynamic and interactive components is HTML5 incorporating JavaScript. HTML5 integrates its dynamic components well with standard text and images, and easily ports between PC browsers and mobile devices. Potential concerns include ease of use for content developers (as compared to, for example, PowerPoint when developing animated slide shows), and level of penetration of the necessary browser technology. Given that alternatives such as Flash are at least as problematic in terms of a typical user having access (since they require plugins), the problem of penetration seems to be low and quickly receding, in that we can expect that college-level students tend to have access to moderately up-to-date browser technology. Thus, we currently advocate use of HTML5 technology for developing the dynamic components of the electronic textbook.

7 Integrated Assessment and Progress Monitoring

There is much evidence that AVs foster effective learning when presented in a way that forces the student to actively engage with the visualization instead of passively viewing it (Hundhausen et al., 2002; Naps et al., 2002). Additional engagement can be created by having the student respond to interactive questions. Although many AVs include questions, few do so in a way that allows an instructor to monitor their students' progress. More typically the student's interaction with the system produces immediate feedback to the student, but the assessment of that interaction is not recorded in a way that the instructor can access. Nor do the students' answers provide a persistent record for the student that a section has been mastered, or integrate with navigation through the content such as provided by an "intelligent tutor". Two AV systems that do support this sort of integrated assessment are TRAKLA2 (Malmi et al., 2004; Malmi and Korhonen, 2008) and JHAVÉ. (Naps, 2005).

Although the TRAKLA and JHAVÉ systems support online assessment of students' using the visualizations, they both do so in a unique, non-portable way. We seek an approach that can work with a variety of presentation as well as the electronic textbook structure itself (some but not all questions will come within AVs). One possibility is a decoupled approach to assessment, where the actual assessment process is done by following a link to a third-party site that provides the relevant series of questions. This might take place at the end of each section in the textbook. Among the factors that will have to be considered in developing this protocol are:

- Developing effective strategies for assessing student responses. Assessing multiple-choice, multiple-selection, and fill-in-the-blank questions is straightforward. But automated assessment of textual responses is clearly much more difficult.
- Support for richer types of activities specific to CS, such as small programming tasks evaluated by comparing output from the proposed solution to the answer key.
- How to represent test questions. There exist standardized question representations, such as the IMS Question and Test Interoperability specification (QTI: `http://www.imsglobal.org/question/qtiv1p2/imsqti_oviewv1p2.html`).
- How to store student responses and progress in a fashion that makes it easy for instructors to analyze their students' results.
- How to store assessments of students' work in a fashion that respects their privacy.
- How to "loosely couple" the assessment system with the electronic textbook. Because the visualizations watched by our learners will be launched out of the hypertextbook,

we will need to have a mechanism for recognizing the learner's identity so as to interact with the quiz/assessment database that might be stored on a different server.

8 Connexions and the Creative Commons

Our final consideration is the broader context in which development of an electronic textbook should take place. Such a project is a huge undertaking. As evidence of this (besides our experiences with the extraordinary amount of time that it takes to develop high-quality AVs), consider that there exist few examples of the type of artifact that we seek to create, as stated in Section 3. Ideally, a broader community can be encouraged to contribute to the project, much in the style of an open source software development effort. The authors have many collaborators within the broader algorithm visualization community, and ideally we could leverage these collaborations to develop the materials. Since we envision the materials to be distributed with a GPL or Creative Commons license, intellectual property rights will be less of an issue than if a commercial publisher were involved.

The Connexions Project (`http://cnx.org`) is presently the largest collection of online textbooks developed with a creative commons license model. Connexions is more than a collection of publicly available online textbooks. They have developed a "creative commons" infrastructure that makes it easy for authors to reuse and combine pieces of textbooks, or to make their own altered version of an existing textbook. We envision developing our project in such an infrastructure to support sharing and reuse of educational chunks. There are integration concerns related to, for example, HTML5 technology or other technology for developing dynamic presentations. Integration of assessment is also of concern, since Connexions has only recently begun developing support for assessment.

We envision a multistage process to develop the hypertextbook project. The first step is to devise a complete management plan and to define the development workflow within the chosen implementation infrastructure. Next is to define a detailed "storyboard" defining a detailed layout for the entire hypertextbook, with all of the text and detailed descriptions of all places where interactive activities or presentations are desired. Third is to then begin an open development process where submissions from interested parties are provided for specific activities called for in the Storyboard under a reviewing process. If developed in this way, the hypertextbook will become "owned by" the broader community of AV developers.

References

A. Alharbi, F. Henskens, and M. Hannaford. Integrated standard environment for the teaching and learning of operating systems algorithms using visualizations. In *5th International Multi-Conference on Computing in the Global Information Technology*, pages 205–208, 2010.

M.H. Brown. *Algorithm Animation.* MIT Press, Cambridge, Massachussets, 1988.

Pierluigi Crescenzi and Carlo Nocentini. Fully integrating algorithm visualization into a CS2 course: A two-year experience. In *Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 296–300, 2007.

E. Gramond and S.H. Rodger. Using JFLAP to interact with theorems in automata theory. *ACM SIGCSE Bulletin*, 31(1):336–340, 1999.

M.T. Grinder, S.B. Kim, T.L. Lutey, R.J. Ross, and K.F. Walsh. Loving to learn theory: Active learning modules for the theory of computing. In *Proceedings of the 33rd ACM SIGCSE Technical Symposium on Computer Science Education*, pages 371–375, 2002.

K. Hew and T. Brush. Integrating technology into K12 teaching and learning: current knowledge gaps and recommendations for future research. *Educational Technology Research and Development*, 55:223–252, 2007.

C.D. Hundhausen, S.A. Douglas, and J.T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13:259–290, June 2002.

V. Karavirta. Towards seamless merging of hypertext and algorithm animation. In *Proceedings of the 5th Program Visualization Workshop*, pages 105–114, 2009.

J.H.-Y. Kim and H.-Y. Jung. South Korean digital textbook project. *Computers in the Schools*, 27(3 & 4):247–265, 2010.

L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, and P. Silvasti. Visual algorithm simulation exercise system with automatic assessment: Trakla2. *Informatics in Education*, 3(2):267–288, September 2004.

L. Malmi and A. Korhonen. *Active Learning and Examination Methods in a Data Structures and Algorithms Course*, pages 210–227. Number 4821 in LNCS. Springer-Verlag, 2008.

T.L. Naps. Jhavé: Supporting algorithm visualization. *IEEE Computer Graphics and Applications*, 25:49 – 55, September 2005.

T.L. Naps, S. Cooper, and twelve more authors. Evaluating the educational impact of visualization. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 124–136, 2003.

T.L. Naps, G. Rössling, and nine more authors. Exploring the role of visualization and engagement in computer science education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, 2002.

S.H. Rodger and T.W. Finley. *JFLAP-an interactive formal languages and automata package.* Jones & Bartlett Learning, 2006.

R.J. Ross. Hypertextbooks and a hypertextbook authoring environment. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, page 133–137, Madrid, Spain, 2008. ACM.

R.J. Ross and M.T. Grinder. Hypertextbooks: Animated, active learning, comprehensive teaching and learning resources for the web. In S. Diehl, editor, *Software Visualization*, pages 269–284. Springer, 2002.

G. Rössling, T. Naps, and nine more authors. Merging interactive visualizations with hypertextbooks and course management. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 166–181, 2006.

G. Rößling and T. Vellaramkalayil. First steps towards a visualization-based computer science hypertextbook as a Moodle module. In *Proceedings of the 5th Program Visualization Workshop*, pages 47 – 56, 2009.

Guido Rößling, Markus Schüer, and Bernd Freisleben. The ANIMAL algorithm animation tool. In *Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 37–40, 2000.

C.A. Shaffer, M. Akbar, A.J.D. Alon, M. Stewart, and S.H. Edwards. Getting algorithm visualizations into the classroom. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE'11)*, pages 129–134, 2011.

C.A. Shaffer, M.L. Cooper, A.J.D. Alon, M. Akbar, M. Stewart, S. Ponce, and S.H. Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, 10:1–22, August 2010.

N. Titterton, C.M. Lewis, and M.J. Clancy. Experiences with lab-centric instruction. *Computer Science Education*, 20(2):79–102, 2010.