WWW INTERACTIVE LEARNING ENVIRONMENTS FOR COMPUTER SCIENCE EDUCATION

David Carlson, Mark Guzdial, Colleen Kehoe, Viren Shah, John Stasko
Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
Email: dmc,guzdial,colleen,viren,stasko@cc.gatech.edu

ABSTRACT

The wide accessibility of the World Wide Web makes it a perfect base for developing computer science courseware modules. Since learning involves more than just receiving transmitted information, courseware must be interactive and encourage student engagement, which is a challenge on the Web architecture. This article describes an ongoing effort to develop World Wide Web-based computer science courseware modules that will use interactive components as integral parts of the material, in order to promote student involvement. It also discusses the proposed usage of new technology such as HotJava in this framework.

INTRODUCTION

The evolution of the World Wide Web (WWW) in this decade has led to a phenomenal increase in the ability to access hypermedia information. This has resulted in a number of systems [3, 1] that offer tutorial or educational material on a variety of topics. These documents, for the most part, do not emphasize the use of interactive components as fundamental pieces of the learning environment. Yet interactivity is a critical feature. Cognitive science has shown that the "transmission model" of learning is wrong – students do not learn by simply receiving information[2]. Rather, people learn *constructively*, which is to reflect on material, interact with it, and create an understanding[5]. Thus, for courseware to be effective on the WWW, it must allow and even encourage students to be engaged and reflect on interesting situations.

Computer science is an excellent domain for which to build interactive courseware because exploring the concepts and using the concepts can both be supported with software. For example, to explore concepts, we can use algorithm visualizations, provide sample applications to run, or demonstrate graphical routines with actual output. When students build software as part of their learning activity, we can provide intelligent guides to programming, libraries of sample programs and components for reuse, and forums for collaborative problem-solving. This kind of support provided to help students learn through doing is called *scaffolding*[11, 4].

Our group is in the process of building a WWW-based learning environment, consisting of modules and authoring software, to aid in computer science instruction. One of the foci of this effort is to go beyond the conventional multimedia systems by using *interactive learning* as an integral component of the educational process.

PROJECT GOALS

The primary goal of this effort is to provide a multimedia-based, scaffolded framework to assist in the instruction of computer science courses. The modules produced will support introductory, as well as intermediate and advanced computer science courses, e.g. computer graphics, user interfaces, and visualization. The WWW was selected as the underlying delivery system because of the many advantages it offers:

- Wide accessibility
- Access to audio, images, animations, and other applications through the use of CGI scripts
- Portability
- Ease of updating and modifying courseware materials

The construction of the modules upon a multimedia, platform-independent, widely accessible base allows it to have a wider audience than otherwise possible.

The project itself consists of several facets, and as such has multiple objectives. These objectives are discussed in detail in the subsequent sections.

MEETING LEARNERS' NEEDS

In creating these modules, we want to provide the kind of support that students need to learn effectively. We know something about the needs of learners and how to design for them[8]. For example:

- Students have a difficult time dealing with abstractions. Through the use of visualizations and multiple representations, we can help make the abstract more concrete.
- Students often do not know a good process to use when undertaking projects. Through collaborative learning forums and other forms of scaffolding, we can support students in taking on a good process.

We want to make meeting these needs an integral part of our courseware, not just an add-on. Throughout our materials, we look for ways to weave in the kinds of supports that will lead to effective learning. To a great extent, these supports need to be interactive, to engage students and to allow them to work with material. Particularly when talking about scaffolding supports, interactivity is a critical feature – support which can't be used while engaged in the designing and programming process, can't help in supporting learning through the experience.

MEETING AUTHORS' NEEDS

The design of the module architecture, as well as the construction of various modules is just one part of a complete courseware system. It needs to be complemented by pedagogically informed tools that aid the author/teacher in the construction and maintenance of the modules. These tools should facilitate lesson design, as well as make the authoring process faster and easier. At a more basic level, the tools should allow the author to focus on the content and not on the specifics of the presentation or the tools. The project is currently building tools that will help the author construct the modules by providing various levels of pedagogical support. Tools to help maintain the module are also under development.

ANALYSIS OF INTERACTIONS

The third goal of the project is to be able to study the interactions with the courseware architecture in order to learn how authors interact with the tools, and how students utilize and learn from the modules. For the system to be successful, the authors must be able to construct modules, and the authored modules must lead to student learning. The evaluation process will take both these criteria into account, and will provide feedback to the system developers as well as to the module authors.

This article concentrates on the module environment, and the use of interactive components in the architecture.

SYSTEM ARCHITECTURE

The organization of the courseware modules is predicated towards a dual-faceted architecture. The material is classified into a course-independent section, and a course-dependent section. The course-dependent section contains material relevant to a particular instance of a course. This will include such material as assignments, exams, supplementary readings, and a schedule of lectures and topics. It will also have topical materials that are specific to the course instance. The course-independent section contains materials that are relevant to the course topic matter in general, and which can be used by different instances of the course.

The rationale behind classifying the materials into two distinct sections lies in the two different proposed uses for the modules. The separation allows different access points for each of the proposed uses. The class view, which is engendered by a particular instance of the course, is accessed from a *syllabus* page. This page is organized so as to emphasize information germane to the class with appropriate links to the course-independent pages. The second view is the class-independent view, which is accessed by a *table-of-contents* page. This page lists all the materials in the course-independent section in a categorized and indexed manner. Additionally, it does not contain any links to the course-dependent section, and thus is not limited in scope to a time period or a specific audience. Finally, the course-dependent pages will contain links to WebCaMILE[7], a collaborative forum for discussion of the material.

The architecture described above enables the use of the modules as either a supplement to a course by using the class view, or in a stand-alone manner by using the class-independent view. Thus, it allows the use of the modules in a flexible fashion that can be adapted to the intended audience. In particular, we envision publishing the addresses of the table-of-contents pages on the WWW for outside audiences.

In the initial phase of the project, we decided to construct a module for an undergraduate computer graphics course. Computer graphics was chosen as the topic since it has much potential for incorporating various multimedia components. It is also a field that is well suited for interactive content. This module, in its current incarnation, is intended primarily as a supplement to an undergraduate computer graphics course being taught on-campus at Georgia Tech. It is an evolving target which is being updated based on learner feedback[1].

INTERACTIVITY IN COURSEWARE MODULES

The main ingredient of a learner-centered environment is the interactivity. As the WWW is the delivery system for the courseware modules, the interactive components are constrained by the current abilities of the HTTP protocol and WWW browsers. Until recently, the only means of achieving interactivity within a WWW page is through the use of the Common Gateway Interface (CGI) standard. The CGI extensions define interactivity either through the use of built-in features such as Forms and Imagemaps, or by using scripts to interface with external applications. Note that these two sets of features are not mutually exclusive, and are often used in conjunction with each other to set up interactive components.

Currently, the courseware modules in our project utilize cgi scripts and forms to provide interaction to the learner. These two features are more suited to the types of interactivity that the modules contain than the imagemap feature. For this reason, the use of imagemaps in the modules is limited. The modules also contain links to WebCaMILE, a collaborative learning environment.

The interactivity in the modules makes use of both HTML forms, as well as external applications. The interactive components can be categorized into two groups – quizzes and exploratory visualizations. Due to their nature, the quiz components are well suited to HTML Forms. The various form elements make it easy to construct simple, multiple-choice questions. They can also be used to create more complex quizzes that involve exploration via interactive animations. Thus, the quizzes in the modules are built using the HTML Forms feature.

The interactive visualizations, in general, require a much richer set of features. Hence, it was decided to use external applications to build these visualizations, and then use cgi scripts to interface with the WWW browser. Using this method, all the visualizations in the modules are built using the XTango[9] or the Polka[10] animation systems. These systems are ideally suited to show animations of various algorithms and dynamic processes. Figure 1 shows an interactive animation of a polygon clipping algorithm which uses the XTango animation system. The animation is used as interactive content in the computer graphics module.

[1]The URL for the computer graphics module is http://www.cc.gatech.edu/gvu/multimedia/nsfmmedia/nsfmmedia.html

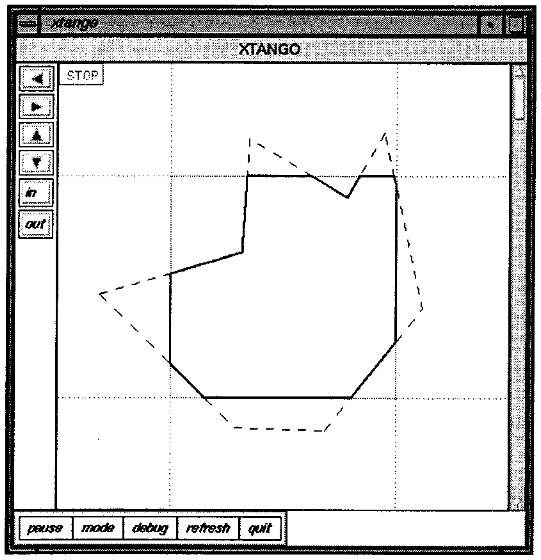
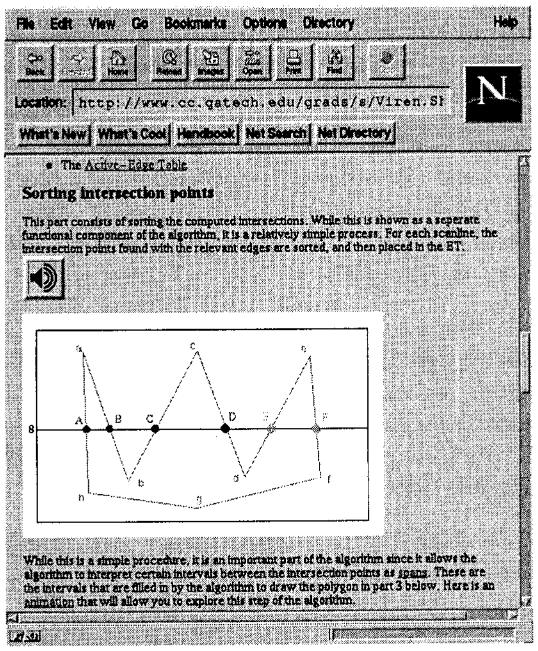


Figure 1: Interactive content using the *XTango* system

The above interactions provide support for representing abstract concepts in more concrete forms. We also need interaction that will provide scaffolding by allowing students to collaborate and participate in discussions forums. WebCaMILE (Web-based Collaborative and Multimedia Interactive Learning Environment) provides a forum for sharing, discussing and reflecting, and several information bases to aid students in locating and making sense of multimedia information. Since WebCaMILE is WWW-based, it is an ideal system for encouraging the students to discuss the materials in the module, as well as various issues that arise during the course. WebCaMILE links are present in the course-dependent pages of the module. For example, a WebCaMILE discussion on a project can provide students with a forum for discussing various design issues.

The computer graphics module contains several interactive components. The components consisting of forms can be embedded within a module page. When the form is submitted, a cgi script is called and appropriate action is taken. This allows the quiz to be an integral part of the page. On the other hand, in order to call the external application, a cgi script is called which performs various display related functions, and then launches the application. One advantage of using external applications is that the learner could simultaneously peruse the module while interacting with the application. This general methodology for including interactive content is also being followed in other modules. Figure 2 shows a module page from the computer graphics module. The page shows various multimedia content including audio, images, and hypertext links to other documents. It also has a link to a cgi script.


File Edit View Go Bookmarks Options Directory Help
Back Home Reload Images Open Print Find
Location: http://www.cc.gatech.edu/grads/s/Viren.St
What's New What's Cool Handbook Net Search Net Directory
a c e A B C D E F 8 b d f h g

- The Active-Edge Table

Sorting intersection points

This part consists of sorting the computed intersections. While this is shown as a seperate functional component of the algorithm, it is a relatively simple process. For each scanline, the intersection points found with the relevant edges are sorted, and then placed in the ET.

While this is a simple procedure, it is an important part of the algorithm since it allows the algorithm to interpret certain intervals between the intersection points as spans. These are the intervals that are filled in by the algorithm to draw the polygon in part 3 below. Here is an animation that will allow you to explore this step of the algorithm.

Figure 2: Computer Graphics module page with multimedia links

This script will give the learner instructions for allowing an external application to be run on the learner's machine. When the learner is ready, the script can be made to execute the external application which is being used as interactive content.

Although this use of interactive components does enhance the module and provide the learners with a constructivist environment, it has some serious disadvantages. The main problem is that the interactivity achieved in the cgi scripts and HTML forms is slow and exhibits a lack of real-time feedback. This is an artifact of the manner in which the HTTP protocol works.

Another factor to consider is that the use of external applications may prove to be a security risk. Also, it may not be possible to display the applications on all platforms due to differences in windowing systems.

INTERACTIVITY: THE NEXT STEP

In order to be able to have full-fledged interactivity on the WWW, and for the interactive components to be an integral part of the system, the following issues must be addressed:

- The WWW browser should allow the embedding of interactive components within a document.
- There should be a feature-rich language (toolkit) that can be used to build the embedded components.
- The components should be portable, and should have the potential to run on a large variety of platforms.
- The security risks should be decreased.

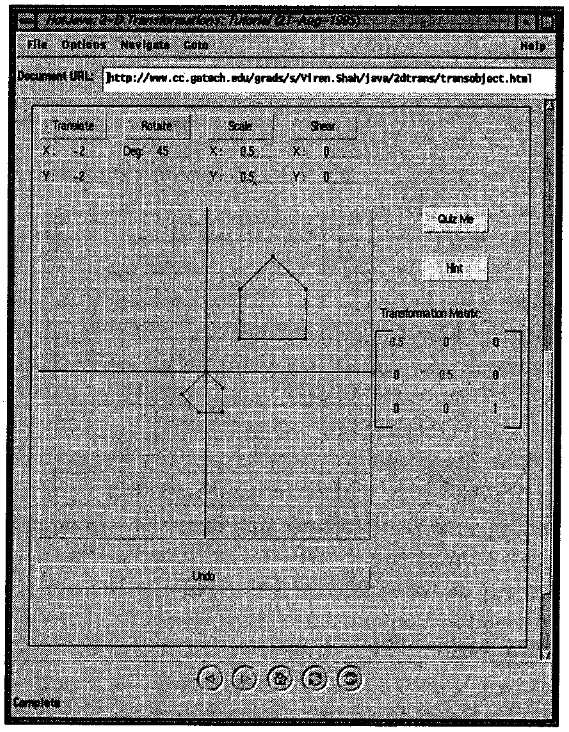
All these issues are addressed to some extent by the *virtual machine* paradigm. In this scenario, the client is extensible through the use of portable byte-code. One such client is the HotJava WWW browser which makes use of the Java[6] language. The way HotJava works is by byte-compiling Java *applets* into architecture neutral bytecode. This bytecode can run on any platform on which Java's runtime environment is present. Java also has an extensive graphics library that can be used for building interactive visualizations. Its interaction capabilities have already been tested since the HotJava browser itself was written in Java. The manner in which HotJava facilitates the inclusion of interactive components is threefold. First, HotJava can transparently migrate software across the network and read in the executable content. This allows the browser to provide a fast response rate, since the code has already been read into the browser. Also, HotJava reads in Java byte-code, and dynamically converts it into native machine code; this makes code written for HotJava portable. Finally, Java does simple authentication and is also more robust, causing it to be a more secure environment.

Given these features, the next step in enhancing the interactive content within the courseware modules is to use Java applets embedded inside the module pages. This is advantageous in two ways (1)The interactive content is now seen as an integral part of the module page, rather than as an add-on or accessory. It also enrichens the learning environment by providing the learner with a composite view of simulations and explanations. (2) All of the interactive content can now be written using one language. This makes it easier to interface and coordinate different components that may be present in the module page.

In order to gauge the impact of using HotJava applets in the courseware modules, several prototype applets that are being used as interactive content in the computer graphics module, have already been constructed. Figure 3 shows a HotJava applet that is being used as an embedded interactive content within the graphics module. The applet allows the learner to explore 2-dimensional transformations by manipulating figures, juxtapositioning transformations, and receiving immediate feedback. The applet also incorporates a testing facility which can be used by the learner to test their knowledge.

CONCLUSION

We have described our work exploring how the WWW can assist computer science education. Our approach is to



Document URL: http://www.cc.gatech.edu/grads/s/Viren.Shah/java/2dtrans/transobject.html

X: -2 Y: -2 Deg: 45 X: 0.5 Y: 0.5 X: 0 Y: 0

Transformation Matrix:

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 3: A *Java* applet used as an embedded interactive component

develop both class-specific techniques and modules useful for individual courses, and more general materials focusing on particular aspects of computer science. Our focus on these latter pages, in particular, has been to emphasize interactivity to promote learning and involvement. This project is on-going with an initial focus on a module for teaching computer graphics. We are currently developing pages and authoring tools to assist instructors and students in a local computer graphics class, and we are developing a module for the area of computer graphics in general.

ACKNOWLEDGEMENTS
This research has been supported in part by a grant from the National Science Foundation (CDA-9414227).

References

[1] A. Rosina Bignall, Dalinda Kae Bond, Judy Cossel Rice, and Phillip J. Windley. Uses of Mosaic in a University Setting. In *The Second International WWW Conference '94: Mosaic and the Web Advance Proceedings*, volume 1, October 1994.

[2] A. L. Brown, J. D. Bransford, R. A. Ferrara, and J. C. Campione. Learning, remembering, and understanding. In Kessen W, editor, *Handbook of Child Psychology: Cognitive Development*, volume 3, pages 77–166. Wiley, 1983.

[3] J. K. Campbell, S. Hurley, S.B. Jones, and N.M. Stephens. Constructing Educational Courseware using NCSA Mosaic and the World Wide Web. In *Electronic Proceedings of the Third WWW Conference '95*, April 1995.

[4] A. Collins, J. S. Brown, and S. E. Newman. Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, pages 453–494. Lawrence Erlbaum and Associates. Hillsdale, NJ, 1989.

[5] S. Farnham-Diggory. Schooling. In J. Bruner, M. Cole, and B. Lloyd, editors, *The Developing Child*. Harvard University Press. Cambridge, MA, 1990.

[6] James Gosling and Henry McGilton. The Java Language Environment: A White Paper. Sun Microsystems, Inc. Unpublished, May 1995.

[7] M. Guzdial, D. Carlson, and J. Turns. Facilitating learning design with software-realized scaffolding for collaboration. In *Proceedings of the Frontiers in Education Conference*. American Society for Engineering Education, 1995. In Press.

[8] E. Soloway, M. Guzdial, and K. E. Hay. Learner-centered design: The challenge for HCI in the 21st century. *Interactions*, 1(2):36–48, 1994.

[9] John T. Stasko. Animating algorithms with XTANGO. *SIGACT News*, 23(2):67–71, Spring 1992.

[10] John T. Stasko and Eileen Kraemer. A methodology for building application-specific visualizations of parallel programs. *Journal of Parallel and Distributed Computing*, 18(2):258–264, June 1993.

[11] D. Wood, J. S. Bruner, and G. Ross. The Role of Tutoring in Problem-solving. *Journal of Child Psychology and Psychiatry*, 17:89–100, 1975.