Stimulating Introductory Engineering Courses with Java

Ronald Danielson
Sally Wood
School of Engineering
Santa Clara University
Santa Clara, California 95053

Abstract - *The traditional combination of text, lectures, and laboratories that constitutes the learning environment for most introductory engineering courses has inherent restrictions that limit student success. Texts are static; attempts to allow students to reproduce a dynamic lecture experience outside of class (e.g., videotape) are ineffective and time consuming; even laboratories allow students to explore the impact on behavior of only a few sets of inputs. Supplemental materials such as simulations and animations, that are dynamic and allow the student to observe and evaluate the behavior of systems under different conditions, help students to truly understand concepts and develop a sound basis for more advanced studies. The capabilities of current technologies, such as Java and the World Wide Web, make it attractive to develop such materials. This paper discusses our experiences to date in creating Java–based supplemental courseware for lower division engineering courses, describes the courses and supplemental project involved, and suggests some approaches to structuring a collection of such materials.*

Introduction

The traditional combination of text, lectures, and laboratories that constitutes the learning environment for most introductory engineering courses has inherent restrictions that limit student success. Texts are static, making it very difficult to instill in students an appreciation for the changes that occur in devices and systems over time, and they provide only a single path through material with many levels of interconnections. Lectures are dynamic and offer the potential to explore time–varying behavior, but lecture time is scarce and usually strictly choreographed, and students can't reproduce the experience outside of class. Even laboratories allow students to explore only a few sets of inputs and evaluate their impact on behavior. This traditional combination is sometimes successful for students who are fluent in mathematics and comfortable with abstract thinking. However, it has often failed for students with a poor education in mathematics (a type of student becoming more common in engineering school classrooms) and, by reputation, discourages many qualified students from entering engineering programs.

Supplemental materials such as interactive simulations and animations, that are dynamic and allow the student to observe and evaluate the behavior of systems under different conditions, help students to truly understand concepts and develop a sound basis for more advanced studies. For example, the transition from transient to steady state response of mechanical and electrical systems is not obvious from mathematical representations. However, it is much more clear when the time response of differential equation solutions is observed or physical models are simulated [3]. Multiple media can offer a variety of visualization aids to better match the different cognitive styles of the students, and the interactive component requires students to control their exploration of new material in ways that the predominantly passive reading of text and reference books cannot. Evidence suggests this hypermedia approach is particularly relevant for beginning students [2].

Students in engineering courses may especially benefit from dynamic materials. The topics are complex, layered, and interdependent. Students are often so inundated with facts and details that they may fail to understand the broader context of their education, and thus fail to develop the ability to absorb new data and observations in context, a skill that is needed during their professional careers.

Current technologies make it attractive to develop interactive courseware, and we are in the midst of a project to do so for several introductory engineering courses. This paper discusses our experience with this process to date, details the courses and topics involved, and offers some preliminary thoughts on structuring a collection of courseware modules.

Courseware Development Trends

Advances in technology have altered the focus of courseware development, away from user interface issues of the mechanics of presenting material and interacting with a student and towards content and presentation strategies. Prior to the development of windows–based operating systems, the major effort of any educational software development project was the creation of an effective user interface. When standard user interfaces such as SunView and Motif became widely available, courseware development efforts could focus more on pedagogical content rather than details of the user interface. The main problems that limited the effectiveness of courseware of this type were associated with platform dependence. Offering courseware on only a single

platform limited distribution and reduced the pool of potential users. Meeting specific requirements of multiple hardware or operating systems siphoned time from development activities and reduced both quality and the scope of coverage.

Many current courseware development efforts, including those discussed here, exploit the advantages of recent technology. Modern workstations and software systems offer many more options for module design, particularly the possibility of combining additional media (audio, video, and animations) to support and reinforce learning. The World Wide Web offers a standard interface and relatively easy delivery of multiple media, to enrich the presentation and to allow students to choose a combination of media that fits best with their preferred learning styles. Java provides platform independence, which allows implementation of dynamic simulations without having to support multiple versions, and offers hope of reaching an audience broad enough to offset development costs. Since students can easily access modules stored on a remote server, problems of distributing software and updates are minimized. HTML provides a means for an instructor to quickly organize course–specific sequences of courseware modules selected from libraries developed by many different organizations (provided the modules are designed to facilitate such reuse).

This still leaves many issues to be addressed before courseware development can be considered a "solved problem," including determining appropriate combinations of media to describe and illuminate dynamic behavior, providing mechanisms to allow instructors to guide students to suitable supplemental materials, and developing ways to allow quickly combining individual modules to produce coherent sequences of enrichment materials.

Our Approach

Previous efforts [3] have resulted in the development of two Motif–based tutorial collections for undergraduate electrical and computer engineering courses. One is focused on introductory material for digital systems and digital signal processing, and the other is focused on the mathematical concepts common to most fields of engineering. Although they cover different topics, there is a common style and a similar selection of interaction activities. In both cases the user requires no training, and the user interface is a close as possible to "self evident." A typical module allows the student to control the parameters, initial conditions, and several driving inputs of a mathematical differential equation using sliders. The mathematical solution steps and a graphic display of the results appear on the screen. Text is available to explain the operation of the tutorial and guide the user through a sequence of examples which explain basic concepts.

Our current efforts are guided by the philosophy of providing a series of interactive modules that allow students to explore fundamental concepts by manipulating variable parameters in a controlled environment and evaluating the resulting behavior. We think in terms of different forms of interaction that a module may provide about a topic.

- *Lectures* provide information about a topic or about using the module itself, and primarily display passive text or hypertext.
- *Demonstrations* are dynamic components explaining the behavior of some aspect of the topic that a student may step through but can't change.
- *Examples* are dynamic components that direct a student to set initial conditions and then describe the resulting behavior and the reasons for it.
- *Free interactions* are dynamic components that provide a student full control to explore the underlying concept.

Any of these may employ a variety of media to provide the desired experience, and they are often combined (e.g., an example followed by free interaction) to achieve a pedagogical objective.

Our primary goal is not to produce self–standing tutorial systems that would deliver an entire course via the World Wide Web. Instead, we are creating materials that supplement the capabilities of traditional texts and lectures and are applicable in a wide variety of situations: to enliven lectures, to provide virtual objects that form the basis for scheduled labs, to be accessible during office hours to clarify concepts, and to be used outside of formal learning environments by students to help them master topics in a particular course.

All our modules use HTML for text presentation, usually providing brief overviews of a topic with links to more in–depth explanatory materials. Currently, most of the non-textual information is graphics or animations, although audio is sometimes used for particular topics. Java applets control the animations, sometimes using Java Beans components. We are developing applet libraries and HTML templates to make it easier for other faculty to develop similar materials on topics of their choice.

Courses and Topics

We have been developing Java–based courseware to address some of the most problematic topics in three lower-division engineering courses: an introductory course on fundamental concepts for electrical engineering majors, an introductory Java programming class, and a course covering fundamental concepts of digital technologies for non–engineering majors.

Introductory Electrical Engineering

There are two primary goals of the introductory electrical engineering course. The first is that it should present a coherent introduction to the field of electrical engineering, with hands on laboratories appropriate to the limited experience of incoming freshman students. The development of subfields is presented in the context of enabling technologies that have provided the foundation for innovation. The second objective is the introduction of fundamental concepts that will be used in later courses so that students will have "the big picture" in mind as they study more focused individual areas in great detail. As systems become increasingly complex, the ability to understand them at several different levels of integration becomes essential. Breaking a system into functional blocks partitions the problem so that each block can be considered separately.

The topics covered in the course include the basic electrical quantities of charge and voltage, and their relationship to electrical power and energy; transducers and component behavior; sinusoidal waveforms and time constants; communication signals, bandwidth, multiplexing, switching and routing; digital circuits for communication and computing; microelectronics; control systems; and power generation. The current enabling technologies of wireless communication and increasingly high levels of integration in microelectronic circuits are the basis for future projections.

This course is presented as a mixture of lecture, group laboratory experience using physical experiments, Java simulations and animations to prepare for laboratories and extend them later, interactive tutorials to help communicate concepts and relationships, and MATLAB assignments for computation, graphing, and system modeling. Extensive use of auditory feedback is used in both the physical experiments and the MATLAB simulations. A more complete description of this course with some specific examples may be found in [4]. The course has been offered several times and most of the components of the course and the experiments have been tested. New web based tutorials are being developed extending the existing set, and the Java simulations are in progress.

Introductory Programming

The objectives of the introductory programming course are to teach students a particular programming language, familiarize them with fundamental concepts associated with programming, and give them practice in problem solving and design in the context of program development. Most beginning students struggle with the syntax of the particular programming language used, but their real conceptual difficulties occur with topics such as control flow, subprogram invocation (particularly for recursive subprograms), parameter linkage, and dynamic memory allocation. Object–oriented languages add encapsulation, protection, and inheritance to that list. A language like Java also adds issues of window APIs and multithreading. All of these topics are candidates for courseware development.

We have implemented a series of modules for Java, the language for the introductory course beginning in 1998–99. For example, Figure 1 shows a simple module for an if statement, displaying program text and flowchart. Steps through the flowchart are synchronized with a highlight of the corresponding line of the source program segment. Students can control the rate of the animation using a slider, and can enter values for the variables in the conditional expression and watch the effect on behavior.

This is the one area where we are creating a full tutorial on a complex topic [1], primarily as a test of some of our ideas on canonical module structures and composition methods, but also as an aid for students who completed an earlier version of this course in C++ and want to develop Java skills. The coverage has been expanded to include topics such as awt (abstract windowing toolkit), exception handling, multithreading, and advanced object concepts. Explanatory text describes each language feature, program fragments that explicate the topic are shown, and graphics and animations help the student understand concepts. Figure 2 presents an example from the module on multithreaded programming in Java.

Digital Technologies

"Understanding Digital Technologies" is a service course offered by the computer engineering department to the broader campus population. It is one of a dozen courses campus–wide (the introductory electrical engineering course described above is another) that satisfy a university core curriculum requirement in Technology. The course is typically taken by students in the humanities, social sciences, and business, and provides them an overview of the major technical areas related to digital computers: semiconductor devices, computer architecture, software development and systems, and computer networks. The principal objectives for the course are an intuitive understanding of the workings of digital computer technology, an ability to analyze new technologies and place them into context, and an appreciation of the impact of these technologies on society.

Topic areas that students have had difficulty understanding in the past include the behavior of a pn junction, logic circuits, instruction and data flow in computer architectures (particularly pipelined and superscalar machines), cache and paged memory operation, and network protocols and routing. We are developing two sets of courseware modules, one for network concepts (circuit vs. packet switching, the operation of a protocol stack, and the idea of routing between different networks) and another for introductory logic design (logic functions, building circuits, and timing diagrams). Both sets of materials employ similar combinations of textual explanations enlivened by animations.

This course allows us to test the effectiveness of our courseware in supporting "outreach" efforts to provide technical education for non–technical students. We also expect that many of the modules developed to support this course will have wide applicability for other courses both inside and outside the engineering school.

Long Term Objectives

Effective courseware must be relatively easy to create, so developers may concentrate on content and pedagogy, and must provide an effective learning experience for a wide variety of students. An earlier paper [5] discussed issues related to development tools and some thoughts on adapting both the level of material presented and the media chosen to reflect a student's preferences and past performance. We are also concerned about how best to organize a collection of courseware modules and how to facilitate combining a series of modules from a library to produce a set of supplemental courseware for a particular class.

Ideally, the collection structure should reflect a cognitive model of the subject. This provides suggestions for what additional modules should be developed and makes it much easier for faculty familiar with the subject to select appropriate modules from the collection. But the same concept often underlies multiple engineering subject areas (for example, the principle of locality of reference in computer engineering) and engineering curricula reflect this. A student will see the same topic at different levels of complexity several times in her undergraduate career. So the relationship between subjects and topics is an acyclic graph rather than a tree. We would like the inheritance hierarchy of our Java courseware to mirror this structure, but Java allows only single inheritance.

Similarly, the courseware modules that cover various topics should provide some consistency in presentation format, so that course–specific assemblies of modules offer familiar interfaces. At the same time, we don't want to unduly constrain module developers from creatively designing unique interactions that are well suited to a particular topic.

We are deliberately exploring different implementation styles in the first sets of courseware that we are developing, to gain experience with the unique aspects of Java and the World Wide Web. But it will be essential to define a topic structure and implementation guidelines for developers as the courseware collection grows. Our preliminary thoughts are to create a library of general classes (logic functions, signals, program fragments) that can be extended to create specific instances appropriate for each topic. Similarity of interaction can be provided by defining a set of Java interfaces, so that modules that implement a particular interface will offer uniformity of interaction. For example, different interfaces may apply to modules that offer different combinations of interaction experiences (lecture, demonstration, example, and free interaction) or different combinations of media. Modules developed under this structure are similar to templates, in that the interface provides the outline of what must be implemented and inheritance may well fill in much of the needed code. This should result in more efficient courseware development.

Summary

Current computing and communication technologies, such as Java and the World Wide Web, offer great potential for development of courseware. Engineering courses cover complex, layered and interdependent topics, and can overwhelm students in a rush of fact and detail. Because of this, students in introductory engineering courses may especially benefit from interactive courseware that provides simulations and animations that allow them to explore new concepts more fully. Such courseware modules are being developed to support three introductory engineering classes. Experience to date indicates they can improve the educational experience in several ways, but there are still significant issues to be addressed in developing courseware modules. Exploiting the inheritance and interface capabilities of the Java language to create module templates may be one approach toward more efficient module development techniques, and may increase the ease with which consistent courseware sequences can be composed.

Acknowledgments

We are grateful to Hewlett–Packard and 3Com for donations of computer and networking hardware and software.

References

[1] Bhimaraju, G. "A Web-Based Java Tutorial About Java," MS Thesis, Computer Engineering Department, Santa Clara University, June 1998.

[2] Najjar, L. "Multimedia Information and Learning," *Journal of Educational Multimedia and Hypermedia*, 5(2), 1996, pp. 129–150.

[3] Wood, S., "A New Approach to Interactive Tutorial Software for Engineering Education," *IEEE Transactions on Education*, 39(3), 1996, pp. 399–408.

[4] Wood, S., "A Concept Oriented Freshman Introductory Course Utilizing Multimedia Presentations and Group Laboratory Experience," *Proceedings of the 1998 Frontiers in Education Conference*, IEEE Press, 1998, to appear.

[5] Wood, S. and R. Danielson, "Web–Based Enrichment Courseware for Introductory Engineering Students," *Proceedings of the International Conference on Computers and Advanced Technology in Education*, International Association of Science and Technology for Development, 1998, pp. 121 - 125.

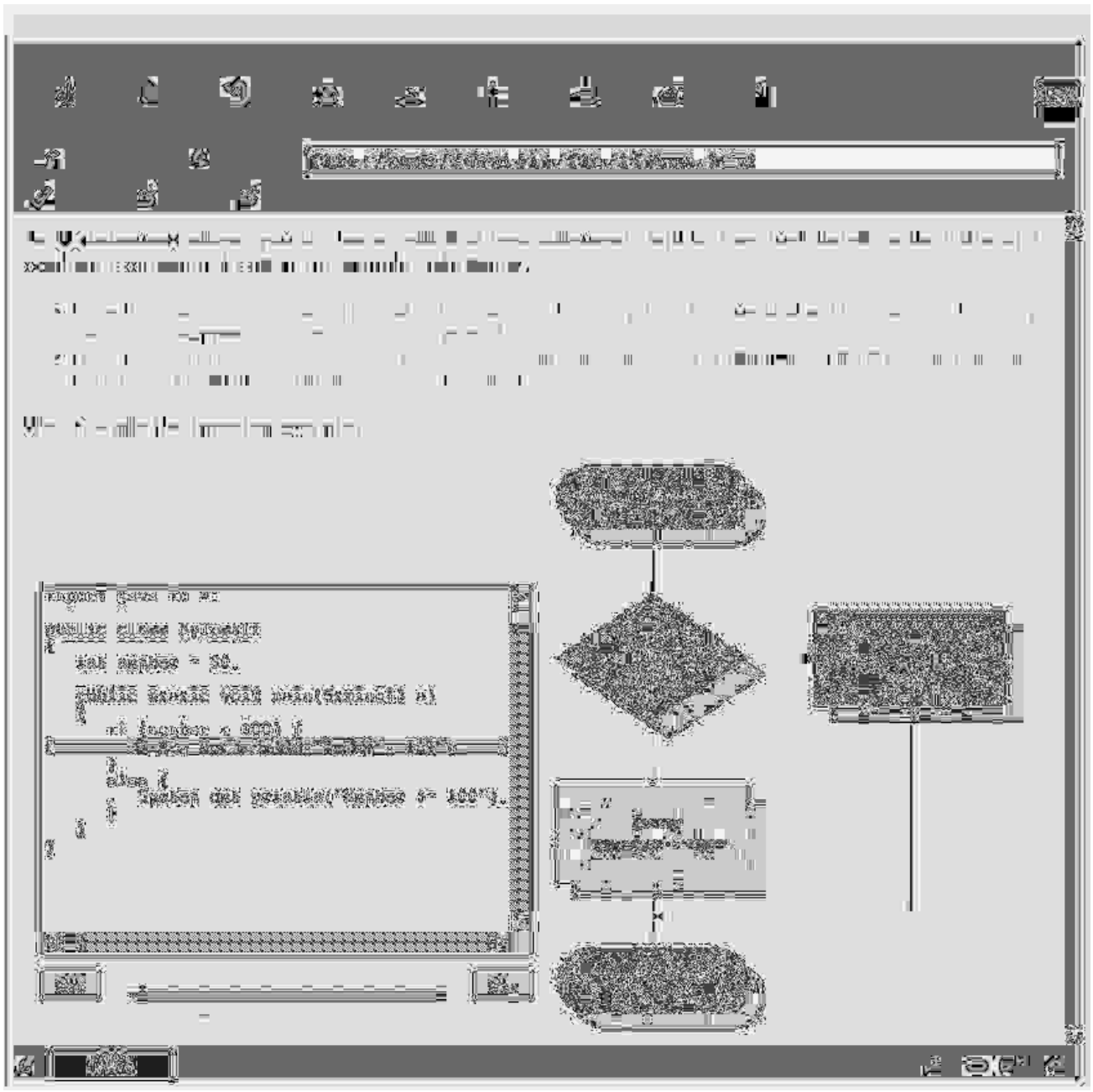

Figure 1. Control Flow for if Statement

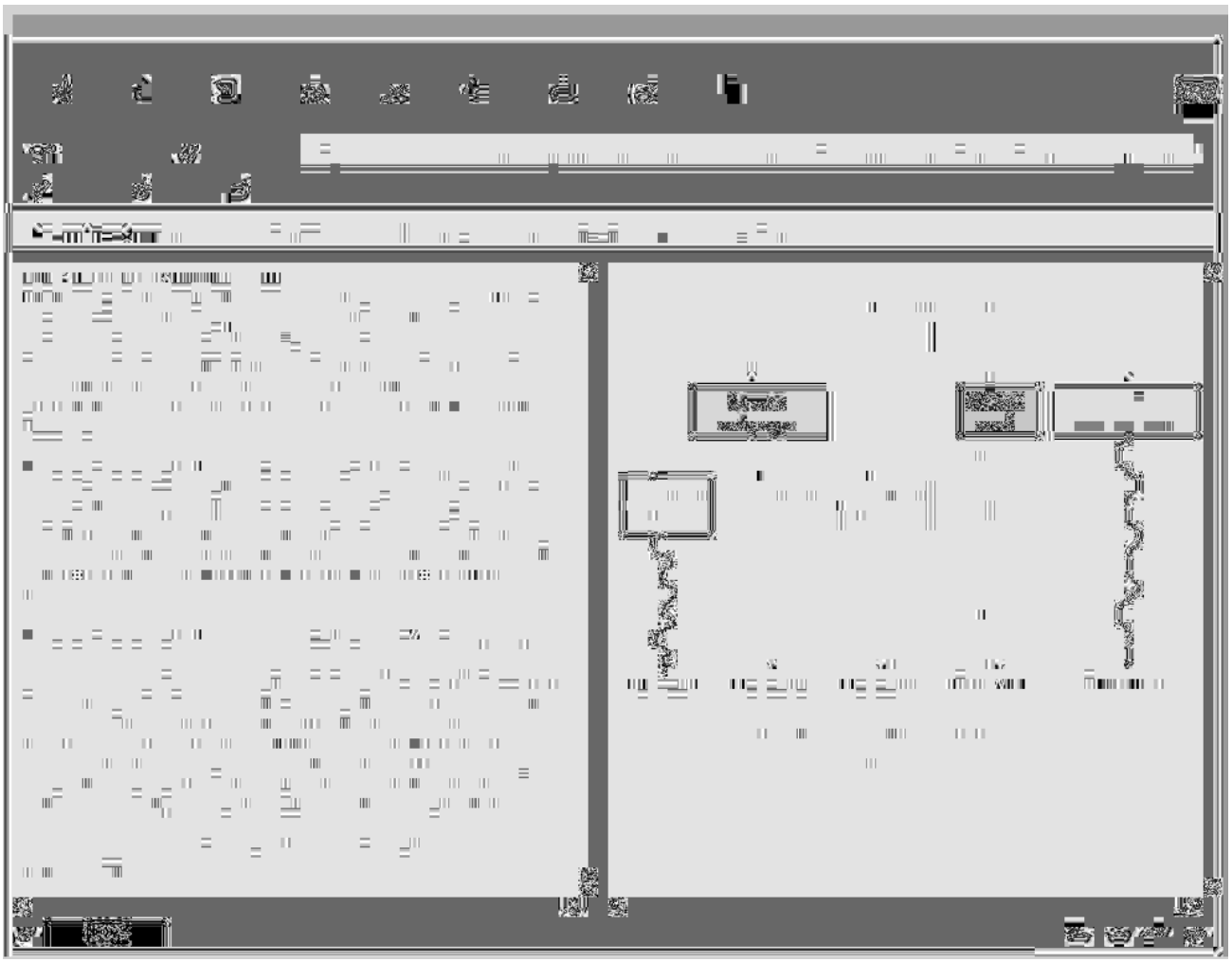

Figure 2. Fragment from Java Multithreading Tutorial