

Teaching Applet Programming to Non-Majors — Virtually

Scott Dexter

Department of Computer and Information Science

Brooklyn College of CUNY

2900 Bedford Avenue

Brooklyn, NY 11210

sdexter@sci.brooklyn.cuny.edu

Abstract. *We discuss the development and deployment of a “partially virtual” computer science course for non-majors in which the Web mediates a significant fraction of the course. In particular, approximately a quarter of the course is dedicated to teaching Java applet programming through the use of a Web-based programming environment. We address the development of the curriculum, the integration of several Web-based tools, and make some preliminary observations about the pedagogical effectiveness of and student response to this approach.*

INTRODUCTION

While the technology that supports distance learning has developed extremely rapidly in recent years, pedagogical techniques that make the best use of this technology to create effective learning environments have been slow to develop. At Brooklyn College, we are experimenting with a “partially virtual” course structure. In a project funded by FIPSE, faculty in every part of the College’s Core Curriculum are developing and offering “partially virtual” Core courses. In these experimental classes, 1/2 to 2/3 of the course takes place in the traditional classroom setting while the remainder is entirely mediated by the Web. This “wholly virtual” component of the course takes a variety of forms, from, for example, online discussions to virtual laboratories. As part of this project, we have developed a partially virtual curriculum for the Computer Science and Mathematics Core.

The virtual curriculum we have developed has four main components: self-guided lessons on the history of computing and the internet using resources available on the Web; lessons on simple HTML using a JavaScript-based tutorial tool; a discussion of basic number theory (bases, modular arithmetic, and prime numbers) augmented by JavaScript exercise-generation tools; and a series of lessons on writing simple Java applets, mediated by a Web-based tool called WebToTeach [1], described below.

In this paper we discuss the design and development of this curriculum, with a particular focus on the role of WebToTeach in making applet programming accessible to non-major novice programmers. We conclude with some preliminary qualitative observations about the pedagogical effectiveness of and the student response to this approach.

CURRICULUM

Historically, the curriculum of the Mathematics and Computer Science core course has consisted of an eclectic mix of topics from mathematics and computer science, most commonly logic, probability, and an introduction to programming in Pascal. In our development of the curriculum of this partially virtual form of this course, we sought not only to incorporate effective asynchronous pedagogy but also to create a more modern and unified structure. By organizing the course around the theme of networked and digitized communication, we have been able to create a course which both addresses fundamental concepts of computer science (such as algorithmic problem-solving) and gives students insight into modern applications of computer science (such as how the design and structure of the Internet impacts their experience as users). Simultaneously, we introduce a variety of mathematical concepts that are both fundamental and demonstrably relevant: binary representation of information and the utility of different base systems; modular arithmetic for error detection/correction and cryptography; and prime numbers for cryptography. In this section we give a brief overview of each of these sections of the course.

Introduction to the Internet

When students register for this course, they are not told that certain sections are especially Web-intensive. While many students have computers and Internet access from their homes, other students are at a significant disadvantage. Those that rely on the College’s computing facilities for Internet access find that the one-hour session limit imposed in the labs has a significant impact on their ability to complete the coursework. The remaining 15-20% have little to no experience using the Web, email, or other networked applications.

Like most users, even those students with some experience have little understanding of the underlying mechanisms. Thus, we spend the first few weeks of the course introducing the Internet — its history, its use, and its structure and design. This we accomplish both by in-class demonstrations, lectures, and discussion and by directed studies of the Internet (as well as a brief history of computer

science in general), in which students discover the answers to questions by browsing resources on the Web.

HTML

In order more clearly to illustrate the structure of the Web (and also to provide students with the technical knowledge to “self-publish” on the Web), we spend two “virtual” sessions on HTML, prefaced with a short in-class lecture on HTTP and the “layered” relationship among network protocols as well as the philosophy of hypertext and markup languages.

These virtual sessions are structured around a JavaScript tool that allows students to see instantly the interpreted form of their HTML source. After a number of exercises using this tool (introducing the usual variety of basic markup tags), students then design a simple personal home page incorporating these elements.

Number Theory

Number-theoretic topics are scattered throughout the course; all are taught similarly. An introductory lecture sketches the relevance of the topic to applications discussed earlier, then students use interactive JavaScript tools to study further examples and to test their knowledge.

Arising from the discussion of the Internet and digitized communication in general, the topic of information representation and base systems allows students to explore the role of binary notation in digitizing information.

A more general discussion of information representation and the utilization of error detecting and correcting codes motivates the idea of modular arithmetic, which is continued into our discussion of cryptography.

Finally, a broad discussion of network security issues, including types of threats as well as defenses against them, introduces the idea of cryptography, especially the RSA cryptosystem, which provides a platform for an exploration into prime numbers, modular arithmetic, and factorization.

Applet Programming

An introduction to programming is a requisite element of any introductory computer science course; in our case we chose to maintain our alignment with networking and introduce programming in the context of Java applets. While applets initially appear to be excessively complex (both syntactically and conceptually) for an introductory course, we believe this is strongly compensated for by these factors: (1) applets are “real” programming (as we illustrate during our discussion of the Web), (2) applets produce visually appealing output, and (3) WebToTeach (discussed below) allows us to present applets in a gradual, friendly fashion.

While time constraints limit the complexity (and hence the practical utility) of the applets we ask our students to create, there is ample opportunity to expose the challenges

and processes of programming — the syntactic and conceptual rigor required, the idea of ‘layers’ of functionality, and the algorithmic approach to problem-solving

WEBTOTEACH

WebToTeach is the keystone of the applet programming part of the course. Developed at Brooklyn College, WebToTeach provides a Web-based environment for managing computer science programming exercises.

WebToTeach is motivated by the thesis that small-scale exercises, allowing students to work with and master “micro-concepts,” are an integral part of computer science pedagogy, much as in the pedagogy of related disciplines such as language arts and mathematics. But in computer science, the multiplicity of both “right” and “wrong” answers (that is, program fragments intended to solve a particular algorithmic problem) offers a significant logistical obstacle to sufficiently rapid evaluation of (and feedback on) such exercises. WebToTeach moves into this gap with a self-contained environment in which students receive instantaneous feedback based on the results of the instructor-specified tests. At the same time, instructors can monitor student progress in real time and offer suggestions as students develop their solutions.

In this section we give a brief overview of the WebToTeach architecture, then address specifically the utility of WebToTeach for the beginning programmer, then finally remark on the particular experience of using WebToTeach to introduce Java applet programming.

An Overview

WebToTeach provides both a simple interface students use to submit their solutions and a sturdy scaffold for testing these submissions according to the instructor’s specifications. As students are developing their solutions, the instructor may at any time view a student’s latest submission to offer comments or to estimate the class’s collective mastery of the material.

WebToTeach allows exercises on many scales: solutions may take the form of programming fragments or of complete programs; programs may occupy a single file or may require multiple file submission. Instructors specify an exercise by providing instructions (of course) and indicating whether it is to be a single- or multiple-file submission. They may also provide data and/or source files that are hidden from students. They then provide a series of tests: for single-file submissions, these tests are composed of additional fragments prepended and appended to the submission to form a complete program. In the case of multiple-file submissions, instructors provide a test driver. These tests may also utilize the hidden source and data files. Finally, the instructor specifies how the program results are

to be evaluated, e.g. by querying exit codes or checking the correctness of the output.

From the student's perspective, the WebToTeach interface is much simpler. Each exercise is presented simply as the instructor's directions, one or more text areas (one per file to be submitted), and a button labeled "Submit." Immediately after a solution is submitted, WebToTeach applies the instructor-specified tests and responds with a message — either "Congratulations" or "Redo." If one of the tests fails, WebToTeach also provides the output from the program/test driver, which is of course customizable by the instructor to include appropriate hints, commentary, etc.

As students are developing, submitting, and resubmitting their solutions, the instructor may inspect any student's most recent submission (and is also informed of the number of unsuccessful previous submissions). This information represents valuable feedback to the instructor, who may then either post comments on that particular student's work, or may provide appropriate amplification and clarification to the entire class.

WebToTeach and the Novice

The experience of being introduced to programming, especially for students with little or no prior computing experience, is widely understood to be a painful one. In addition to learning the syntax and semantics of an artificial language, students must learn a new kind of intellectual discipline — and must also adapt to a computing environment appropriate for programming. WebToTeach, used as an introductory "programming environment," allows us to provide some separation of these concerns

For students with experience using the Web, WebToTeach represents a simple interface with a familiar appearance — the programming assignments given to introductory students consist simply of instructions, a text area in which a solution may be typed (or pasted), and a button for submitting the solution. For students with less experience with the Web (or computing in general), the interface may be experienced less intuitively, but is still much easier to master than that of most popular IDEs (or command-line based systems).

The architecture of WebToTeach further allows beginning students to focus exclusively on issues of syntax/semantics or on the problem of algorithmic thinking. Combining the two areas — i.e. creating entire programs — may be deferred until students have acquired some mastery of the two independently. For example, students learning about conditional statements may be asked first to determine the effects of a number of conditional statements (WebToTeach provides automated feedback for fill-in-the-blank/fill-in-the-number questions, as well as multiple choice and true/false questions); then they might be asked to provide conditional statements to accomplish simple tasks; they might then be given a collection of conditional statements and asked to arrange them to accomplish a more

sophisticated task; finally, they would write complete programs incorporating these statements.

While it would be difficult or impossible for an instructor to provide timely feedback and guidance on each of these exercises (for, say, 3 classes of 30 students each), WebToTeach allows the instructor to craft these exercises so that students can receive adequate feedback to complete a laboratory experience such as this one in a few hours.

WebToTeach and Applet Programming

Applet programming is demanding: it requires some familiarity with object-oriented programming, it imposes a number of arcane-seeming syntactic requirements, and, of course, it demands familiarity with common programming constructs. WebToTeach's facilities for fragmentary exercises offer a simple mechanism for assisting students over these hurdles quickly. Students can begin learning about objects and messages directly, without the distraction of the applet architecture, then learn more specifically, for example, about the functionality of Java's `java.awt.Graphics` class, then produce a complete simple applet by adding the appropriate 'boilerplate.' In subsequent lessons, students can explore conditional statements, looping constructs, and other applet methods in the context of program fragments, then incorporate these ideas into more complex applets.

WebToTeach also allows students to focus on the generation of the applets themselves free of the complexities of the applet compilation and execution model. Rather than having to use the Java compiler to produce class files, then create the HTML files incorporating the applet classes, and finally view/execute the applets using a browser, students may rely on WebToTeach's framework, which fully automates this process and displays the executing applet directly.

Java provides particular advantages for testing within the WebToTeach framework. Languages such as BASIC, Pascal, and C are commonly used as introductory programming languages in this type of course, but offer limited interaction with automated testing tools. In particular, without special-purpose compiler-like tools, only a program's *behavior* is susceptible to testing. While this is generally adequate for assessing correctness, it limits the type and quality of feedback that the automated tests may provide. Java, however, provides a strong capacity for *reflection*. By exploiting reflection, we may design tests that offer feedback not only on an applet's behavior but also on its structure; we discuss this in more detail in [2]. For example, tests may detect incorrect class declarations and incorrect parameter passing, as well as incorrect method behavior. While errors such as these are of course detected by the Java compiler, these tests effectively circumvent the compiler's output, allowing us to replace the compiler's messages (which are often inscrutable and intimidating in the eyes of the beginning programmer) by hints and suggestions for resolving the problem.

RESPONSE AND EFFECTIVENESS

Most students meet the prospect of taking a course based largely on the Web with interest and enthusiasm, although some, particularly those with little Internet experience, find the course structure initially intimidating. Here again, issues of access play a prominent role determining student's engagement with the course material. Those students who have little previous or current contact with the Internet find themselves at a significant disadvantage. Some are able to make time to master the extra material necessary even to navigate the course effectively; others either opt to drop the course early on or else find themselves unable to maintain sufficient contact with the material, with the most likely outcomes of withdrawal from the course or failure.

Our ambitious "virtual" curriculum — nearly half the class sessions are designed to take place on-line — forces a choice between a steady level of reduced class contact (i.e. meeting once instead of twice per week for the entire course) or lengthy periods of solely virtual coursework. We chose the former, beginning virtual coursework in the second week of classes. This had an adverse impact on the class's "chemistry" in the classroom, as both students and instructor had difficulty getting to know each other as a group and as individuals. The result was an extremely low level of in-class questions and discussion and a nearly nil utilization of office hours. A re-organization of the syllabus midway through the course, converting some virtual sessions into face-to-face sessions, brought a marked improvement in the quality of discussion and a somewhat increased student presence in office hours. Pragmatically speaking, it appears that a one-third virtual course is the maximum virtual content that can be sustained without significant harm to face-to-face interaction.

The applet programming section of the course is unique within the course curriculum in that it is designed to be taught almost exclusively virtually — that is, only one in-class session is dedicated to Java, with the bulk of the teaching accomplished in seven virtual sessions. The 'micro-exercise' basis of the module, with its alternation of small explications and reinforcing exercises, makes this a feasible proposition. Many students are successful working within this framework, completing the exercises in timely fashion and performing well on written exams. But this approach relies heavily on students' ability to motivate themselves and to find the material inherently interesting, on their willingness to work with the guidance of a computer rather than a human instructor, and, to some extent, on their predisposition to think algorithmically. Students who do not match this profile struggle to understand the purpose of the lessons and the instructions given with the exercises, find themselves at a loss to begin tackling the exercises themselves, and have difficulty developing a framework in which they can understand the feedback from WebToTeach. The course itself, of course, is intended (indeed, it is required) for non-majors who in general enter the class with

low motivation, with the result that many of them are not well-prepared to take on the challenges posed by WebToTeach.

Distance learning offers a dual promise: it provides a "learner-centered" or "active learning" environment, and it saves instructional labor. Our experiences suggest that these concepts are not compatible with each other. WebToTeach offers an innovative way for students to learn about programming; indeed, many students are able to grasp the necessary concepts quickly and completely. The students are certainly at the center of this project: they are able to re-submit and re-think their work as many times as necessary; feedback from the system is designed specifically to enhance this process. But the instructor's role becomes larger as well: not only must she craft each exercise, its tests, and its feedback messages; she must also maintain a vigilant virtual presence, augmenting automated feedback by detecting confusion and misunderstanding, interacting with students individually as well as with the class. Creating active learning in this course relies not only on innovative technology but also committed motivation, guidance, and instruction.

REFERENCES

- [1] Arnow, D. and Barshay, O. "WebToTeach: An Interactive Focused Programming Exercise System," *Proceedings of the 1999 Frontiers in Education Conference*, November 1999.
- [2] Dexter, S. "On Automated Checking of Java Applets," *Proceedings of the 2000 Consortium for Computing in Small Colleges Northeast Regional Conference*, April 2000.