Experiences Using Tablet PCs in a Programming Laboratory

Stephen H. Edwards and N. Dwight Barnette
Virginia Tech, Dept. of Computer Science
660 McBryde Hall, Mail Stop 0106
Blacksburg, VA 24061 USA
+1 540 231 6931
edwards@cs.vt.edu, net@vt.edu

ABSTRACT

This experience report describes lessons learned using first generation tablet PCs to support active learning in an undergraduate computer science laboratory course. We learned that tablet PCs are poorly matched to typical CS laboratory tasks: writing, compiling, and testing programs. Pen-based input is inadequate for typical program editing tasks, and a pen is less effective than a mouse when typing at a keyboard. Students show a clear preference for desktop computers in this environment. Nearly three quarters of our students preferred a lab supporting wireless connectivity, however. Students also believe that the use of movable, reconfigurable furniture allows them to work in arrangements that are more natural during lab. Overall, students preferred the flexibility provided by wireless network access, freedom from cables, and movable furniture, but felt tablets were ineffective for programming tasks.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education; C.5.3 [**Computer System Implementation**]: Microcomputers—portable devices.

General Terms

Human Factors

Keywords

Tablet, pen, wireless, programming, reconfigurable lab, furniture, CS1.

1. INTRODUCTION

Wirelessly enabled tablet PCs offer the promise of portable, any-time anywhere accessibility with the convenience of pen-based input and note-taking. There is growing interest in incorporating this new technology into active learning classroom experiences [3] as well as enhancing lecture presentations [2]. Some educators are even experimenting with pen-based markup and grading of student work by teaching assistants [5].

Thanks to a hardware donation from Microsoft, we were able to set up a laboratory classroom equipped with tablet PCs for use in freshman CS1 courses. In fall 2003, we began investigating the impact that tablet PCs have in a computer science lab setting. This experience report describes the lessons learned using first generation tablet PCs to support active learning in an undergraduate computer science laboratory course.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGITE'04, October 28–30, 2004, Salt Lake City, Utah, USA.
Copyright 2004 ACM 1-58113-936-5/04/0010...$5.00.

2. CONTEXT AND GOALS

Virginia Tech is currently in the process of overhauling its core undergraduate curriculum to infuse more effective teaching practices and improve the effectiveness of our teaching. Some of the changes being incorporated include the use of lab-based teaching across the freshman year, the use of pair programming in closed lab sessions, the adoption of an aggressive objects-first pedagogy, and the inclusion of test-driven development from the first lab assignment. We also have begun moving away from a traditional lecture mode of delivery and toward more active, participatory teaching approaches.

In addition to efforts to incorporate new teaching strategies, we have also begun examining how the environment and equipment used in the classroom can augment the learning process. This exploration of tablet PCs is part of a longer term strategy to evolve our teaching laboratories into highly flexible, adaptive spaces that can serve many teaching styles, and that can also serve as the hub for a broader learning environment without walls or physical space constraints.

We began down this path by placing a pool of twenty tablet PCs in a non-traditional room furnished with movable, reconfigurable seating and tables with casters. This arrangement provides the ability to switch from lecture-style seating, to small discussion groups, to a lab format, to anything else, without equipment getting in the way. Students can push the tables and rearrange their seats so that they can work together. They have the freedom to arrange themselves in a way most conducive to their learning and comfort. No desktop machines or wires restrict this arrangement. Wireless tablets allow lab assistants freedom to move about the space. Students do not have their view obstructed by rows of desktop machines, and each can easily take notes or make annotations as they follow along with the group.

In this alternative classroom, students—or a pair working together—check out a wireless-enabled tablet PC for each lab ses-

sion. These tablets served as **highly mobile access points** running on batteries**,** rather than conventional personal computers. Student accounts within this learning environment are maintained on a separate server and files are accessed transparently across the network. A student's desktop can be accessed from any wireless tablet, from his or her own wireless notebook, or over the internet, making their personal environment transportable among devices. An individual's file space is even uniformly accessible from both Windows and Unix machines. In the future, we envision adding additional support to provide access to server-hosted Windows software through Windows Terminal Services, and allow students to remotely connect to our Unix servers. Students with wireless notebooks then could seamlessly "join" our departmental lab environment easily, whether they are physically located in a laboratory, attending a course lecture in a different building, or studying in the library.

We began exploring the use of such a tablet-equipped classroom in our introductory programming course (CS1) for computer science majors. The CS1 course at Virginia Tech, CS 1705: Introduction to Object-oriented Design I, is taught in a two lecture hour, two lab hour format. In fall 2003, approximately 200 students enrolled in this course were divided into two lecture sections and eight closed lab sections. The two lecture sections both met in lecture halls with wireless support, where the instructor used a tablet for presentation and students were allowed to bring their own wireless-enabled devices. Four of the closed lab sessions met in a traditional computer-equipped classroom for weekly instruction, with regular desktop computers arranged on tables in rows. The remaining four lab sections met in the non-traditional tablet classroom. During registration and course selection, students were unaware of the use of tablets in this course, and were unaware of which lab sections were using tablets or desktops. As a result, students did not self-select their treatment condition, providing a better approximation to a random condition assignment. All of the students were surveyed to collect their subjective experiences at the conclusion of the course. This paper describes the experiences gained during one semester of teaching in this environment.

3. EXPERIENCES DURING LECTURE

Educators involved in the University of Washington's Classroom Presenter project [2] or its earlier Microsoft predecessor have already described experiences in using tablets to enhance the delivery of traditional PowerPoint-style lectures. Classroom Presenter focuses on supporting the instructor's ability to use digital ink to write directly on PowerPoint slides being presented from a tablet, and the ability of students using wireless devices to provide feedback to the speaker in real time during the presentation. However, one of our teaching strategies was to move away from lecture-based delivery. As a result, we did not rely on PowerPoint presentations as our primary delivery mode in CS1.

Instead, we chose to teach lecture classes in the form of discussion-oriented "live" programming example sessions. Students had weekly reading assignments they were to complete in advance of class. For each lecture meeting, we designed a suitably sized example programming task that used or exemplified the concept to be covered. The corresponding lecture then consisted of a brief review of the important concepts to be explored or applied in that session, a short discussion of the problem at hand, and then an interactive construction of a programming solution to the problem using the same environment and tools that students use in the lab.

This teaching approach significantly altered the lecture experience. Students watched "over the shoulder" as the instructor wrote and discussed a live code example. There was a greater degree of interaction, with students suggesting possible solution strategies, identifying problems they could see, and asking why not do it another way. If a student poses a "what would happen if …" question, the instructor can answer by interactively trying the suggestion on the spot. If the example does not work as expected, the instructor can put the students in control, asking for explanations of the erroneous behavior, suggestions for fixing it, or strategies for tracking down the source of a defect. Our CS1 students use BlueJ as their development environment, which significantly aided our ability to interactively create objects, try out methods, and explore other issues on the fly without spending time writing one-off main programs.

This kind of interactive programming provides many opportunities to demonstrate how to write basic test cases as you develop a program, the value of doing so even in simple student-level programs, and the techniques of practical debugging. Students who bring wireless-enabled devices to lecture are able to download the starting materials for the example from the course web site and follow along, or even try out their own ideas live during the class session. While this teaching strategy does require a bit more preparation and makes the class time less predictable, it results in a livelier, more engaging classroom experience and was preferred by instructors and students alike.

3.1 Writing Programs with a Tablet

Because so much lecture time was devoted to creating, viewing, and modifying program code, we were interested in how effectively a tablet PC would be for this teaching style. Instructors in our CS1 course used the same kind of tablet PCs as found in our tablet classroom: a Toshiba Portégé 3500. The Toshiba unit is a "convertible" tablet with a form factor similar to a lightweight notebook with a traditional clamshell case containing a compact keyboard, as shown in Figure 1. After opening the case, the screen can be rotated and folded back down over the keyboard so the tablet can be used like a clipboard, entering data with a pen. Alternatively, without rotating or folding down the display, it can be used like a more conventional notebook computer. This is in

Figure 1: A "convertible" tablet can also be used as a notebook.

contrast to the more radical "slate"-style tablets that weigh less but require the use of an external or on-screen keyboard when typing instead of using the pen.

For basic programming tasks, the Toshiba performed reasonably as a compact, wireless notebook. For pen-based input, Microsoft's Windows XP Tablet PC Edition provides full support for pen-based textual input into most applications, including BlueJ. Handwriting recognition performs acceptably for both printed and cursive handwriting, without requiring the use of specialized letter shapes like many PDA devices. Researchers have been experimenting with alternative notations for pen-based program input for decades [1], but we used the built-in pen-based text input support provided by the operating system since no special-purpose pen-based code entry applications were available.

Unfortunately, in our experience, pen-based input does not work well for program code entry and editing. The handwriting recognition provided within the operating system appears to be optimized for writing English prose. The system makes heuristic judgments about proper capitalization, word spacing, and placement of punctuation when converting digital ink into ASCII text. If you are writing in a document or report, the heuristics generally work appropriately in translating hand-written characters into appropriate phrases or sentences. However, program text does not follow the normal conventions of free-form prose. For example, "camel case" identifiers (e.g., a `ClassName` or a `methodNameLikeThis`) that consist of a few smaller words run together are typically recognized as a series of words separated by spaces instead of one identifier. Conventions about where white space is used around parentheses or periods often differ between programming style conventions and free-form prose. The lack of a tab key (or its common "shift-tab" dual) to adjust or correct indentation levels quickly is also an issue. Our experience has been that program text entered with the pen requires a significant number of small corrective edits such as deleting or inserting spaces or changing letter capitalization on each individual line. These edits take time, and significantly reduce the speed with which code can be entered. As a result, it appears that the pen as an input device is poorly matched to the task of entering and editing program text, at least with the current generation of recognition software.

3.2 Lecturing With a Tablet

Because pen-based input is a poor match for code editing and much of our lecturing involved live programming, tablets were used in their clamshell-style notebook orientation during our lectures. The instructor used a tablet cord-free, and controlled a separate computer wired to the lecture hall's projection equipment. This approach gave the instructor greater mobility and freedom during class, which was useful. One of the instructors frequently took the tablet out into the audience and sat down with the students, for example. This freedom also allows one to take the tablet out to a student's location and let the student take control, asking him or her to implement a suggestion they have just made or try out a question they have just asked.

Unfortunately, most software development applications are not "ink-aware", and thus are limited in how they can support pen-based input. For example, most code editing tools can receive text-based input via the pen—the operating system uses its built-in handwriting recognition to convert the pen strokes into textual characters and then sends those characters to the application. However, one cannot directly annotate code by drawing lines, circling, underlining, etc., unless the application can deal with digital ink in its native form. While some tools have this capability—for example, Classroom Presenter allows you to directly ink on PowerPoint-style slides during a live presentation—most software development tools do not. Because of our teaching style, the tablet's pen was reduced to a simple pointing device during lecture. Overall, pen-based input provided little value in our classroom.

3.3 Wireless Lecturing

While pen-based input was of questionable value, students and faculty alike enjoyed wireless support in the classroom. The extra mobility provided to the instructor was a welcome change from being glued to the podium, and offered new opportunities for involving students directly in the teaching process. Further, those students who did have wireless devices were encouraged to bring them and to follow along. Eventually, 14% of students adopted the habit of always bringing their wireless notebook to class. We expect this trend to increase significantly, since students in future years will be required to own wireless notebooks by our College of Engineering, something that was not required for the students involved in this study.

Although many instructors fear that students in such a situation will be reading e-mail or playing games during class, this did not appear to be typical behavior in our lectures. Instead, students who brought their own notebooks began asking the instructors to make each lecture's starting materials available electronically so they could follow along. After class, students would occasionally demonstrate errors and ask for assistance right on the spot on their own machines, rather than e-mailing questions later or simply walking away puzzled. In a survey given to the students at the conclusion of the course, many students found that the use of a wireless tablet by the instructor during lecture meetings aided in their understanding.

4. EXPERIENCES IN THE LABORATORY

4.1 Tablets in the Lab

Half of the students attended closed lab sessions in our tablet-equipped laboratory classroom, with the other half using a traditional lab with desktop machines. In each two-hour lab session, students worked in pairs solving a two-part programming problem. Students used pair programming in the lab session [6], with one student serving as the "driver" and controlling the keyboard while the other served as "navigator" and watched for errors and made design suggestions. Half way through the lab period, students would switch roles. Student pairs were assigned so that each student worked with a different partner in each lab session.

Because of the way our lab sessions were structured, student tasks during lab included: using a web browser to read the assignment and look up reference details, using BlueJ to write, compile, test and revise programs, and communicate with a partner to design, develop, assess, and refine a solution. Although the specific programming language and software tools may differ, these basic tasks are common to many lab-based introductory courses.

Unfortunately, through this experience, we learned that tablet PCs are poorly suited to the tasks our students were performing. As

indicated in Section 3.1, pen-based input is not as effective as a keyboard for basic programming tasks. Unsurprisingly, students uniformly used their tablets in a traditional notebook-style clamshell orientation, typing at its built-in keyboard. Further, when the pen is reduced to a simple pointing device, it appears to be less effective and less accurate than a mouse. This is evident, when one considers that to use a pen as a pointing device while typing, one must stop typing, pick up the pen with one hand, point, and then put the pen down again before resuming. As the course progressed, several students began to bring in their own external mice to use in the laboratory, even though the tablet PCs were also equipped with touchpads in addition to their pens. One student went so far as to bring in his own external keyboard as well as a mouse to use with a tablet. At that point, the lab instructor told the student: "If you start bringing in your own monitor, you've gone too far"!

Indeed, screen real estate was also an issue. Because of our pedagogical decision to use pair programming, two students worked at each machine. The Toshiba Portégé has a twelve-inch screen that supports a 1024x768 resolution. For one individual holding the tablet like a clipboard, this screen configuration is just manageable. For two students trying to work together, the small screen is a serious constraint. Because of the tablet's small screen size and small keyboard size, it is difficult for students to work together, particularly when one of them needs to type. While one might expect the small, cordless body to make it easier for two students to push the machine back and forth so both can contribute, students clearly expressed dissatisfaction with this style of machine for CS1 laboratory work.

In the student survey, students expressed a clear preference for the use of desktops—which sport full-size screens, full-size keyboards, and mice—rather than tablets in this environment. Students unanimously agreed that an external mouse attached to a tablet PC is necessary for lab work, and 70% agreed that a larger monitor than that of a tablet PC (12 inches, in our case) is necessary for lab work. Compared to desktop machines, 86% of students agreed that tablet PCs are more difficult to use for computer programming lab tasks. Two thirds of the students did not believe that wireless tablet PCs allowed them to work together with a lab partner more effectively. Finally, two thirds of the students who used tablets in lab claimed they were less likely to purchase a tablet PC as the result of experiences in this course, with only one student claiming they were more likely to purchase a tablet.

4.2 A Wireless Lab

As in lecture, experiences during lab with wireless connectivity were much more positive. Unlike lecture, during lab sessions students did not bring their own notebooks. Instead, students used the machines provided in the laboratory exclusively. Wireless access was not used in the traditional desktop laboratory, and was only available in the alternative tablet room.

Students perceived clear advantages to having wireless support. All else being equal, 72% of students would prefer to have lab in a room supporting wireless connectivity. Further, 80% of students found that the use of movable, reconfigurable furniture allows them to work in arrangements that are more natural during lab. Overall, students preferred the flexibility provided by wireless access and movable furniture, but felt tablets were ineffective for programming tasks.

5. EFFECTS ON STUDENT PERFORMANCE

Because only half of the students in our CS1 course attended lab sessions in the tablet classroom, we had an opportunity to compare student performance between tablet users and students using traditional desktop machines during lab. Figure 2 summarizes student performance between the two groups, showing the mean and ranges for cumulative student scores on all lab assignments, on all programming assignments outside of lab, and across all graded work assigned in the course. An analysis of variance was conducted and no significant differences were found between students in the two lab environments, or between students taught by different instructors. Thus, while tablets may have been an inconvenient for students to use on laboratory programming tasks, they do not appear to have had any appreciable effect on student learning or on outcomes.

6. SUMMARY

As part of out curricular redesign efforts, we were afforded an opportunity to experiment with the use of tablet PCs in a freshman-level laboratory programming course. In this experience, we found that both instructors and students considered wireless access in both the lecture hall and in the laboratory to be valuable and to add to the classroom experience. Further, removing the wires and cables from the lab and using movable, easily recon-

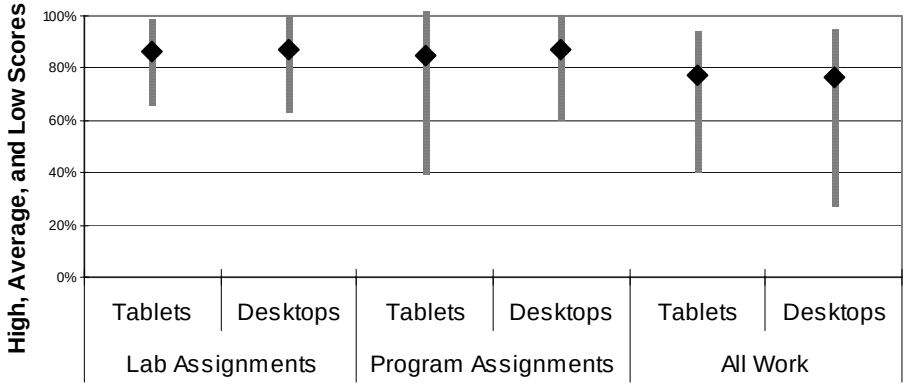

Figure 2: Comparative performance of students on course work.

figurable tables and seating made the lab more conducive to working in pairs and small groups. Overall, students expressed a clear preference for wireless access and the freedom afforded in this new space.

On the other hand, this experience also indicates that tablet PCs are a poor match for typical CS laboratory tasks: writing, compiling, and testing programs. Pen-based input is inadequate for typical program editing tasks, and a pen is less effective as a pointing device than a mouse when typing at a keyboard. Students show a clear preference for desktop computers in this environment. Having students work in pairs may exacerbate these difficulties. Overall, students and instructors felt that tablets were ineffective for lab-based programming tasks.

At the same time, however, our university is moving inexorably toward a point where all incoming students will have wireless, portable computers (not necessarily tablets)—a requirement that incoming computer science freshmen must begin meeting next year. We have seen clear benefits to wireless, portable access in both lecture and lab. In order to move forward in our long-term plan to evolve our teaching laboratories into highly flexible, adaptive spaces serving many teaching styles, we can apply the lessons learned through this experience. For example, we are considering changing the structure of our undergraduate lab facility from its current model: rows of identically configured desktop machines. Instead, we could employ a wireless, cordless work area in the center with flexible, movable furnishings, where the walls are lined with tables containing simple "docking stations" with external flat panel displays, external keyboards, and mice. Students could then bring their own notebook into the facility, and work as-is, or plug in to the provided external devices as needed. All machines will still have transparent access to the shared lab facilities via the wireless network. Such a strategy capitalizes on the portability and convenience of soon-to-be-ubiquitous wireless notebooks, while also addressing their biggest shortcomings for laboratory programming tasks. Further, the reduced hardware cost and maintenance burden such a facility would provide may allow support staff more time to support the required networking and server infrastructure. We are also looking forward to the experiences that other educators have with newer generation tablet PCs. Leveraging tablets effectively hinges on matching the technology to user tasks where pen-based actions are faster and more effective than the alternatives.

7. ACKNOWLEDGMENTS

This work is supported in part by Microsoft Corporation. Any opinions, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of Microsoft.

8. REFERENCES

[1] Anderson, R.H. Programming on a tablet: A proposal for a new notation. In *Proc. Symp. Two-dimensional Man-machine Communication*, ACM Press, 1972, pp. 113-123.

[2] Anderson, R., Anderson, R., Simon, B., Wolfman, S.A., VanDeGrift, T., and Yasuhara, K. Experiences with a tablet PC based lecture presentation system in computer science courses. In *Proc. 35th SIGCSE Technical Symp. Computer Science Education*, ACM, 2004, pp. 56-60.

[3] Berque, D., Bonebright, T., and Whitesell, M. Using pen-based computers across the computer science curriculum. In *Proc. 35th SIGCSE Technical Symp. Computer Science Education*, ACM, 2004, pp. 61-65.

[4] Dray, S., Siegel D., Feldman, E., and Potenza, M. Why do version 1.0 and not release it?: Conducting field trials of the tablet PC. *Interactions*, 9(2):11-16, March, 2002.

[5] Popyack, J.L., and Herrmann, N. Electronic grading: When the tablet is mightier than the pen. *Syllabus*, January 2003, pp. 18-20.

[6] Williams, L., Upchurch, R.L. In support of student pair-programming. In *Proc. 32nd SIGCSE Technical Symp. Computer Science Education*, ACM, 2001, pp. 327-331.