Work In Progress: Developing Tablet PC Animations for Computer Science Courses

Roy P. Pargas and Adam Goodbar
Department of Computer Science, Clemson University
Clemson, SC 29634-0974  {pargas, agoobda}@cs.clemson.edu

Abstract - The Tablet PC can be a tool for developing Ink-based animated software supporting the teaching of computer science concepts. We report on a Tablet PC software development course, called CPSC 481, taught in fall 2005 in which each student developed a large Tablet PC application/animation intended to serve a pedagogical tool in some course. At the end of the semester, the students presented their work in a formal class mini-conference and submitted documentation: a User's Manual and a Technical Reference Manual. We explain how CPSC 481 was taught, briefly describe each of the animations developed by the students, and explain how we will use some of the animations in exercises geared towards teaching concepts in a future data structures and algorithms class. Five students presented their work as poster presentations at WIPTE 2006. The students were enthusiastic in their evaluation as shown in the summary.

Index Terms – Algorithms, Animated data structures, Interactive learning, Tablet PC software development

INTRODUCTION

In the fall 2005 semester, this instructor taught a course, called CPSC 481, on Tablet PC software development. Each student was required to develop a Tablet PC software application/animation that supported the teaching or learning of course material taught in some other course. Some students chose to develop animations for algorithms in a data structures course. Others chose to develop tools to help students in math, statistics, or the Russian language.

All of the applications are interactive. For example, using one application, students can build an arbitrary directed and weighted graph, select a start node and observe Dijkstra’s algorithm develop the shortest path between the start node and all other nodes. Such an animation enables a student to work with the algorithm at his/her own pace and for as long as he/she wants. The animations work on regular computers, i.e., they do not require that the student own a Tablet PC.

The remainder of this paper describes the course and the student projects. We conclude with lessons learned, how the students evaluated CPSC 481, and what we plan to do in the fall 2006 rendition of this course.

COURSE STRUCTURE

CPSC 481 was taught as a 3-credit hour computer science elective to senior undergraduate / first-year graduate students. Students were computer science or computer engineering majors with at least six computer science courses.

The fifteen-week semester was divided into three phases. In Phase 1 (weeks 1-6), the instructor lectured on topics such as: (a) Visual Studio.NET, (b) C#, and (c) APIs specifically designed for Tablet PC software development. Students were given six small (one-week) programming assignments. The primary objective was to provide students with as many C# programming examples involving the Tablet PC API as possible. Jarrett and Su [1] have written the perfect text for this type of a course, one that provides many small and clearly written samples of code, each demonstrating one or another feature or facet of the Tablet PC API.

In Phase 2 (weeks 7-12), the students selected, designed, and implemented a large project. Students (a) picked from a list of project topics provided by the instructor, (b) proposed a solution design, (c) developed a list of software development tasks with completion dates, and (d) implemented each task according to the schedule. Students met with the instructor at least once a week to discuss ideas and to provide evidence of progress. In the final Phase 2 meeting at the end of week 12, the students were expected to demonstrate at minimum a working draft of the complete project.

Phase 3 took place during weeks 13-15 of the semester. At a mini-conference organized by the instructor as part of the course, each student gave a 25-minute presentation of their work including a 10-minute overview, a 10-minute demonstration of their software, and a 5-minute question and answer period. Students dressed in business attire as if they were giving a formal presentation at a conference. The instructor acted as session chair, friends were encouraged to attend, and every effort was made to give the students the experience of presenting a paper at a professional meeting.

Also required in Phase 3 was written documentation: (a) a 5- to 10-page Technical Reference Manual, and (b) a 5- to 10-page User Manual. The former included information that a future student would need to extend the software. The latter was intended to guide a user through the program with screenshots depicting functions provided by the software. There were no quizzes, tests, or final examination.

STUDENT PROJECTS

Seven projects were developed in fall 2005. The target users are presumed to be students learning some course material.

1. **Graph Algorithms**: Students draw graphs labeling nodes and edges, and then run animated graph algorithms, such

---

Abstract - The Tablet PC can be a tool for developing Ink-based animated software supporting the teaching of computer science concepts. We report on a Tablet PC software development course, called CPSC 481, taught in fall 2005 in which each student developed a large Tablet PC application/animation intended to serve a pedagogical tool in some course. At the end of the semester, the students presented their work in a formal class mini-conference and submitted documentation: a User's Manual and a Technical Reference Manual. We explain how CPSC 481 was taught, briefly describe each of the animations developed by the students, and explain how we will use some of the animations in exercises geared towards teaching concepts in a future data structures and algorithms class. Five students presented their work as poster presentations at WIPTE 2006. The students were enthusiastic in their evaluation as shown in the summary.

Index Terms – Algorithms, Animated data structures, Interactive learning, Tablet PC software development

INTRODUCTION

In the fall 2005 semester, this instructor taught a course, called CPSC 481, on Tablet PC software development. Each student was required to develop a Tablet PC software application/animation that supported the teaching or learning of course material taught in some other course. Some students chose to develop animations for algorithms in a data structures course. Others chose to develop tools to help students in math, statistics, or the Russian language.

All of the applications are interactive. For example, using one application, students can build an arbitrary directed and weighted graph, select a start node and observe Dijkstra’s algorithm develop the shortest path between the start node and all other nodes. Such an animation enables a student to work with the algorithm at his/her own pace and for as long as he/she wants. The animations work on regular computers, i.e., they do not require that the student own a Tablet PC.

The remainder of this paper describes the course and the student projects. We conclude with lessons learned, how the students evaluated CPSC 481, and what we plan to do in the fall 2006 rendition of this course.

COURSE STRUCTURE

CPSC 481 was taught as a 3-credit hour computer science elective to senior undergraduate / first-year graduate students. Students were computer science or computer engineering majors with at least six computer science courses.

The fifteen-week semester was divided into three phases. In Phase 1 (weeks 1-6), the instructor lectured on topics such as: (a) Visual Studio.NET, (b) C#, and (c) APIs specifically designed for Tablet PC software development. Students were given six small (one-week) programming assignments. The primary objective was to provide students with as many C# programming examples involving the Tablet PC API as possible. Jarrett and Su [1] have written the perfect text for this type of a course, one that provides many small and clearly written samples of code, each demonstrating one or another feature or facet of the Tablet PC API.

In Phase 2 (weeks 7-12), the students selected, designed, and implemented a large project. Students (a) picked from a list of project topics provided by the instructor, (b) proposed a solution design, (c) developed a list of software development tasks with completion dates, and (d) implemented each task according to the schedule. Students met with the instructor at least once a week to discuss ideas and to provide evidence of progress. In the final Phase 2 meeting at the end of week 12, the students were expected to demonstrate at minimum a working draft of the complete project.

Phase 3 took place during weeks 13-15 of the semester. At a mini-conference organized by the instructor as part of the course, each student gave a 25-minute presentation of their work including a 10-minute overview, a 10-minute demonstration of their software, and a 5-minute question and answer period. Students dressed in business attire as if they were giving a formal presentation at a conference. The instructor acted as session chair, friends were encouraged to attend, and every effort was made to give the students the experience of presenting a paper at a professional meeting.

Also required in Phase 3 was written documentation: (a) a 5- to 10-page Technical Reference Manual, and (b) a 5- to 10-page User Manual. The former included information that a future student would need to extend the software. The latter was intended to guide a user through the program with screenshots depicting functions provided by the software. There were no quizzes, tests, or final examination.

STUDENT PROJECTS

Seven projects were developed in fall 2005. The target users are presumed to be students learning some course material.

1. **Graph Algorithms**: Students draw graphs labeling nodes and edges, and then run animated graph algorithms, such
as Dijkstra’s shortest path algorithm, and minimum spanning tree algorithms (Prim’s or Kruskal’s).

2. B-Tree Animation: Students create empty B-Trees and insert or remove values, observing how the B-Tree expands and contracts in the process. The software also animates how a search for a value is conducted.

3. Finite State Automata: Students build and test deterministic and nondeterministic finite-state automata. The software either accepts a student-specified input string or explains why the string is rejected.

4. Russian Language Tutor: Students practice writing Cyrillic alphabet characters, identify words both in English and in Russian, fill in missing word endings, and categorize Russian phrases. The software can also be used to develop exercises in other languages.

5. Submitting Ink to a Web Application: An web-based classroom interactivity tool called MessageGrid [2,3] was enhanced to allow students to provide simple Ink submissions. Students using MessageGrid can now more naturally submit diagrams of data structures, for example, rather than text descriptions of the data structures.

6. Math and Statistics Teaching Aids: A student can draw an arbitrary function graph and approximate areas under the curve or slopes of tangents to the curve. The software also demonstrates concepts in probability and statistics.

7. MindMaps: A student uses MindMaps to help organize notes and thoughts using diagrams, key words, images, with connecting edges to denote non-linear relationships.

The interested reader can find more information about this course in a WIPTE 2006 paper [4] which focuses solely on the course structure but which does not describe the student projects nor how we plan to use these student projects in the future, as this paper does. All of the student projects were accepted as WIPTE 2006 [5] poster presentations; five (Nos. 1, 3, 4, 6, 7) were actually presented by the students.

CONCLUSIONS AND FUTURE WORK

Projects 1 and 2 (above) and another Tablet PC application animating binary search trees, AVL trees, and red-black trees [6] will be used in a course in data structures and algorithms in the fall 2006 semester. The plan is to assign the students a homework exercise using a Tablet PC animation before discussing the behavior of the data structure in class. Our conjecture is that the ability to interact with these data structures before the lecture on the topic will facilitate student understanding of the material.

We plan to survey the students multiple times during the semester, asking them to compare data structures for which Tablet PC animations are available with those for which animations are not, and whether they feel that the animations help in their understanding of the course material. We will attempt to correlate time spent with the animations with student performance on questions about the data structures.

The Tablet PC software development course was surprisingly easy to teach. The power of the classes and methods provided by the Microsoft Tablet PC SDK coupled with the simplicity and clarity of the text by Jarrett and Su provide a step-by-step guide on developing Tablet PC code.

Eight students completed the course and gave positive evaluations. On a seven-point Likert scale (“Strongly Agree” “Strongly Disagree”), all eight agreed that the format of the course was effective in helping them learn how to program a Tablet PC. The week-by-week bite-sized assignments helped them focus on one or another aspect of the Tablet PC and kept them from being overwhelmed by all of the new programming concepts they had to learn. Seven of eight students said that writing a poster proposal for WIPTE 2006 facilitated and spurred on their project development. Finally six of eight agreed that the in-class evaluation of their assignments helped them learn and did not embarrass them.

This course will be taught again at Clemson in fall 2006. The students in this class will benefit from the availability of all seven projects produced by the fall 2005 students. These projects will serve as programming examples for all.

One project which we will definitely extend is Project #5, extending Ink submission capabilities currently available in MessageGrid. With extended capabilities, for example, MessageGrid will enable: (1) students to submit Ink solutions to problems, (2) instructors to correct student submissions in Ink, and (3) instructors to provide images of a data structure, for example, on which students can write Ink responses to questions. The extension of Ink capabilities to MessageGrid will significantly improve its potential to provide classroom interactivity among instructor and students.

ACKNOWLEDGMENT

This work was supported in part by a 2005 grant from the Microsoft Tablet PC and Computing Initiative (Jane Prey, Program Manager) and by a Tablet PC donation by Hewlett-Packard Company (Robert Reed, University Relations and Wayne Johnson, VP University Relations).

REFERENCES


