Learning Computer Science Concepts using iPhone Applications

Judith O’Rourke, Ian MacDonald, David Goldschmidt
Department of Computer Science
The College of Saint Rose
432 Western Avenue, Albany, New York 12203
{orourkej, macdonai, goldschd}@strose.edu

ABSTRACT
Students use computers and technology at increasingly younger ages. Nonetheless, enrollment in degree-bearing Computer Science programs is not keeping pace with the need for professionals in the field of software development [4]. Even in today’s technology-rich world, Computer Science is still often viewed as an esoteric and cryptic field of study. While students tend to be fluent in using computers and handheld devices, understanding how these devices work and especially how they are programmed largely remains a mystery. To address this problem and attract potential Computer Science students at the middle and high school levels, we have developed iPhone applications that present fundamental Computer Science concepts in the form of a game. In particular, we have implemented Bubble Sort and Binary Search applications. Our approach “talks” to the younger generations by using a medium they are already comfortable with.

INTRODUCTION
Though we live in an abundantly technological age, enrollment in degree-bearing Computer Science programs falls short in producing computing professionals for the American workforce [4]. We find this to be surprising given the widespread use of computers and technology at increasingly younger ages. To introduce Computer Science concepts to future would-be Computer Science students and professionals, we propose a series of instructional games for the iPhone, or more generally, for any programmable mobile device. Figure 1 shows a sample screen shot of a Bubble Sort application.

Our fundamental approach is to reach the younger generations by using a medium they are already comfortable with and even prefer over more traditional classroom tools and software, including a myriad of Web applets and applications.

PROGRAMMING iPHONE OR iTOUCH DEVICES
Programming applications for iPhone and iTouch devices requires an Intel-based Macintosh running Leopard (OS X 10.5.3 or later) and Apple’s free software development kit (SDK). The SDK contains Xcode (3.1.4), which is Apple’s integrated development environment (IDE).
Xcode includes tools for creating, debugging, and compiling Objective-C source code. In particular, Xcode includes the Interface Builder (IB) application that provides a GUI in which elements are dragged from a palette of predefined controls and dropped into a design window. The iPhone Simulator application (see below) is also included in the SDK, allowing programmers to run and test their applications.

Note that Apple’s free SDK does not allow programmers to download applications to an actual mobile device or distribute applications via Apple’s iPhone App store. Instead, a programmer must join Apple’s Developer group for a fee, though this fee is waived for academic institutions. On the plus side, Apple’s Developer group provides many tools and resources to learn and develop iPhone applications, so the financial cost may be appropriate when these benefits are taken into consideration.

**Simulation Environment**

The iPhone Simulator provides a direct means to test mobile applications without having to download to an actual iPhone or iTouch device. Aside from basic functionality, the iPhone Simulator also allows you to simulate rotating the device to the left or right, as well as performing the “shake” gesture.

Though invaluable to the programmer, the iPhone Simulator does currently have some limitations and pitfalls. It cannot simulate the iPhone accelerometer or camera features, which we have therefore avoided. Further, careful attention must be given to mimicking a finger touch (via the click-and-drag of the mouse) when implementing the `touchesMoved()` and `touchesEnded()` methods. While such finger touch movements are more natural on an actual iPhone, simulating the same movements with a mouse are rather unintuitive.

Another limitation of developing iPhone applications is that the development, testing, and deployment must entirely be performed on Apple hardware. Because many schools do not have Apple computers available or determine that such computers are too expensive, this is a hindrance to integrating iPhone applications or their development into the curriculum. To address this problem, writing applications for mobile devices other than iPhone/iTouch can be accomplished using other platforms, including the Java ME platform.

**Programming in Objective-C**

Objective-C is an object-oriented programming language designed by Brad J. Cox in the early 1980s [2]. Objective-C is a combination of the imperative language C (in particular, ANSI C) and the “pure” object-oriented language Smalltalk, one of the first object-oriented languages. Because Mac OS X and the corresponding Cocoa frameworks are written in Objective-C, it is no surprise that iPhone/iTouch applications also use this language.

As with many high-level languages, Objective-C’s corresponding Xcode IDE provides color-coded text editing, syntax error highlighting, auto-completion, and other such beneficial features. Note that Objective-C source code can be programmed and compiled using Xcode or the GNU Objective-C compiler directly in a terminal window.

**A BUBBLE SORT APPLICATION**

Fundamental to Computer Science, bubble sort is often one of the first sorting algorithms students learn. Though certainly not the most efficient sorting algorithm, bubble sort is simple to understand and simple to implement. By using bubble sort, students learn fundamental skills in both algorithm design and programming.

The bubble sort algorithm typically requires a nested loop. Within the inner loop,
each pair of elements is swapped if discovered out of order. For each iteration of the outer loop, a correct element “bubbles” up (or down or right or left) to its final position [1].

Our approach is to teach bubble sort using an interactive interface in which students walk through the algorithm to sort a small list of unsorted numbers. Figure 1 shows the starting point of the Bubble Sort application. When a student clicks the “Start” button, the first pair of numbers (e.g. 7 and 2) “drop down” and are shown in green (see Figure 2).

The student must decide whether to swap the two numbers. To do so, the student drags the “green 2 tile” into the first position where the 7 used to be. To keep a given pair of numbers in the same order, the student must drag either of the green number tiles into their proper position. The application then presents the next pair of numbers (e.g. 7 and 5) and the user keeps going until the list is sorted. Upon completion, the application appears as shown in Figure 3.

If the student attempts to move a green number tile into the wrong position, the green number tile will move back to where the student dragged it from. The expectation is that the student will realize his or her mistake and correct it. In doing so, the mechanics of the algorithm are reinforced. In addition, the student may click on the “How it works...” button to view a description of the algorithm, including pseudo-code.

![Figure 2: First decision point in the Bubble Sort application](image1)

![Figure 3: Successful completion of the Bubble Sort application](image2)

**A BINARY SEARCH APPLICATION**

Before students learn bubble sort, they often study linear search and binary search, gaining an understanding of how much more efficient the binary search algorithm is. Given a list, the problem is to find a target element in the list (or determine that the target element is not in the list at all). The linear search algorithm steps through each element of the list.
until the target is found; if not found, the target has been compared to every element of the list. Therefore, in the worst case, \( n \) comparisons are made, where \( n \) is the size of the list.

A more efficient approach is the binary search algorithm, which requires the list to already be in sorted order. The binary search algorithm uses a binary tree to store all values in sorted order [1]. At each node of the tree, the left branch contains all elements less than the given node; the right branch contains all elements greater than or equal to the given node. For each iteration of the algorithm, if the given node is the target element, the search is successful; otherwise, the search continues “down” either the left or right branch, eliminating up to half of the subtree at each iteration. This process continues until the element is found or a leaf node is reached and no further descent down the tree is possible.

**Making Binary Search Interactive**

Similar to our bubble sort application, our approach is to introduce binary search using an interactive graphical interface. As shown in Figure 4(a), the student is presented a target element to search for (e.g. 39). Initially, the student is not shown the entire binary search tree; instead, the root node of the tree is shown, along with arrows pointing to clickable tags “go left” and “go right.”

At each iteration of the binary search, the student decides whether to follow the right branch or the left branch. For each correct decision, the student sees the next node. From Figure 4(a), when the student correctly selects “go left,” the path from node 47 to node 22 is shown in green (see Figure 4(b)). If the student makes an incorrect choice, the node “shakes,” showing the student that his or her choice was incorrect.

![Figure 4: First steps (a) and (b) of the Binary Search application](image)

Figure 5(a) shows the next step in the example; and Figure 5(b) shows the final step. When the target node is found, the entire tree is revealed.
INTEGRATING iPHONE APPLICATIONS INTO THE CURRICULUM

The key to our work lies in the planning of how to utilize the aforementioned applications in various classroom environments. Our underlying goal is to encourage and enlist students in degree-bearing Computer Science programs at the college level. Since this often begins at the middle and high school levels, we are developing plans to collaborate with such schools to integrate both the use and the programming of iPhone/iTouch applications into the corresponding curricula.

One example of how we plan to accomplish this is to develop summer programs for students and teachers alike, including “teach-the-teacher” sessions. For younger grades, use of our iPhone applications (and other similar apps) may spark early interest in Computer Science, once again through the use of technology already very familiar to such younger generations. Through simple observation, we find many in these younger generations that constantly have their iPhones or mobile devices in hand, communicating with others, playing games (e.g. Sudoku), and the like.

A College-Level Course in Programming Mobile Devices

Also as part of our work, Dr. O’Rourke offers a new course in programming mobile devices (beginning in the Spring 2010 semester). In part, we offer this course to generate interest in Computer Science among non-majors, perhaps “converting” some exploratory or undecided majors into Computer Science majors. The sole prerequisite is one semester of an object-oriented programming course.

While this new course focuses on hands-on iPhone application development, the course also requires students to build interactive learning tools for Computer Science, including sorting algorithms, search algorithms, greedy algorithms, probability, and binomial expansion. Students develop programming skills while learning and reinforcing Computer Science concepts. Further, the developed iPhone applications may be used in other
introductory classes (and in middle and high school programs) to demonstrate Computer Science concepts and encourage more students to study Computer Science.

DISTRIBUTING TO THE Masses
To distribute the applications discussed here, along with future applications, we plan to make them available for free via the iPhone App Store. Students in our introductory Computer Science courses and related service courses will be given direct links to the applications. Those students without an iPhone will be provided opportunities to use college-owned devices, including actual iPhone/iTouch devices and simulation systems. And, as described above, to extend beyond just our college-aged audience, we aim to reach out to local middle and high schools, in part to develop future Computer Science students.

CONCLUSIONS AND FUTURE WORK
Beyond the bubble sort and binary search applications, we plan to develop applications illustrating such concepts as encryption, routing algorithms, bin packing, Sudoku puzzle generation and solving, pairwise sequence (string) alignment, phylogenetic tree building, and many others. Development of these new applications will involve undergraduate students in the aforementioned Programming Mobile Devices course, as well as students from within our graduate program.

While the iPhone continues to enjoy a significant market share, other Java-based mobile devices are beginning to rival this current market leader. Therefore, we also plan to develop Java-based versions for each of our instructional applications.

Finally, with the much-anticipated release of the Apple iPad, future applications may take advantage of a larger screen size and more extensive functionality.

REFERENCES