

Victoria Salekin

McGriff-Bruton Fund

Sewanee: University of the South

September 1, 2010

For five weeks this summer, I did research pertaining to computer science. Specifically, I learned how to write iPhone (and thus iTouch and iPad) applications using Apple's software developing kit, written for Cocoa Touch (the iPhone platform written primarily in Objective-C). Once I had built an adequate knowledge base of C and acquired the necessary experience with the SDK and its templates (layouts for applications), I worked on creating my own application. When I began my research, I already had an idea in mind: an application for athletes that would allow them to enter their maximums on specific lifts (adjusted according to sport or team) and select a week and day of their training cycle before loading the corresponding workout (with the weight to be lifted during each set already calculated for all listed exercises based on the maximums entered by the user).

Over the first week of my internship, I read forums and text on C (the programming language) to acquaint myself with the syntax and commands (my experience was limited to Java at that time). I could not begin to write code and test it on an iPhone simulator until Snow Leopard (the latest Apple operating system) was installed on a computer: the software developing kit (SKD) for writing iPhone, iTouch, and iPad applications was created for machines running on Snow Leopard. Obtaining Snow Leopard proved to be more difficult than anticipated due to the particular log-in system employed by the University in its computer labs: the system is specific to the operating system, so putting Snow Leopard on a single Mac would entail paying for a new license

Victoria Salekin

McGriff-Bruton Fund

Sewanee: University of the South

September 1, 2010

for the log-in application (not to mention updating the operating system) on each Mac designated for student use on campus. This would cost a few thousand dollars before the time required to execute the change were even factored in. As I own a PC, using my own laptop for my research was not an option (Apple does not offer a PC version of the SDK, another tactic to gobble up the market for computers and software). The faculty member I worked with, Dr. Lucia Dale, came up with a solution: she would request that Snow Leopard be installed on the Mac in her office (faculty apparently use a different log-in system in their offices than that in the labs) so we could download the SDK.

Dr. Dale was going out of town during the week in which the software was to be installed, so she entrusted me with the key to her office so that I could get to work. During the second week and part of the third week of my internship, I made my way through the bulk of *Beginning iPhone Development: Exploring the iPhone SDK* and its various projects, omitting a few chapters at the end that would not have any practical application to my own intended project. The first projects were simple enough, involving components such as labels (non-editable text), pickers (the dial-like structure you see in the Timer option in the Clock application on any of Apple's portable devices), alerts (i.e. any notification that pops up onto the display), and buttons. I should point out that by simple I mean primitive: a simple application often entails more work than a non-programmer would likely assume. Buttons, for example, do not magically “do something” when pressed—a program must be notified that an action, e.g. a button press,

Victoria Salekin

McGriff-Bruton Fund

Sewanee: University of the South

September 1, 2010

has occurred and contain information (usually in the form of code) that instructs the program to respond appropriately to the occurrence. Later applications in the book were more complex, incorporating multilevel views—a view is what a user sees on the display—and controllers to manage these views. These applications were key to understanding how to create my own project, which would rely on multiple views: I wanted my application to start with a list (table) of sports and to load a picker with two components (one each for week and day) and fields tailored to the selected sport for the user to enter his or her max efforts on team-specific lifts. From there, I wanted the application to load the customized workout for the particular week and day (selected in the picker). I decided that the cleanest way to build my application (with respect to code) would be to implement a navigation controller, which by default consists of a table that drills down into subviews when an item is selected and includes tabs at the bottom of the display to navigate among the different views. The iPod, of course, is the most obvious implementation of this.

Debugging my applications was especially trying, as the SDK hides some of the underlying mechanisms of each application by delegating work to a program called Interface Builder (IB). In the case that I made a mistake while connecting components of an application in IB (and thus caused some sort of logical error or improper flow of information in my program), determining where the mistake occurred was nearly impossible because IB involves manually dragging the mouse from one component to

Victoria Salekin

McGriff-Bruton Fund

Sewanee: University of the South

September 1, 2010

another (in place of writing code) to create direction-sensitive connections. Java, on the other hand, has no equivalent: everything you see or “get” (including information transfer) is the result of tangible commands typed in the body of the application file.

There were times that I deleted entire applications and rewrote them because I could not find the source of an error. The most difficult aspect of my research was figuring out how to create each layer or subsequent phase of my application when there were no examples of anything similar available: I needed to add a pickerview as a subview to a tableview while using a navigation controller, and the traditional implementation moves from a tableview to another (sub)tableview. The only modification I could find was the addition of a tabview as a subview to a tableview, which was of no use to me. It was only after toiling over the problem for an entire week and through trial and error that I discovered a solution.

Implementation of the final step in creating my application, loading workouts onto a subview from a file contained somewhere within the application itself, altogether eluded me. I tried everything I could imagine, but what I was attempting was something far outside the scope of the book from which I had learned and maybe even above my programming capabilities. There were not any forums on related topics even, and the answers to my biggest questions could only be provided by a high-level computer scientist or programmers with extensive experience with the SDK (the types of people that usually write code as their job). I worked diligently for the remainder of my

Victoria Salekin

McGriff-Bruton Fund

Sewanee: University of the South

September 1, 2010

internship, hoping to complete the application before time ran out, but I made little progress outside of minor modifications to fine tune the program. Although I was disappointed that I was unable to finish the application, I do not feel as if the project were a failure in any way: the complexity of creating the application exceeded my initial expectations and finding solutions to the problems that arose was a feat in itself. I felt an immense amount of satisfaction when I got the navigation controller to work, and this experience has made me truly believe in the value of the process over that of the product. Furthermore, I learned a lot about programming in general and the experience with a new language has proved invaluable in my classes thus far. I suspect it will aid me in graduate school as well if my interest continues and I decide to pursue computer science, applied mathematics, or actuarial studies even.