Chris Weller

CMSI 499: Independent Studies: Introduction To iOS Development

Prof. John David N. Dionisio, PhD

May 5th, 2012

Final Study Report

Motivation for this Course of Study

I have been interested in some form computer programming for a long time, dating back to freshman year of high school when I used to program my TI-84 calculator for math class. I learned Texas Instruments' very rudimentary BASIC language on the calculator in order to create small programs to work out algebra problems that I hated taking the time to mentally figure out. I thought it would be faster to program the calculator to solve the equation and show some of its steps than it would for me to actually sit down and do the work by hand. I realized then that programming was not as easy as it looked (it took me probably twice the amount of time to do the homework) and had a lot more depth than what my calculator could do. Although I never took any formal programming classes, I'm still very much interested in the conceptual and practical sides of programming. Knowing how to execute a task using something simple like AppleScript is handy, but I wanted to gain more of an insight into the more complex world of application and mobile application development. Fortunately I had enough of a background to be able to start learning a new language, but I felt that starting alone would have been a daunting task. Fortunately Steve had been working on building the truck tracking application. Since he had asked me

a few times to generate some graphics for him, I became acquainted with his concept and idea last semester. I approached him about joining in on the process as his graphic designer, but with a desire to join in on the independent study as well. I wanted to learn more about how the programming side of it works so that I could gain a better understanding of not only programming, but of the mentality that goes into the creation and execution of the software that I use every day. Primarily my concern was the visual elements; I was less interested in knowing the very fine details that Steve would need to know in regards to the underlying logic of the application. I looked forward to getting started.

Literature Review

During the beginning of the iOS developing experience, I wanted to be able to catch up at least conceptually to where Steve was able to get practically by the end of last semester. In order to do this, I looked to Lynda.com tutorials to get me started. I selected two specific ones that I watched in full: “Foundations of Programming: Fundamentals”, and “Objective-C Essential Training”, both by Simon Allardice. Foundations of Programming offers a look at programming’s basic concepts from a programming-agnostic viewpoint. Taught in primarily JavaScript, the style of teaching was simple and effective. Simon introduces basic programming fundamentals, such as variables, iteration, if/then statements, and others by writing basic sample code. Although it is taught in one specific language, it really puts an effort into exploring how other languages deal with whatever concept that particular lesson is exploring. It also makes a point to discuss basic syntax and debugging techniques that are common to all

languages so that the new learning developer can keep a perspective on what else is out there. Simon Allardice, the narrator of the tutorials, is a very knowledgeable source of programming information. He is also very good at explaining small details and for really emphasizing the important key take-away notes, and it also helps that he is interesting to listen to and doesn't drone on for hours on end. For me, the set of tutorials gave me a great re-introduction to the concepts that I picked up from the BASIC programming on the calculator. The other Lynda.com tutorial also taught by Allardice, "Objective-C Essential Training", is similar to the Foundations tutorial while focusing specifically on the Objective-C language. It goes into better detail on some of the finer points of Objective-C syntax, such as how to declare a variable and how enumeration works. It even goes into some of the fundamentals of memory management and how to avoid memory leaks. At first it was quite a lot to take in, but it began to make more sense as the semester moved on and I was able to cover the topics in more of a face-to-face discussion. All of the Apple Developer papers are excellent resources, but one in particular was very helpful to me. The *iOS Human Interface Guidelines* document gives an excellent overview of how to think when creating the user experience for the app. From its guidelines on aesthetics (noting consistency and feedback) to giving thoughts on the best orientation for an app's layout, the guidelines really help outline the best way to approach designing an app. These proved very helpful when we came up with what we wanted to see out of the app's visual design. They also helped give me perspective on how effective the apps that I use from day to day are, and how

the smallest detail—such as button size—can make a huge difference in a positive or negative user experience. The Apple Developer documentation also provides a plethora of other links to other specific guideline sets and API implementations that are helpful. These, as well as the invaluable Class Reference libraries, continue to help us.

Survey of Learning

This semester has definitely been a humbling experience for me. Coming into the study I knew that I wasn't very knowledgeable about specifically Objective-C programming, but I thought I had a pretty good handle on the conceptual frameworks of programming itself. What I knew could barely fill a thimble as the saying goes, and I while know for sure that what I have learned is a huge step forward for me, it also revealed the immense depth that the topic has. Although it is a very basic and rudimentary element of programming, learning how to declare a variable in Objective-C and what exactly a pointer is. As one great explanation I read in the forums pointed out, a pointer is like a phone number. It only references the object that is trying to be reached, in this case the person. But by changing the phone number (or the reference location), the pointer doesn't change the person (or object), but only the person that it is looking at. Topics like this were completely new to me coming into the independent study, and have informed me about how little I really knew. Steve has also been a big help in getting me going; he has been able to field many of my silly questions and get me more familiar with the project that he started. Learning about the navigation elements that Apple builds into the platform has

also been interesting. The navigation bar, and its subsequent structure, is the preferred organization for the app. While part of its power is in its ease of application (Apple provided a very helpful class reference guide for it), being able to manipulate the class with code to change out the settings view has been key to how we have been working with our user system. Getting the settings window to alter depending on the user's status is the result of that work, and it has allowed us to better accommodate the software to multiple kinds of users instead of developing different apps for the different kind of user. Working to figure out a system for users to log in helped us figure out how a globally accessed variable can be useful. By setting up the user state as a truck owner, customer, or nil value, we are now able to tailor specific parts of the app to act in a certain specific way depending on who is using it. This has yielded greater control over how we can program the app as well as streamlining it for the kind of user interacting with it. Understanding views has proven to be a challenge, and I admittedly still don't have a complete grasp on them. Conceptually I understand their relationship with the view controller; that the views may not completely understand what its "login" button does, but that the view controller is able to interact with the correct objects to perform the required actions that the button called for. The subtleties of its usage are still a bit of a mystery to me, but I know that it will come with more practice in coding. Some of the most important techniques I picked up during the course revolved around how to troubleshoot. Being able to look at the code and read it how a machine would interpret the statements is a very important skill to have. Since I am still relatively new to the

language, this definitely proves difficult for me at this time, however I recognize its importance and always try it first when attempting to debug the software. Laying little breadcrumbs throughout the process by way of NSLog functions is a great technique for troubleshooting, and is definitely transferable to many languages. These particulars only scratch the surface of what the class has been able to show me.

Most Significant Accomplishments

The application's visual appearance is my single biggest contribution and accomplishment, but not my only one. The visual appearance, of which I'm proud of, consists of a brushed-steel styling of the menus and buttons. The choice of type is to go for a more retro look and feel, something that maybe a diner or airstream might feel right at home with. The chromed edges provide obvious visual cues for separation of the different elements of each view so that the viewer is able to clearly identify the different sections. The buttons are made to appear flat and smooth, but give a deeply pressed appearance when tapped for visual feedback of the touchdown. At present, we have decided to utilize a handful of Apple's standard iOS design elements as we learned about how to manipulate them. Our menu bar/navigation bar are still a form of the stock variety, and we plan on shaping that up to look a lot more integrated once we get more of the core functionality completed. The primary form of altering the appearance is through the .xib files, with some minor coding in regards to the appearance. Working with the system's built in .xib interface design was very helpful, and enabled me to quickly get going on the graphic elements of the

project. Although I feel that this has been my most obvious contribution to the project, I feel that there is still quite a lot to learn in regards to how to program the interface's design with more depth and a richer user experience.

Getting the proper user-state to change and be understood throughout the application's execution took us to the next step in regards to user interaction. Our user's experience has to be able to be modified depending on the type of user, so when we were able to get it to recognize what state it was actually in based on a login screen and global variable, we were very excited. This functionality has paved the way for the usage of truck owner-specific actions and changes to be implemented, such as setting up the schedule, tweeting from inside the app, and reporting the truck's current location. Figuring out the proper structure for getting this to work (customizing the navigation bar's code for different tabs to display) was also a good learning experience.

The Most Significant Challenges

Working on this app has not been easy. For me, the biggest challenge has been trying to catch up with Steve's coding ability and understanding. Since I had no specific background in iOS or Objective-C before the semester began, following along with the discussions that Steve led in the meetings were sometimes quite difficult for me. I have definitely gained more conceptual understanding of the language, but my practical coding knowledge is still fairly weak. This led to some of the concepts going over my head for lack of background and experience, so being able to catch up to and keep up with Steve has been a big challenge. That isn't to say that he doesn't work to help catch me

up; just being a semester behind on the project yielded the trouble. Being brought up to speed on the state of the application was difficult at first. As I learned the mentality behind the basic structure of the application, I began to better understand what needed to be done and how the logic needed to be worked out. Getting to that point of familiarity with the software at first proved to be the hard part. The next challenge is also something that I imagine is the bane of all new programmers: debugging the software! Since learning the syntax has been very tough for me, the debugging side has been equally tough. For example, when I tried to implement a simple UIImage onto a whole view in the menu section of the app, I didn't quite understand the role of the views yet and deleted the view that was there so I could make my image the entire background. Clearly that was not the right thing to do, since the app constantly crashed after that and I couldn't figure out how to get the original view back in order. The failed builds and error messages that ensued were highly frustrating, since I only partially understood the issue. Eventually Steve helped me work it out, and we were able to properly fix the issue while I gained a better understanding of how the views worked. This leads to the other, similar problem that was a constant grief. Confusing error messages. To the experienced programmer, semantic issues and hiding instance variables are small fry problems that only come up once in a while, but to someone like me these small hurdles appear like huge walls. Looking them up online has definitely been useful, as well as the discussions that we have had about them during our meetings. I've begun to be able to recognize and distinguish what some of the common ones are, and I

know that the more I solidify my understanding of basic Objective-C syntax I'll be able to better understand and avoid the frustrating warnings.

Future Design Roadmap

The next steps for the application involve cleaning it up as much as possible. Consolidating code by working in as many method calls as possible is a start, as is a better organization of the application's file structure. The elements of the core functionality are coming to a pretty complete finish, but there is still work to be done. On a design level, the code need to reflect some new properties for the truck objects, specifically storage of images for displaying the trucks' logo on the detail view page, storing an image for the menu, keeping the schedule times stored in the object, etc. Right now we still have a dummy in place to show the basic idea of that particular functionality, so we'll need to figure out exactly all of the properties that we want the truck object to carry with it. Customizing the code for the interface to feel more organic is definitely on my list of things to tackle next. In my opinion, the most successful apps on the market are those that integrate a unique user interface that bases its design on the iOS standard set of interactions but takes the time to completely customize the look for itself. This gives the user a very specific experience: familiarity and freshness. The familiarity comes from the already understood elements that would be emulated: a navigation bar or back button. The freshness comes in applying a new, unique design to those buttons and concepts of organization. I want to learn and implement more animation to the interface. I would like to see simple but effective animated elements, like the login button moving away and

splitting the screen up during the login menu's transition instead of just a wipe. When I get the familiarity of learning these elements, I'll be better prepared to design a more integrative app. Designing a proper settings screen is still required, and we haven't completely discounted the idea of using a settings tab in the general settings of the phone either. Storing the user's information onto the device, as well as a few other specific points of data, still needs to be implemented.

My personal growth in iOS development is still yet to be determined. I know that what I would like is at least a basic understanding of typical logic utilized in Objective-C and enough programming experience to implement it in simple ways. On the design side, I'd love to work towards getting a better handle on how iOS applications use core animation to their advantage and apply that knowledge to my own work. Design is my preferred path, and I know that given the time to learn more about iOS conventions and the Objective-C language I will feel more comfortable with it.

Final Thoughts

The experience of the course has definitely been a positive one. I've thoroughly enjoyed exploring the Objective-C language and applying it to an application as a way of learning it. It has been difficult to pick up such a complex language with little background, and I have definitely felt the struggle. On any other type of course, I typically prefer a one-on-one style of learning because I am usually very intrinsically motivated to learn, however I found that this particular time might not have been the most effective for me. Since my

schedule this final semester of school has been so hectic, I am not sure I was able to give the application as much attention as I would have liked, and honestly feel that I could do a lot more given more time. This is why part of my post-course roadmap involves more in-depth visual work, because I feel I'll be able to devote more time to the project. In a classroom setting, there typically is more drilling and structure, which may benefit my learning style in this type of discipline. I find that having a project of my own is usually a much better motivating factor than simple class assignments, but I wonder if that would be different here. I think that having more of a background in knowing the fundamentals in more of a practical way would have helped me be more creative with my coding. I find myself tripping over the syntax when I understand exactly what the conceptual logic ought to be, and I know that that is something that comes with time and practice. Another thing that I am personally very interested in is using Objective-C in the desktop environment to create applications that have functional use in the film industry. The niche in data management is very new to the film production side of the industry, and it is being filled with effective but overpriced software solutions. Since so many of the old guard in the film industry doesn't fully understand the technology, they tend to embrace and apply the older patterns of thought onto the new gear, which yields a requirement for simplicity and data security. I think that these needs can be easily met with software that considers their needs and is affordable to produce. I would like to at some point branch off into this direction. I have definitely had a good

experience, and I look forward to further exploring the language on this project and others.