Kai Austin

7 May 2014

Software Systems Final Project Report: Objective-C and iOS

Who is in the group?

Our team was made of Rachel Binaz, Aiden McLaughin, and Lyra Silverwolf. We chose to work fairly independently due to the nature of our method of study being at different paces, each of us beginning with the online site Code School and then turning to the book.

What was your learning goals?

Our goal was to learn Objective-C in the context of making an iOS app. My own personal goal was to get a head start into being familiar with the iOS development environment to be the alternate pair to my past experience developing apps in Android (Java).

What did you explore?

Code School and Head First iPhone and iPad Development. Code School served as an introduction to Objective-C and provided an environment for testing out a bit of code. Head First provided as primarily a tutorial for working with XCode, the environment for building iOS apps and simulations of them. Unfortunately, I was unfortunately not able to work directly with Xcode since I do not have (or did not have access to in time) an Apple environment for it to be functional.

How goal was achieved?

Code School served as an introduction to working with Objective-C. It was composed of 5 lessons, each of which revolved around the core workings of Objective-C independent of the iOS environment. It was a bit of a mouthful in the beginning due to the contrast between Objective-C and C, which is what I found the most surprising. For some reason, I had expected something more along the lines of an intermediary “C to Java” language.

I then began reading the Head First Book. It was less about Objective C than I expected, so there was an assumption that I was already familiar with the language. Which I was thanks to Code School. Head First served as a great tutorial for the functioning of XCode. The program itself is fairly robust, so any Objective-C code writing would have been kept to a minimum whether I knew it or not. As mentioned, I did not work directly with XCode, however my prior experience with Android Studio served as a very useful jumping board for understanding what

was going on and I was able to easily pick up how XCode functions and get a general idea of how I would make an iOS app if need be.

Demonstration of success?

Learning Objective-C was a bit of a mouthful initially. I believe this was because I was learning it out of the intended context (i.e. not in an iOS development setting), and then there was my own expectations of it being more similar to C than it was. It was a bit nice that pointers were slightly more enforced; no matter the type, everything was a pointer. In order to get the value out of anything, the basic syntax was, in extremely loose language:

```
[pointerToValue putItHereFunction:placePuttingValueIn]
```

I did find this extremely annoying. However I will say it is one of the better parts of the Objective-C language since I understand its use. The fact that these can end up being nested inside of one another, almost indefinitely is perhaps the most obnoxious part. Creating classes and objects, especially ones which you passed arguments to seemed especially excessive. I was also not too fond of the fact that once you made a class and declared its functions, etc. in the ".h" file , the class and its functions needed to be declared again in the ".m" file. I am mostly saying these because I am a minimalist, so I am fond of "pretty code." Object-C was far from it, and pretty clunky. However, like with all languages, it works for its intended us.

Enough with bashing Objective-C. In terms of iOS development, as mentioned, I have previous experience with working in an Android development environment (Android Studio). Because of this, I was able to pick up the basics of XCode quite easily – though experience would require practice over time to become more comfortable with the environment. Much of working with XCode actually requires minimal Object-C knowledge. Though that was likely because the Head First book was targeted at beginners and programs would become more complex over time beyond the given samples in the book. However, XCode basic use is:

1) Place a controller, such as a button
2) Give it an Event, what to listen for or how the user would interact with the controller, such as a click
3) Give that controller an action for what to do when the event occurs.

Step #1 is pretty much a drag and drop, very visual – which is super useful in terms of design. Like all things with Apple, they are heavy on making you see what you are doing. Step #2 is again, super easy and just a right click to assign an event for whatever type of controller was

placed. Step #3 is the only part which may actually involve code, such as changing the view or making a pop up.

There are other functions as well. For example, suppose we have information in a data structure or some form of database, then again, it is fairly easy to assign their values to objects on the page. Switching views is as easy as telling a button to go to the next page of a “Story board” which is like a visual flow chart of every view the app may have. Meanwhile, much of the Objective-C for setting up these controllers, views, etc. is written for the developer while this is taking place. They merely take care of the minor more customizable details. Once the general basic flow of development is established, oddly making an app just seems to be more about design than actual programming skill.

Seeing how Objective-C was used in XCode helped to clear up some of my initial confusions of the language for why certain things worked the way they did. I.E. Context was extremely appreciated. Knowing this, when I get the chance to start developing an iOS app, I already have the basic ground work and familiarity with the development environment to jump the learning curve and get straight to it.