DEVELOPMENT OF AN iPHONE APPLICATION AND A STUDY OF ITS DESIGN ISSUES

Xiaobing Hou, Richard Schaefer and Racquel Brown, Central Connecticut State University

Abstract

Mobile applications is a fast-growing research area and provides new research opportunities for students in areas of computer science, networking technology and information technology. This paper discusses the development of an Apple iPhone application, which provides an interactive campus map for new students or visitors trying to navigate the campus; or, for existing students, faculty and staff to help them find a building in an obscure corner of the campus which they may never have visited. This work is a faculty-student research project funded by the university. The project successfully promoted student engagement in hands-on research, a High-Impact Practice identified by the American Association of Colleges and Universities. In addition, the project explored some characteristics of Apple-based application design issues. The success of the project also provided a framework for faculty and students to develop similar projects, with components usually available on campus.

Introduction

The mobile phone industry has exploded over the last five years. One estimate is that there are more mobile phones in the U.S. than people [1]. A research project involving faculty and students will help faculty explore new research ideas, and students gain real-world experience and cutting-edge technologies. This project required project formulation, platform and development tools selection, application design, application development and debugging, documentation and presentation. Additionally, various design and development issues were studied. In the future, similar projects could include all of the above or a subset if the same platform and development tools are used.

There are currently only a few mobile phone applications specifically targeted at faculty, staff and student functions and capabilities on each individual campus due to limited user pools, which leaves a large number of hands-on research opportunities for students, as they are familiar with their own daily life and needs. The ability of students to access much of the content and capability of a university website—such as the interactive campus map, faculty and staff directory, online instruction and many other functions from mobile devices—is a significant strategic direction that must be pursued.

The intent of the application developed in this project was to provide a mobile analog to the interactive campus map that utilizes built-in mobile device capabilities for mapping and self-location to guide people to buildings on campus. This application will simplify navigation of the campus and assist people in finding obscure buildings, which they may not have visited previously. The existing physical signage will not help students or visitors locate and navigate to a building that is on the opposite side of the campus from their current location, nor does it provide any indication as to how close the person is to their destination. This application provides both of those capabilities.

Platform

This project targeted the Apple iPhone. While Android-based phones have surpassed iPhones in terms of total market share [2], Android devices are notorious for manufacturer-specific implementations of the Android OS along with differences in application development. The Android phone marketplace is divided among several major manufacturers including HTC, Samsung and even Google with their acquisition of Motorola. The iPhone market is monolithic, meaning that all iPhones are essentially the same. Almost all iPhones run the same version of the iPhone OS (iOS). All iPhone apps are distributed through a tightly controlled Apple App Store. There are currently more than 425,000 applications in the iPhone App Store with over 15 billion downloads [3].

The development tools for Apple iPhone are:

- Apple Xcode - an Object-Oriented superset of C++ which includes methods for invoking iPhone and iOS internal functionality
- HTML/CSS/JavaScript - standard web application development tools that do not have the capability to easily integrate with iPhone/iOS capabilities
- Independent third-party development environments such as MonoTouch [4]. These environments generally contain the same capabilities as Apple Xcode.

This project utilized Apple Xcode for development of the proposed application. This decision was motivated primarily

by the cost of third-party tools at the time the application was written. Xcode includes a mature Integrated Developer Environment (IDE) and, when combined with the iPhone Software Development Kit (SDK), provides native access to iPhone capabilities including mapping, data management, screen management and application management. Third-party applications are fairly expensive. MonoTouch retails for $399 and has no advertised educational or student discount.

Design

The project utilized iPhone/iOS MapKit services to determine the user's current location. MapKit services display a satellite or birds-eye view map of the campus, centered on the user's location. A selector, not yet determined, will be used to select the building to be located. A database of buildings and their spatial locations will be used to place a marker on the map to indicate the location of the chosen building. The marker will be interactive, enabling display of basic information about the building.

The interactive nature of MapKit services enables the map to be updated in real-time as the user navigates the campus in order to ensure that he/she is proceeding in the correct direction. Opening another application on the phone will terminate the mapping application and reduce battery drain caused by the use of the internal GPS required for MapKit services.

iPhone applications constructed using the Xcode IDE and the iPhone SDK utilize Objective C, a proprietary extension of C++, and a Model-View-Controller (MVC) software pattern. The basic design of the application is represented in the diagram of Figure 1.

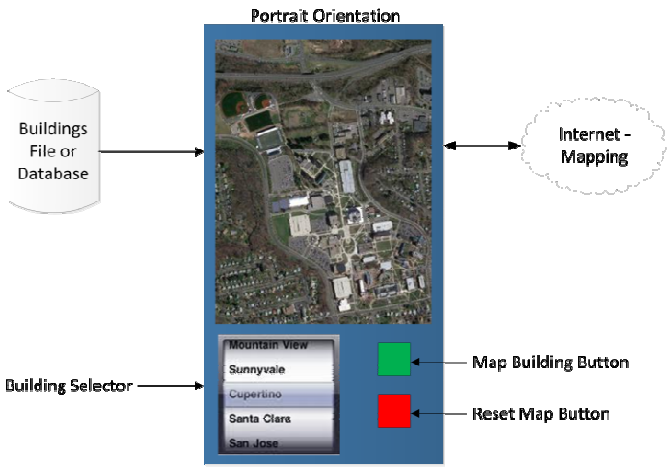

Figure 1. Design Diagram

The application was designed to suppress the selector and buttons when the phone is rotated to Landscape mode in order to increase the size of the map providing more detail, as shown in Figure 2. The application was developed on a Mac and tested via a direct download of the application to the iPhone via a USB connection.

Figure 2. Landscape Mode of the Design

After completing this design, some choices were made based on the results of prototyping and lessons learned about the intricacies of the iPhone SDK:

- The application was designed using the Storyboard visual design interface of Xcode. This limits the application to iOS Version 5 or greater. This decision was supported by statistics showing that as of November, 2011, 60% of all iPhones were using Version 5 or greater. This will only increase over the short-term since all new iPhones are sold with the latest version of iOS (5.0.1 or greater).
- The User Interface utilizes two screens rather than one with a "fly-over" modal pop-up. This change was required due to the limitations of the iOS SDK for iPhone in which all Views display by default in full-screen mode.
- A Property List (Plist) was chosen as the data store for the building data. This choice was made over other data storage options. The text/CSV file option requires significantly more coding to load the data and instantiate the data source for the selector (Picker). SQLLite supported by iOS has no capability for batch loading of data and would have required the creation of an application just to provide a data entry function to build the SQLLite database. iOS also supports a built-in framework called Core Data. This

framework is essentially a wrapper around SQLLite that exposes the back-end database as a set of classes with appropriate methods and properties. Core Data handles persistence, transactions and advanced error handling. Core Data also has no methods for importing data to initialize the database so a secondary data management application would have had to be designed and written.

- The map was designed to support selection of only one building at a time and zooms to include a minimum amount of geography in order to display the user's position and the selected building. If the user is not located on the campus, a notification is displayed and the auto-zoom feature is disabled. The minimum geography displayed in these conditions is the entire campus. The map can be manually zoomed via gestures on the iPhone screen.

Architecture

Application Lifecycle

iPhone applications are governed by the Model-View-Controller architecture, as noted above. These applications are also event-based applications. The application lifecycle is shown in Figure 3 [5].

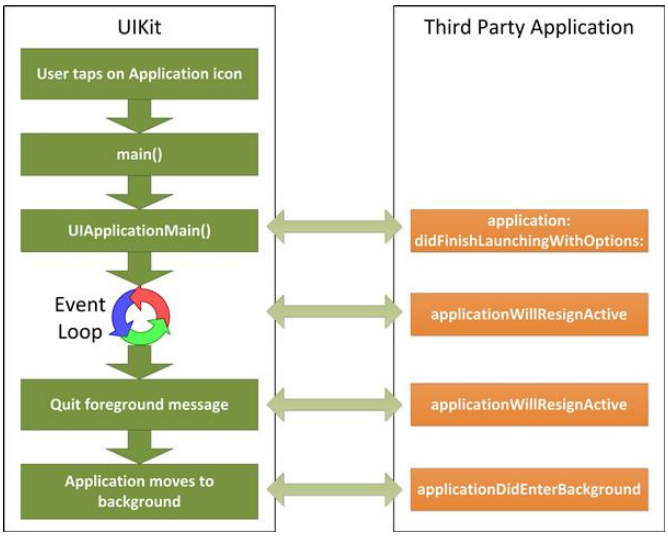

Figure 3. Typical Application Life Circle

iOS detects the user tapping the application icon. The application is loaded ("main()") and initialized. This fires the "didFinishLaunchingWithOptions" method in the Application Delegate where the developer can implement application-specific initialization code. The Interactive Map application uses this method to instantiate and initialize application-scope variables such as the Building List. Once fully initialized, including the user code in the "didFinishLaunchingWithOptions" method, the application waits for event messages ("HandleEvent").

The application is responsible for establishing the appropriate functions to handle event messages, not limited to application events, but also external events such as device rotation, pressing the "Home" button or other events. Event handling is the primary responsibility of the current View and the controls within that View. Views are managed via a Stack. An event that causes a new View to assume control of the Window pushes the current View down on the Stack and places the new View at the top of the Stack. An event replacing the new View with another new View repeats this process. An event returning to a previous View pops the current View off the Stack making the previous View current again. All of the View management methods are invoked for each of these operations, i.e. "viewDidBecomeActive" for a new View and "viewWillDisappear" for a View being replaced.

The user pressing the Home button or any other action, whether user-initiated or system-initiated (e.g., an incoming phone call), that causes the application to give up its ownership of foreground processing fires the "applicationWillResignActive" and "applicationDidEnterBackground" methods in the Application Delegate. These methods enable the programmer to take actions that would be appropriate to support or deny background processing for the application. Theoretically, the "applicationWillResignActive" method could be used to terminate the application if background multi-tasking processing is not desired. This is not necessary in version 5 of iOS or later. A configuration option in the "info.plist" file for the application, "UIApplicationExitsOnSuspend", can be specified that tells iOS the application is to be terminated completely if it is removed from foreground processing. The Interactive Map application was configured to terminate if it resigns control of foreground execution. This results in significant power savings since Location Services updates are no longer in effect and the GPS is turned off.

First Responder and Delegation

The basic structure of the iOS application utilizes two concepts known as "First Responder" and "delegation" to provide the functions to process event messages as well as to extend the iOS object model. The First Responder is the module that loaded the currently active view. The First Responder is the default handler for all event messages that occur while the view is instantiated.

The Delegate is a construct that is used in place of subclassing and inheritance in order to enable extending the methods available to an object. A Delegate implements a Protocol which defines the requirements to communicate with the Delegate. This includes any methods supported to handle event messages. The basic Delegate implemented in all applications is the Application Delegate or App Delegate. The App Delegate handles all events at the application level and can also be used to instantiate and initialize application-level data elements and structures. Delegates are also implemented in iOS for View Controllers and Navigation Controllers, two constructs used in this application. The full protocol for each Delegate is documented in the iOS Developer Center at Apple.com. [6]

Persisting Data

The arguments over methods for persisting data within the application for use by multiple Views and Delegates within the application resemble the religious war between Microsoft and Apple fanatics. There are three basic techniques for persisting data:

- Global Variables
- App Delegate Variables and Objects
- Singleton Classes

Simple applications might use a Global Variable defined in the project.pch file. This practice is generally frowned upon for more than single-view prototype or demonstration applications, since it is not thread-safe nor does it ascribe to the ARC memory management model. Many Developers recommend the use of objects and variables defined in the App Delegate. There is only one App Delegate, its scope is application-wide and it invokes the Singleton pattern making it thread-safe. Some argue that the App Delegate should be left to the responsibilities defined for that delegate protocol, handling initialization, termination, memory errors, etc. It is important to note also that since the App Delegate is the first user code executed, placing a large amount of variable initialization in the App Delegate slows down application startup and affects user experience.

The use of a Singleton class specifically designed to define and manage variables as objects within the Singleton with appropriate getters and setters is widely discussed and has its own support base. The Singleton model can be made thread-safe but is not so by default. Care must be taken to ensure thread-safety in the coding of the Singleton class definition. Also, the use of a Singleton class requires more coding for simple variables than just the definition of the variable. Since these variables are properties of the Singleton class, it is necessary to create the appropriate getters and setters. The Xcode @synthesize command can be used when the Singleton is instantiated in a module to automatically generate primitive getters and setters for basic variables, but more complex objects such as arrays, dictionaries and other classes must be fully coded in the Singleton class definition.

This application used the App Delegate model for global data. There were only a couple of variables to be initialized to default values, and the array of building dictionaries which was loaded directly from the Buildings.plist file. This did not materially affect application initialization performance. The majority of the application initialization is involved in determining the user's location and initializing the map, activities which take place in the first View loaded, not the App Delegate.

Structure and Function

The application was developed using Xcode 4.3 and the current iOS SDK. The Storyboard shown in Figure 4 depicts the basic structure of the application.

Figure 4. Storyboard

This Storyboard shows two View Controllers embedded in a Navigation View Controller. The Navigation View Controller automatically provides the forward (Buildings) and back (Map) buttons and the transition between the two Views. The connection between the two View Controllers is called a "segue" and enables properties and methods that can be used to control the transition. Much of the definition of the View Controllers, the Navigation View Controller and the segue are defined in properties in the Xcode IDE, eliminating dozens of lines of code required in previous versions of the SDK and down-level applications that support iOS versions prior to Version 5.

The primary View Controller is the Mapview. This is the first View displayed when opening the application. A screen-capture JPG file is displayed until the application is fully initialized. Once the application is initialized, a real-time map of the campus is displayed. If the user's location is

within the bounds of the campus map, the location is displayed on the map. If the user is not located on the campus, an alert message is displayed and the auto-zoom functionality of the application is disabled.

Once the map is displayed, the user can then select a building by pressing the Buildings button. This transitions to the Dataview View Controller using the segue displayed on the Storyboard, the arrow connecting the Mapview and the Dataview. This segue is defined as a modal push segue, which means the new View is in control of the screen and cannot be dismissed by other than the appropriate button (modal); at this point, the new View is pushed onto a stack of views maintained by the OS. Returning to the original View is accomplished by popping the current View off the view stack and displaying the new "current" View.

Once on the Select Building screen, the user spins the selection control, a UIPickerView in iOS terminology, to select the desired building. The selected building is displayed in a text field below the Picker. Once the desired building is selected, the user returns to the map via the Map button on the Select Building screen. This pops the View from the stack and causes the original Mapview to be displayed.

Upon return to the Mapview, the application checks to see if a building was selected. An alert is displayed if no building was selected. If a building is found, the appropriate positional data is retrieved from the array of buildings and a pin is dropped onto the map. If the user's current location is on the campus, the auto-zoom code in the application determines a new set of bounds based on a rectangle constructed using the user's location and the building's location, then zooms the map to that rectangle. This provides greater detail for user navigation from their position to the building. The map is fully capable for panning and zooming and, while in motion, the user's location is updated every 45 seconds. This enables the user to track their progress towards the intended building in real-time, limited by the accuracy of the current user location.

A screenshot of the implemented application is shown in Figure 5, where the user's position is represented by the blue dot and the selected building is represented by the red pin. The blue circle surrounding the dot shows the relative accuracy estimate. For more information on the accuracy of iPhone locations, see the study by Zandbergen [7].

Data Gathering

The application requires the following properties for each Building: Building Name, Latitude and Longitude. The data are gathered using a HyperText Application (HTA), essentially an HTML page that runs locally on the user's desktop. This HTA displays a Google map of the campus and a drop-down of the buildings. The process is to select a building from the drop-down, locate the building on the map and click on it with the mouse. This would generate an entry in a textbox containing the three building properties. The latitude and longitude are retrieved from Google via map interaction implemented in Javascript. The building data is copied from the textbox, saved as a text file, imported into an Excel spreadsheet and used to generate the XML for the Buildings.Plist file.

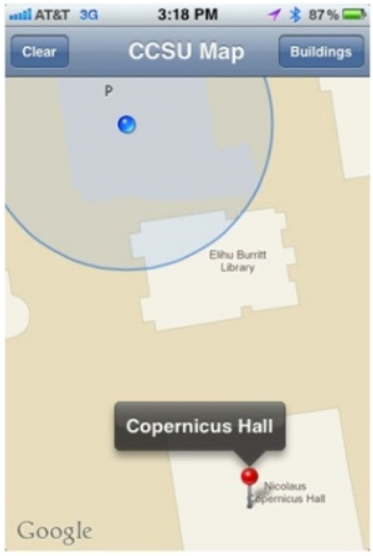

Figure 5. Screenshot of the Application

More Design Issues

Application design patterns that apply to most application platforms do not necessarily apply to Apple-based platforms. Any attempt to defend a design choice by Apple-only developers—based on previous experience or academic knowledge of what are considered standard design patterns—results in an avalanche of negative comments and little help from the Apple community. Their mantra is to "forget your tired old techniques and embrace the platform as Apple wrote it".

One example of this was a request to the developer community for assistance in locating the event which would trap the Back (Map) button in the Navigation Controller Delegate to enable enforcement of a business rule that a building must be selected before returning to the map. The concept that the Back button would be intercepted and not allowed

to perform its designed function was met with a number of criticisms of the overall design.

Ultimately, one experienced developer did intervene in the piling on of abuse and pointed out that Delegate and Protocol appeared to meet our needs. However, a small caveat at the bottom of the Delegate definition negated the entire ability of the Delegate to trap the Back-button event. There is a way to programmatically replace the Back button with an application-generated button and trap that button's click event, but this was discovered much too late in the development process and would have required a complete redesign of the application. One of the more experienced developers submitted a feature request to Apple to enable the Navigation Delegate protocol to support an event for the Back button in order to enable enforcement of business rules prior to honoring the request to navigate back to the previous screen.

Other issues arise in using the Apple-specific development tools (i.e., Xcode) for iPhone/iPad development. Each new release of iOS requires a new version of Xcode. One cannot upgrade iOS on the target device without installing the latest version of Xcode and vice-versa. New or changed APIs in each release of iOS are then immediately applicable to the application. One example of the impact of how this affects application development is the use of iPhone Maps in the application. The original application was written based on the iPhone Maps provided by Google Maps. The release of iOS 6 replaces the underlying Google Maps with an Apple-internal map function. This could result in changes to the Map Service APIs, and empirical evidence demonstrates significant degradation of map quality with the initial release of iOS 6 Apple Maps. This could directly affect the application developed in this project. No other specific methods used in this current project have changed in iOS 6 in any way that would affect the functionality of the program.

Many companies are taking an approach that uses hybrid technologies such as Mono, described earlier in this paper, to attempt to mitigate the impact of iOS changes by replacing native Apple applications with hybrid HTML5/ Javascript/Web Service applications using Javascript libraries that can access the device's internal functions (e.g., dialing, GPS, etc.) but selectively choose alternative technologies (e.g., Google Maps) where appropriate. This breaks down Apple's "walled garden" to a certain extent, offering the developer more flexibility in application development, and the end-user more stability with respect to iOS changes. Were these technologies more mature, and available at reasonable or no cost for educational purposes, at the time this project was undertaken this approach probably would have been chosen rather than Apple's native development environment. This would have enabled the use of iOS Location Services to determine the user's location, rather than web-service-based calls to Google Maps to provide the mapping services independent of iOS. This would have insulated the application from the iOS Map changes.

Conclusions

This project provided an opportunity for faculty and two graduate students to develop a mobile application and investigate the possibility of creating a framework for more similar projects. The application developed provides a useful tool for students and visitors to find buildings on campus, regardless of the user's location or the building's location. It also provides a handy reference to the campus when the user is not on the premises as it will still allow selection and location of buildings and the map may be manually zoomed using standard iPhone gestures to show additional map detail. The project, if published via the Apple App Store and updated regularly with new building data, could be a valuable tool for students and visitors to the campus.

In addition to the interactive map application developed here, more achievements have been obtained from this project:

- This project successfully solicited a faculty-student research fund from the university, which provided a Mac computer to partially support the application development.
- Two graduate students got involved in this project and fulfilled their capstone project requirements.
- In addition to successfully developing a mobile application, the more valuable achievement was several design and development issues were discovered and studied during the course of the project. Students would not be able to get such experience and knowledge without this project.

The success of this project demonstrates the feasibility of a framework of student hands-on research projects on mobile application development. Similar projects can be developed for undergraduate or graduate capstone projects. Using similar devices and application development tools, different mobile applications can be developed. The applications can be realistic if faculty and students collaborate with various departments or organizations on or off campus. The following are the components required for such projects:

- Devices – An Apple Mac computer (Mac Pro, MacBook Pro, Mac mini, etc.) is required for programming and an iPhone for testing.
- Development tools - Apple Xcode is used to invoke iPhone and iOS internal functionality, and HTML/

CSS/JavaScript for web application development. Optionally, independent third-party development environments such as MonoTouch can be used instead of Apple Xcode.

- Curriculum – Before starting the research work, students should be knowledgeable on areas including collaborative project development, information systems in business, research skills, Internet technology, foundations in computer science for arrays and dictionary used to store the building data and stacks used in view management, and software engineering for the design and development processes.
- Collaboration - To target a real-life application, it is critical for the team to work with users in various departments or student groups on campus or local business. One advantage is that the students know themselves very well. Another one is that mobile applications is a new and fast-growing market. Therefore, such applications are in great need. Students can easily find users and develop a mobile app for free or at very low cost.

References

[1] CTIA. (n.d.). CTIA Wireless Quick Facts. Retrieved October 6, 2011, from http://www.ctia.org/advocacy/research/ index.cfm/aid/10323.

[2] Gartner. (2011, April 7). Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012. Retrieved October 6, 2011, from http://www.gartner.com/it/page.jsp?id=1622614.

[3] Apple. (2011, July 7). Apple's App Store Downloads Top 15 Billion. Retrieved from http://www.apple.com/pr/library/2011/07/07Apples-App-Store-Downloads-Top-15-Billion.html

[4] Xamarin. (n.d.) Retrieved October 6, 2011 from: http://xamarin.com/

[5] Adi Saxena. (2010, October 29). Application's Life Cycle in iOS4. Retrieved from: http://www.codeproject.com/Articles/121681/Application-s-Life-Cycle-in-iOS4

[6] Apple Corporation. (n.d.). iOS 5: Understanding Location Services. Retrieved October 25, 2011, from Apple Developer Library: http://support.apple. com/kb/HT4995

[7] Zandbergen, P. A. (2009). Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. *Transactions in GIS, 13*(s1): 5–26.

Biographies

XIAOBING HOU is currently an Assistant Professor at the Computer Electronics and Graphics Technology Department at Central Connecticut State University. He received the Ph.D. degree in Information Science from the University of Pittsburgh in 2006. Dr. Hou's teaching and research interests are in the area of computer networking and information security. He is a member of IEEE and ACM. Dr. Hou may be reached at xhou@ccsu.edu

RICHARD SCHAEFER is a recent MS graduate of the Computer Information Technology program at Central Connecticut State University. Richard currently works for Hartford Financial Services Group as an Enterprise Solutions Architect. He may be reached at rschaeferiii@gmail.com

RACQUEL BROWN received her A.S. in Electronic Engineering Technology from University of Hartford in 2003, a B.S. in Computer Engineering Technology from University of Hartford in 2005, and an M.S in Computer Information Technology from Central Connecticut State University in 2012. She is currently a Data Center IT Analyst at United Technologies Corporation where she improves IT Compliance throughout UTC. She also assists business unit customers with IT projects, global Data Center strategy efforts, and is an administrator of Web Hosting applications. Her interests include Network Engineering and IT Security. She may be reached at rabrownred@yahoo.com