Advanced Programming for Mobile Devices - Software Report

James Donohue - james.donohue@bbc.co.uk

Contents

1 Abstract

2 Introduction

This software report accompanies the completed iOS app *FindMyBike*. It continues some of the discussion begun in the proposal already submitted.

The Xcode source code for *FindMyBike* can be found in the zip file provided with this document.

2.1 Specifications

The development environment used for the app adheres to the requirements of the assignment brief:

- Xcode 8
- Swift 3.1
- iOS 10.3 SDK
- Platforms: iPhone 6 and later (iOS simulator or physical device)

The following Apple frameworks were used in building the application:

- CoreLocation
- CoreBluetooth
- UserNotifications
- UIKit
- Foundation

No other third-party libraries or frameworks were used.

3 User guide

TBC

3.1 Evaluation

A key goal in the design of the application was simplicity and clarity. The Apple Human Interface Guidelines (2017) state that 'clarity' is one of the themes that differentiates iOS from other platforms, defining it in terms of legibility of text and 'a sharpened focus on functionality'.

One example of applying this theme in practice is the content and layout of the cells in the main table view. Rather than specifying a particular size and weight for the text in the `UILabel` components, predefined iOS text styles of 'Headline' and 'Subhead' were used. This enables the platform's Dynamic Type mechanism to respond to user preferences and accessibility settings (Apple, 2016) and thereby ensure legibility.

TODO: add figure?

The principle also applies to the choice of what level of detail is appropriate for each level of navigation. For example, the table view cells could have included the iBeacon settings for each bike (UUID, major and minor) but this would have created a more cluttered user interface without adding much value. Instead it was decided to only show these values when the user actually edits their bike details.

Another usability factor that was considered was discoverability, meaning the level to which a new user can determine what actions can be performed and how to perform them. The right chevron displayed alongside the 'My Bike' cell is used as a visual hint that the user can tap the cell in order to view more information. The chevron (known as a 'disclosure indicator' in iOS parlance) signals what Norman (Norman, 2013) terms an 'affordance', namely the ability to access a more detailed view. Note that while other ways of signalling this affordance are possible, by using the convention of the standard table view disclosure indicator, the user can apply the knowledge they have learned from using other iOS apps. This relates to the stated Apple design principle of 'consistency', specificaly that an app should implement "familiar standards and paradigms by using system-provided interface elements" (Apple, 2017).

3.2 Development issues

3.2.1 Delegation

The classes provided by the Cocoa frameworks make extensive use of the *delegation* pattern "in which one object in a program acts on behalf of, or in coordination with, another object" (Apple, 2015). This

is in contrast to a model where behaviour is inherited (and may be overridden) from class to subclass. Inheritance in object-oriented programming creates a tight coupling between a parent class and its subclass and breaks encapsulation, whereas composition enforces a 'black box' approach based on well-defined interfaces, and is therefore usually preferred (Gamma et al., 1994).

Although delegation allows complex behaviours to be composed dynamically at run time, one price (as Gamma et. al highlight, p.21) is that it make code harder to understand. This is evidenced in the app in the relationship between `MainViewController`, `RangingTableViewController` and `EditBikeController`. Since these classes only maintain a weak reference to an instance of the user's bike, for it to be persisted it must be passed "up" through two levels of composition to the `BikeRegistry` class, via the `BikeChangedDelegate` protocol. If the app evolved to include a large number of additional controllers this approach could become excessively complex.

3.2.2 Nerd

So (Keur and Hillegass, 2015)

3.3 Marketing

3.3.1 Network effects

There are two, slightly different features offered by *FindMyBike*:

- it allows a user to detect their proximity to their own bike, and therefore may allow them to locate it (for example, if they have forgotten where they parked it).
- it allows users to know if their missing bike has been detected by other users of the app, and therefore possibly recover it in the event of theft

The first feature above does not depend on any other app users. However, the effectiveness of the second feature is dependent on the number of users with the app installed that frequent the area of the missing bike. This feature is likely to be more useful in densely-populated areas (if a missing bike is in a desert, it is unlikely that many app users will enter its beacon region). But even if a large city, the usefulness of this feature is in proportion to the number of active app users. Each new user that installs the app in a densely-populated area increases the value of the app for other users, a phenomenon known as a 'positive network externality'(Shapiro and Varian, 1999), or network effect, which was first observed in communication technologies.

Aquiring enough users for this network effect to apply can be seen as a 'chicken-and-egg' problem. One way of resolving it is to ensure that the 'single-user' benefits are compelling enough if there are no other users of the network (Choudary, 2014). By marketing the app's 'single-user' benefits first, it is hoped that enough users will start to use it that the second use becomes viable and eventually becomes more compelling than the 'single-user' mode.

References

Apple (2015) *Cocoa Core Competencies: Delegation*, Available at: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Delegation.html (Accessed: 22

September 2017).

Apple (2017) *Human Interface Guidelines: iOS Design Themes*, Available at: https://developer.apple.com/ios/human-interface-guidelines/overview/themes/ (Accessed: 22 September 2017).

Apple (2016) *Text Programming Guide for iOS: Using Text Kit to Draw and Manage Text*, Available at: https://developer.apple.com/library/content/documentation/StringsTextFonts/Conceptual/TextAndWebiPhoneOS/CustomTextProcessing/CustomTextProcessing.html#//apple_ref/doc/uid/TP40009542-CH4-SW65 (Accessed: 22 September 2017).

Choudary, S. P. (2014) *Building the Next WhatsApp or Instagram: The Network Effect Playbook*, Available at: https://www.wired.com/insights/2014/03/building-next-whatsapp-instagram-network-effect-playbook/ (Accessed: 22 September 2017).

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) *Design Patterns: Elements of Reusable Object-Oriented Software*, Upper Saddle River, NJ, Addison-Wesley.

Keur, C. and Hillegass, A. (2015) *iOS Programming: The Big Nerd Ranch Guide*, 5th Edition, Atlanta, GA, Big Nerd Ranch.

Norman, D. (2013) *The Design of Everyday Things*, Revised and expanded edition, New York, NY, Basic Books.

Shapiro, C. and Varian, H. R. (1999) *Information Rules: A Strategic Guide to the Network Economy*, Boston, MA, Harvard Business School Press.