Übungsblatt 3

Abgabe

Geben Sie Ihre Lösung zu Aufgabe 3 und 4 bis zum 7.11.2012 12:00 Uhr in Uniworx ab. Alle Aufgaben sind einzeln zu bearbeiten.

Aufgabe 1

Lesen Sie Kapitel 6 (User Experience Guidelines) der Apple Mobile Human Interface Guidelines.

Aufgabe 2

Unter http://goo.gl/P0A8r steht ein gemeinsames Google Doc bereit. Tragen Sie sich in die Tips & Tricks Tabelle ein, um Ihren Kommilitonen einen Tip zur iOS Programmierung zu geben (z.B. zu NSLog, XCode Shortcuts...). Zunächst brauchen dies nur zwei Kursteilnehmer für nächste Woche machen (FCFS).

Aufgabe 3 (für Master-Studenten)

Machen Sie sich mit der Funktionalität von View Rotation in der Klasse UIViewController vertraut.

- Beschreiben Sie, wie die Anwendung auf eine Änderung der Geräteorientierung reagieren kann.
- Wann ist es sinnvoll, die möglichen Orientierungen in der Anwendung einzuschränken und wann nicht (z.b. nur Portrait oder nur Landscape)?
- Welche Unterschiede gibt es in den Design Guidelines zwischen iPhone und iPad?

Beschreiben Sie diese Aspekte und fügen Sie evtl. Abbildungen hinzu. Geben Sie für diese Aufgabe eine PDF ab (max. 1 Seite).

Aufgabe 4

Erstellen Sie eine iPhone App mit Tab Bar, Web View und Map View anhand der folgenden Schritte.

- Erstellen Sie ein neues XCode-Projekt (Tabbed Application, iPhone, Storyboards, ARC).
- Schauen Sie sich das Storyboard an und machen Sie sich mit dessen Elementen vertraut. Verschaffen Sie sich in der Dokumentation einen Überblick über die Klassen UITabBarController, UITabBar und UITabBarItem.
- Machen Sie die AppDelegate zu einem UITabBarControllerDelegate und fügen Sie in der didFinishLaunching...-Methode folgendes ein:

```
UITabBarController *tbc = (UITabBarController*) self.window.rootViewController;
tbc.delegate = self;
```

- Implementieren Sie in der AppDelegate außerdem die Methode didSelectViewController so, dass als Debug-Ausgabe (NSLog) der Index des aktuell ausgewählten View Controllers angezeigt wird.
- Ändern Sie im Storyboard die Beschriftung der Tab Bar Items von “First” zu “Web” und von “Second” zu “Maps”. Löschen Sie bis auf das Tab Bar Item alle Elemente der beiden View Controller. (Optional: Erstellen Sie eigene PNGs für die Tab Bar Items.)
- Ziehen Sie eine Web View auf den First View Controller. Legen Sie anschließend ein IBOutlet für diese Web View an. Fügen Sie der Web View durch folgende Zeilen statischen Inhalt hinzu und lassen Sie die Anwendung laufen:

```
NSString *html = @"Bacon ipsum dolor sit amet leberkas laboris chicken.";

[self.myWebView loadHTMLString:html baseURL:nil];
```

- Zeigen Sie nun eine wirkliche Webseite an. Kommentieren Sie dazu zunächst die beiden Zeilen wieder aus. Erzeugen Sie dann ein NSURL-Objekt für eine Webseite Ihrer Wahl, sowie ein NSURLRequest-Objekt. Finden Sie eine geeignete Methode der Klasse UIWebView, um die Seite anzuzeigen.
- (Optional) Verkleinern Sie die Web View und fügen Sie dem First View Controller zwei Buttons hinzu, um auf der Webseite vorwärts und zurück gehen zu können.
- Fügen Sie im Storyboard dem Second View Controller eine Map View hinzu und aktivieren Sie im Attributes Inspector der Map View “Shows User Location”. Legen Sie ein IBOutlet an und importieren Sie im Second View Controller <MapKit/MKMapView.h>. Fügen Sie Ihrem XCode-Projekt außerdem das MapKit hinzu (Projektübersicht > Build Phases > Link binary with libraries > + > MapKit.framework).
- Lassen Sie die Anwendung laufen. Im iOS Simulator kann die Location unter Debug > Location > Custom Location gesetzt werden.
- Zeigen Sie jetzt nicht die aktuelle User Location, sonders setzen Sie die Location im Code auf den Marienplatz oder einen anderen Ort Ihrer Wahl.
- (Optional) Erstellen Sie einen einen dritten View Controller und verknüpfen ihn mit dem TabBarController. Erzeugen Sie zwei Textfelder, in die man Koordinaten eintragen kann. Die Map View zeigt dann eine Location basierend auf diesen Koordinaten an.

Hinweis: Seit XCode 4.4 (Juli 2012) gibt es einige Neuerungen in der Objective-C Programmierung. Achten Sie bitte auf folgende Dinge, falls Sie mit Tutorials arbeiten, die älter sind:

- Properties: Neben @property in .h war früher zusätzlich ein @synthesize in .m notwendig, damit automatisch Getter/Setter erstellt werden. Dies geschieht nun automatisch durch den Compiler (Auto-Synthesis).
- Syntax: Genauso wie man NSStrings durch @”Hello World” initialisieren kann, können nun auch NSNumbers, NSDictionaries und NSArrays erstellt werden (z.B. @[“Yellow”, “Green”, “Red”]). Siehe http://mobile.tutsplus.com/tutorials/iphone/objective-c-literals/.
- Methoden: Die Reihenfolge, in der Methoden in .m implementiert werden, ist nun egal.