Overview

Create three simple apps to test new functionality.

- Part 1: Introduction to CoreLocation
- Part 2: Introduction to Local Notification
- Part 3: Introduction to Push Notification
- Part 4: Begin your Final Project

Submission

Lab evaluation form
Online Lab feedback form

Note: If you are unable to complete the lab during this week's lab period, please be ready to demonstrate it at the beginning of the next lab.

Part 1. Introduction to CoreLocation

The CoreLocation framework creates an abstraction around the user's location (whether determined from WiFi, cellular towers, or GPS and the iPhone's compass).

CLLocationManager Class Reference

Step 1) Create a new project called **CoreLocation** using the View-based application template.
Step 2) Add the **CoreLocation Framework** to your project.
Step 3) Import the CoreLocation / CoreLocation.h header .
Step 4) Add the **CLLocationManagerDelegate** protocol to your view controller.
Step 5) Create a class instance variable named **manager** of type **CLLocationManager ***.
Step 6) In the **viewDidLoad** method:

1. Allocate and initialize **manager**.
2. Set the delegate property of the **manager** to the view controller.
3. Call **startUpdatingLocation** on **manager**.

Step 7) Implement the methods of the **CLLocationManagerDelegate** protocol.
You should be accustom to Ctrl-clicking (right-clicking) on the delegate protocol, and selecting **Jump to Definition**, but the methods of the protocol you must implement are listed below for your convenience:
- (void)locationManager:didUpdateToLocation:fromLocation:
- (void)locationManager:didFailWithError:

Step 8) Use the CoreLocation framework to display the user's location. You are free to choose any display method (NSLog to console, UILabel, adding an annotation to an instance of MKMapView, etc). You are encouraged use your app from Lab 8 and add push pins onto a MKMapView.

Step 9) Test your application **on the device**. Once you call startUpdatingLocation, the locationManager:didUpdateToLocation:fromLocation: method should start running. This method will run every time a new location is received (by default), or when you move out of a certain range in meters (set by the **distanceFilter** property of the CLLocationManager).

Step 10) Demo your test app to the TA.

Part 2: Introduction to Local Notification

Local notifications are essentially used to create an alarm that can be displayed even when your application is not running (or in the background). Additionally, if your app is in the background, you could display an alert if some activity occurs in order to bring attention to your app.

Step 1) Review the documentation on Sending Local Notifications, then create a new project called **EggTimer**:

Notification Documentation

Step 2) Create an instance of UILocalNotification and set it to go off in the near future (ex. 10 seconds)

Step 3) Schedule the notification. You'll find example code in the documentation link above. Also, the following method may be the easiest way for you to create a NSDate to give as an argument for the setter of the fireDate property.

+ (NSDate *) dateWithTimeIntervalSinceNow:(NSInteger) seconds;

Step 4) Test your app. Make sure that after you schedule your Local Notification, you **press the home button** so your **app is in the background**. After 10 seconds, an alert window should appear.

Step 5) Update your test app so it is an egg timer application that allows the user to set a timer (in seconds). Your app should also have a start button. When the user presses start, set up a NSTimer to decrement the display every second and also fire a Local Notification when time has run out.

Step 6) Demo your egg timer app to the TA.

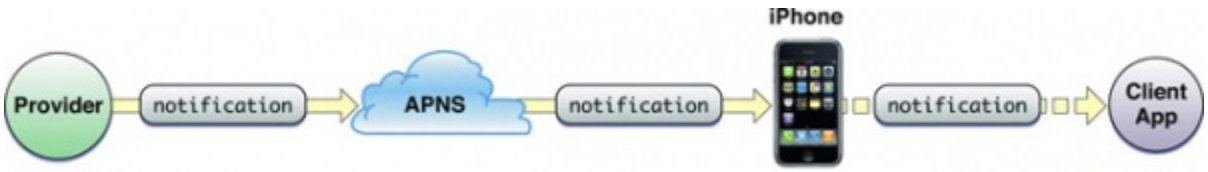
Notes: To remove the badge on your app icon, you could send a new notification with badge=0, or in code:

[UIApplication sharedApplication].applicationIconBadgeNumber = 0;

Part 3: Introduction to Push Notification

Push Notifications is a way for a 3rd party (a provider) to alert the user, even when your application is not running. Common uses of Push Notifications are the badge icon on the E-mail app increasing, or when the Facebook app alerts you that someone sent you a message. There are a few steps to make your App ready for Push Notifications.

1) In the App Delegate, register for Push Notifications with APNS (Apple Push Notification Service).
2) Once the device contacts APNS, you'll receive a Device Token (a unique ID for the device).
3) At some point, have the provider send a Push Notification over SSL to the device with a given Device Token.



Step 1) Push Notification was not enabled in the mobile profile you downloaded in Lab 1. You'll need to download the certs again (and install them). Create a new project called Push and make sure the company identifier is edu.iastate.cpre388 (This will ensure the **Bundle ID** for your project is: **edu.iastate.cpre388.Push**)

Step 2) Prepare your device. The device will need a Wifi connection, so register it with the IASTATE network if you have not done so already. Check in the Xcode organizer that the Push Mobile Profile is on your device.

Step 3) Register for Push Notification in the **App Delegate**. Since this usually follows a similar procedure for any app that implements Push Notifications, a sample implementation is listed below:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Register for Push Notification
    UIRemoteNotificationType types = UIRemoteNotificationTypeSound | UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeAlert;
    [application registerForRemoteNotificationTypes:types];

    // Show window
    [window makeKeyAndVisible];
    return YES;
}

- (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error {
    NSLog(@"Error: [%@]", error.description);
}

- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
     NSLog(@"Device Token: [%@]", deviceToken);
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo {
    NSLog(@"Push Received while application was running");
}
```

Step 4) Run the app **on the device**. Push Notification will not work in the simulator.
Step 5) Once you receive the device token, then you can run one of the following server-side scripts in a browser to send a Push Notification to your device:

http://www.chaddington.com/push.php?deviceToken=YOUR_DEVICE_TOKEN
http://www.chaddington.com/push2.php?deviceToken=YOUR_DEVICE_TOKEN&alert=hello&badge=5&sound=default

Replace YOUR_DEVICE_TOKEN with the 40 character hex string given in the NSLog. Do not put spaces in your Device Token, and make sure to run this script while your application is in the background.

Step 5) Demo your test app to the TA.

Part 4: Begin Your Final Project

Read the Final Project document on the course website. Start to develop your project concept, create a rubric and schedule, and discuss it with the TA.

Your rubic and schedule will be due with Lab 10.