Multitasking

iOS App Development
Fall 2010 — Lecture 24

Questions?

Announcements

- iOS 4.2.1 out yesterday
  - Finally brings multitasking to the iPad
  - Welcome to use for final project if you like
  - Lab will remain at 4.1 for the rest of the semester
- Final project — progress report
  - Due tomorrow night by 11:59pm
- No class Thursday — Happy Thanksgiving!

Today's Topics

- iOS 2.x-3.x lifecycle
- iOS 4.x lifecycle
- Multitasking in iOS 4.x

Notes

- I'm showing the relevant portions of the view controller interfaces and implementations in these notes
- Remember to release relevant memory in the -dealloc methods — they are not shown here
- You will also need to wire up outlets and actions in IB
- Where delegates or data sources are used, they too require wiring in IB

Back in the Day
(iOS 2.x–3.x)

No Multitasking

- One of the original complaints of the iPhone in the 2.x and 3.x days was that the device didn't support multitasking
- Not quite true...
  - iOS has been a fully pre-emptive multitasking operating system since its release
  - However, 3rd party apps were not allowed to continue running when another app was launched

Apple's Rationale

- Didn't necessarily feel that multitasking was appropriate for a mobile device with a small screen
- Running apps in the background causes a greater use of the CPU and as such consumes more power which drains the battery faster
  - Users might perceive poor battery life
- Possible for the responsiveness of the foreground app to suffer if a backgrounded app was eating up CPU cycles
  - Users might perceive poor app/device performance

iOS 4.0 & Later

Multitasking Services

- Fast app switching
  - Resume quickly, preserve state
- Push notifications
  - Respond to a notification from a remote server
- Local notifications
  - Push-style notification delivered at a predetermined time
- Task completion
  - Extra time to complete a task

Multitasking Services

- Background audio
  - Play audio content to the user while in the background
- Location — Navigation
  - Keep users continuously informed of their location
- Location — Significant location change, region monitoring
  - Respond to location changes while in the background
- Voice over IP
  - Make and receive phone calls using an Internet connection

Detecting if Multitasking is Available

- You can reliably determine if multitasking support is available based on whether or not UIDevice responds to the isMultitaskingSupported selector...

```
UIDevice* device = [UIDevice currentDevice];
BOOL backgroundSupported = NO;
if ([device respondsToSelector:@selector(isMultitaskingSupported)]) {
  backgroundSupported = device.multitaskingSupported;
}
```

- You might use this code to first check if backgrounding is supported and if so, wrap multitasking-related code inside conditional statements
- To simplify things in these slides, we'll assume it's available

Simulator Support

✔ Fast app switching

✘ Push notifications

✔ Local notifications

✔ Task completion

✘ Background audio

✘ Location — Navigation

✘ Location — Significant location change, region monitoring

✘ Voice over IP

Fast App Switching

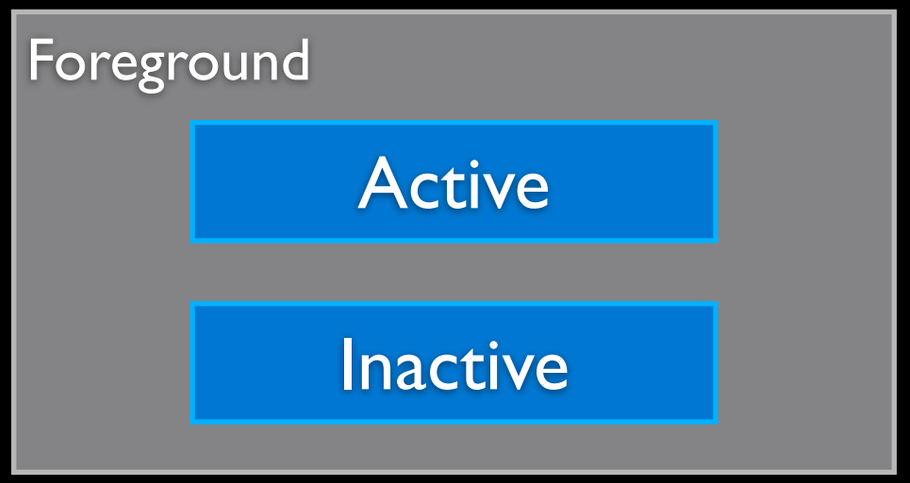
App Lifecycle — Before iOS 4.0

Active

Inactive

Not Running

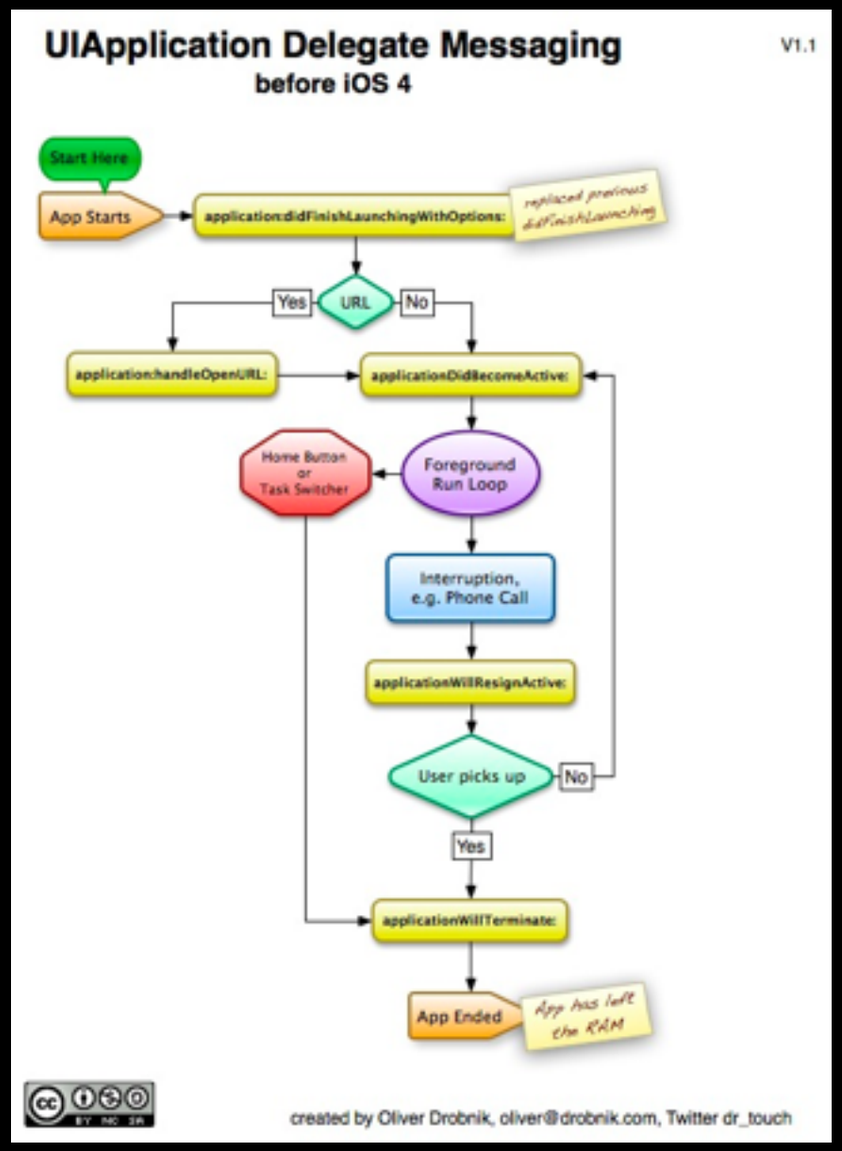
App Lifecycle — iOS 4.0 or Later

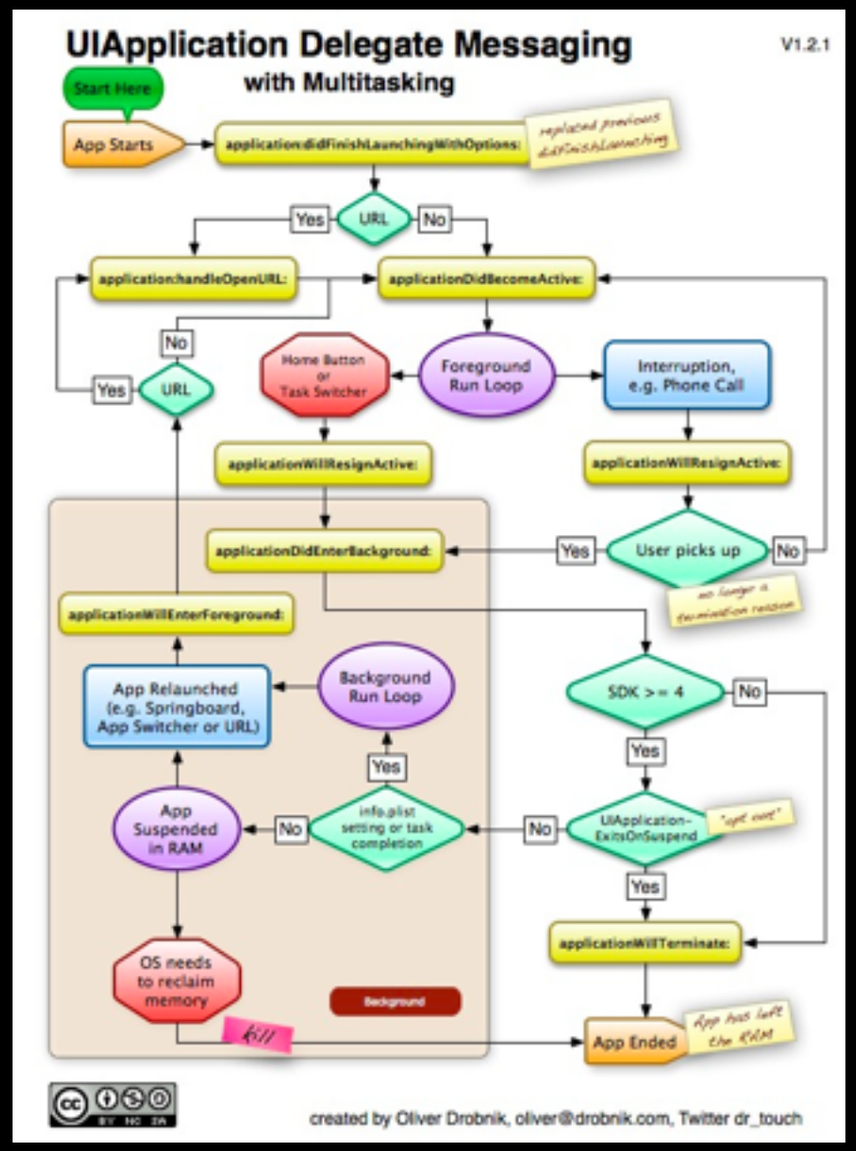


WWDC 2010 Session — Adopting Multitasking on iPhone OS, Part 1

Complete Flowcharts

- http://www.drobnik.com/touch/2010/07/understanding-ios-4-backgrounding-and-delegate-messaging/





Fast App Switching

- By default if you build against iOS 4.x you get fast app switching for free
- If you would like to opt out of fast app switching, set the UIApplicationExitsOnSuspend key to true in your app's Info.plist

Key	Value
▼ Information Property List	(13 items)
Application does not run in background	☑
Localization native development region	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone environment	☑
Main nib file base name	MainWindow

Being a Responsible Multitasking App

- When in the background...
  - Do not make any OpenGL ES
  - Cancel any Bonjour-related services
  - Be prepared for network-based connection failures
  - Save your app state before moving to the background
  - Release any unneeded memory
  - Stop using shared system resources

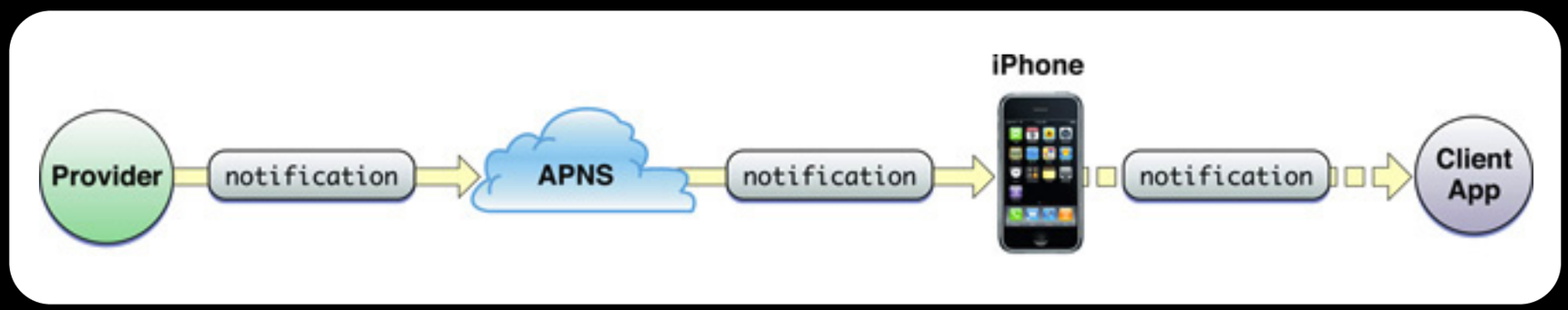
Being a Responsible Multitasking App

- When in the background...
  - Avoid updating your windows and views
  - Respond to connect and disconnect notifications for external accessories
  - Clean up resources for active alerts when moving to the background
  - Remove sensitive information from views before moving to the background
  - Do minimal work while running in the background.

Push Notifications

Push Notifications

- Push notifications are delivered to iOS clients though Apple's Push Notification Service (APNS)
- Typically you have a server (known as the provider) where notifications originate based on the logic of your application
- Sent to APNS where it's delivered to iOS where it is delivered to the user and/or app



Notification Payload

- Each notification contains an associated payload
- This payload specifies how a user should be alerted to the incoming notification
- Designed to be high-performance notification system, thus the payload size is small
  - Must be no larger than 256 bytes in size
- Providers must compose and deliver a valid JavaScript Object Notation (JSON) dictionary object

JSON

- Subset of JavaScript language to represent basic types
  - Numbers, booleans, arrays, dictionaries/objects & null
  - Maps fairly cleanly into most languages

```
{
    "Image": {
        "Width":  800,
        "Height": 600,
        "Title":  "View from 15th Floor",
        "Thumbnail": {
            "Url":    "http://www.example.com/image/481989943",
            "Height": 125,
            "Width":  "100"
        },
        "IDs": [116, 943, 234, 38793]
    }
}
```

- See RFC 4627 or json.org for detailed information

Types of Notifications

Badges

Alerts

Sounds

Paid Developers Only

- In order to utilize the APNS, you must be a member of one of the paid iOS developer programs
  - As such, we won't explore this option in class any further
- Setup steps
  - Generation and installation of an SSL certificate and keys
  - Creation of provisioning profile
- Notification steps
  - Establish TLS connection with APNS
  - Compose and push messages
  - Obtain feedback

Local Notifications

UILocalNotification

- Local notifications are instances of the UILocalNotification class
- This class has several types of properties which can be categorized as follows...
- Schedule time — the date and time (and repeatability) of the notification
- Notification type — enough data to deliver the notification (e.g. badge number, title & message of the alert, or name of sound to play)
- Custom data — can also provide a dictionary of custom data what can be read by the app

UIApplication

- UIApplication provides several methods for managing notifications...
  - Add a local notification
  - Get a listing of all pending local notifications
  - Cancel a the specified local notification
  - Cancel all pending local notifications

```
- (void)scheduleLocalNotification:(UILocalNotification *)notification;
- (void)cancelLocalNotification:(UILocalNotification *)notification;
- (void)cancelAllLocalNotifications;
- (NSArray *)scheduledLocalNotifications;
```

The General Process

- Regardless of the type of local notification provided, the general flow for creating a notification is as follows...

```
UILocalNotification *notification = [[[UILocalNotification alloc] init] autorelease];
if (notification) {

  notification.fireDate = /* some NSDate instance */;
  notification.timeZone = [NSTimeZone defaultTimeZone];

  // configure notification based on type (e.g. badge, alert, sound)

  // optionally provide a dictionary of custom data to be used by app
  notification.userInfo = /* some NSDictionary instance*/;

  [[UIApplication sharedApplication] scheduleLocalNotification:notification];

}
```

Badges

- To display the notification as a badge number, simply set the following the property on UILocalNotification...

```
// 0 means no change. defaults to 0
@property(nonatomic) NSInteger applicationIconBadgeNumber;
```

- To clear the badge, you must utilize a similar property off of UIApplication

```
// set to 0 to hide. default is 0
@property(nonatomic) NSInteger applicationIconBadgeNumber;
```

- Note that this UIApplication property can be used create a badge whenever you want — it's not limited to notifications

Alerts

- To display the notification as an alert, at a minimum you must set the alertBody property of UILocalNotification...

```objc
// defaults to nil. pass a string or localized string key to show an alert
@property(nonatomic,copy) NSString *alertBody;

// defaults to YES. pass NO to hide launching button/slider
@property(nonatomic) BOOL hasAction;

// used in UIAlert button or 'slide to unlock...' slider in place of unlock
@property(nonatomic,copy) NSString *alertAction;

// used as the launch image (UILaunchImageFile) when launch button is tapped
@property(nonatomic,copy) NSString *alertLaunchImage;
```

Sounds

- To play the notification as a sound, you set the soundName property of UILocalNotification to point to a sound file that's in your app bundle...

```
// name of resource in app's bundle to play or UILocalNotificationDefaultSoundName
@property(nonatomic,copy) NSString *soundName;
```

Handling Notifications App Not Running in the Foreground

- If it's an alert, and the user presses the action button the following method is called on the app delegate...

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions;
```

- The userInfo dictionary my be retried by querying the launchOptions for the following key...

```
[launchOptions objectForKey:UIApplicationLaunchOptionsLocalNotificationKey];
```

- Note that no user data is available if the notification is a badge or sound and the user foregrounds the app,

Handling Notifications
App Running in the Foreground

- Notifications received while the app is in the foreground are delivered to the application through the following app delegate method...

```
- (void)application:(UIApplication *)application
        didReceiveLocalNotification:(UILocalNotification *)notification;
```

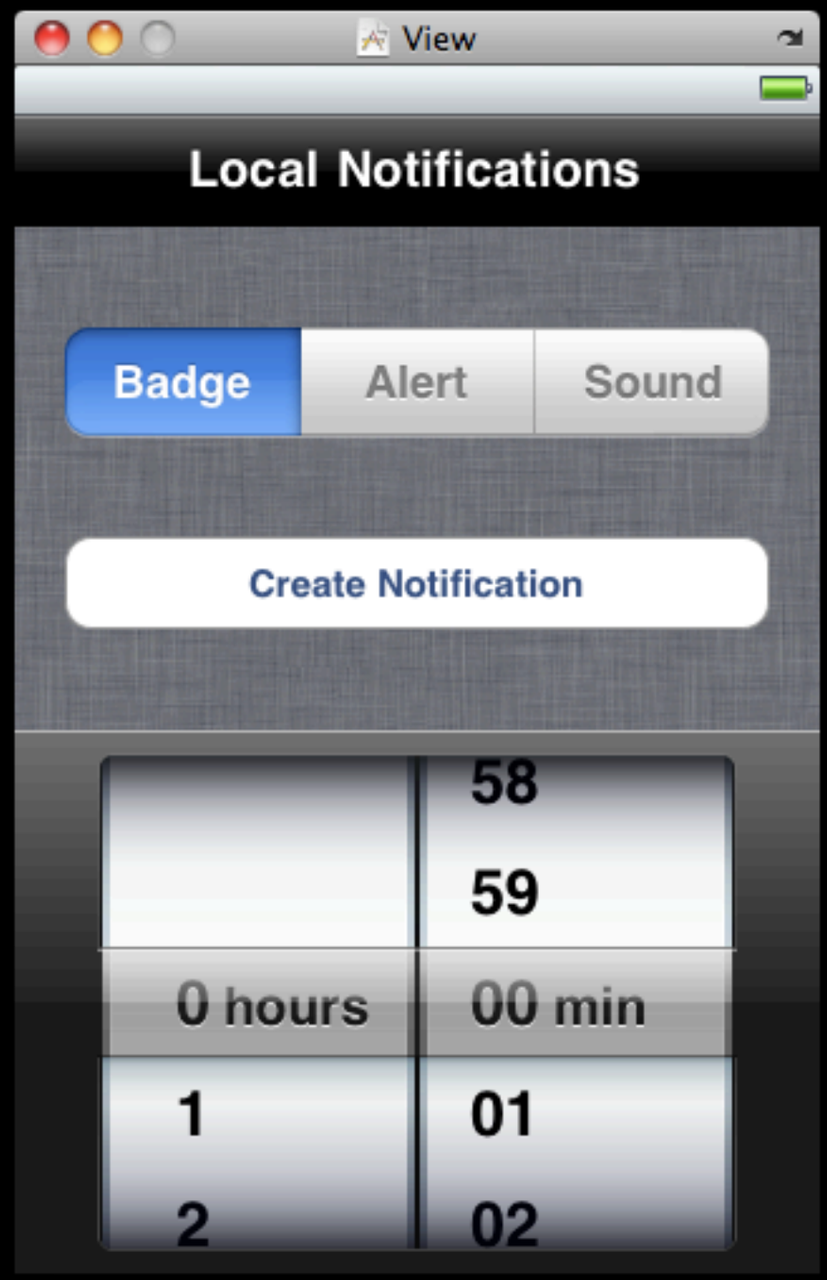
- Note that if the app is running in the foreground, the default behavior of the notification is suppressed
  - Your app should handle it and perform what's necessary to handle the notification

Local Notifications

Demo

LocalNotificationViewController.xib

- Segmented control for selecting notification type
- Timer picker to signify how far in the future to set the notification for
- Button to call action which creates and schedules the notification



LocalNotificationViewController.h

```
#import <UIKit/UIKit.h>

@interface LocalNotificationViewController : UIViewController {

}

@property(nonatomic, retain) IBOutlet UISegmentedControl *type;
@property(nonatomic, retain) IBOutlet UIDatePicker *datePicker;

-(IBAction)createNotification;

@end
```

LocalNotificationViewController.m

```
#import "LocalNotificationViewController.h"
enum { kBadge, kAlert, kSound };

@implementation LocalNotificationViewController

@synthesize type, datePicker;

-(IBAction)createNotification {

  UILocalNotification *notification = [[[UILocalNotification alloc] init]
                                      autorelease];

  if (notification) {

    NSDate *date = [NSDate date];
    NSDate *fireDate = [date dateByAddingTimeInterval:
                        self.datePicker.countDownDuration];

    notification.fireDate = fireDate;
    notification.timeZone = [NSTimeZone defaultTimeZone];

    /* ... */
```

LocalNotificationViewController.m

```
/* ... */

if (type.selectedSegmentIndex == kBadge) {
  notification.applicationIconBadgeNumber = [[UIApplication sharedApplication]
                                             applicationIconBadgeNumber] + 1;
}
if (type.selectedSegmentIndex == kAlert) {
  notification.alertBody = @"This is your alert message!";
  notification.alertAction = @"Open App";
  notification.hasAction = YES;
}
if (type.selectedSegmentIndex == kSound) {
  notification.soundName = UILocalNotificationDefaultSoundName;
}

NSDictionary *data = [NSDictionary dictionaryWithObject:date forKey:@"scheduled"];
notification.userInfo = data;

[[UIApplication sharedApplication] scheduleLocalNotification:notification];

  }
}

/* ... */

@end
```

The Resulting Notifications

Task Completion

Task Completion

- Allows an app to complete a task without running in the foreground
- Users may exit the app and perform other tasks while they wait for a task to complete
- There is a time limit imposed on task completion which varies based on a number of factors

Task Completion API

- Tasks for completion are sandwiched between a pair of begin and end task completion calls
- To signify the beginning of a task, you call the following UIApplication method and specify an ObjC 2.0 block which will be invoked if time runs out before the task completes...

```
- (UIBackgroundTaskIdentifier)beginBackgroundTaskWithExpirationHandler:
                                                    (void(^)(void))handler;
```

- You close the task completion section with another call to UIApplication signifying the end of the task...

```
- (void)endBackgroundTask:(UIBackgroundTaskIdentifier)identifier);
```

Task Completion API

- If the long running task is long enough that it requires running in the background it likely requires running in a thread other than the main UI thread
- We've seen a couple of ways to run code asynchronously, for this example we'll leverage Apple's recently introduced Grand Central Dispatch
- We'll leverage the following method where we pass a queue to run on and a block of code to execute

```
void dispatch_async(dispatch_queue_t queue,
                    dispatch_block_t block);
```

General Flow for Task Completion

```
UIApplication *app = [UIApplication sharedApplication];

// bgTask is an ivar so we can change in the dispatch block that follows
bgTask = [app beginBackgroundTaskWithExpirationHandler:^{

  // handle the case where the task runs out of time to complete
}];

// start the long-running task (usually in another thread), such as...
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{

  // perform the long running task

  // signal end of background task
  [app endBackgroundTask:bgTask];
  bgTask = UIBackgroundTaskInvalid;

});
```

Task Completion

Demo

Demo

- One common use of task completion is to complete downloads or uploads once the app is backgrounded
- In this demo, we'll allow the user to upload a photo to a web server
- We'll use the same approach that you'd use if you clicked a file upload dialog in a web browser and submitted the page
- We could do this all manually using the APIs provided by the various built-in network APIs
  - But, that would be semi-tedious to create the request
  - Let's use a 3rd party library that's well suited for this

ASIHTTPRequest

- We'll make use of the open source ASIHTTPRequest project
- Our app's going to use the ASIFormDataRequest class
- This class provides a high-level API which mimics the request a web browser would make if performing a form submission
- We can use this method to perform a file upload
- For more information (including build instructions) see...
  - http://allseeing-i.com/ASIHTTPRequest/

TaskCompletionViewController.xib

- Not the most interesting of UIs...
- Simple button which starts long running task
  - The file upload in our case

TaskCompletionViewController.h

```
#import <UIKit/UIKit.h>

@interface TaskCompletionViewController : UIViewController {
    UIBackgroundTaskIdentifier bgTask;
}

- (IBAction)doUpload;

@end
```

TaskCompletionViewController.m

```
#import "TaskCompletionViewController.h"
#import "ASIFormDataRequest.h"

@implementation TaskCompletionViewController

- (IBAction)doUpload {
  UIApplication *app = [UIApplication sharedApplication];

  bgTask = [app beginBackgroundTaskWithExpirationHandler:^{
    NSLog(@"Error: background task expired");
  }];

  // Start the long-running task and return immediately.
  dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{

    // build request
    NSURL *url = [NSURL URLWithString:@"http://127.0.0.1/~Dan/upload.php"];
    ASIFormDataRequest *request = [ASIFormDataRequest requestWithURL:url];
    NSString *photo = [[NSBundle mainBundle] pathForResource:@"photo" ofType:@"jpg"];
    [request setFile:photo forKey:@"file"];

    /* ... */
```

We'll just use a sample photo added to the bundle, you'd likely get it from some place more interesting

TaskCompletionViewController.m

```
/* ... */

// send request
[request startSynchronous];
NSError *error = [request error];
if (!error) {
  NSLog(@"Response: %@", [request responseString]);
} else {
  NSLog(@"Failure: %@", error);
}

    // signal end of background task
    [app endBackgroundTask:bgTask];
    bgTask = UIBackgroundTaskInvalid;

  });
}

/* ... */

@end
```

Server Side Scripts

- For this simple demo, we'll use a pair of very basic PHP server-side scripts
- upload.php
  - Take the file upload and saves it off to disk
  - Returns a text file indicating success or failure
- view.php
  - Displays the uploaded image
  - Or, an plain text file with an error message if an error
- Note: these scripts are very basic and lack the robustness of something you'd put into production

upload.php

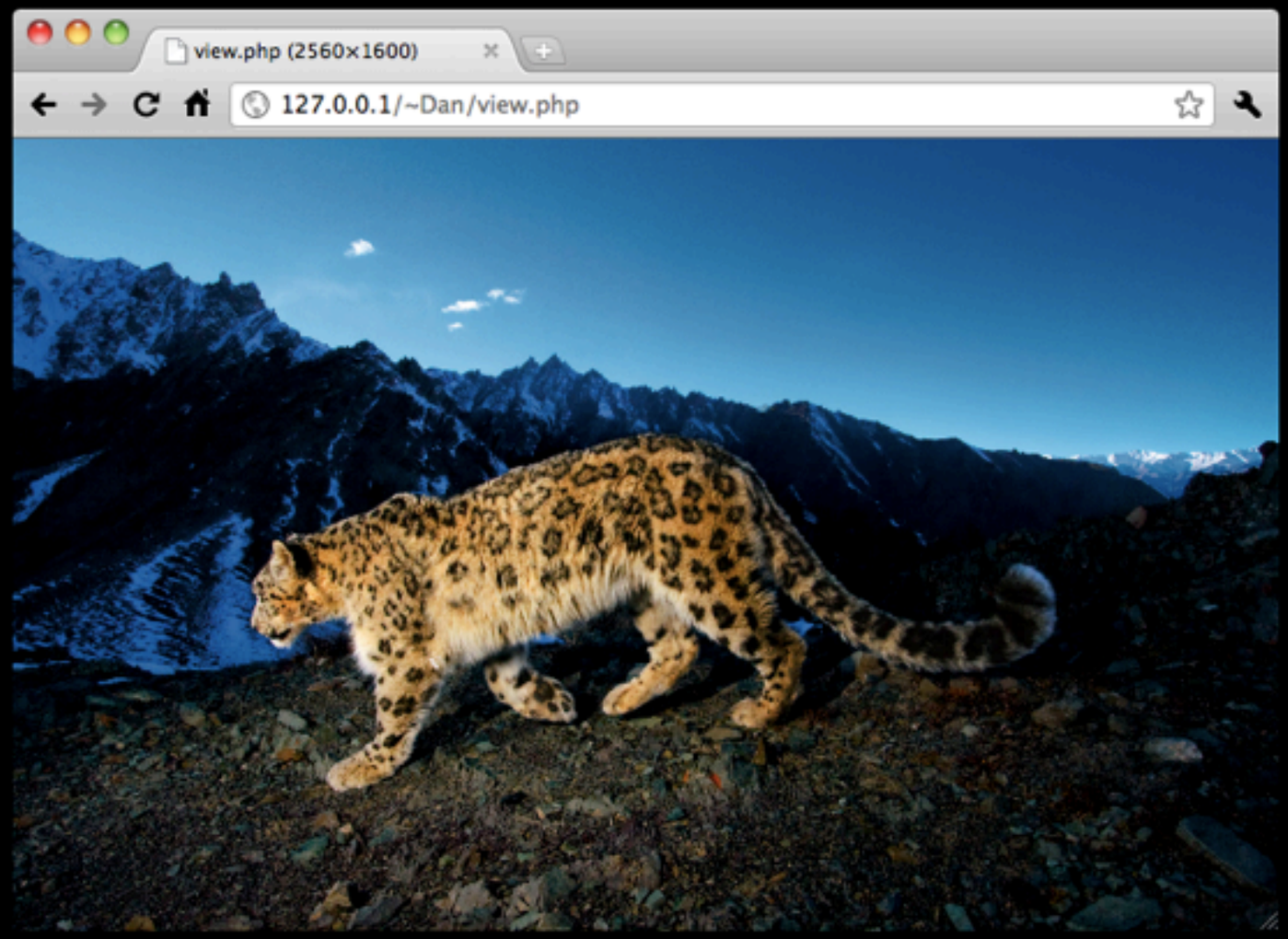
```php
<?
  // no need for the following line, it's just there to
  // simulate it taking a while to upload a large photo
  sleep(15);

  header('Content-Type: text/plain');
  if($_FILES['file']['type'] == 'image/jpeg') {
    if(move_uploaded_file($_FILES['file']['tmp_name'], "/tmp/uploaded-file.jpg")) {
      print "Success - file uploaded";
    } else {
      print "Failure - an error occurred";
    }
  } else {
    print "Failure - not a JPG?";
  }
?>
```

view.php

```
<?
  $image = file_get_contents("/tmp/uploaded-file.jpg");
  if($image === FALSE) {
    header("Content-Type: text/plain");
    print "Failure - unable to retrieve image";
  } else {
    header("Content-Type: image/jpeg");
    print $image;
  }
?>
```

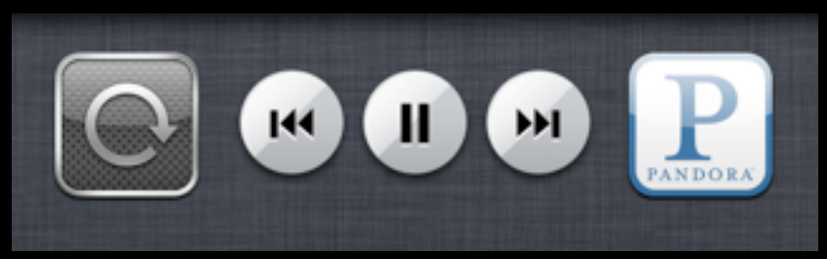
The Resulting App & Script



Background Audio

Background Audio

- Prior to iOS 4.0 apps such as Pandora and other streaming audio apps were pretty limited
  - Only played music if it was in the foreground
- In iOS 4.0 an app may continue to play audio in the background while utilizing other apps
- Capable of integration with player controls in the multitasking bar



Declaring Background Audio

- If your app wants to perform background audio, you'll need to add the UIBackgroundModes key (an array) and add the audio key as an entry...

Key	Value
▼ Information Property List	(13 items)
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow
▼ Required background modes	(1 item)
Item 0	App plays audio ▲▼ +

Becoming "the" Running Audio App

- If we want our app to be the app that iOS believes should be playing background audio and be manageable by the audio controls in the dock, we need to do a couple more things...
  - Become the first responder
  - Become the receiver of dock audio control events
  - Respond to the dock audio control events

AVAudioSession

- We've looked at utilizing the AVAudioSession class before to declare that we're performing a certain sort of audio control
- If we want to perform audio playback, we'll need to call the set category method...

```
- (BOOL)setCategory:(NSString*)theCategory error:(NSError**)outError;
```

- Specifying that we want to perform audio playback...

```
extern NSString *const AVAudioSessionCategoryPlayback;
```

Responding to Audio Controls

- In order to respond to the audio controls (like the iPod app), we'll need to be able to become the first responder, thus we'll have to implement the following to return YES...

```
- (BOOL)canBecomeFirstResponder;
```

- We'll also have to invoke the method form our app (likely on the view controller, but anything that's a UIResponder)...

```
- (BOOL)becomeFirstResponder;
```

Responding to Audio Controls

- To respond to the controls, we'll have to ask UIApplication to receive those remote control events...

```
- (void)beginReceivingRemoteControlEvents;
```

- Lastly, we have to override UIResponder's method for handling those events...

```
- (void)remoteControlReceivedWithEvent:(UIEvent *)event;
```

Background Audio

Demo

Demo

- For this demo we're going to extend the audio player app from the Audio & Video lecture
- Currently, if you tap the home button and suspend the app, the audio stops
- Let's add the ability to background the app and control audio playback using the dock and lock screen controls
- It turns out we only need to add a few methods to PlayerViewController.m to leverage background audio

PlayerViewController.m

```
#import "PlayerViewController.h"

@implementation PlayerViewController

@synthesize player, playOrPauseButton, durationLabel, currentPositionLabel;
@synthesize currentPositionProgress, volumeSlider, leftMeter, rightMeter;

// added so we can become 1st responder to handle remote (dock) events
- (BOOL)canBecomeFirstResponder {
    return YES;
}

/* ... */
```

PlayerViewController.m

```
/* ... */

// added so the app is the only app playing audio, set to be first responder
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];

    // We want to perform long-term media playback
    AVAudioSession *session = [AVAudioSession sharedInstance];
    NSError *error = nil;
    [session setCategory:AVAudioSessionCategoryPlayback error:&error];
    if (error) {
       NSLog(@"Error: %@", [error localizedDescription]);
    }

    [[UIApplication sharedApplication] beginReceivingRemoteControlEvents];
    [self becomeFirstResponder];
}

/* ... */
```

PlayerViewController.m

```
/* ... */

// added to respond to remote control events
- (void)remoteControlReceivedWithEvent:(UIEvent *)event {
    switch(event.subtype) {
       case UIEventSubtypeRemoteControlTogglePlayPause:
          NSLog(@"Toggle play/pause state");
          [self playOrPause];
          break;
       case UIEventSubtypeRemoteControlNextTrack:
          NSLog(@"Skip to next track");
          break;
       case UIEventSubtypeRemoteControlPreviousTrack:
          NSLog(@"Skip to previous track");
          break;
    }
}

/* ... */

@end
```

The Resulting App

The Resulting App

Bringing up the Audio controls shows our app, when we tap pause the audio stops

The Resulting App

You can also use the lock screen controls

Location — Navigation

Navigation

- Another common request prior to iOS 4.x was the ability to have an iOS device act as a GPS navigation unit
- Let's say you wanted to have a turn-by-turn directions app, that read the directions as you approached different points along the path
  - We could have background audio read the turns aloud
  - We'd also need location changes to constantly so we knew when to announce a turn, etc.

Declaring Background Location & Audio

- If your app wants to perform location & audio, you'll need to add the UIBackgroundModes array key again — adding the location & audio keys as entries...

Key	Value
▼ Information Property List	(13 items)
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow
▼ Required background modes	(2 items)
Item 0	App plays audio
Item 1	App registers for location updates

Accuracy for Navigation

- If you're building a navigation based app it's recommended that you set the desiredAccuracy property on the CLLocationManager instance...

```
@property(assign, nonatomic) CLLocationAccuracy desiredAccuracy;
```

- There's a CLLocationAccuracy constant defined specifically for navigation purposes...

```
extern const CLLocationAccuracy kCLLocationAccuracyBestForNavigation;
```

Getting Updates

- You obtain updates the normal way by instantiating a CLLocationManager instance and set a delegate...

```
@property(assign, nonatomic) id<CLLocationManagerDelegate> delegate;
```

- Location updates can then be turned on and off via...

```
- (void)startUpdatingLocation;
- (void)stopUpdatingLocation;
```

- Updates received via the delegate method...

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation;
```

Putting it Together

- Whenever you receive a location update...
  - Announce turns, etc. based on the current location
  - If the app is in the foreground, update the navigation UI

Location — Tracking

Location Tracking

- There are 2 other different types of location tracking options...
  - Significant location changes
    - Fires when switching between cell towers
  - Region monitoring
    - Detects when the device has entered one or more geographical regions of interest

Location Tracking

	Significant Location Changes	Region Monitoring
Uses less power than standard location services	✔	✔
Resumes suspended applications	✔	✔
Launches terminated applications	✔	✔
Notifications are not coalesced		✔
Supported on iPhone 4	✔	✔
Supported on iPhone 3GS	✔	

Monitoring Significant Location Changes

- Requesting significant location changes is very similar to receiving normal location changes
- The methods to stop and start are...

```
- (void)startMonitoringSignificantLocationChanges;
- (void)stopMonitoringSignificantLocationChanges;
```

- Updates are received via the normal...

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation
           fromLocation:(CLLocation *)oldLocation;
```

- You can detect if these are available via...

```
+ (BOOL)significantLocationChangeMonitoringAvailable;
```

Monitoring Region Changes

- Requesting significant location changes is a little different than the other location change APIs
- The methods to stop and start are...

```
- (void)startMonitoringForRegion:(CLRegion *)region
                  desiredAccuracy:(CLLocationAccuracy)accuracy;
- (void)stopMonitoringForRegion:(CLRegion *)region;
```

- You can detect if region monitoring is available via...

```
+ (BOOL)regionMonitoringAvailable;
+ (BOOL)regionMonitoringEnabled;
```

CLRegion

- A CLRegion object is used to model a geographic area
- As of iOS 4.1, the only region that's supported is a circle, which can be constructed using the following method...

```
- (id)initCircularRegionWithCenter:(CLLocationCoordinate2D)center
                            radius:(CLLocationDistance)radius
                        identifier:(NSString *)identifier;
```

- The CLRegion instance currently has the following properties/methods...

```
@property (readonly, nonatomic) CLLocationCoordinate2D center;
@property (readonly, nonatomic) CLLocationDistance radius;
@property (readonly, nonatomic) NSString *identifier;
- (BOOL)containsCoordinate:(CLLocationCoordinate2D)coordinate;
```

- The max radius supported can be queried using...

```
@property (readonly, nonatomic) CLLocationDistance maximumRegionMonitoringDistance;
```

Monitoring Region Changes

- The following delegate methods can be used to identify when the device has entered or exited a monitored region...

```
- (void)locationManager:(CLLocationManager *)manager
  didEnterRegion:(CLRegion *)region;
- (void)locationManager:(CLLocationManager *)manager
  didExitRegion:(CLRegion *)region;
```

Executing Code

- From the update location & region methods, you can then make use of the task completion technique to perform some arbitrary tasks

Voice Over IP

Voice Over IP

- Let's say you were implementing a voice over IP app, we'd want to be able to...
  - Receive incoming calls (in foreground or background)
  - Make outgoing calls
  - Stay connected to a call while entering the background

Declaring Background Audio and VOIP

- Again, we'll need to add to the UIBackgroundModes array key again — this time adding the audio & voip keys as entries...

Key	Value
▼ Information Property List	(13 items)
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow
▼ Required background modes	(2 items)
Item 0	App plays audio
Item 1	App registers for location updates

Voice Over IP

- For more details on implementing a VOIP app, check out...
  - WWDC 2010 Adopting Multitasking on iPhone OS, Part 2

Additional Resources

- The following sections from the "iOS Programming Guide"
  - Executing Code in the Background
    - http://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html
  - Opting out of Background Execution
    - http://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/CoreApplication/CoreApplication.html#//apple_ref/doc/uid/TP40007072-CH3-SW24

Additional Resources

- The following WWDC 2010 Session Videos (free on iTunes) are also very informative for the materials in this lecture...
  - Session 105 — Adopting Multitasking on iPhone OS, Part 1
  - Session 109 — Adopting Multitasking on iPhone OS, Part 2
  - Session 129 — Implementing Local and Push Notifications

For Next Class

- View Controller Programming Guide for iOS — iPad-Specific Controllers section...
  - http://developer.apple.com/library/ios/#featuredarticles/ViewControllerPGforiPhoneOS/iPadControllers/iPadControllers.html
- iOS Application Programming Guide — Creating a Universal Application section...
  - http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BuildTimeConfiguration/BuildTimeConfiguration.html#//apple_ref/doc/uid/TP40007072-CH7-SW24