

# Adopting Multitasking in Your App

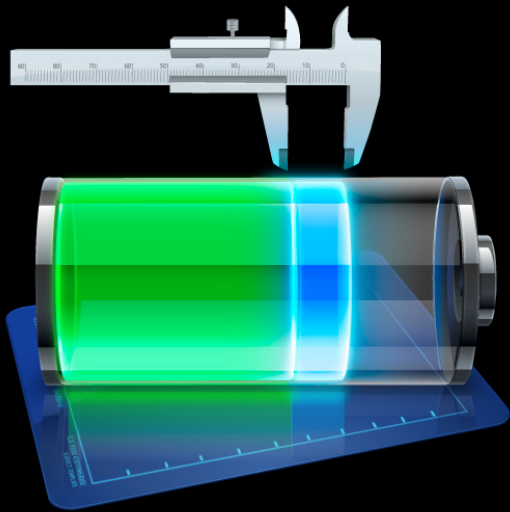
Session 320

**Dave Myszewski, Charles Srisuwananukorn**

iOS Performance

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# Introduction



- Multitasking provides services that work on behalf of apps
- Apps do not need to run all the time
- Good for the performance
- Good for battery life

# Supported on All iOS Devices

iOS



# What You'll Learn

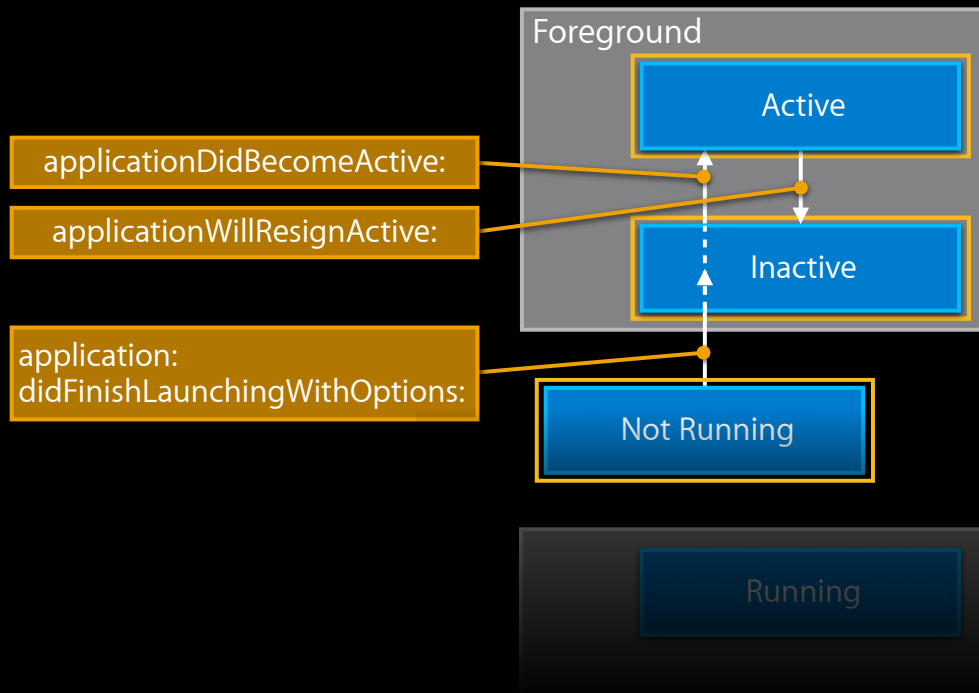
- App lifecycle
- Best practices
- Multitasking services



# App Lifecycle

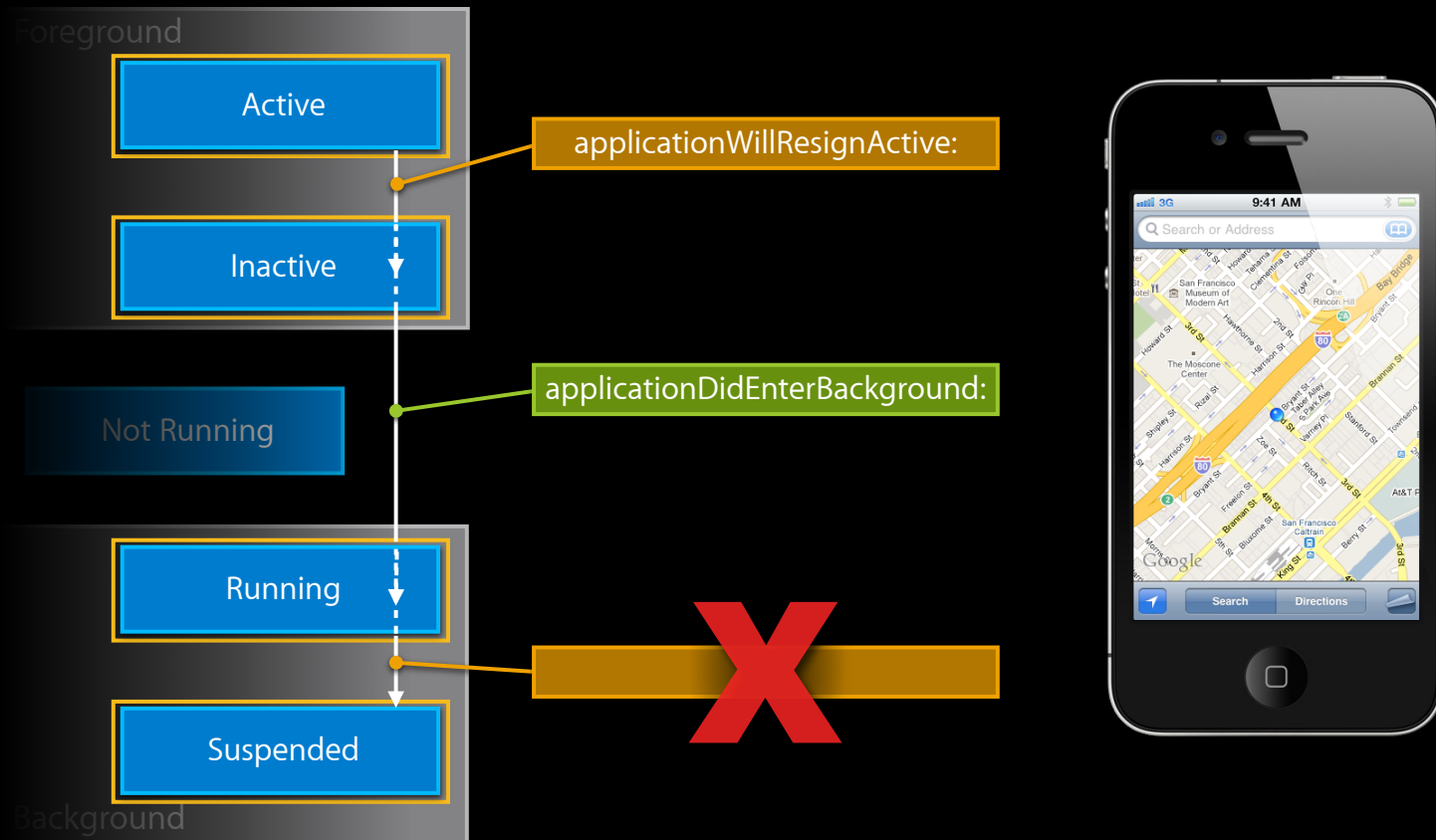
# UIApplicationDelegate Callbacks

## Launch and active/inactive



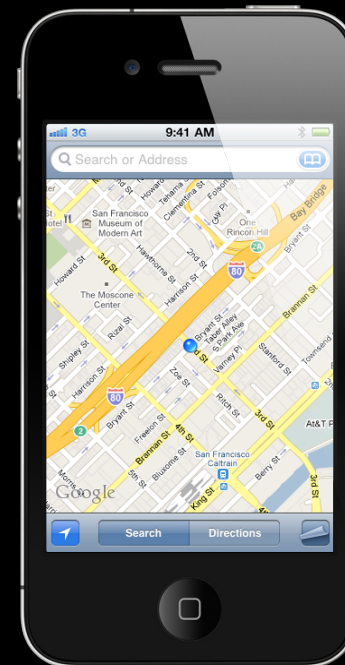
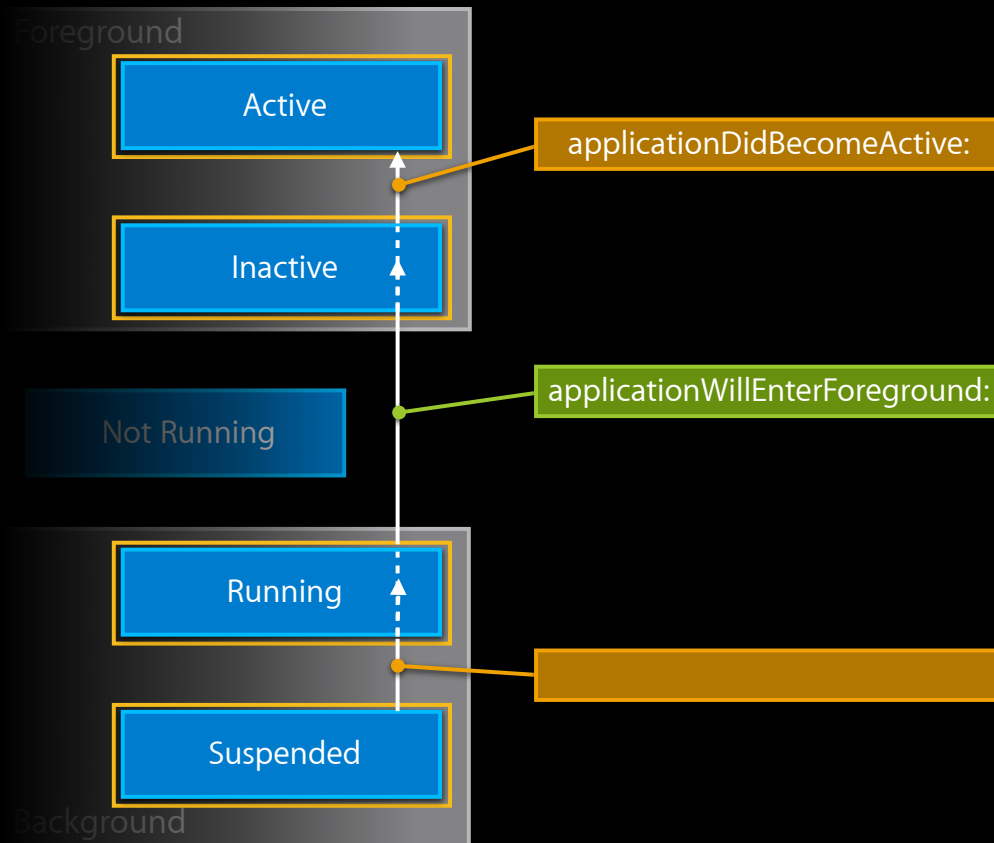
# UIApplicationDelegate Callbacks

## Switching from an app



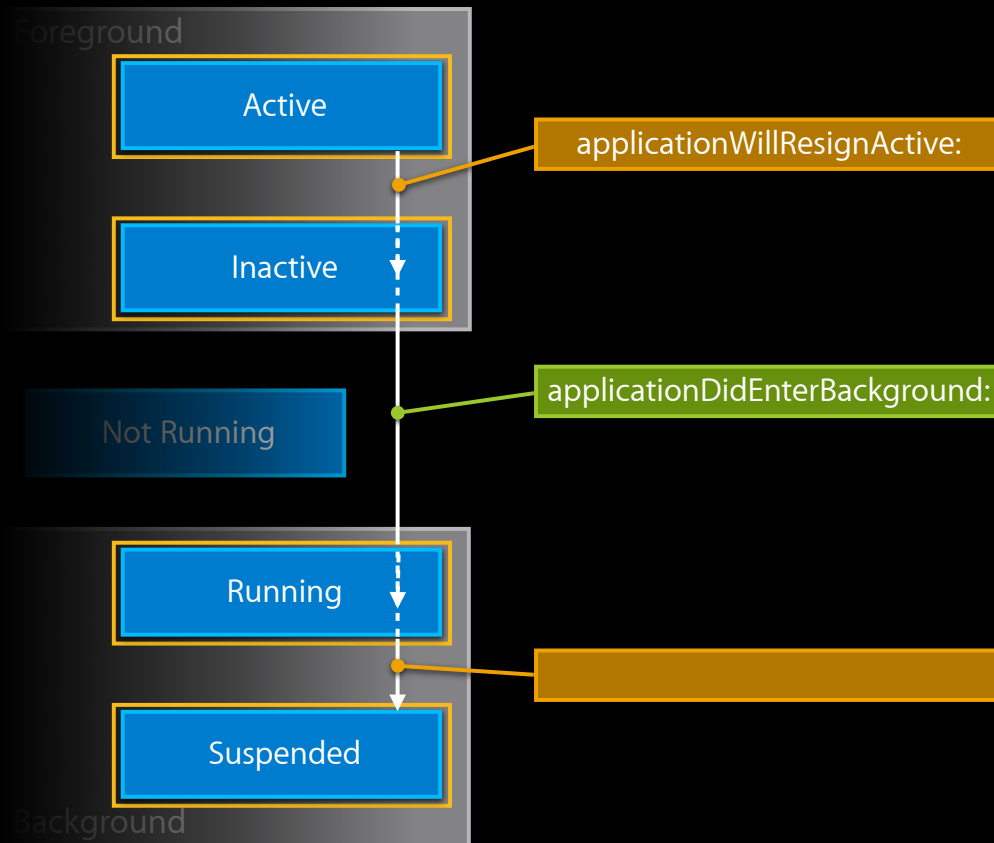
# UIApplicationDelegate Callbacks

## Switching to an app



# UIApplicationDelegate Callbacks

## Device lock



# Lifecycle Notifications

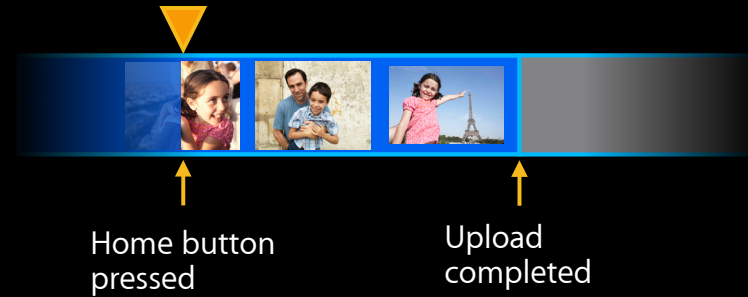
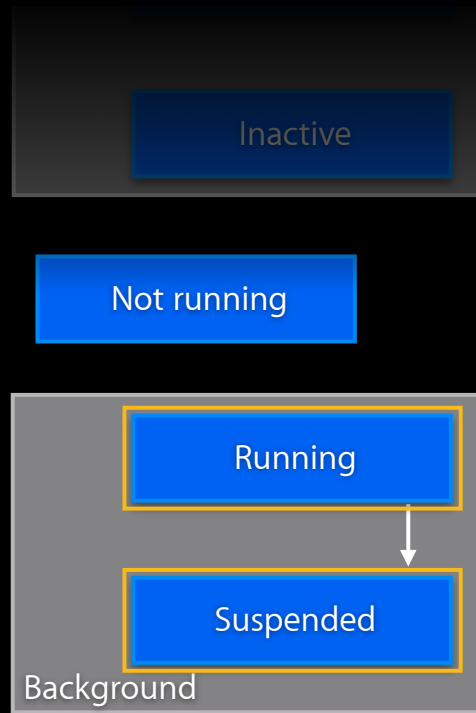
UIApplicationDelegate Callback	Notification
application:didFinishLaunchingWithOptions:	UIApplicationDidFinishLaunchingNotification
applicationWillTerminate:	UIApplicationWillTerminateNotification
applicationDidBecomeActive:	UIApplicationDidBecomeActiveNotification
applicationWillResignActive:	UIApplicationWillResignActiveNotification
applicationDidEnterBackground:	UIApplicationDidEnterBackgroundNotification
applicationWillEnterForeground:	UIApplicationWillEnterForegroundNotification

# Running in the Background

- Multitasking services provide three ways to run in the background
  - Continue your current task
  - Run on external triggers
  - Run on targeted networking events

# Task Completion

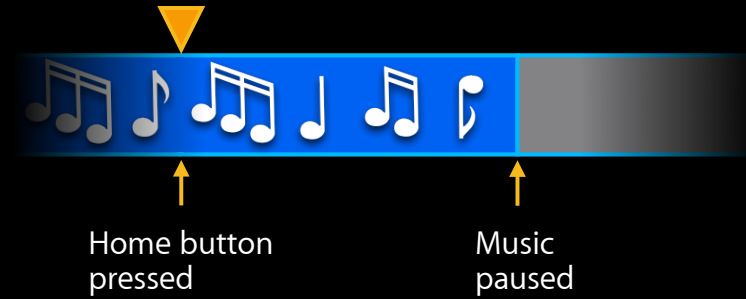
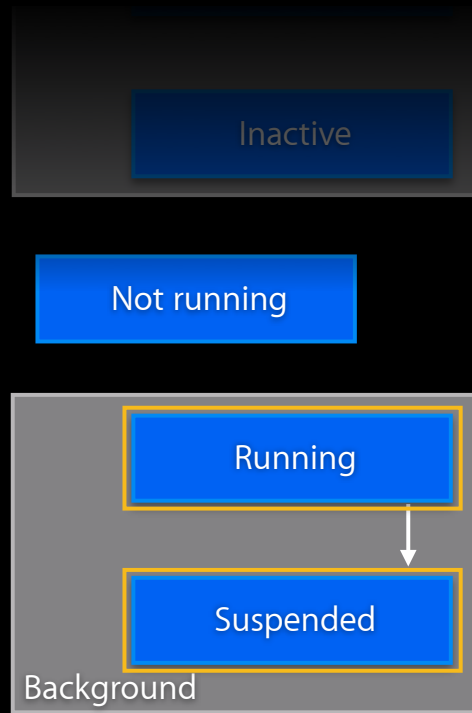
## Application lifecycle





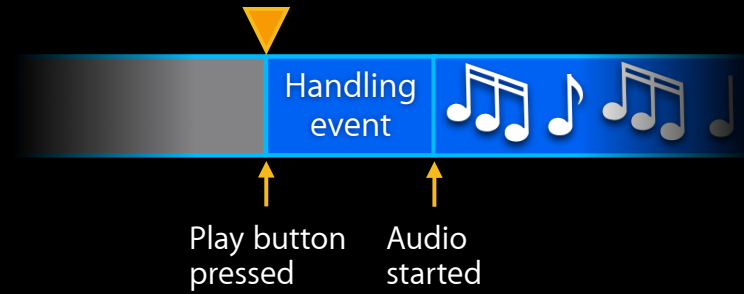
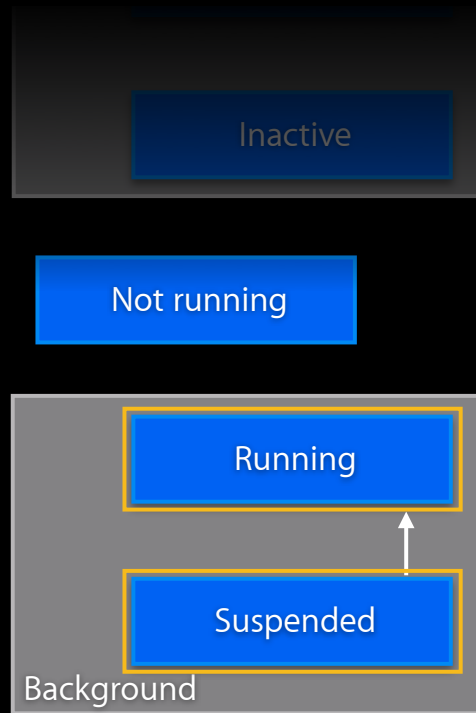
# Background Audio

## Application lifecycle



# Background Audio

## Remote control



# Best Practices

# Best Practices

- System resources
  - Memory
  - OpenGL
- Gracefully resuming from the background
  - Preserving state
  - Networking
  - System notifications

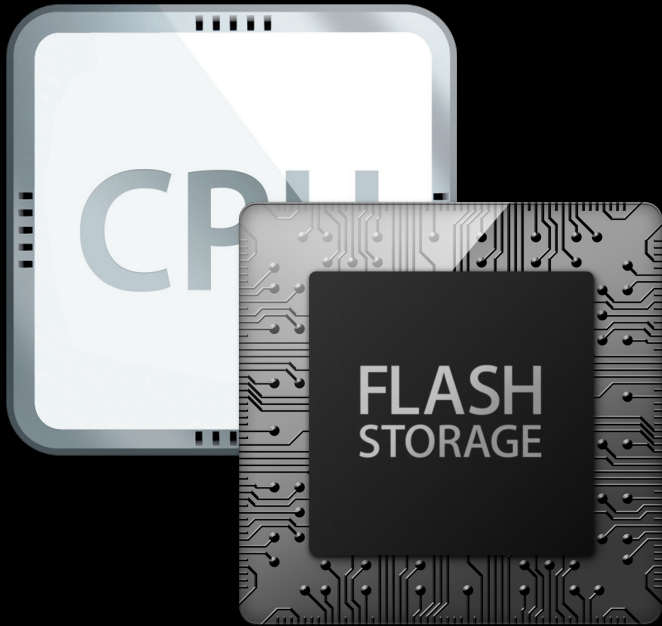


# Best Practices



- Apps share system resources
  - CPU
  - I/O
  - Memory
  - GPU
  - Network

# Best Practices



- System prioritizes some resources for the foreground app
  - CPU
  - I/O



# Best Practices



- Some resources are off limits
  - GPU (OpenGL)

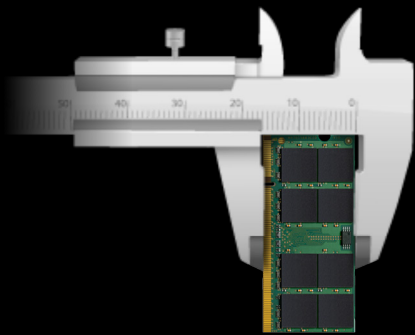


# Best Practices



- Other resources, your app can help manage
  - Memory

# Memory



- Apps share a limited amount of memory
- iOS ensures that the device has the memory it needs
  - Sends running apps memory warnings
  - Terminates apps

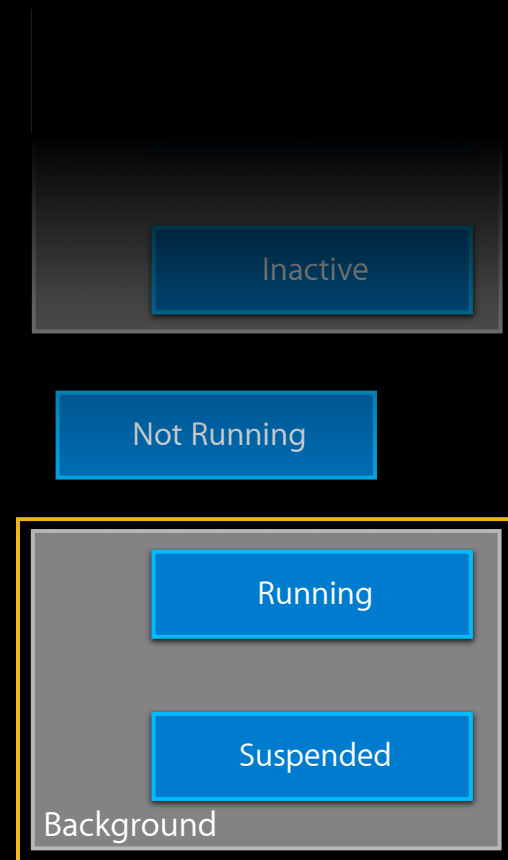
# Memory

## Memory warnings

- iOS sends notifications to running apps to free memory
- Only sends warnings when freeing memory is crucial
- Suspended apps do not receive memory warnings

# Memory

- Your app should free memory on entering the background
- OS and frameworks free some memory going to the background
- Apps using less than 16 MB of dirty memory are written to disk
- Balance memory footprint and speed to resume



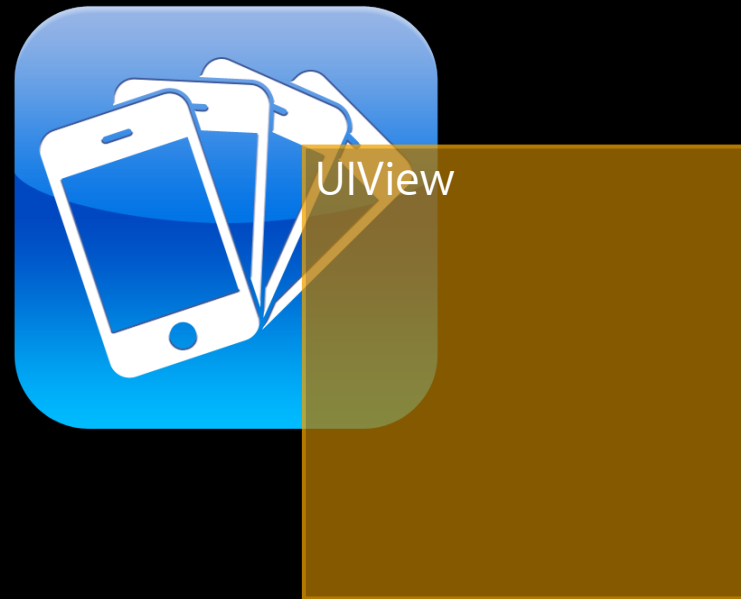
# Memory

View backing stores



# Memory

## View backing stores



# Memory

## View backing stores

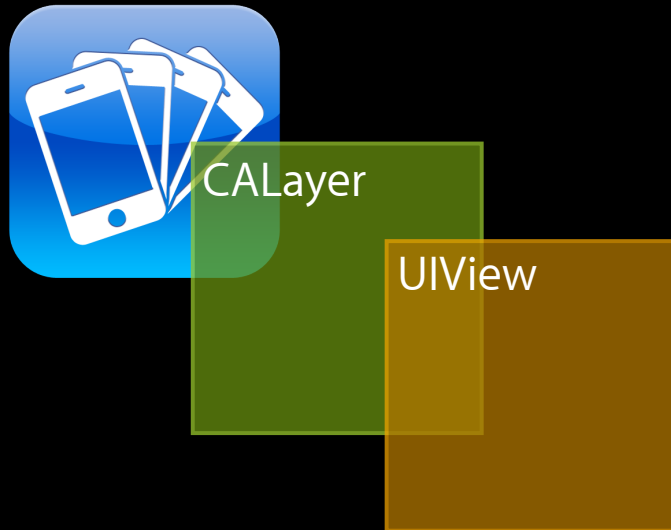


CALayer

UIView

# Memory

## View backing stores



- Every UIView has a CALayer
- UIViews that draw themselves have bitmap backing stores
- iOS may reclaim backing stores while app is in the background
- If reclaimed, iOS calls the view's `-drawRect:` for content



# Memory

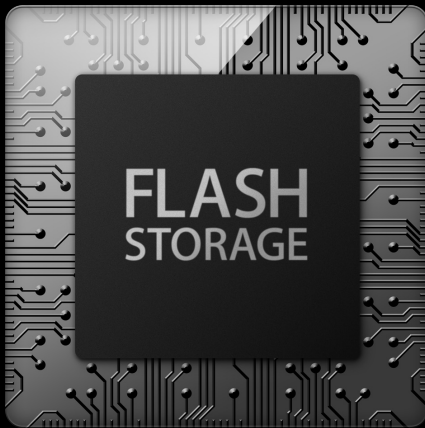
## UIImage cache



- UIKit caches images loaded with `-[UIImage imageNamed:]`
- Cache is purged on entering the background

# Memory

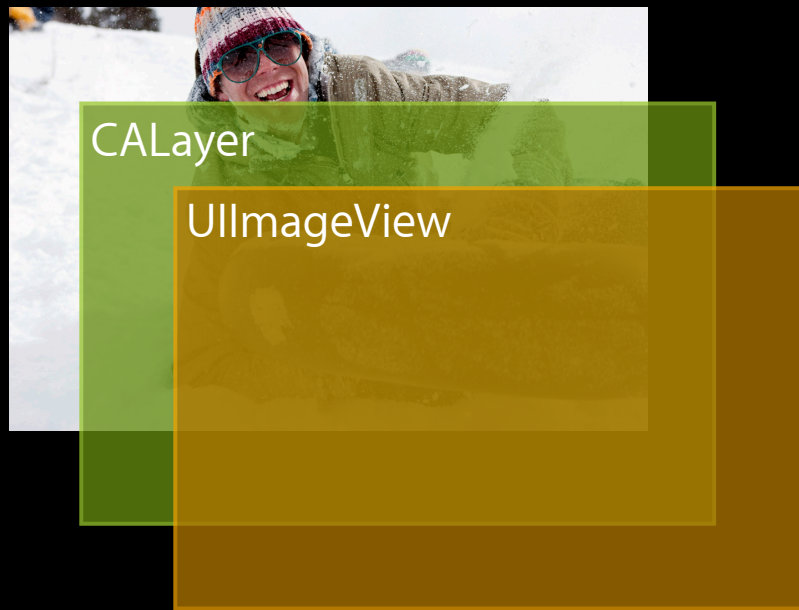
## Disk caches



- Many frameworks cache data in memory
  - SQLite
  - Core Data
  - NSCache
- Caches are emptied on entering the background

# Memory

## UIImageViews



- UIImageViews also have CALayer
- CALayer uses the image directly
- Images are not automatically reclaimed
- Detach large images from the view hierarchy on suspend
- Unload offscreen UIImageViews
- But beware of decompression on resume

# Memory

## Caches

- Flush application caches
- But not if resuming takes as long relaunching from scratch
- Consider using `NSCache` and `NSPurgeableData`

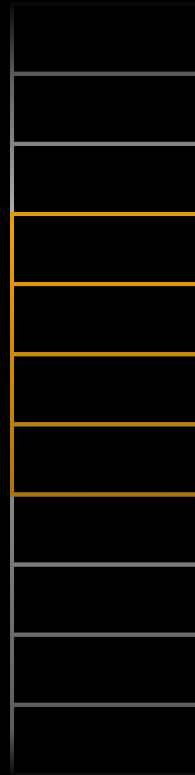
# Memory

## NSCache and NSPurgeableData

- NSCache
  - Caches objects in memory
  - Evicts objects as necessary
  - Evicts objects when entering the background
- NSPurgeableData objects in an NSCache are not evicted
  - Instead they become reclaimable when not in use

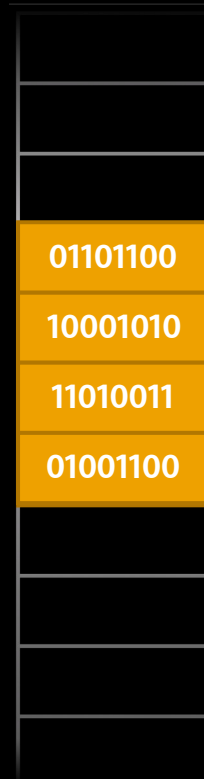
# Memory

## Memory-mapped files



# Memory

## Memory-mapped files



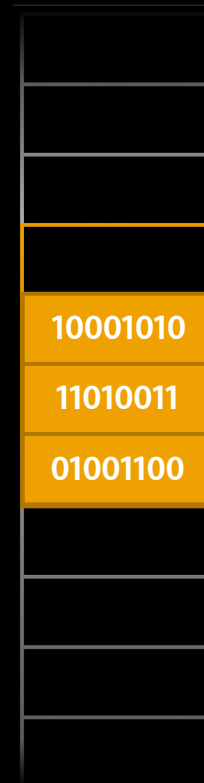
# Memory

## Memory-mapped files

- Map files into memory instead of reading them if possible

```
+ [NSData  
dataWithContentsOfMappedFile:]
```

- iOS reclaims pages from mapped read-only files automatically





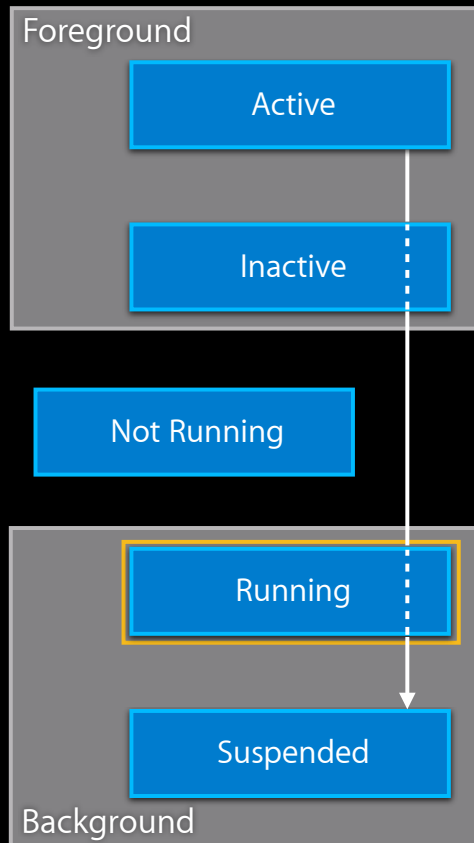
# Memory

## Summary

- Free memory when you receive a memory warning
- Remove UIImageViews from the view hierarchy unless decompression is a problem
- Use NSCache and memory-mapped files when appropriate
- Balance memory footprint and speed to resume

Demo

# OpenGL



- iOS terminates apps that use OpenGL in the background
- Stop animation timer when entering the background

# OpenGL

Exception Type: EXC\_CRASH (SIGABRT)

...

Thread 0 Crashed:

0 libsystem\_kernel.dylib 0x33b35a1c \_\_pthread\_kill

1 libsystem\_c.dylib 0x30bac3b4 pthread\_kill

2 libsystem\_c.dylib 0x30ba4bf8 abort

3 IMGSGX535GLDriver 0x355229ae glReturnNotPermittedKillClient

...

7 OpenGL 0x354ade4e glFinish

...

# OpenGL

## Starting and stopping animation

```
- (void)startAnimation {  
    if (!animating) {  
        CADisplayLink *aDisplayLink =  
            [[UIScreen mainScreen] displayLinkWithTarget:self  
                selector:@selector(drawFrame)];  
        [aDisplayLink setFrameInterval:animationFrameInterval];  
        [aDisplayLink addToRunLoop:[NSRunLoop currentRunLoop]  
            forMode:NSDefaultRunLoopMode];  
        self.displayLink = aDisplayLink;  
        animating = YES;  
    }  
}
```

# OpenGL

## Starting and stopping animation

```
- (void)stopAnimation {  
    if (animating) {  
        [self.displayLink invalidate];  
        self.displayLink = nil;  
        animating = NO;  
    }  
}
```

# OpenGL

## Starting and stopping animation

```
- (void)awakeFromNib {  
    NotificationCenter *center = [[NSNotificationCenter defaultCenter];  
    ...  
    [center addObserver:self  
                selector:@selector(applicationWillResignActive:)  
                name:UIApplicationWillResignActiveNotification  
                object:nil];  
    ...  
}
```

# OpenGL

## Starting and stopping animation

```
- (void)applicationWillResignActive:(NSNotification *)notification {  
    if ([self isViewLoaded] && self.view.window) {  
        [self stopAnimation];  
    }  
}
```



# OpenGL

## Starting and stopping animation

```
- (void)awakeFromNib {  
    NotificationCenter *center = [[NSNotificationCenter defaultCenter];  
    ...  
    [center addObserver:self  
                selector:@selector(applicationDidBecomeActive:)  
                name:UIApplicationDidBecomeActiveNotification  
                object:nil];  
    ...  
}
```

# OpenGL

## Starting and stopping animation

```
- (void)applicationDidBecomeActive:(NSNotification *)notification {  
    if ([self isViewLoaded] && self.view.window) {  
        [self startAnimation];  
    }  
}
```

# OpenGL

## Starting and stopping animation

```
- (void)awakeFromNib {  
    NotificationCenter *center = [[NSNotificationCenter defaultCenter];  
    ...  
    [center addObserver:self  
                selector:@selector(applicationWillTerminate:)  
                name:UIApplicationWillTerminateNotification  
                object:nil];  
    ...  
}
```

# OpenGL

## Starting and stopping animation

```
- (void)applicationWillTerminate:(NSNotification *)notification {  
    if ([self isViewLoaded] && self.view.window) {  
        [self stopAnimation];  
    }  
}
```

# Best Practices

- System resources
  - Memory
  - OpenGL
- Gracefully resuming from the background
  - Preserving state
  - Networking
  - System notifications

# Preserving State

## Common state

- Return to exactly where you left off
- Save common UI state
  - Selected tab bar
  - Scroll position

# Preserving State

## App-specific content

- User input, like the last number entered in Calculator
- For networking apps, save the last search query
  - Ideal: Save enough state to return to exact content
  - Fallback: Reissue query

# Preserving State

## Games

- Turn-by-turn games should save after each turn
- Games with substantial, frequent state updates save periodically
  - When the application enters the background
  - Between levels, rooms



# Networking

## Sockets and suspension

- Sockets may disconnect while suspended
  - Be prepared for errors on resume
- Suspended apps cannot accept incoming connections
  - Close listening sockets before suspend
  - Reopen listening sockets on resume

# Networking

## Bonjour

- Bonjour operations may be cancelled while app suspended
- Restart Bonjour services if necessary on resume

# System Notifications

- System change notifications not delivered to suspended app
- System coalesces and queues notifications
- Delivered when app resumes

# Settings and Locale Changes

- Preferences and locale may be changed in Settings app

Event	Notification
Preference changed in Settings	NSUserDefaultsDidChangeNotification
Language or locale change	NSCurrentLocaleDidChangeNotification

# Notifications Delivered on Resume

Event	Notification
Accessory connected	EAAccessoryDidConnectNotification
Accessory disconnected	EAAccessoryDidDisconnectNotification
Device orientation change	UIDeviceOrientationDidChangeNotification
Time changes significantly	UIApplicationSignificantTimeChangeNotification
Battery level change	UIDeviceBatteryLevelDidChangeNotification
Battery state change	UIDeviceBatteryStateDidChangeNotification
Proximity state change	UIDeviceProximityStateDidChangeNotification
Protected file status change	UIApplicationProtectedDataWillBecomeUnavailable
	UIApplicationProtectedDataDidBecomeUnavailable
External display connected	UIScreenDidConnectNotification
External display disconnected	UIScreenDidDisconnectNotification
Screen display mode change	UIScreenModeDidChangeNotification
Preference changed in Settings	NSUserDefaultsDidChangeNotification
Language or locale change	NSCurrentLocaleDidChangeNotification

# Multitasking Services

# Task Completion

## Best practices

- Use it! Users expect it
- Finish as quickly as possible
- If you can't finish in time:
  - End your background task in your expiration handler
  - Store what you can to resume where you left off

# Background Audio

5

- Audio system provides many audio services
  - Prioritizing audio
  - Mixing and ducking
  - Headsets
  - External speakers
  - Display remote (new to iOS 5)





# Background Audio

## Audio interruptions

- Handle audio interruptions
- During an interruption
  - Audio system silences interrupted application
  - Update UI appropriately
  - Resume after the interruption

# Background Audio

## Audio interruptions

- In `beginInterruption`
  - Stop downloading the stream
  - Update UI
    - Play/Pause button
    - Play time
  - Stop visualizations

# Background Audio

## Audio interruptions

- In `endInterruptionWithFlags:`
  - Resume audio if `AVAudioSessionInterruptionFlags_ShouldResume` is set
  - Audio should resume for phone calls
  - Audio should not resume if iPod interrupts

# Location Tracking

- Significant location changes
  - Sends a notification after changing cell towers
- Region monitoring
  - Sends a notification upon entering and exiting regions of interest

# Location Tracking

## Significant location changes













# Location Tracking

## Region monitoring



Entered region

# 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		

# Newsstand



- Multitasking requirements in Info.plist
  - Add `newsstand-content` to `UIBackgroundModes`
  - Add `UINewsstandApp` key
- Push notifications allow once/day content downloads



# Newsstand

## Best practices



- Minimize resource downloads
  - Consumes disk space
  - Items automatically evicted when space is needed
  - Battery life
- Minimize number of downloads

# Summary

- Multitasking in iOS provides services that do work on your behalf
- Respond to memory warnings
- Minimize background memory usage
- Resume back to where you were when you suspended
- Balance memory footprint in the background with speed to resume

# More Information

## Michael Jurewitz

Developer Tools and Performance Evangelist  
[jurewitz@apple.com](mailto:jurewitz@apple.com)

## Documentation

iPhone Application Programming Guide  
<http://developer.apple.com/iphone>

## Apple Developer Forums

<http://devforums.apple.com>

# Related Sessions

iOS Performance in Depth

Presidio  
Thursday 4:30PM

iOS Performance and Power Optimization with Instruments

Presidio  
Wednesday 4:30PM

Building Newsstand Apps

Russian Hill  
Thursday 11:30AM

