Adopting Multitasking on iPhone OS

Part 1

David Myszewski, David Goodwin
iPhone Performance

Introduction

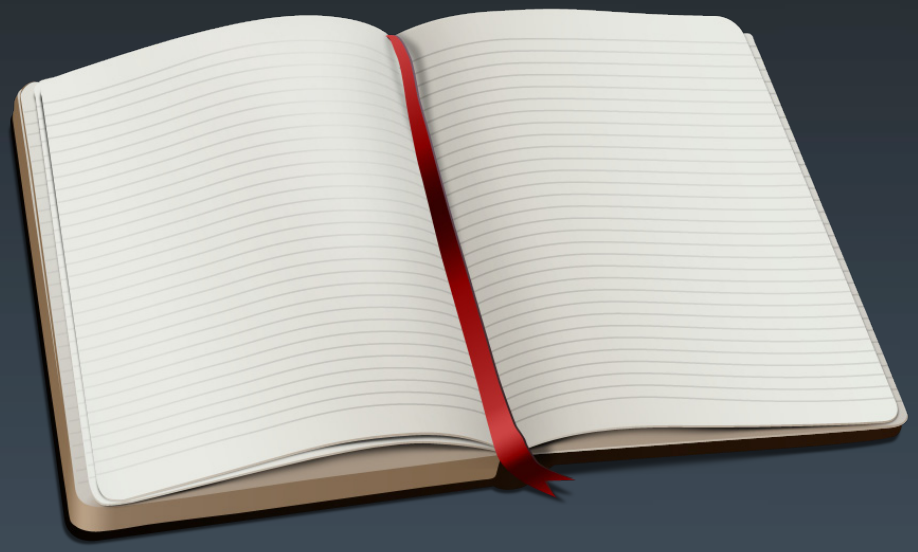
- Multitasking is a major feature of iOS 4
- Improves user experience
- Enhances existing apps
- Enables new app classes
- Every app should adopt multitasking

What You'll Learn

- Overview of iOS 4 multitasking
- How to enable multitasking in your app
  - APIs
  - Responsibilities
  - Best practices
- Using development tools to implement multitasking

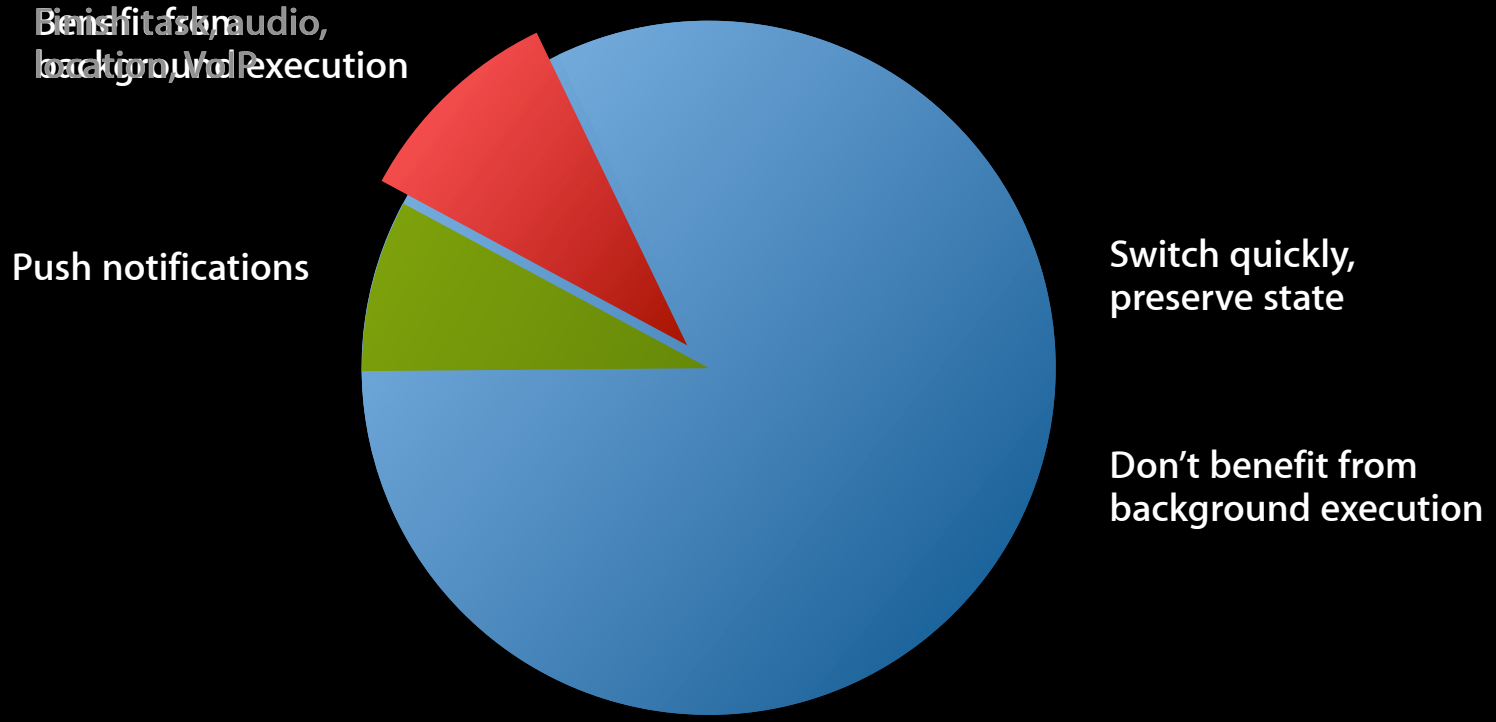
Demo

iPhone OS 3 vs. iOS 4



iOS 4 Multitasking Philosophy

- General purpose concurrency is not the solution for mobile devices



Fast App Switching

Multitasking benefits for all iOS 4 apps

- App resumes immediately
- App state is preserved
- Tight integration with multitasking UI
- Demonstrates supported, up-to-date app

Understanding the Services

- Fast app switching

  Resume quickly, preserve state

- Push notifications

  Respond to a notification sent from a remote server

- Local notifications

  Push-style notification delivered at a predetermined time

Understanding the Services

- Background audio

  Play audible content to the user while in the background

- Task completion

  Extra time to complete a task

- 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

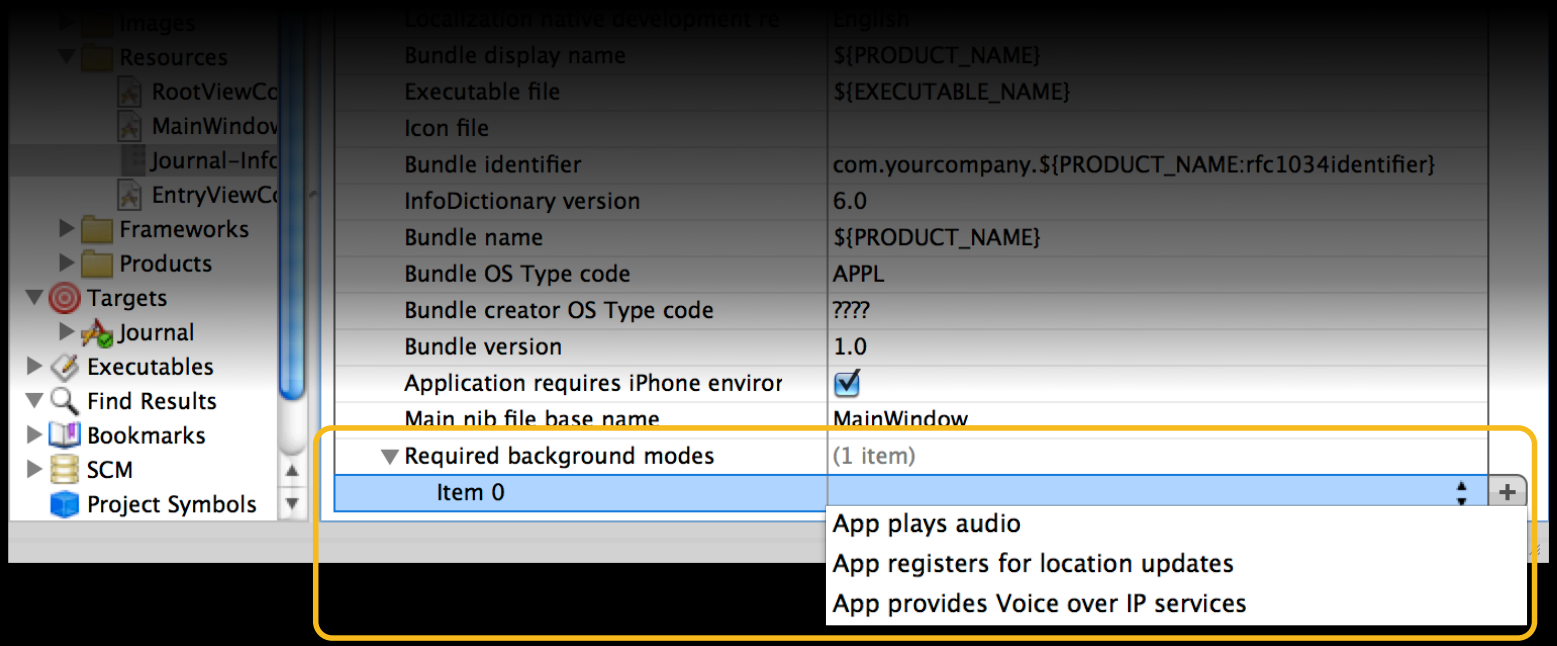
Multitasking Services

Details

Adopting Multitasking on iPhone OS, Part 2	Mission Tuesday, 3:15PM-4:15PM

How to Enable Multitasking

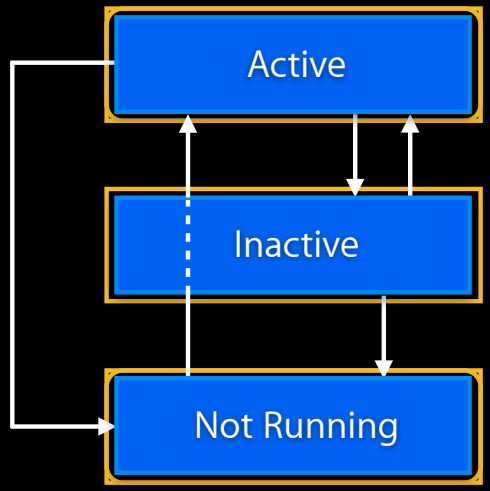
- Build with iPhone SDK 4
- Fast app switching enabled by default
- Background audio, location, and VoIP require explicit declaration in app's Info.plist



App Life Cycle

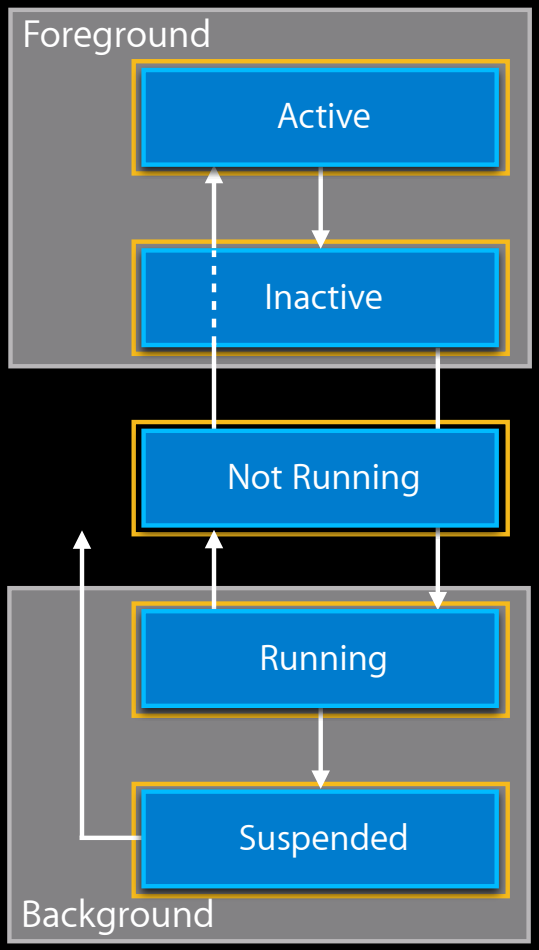
App Life Cycle

iPhone OS 3 review



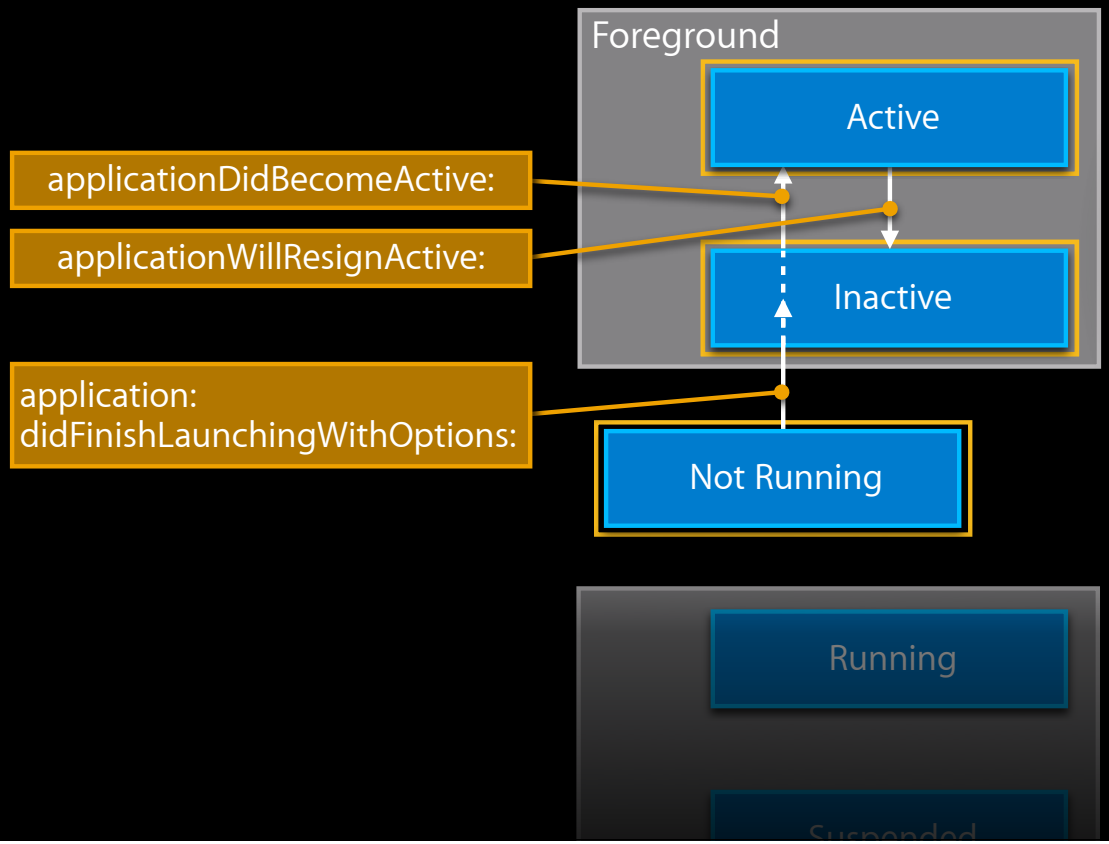
App Life Cycle

iOS 4



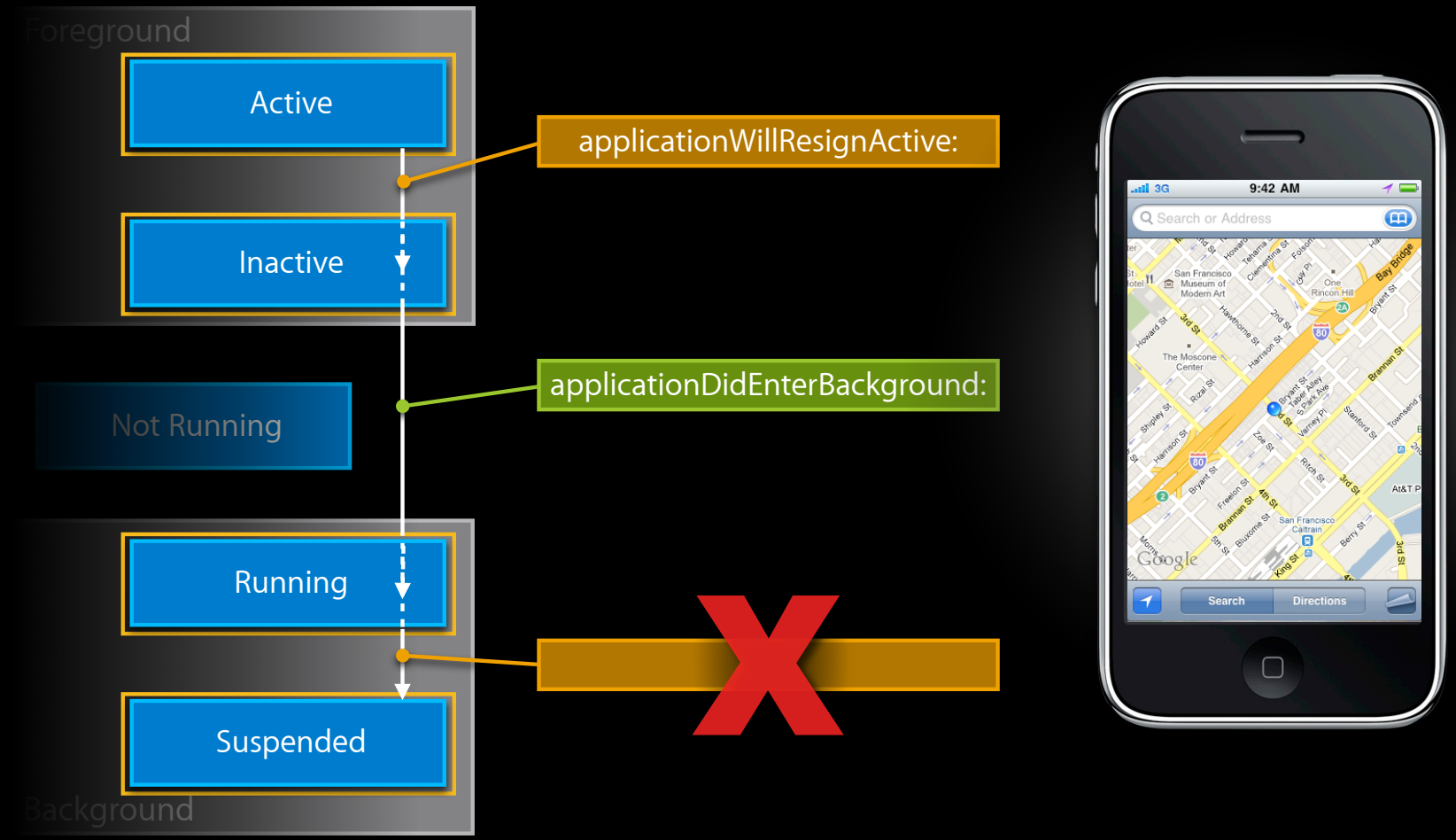
UIApplicationDelegate Callbacks

Launch and active/inactive



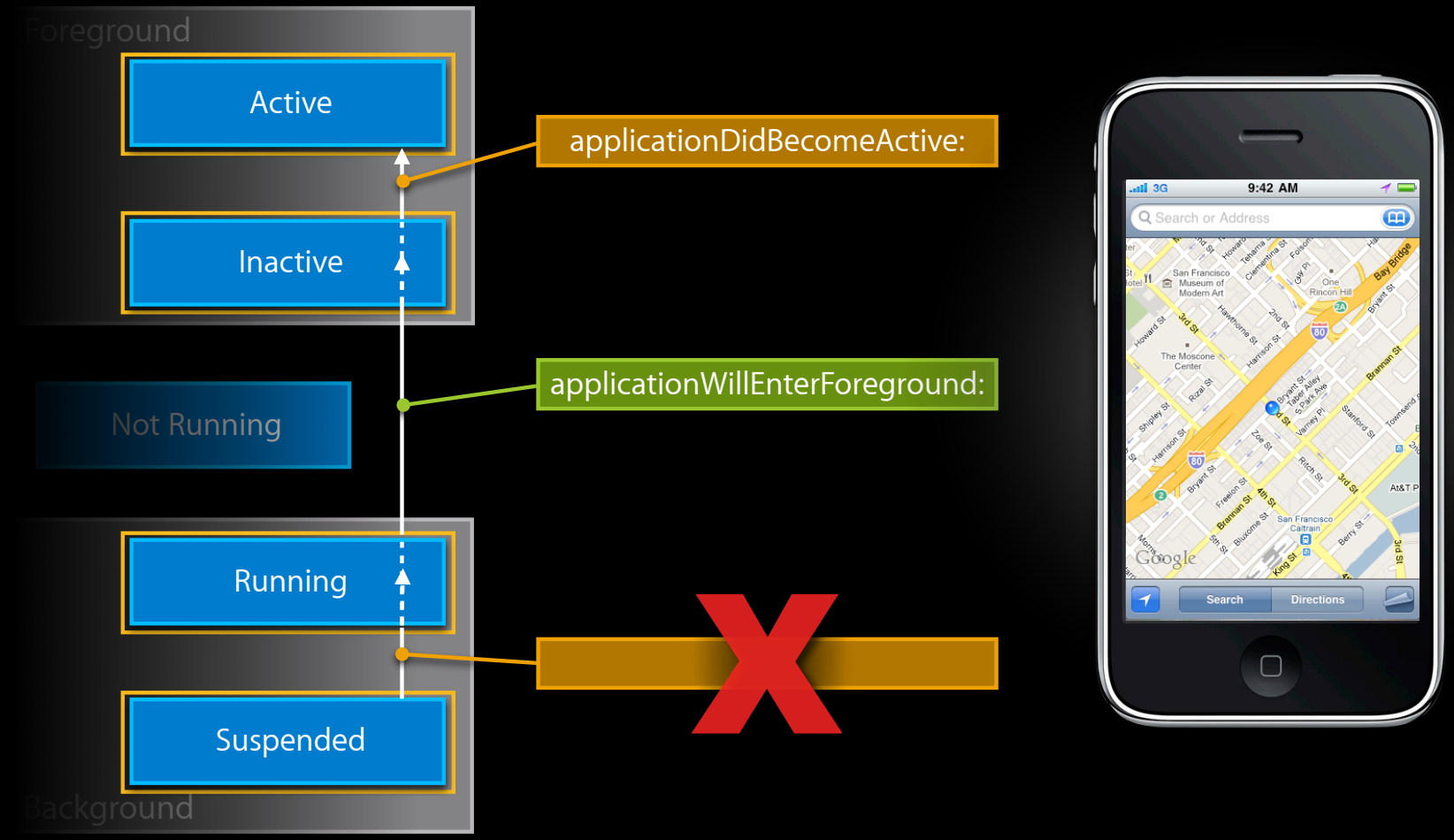
UIApplicationDelegate Callbacks

Switching from an app



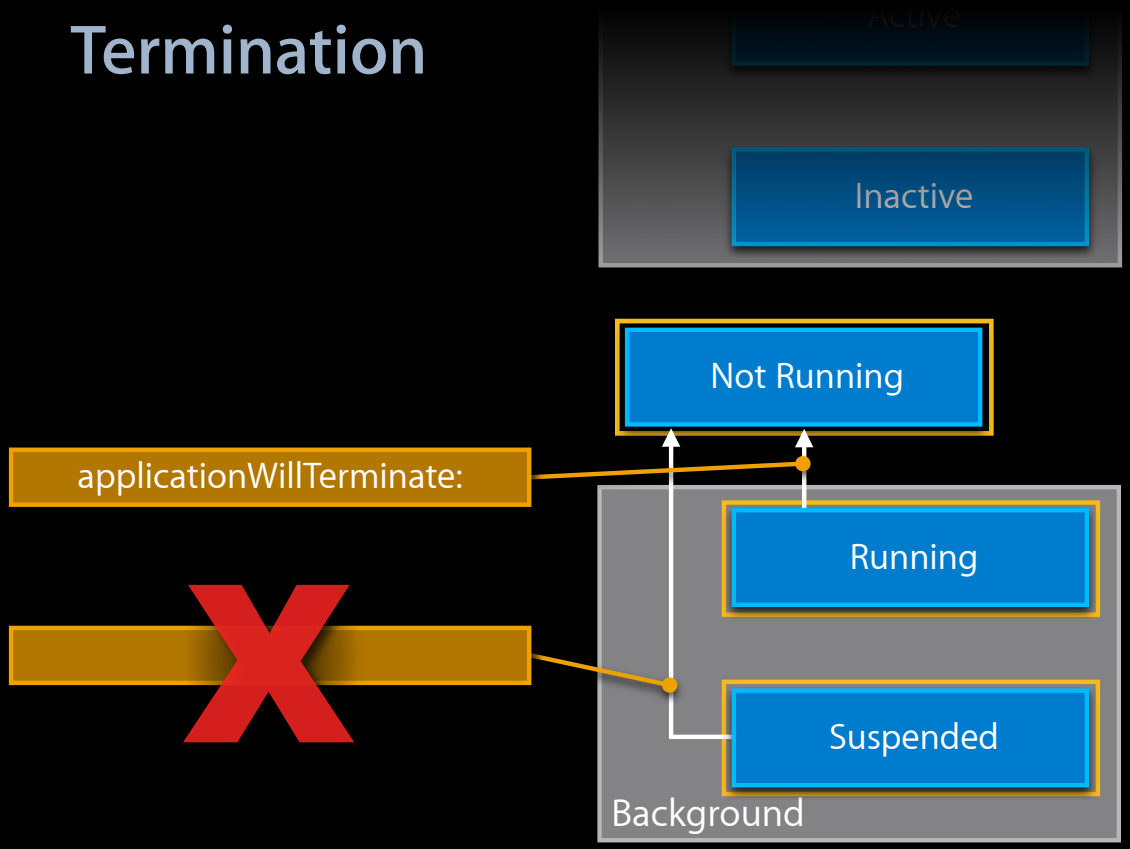
UIApplicationDelegate Callbacks

Switching to an app



UIApplicationDelegate Callbacks

Termination



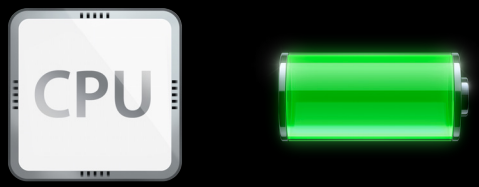
Life Cycle Notifications

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

Responsibilities and Best Practices

System Resource Management

- Goals
  - Preserve foreground app usability
  - Preserve battery life



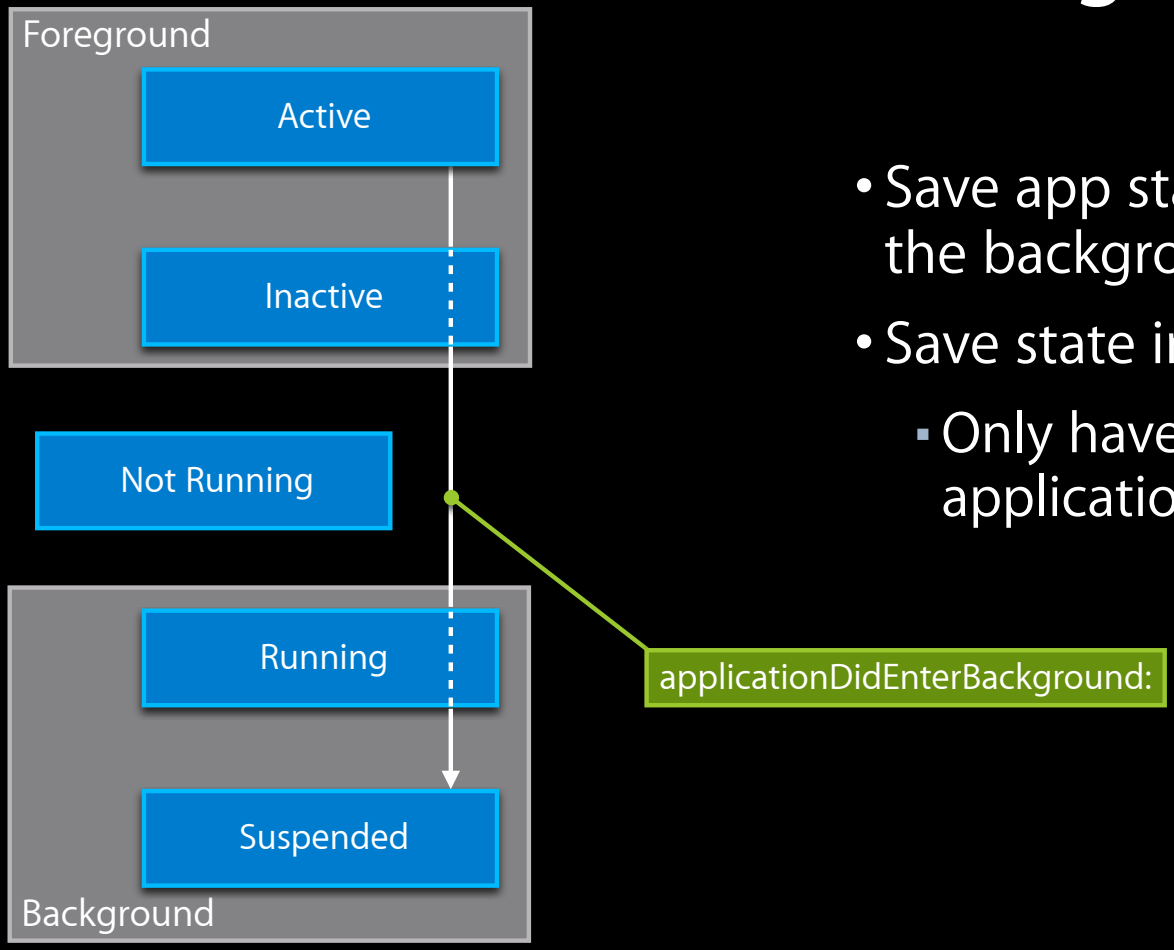
- System resources shared by all apps
- Multitasking apps should minimize system footprint in the background

Responsibilities and Best Practices

Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

Saving App State



- Save app state when entering the background
- Save state incrementally
  - Only have a few seconds in applicationDidEnterBackground:

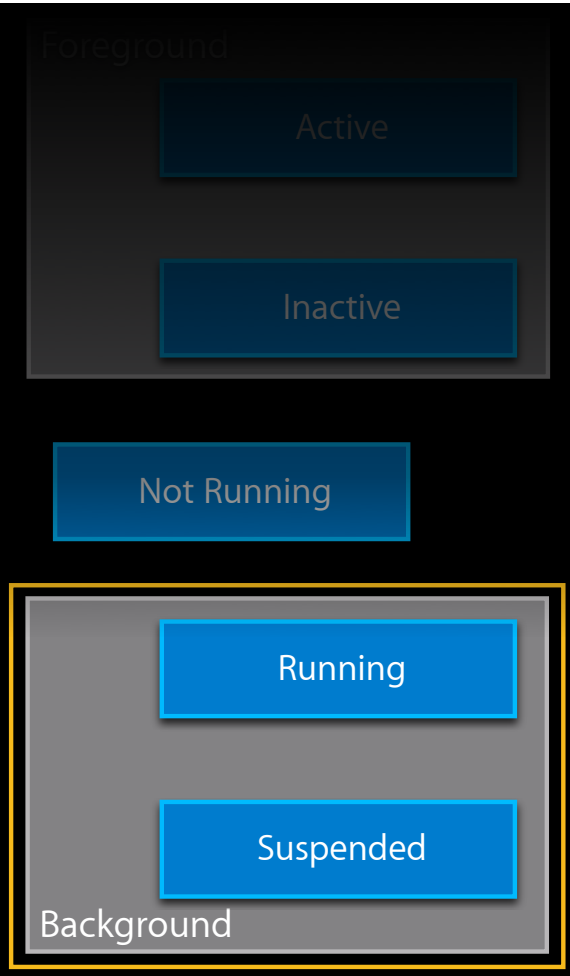
Responsibilities and Best Practices

Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

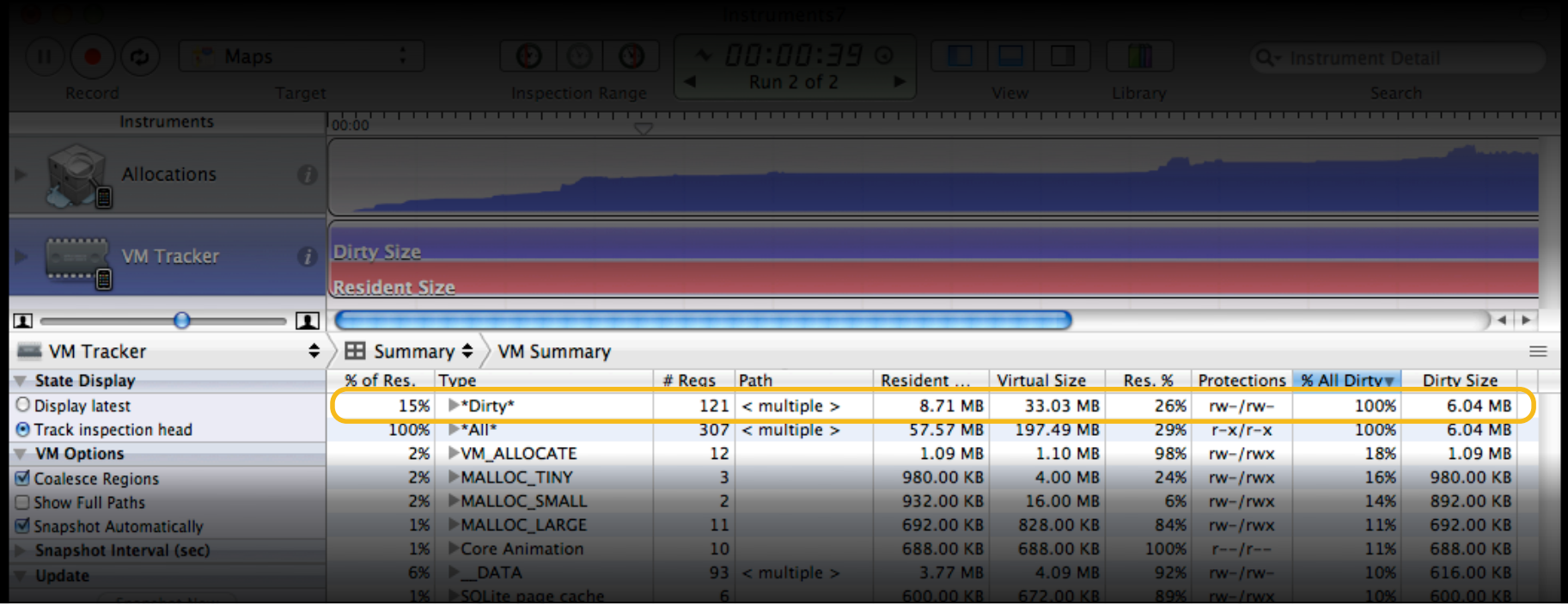
System Memory

- Memory is a limited resource
- System terminates apps when memory exhausted
- Priority
  - Suspended (high memory usage)
  - Background running (high memory usage)
  - Suspended (low memory usage)
  - Background running (low memory usage)
- Termination due to low memory is a normal condition



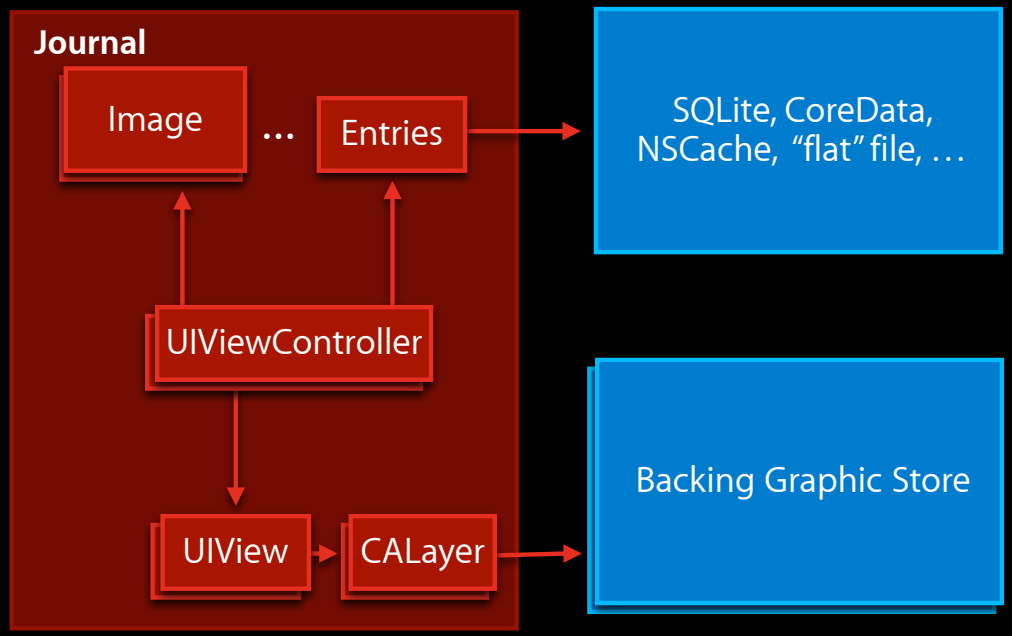
Reducing Memory Usage

- Using less memory allows more apps to remain alive
- Low memory usage increases chance that your app stays alive
- Reduce memory use in applicationDidEnterBackground:



App Memory Usage

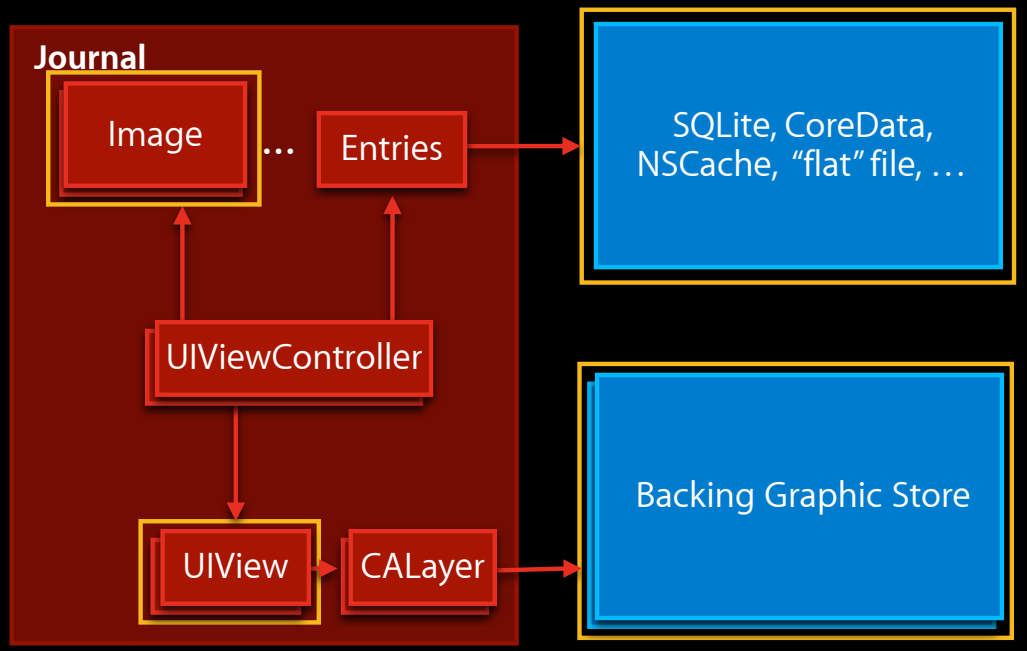
- App state
- App files and databases
- Views and layers
- Backing stores
- View controllers



Reducing Memory Usage

Reductions performed by system

- Free backing graphic store
- Release non-visible UIViewController views
- Flush [UIImage ImageNamed] cache
- Reclaim SQLite page cache, CoreData, NSCache



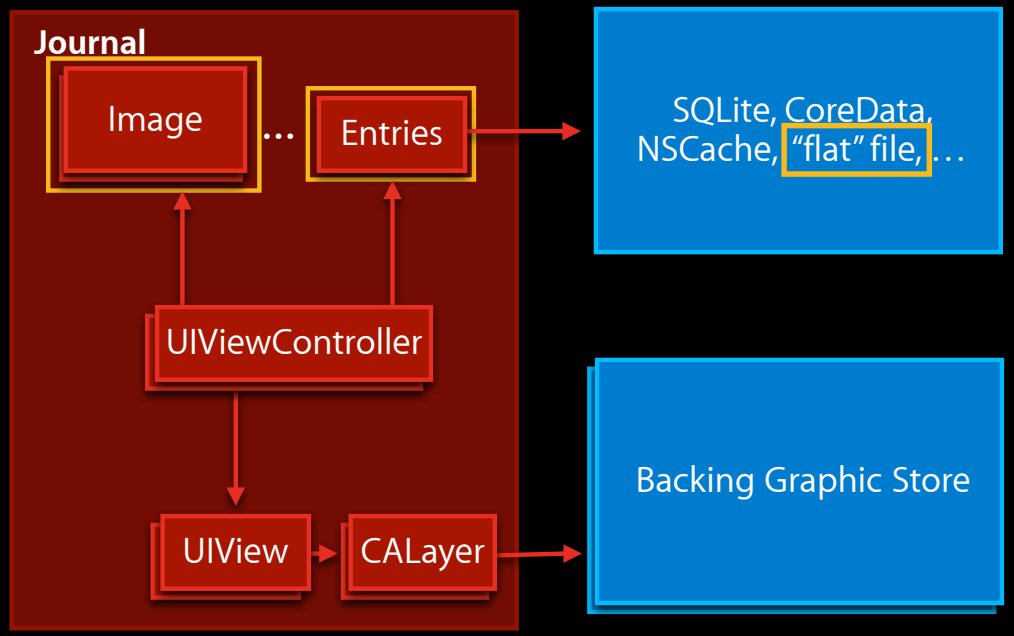
Reducing Memory Usage

Reductions performed by your app

- Release images

```
[myImage release];
```

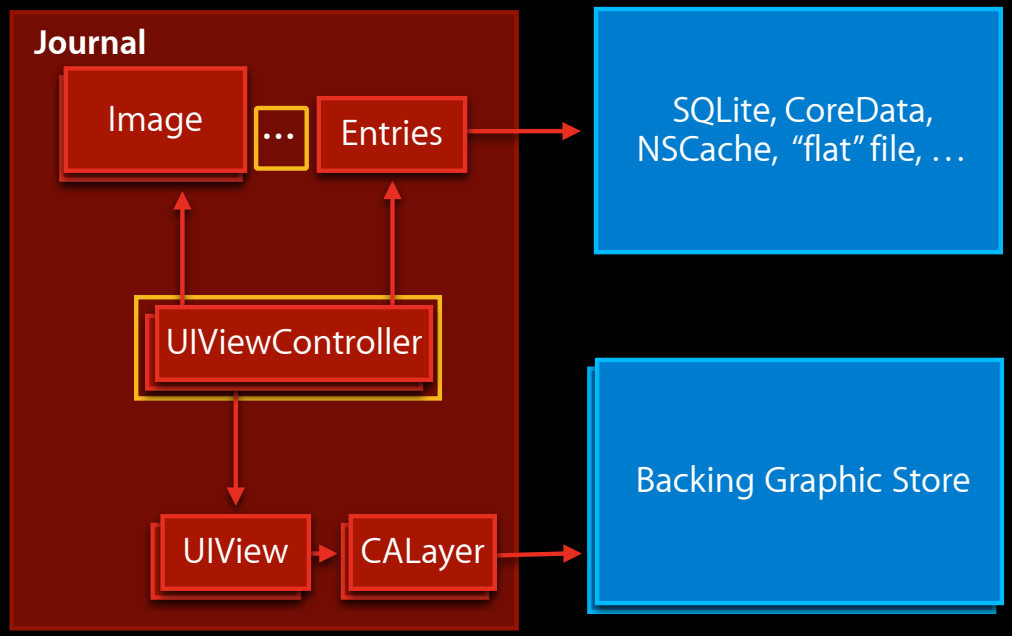
- Flush caches of data that can be regenerated
- For read-only or sparsely written files use mmap()
  - System only uses memory as needed
  - Read-only memory can be reclaimed without terminating app



Reducing Memory Usage

Tradeoffs

- Typically don't release view controllers
- Other state—Must tradeoff memory savings vs. time to recreate on resume



Responsibilities and Best Practices

Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

Prepare UI for Background

- Pause app if appropriate
  - Update UI to show pause
  - Typically performed in applicationDidResignActive:
- Remove alerts and actionsheets if appropriate

```
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex
                              animated:(BOOL)animated
```

- Prepare for screenshot
  - Hide sensitive information
  - Stop animations

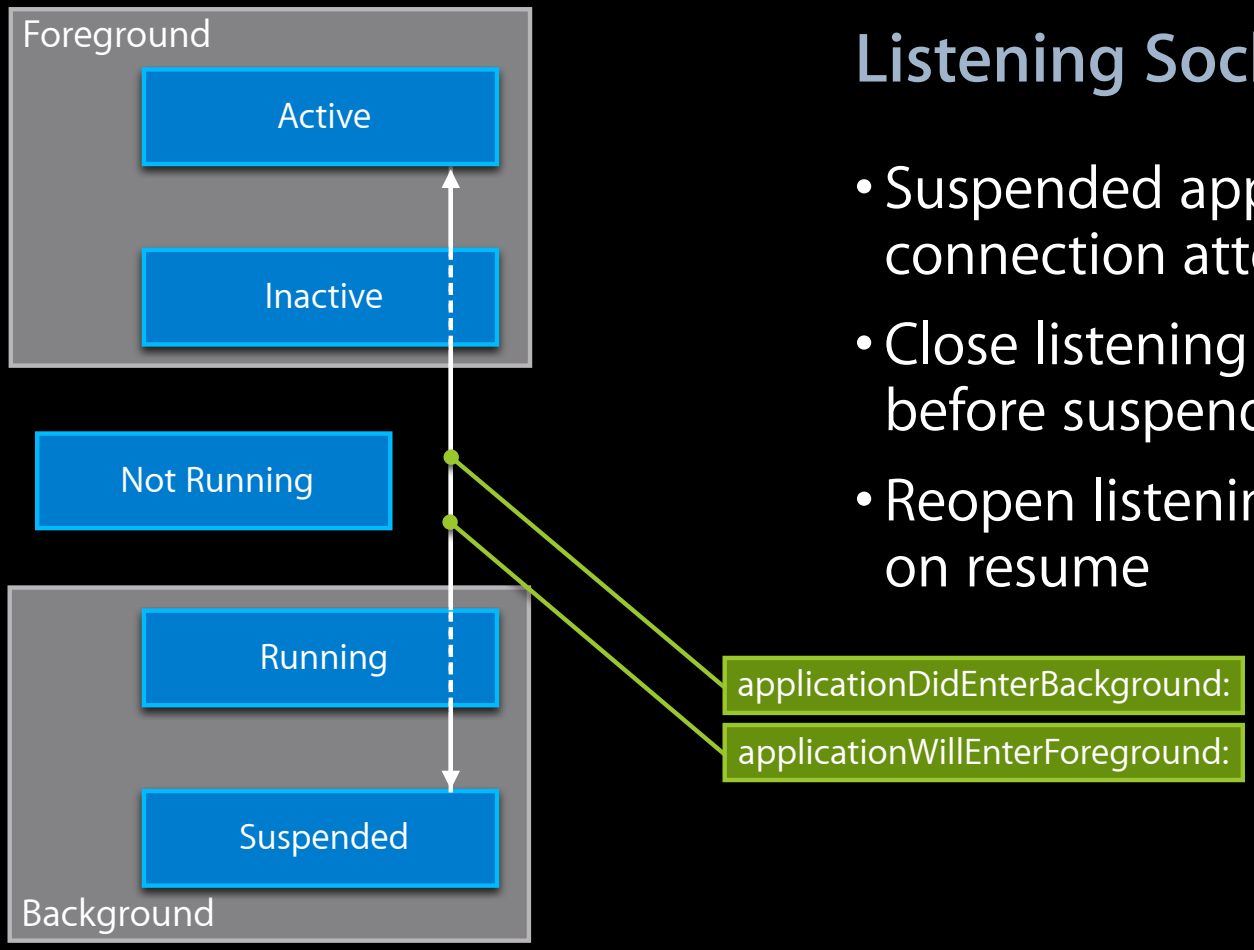
Responsibilities and Best Practices

Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

Network

Listening Sockets

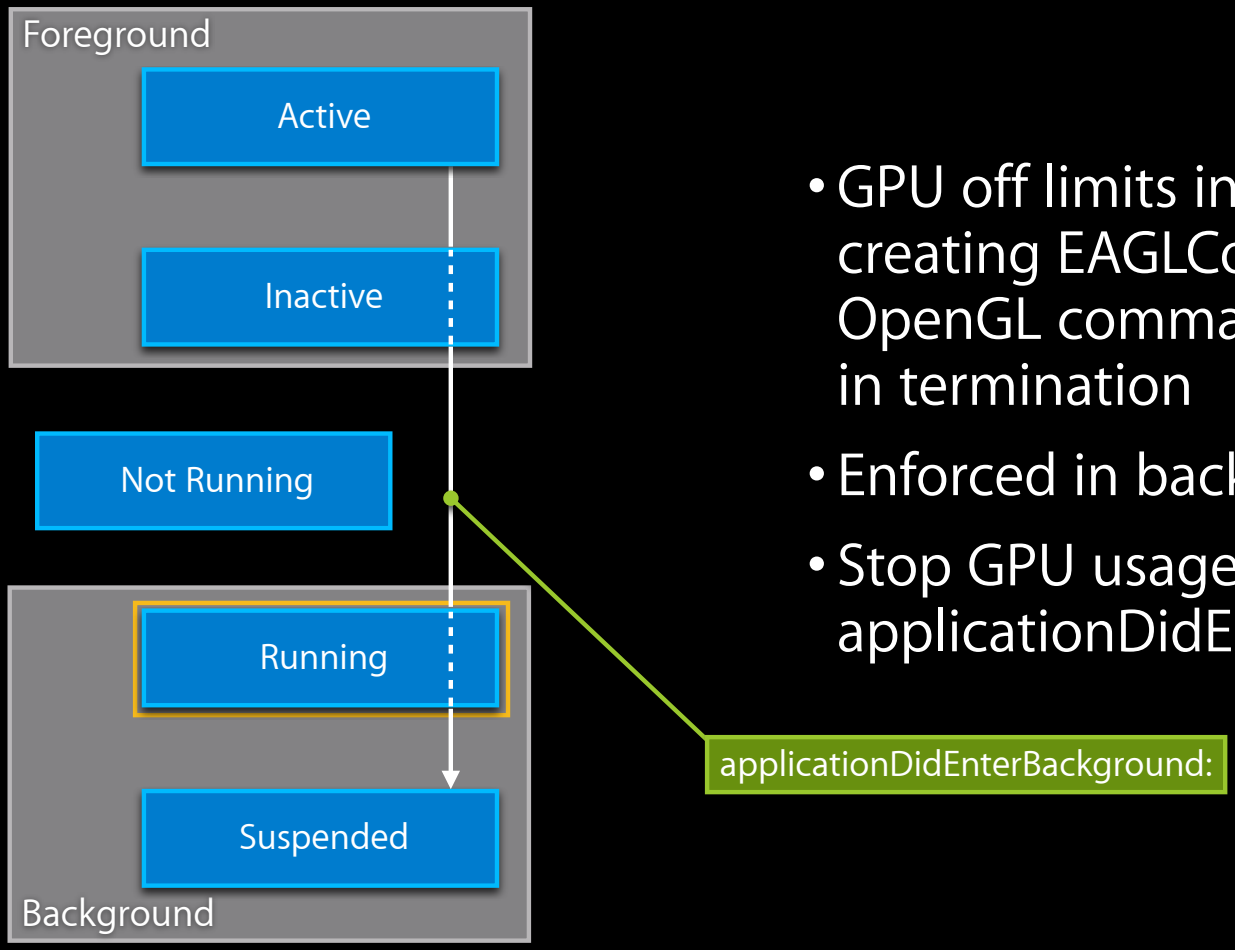


- Suspended app cannot respond to connection attempts
- Close listening sockets before suspend
- Reopen listening sockets on resume

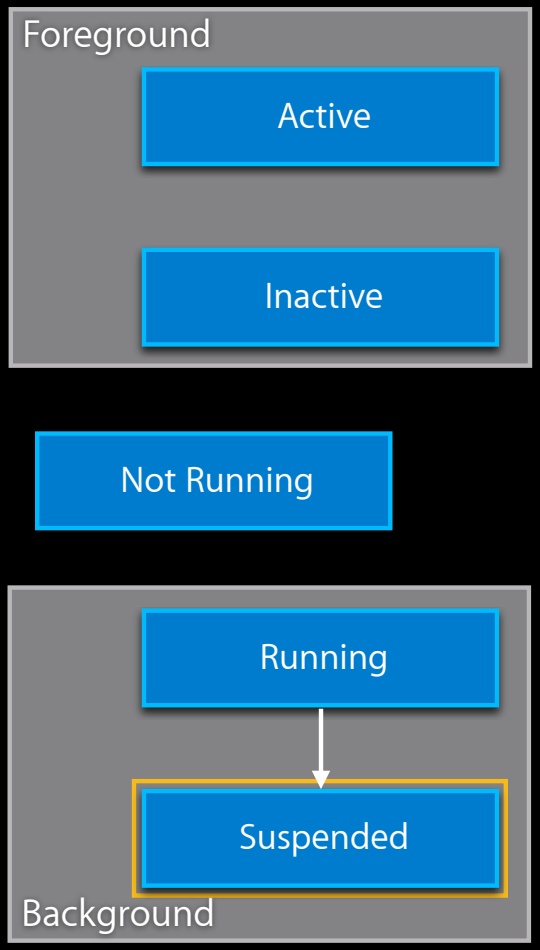
Bonjour

- Suspended app cannot accept incoming connection attempts
  - Should not advertise service
  - Should not browse for service
- Bonjour operations may be cancelled while app suspended
- When app resumes
  - Be prepared for errors
  - Restart Bonjour services if necessary
  - Update UI

GPU



- GPU off limits in background—creating EAGLContext or issuing OpenGL commands results in termination
- Enforced in background running state
- Stop GPU usage before returning from applicationDidEnterBackground:



Shared System Data

- Holding exclusive access to shared data while suspended not allowed—results in termination
- Shared data accessed by API
  - Calendar
  - Address book
  - Music and Media libraries
- Enforced upon entry to suspended state
- Stop API usage before returning from applicationDidEnterBackground:

Responsibilities and Best Practices

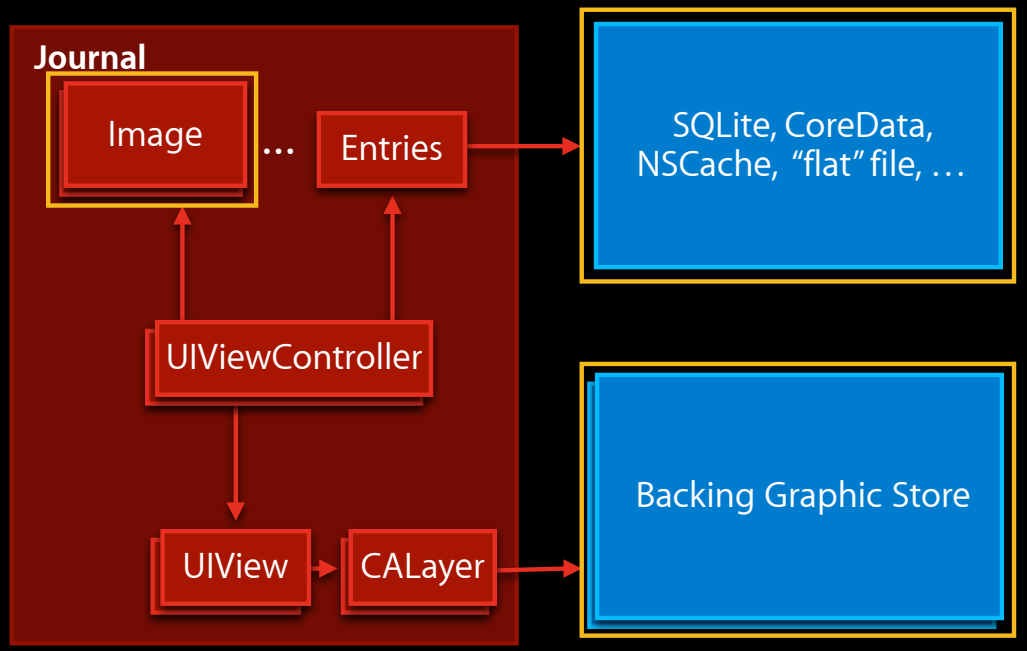
Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

Restore App State

Restores performed by system

- Backing graphic store
- ImageNamed cache will repopulate lazily as images are requested
- SQLite, CoreData, NSCache restored as needed
- Memory mapped files will reload as needed



Restore App State

Best practices

- No one right way to restore app state on resume
- Tradeoff—Memory size reduction vs. time and effort to restore state

- Consider lazy restore for your app data
  - Release objects as entering background
  - Restore objects on demand—As they are needed

Restore App State

Lazy state restore

```
- (MyData *) dataAtIndex:(unsigned)index {
    if (_data[index] == nil) {
        _data[index] = [DataController loadDataAtIndex:index];
        [_data[index] retain];
    }
    return _data[index];
}
```

```
- (void) applicationDidEnterBackground:(UIApplication *)app {
    ...
    for (i = 0; i < _dataCount; i++) {
        [_data[i] release];
        _data[i] = nil;
    }
}
```

Responsibilities and Best Practices

Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

System Changes While Suspended

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

- App must be prepared to handle burst of notifications
  - Avoid delaying app responsiveness
  - Avoid rapid UI updates

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

Responsibilities and Best Practices

Outline

- Entering the background
  - Save app state
  - Reduce memory usage
  - Prepare UI
  - Stop Bonjour and network listening
  - Stop GPU usage
  - Stop shared system data access
- Resuming from suspend
  - Restore app state
  - Handle system changes
  - Restore networking

Network

Established Connections

- Network connections can be lost for many reasons
  - Network conditions
  - Device location
- When suspended app resumes, app must be prepared for lost network connections
  - Handle errors
  - Reestablish connections

Example applicationDidEnterBackground:

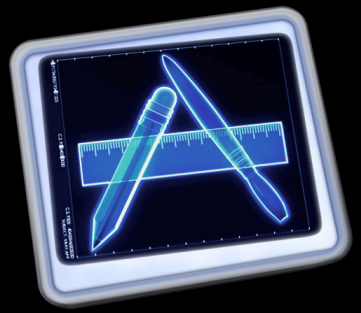
```
- (void) applicationDidEnterBackground:(UIApplication *)application {
    // save app state
    [self saveState];

    // reduce memory usage
    [self releaseImages];
    [self flushCaches];

    // prepare UI
    [self cancelAlert];
    [self pauseUI];

    // close listening sockets
    [self closeServiceSocket];
}
```

Example applicationWillEnterForeground:

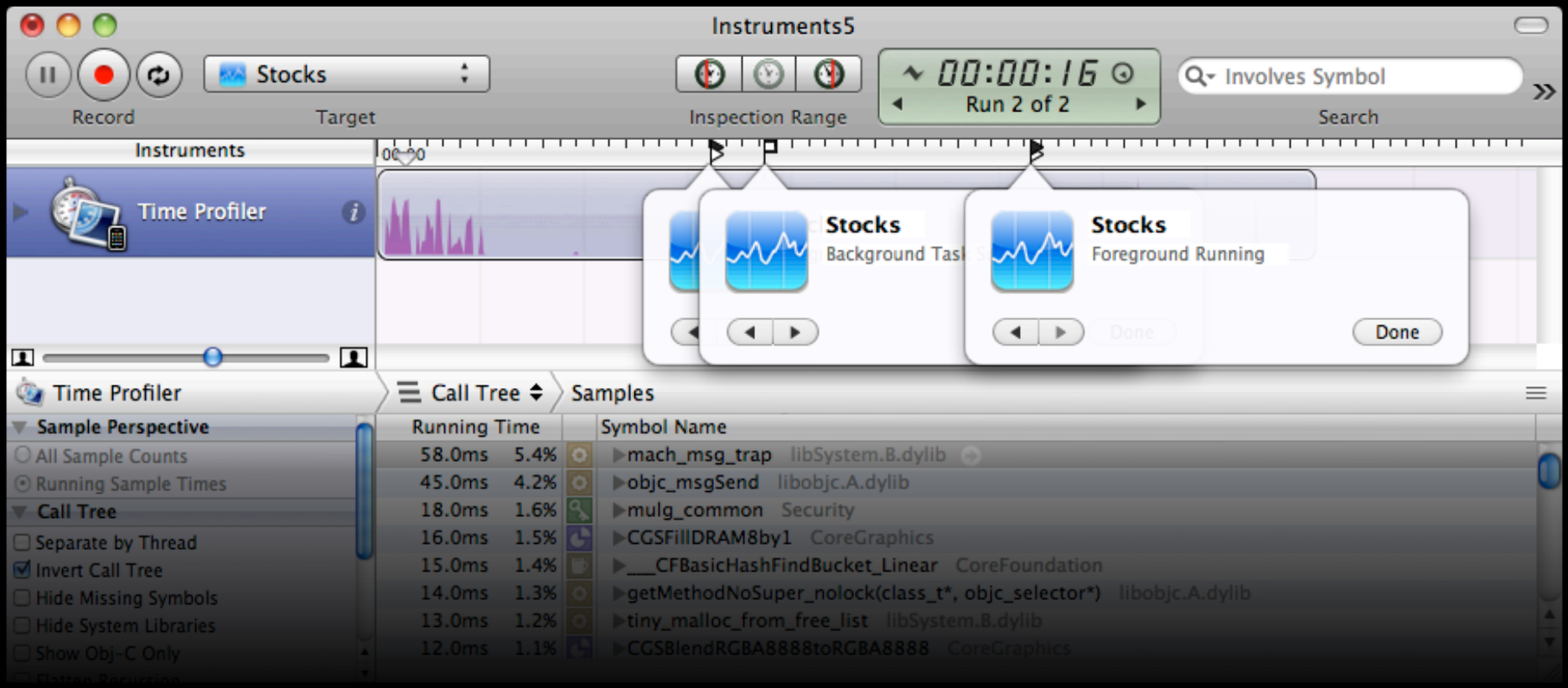
```
- (void) applicationWillEnterForeground:(UIApplication *)application {
    // restoring state lazily so no explicit restore

    // open listening sockets
    [self openServiceSocket];
}
```

Multitasking and Development Tools

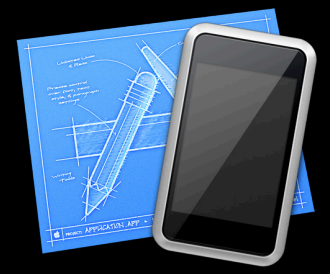
Instruments



- App life cycle events are flagged



Simulator



- Simulator is not a replacement for device
- Supported
  - Fast app switching
  - Task completion
  - Local notifications
  - Multitasking UI
- Not supported
  - Background audio, location, and VoIP
  - Significant location change, region monitoring
  - Push notifications

Debugger

- Debugging app changes some behavior,
  so must also test outside of the debugger
- No time limit in UIApplicationDelegate callbacks
- No time limit in task completion handler

Summary

- iOS 4 provides multitasking benefits
  while preserving device usability and battery life
- All apps can benefit from fast app switching
  - Resume quickly
  - Preserve state
  - Fully integrate with multitasking UI
- Additional multitasking services enable app improvements
  and new classes of apps

Related Sessions

Adopting Multitasking on iPhone OS, Part 2	Mission Tuesday 3:15PM
Mastering Xcode for iPhone OS Development, Part 1	Mission Tuesday 2:00PM
What's New In Instruments	Presidio Wednesday 11:30AM
Introducing Blocks and Grand Central Dispatch on iPhone	Russian Hill Wednesday 11:30AM
Simplifying Networking Using Bonjour	Nob Hill Wednesday 10:15AM
Core OS Networking	Pacific Heights Tuesday 9:00AM

Labs

Multitasking Lab 1	Application Frameworks Lab D Tuesday 4:30PM
Multitasking Lab 2	Application Frameworks Lab A Wednesday 2:00PM
Multitasking Lab 3	Application Frameworks Lab A Friday 11:30AM

More Information

Michael Jurewitz
Developer Tools and Peformance
jurewitz@apple.com

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

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

Apple Developer Forums
http://devforums.apple.com


WWDC10