Performance Optimization on iPhone OS

Erik Neuenschwander
Manager, iOS Performance

Ben Weintraub
iOS Performance Engineer

Introduction

- Performance is important!
- Key aspect of App Store reviews
- You have the tools and skills to improve performance
- Today: Cover common cases and strategies

What You'll Learn

- How to test and measure performance scenarios
- How to improve key scenarios
  - Speedy launches
  - Smooth scrolling
  - Slim memory footprint
- How to prioritize performance issues

Measuring Performance

Measuring Performance

Techniques

- Measure first
- Manual testing
- Automated testing

Measuring Performance

Techniques

- What numbers should you measure?
- Everything affects performance

Measuring Performance

Techniques

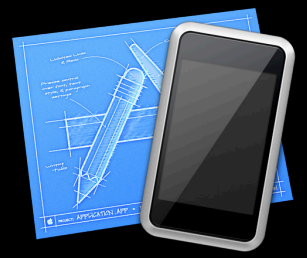
- Guessing is overrated
- Focus on scenarios
  - Measure things in turn
  - Change
  - Re-test
- It has to feel right

Measuring Performance

Tools

- Logging
  - `NSLog(@"That took %g seconds\n", timeWeWaited);`
  - Log to file
- Instruments
- Simulator?

Simulator or Device?



- Simulator is fast and easy
  - Uses Mac hardware
  - Unrealistic “iOS” performance profile
- Exception: memory footprint
  - Simulator is a good model
  - More speed and features in Instruments
- Device is the final arbiter
  - Use for all speed-related testing
  - Verify memory fixes perform as expected

Measuring Performance

Tips

- Measure early, measure often
- Record “good” results as a baseline
- Turn off or remove logging for submitted app!
- Test every device you will support
- At minimum test the oldest device (today: iPhone 3G)

Key Scenarios

Speedy Launches

Speedy Launches

Importance

- First thing a user experiences or demos
- Very common scenario
  - Non-multitasking “launch”
  - Multitasking “resume”
- Slow performance will cause OS to abort your app
  - Maintain system responsiveness

Speedy Launches

Watchdog

- The OS is watching you
- "Wall clock" time to gate
- Values subject to change

	Maximum Time
Launch	20 sec
Resume	10 sec
Suspend	10 sec
Quit	6 sec
Complete operation (iOS 4)	10 min

Speedy Launches

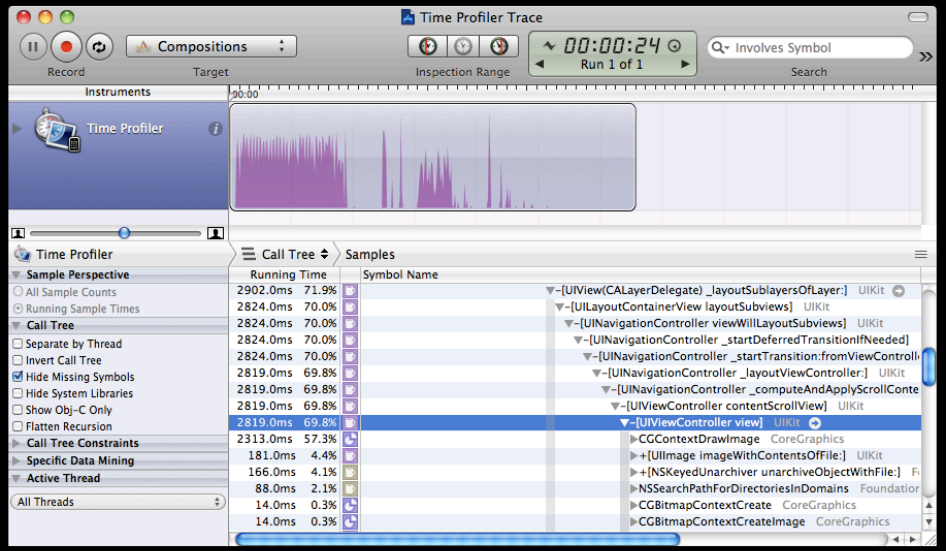
Measuring launch

- Test with realistic data sets
- Use Time Profiler Instrument

Speedy Launches

Time Profiler

- Collects backtraces at regular intervals
- Useful to see where execution time goes
- Look for work to defer, do on demand
- Necessary work
  - sort by time
  - speed the slowest parts



Demo—Speedy Launches

Time Profiler Instrument

Speedy Launches

Tips

- System watchdog will terminate slow apps
- Collect trace with the Time Profiler Instrument
- Do less work
  - Defer work out of startup
- Do not block on slow operations
  - Never do networking on your main thread
- Optimize time-consuming activities
  - Launch-time data set needs to have a size limit
- Collect new traces to quantify results

Smooth Scrolling

Smooth Scrolling

Importance

- Table Views are popular
- Direct Manipulation UI demands responsiveness

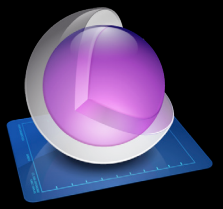
Smooth Scrolling

Measuring scrolling

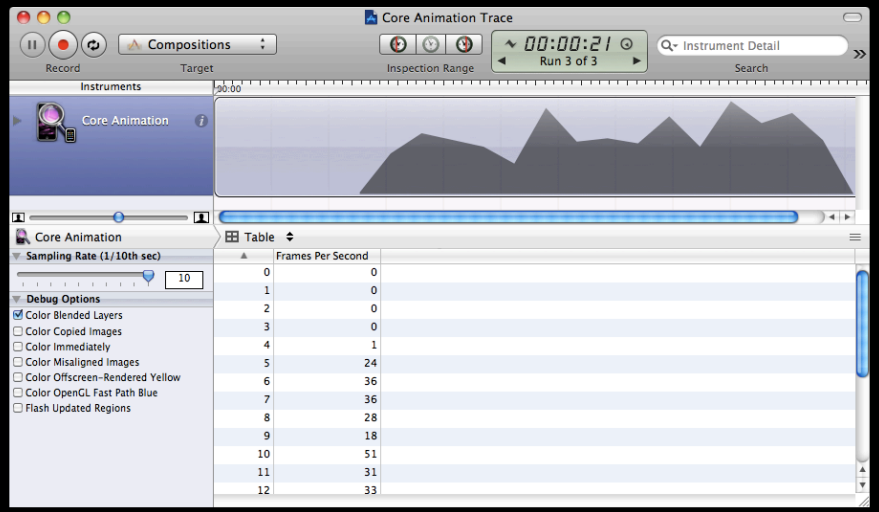
- Frames Per Second (FPS—“fips”)
- 60 FPS == smooth
- Core Animation Instrument

Smooth Scrolling

Core Animation



- Measures FPS in realtime
  - Sub-second requires you to do math
  - Example: 18 frames in 0.3 s == 60 FPS
- Visual hints give insight into important rendering details
  - Color Blended Layers
  - Many others

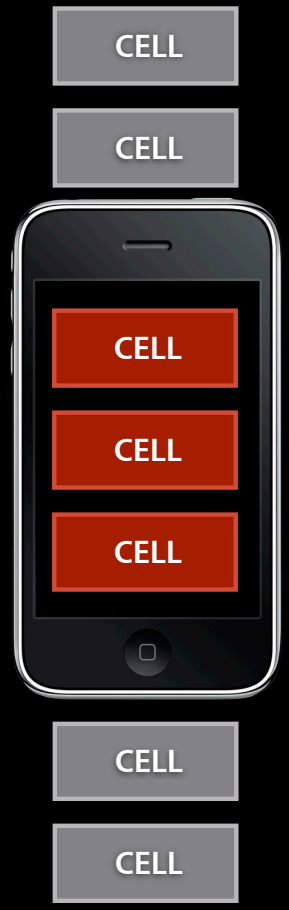


Demo—Smooth Scrolling

Testing and measurement

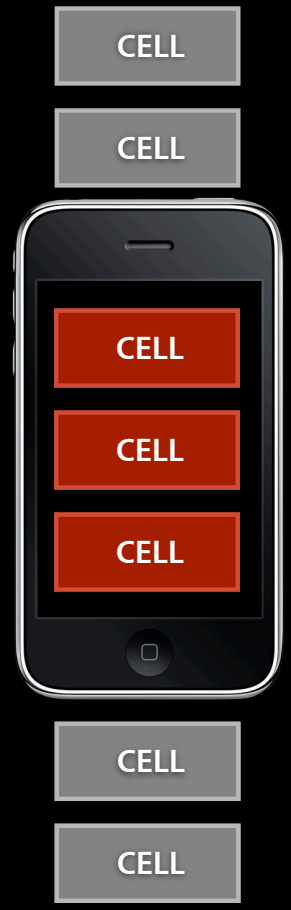
Smooth Scrolling—Cell Reuse

Before



Smooth Scrolling—Cell Reuse

After



Demo—Smooth Scrolling

Efficient drawing

Smooth Scrolling

Tips

- Test scrolling scenarios
  - Easy to see via manual testing
  - Automated testing now supported
- Measure FPS with Core Animation Instrument
  - 60 FPS == good
- Reuse cells
  - `-[UITableViewCell initWithStyle:reuseIdentifier:]`
  - `-[UITableView dequeueReusableCellWithIdentifier:]`
- Stay opaque/green
  - Even for `UILabel`! OS will handle "blue background" for you

Slim Memory Footprint

Slim Memory

Importance

- iOS has no swap
- Under memory pressure, OS will terminate apps
  - Maintain system stability

Slim Memory

Jetsam

- Watches memory pressure
- Instant lightweight termination of applications
- More critical with multitasking
  - Apps with small footprint preserved longer
- Stay safe, stay low

Slim Memory

Areas to focus on

- Avoidable spikes
- Leaks
- Abandoned memory

Slim Memory

Avoidable spikes

- Definition: individual brief allocations all present simultaneously
- Processing large quantities of data
  - Break into independent batches
- Autoreleased objects
  - Reduce object lifetimes

Slim Memory

Autorelease

- “Used to avoid worrying about retain/release”

Slim Memory

Autorelease

- ~~"Used to avoid worrying about retain/release"~~
- Used to return objects without retaining them
  - Caller will retain if needed
- Objects added to an `NSAutoreleasePool`
- Pool calls `release` at the next turn of the runloop
- Deallocation may happen then, if retain count goes to zero
- In the meantime, memory can spike

Demo—Slim Memory

Allocations Instrument

Slim Memory

Autorelease

- A little more expensive than retain/release
- In designing your code
  - Only use at framework boundaries
  - Explicit retain/release when lifetime is managed by one entity
- When using API
  - Use Instruments to watch memory allocation pattern
  - Wrap intensive autorelease API uses in nested autorelease pool

Slim Memory

Leaks

- Definition: allocated memory that is inaccessible
- Leaks Instrument examines the heap for leaked memory
- Most useful when app launched from the Instrument
- Identify moment of allocation
  - Not often the problem, but gives useful context
- Common mistakes
  - Unbalanced retain/release
  - Forget to release property's original value

Slim Memory

Abandoned memory

- Definition: leftover; accessible, but will never be used again
- Allocations Instrument offers Heapshot feature
- Two snapshots in time
- Look at (unexpected) differences

Demo—Slim Memory

Allocations Instrument

Slim Memory

Tips

- Jetsam will kill your app
- Leaks, abandonments, spikes
- Leaks Instrument
- Allocations Instrument with Heapshot
- Target autorelease use
- Nested autorelease pools

Prioritizing Performance Issues

Prioritization

Where performance fits

- There can be “show stopping” performance issues
- Establishing clear goals early can help consensus

Prioritization

Judging priority

- Frequency
  - Common scenario?
  - Consistently reproducible?
- Severity
  - Unresponsive for multiple seconds?
  - Single-digit FPS?
  - Cause a watchdog or Jetsam?

Prioritization

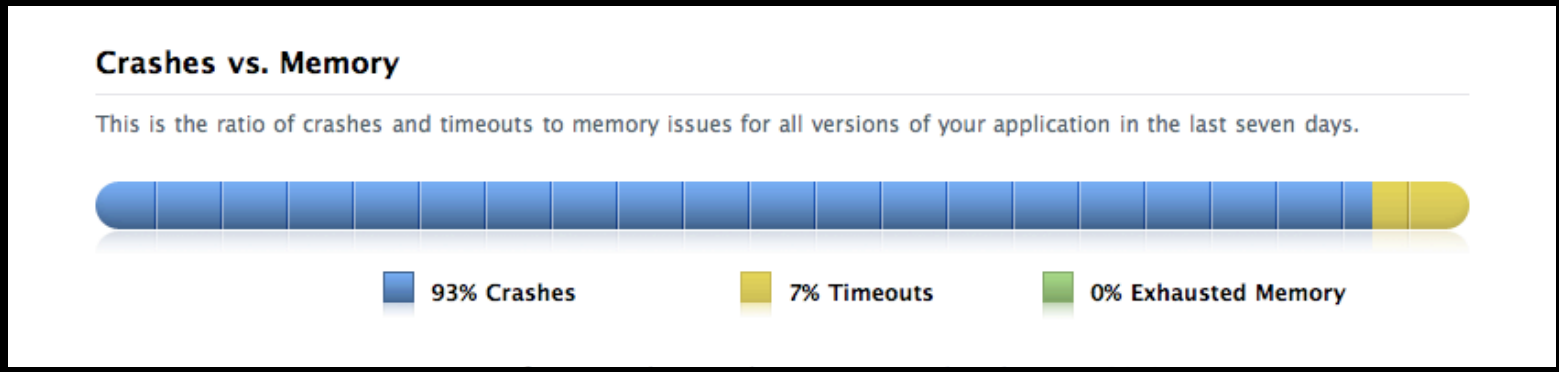
Judging priority

- Watchdog and Jetsam terminations look like a crash
- Top “crash” may not be a crash!

iTunes Connect

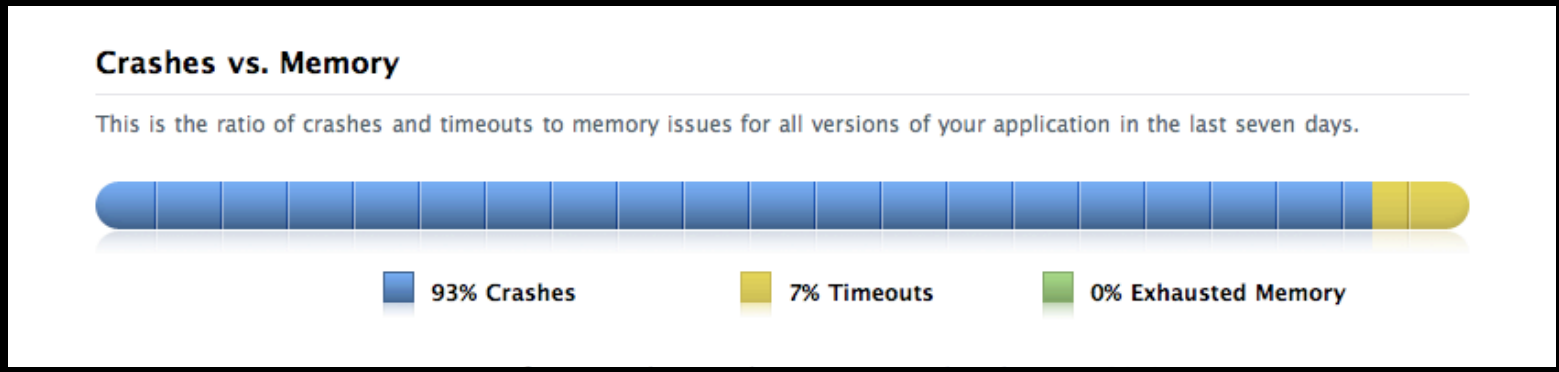
iTunes Connect

Weighting issues



iTunes Connect

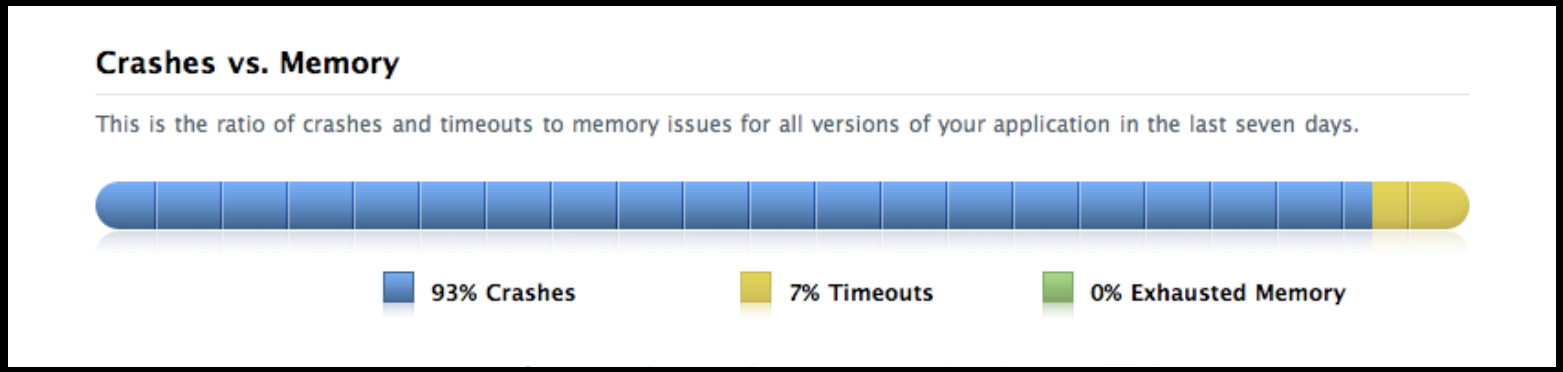
Watchdog reports

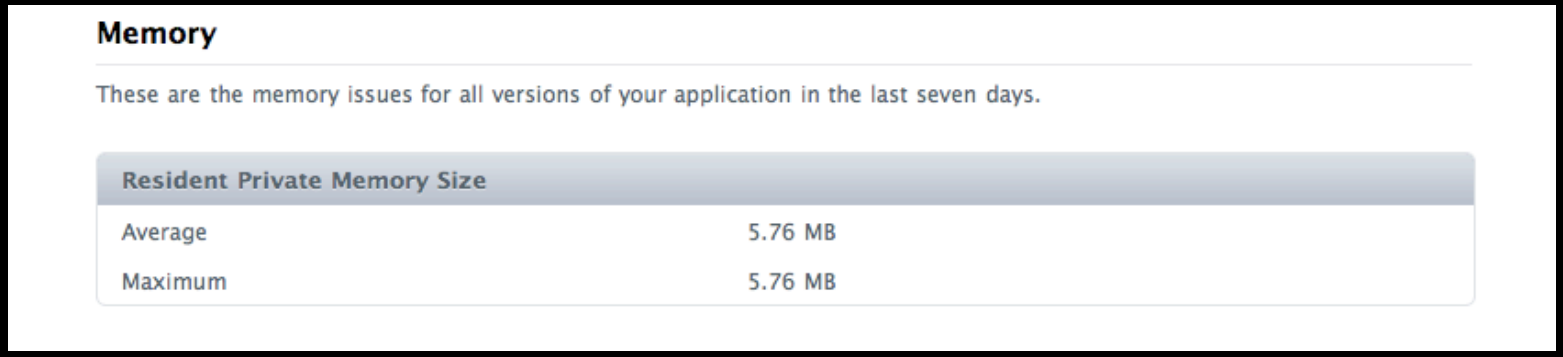


Timeouts		
Timeout on Launch	67% of submitted timeouts	Download Report
Timeout on Quit	33% of submitted timeouts	Download Report
Force Quit by User	0% of submitted timeouts	No Report Available

iTunes Connect

Jetsam reports





Wrapping Up

Review

- Performance is critical
- Measure with Instruments
- Performance testing is best done on an older device
- Key Areas
  - Launch
  - Scrolling
  - Memory
- Develop clear performance goals
- Visit iTunes Connect for performance-related reports

More Information

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

Michael Jurewitz
Developer Tools and Performance Evangelist
jurewitz@apple.com

Documentation
iPhone OS Performance Overview
http://developer.apple.com/iphone/library/documentation/Performance/Conceptual/PerformanceOverview/

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

Related Sessions

Advanced Performance Optimization on iPhone OS, Part 1	Mission Thursday 3:15PM
Advanced Performance Optimization on iPhone OS, Part 2	Mission Friday 11:30AM
Optimizing Core Data Performance on iPhone OS	Presidio Thursday 4:30PM

Advanced Memory Analysis with Instruments	On Video
Advanced Performance Analysis with Instruments	On Video
Building Animation Driven Interfaces	On Video

Labs

iPhone OS Performance Lab	Developer Tools Lab A Thursday, 4:30PM
Animation Lab	Application Frameworks Lab C Thursday, 4:30PM
OpenGL ES Lab	Graphics and Media Lab A Thursday, 9:00AM
iPhone OS Performance Lab	Developer Tools Lab A Friday, 9:00AM


WWDC10