Managing and Optimizing Memory Usage in iPhone Applications

Danton Chin
danton@iphonedeveloperjournal.com

Intro

- Danton Chin
  - independent consultant
    - iPhone, Flex, AIR, Java development
    - experience in enterprise development and J2ME
    - blog at http://iphonedeveloperjournal.com/
    - email: danton at iphonedeveloperjournal dot com

Agenda

- 100 level or introduction to managing memory
  - Broad understanding of the memory landscape
- iPhone differences
- Memory Management Model
- Available tools for analyzing memory problems
- Resources
- Q&A

It's obvious but ...

≠

images from http://www.apple.com/

What are the differences?

CPU	Single core ARM chip (Samsung S5L8900)
Virtual Memory	• Limited • No swap space • 128 MB shared with system and other iPhone apps
GPU	PowerVR MBX Lite (Imagination Technologies)
Power	Battery
Other	GPS chip (3G), etc.

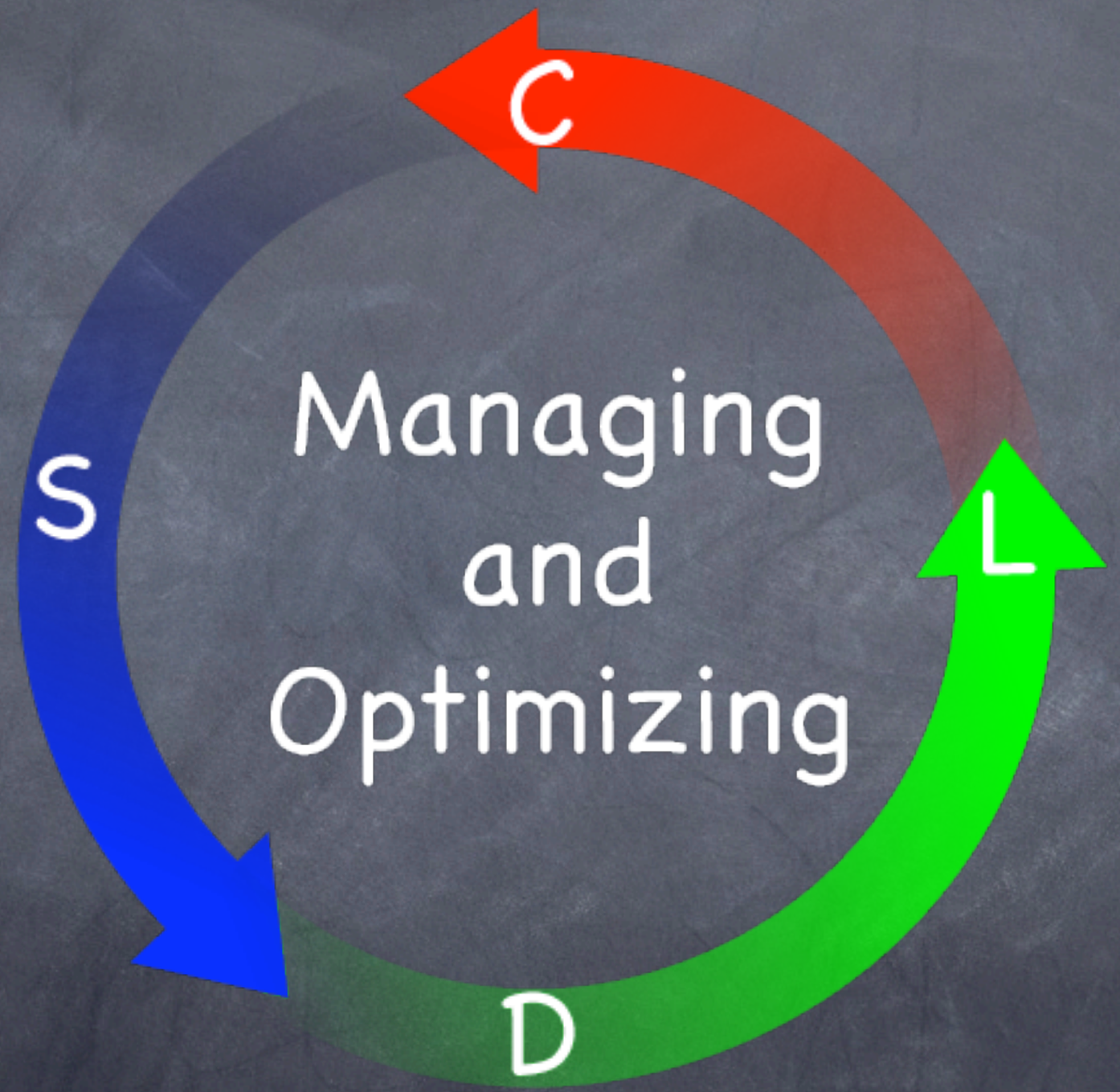
Big Job

Optimize Drawing Code

Minimize Battery Usage

Minimize Memory Usage

Optimize CPU Bound Code


C
S
Managing
and
Optimizing
L
D


San Jose, CA
360 iDev
slide to rock
March 2-4, 2009

Today

iPhone OS

- variant of Mac OS X
- 128 MB
  - Virtual memory system is always on but there is no swap file/space
  - Based on reports on the net your app may be able to use up to ≈ 46 MB before you run into a low memory condition
- No GC
  - must use the managed memory model or reference counting

Two Stage Response to Low Memory Condition

1. nonvolatile, static memory pages are removed such as code pages
   - volatile memory pages are never swapped out
2. once free memory falls below a "certain threshold" system will ask your application to free up memory

3 Ways to Register for Memory Warnings: 1

- implement in your application delegate

```
-(void) applicationDidReceiveMemoryWarning:(UIApplication *)application
{
    // release images, data structures, etc.
}
```

3 Ways to Register for Memory Warnings: 2

- overrride in your UIViewController subclass

```
-(void)didReceiveMemoryWarning:
{
    [super didReceiveMemoryWarning];
    // release any views that are not needed
}
```

3 Ways to Register for Memory Warnings: 3

- register to receive the notification

  UIApplicationDidReceiveMemoryWarningNotification

...

```
[ [ NSNotificationCenter defaultCenter ]
    addObserver:self
          selector:@selector(appropriateMethodName:)
            name:UIApplicationDidReceiveMemoryWarningNotification
            object: nil ];
```

...

Demo

- Using the Simulator to simulate low memory

applicationWillTerminate:

5..4..3..2..1..0

Application Memory Management and Optimization Goals

- Reduce the memory footprint of your application
- Eliminate memory leaks

Reduce Memory Footprint

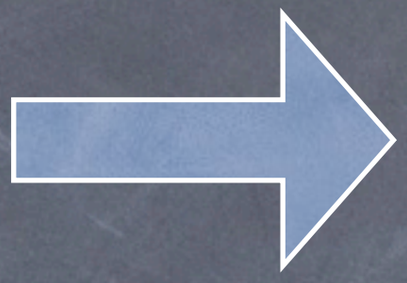
- Load resources lazily
- Reduce size of plists and images
    - use the NSPropertyListSerialization class to write the property list file out in a binary format
    - for images in PNG use pngcrush -iphone
- Limit the amount of data in memory
    - Use SQLite
- Use -mthumb compiler flag for apps without floating-point calculations

Managed Memory Model

- reference counting mechanism
- ownership metaphor

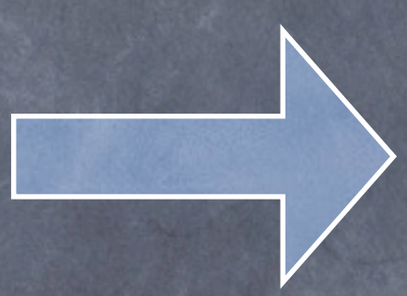
If you create it you own it

alloc



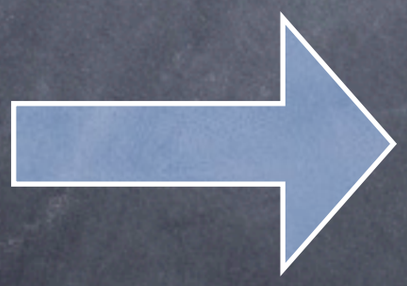
reference count: + 1

copy



reference count: + 1

retain



reference count: + 1

Relinquish Ownership

release

reference count

- 1

reference count

autorelease

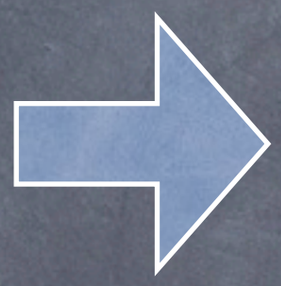
at some future

point in time

- 1

When no one owns an object ....

when reference count equals 0



destroyed/ deallocated/ memory is freed

Recap

image from Scott Stevenson's Learn Objective-C article at http://cocoadevcentral.com/d/learn_objectivec/

Objective-C Gangsta Sign

by Flickr member Dirtae

Dealloc

- implement a dealloc method
  - release instance variables
  - invoke dealloc in superclass

```
-(void)dealloc
{
   [ someInstanceVariable release];
   [super dealloc];
}
```

Convenience Constructors

- object instances returned from convenience constructors of the form +className... are NOT owned by you and do not need to be released

NSString *string

= [NSString stringWithFormat:@"360|iDev";

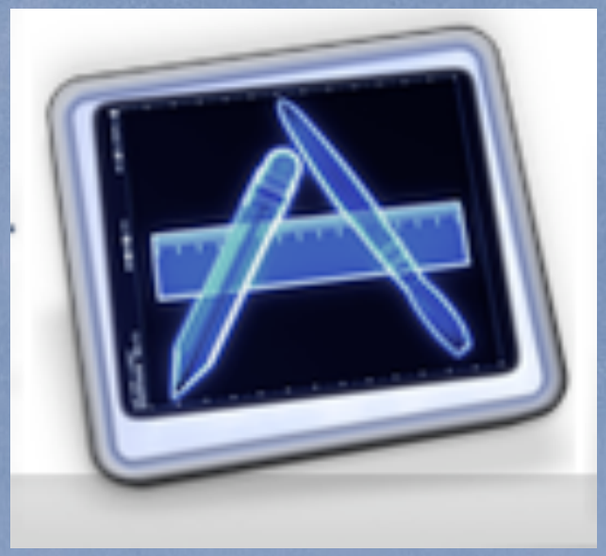
autorelease vs release: Why Wait?

- should be lazy about allocating memory
- but we should not be lazy about releasing an object once we are done with it
  - use release instead of autorelease when you can

Local Autorelease Pools

- if you create a lot of temporary autoreleased objects you may want to create a local autorelease pool to reduce the memory footprint of your app
  - autorelease pools are stacked
  - new pools are added to the top
  - autoreleased objects are added to the top autorelease pool
  - drained/deallocated pools are removed from the stack

What can we use?



Instruments

Shark

Clang Static Analyzer

DTrace

Works on Simulator and Device

Static analysis of code

Keep an eye on this one

Now

Early state

Future may be able to write your own

What tools don't work?

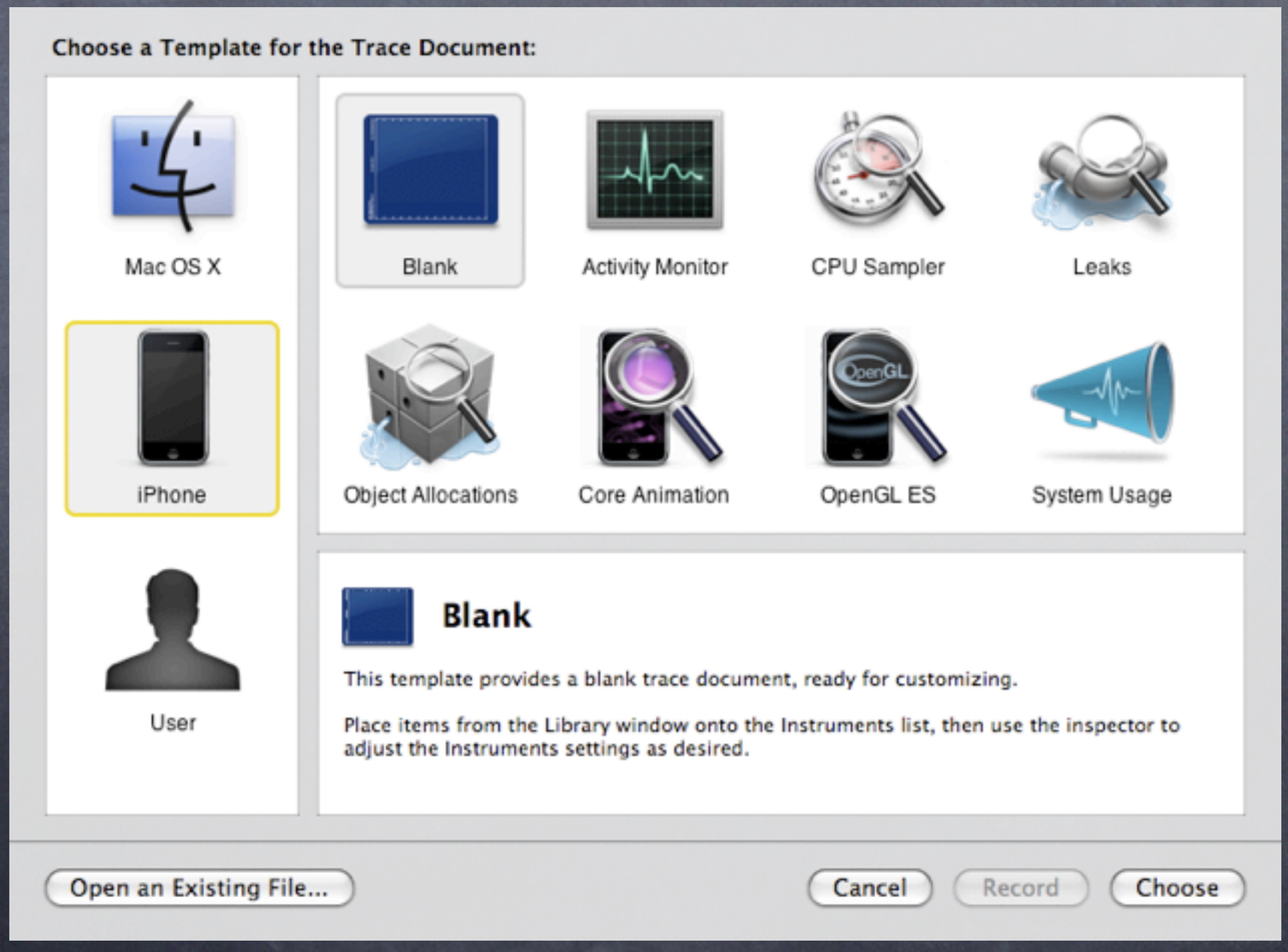
- MallocDebug, OpenGL Driver monitor, OpenGL Profiler, Saturn, heap, leaks, vmmap
- Activity Monitor, BigTop, Quartz Debug, Spin Control, Thread Viewer, fs_usage, sc_usage, top
- gprof, malloc_history, vm_stat, otool, pagestuff, and a few others

Instruments

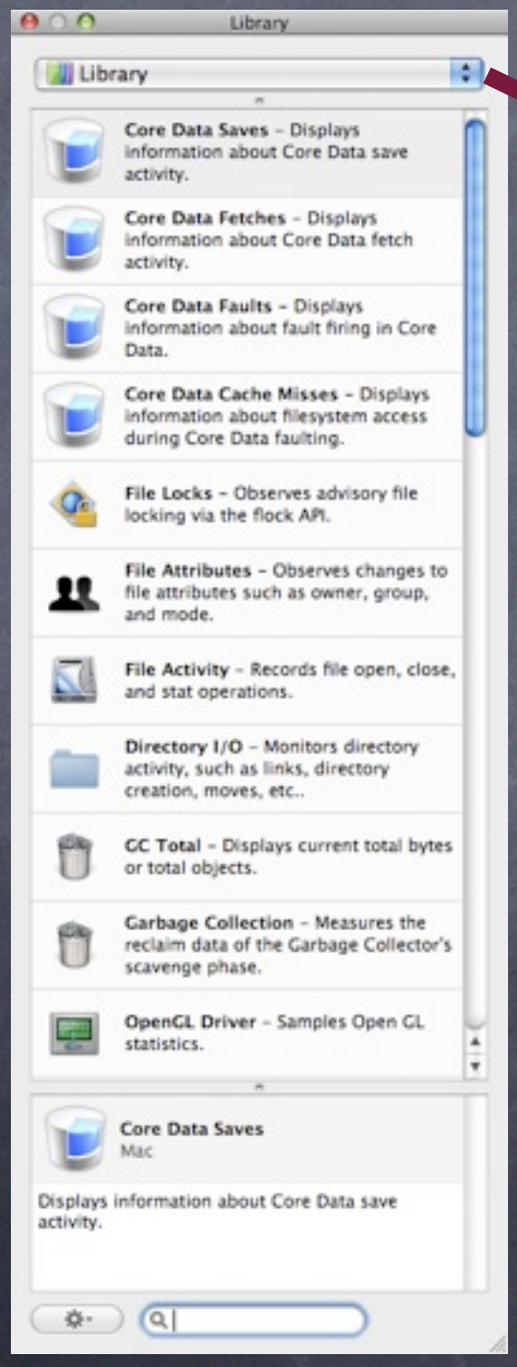
- Xcode 3.0+, Mac OS X 10.5+
- A trace document contains data gathered by instruments
  - instruments are probes that collect data about a specific performance metric
  - usually have less than ten instruments per trace document
- Launch from Xcode > Run > Start With Performance Tool > Leaks

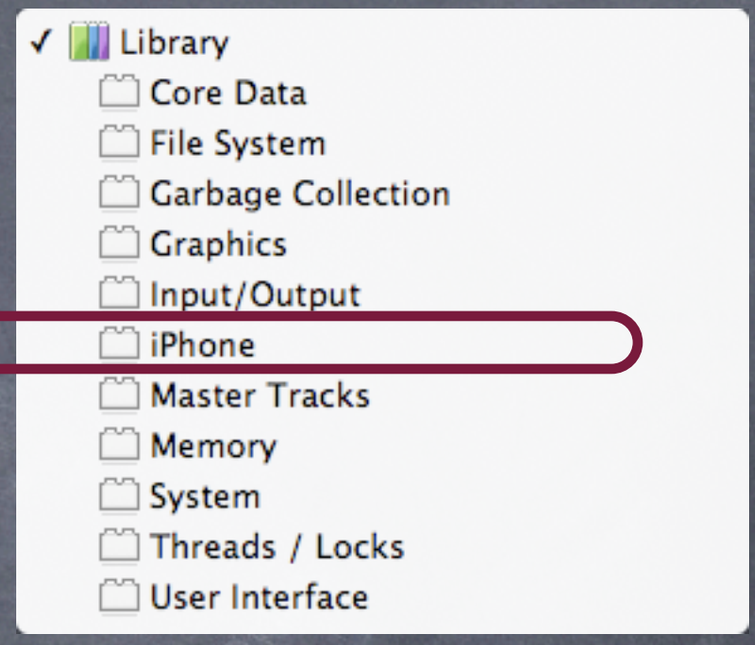
Creating a Custom Trace Document/Template

- Instruments > File > New >



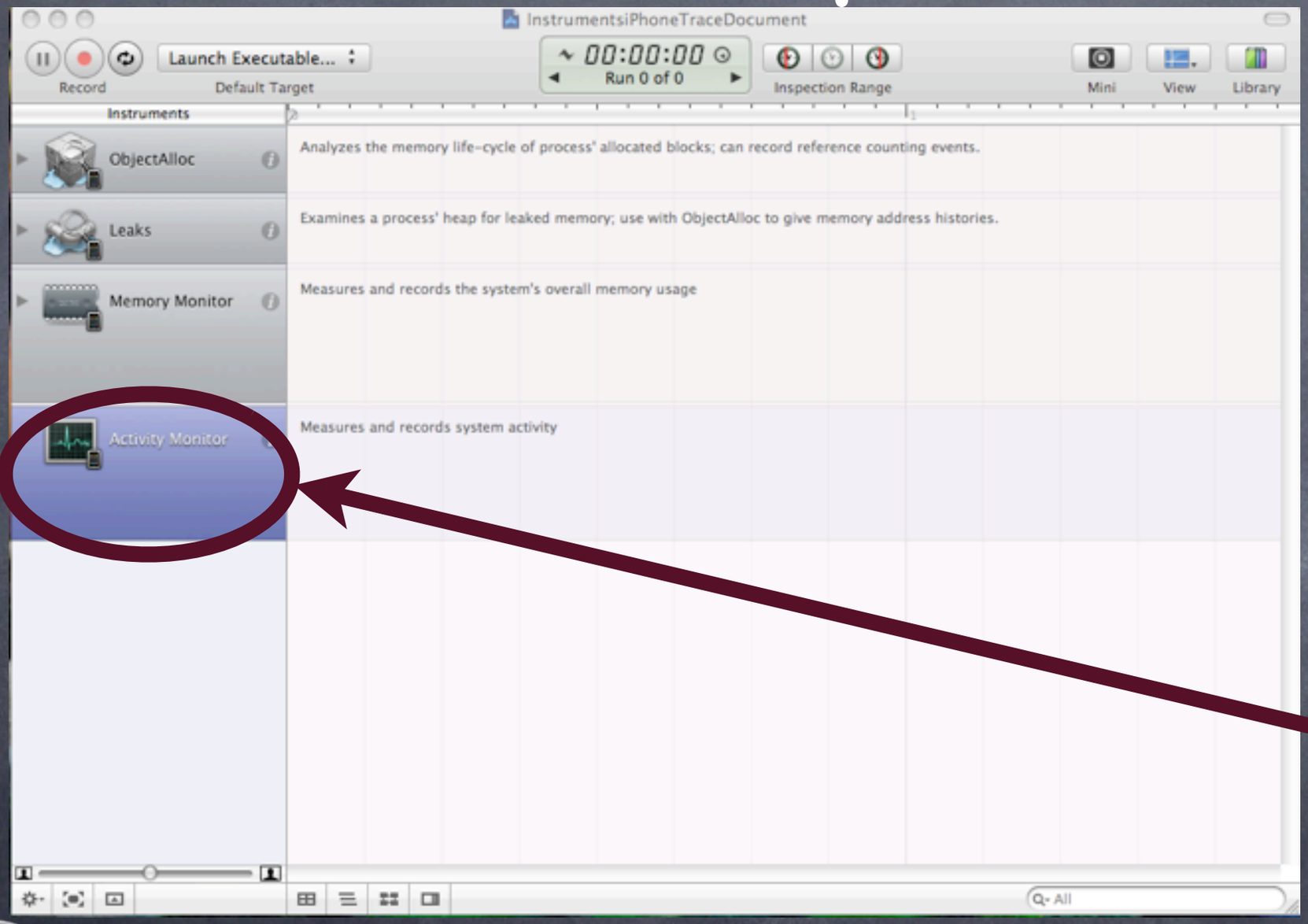
Creating a Custom Trace Document/Template

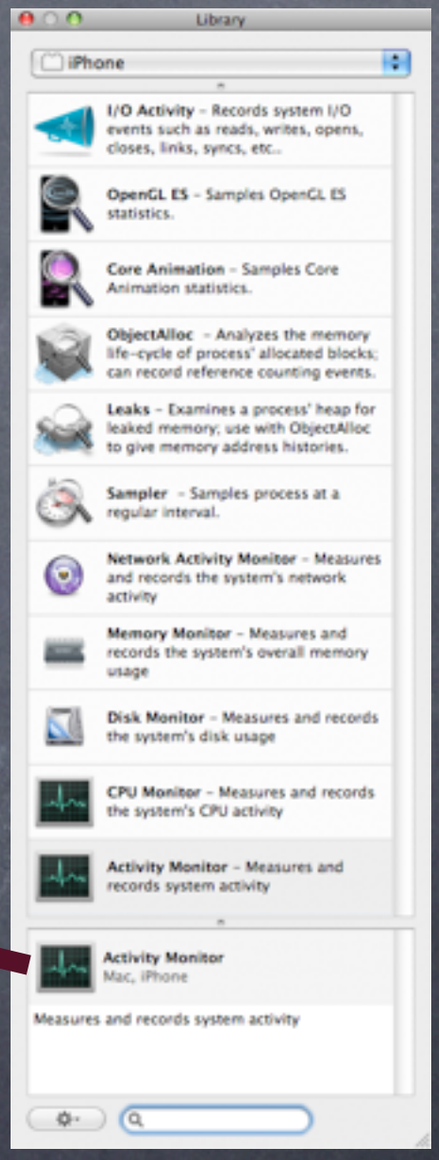




Custom Trace Document/ Template

iPhone instrument library

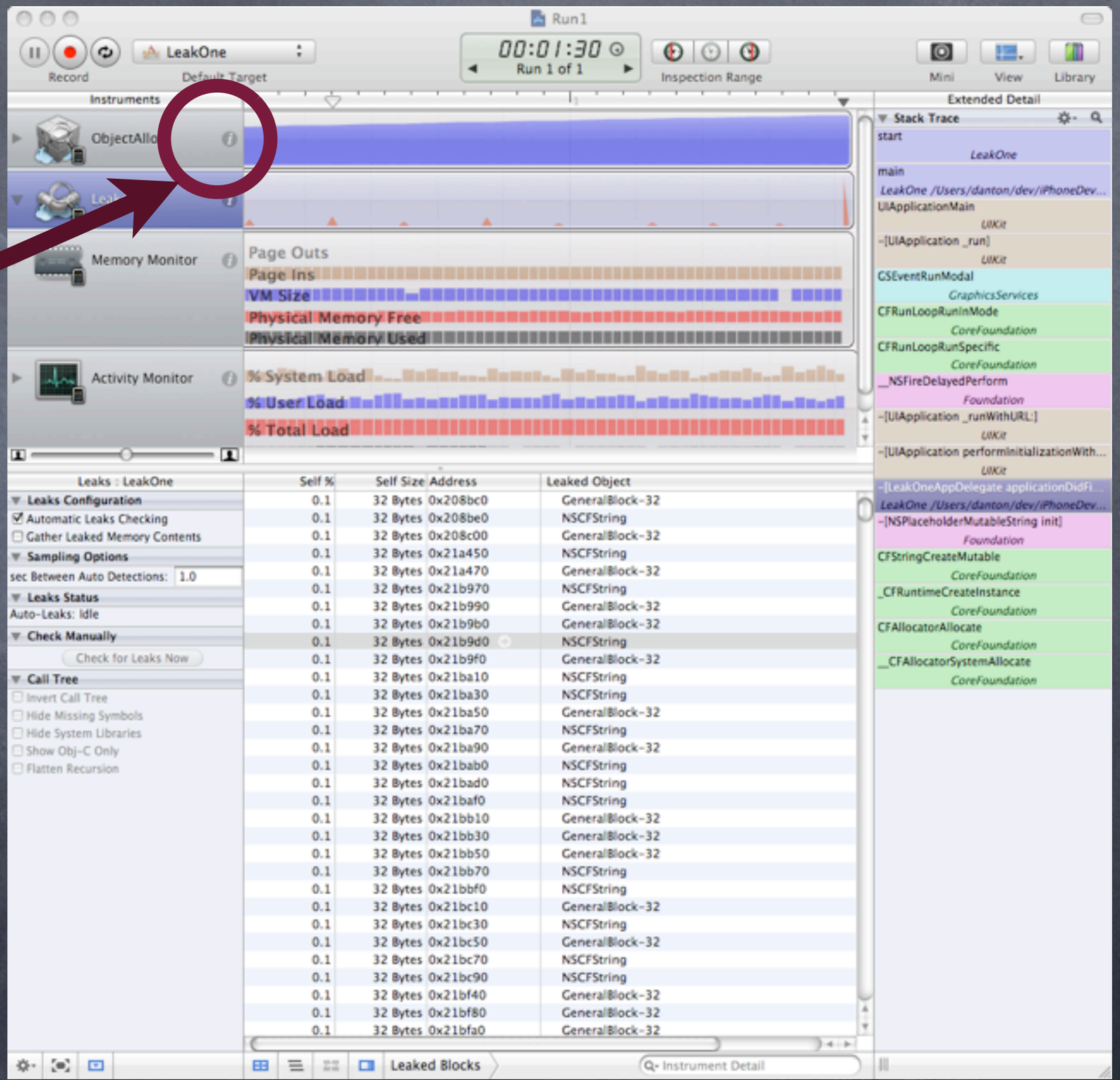




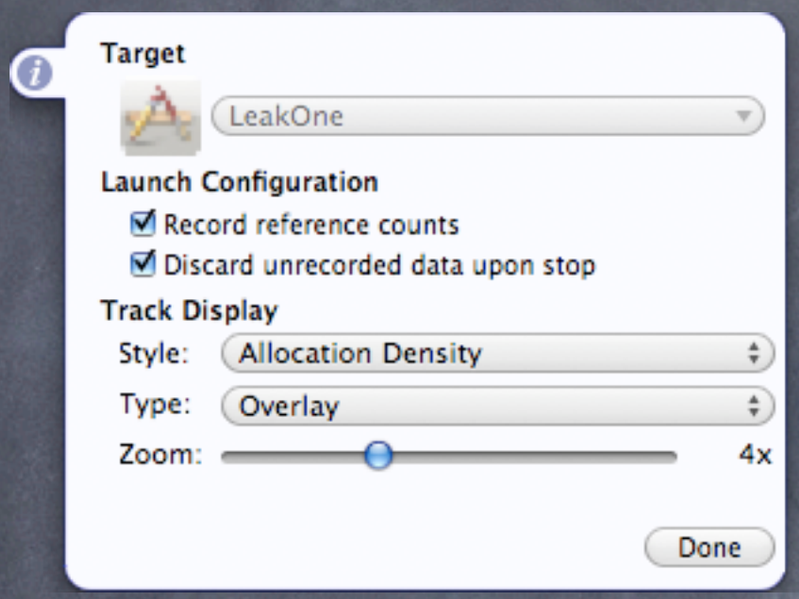
Instruments > File > Save As Template

Custom Trace Document Results

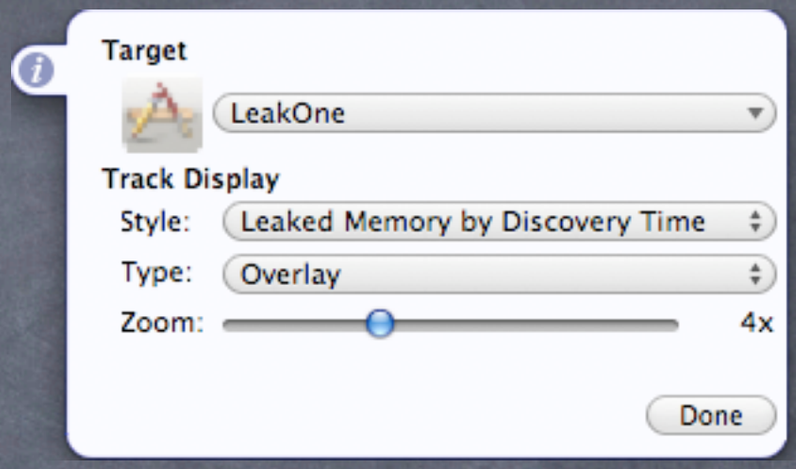
inspector button
for each instrument



Inspector Windows

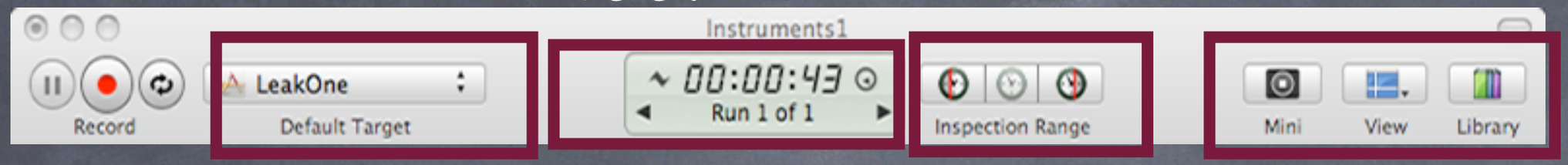


ObjectAlloc inspector window

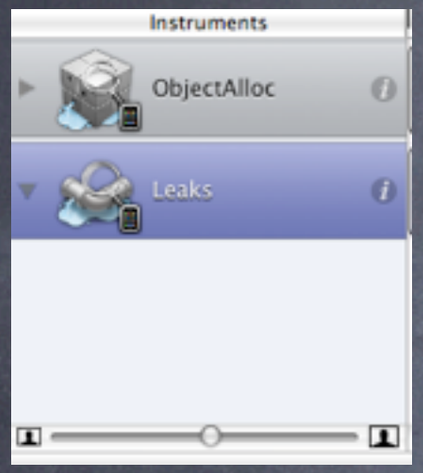


Leaks inspector window

Toolbar

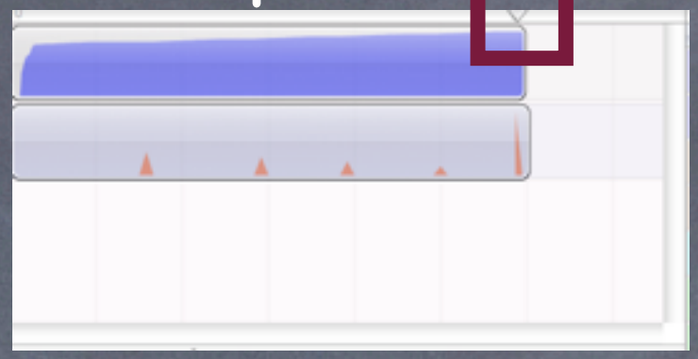


instruments pane



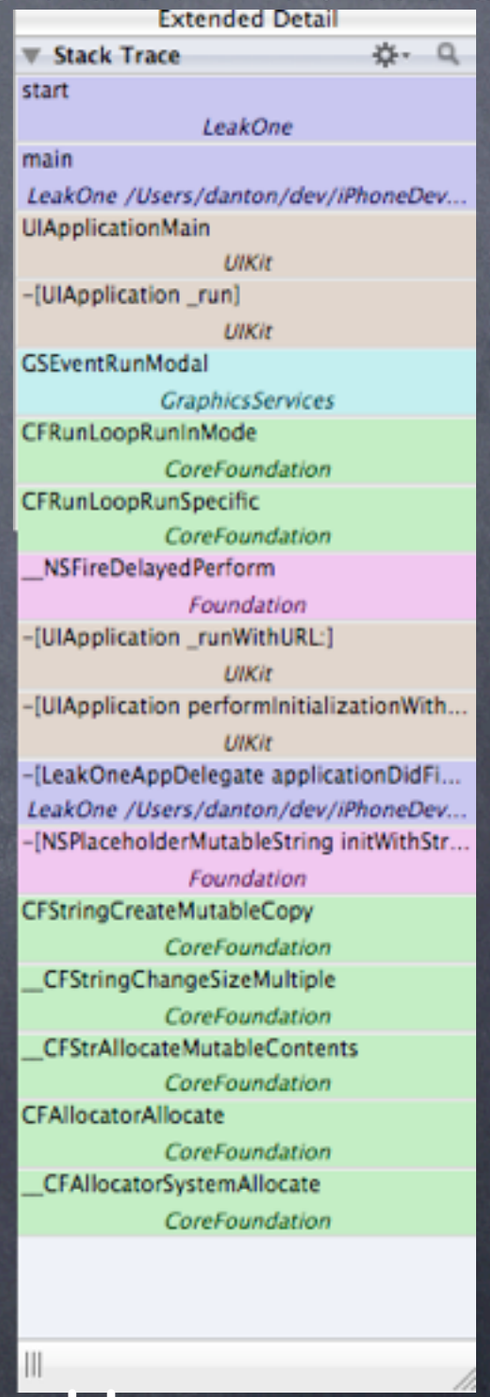
Track pane

playhead

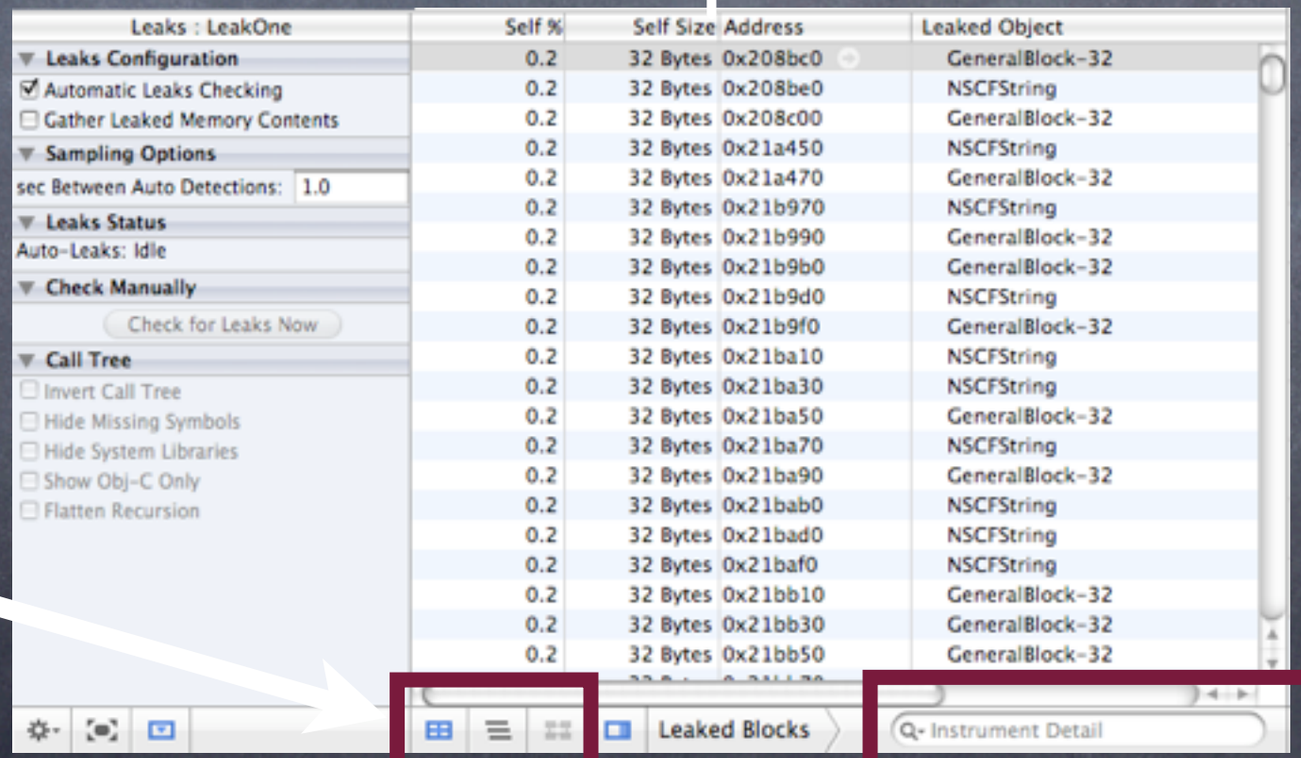


Stack trace

top



Detail pane



View type selector: table, outline, diagram

search

bottom

ObjectAlloc: Table View

InstrumentsDetail

Record | LeakOne (Default Target) | 00:00:43 Run 1 of 1 | Inspection Range | Mini | View | Library

Instruments | Extended Detail

ObjectAlloc

ObjectAlloc : LeakOne	Graph	Category	Net Bytes	# Net	Overall Bytes	# Overall	# Allocations (Net / Overall)
	☑	* All Allocations *	334512	5028	6748432	90621	+++
	☐	GeneralBlock-16	8864	554	464688	30104	
	☐	GeneralBlock-128	-251360	-2096	1369760	13732	
	☐	GeneralBlock-32	-36928	-1685	268480	12087	
	☐	GeneralBlock-144	-284384	-2092	1083184	9513	
	☐	CFString	48720	1462	166528	5085	
	☐	CFDictionary	4352	64	209776	3273	
	☐	CFDictionary (key-store)	8608	49	55248	2261	
	☐	CFDictionary (value-store)	8608	49	55248	2261	
	☐	CFString (store)	34000	1057	330800	2254	
	☐	GeneralBlock-80	688	6	170288	2149	
	☐	GeneralBlock-96	1008	10	203952	2130	
	☐	GeneralBlock-64	-352720	-6303	-150032	1088	
	☐	GeneralBlock-176	-335696	-2099	-149840	1065	
	☐	CFCalendar	32	1	33760	1055	
	☐	GeneralBlock-400	-556512	-2108	286688	1054	
	☐	CFNumber	11792	737	11824	739	
	☐	GeneralBlock-48	-250032	-6264	-248880	90	
	☐	CFSet (key-store)	768	18	1728	68	
	☐	CFSet	1488	22	4368	67	
	☐	CFArray	1072	32	2160	61	

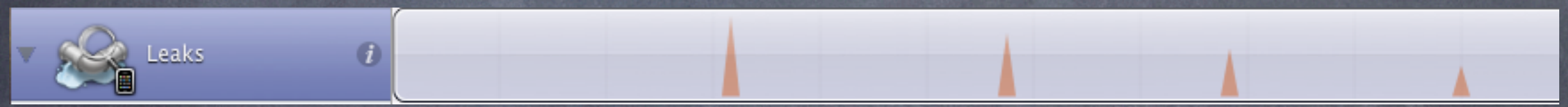
Allocation Lifespan
- All Objects Created
- Created & Still Living

Call Tree
- Separate by Thread
- Invert Call Tree
- Hide Missing Symbols
- Hide System Libraries
- Show Obj-C Only
- Flatten Recursion

Call Tree Constraints

Object Summary | Category

Leaks Display: Leaked Memory By DiscoveryTime



Demo

Clang Static Analyzer

Clang Static Analyzer

- LLVM (Low Level Virtual Machine) is the parent project (http://llvm.org/) partially funded by Apple and announced at WWDC in 2008
  - started by Chris Lattner in 2000
- Clang is a sub-project and is the C/C++/Objective-C front end to gcc
- Clang Static Analyzer is a subproject of Clang

Clang Static Analyzer

About

- Performs a static analysis of your code
- Wants to use the strengths of a compiler to find bugs without test cases
- Not a replacement for testing

Clang Static Analyzer

Goals

- To find "deeper" bugs not merely syntactic ones such as
  - memory leaks
  - buffer overruns
  - logic errors

Clang Static Analyzer

Benefits

- Allow you to find bugs early
- Open Source
- Command line tool that could be integrated into your own build process at some point down the road

Clang Static Analyzer

Drawbacks

- Very early in the development cycle -- the version I used was at 0.167 with builds every night
- There are bugs
- There are false positives or bugs that the Clang Static Analyzer reports as bugs that are not bugs
- Needs more documentation

Clang Static Analyzer

Downloading and Installing

- Download from http://clang.llvm.org/StaticAnalysis.org
- Installation is easy:
  - uncompress into your favorite folder
  - set the environmental variable for the search path for your favorite shell

Clang Static Analyzer

Using

- Easy to use
  - set the default directory to your project
  - enter and execute the only command you need to know:
    - scan-build [command options] xcodebuild
      - scan-build is an executable script
      - xcodebuild part of Xcode

Clang Static Analyzer

Command Line Options

- command options
  - -k keep on going to the build command even if it fails
  - -v, -v -v, -v -v -v verbosity levels
  - -V invokes scan-view
  - -plist outputs the results, if any, to a plist
  - --status-bugs causes scan-build to exit with a 1 if bugs found or zero if none found

Clang Static Analyzer

Report Summary

LeakApp - scan-build results

User:	danton@danton-chins-macbook-pro-101.local
Working Directory:	/docs/360iDevPresentation/LeakApp
Command Line:	xcodebuild
Date:	Tue Mar 3 21:24:35 2009
Version:	checker-0.167 (2009-03-02 16:22:34)

Bug Summary

Results in this analysis run are based on analyzer build **checker-0.167**.

Bug Type	Quantity	Display?
All Bugs	1	☑
Memory (Core Foundation/Objective-C)		
leak	1	☑

Reports

Bug Group	Bug Type ▾	File	Line	Path Length			
Memory (Core Foundation/Objective-C)	leak	LeakAppAppDelegate.m	26	3	View Report	Report Bug	Open File

Clang Static Analyzer

Report Detail

```
1  //
2  //  LeakAppAppDelegate.m
3  //  LeakApp
4  //
5  //  Created by Danton Chin on 3/2/09.
6  //  Copyright http://iphonedeveloperjournal.com 2009. All rights reserved.
7  //
8
9  #import "LeakAppAppDelegate.h"
10
11 @implementation LeakAppAppDelegate
12
13 @synthesize window;
14
15
16 - (void)applicationDidFinishLaunching:(UIApplication *)application {
17
18
19         for (int i=0; i < 100000; i++) {
```

1 Loop condition is true. Entering loop body

```
20
21
22             NSMutableString *test = [[NSMutableString alloc] initWithString:@"Welcome To "];
```

2 Method returns an Objective-C object with a +1 retain count (owning reference)

```
23
24             [test appendString:@"360iDev!"];
25
26             NSLog(@"%@", test);
```

3 Object allocated on line 22 and stored into 'test' is no longer referenced after this point and has a retain count of +1 (object leaked)

```
27
28             // to eliminate leak release it
29             // [test release];
```

Clang Static Analyzer

Bottom Line

- Very promising
- Use it and get involved

Resources

- **iPhone Memory**
  - Memory on the iPhone http://vafer.org/blog/20081128082605
  - What the iPhone specs don't tell you http://furbo.org/2007/08/21/what-the-iphone-specs-dont-tell-you/
- iPhone Application Programming Guide at http://developer.apple.com/iphone, 2008-10-15
- Performance Overview: General, 2008-03-28 http://developer.apple.com/iphone/library/documentation/Performance/Conceptual/PerformanceOverview/PerformanceOverview.pdf

Resources

- **Apple Documentation on memory at** http://developer.apple.com/iphone/
- Memory Usage Performance Guidelines, 2008-07-02
- Memory Management Programming Guide for Cocoa, 2008-11-19
- Memory Management Programming Guide for Core Foundation, 2008-10-15

Resources

- **Memory Management**
  - CS193P - Lecture 11: Performance, Stanford University at http://www.stanford.edu/class/cs193p/cgi-bin/index.php
  - 9 iphone Memory Management Links and Resources by Peter Cooper at http://www.mobileorchard.com/iphone-memory-management
  - Memory Management by John Muchow at http://macdevelopertips.com/objective-c/objective-c-memory-management.html
  - Memory Management with Objective-C/Cocoa/iPhone by Mehmet Akten at http://memo.tv/memory_management_with_objective_c_cocoa_iphone
  - Very Simple Rules for Memory Management in Cocoa by MMalcolm Crawford at http://www.stepwise.com/Articles/Technical/2001-03-11.01.html
  - Learn Objective-C by Scott Stevenson http://cocoadevcentral.com/d/learn_objectivec/

Resources

- Memory Management Videos
  - Memory Management in Objective-C by Scotty at http://www.mac-developer-network.com/category/videotraining/beginner/ (not free)
  - Coding in Objective-C. Episode 2: Memory Management by Bill Dudney at http://www.pragprog.com/screencasts/v-bdobjc/coding-in-objective-c-2-0/ ($5)

Resources

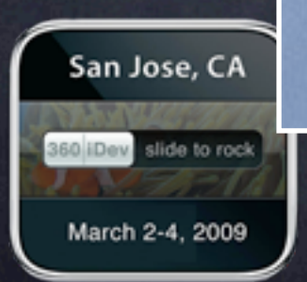
- Instruments
- Instruments User Guide. Performance Tuning: Instruments, http://developer.apple.com/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/index.html

Resources

- Shark User Guide. Performance Tuning at http://developer.apple.com/documentation/DeveloperTools/Conceptual/SharkUserGuide/index.html

Resources

- **LLVM and Clang**
  - LLVM home page at http://llvm.org/
  - Clang Static Analyzer home page at http://clang.llvm.org/StaticAnalysis.html
  - setting the path varible for tsch and bash at http://iphonedevelopment.blogspot.com/2009/02/clang-static-analyzer.html
  - LLVM Developer's 2008 Meeting at http://llvm.org/devmtg/2008-08/
    - Finding Bugs With the Clang Static Analyzer by Ted Kremenek, Apple Inc. Video and Slides (part of the link above)



Q & A