Advanced Memory Analysis with Instruments

Daniel Delwood
Performance Tools Engineer

Memory Analysis
What’s the issue?

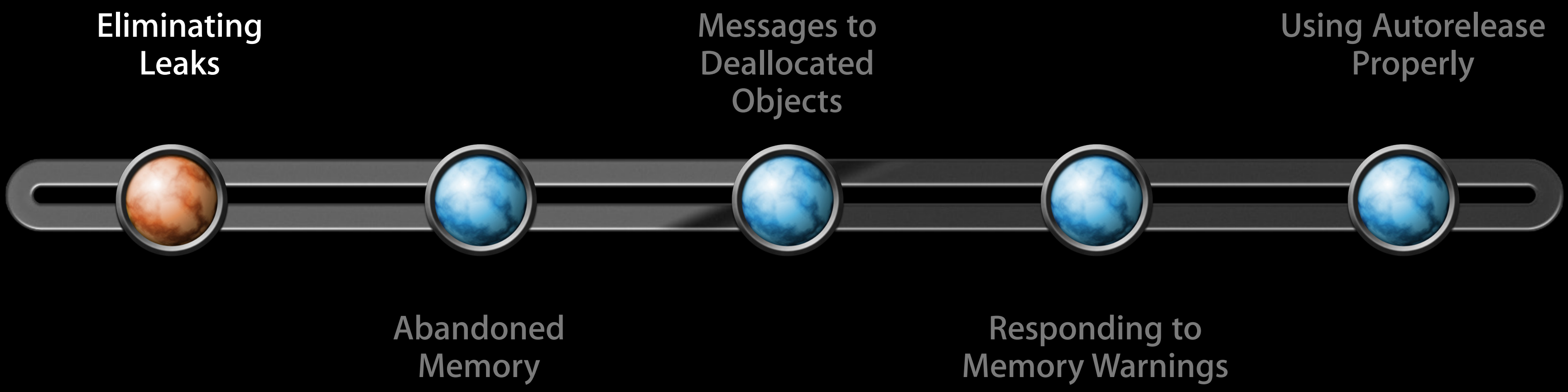
- Memory is critical to performance
- Limited resource
    - Especially on iPhone OS

Memory Analysis

When to use Instruments

- Understand your app’s memory usage
- Reduce wasted memory
- Diagnose memory related crashes
- Be proactive about usage
    - Avoid termination
    - Better multitasking citizen

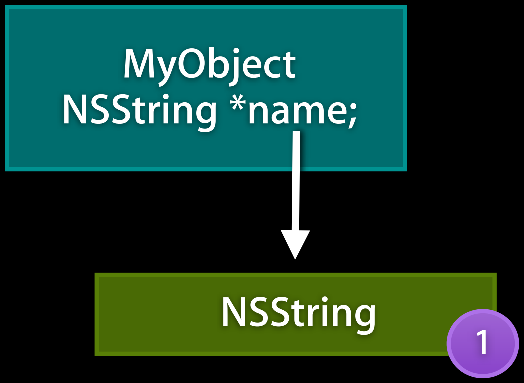
Memory Analysis


Eliminating Leaks
Abandoned Memory
Messages to Deallocated Objects
Responding to Memory Warnings
Using Autorelease Properly

Eliminating Leaks

What constitutes a 'leak'?

- Allocated memory that can no longer be reached
- No more pointers to it

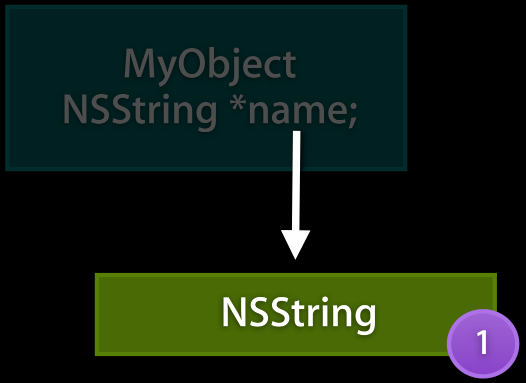


```
- (id)init {
    if (self = [super init]) {
       name = [[NSString alloc] initWithFormat:...];
    }
    return self;
}
```

Eliminating Leaks

What constitutes a ‘leak’?

- Allocated memory that can no longer be reached
- No more pointers to it



```
- (void)dealloc {

   [super dealloc];
}
```

missing: [name release];

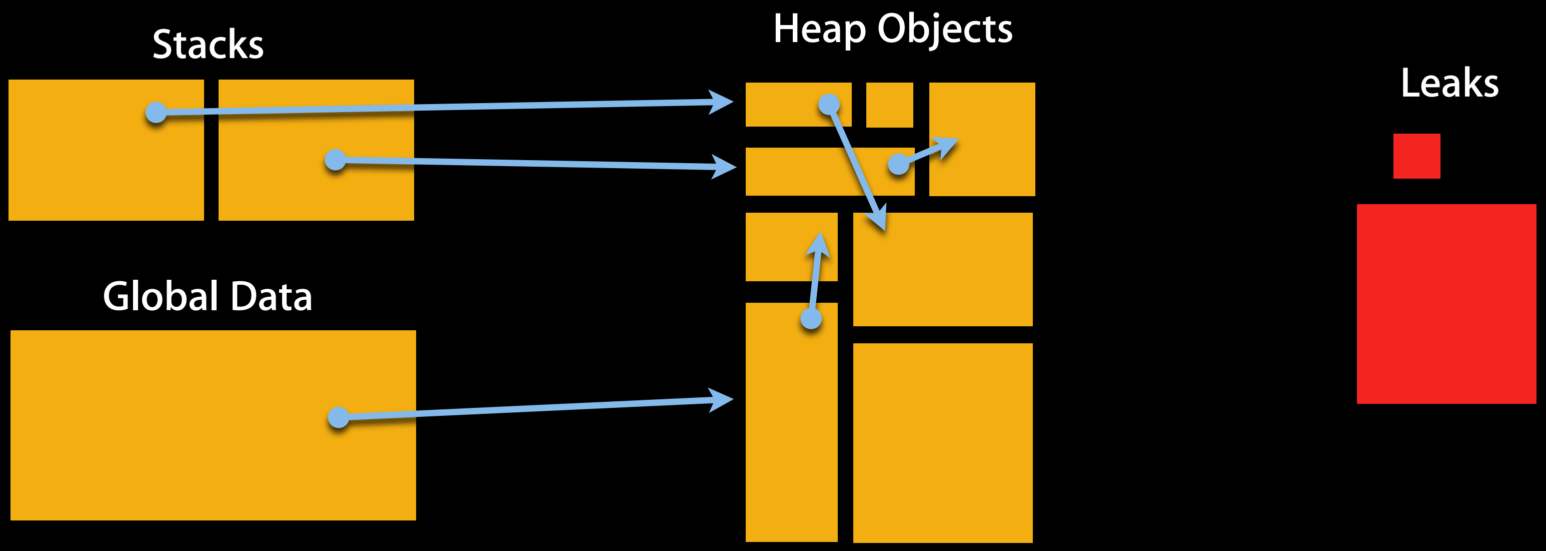
Eliminating Leaks

How do you find them?

Eliminating Leaks

Leaks instrument

- Identifies leaked memory
- Conservative memory analysis
- Misses some, but reliable



Eliminating Leaks

Found the leaked object! Now what?

Eliminating Leaks

Allocations instrument

- Tracks all ‘malloc’ heap allocations
- C, Objective-C, C++
- Malloc/Free/Retain/Release/Autorelease
- Type statistics
- Call Trees
- Incurs overhead

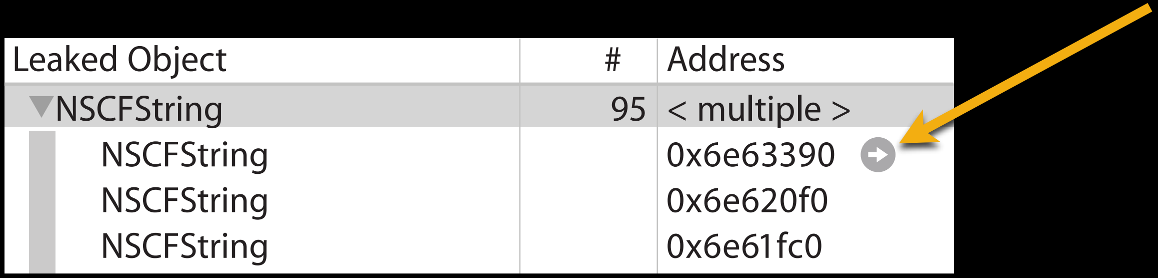
Finding/Fixing Leaks Demo

Daniel Delwood
Performance Tools Engineer

Eliminating Leaks

Can’t just look at the allocation point!

- Allocation backtrace isn’t the whole story
- Framework-created objects can be leaked by app code
- Focus on a single instance to investigate



Leaked Object	#	Address
▼NSCFString	95	< multiple >
NSCFString		0x6e63390
NSCFString		0x6e620f0
NSCFString		0x6e61fc0

Memory Management Programming Guide for Cocoa

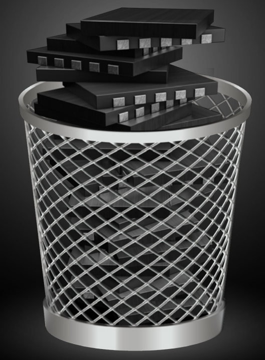
http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/MemoryMgmt/

Memory Analysis

Eliminating Leaks
Messages to Deallocated Objects
Using Autorelease Properly
Abandoned Memory
Responding to Memory Warnings

Abandoned Memory

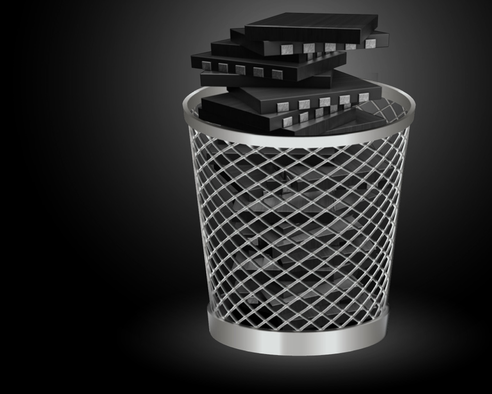
What is it?



- Leaked memory
  - “Allocated memory that can no longer be reached”
  - Inaccessible—no more pointers to it
- Abandoned Memory
  - “Accessible allocated memory that is never used again”
  - Wasted or forgotten memory
  - Occurs also when garbage collected

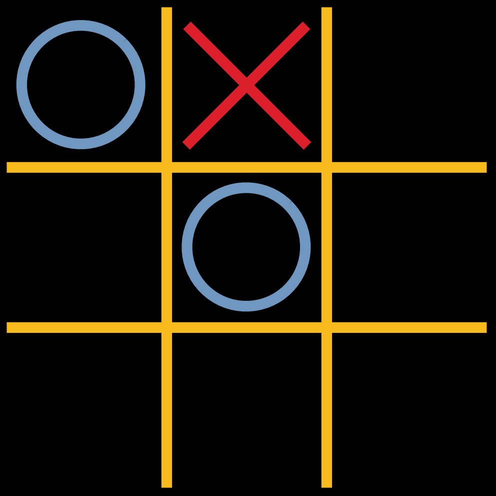
Abandoned Memory

Examples



- Extraneous information

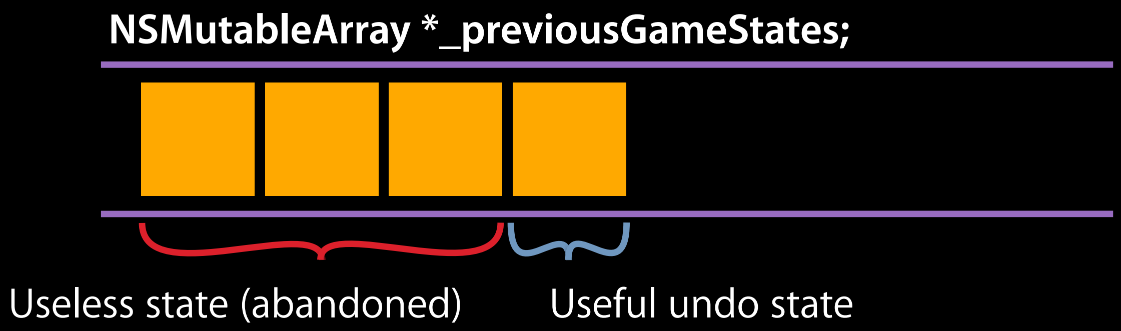
```
- (void)updateBoardWithMove:(TicTacToeMove*)move {
    [_previousGameStates addObject:[self currentGameState]];
    ...
}
```

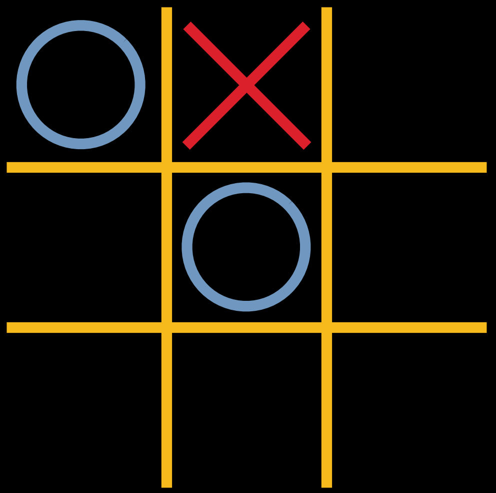


Abandoned Memory

Examples

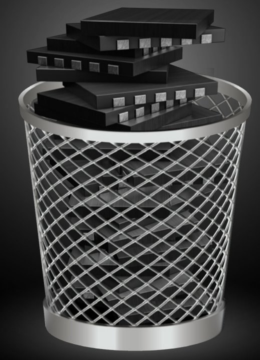
- Extraneous information


NSMutableArray *_previousGameStates;
Useless state (abandoned)
Useful undo state



Abandoned Memory

Examples



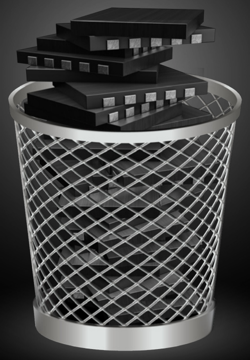
- Faulty cache

```
- (NSImage*)_imageInDirectory:(NSURL*)url index:(NSUInteger)index {
    NSImage *image = [_imageCache objectForKey:[NSString stringWithFormat:@"%@, %lu", url, index]];
    if (!image) {
        NSURL *imageURL = [[[NSFileManager defaultManager] contentsOfDirectoryAtURL:url ...
        image = [[[NSImage alloc] initWithContentsOfURL:imageURL] autorelease];
        [_imageCache setObject:image forKey:[NSString stringWithFormat:@"%d, %lu", url, index]];
    }
    return image;
}
```

Abandoned Memory

Examples

- Faulty cache



```
- (NSImage*)_imageInDirectory:(NSURL*)url index:(NSUInteger)index {
    NSImage *image = [_imageCache objectForKey:[NSString stringWithFormat:@"%@, %lu", url, index]];
    if (!image) {
        NSURL *imageURL = [[[NSFileManager defaultManager] contentsOfDirectoryAtURL:url ...
        image = [[[NSImage alloc] initWithContentsOfURL:imageURL] autorelease];
        [_imageCache setObject:image forKey:[NSString stringWithFormat:@"%d, %lu", url, index]];
    }
    return image;
}
```

Abandoned Memory

Examples

- Faulty cache

```
- (NSImage*)_imageInDirectory:(NSURL*)url index:(NSUInteger)index {
    NSImage *image = [_imageCache objectForKey:[NSString stringWithFormat:@"%@, %lu", url, index]];
    if (!image) {
        NSURL *imageURL = [[[NSFileManager defaultManager] contentsOfDirectoryAtURL:url ...
        image = [[[NSImage alloc] initWithContentsOfURL:imageURL] autorelease];
        [_imageCache setObject:image forKey:[NSString stringWithFormat:@"%d, %lu", url, index]];
    }
    return image;
}
```

@”file://localhost/Library/Desktop%20Pictures/Abstract/, 2” ≠ @”1484592, 2”

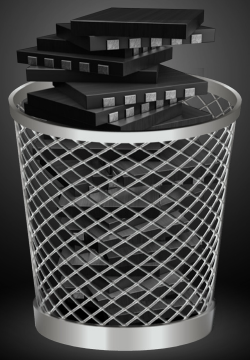
Abandoned Memory

How to detect it

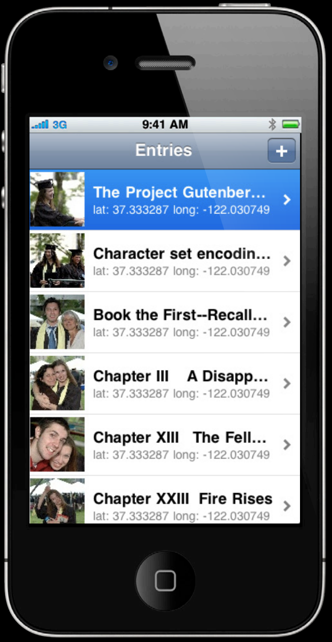
- Basic principle
  - "Memory should not grow without bound when repeating an operation that returns the user to the same state"
- For example:
  - Pushing and popping a view controller
  - Opening and closing a window
  - Changing app preferences back and forth

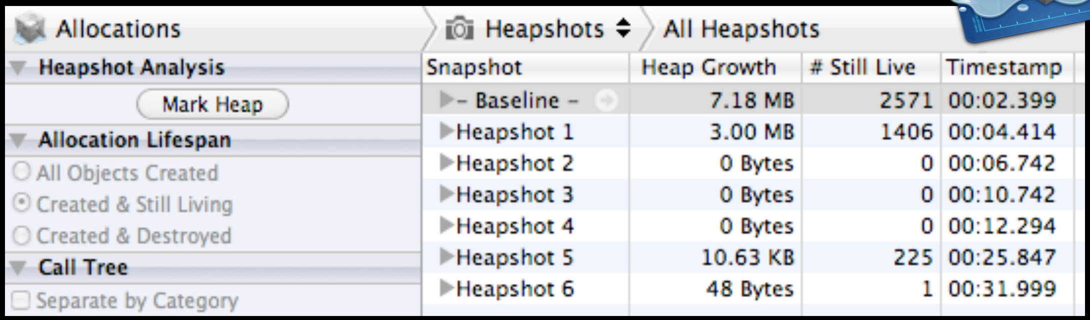
Abandoned Memory

How to detect it



1. Get your application into a starting state
2. Perform an action and return to that state
3. Take a snapshot of the heap


3G
9:41 AM
Entries
+
The Project Gutenber...
lat: 37.333287 long: -122.030749
Character set encodin...
lat: 37.333287 long: -122.030749
Book the First--Recall...
lat: 37.333287 long: -122.030749
Chapter III A Disapp...
lat: 37.333287 long: -122.030749
Chapter XIII The Fell...
lat: 37.333287 long: -122.030749
Chapter XXIII Fire Rises
lat: 37.333287 long: -122.030749



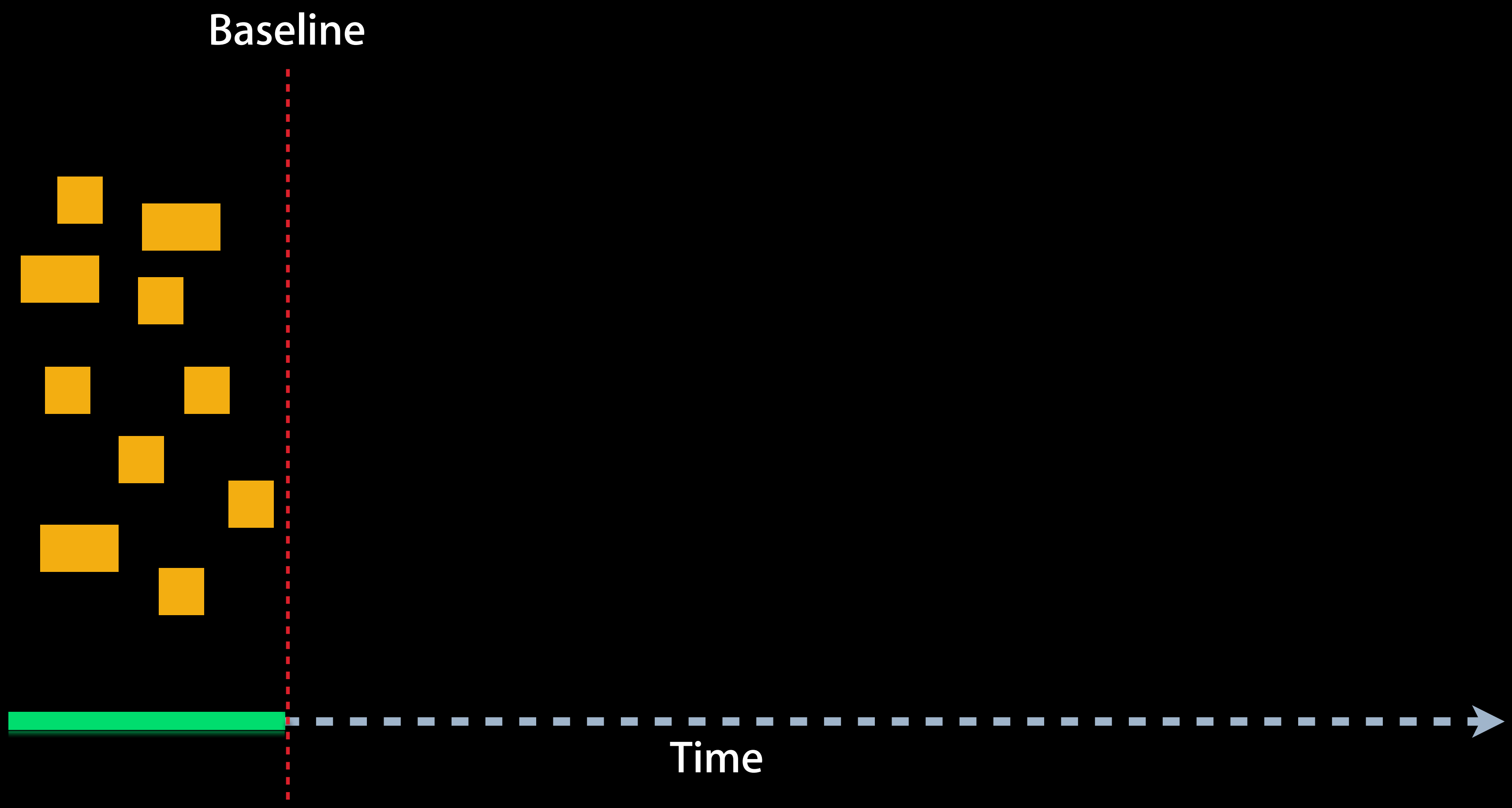
Snapshot	Heap Growth	# Still Live	Timestamp
– Baseline –	7.18 MB	2571	00:02.399
Heapshot 1	3.00 MB	1406	00:04.414
Heapshot 2	0 Bytes	0	00:06.742
Heapshot 3	0 Bytes	0	00:10.742
Heapshot 4	0 Bytes	0	00:12.294
Heapshot 5	10.63 KB	225	00:25.847
Heapshot 6	48 Bytes	1	00:31.999

Abandoned Memory Demo

Daniel Delwood
Performance Tools Engineer

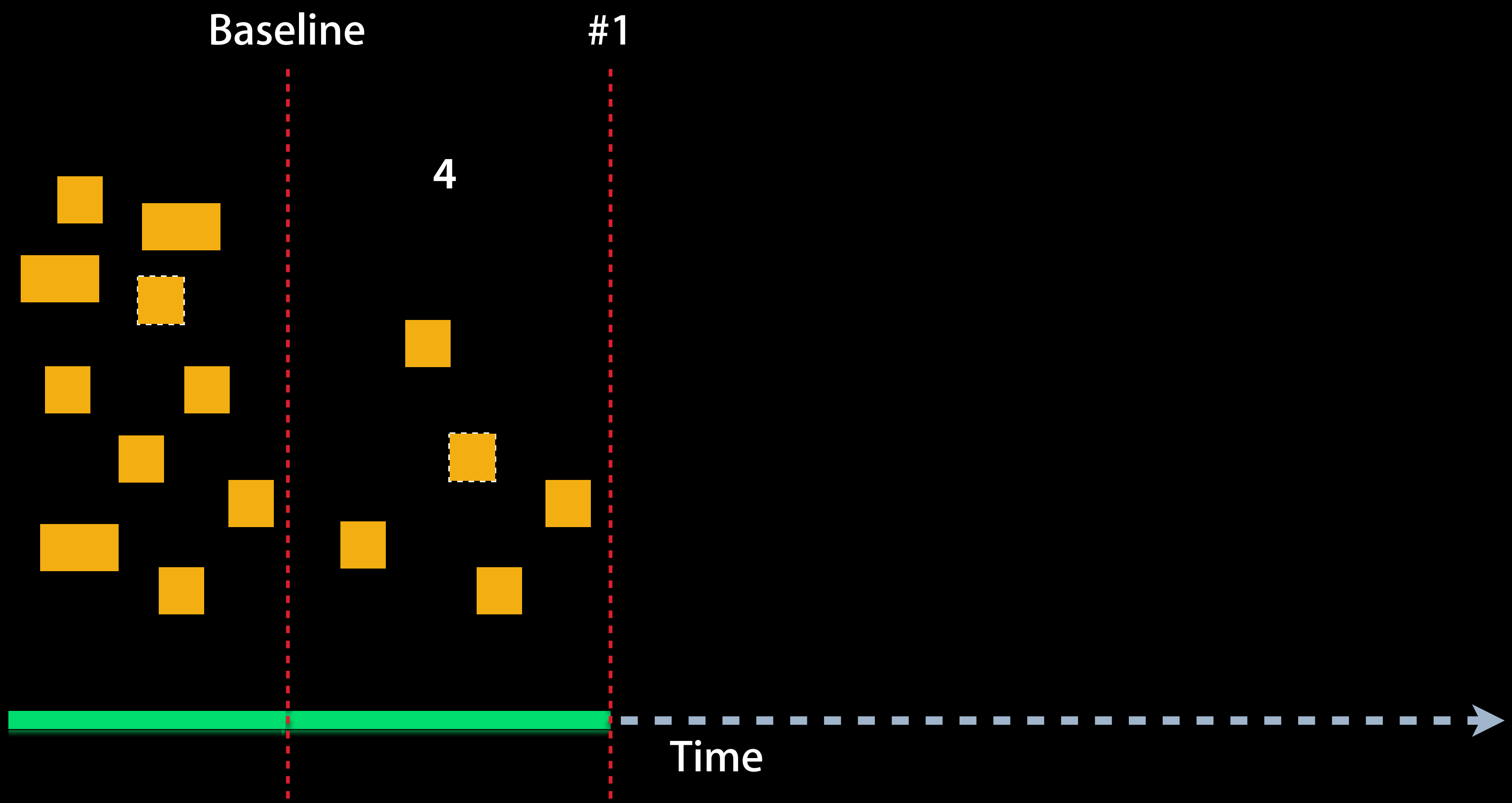
Abandoned Memory

Heapshot details



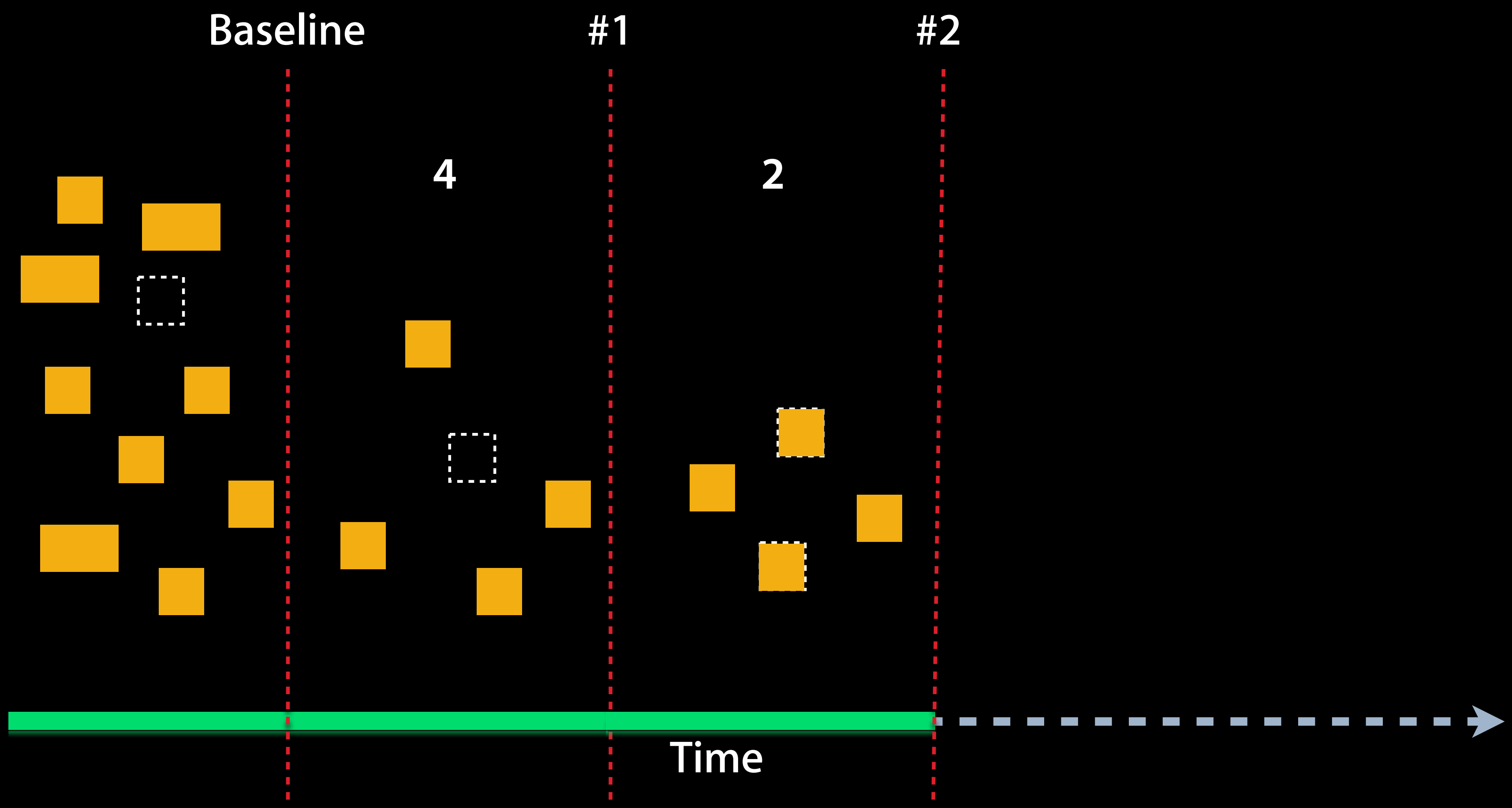
Abandoned Memory

Heapshot details



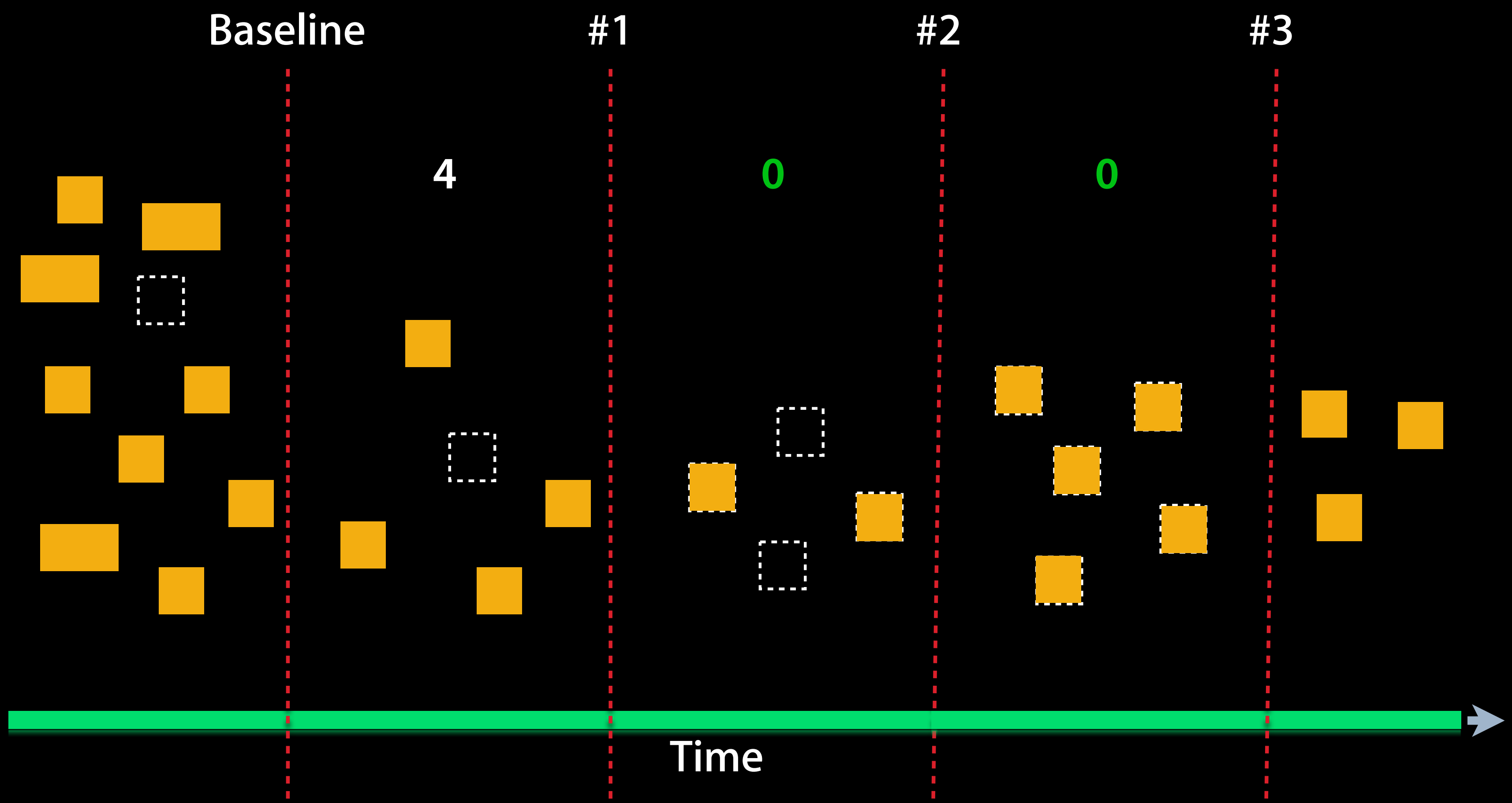
Abandoned Memory

Heapshot details



Abandoned Memory

Heapshot details



Memory Analysis

Eliminating Leaks
Messages to Deallocated Objects
Using Autorelease Properly
Abandoned Memory
Responding to Memory Warnings

Victor Hernandez
Performance Tools Engineer

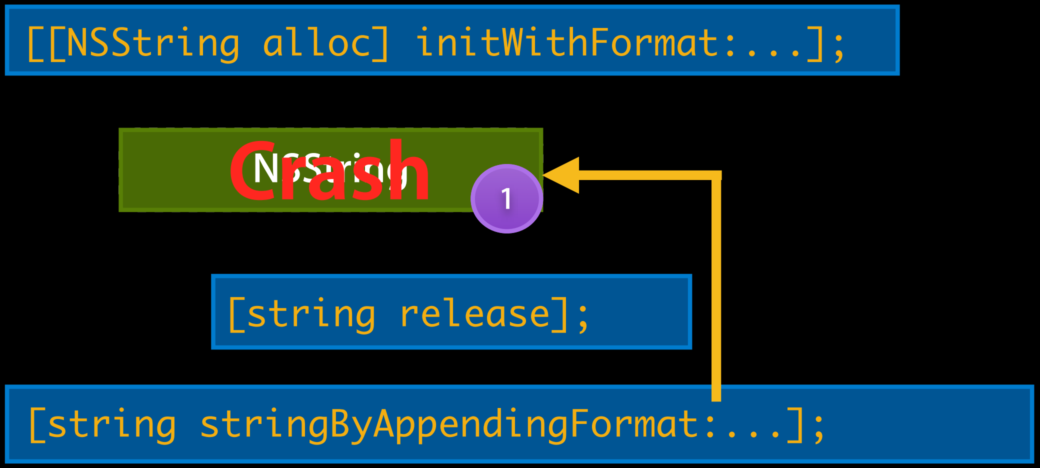
[redacted]

```
Exception Type:  EXC_BAD_ACCESS (SIGBUS)
Exception Codes: KERN_PROTECTION_FAILURE at 0x00000010
Crashed Thread:  0

Thread 0 Crashed:
0   libobjc.A.dylib     0x0000286c objc_msgSend + 16
1   Foundation          0x0001219c  -[NSString stringByAppendingFormat:] + 84
2   Reader              0x000031d4  -[RootViewController tableView:cellForRowAtIndexPath:] + 32
3   UIKit               0x0007e18c  -[UITableView _createPreparedCellForGlobalRow:withIndexPath:] + 492
4   UIKit               0x0007ded8  -[UITableView(UITableViewInternal) _createPreparedCellForGlobalRow:] + 28
5   UIKit               0x000530e2  -[UITableView(_UITableViewPrivate) _updateVisibleCellsNow:] + 930
6   UIKit               0x000514da  -[UITableView layoutSubviews] + 134
7   UIKit               0x0000f874  -[UIView(CALayerDelegate) _layoutSublayersOfLayer:] + 20
8   CoreFoundation      0x000277f8  -[NSObject(NSObject) performSelector:withObject:] + 16
```

Messages to Deallocated Objects

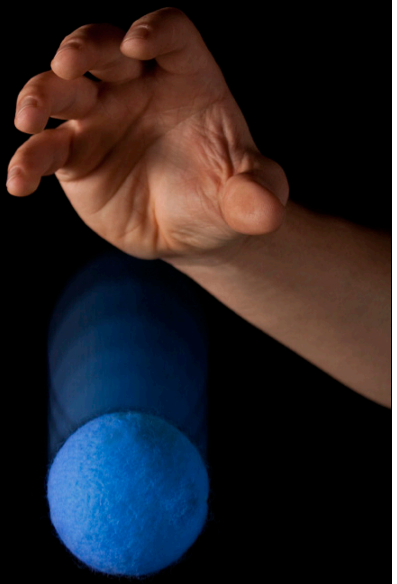
Over-released objects



```
[[NSString alloc] initWithFormat:...];
```

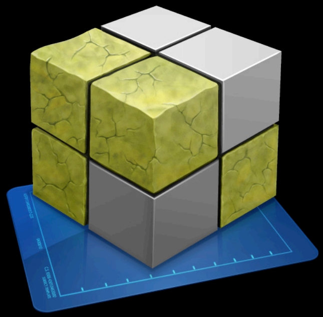
```
[string release];
```

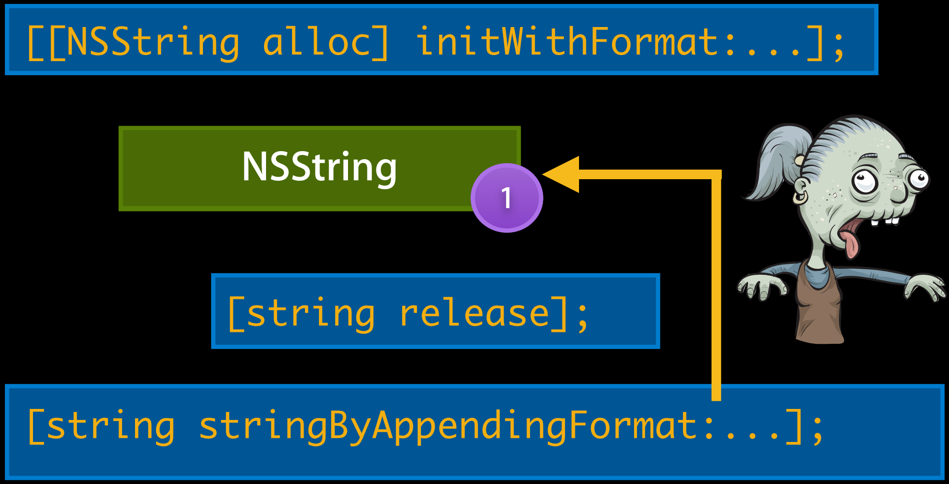
```
[string stringByAppendingFormat:...];
```



Messages to Deallocated Objects

NSObject ➜ NSZombie





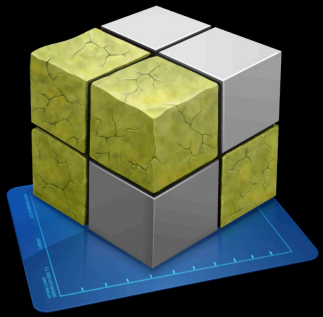
```
[[NSString alloc] initWithFormat:...];
```

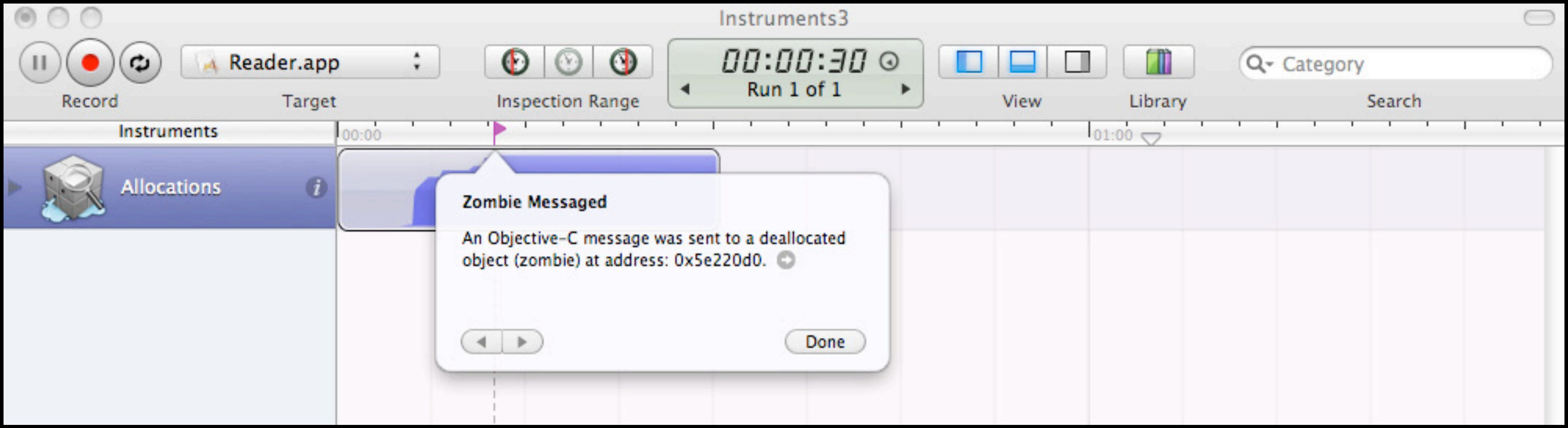
```
[string release];
```

```
[string stringByAppendingFormat:...];
```

Messages to Deallocated Objects

Detect them with Zombies template

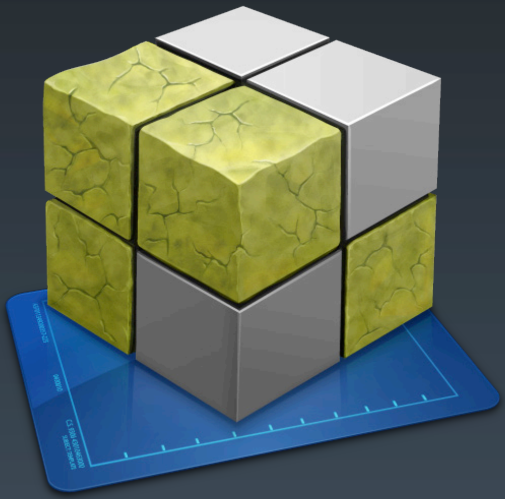




Zombie Messaged

An Objective-C message was sent to a deallocated object (zombie) at address: 0x5e220d0.

Zombies Demo



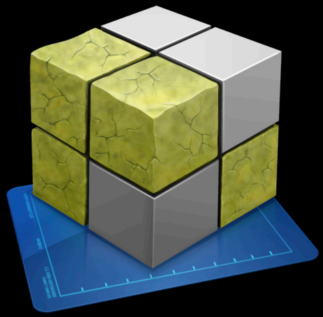
Victor Hernandez
Performance Tools Engineer

"A received object is normally guaranteed to remain valid within the method it was received in (exceptions include multithreaded applications and some Distributed Objects situations, although you must also take care if you modify an object from which you received another object)."

Memory Management Programming Guide for Cocoa
http://developer.apple.com/iphone/library/documentation/Cocoa/Conceptual/MemoryMgmt/

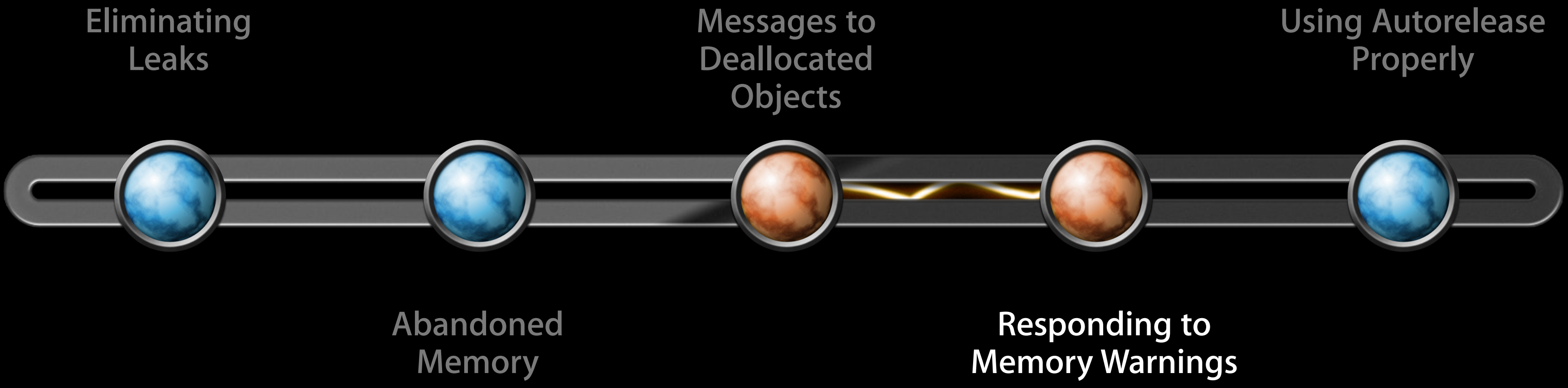
Messages to Deallocated Objects

Zombies template



- Causes memory growth—use iPhone/iPad Simulator
- Not suitable to be used with Leaks
- Last objc message is not always to blame

Memory Analysis


Eliminating Leaks
Abandoned Memory
Messages to Deallocated Objects
Responding to Memory Warnings
Using Autorelease Properly

Responding to Memory Warnings

A fact of life on iPhoneOS

- When system needs memory, notifications go out
- Multitasking increases memory demands
- Respond or be terminated

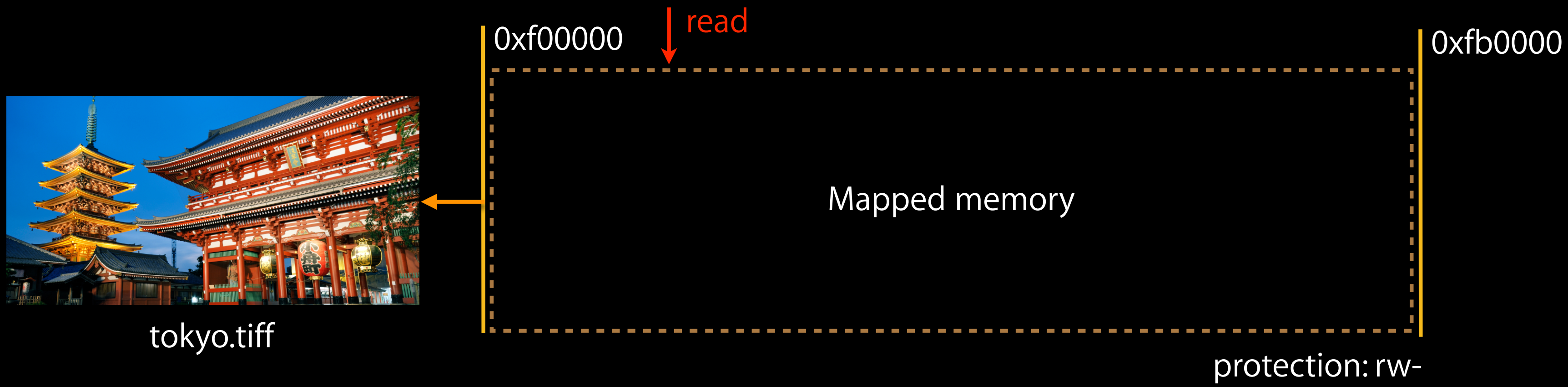
```
- (void)didReceiveMemoryWarning {
    ...
}
```

```
- (void)applicationDidReceiveMemoryWarning:
      (UIApplication *)app {
    ...
}
```

Responding to Memory Warnings

Deciding what memory to free

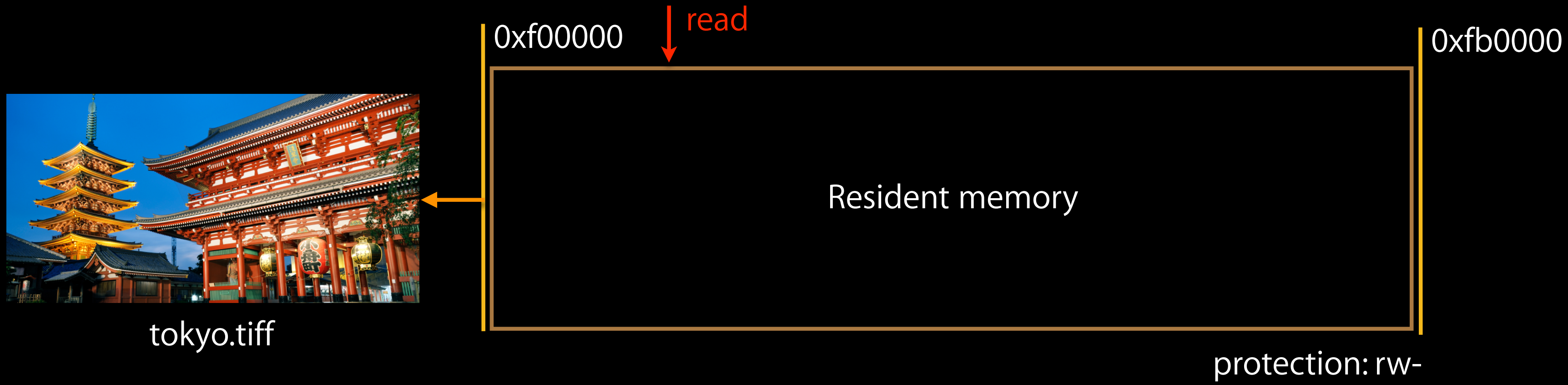
- Based on resident, dirty pages
- Instruments helps you identify that memory



Responding to Memory Warnings

Deciding what memory to free

- Based on resident, dirty pages
- Instruments helps you identify that memory

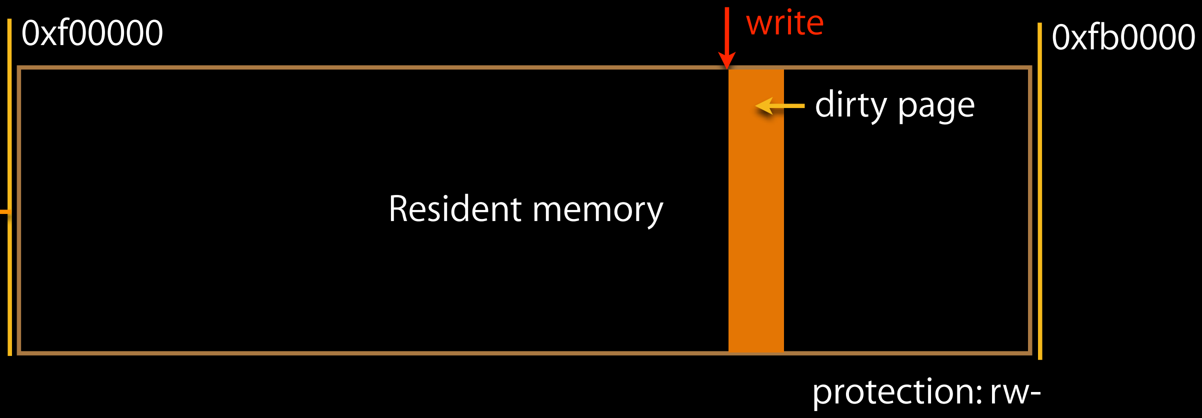


Responding to Memory Warnings

Deciding what memory to free

- Based on resident, dirty pages
- Instruments helps you identify that memory

tokyo.tiff



Responding to Memory Warnings

Watching your Virtual Memory

Responding to Memory Warnings

VM Tracker instrument

- Takes snapshots of virtual memory
- Similar to `vmmap`
- More granular than Activity Monitor instrument
- For each region and each page:
    - Categorizes by type
    - Identifies protection
    - Reports resident, dirty state

Responding to Memory Warnings

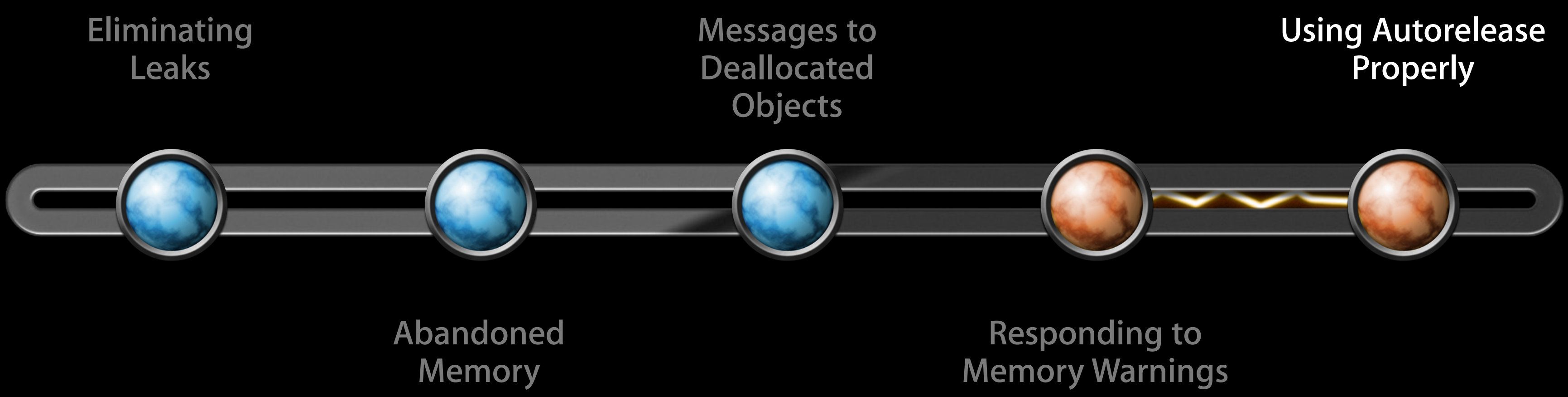
Checking your efforts

- Proactively check your work
- Use simulator to manually trigger a memory warning
- Use VM Tracker to see your app respond

VM Tracker Demo

Victor Hernandez
Performance Tools Engineer

Memory Analysis

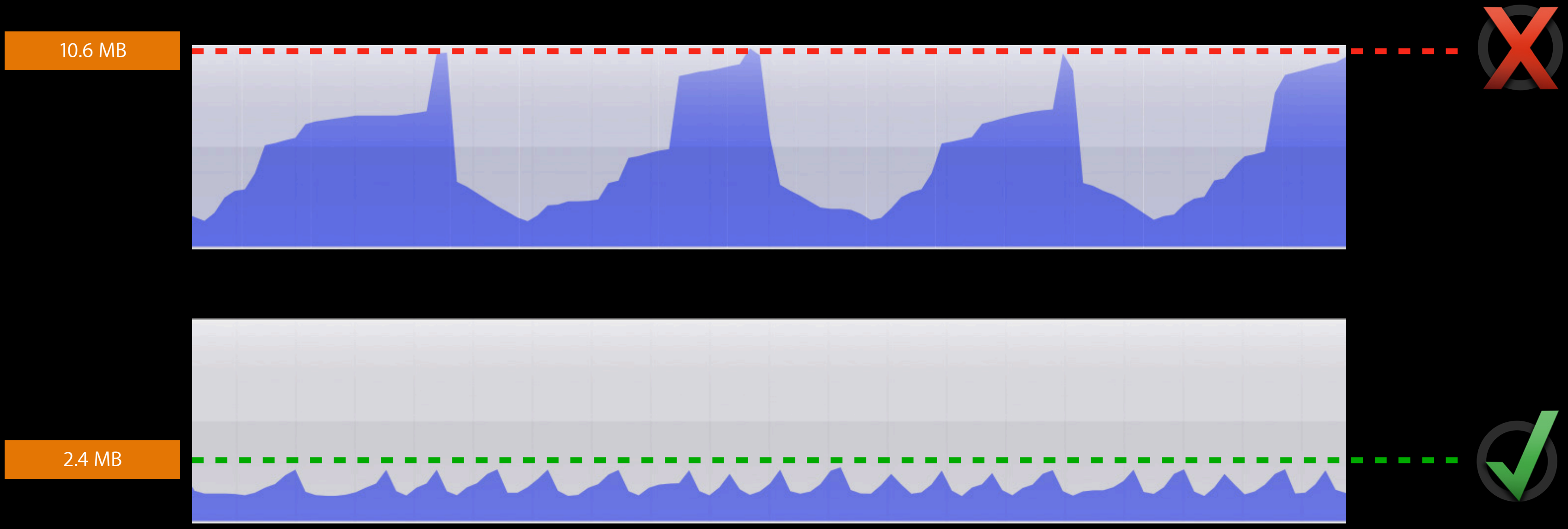

Eliminating Leaks
Abandoned Memory
Messages to Deallocated Objects
Responding to Memory Warnings
Using Autorelease Properly

Daniel Delwood
Performance Tools Engineer

Using Autorelease Properly

Memory high-water mark matters

- Use Allocations and VM Tracker graphs to identify spikes



Using Autorelease Properly

Memory high-water mark matters

- Use Allocations and VM Tracker graphs to identify spikes
- Be careful of autoreleased allocations in loops

could return a new autoreleased object every time

returns an autoreleased immutable copy

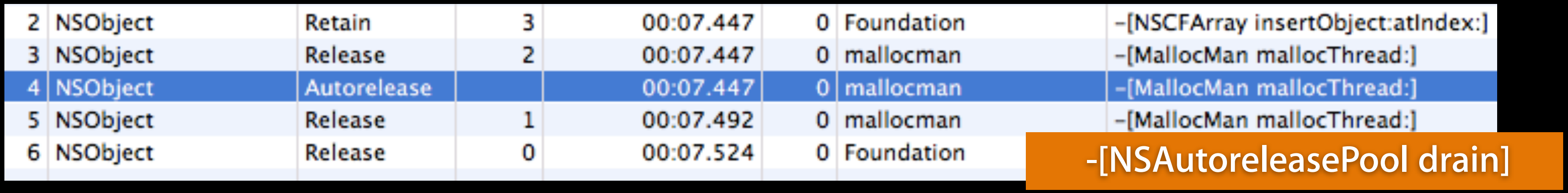
```
for (i = 0; i < database.lastEmployee.number;
    Person *employee = [database employeeWithNumber:i];
    if ([[tableView selectedRowIndexes] containsIndex:employee.groupID]) {
        [[groupListsByID objectForKey:[NSNumber numberWithInt:groupID]] addObject:employee];
    }
}
```

returns autoreleased NSNumber

Using Autorelease Properly

Memory high-water mark matters

- Use Allocations and VM Tracker graphs to identify spikes
- Be careful of autoreleased allocations in loops
- No magic! `–autorelease` is just a delayed `–release`



2	NSObject	Retain	3	00:07.447	0	Foundation	–[NSCFArray insertObject:atIndex:]
3	NSObject	Release	2	00:07.447	0	mallocman	–[MallocMan mallocThread:]
4	NSObject	Autorelease		00:07.447	0	mallocman	–[MallocMan mallocThread:]
5	NSObject	Release	1	00:07.492	0	mallocman	–[MallocMan mallocThread:]
6	NSObject	Release	0	00:07.524	0	Foundation	-[NSAutoreleasePool drain]

Summary

- Memory is a limited resource
- Instruments helps you avoid wasting/mis-using memory
- Be proactive and profile your app

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

Instruments Documentation
Instruments User Guide
Xcode documentation

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

Related Sessions

What's New in Instruments	Presidio Wednesday 11:30AM
Advanced Performance Analysis with Instruments	Mission Thursday 9:00AM
Performance Optimization on iPhone OS	Presidio Thursday 2:00PM
Advanced Performance Optimization on iPhone OS, Part 1	Mission Thursday 3:15PM
Advanced Performance Optimization on iPhone OS, Part 2	Mission Friday 11:30AM
Automating User Interface Testing with Instruments	Marina Wednesday 2:00PM

Labs

iPhone OS Performance Lab	Application Frameworks Lab B Wednesday 9:00AM – 11:15AM
Mac OS X Performance Lab	Developer Tools Lab A Tuesday 4:30 – 6:30PM
iPhone OS Performance Lab	Developer Tools Lab A Thursday 4:30PM – 6:00PM
iPhone OS Performance Lab	Developer Tools Lab A Friday 9:00AM – 11:15AM
Mac OS X Performance Lab	Application Frameworks Lab C Friday 11:30AM – 1:00PM

WWDC10

[redacted]