360|iDev

KiGi

DEBUGGING TIPS & TECHNIQUE

with **Xcode 7 & Friends**

Instructor:

Kendall Helmstetter Gelner

@kendalldevdiary

Kendall.Gelner@KiGiSoftware.com

materials found:

http://tinyurl.com/KDebug360iDev2015

WHO AM I

- Full-Time iPhone developer since release of SDK.
- Independent consultant for iOS development.
- Over two decades programming experience from corporate to end user.
- Over 50k Reputation on StackOverflow, great for ObjC or Swift questions!

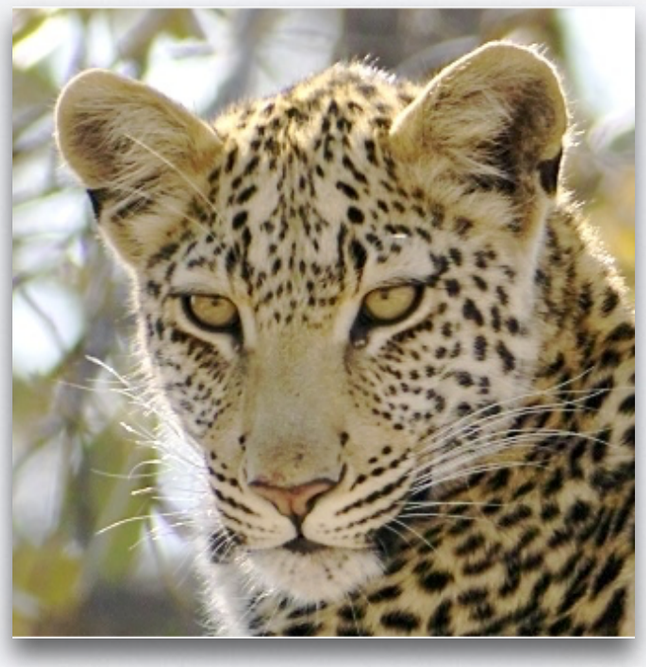
THINGS OF NOTE

- Slides and sample project on DropBox:

http://tinyurl.com/KDebug360iDev2015

- Xcode/iOS dev tips, tricks and discoveries on twitter:

@kendalldevdiary



FILES ON USB DRIVE

- Copy of these slides.
- Third party tools to install (not in BitBucket repo).
- DebuggingAppV5 (use with Xcode7 Beta 5, also on USB).
- Also copy of DebuggingAppV5, meant as an emergency backup to restore from.
- Buggy for a reason, careful using code from it! Third party parts OK.

ON DEBUGGING

“Beauty is more important in computing than anywhere else in technology because software is so complicated. Beauty is the ultimate defense against complexity.”

— David Gelernter

“The computing scientist’s main challenge is not to get confused by the complexities of his own making.”

— E. W. Dijkstra

“The most effective debugging tool is still careful thought, coupled with judiciously placed print statements.”

— Brian W. Kernighan, in the paper Unix for Beginners

BE THANKFUL FOR TOOLS

THE PHILOSOPHY

Debugging is giving yourself paths of visibility into your application, gathering enough information for inspiration to strike.

Inspiration arises from preparation and repetition.

Think of application as a data stream flowing through your classes; knowing where water should be is as important as knowing where it should not.

TRUTH AND LIES: THE DUALITY OF DEBUGGING

- The fastest way to understand why something is not working, is to first understand why it works.

- Debugging is as much about verifying what you think your program is doing as it is finding what is wrong.

- When all else is verified working, whatever remains, no matter how unlikely, is the Trouble.

THREE SECTIONS

- Visual Technique & Debugging
- Debugging Flow
- Debugging State

Demos for each tool or concept, brief Hands-on labs between sections!

VISUALS

VISUAL TECHNIQUE

- Add accessibility values for views. It helps when looking at a view hierarchy, and it's good for users too!
- You can add identifiers for layout constraints also, helps when examining later.
- Make use of container view controllers to keep things simple.
- Try out animations in Playgrounds before using in code.

SIMULATOR FEATURES

- Some simulator features to help examine visual aspects of the app:

Slow Animation

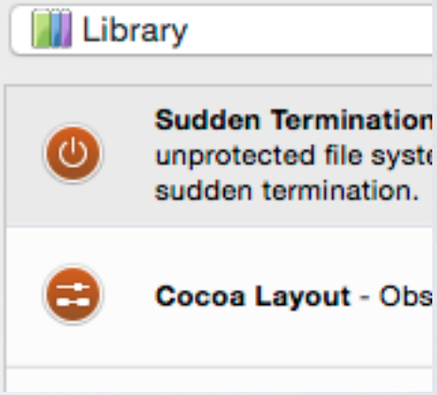
Graphics:

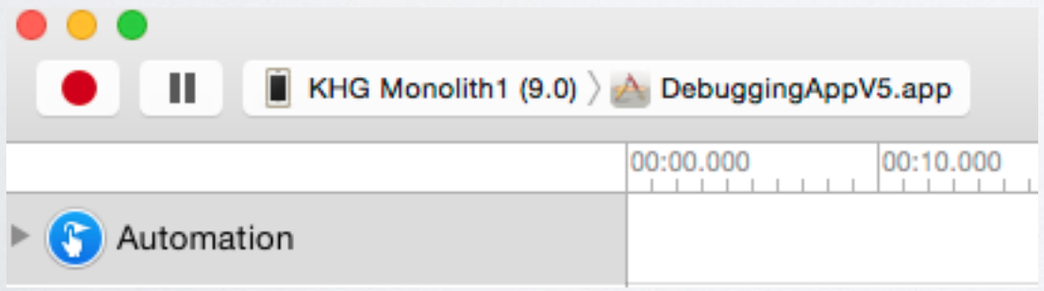
Blending Alpha/No Alpha

Misaligned

INSTRUMENTS REFRESHER

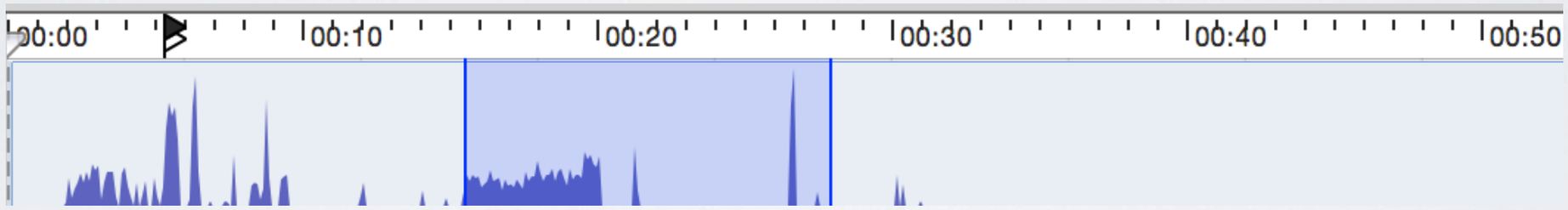
- Make sure you can run first, instrument may use older code.
- Can always stack more instruments in from Library.
- Run Profile from Xcode first, after that Instruments will remember details of how to run your app.





INSTRUMENTS REFRESHER

- Review options for each instrument in right side bar.
- Can add flags for points of interest with Cmd-Down.
- ‚Narrow regions of interest by dragging, no need for Alt.

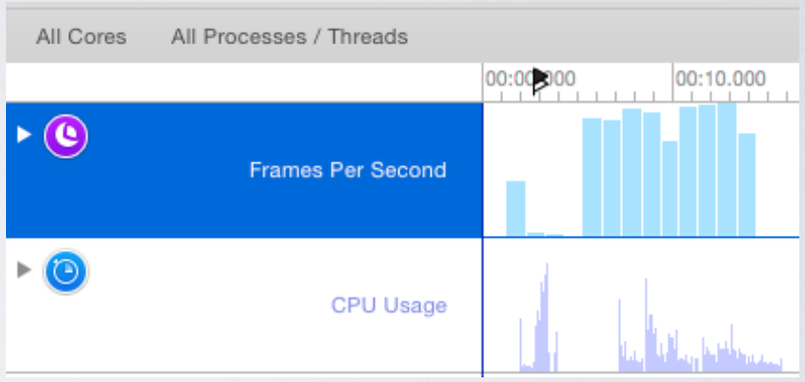


- Why wait? You can attach Instruments to a running process. Or you can attach to a debugger when you started with Profile.
- Starting with profile means you are running a Release build.

INSTRUMENT: CORE ANIMATION

- Measurable UI performance on device.
- Useful visual inspection of hidden graphics attributes.
- Limited examination of structure for third party apps.

INSTRUMENT: CORE ANIMATION



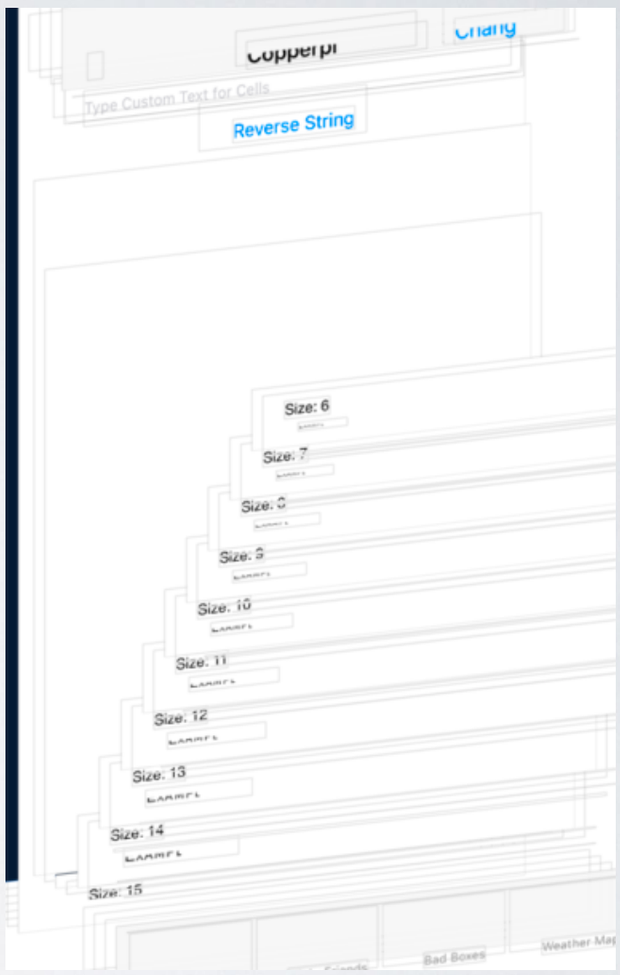
- Device only.
- Framerate display to indicate smooth UI.
- Color blended layers shows opacity at work.
- Color Misaligned shows you images that have been scaled.
- Flash Updates shows you when things are redrawn.

XCODE:
VIEW DEBUGGING

- Have to be running iOS8+
- Activated with debug control or:

  Debug->View Debugging->Capture View

- Pauses app and breaks out views.
- Can see view details on right, and left side in stack explorer.



XCODE:
VIEW DEBUGGING

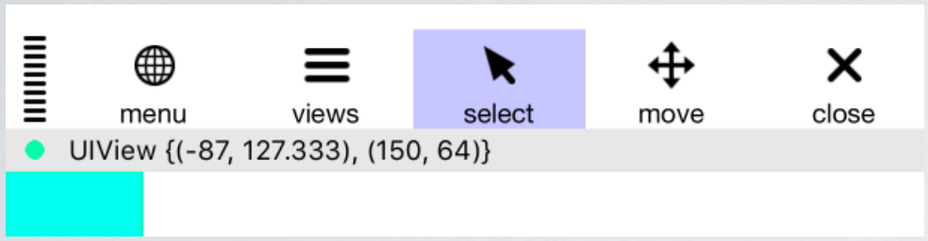
- Turning off clipping shows you views that have drifted off or are oversized.
- Can review constraints.
- Can discard front or rearmost views from view.

REVEAL

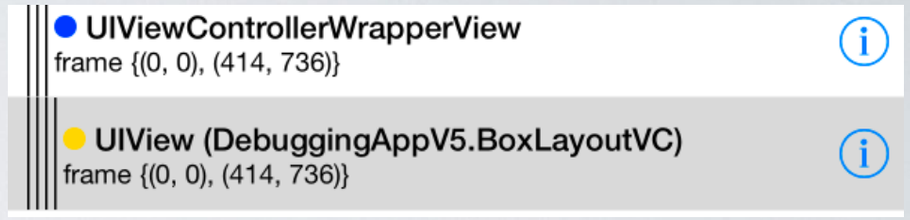
- Included in your "Extra" folder.
- Find at RevealApp.com -Trial version, costs $89 US.
- Shows breakout of views, allows manipulation.
- Just add framework (embedded in app bundle) and run!
- Well, OK, also add **-ObjC -lz** linker flags before you run.
- Well, OK, may also have to disable Bitcode.

PONY DEBUGGER

- Podspec in your "Extra" folder.
- Use XCConfig file to configure addition for projects.
- Install Xcode command line tools.
- Install & run the server (PonyDebuggerQuickstart.txt).
- Then add code to attach to Pony server.
- Install too involved to demo today.



FLEX

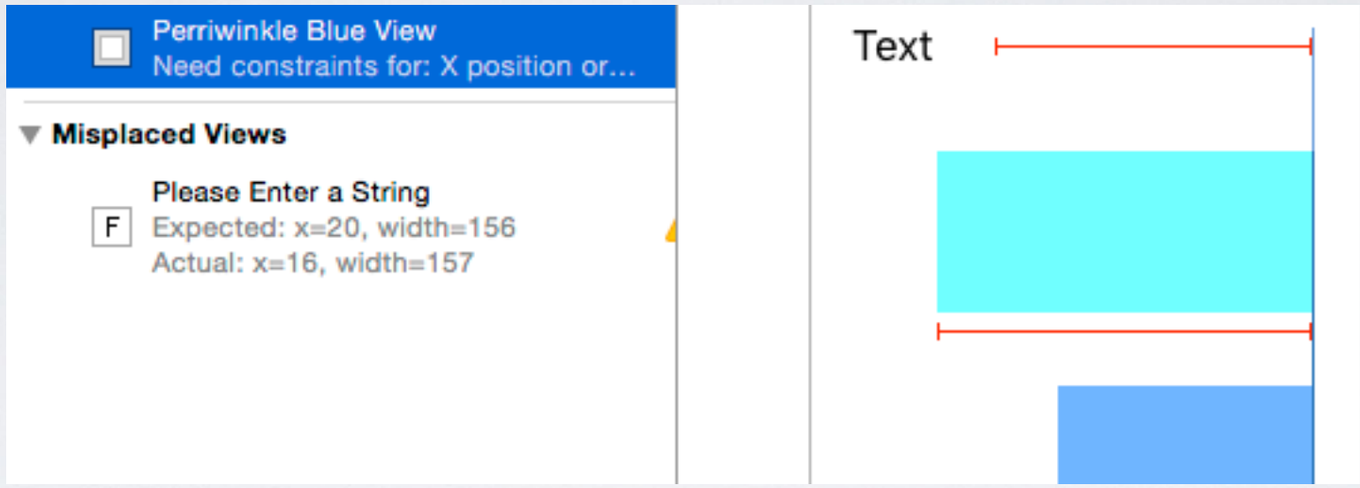


- In-App debugging console from Facebook (!)
- In Extras folder under Flex, on Github.
- Install as framework, to present UI one line of code:

```
[[FLEXManager sharedManager] showExplorer];
```

- For views - select, then examine/modify.
- Useful global screen/device examination also.
- File browser can mail app files.

XCODE:
CONSTRAINT DEBUGGING



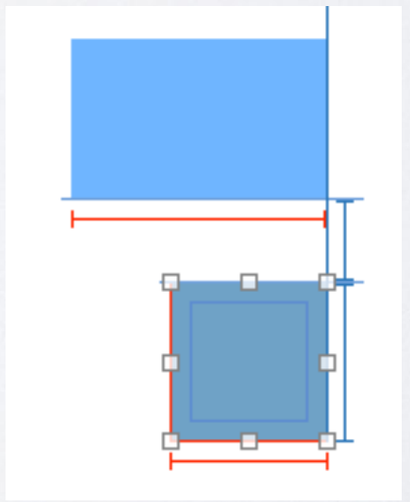
- First step is InterfaceBuilder warnings.
- Possible to examine constraints with Xcode view debugging.
- Can dynamically alter constraints with Reveal.
- Don’t forget the constraint names for key constraints!

XCODE:
CONSTRAINT DEBUGGING

- Think about constraints as width/height + where something should go.

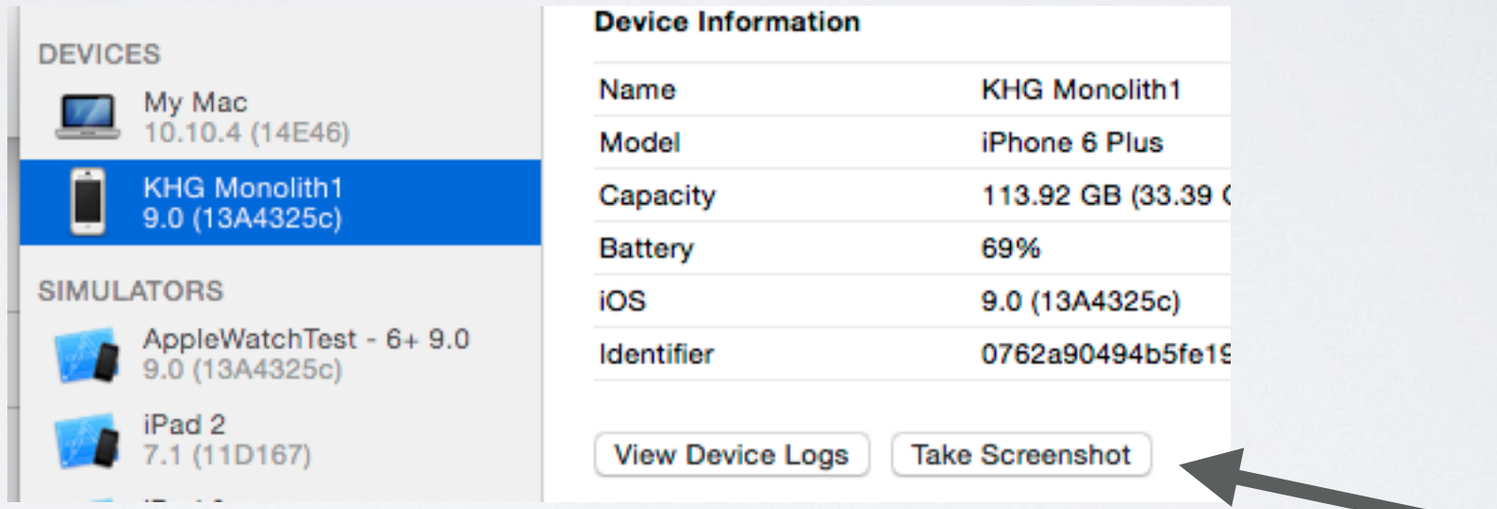
- A single error can cascade into many, so find the root of what the problem constraint is linked to.

- Xcode frame previews very helpful.



SCREEN SHOTS

- Grab from simulator (copy screen to clipboard).
- Also can ask device (from Devices) to take screen shot, send to desktop.



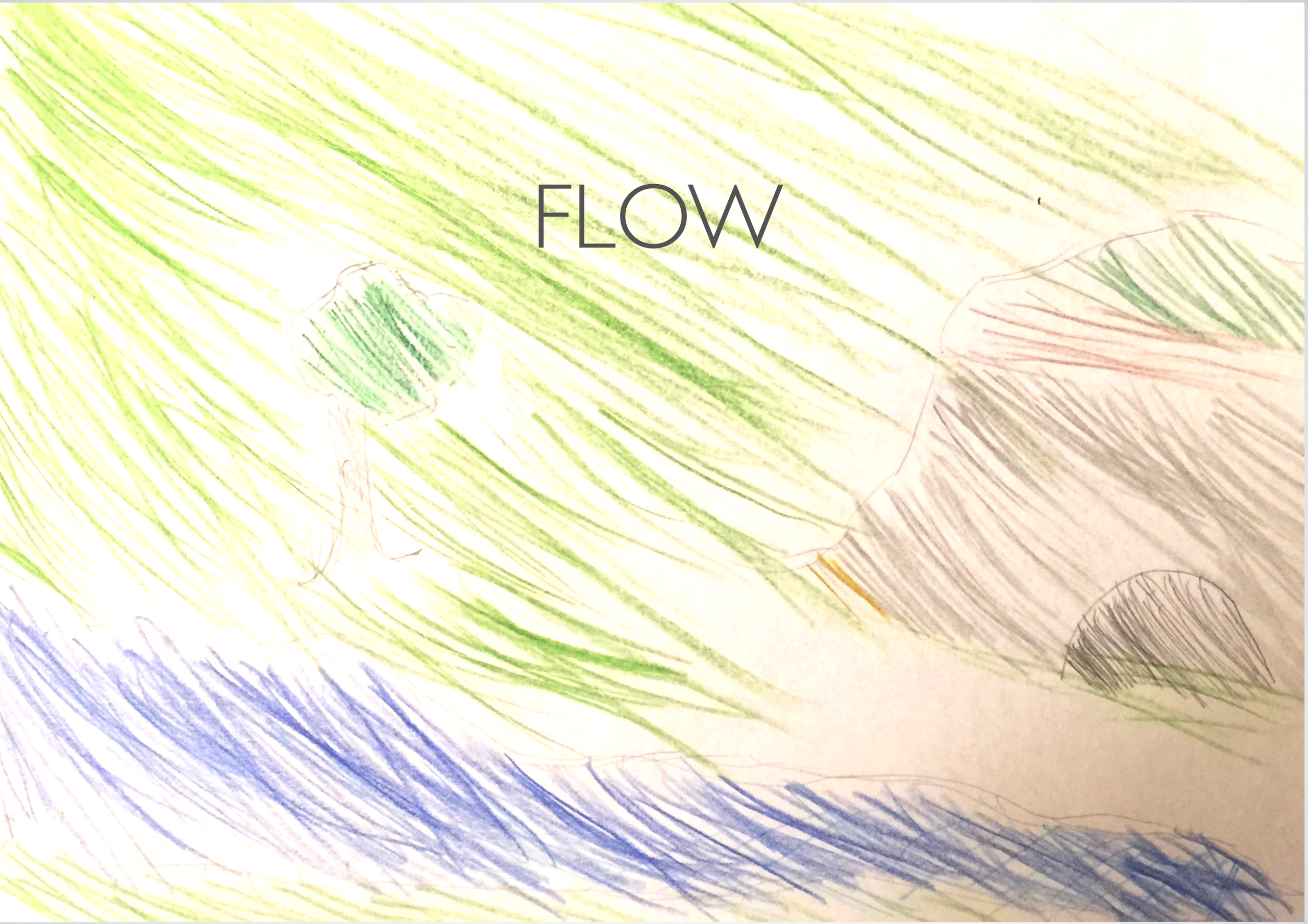
- From time to time, capture and save screens from your app for a baseline of what normal looks like.

LAB TIME

15 MIN

- Explore Core Animation effects on your favorite app.
- Fix (or add) constraints in Bad Boxes to remove warnings.
- Try using Xcode view examination.
- Load Reveal or FLEX into the app and examine application.
- Take a screen shot while Core Animation blending is on - what happens?

FLOW



FLOW TECHNIQUE

- Consider making parts of your app small frameworks, easier to try out techniques and test.

- Careful of memory management around blocks - delegates and long-term things like notification blocks.

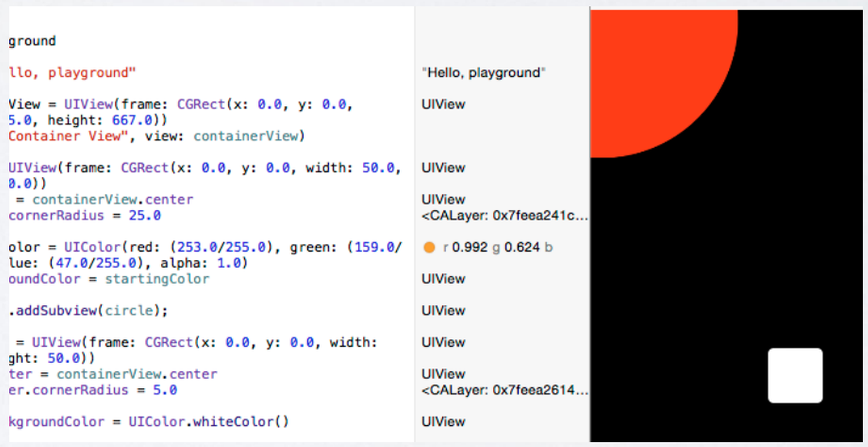
- Make it a habit to remember what memory and CPU graphs generally look like as your app runs, so you know when they go wrong.

PLAYGROUNDS: TESTS YOU ENJOY

- Playgrounds are (is?) code that runs whenever you come to visit.

- Allows you to quickly explore how code reacts to different (programmatic) input - sadly not interactive.

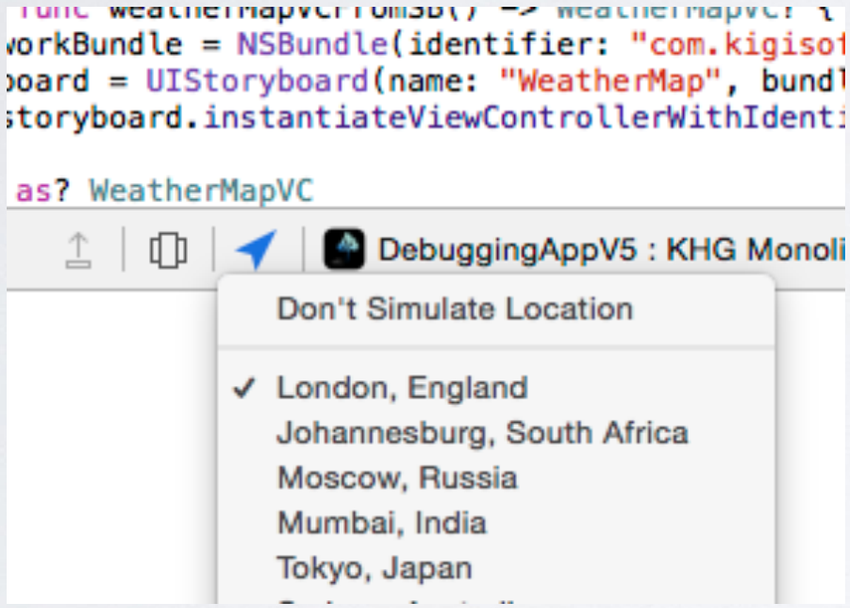
- Allows you to visualize results.

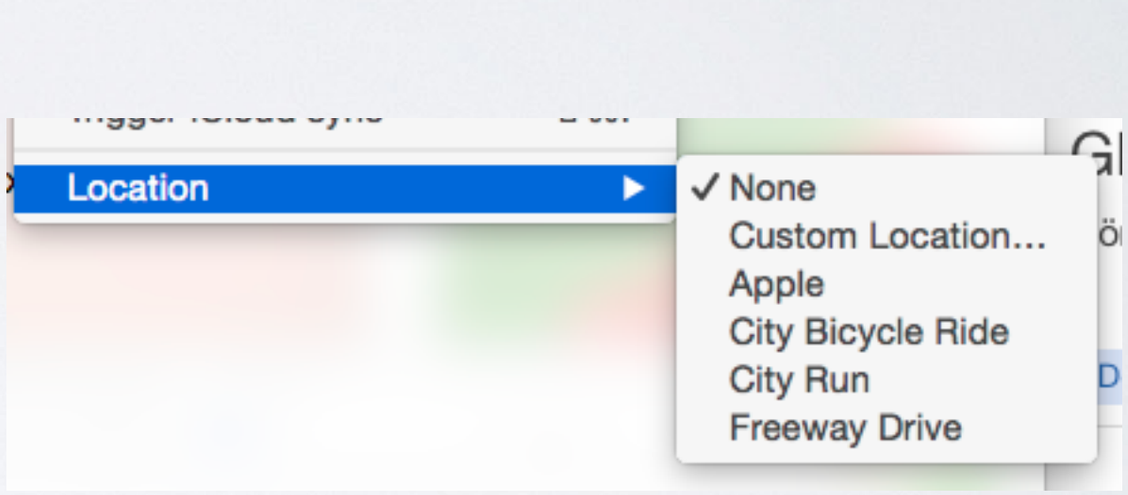


- Playgrounds can import any code from frameworks in the same Workspace.

SIMULATING LOCATION

- You can set simulated location in the simulator or device.



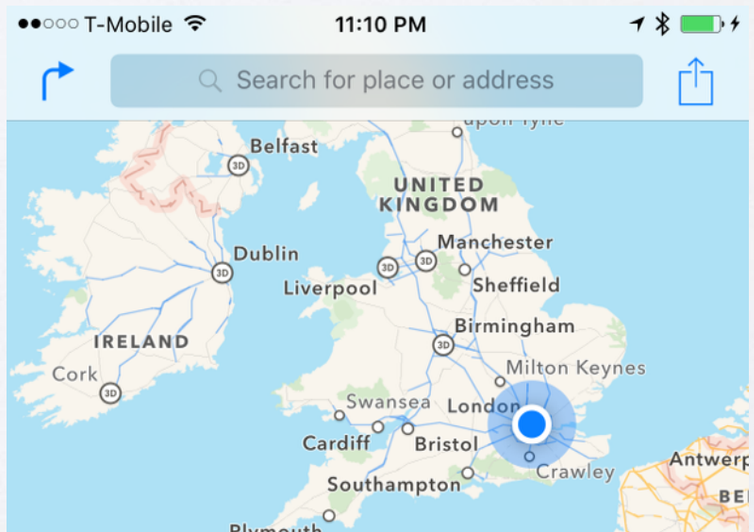


SIMULATING LOCATION

- You can have Xcode run through a GPX file simulating location.

- You can also use the Automation instrument to more accurately simulate location (includes direction and speed), does not work on device.

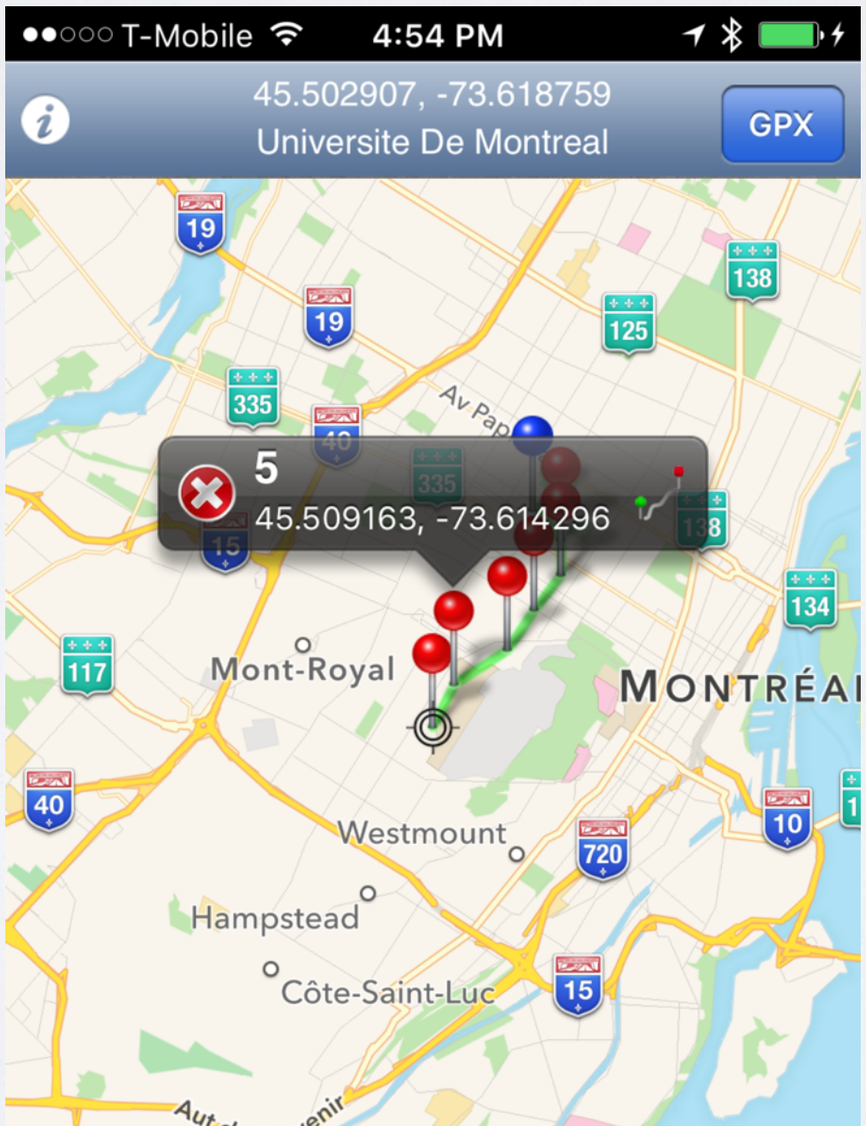
- If you simulate location on device, make sure to have Xcode stop simulation!



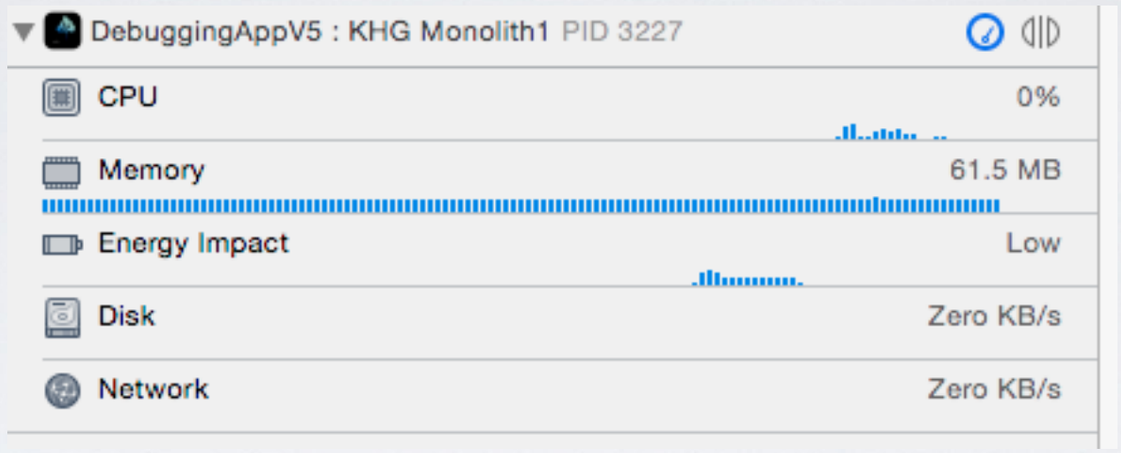
SIMULATING LOCATION

- How to make a GPX file? GPX Creator (free, w/ads):

https://itunes.apple.com/us/app/gpx-creator/id535468654?mt=8



INSTRUMENT: XCODE DEBUGGER



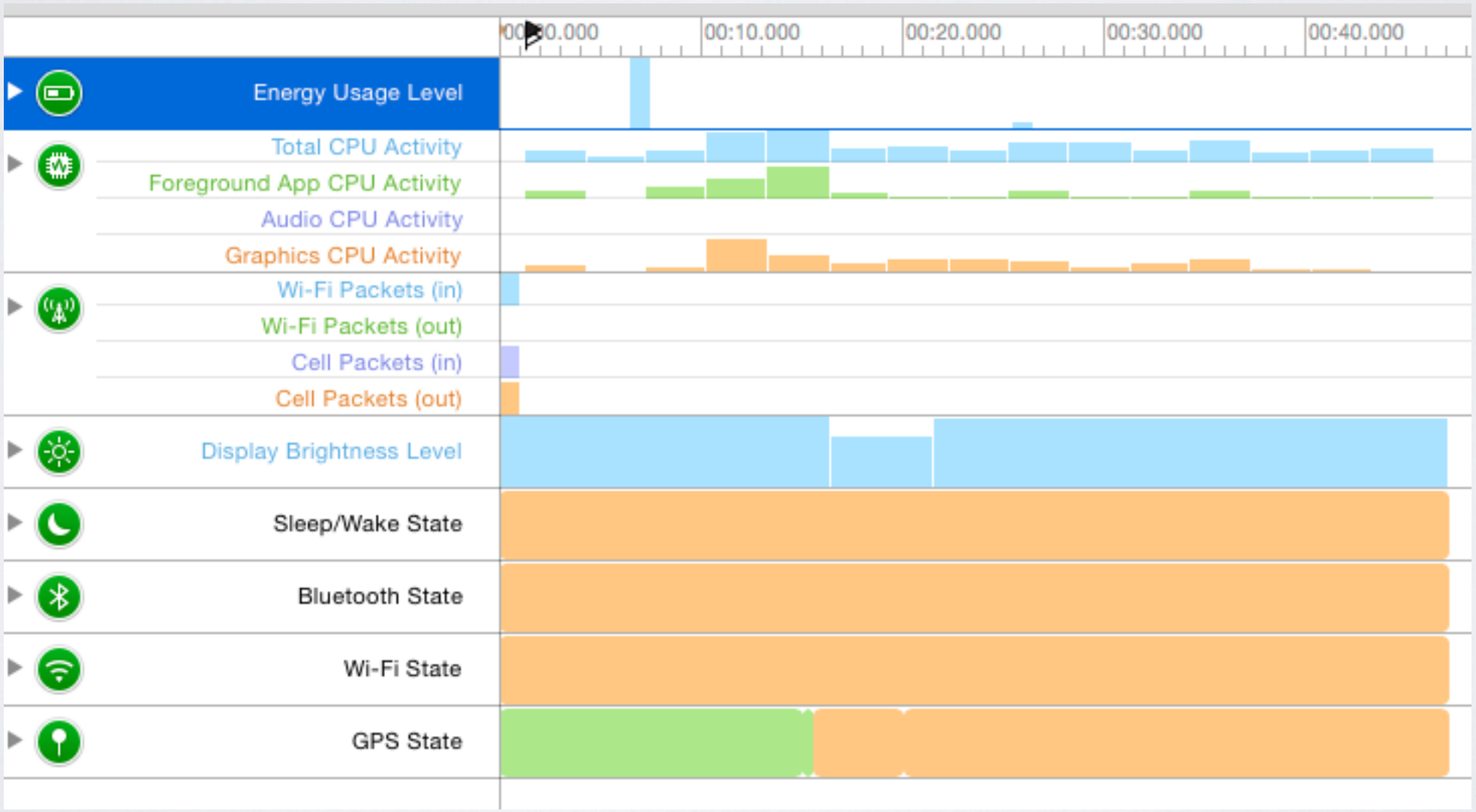
- Simplified/Summarized view of data you would get from many Instruments.
- Found in sidebar to left when running, each item has full page summary.
- Can launch full Instrument from summary panels (also transfer running apps).

NEW INSTRUMENT: ENERGY USE

- Fun overview of entire app operation!
- Can oversee any app, not just yours.

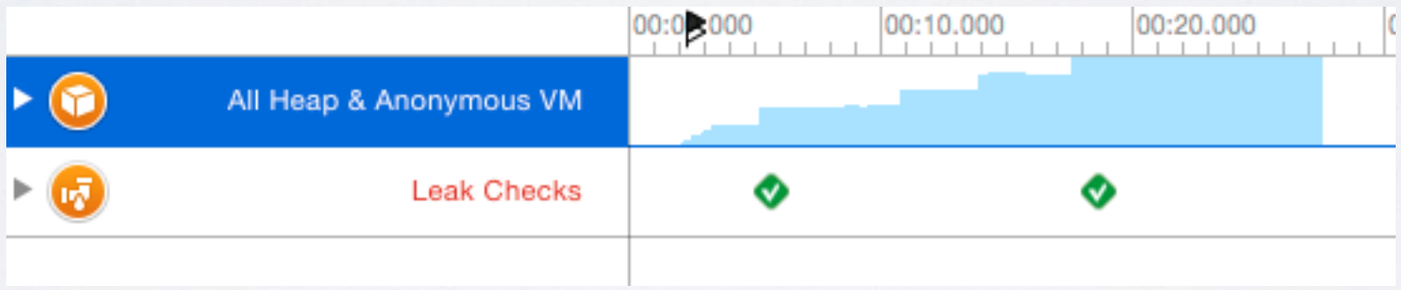
NEW INSTRUMENT: ENERGY USE

Family of many instruments



LEAKS VS CLOGS

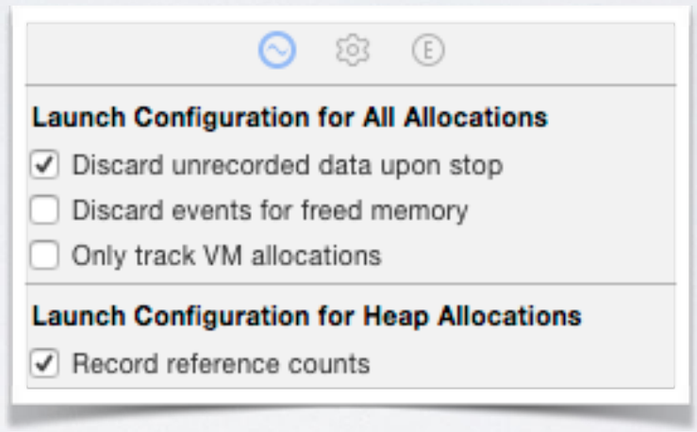
- ARC prevents common memory issues, but not all!
- Leaks are like a murder mystery.
- You can also "clog" memory with memory you didn't know you were still using.



- With ARC, mostly use Object Alloc instrument with Heapshot (now "Mark Generation").

INSTRUMENT: ALLOCATIONS (MEMORY)

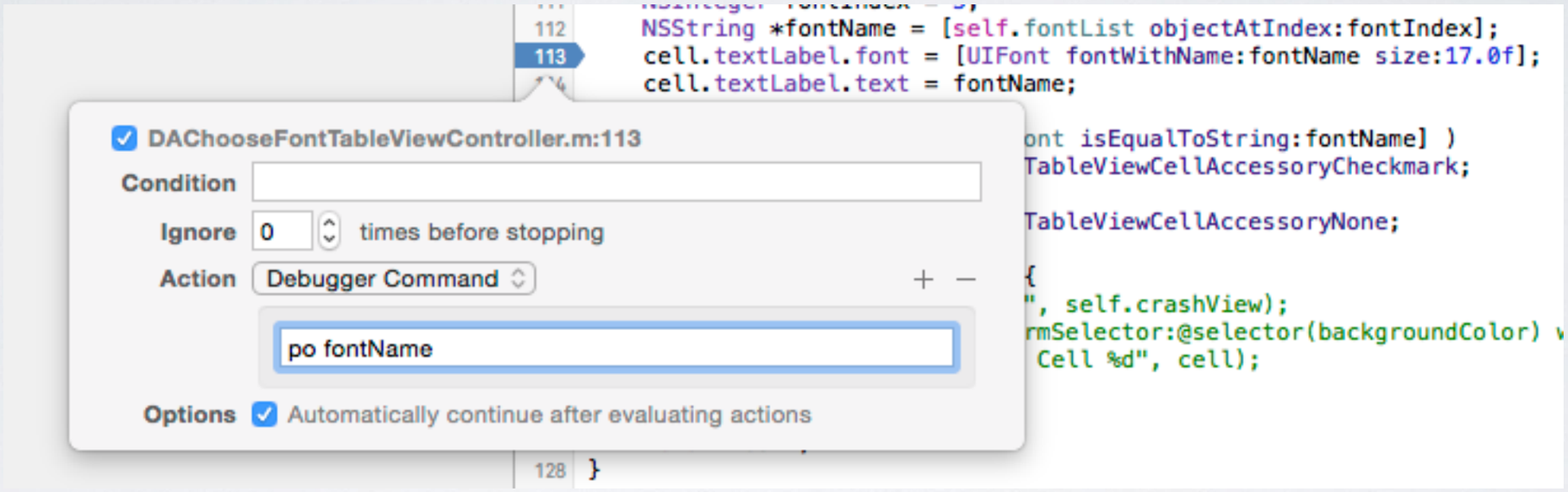
- Leaks sets up Allocations better than Allocations alone.



- Instrument used for memory diff (mark generation).
- Usually good to look for highest level object memory use.
- Good to save baseline performance from time to time for later review.

TEMPORARY LOGGING

- The simplest use of breakpoints is not to break.



- Expressions need massaging sometimes.

- Use expression in language where the breakpoint lies.

BETTER LOGGING

- NSLog/print() always outputs, even for customers.
- Evaluating data for NSLog can slow things down.
- Instead - consider DNSLog, #DEFINE away for AppStore.
- Import DNSLog.h file in each file to use, PCH files gone.
- debugDescription can also be implemented for extra data.

BETTER LOGGING

- For Swift, no #DEFINE, use libraries.
- One Swift option is XCGLogger (in Extra/Logging)
- Add code directly to project for easier use.

```
let log = XCGLogger.defaultInstance()
```

```
log.debug() {
    let result = "New location : \(firstLocation)"
    return result
}
```

- Can use closures when logging to get same effect of no-cost logging of complex statements.

ADVANCED LOGGING

- Consider using blocks for expensive log calculations.
- Local notification logging allows for background logging.
- Don't forget to ask for permission to use notifications!

REMOTE LOGGING

- Analytics packages like Flurry can tell you where users go in the app, and where they stumble.

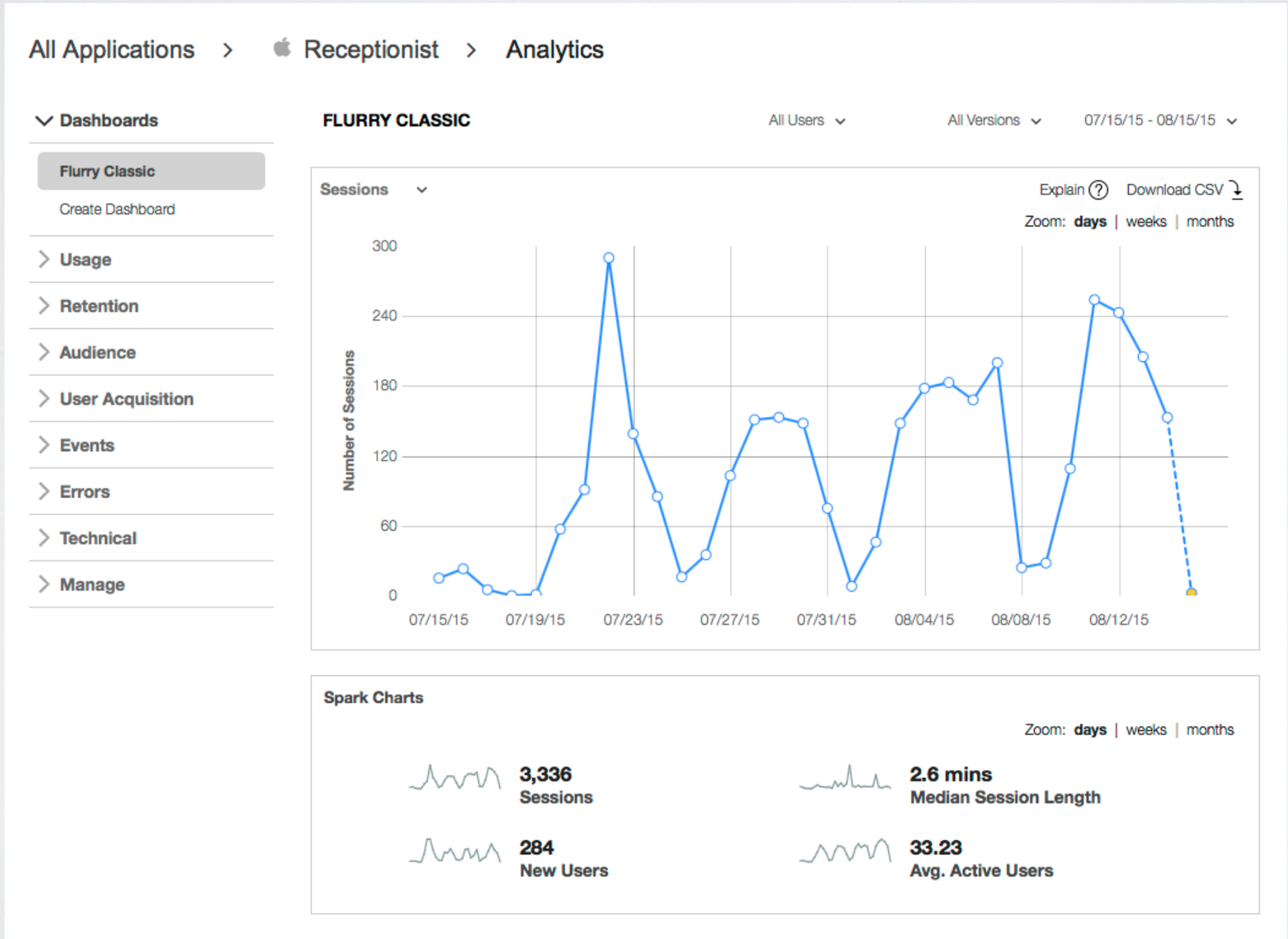
- Very easy to add tracking.

```
- (IBAction)approvePressed:(id)sender {
    [Crittercism leaveBreadcrumb:@"User approved image"];
    [Flurry logEvent:@"LIVE_CAM_IMAGE_APPROVED"];
    [self.delegate tookPhotoWithImage:self.capturedProcessedImage];
    [self cleanupPhotoStuff];
}
```

- Some users sensitive to tracking, may want to opt in or at least inform.

REMOTE LOGGING

Sessions



REMOTE LOGGING

Events

Event Summary Statistics Explain Download CSV

Event Name	Total Event Occurrences	Event Occurrences (Daily Avg)	Unique Event Users (Daily Avg)	Events per Session (Daily Avg)	Analyses
HOME_STATE_HOME	20,697	646.78	21.63	6.76	
TIMEOUT FIRED	14,037	438.66	22.91	4.59	
BUTTON_PRESSED	10,147	317.09	20.91	3.32	
PUSHER_SETUP	10,112	316.00	18.16	3.30	
PUSHER_CONNECTED	5,793	181.03	17.13	1.89	
HOME_STATE_GRID	5,358	167.44	16.31	1.75	
CHAT_MESSAGE_SENT	5,118	159.94	18.16	1.67	
CHAT_READY	4,849	151.53	18.16	1.58	
PUSHER_FAILEDCONNECT	4,522	141.31	1.63	1.48	
APP_SUSPEND	4,450	139.06	30.22	1.45	

REMOTE LOGGING

Paths

- Start Session (3,250 times, 100.0%)
  - APP_OPEN (1,228 times, 37.8%)
    - LOGIN_SUCCESS (751 times, 23.1%)
      - HOME_STATE_HOME (455 times, 14.0%)
      - APP_SUSPEND (164 times, 5.0%)
      - HOME_STATE_GRID (80 times, 2.5%)
      - HOME_STATE_COPY (17 times, 0.5%)
      - HOME_STATE_CITIZENSHIP (15 times, 0.5%)
      - OPEN_CAMERA_VIEW (11 times, 0.3%)
      - LOGIN_SUCCESS (5 times, 0.2%)
      - APP_TERMINATED (3 times, 0.1%)
      - CONFIG_RELOADED (1 times, 0.0%)

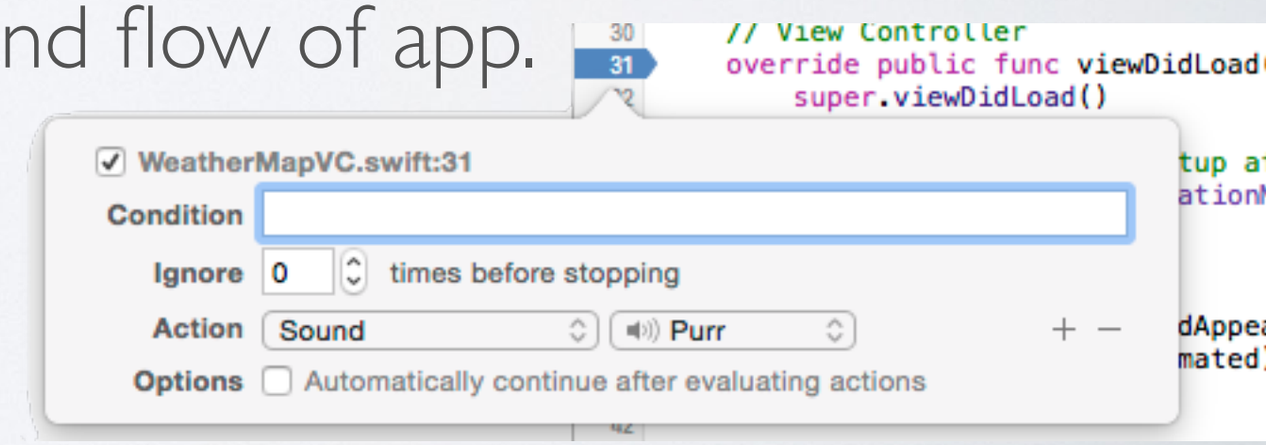
Who had issues?

Page 1

Session Time	Version	Details
08/14/15 10:05:22 MDT	2.59.0 (iPad)	Apple iPad Air 2 - Wi-fi User ID:[redacted]
2) PUSHER_FAILEDCONNECT		
4) PUSHER_FAILEDCONNECT		
6) PUSHER_FAILEDCONNECT		
8) PUSHER_FAILEDCONNECT		
10) PUSHER_FAILEDCONNECT		
12) PUSHER_FAILEDCONNECT		

DEBUGGER AS AUDIO PROBE

- Covered in older WWDC video, but there is a better why...
- Understand relations between calls in an API, or the flow of data in your app.
- Also, speech (numbers only...)!
- You can use your own sounds placed in ~/Library/Sounds
- Put in key places to understand flow of app.



THE DEBUGGER TOUCH

- Newer devices like AppleWatch or Macbook allow app to touch you!

- Patterns of touch can inform you of key app status…

```
WKInterfaceDevice.currentDevice().playHaptic(.DirectionUp)
```

- No indication of touch in simulator, could play sounds…

INSTRUMENT: CUSTOM (DTRACE)

A Eulogy

INSTRUMENT: TIME PROFILER

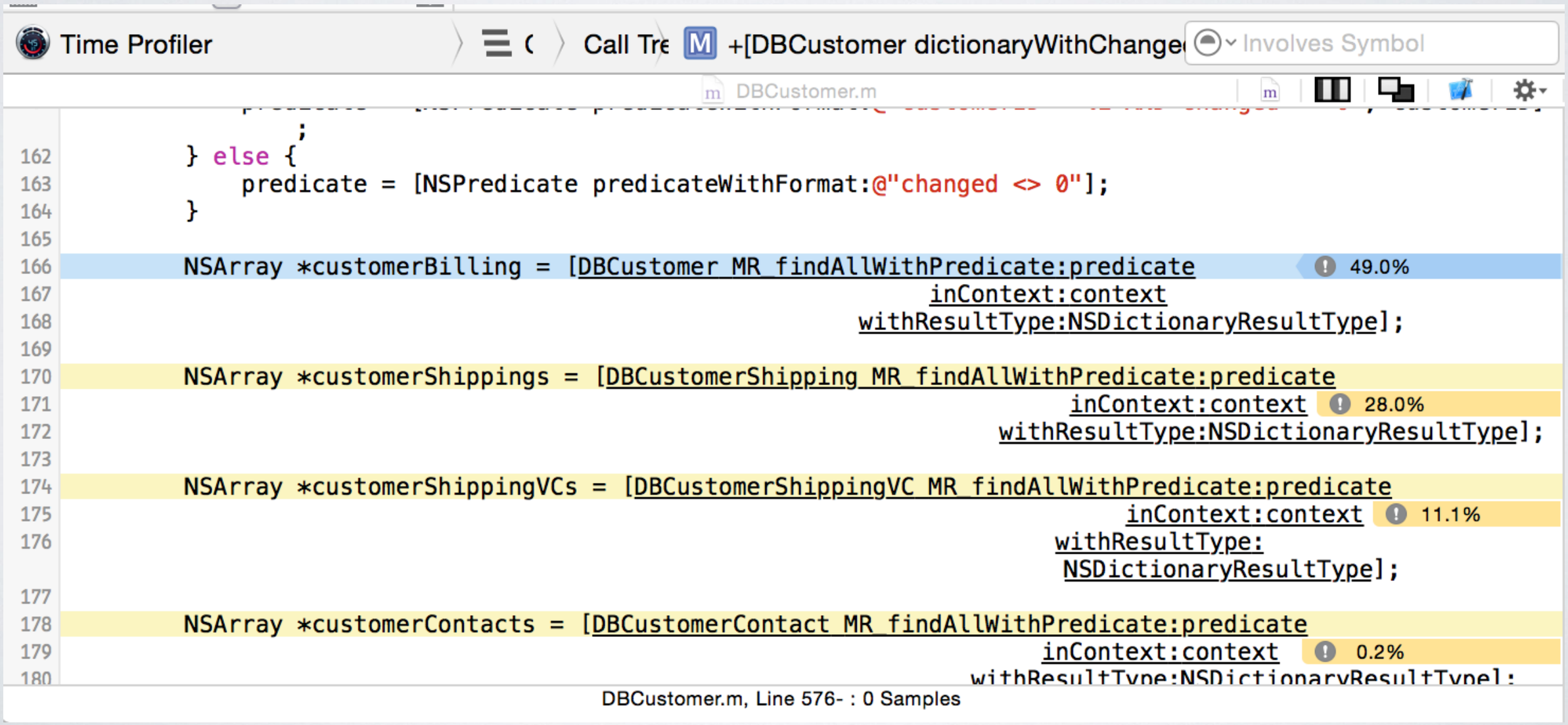
- Great way to check why an app is busy - or why it's not.
- Helps you focus on worst performance first.
- Check out Core view to ensure proper use of cores on device.
- Xcode simpler Profiler view can act as guideline for what needs Time Profiler.
- Record Waiting Threads can show causes for pauses.

INSTRUMENT: TIME PROFILER

That's a lot of fetch!

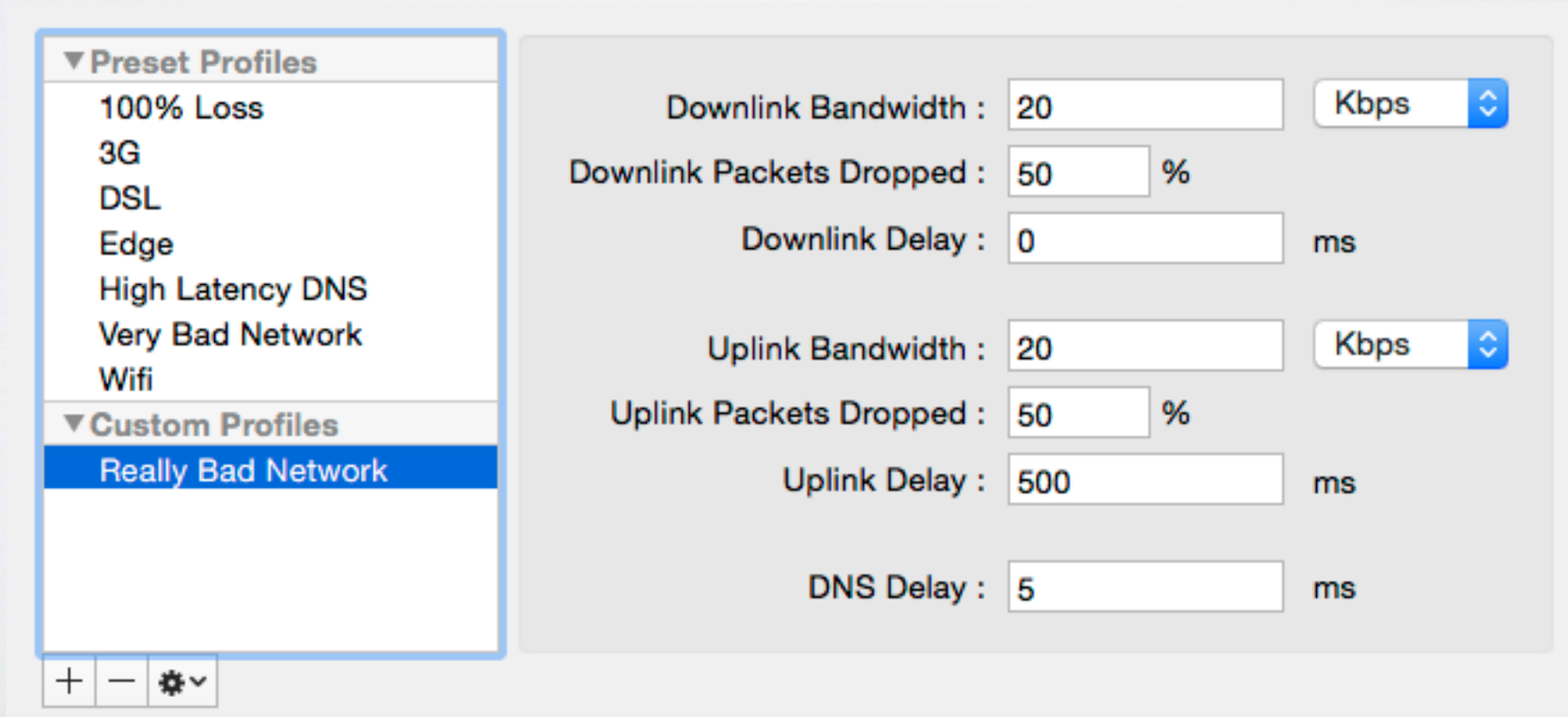
INSTRUMENT: TIME PROFILER

Drilling down to code, now we know what to improve…



NETWORK LINK CONDITIONER

- In system control panel, installed for you.
- Affects all the apps on the system when enabled!
- Existing settings are OK, but you can make a better version of bad networks...



CHARLES

- Also in "Extra" folder.
- Costs money, but is worth it (free LONG trial).
- Examine, modify, or replay requests.
- Easily filter to see requests that matter.
- Set HTTPS hosts to proxy traffic (install Charles local cert):

  http://www.charlesproxy.com/ssl

- Make sure a "curl" or browser call works before you try to code against something or look for bugs.

LAB TIME

15 MIN

- Use audio breakpoints in viewDidLoad to see if two or more tabs load all at once, or on demand…

- Use time profiler and explore time used when scrolling debugging app (or your own)!

- Try setting custom locations for an application.

- Load Charles and try editing some network calls.

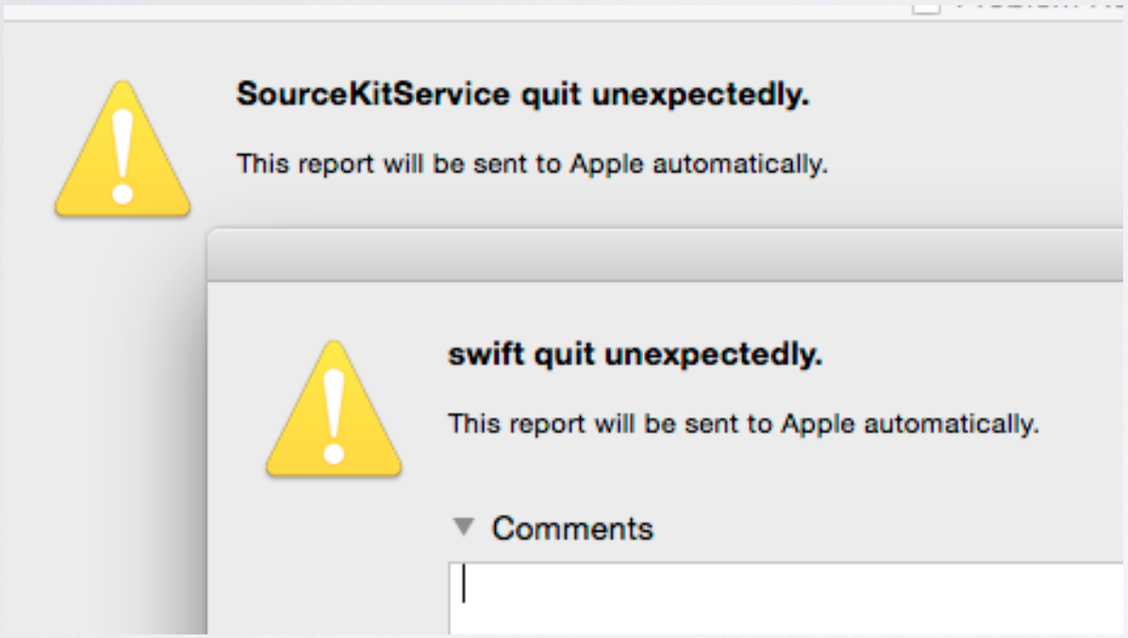
STATE

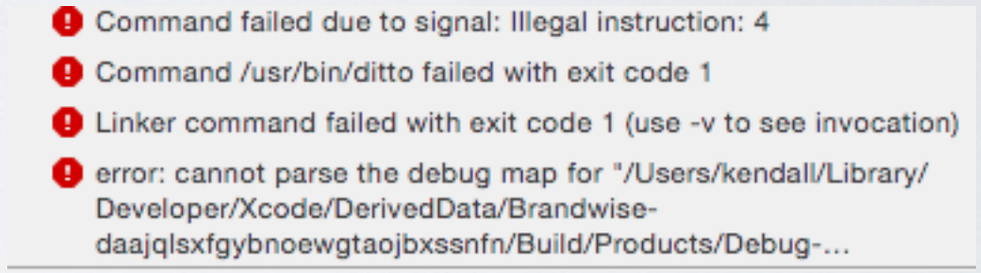
STATE TECHNIQUE

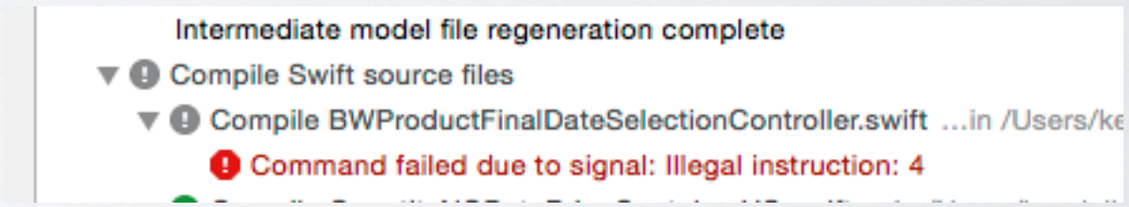
- Name GCD Queues, to more easily tell what is where.
- Use crash trackers to help find crashes you can't duplicate.
- Listen to, and repair warnings!
- When using Swift, avoid "!" if at all possible

COMPILER CRASHES

- Debugging An Example - Swift Compiler Crash:







- Look at first error in compilation logs.

- Process of elimination!

LLDB - USEFUL COMMANDS

- po "recursiveDescription" works on any view to dump view hierarchy (in ObjC).

- You can use myArray.map { $0.someValue } in LLDB to get out specific attributes you want to see!

- if you crash is in objc_msgsend, you can at least find the method name called with:

```
p $ARG1
```

LLDB - USEFUL COMMANDS

- Set property watchpoints (max two on device) with:

```
watchpoint set variable self->_myProp
```

Swift: `watchpoint set variable self.myProp`

```
watchpoint delete <number>
```

- thread until <linenumber> for quick jump (or use mouse)
- If LLDB does not "see" a class, try using NSClassFromString.

LLDB - USEFUL COMMANDS

- To see types in Swift use:

```
frame variable -d r myVariable
```

- To help LLDB understand types for ObjC:

```
expr @import UIKit
```

Add as continue breakpoints!

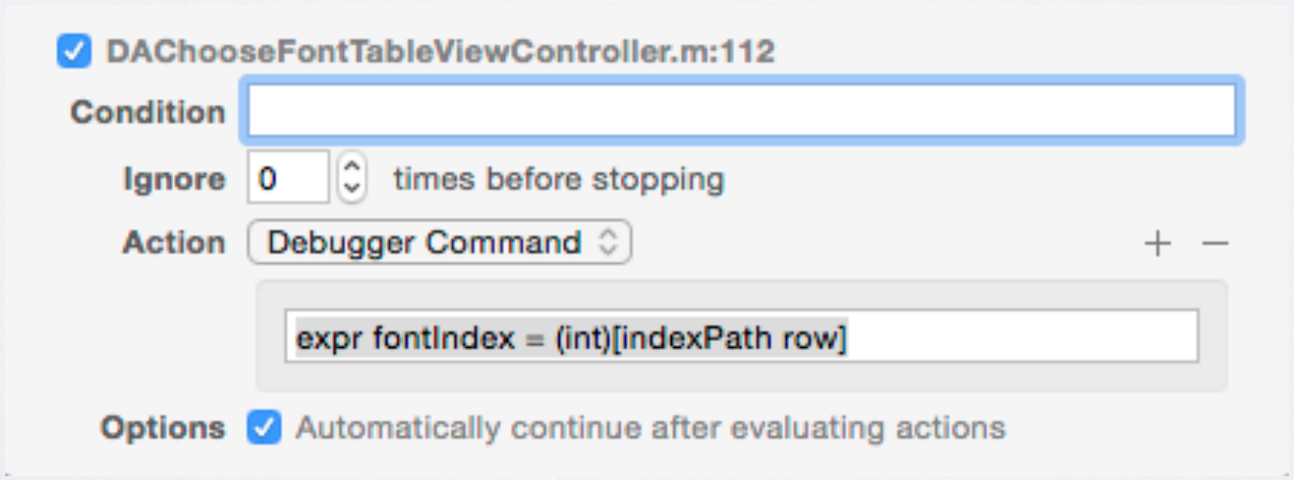
```
expr @import Foundation
```

- You can un-mangle a swift name with:

```
swift-demangle
```

LLDB - EXPRESSIONS

- Expressions compile real code to run in app.
- Access with "expr" command (GDB used "p" or "po" to evaluate expressions, so can LLDB).
- Define debugger global variables with "$" prefix (use id for complex types).
- Use with continued breakpoints for real-time patching!



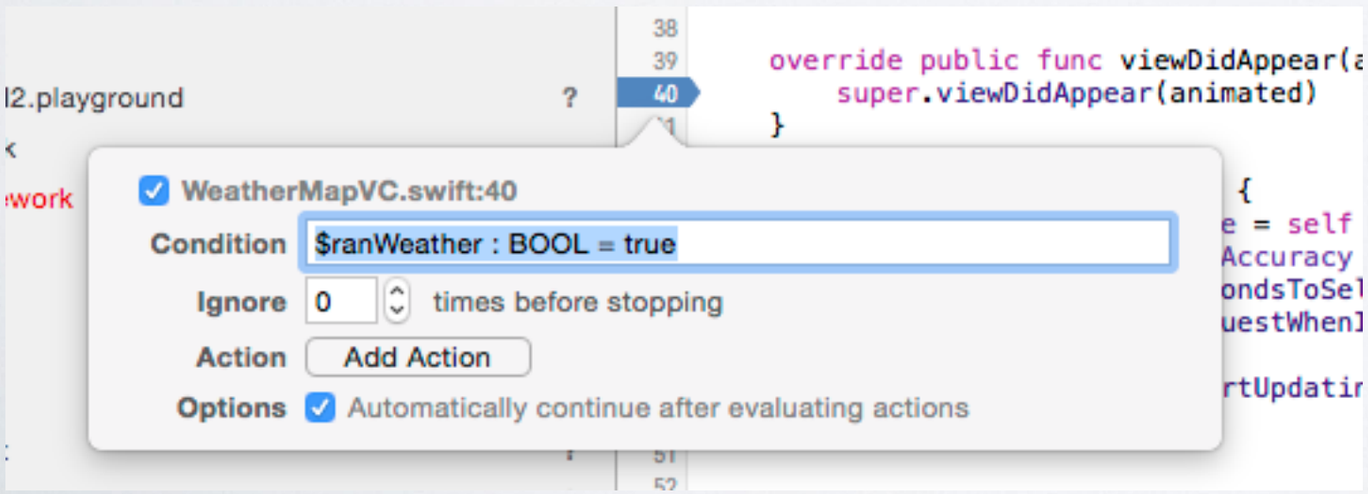
CONDITIONAL

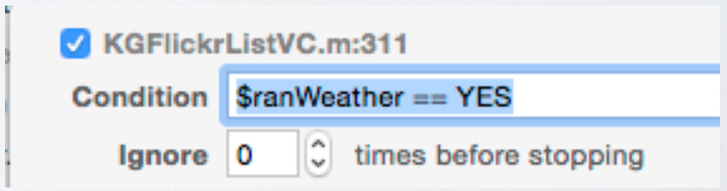
- For any expression used in a breakpoint condition, make sure the LLDB command line accepts it first.

- Conditional breakpoints can use any expression.

- Note you may have to cast return values for every call.

- Can add condition on LLDB defined variable.





STACKFRAME FUN

- Now you can finally see stack frames from before a queue call!
- Can see only queue stack frames (upper right corner control).
- Also means sometimes historical frames you can't examine. The icons for these lines are grey.
- Option key on open/close thread traces opens all.

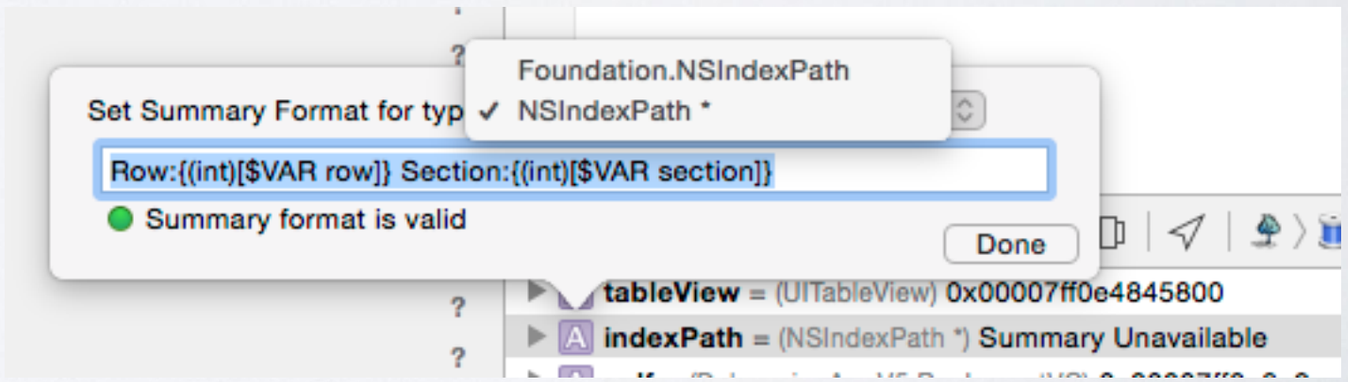
DATA FORMATTERS

- Use to provide better view on your local variables.
- Can define in XCode variable window:

```
TheTextValueIs is {(NSString*)[$var size]}:s
```

- Problem: Format must be in language where you stop.
- Solution : Heavier use of debugDescription() on key classes in debugging extension.
- Future: Xcode has plans…

  Not working yet.



DATA FORMATTERS

- State is: Decay
- Can create custom python summaries
- Easier than it sounds, but has stopped working.
- Examples at

  http://llvm.org/svn/llvm-project/lldb/trunk/examples/customization/bin-utils/

CUSTOM QUICKLOOK

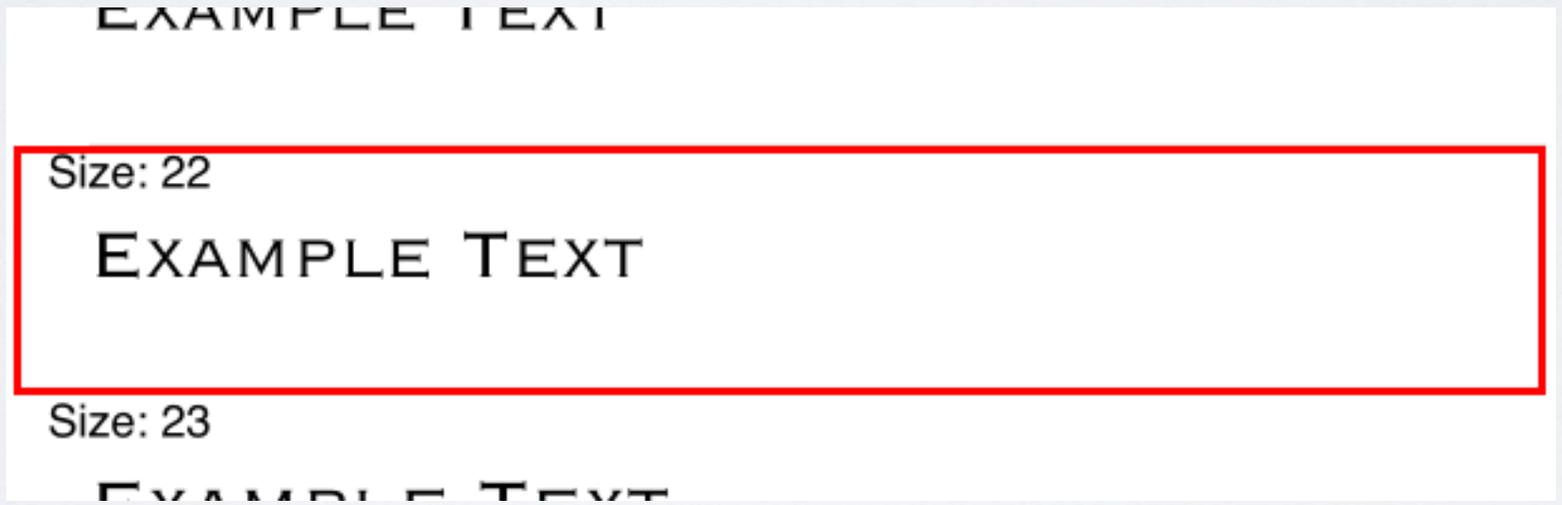
- You can create quick look for your own classes!
- Return either NSAttributedString or UIImage (or more!)!
- Add method **-debugQuickLookObject** (returns **ID**)!

```
- (id) debugQuickLookObject
{
    NSMutableAttributedString * finalString = [[NSMutableAttributedString alloc] initWithString:@""];
    [finalString appendAttributedString:[self attrStringForString:@"font info:" withColor:[UIColor blackColor]]];
    [finalString appendAttributedString:[self attrStringForString:[self.exampleTextLabel.font description] withColor:[UIColor blueColor]]];

    [finalString appendAttributedString:[self attrStringForString:@"\nText:" withColor:[UIColor blackColor]]];
    [finalString appendAttributedString:[self attrStringForString:self.exampleTextLabel.text withColor:[UIColor redColor]]];

    return finalString;
}
```

- QuickLook now also shows data formatter summary strings.

CUSTOM LLDB COMMANDS

- You can create custom LLDB commands in Python.
- Great example from Facebook(!) - Chisel.
- Commands include **border/unborder:**



- Also see view controller hierarchy with **pvc**.

INSTRUMENT: ZOMBIE

- For specialized use, in hunting dow source of crash.
- Not as needed with ARC.
- Fire it up, wait for crash and examine retain/release info.
- A new mystery - who released when they should not have? Or, who maintained a reference that went bad?
- Consider using weak references to fix issues.

LAB TIME

15 MIN

- Try setting a debugger variable in one breakpoint and viewing it a different breakpoint.

- Use EXPR on a continue breakpoint in cellForRow to strip out extra text on flickr user name output (nobody@flickr.com).

- Successfully have the debugger stop at a cell when the tag count is greater than 10, and speak the index.

- Install the debugger commands and experiment with adding a border to views, or add a custom quickLook output.

DATABASE

CORE DATA DEBUGGING

- Instruments can help track saves and other operations.
- Also just before a save, look at ManagedObjectContext updatedObjects or insertedObjects to see what will be saved.
- Output from Core Data objects in debugger may truncate strings! Your full string is there.
- Use valueForKey: to get properties out of a Core Data object in the debugger. willAccessValueForKey: triggers fetch, as does "allObjects" on relationships.
- See WWDC2013 session "Core Data Performance Optimization and Debugging" (Session 211).

CORE DATA STRUCTURE

- Be careful of deletes, very dangerous.
- isDeleted does not really mean isDeleted.
- Inheritance means flat tables.
- Make everything you can optional, validations dangerous.
- Make use of helper libraries, but don't get too far removed.
- Create new model version for every app store release (if needed).

CORE DATA FILES

- Keep in caches directory if not permanent.
- Auto migrate, destroy if that doesn't work.
- iCloud key/value pretty stable, Core Data documents fiddly.
- You can pre-load databases, copy them into writable directory on app start.
- You can copy DB from the simulator directories to keep different cases around (as long as you have the same version) or transplant databases from device.
- Can query Core Data file with SQLLite tools.

CORE DATA FILES

- Structure for storage changed in iOS8, now complex to find sim files.

- Use SimPholders to find.

- Use iExplore to extract from device.

- Or, add code to print out where documents live:

```
#if TARGET_IPHONE_SIMULATOR
// where are you?
NSLog(@"Documents Directory: %@", [[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
inDomains:NSUserDomainMask] lastObject]);
#endif
```

CORE DATA OBJECTS

- Very Good Idea to use Mogenerator to generate CoreData objects for you.
- Creates core data objects you can add behavior to.
- Great place to mediate calls that require Predicates.

CORE DATA SETS

- You can create as many simulator variants as you like in Xcode now.
- Consider a simulator for different custom data sets.
- Also can copy DB files between simulators.

LAB TIME

OPEN

- Use CoreData instruments to examine Flickr saving behavior.
- Experiment with examining data objects from Flickr calls.
- Otherwise, open time to ask more questions or try something again from previous lab!

THANKS FOR ATTENDING!

KiGi

Instructor:

Kendall Helmstetter Gelner

@kendalldevdiary

Kendall.Gelner@KiGiSoftware.com

materials found:

http://tinyurl.com/KDebug360iDev2015

RESOURCES

Much more esoteric Perceptual Debugging talk by me (AltConf 2015):
https://realm.io/news/altconf-kendall-gelner-perceptual-debugging/

Mogenerator:
http://rentzsch.github.io/mogenerator/

Java JRE (for Charles):
https://www.java.com/en/download/

Charles Web App:
http://www.charlesproxy.com

Cycrypt (not talked about, but interesting to contemplate):
http://www.cycript.org
Intro video: https://www.youtube.com/watch?v=5d1cK0nq4GY
Dynamic modification of running code.

LLDB:
http://lldb.llvm.org/tutorial.html - tutorial on common commands
http://llvm.org/svn/llvm-project/lldb/trunk/examples/customization/bin-utils/ - python examples
http://stackoverflow.com/questions/7677613/can-anyone-share-a-sample-lldbinit-file - lldbinit examples
http://lldb.llvm.org/varformats.html - LLDB variable formats

Chisel:
https://github.com/facebook/chisel

SimPholders:
http://simpholders.com