

5th Day - iOS Swift Training in Barcelona

where to find info about bugs?

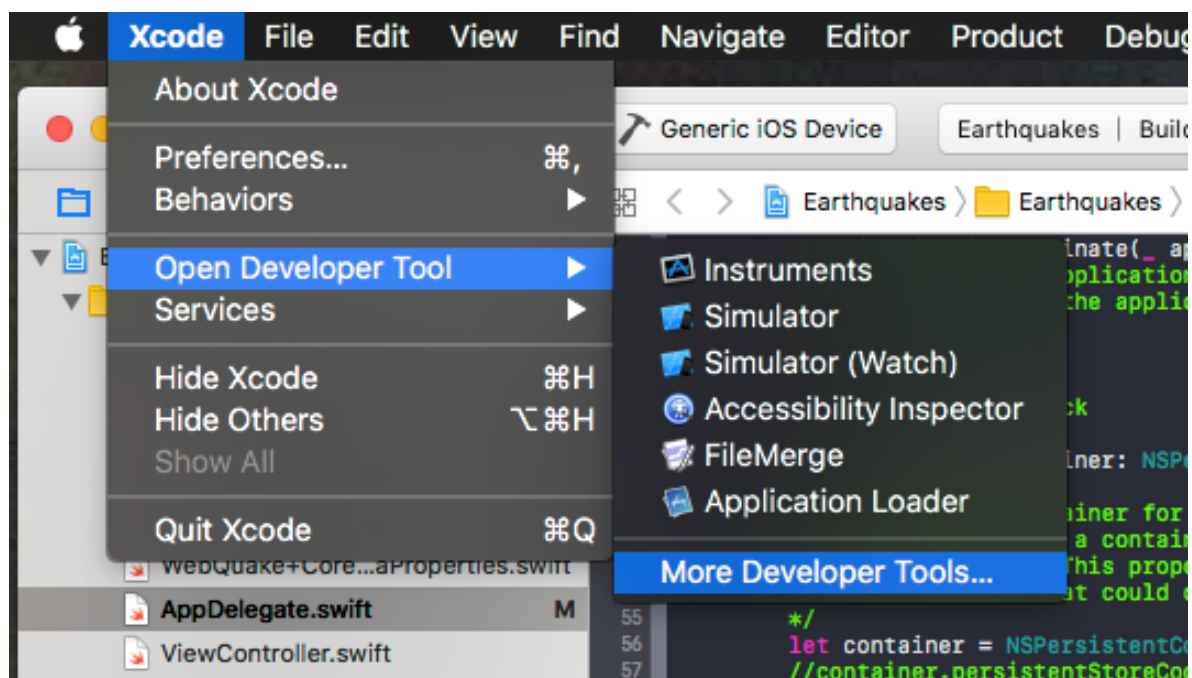
What is available on linux

GM Gold Master: public release

Network Link Conditioner tool

install it

very useful



Touch and Gesture

Touch is a finger touching the screen.

iOS is multi touch. Up to 21 simultaneous touch simultaneously !!

a UITouch for each touch

each object will track one finger.

Apple owns the patents.

On 3D Touch screen you can also track the force of the touch.

You can ask using `UITraitCollection` if the device supports the 3D touch.

iOS also create a `UIEvent`, which contains a collection of touches. It is a gesture.

Formerly `UIResponder` class was used to recognise gestures. It is a very basic class in `UIKit`.

Then they introduced the `UIGestureRecognizer`.

It is an abstract class and you need to subclass it.

Two types of state machines for the recognition

- Discrete (tap, swipe, ..)
- Continuous

Sensors

Accelerometer

Gyroscope

Magnetometer

Barometer

Pedometer

They are also available in the watch (which has the HR too)

Core Motion

```
import CoreMotion
```

Tip: stop it when not in use

Do not create multiple `CMMotionManager` instances in the same VC because it will provide wrong data.

There are continuous and single updates.

`CMDeviceMotion` object combine all the sensors

Motion Activity Manager

to detect the activity (running, walking, etc.)

`CMPedometer`

to count steps, step size, pacing, etc.

To recognise ,multiple gesture you have a protocol.

Sometime gestures get in conflict. You can set a priority using a delegate method.

Apple watch:

The apple watch app is an extension of iPhone app. They can share data. In watchOS the logic runs on the watch.

go to Gesture app..

go to Sensors app..

operation queues are on top of CGD queues

Multimedia

Capturing Video and Photos

Use UIImagePickerController
to peek image from the camera, from the photo album, from the photo library

it looks like the camera.

Your class must conform to UIImagePickerControllerDelegate

Playing Video

import AVKit

AVPlayerViewController
AVPlayer objects controls the video playing

It doesn't work for Youtube videos. The support is available from Google, not from Apple.

QuickTimeStreamingServer is a server that processes all video for you !! Look for it in Apple developer side

Player

You can do everything for video and audio editing, mixing, generation,

import AVFoundation

AVAudioPlayer
AVAudioPlayerDelegate

Metering

Recorder

iPod Library Access

MediaPlayer.framework
MPMusicPlayerController

It provides a peeker to get songs and create playlists
(MPMediaPlayerPeekerController)

+iPodMusicPlayer

UIKit..UI in C

->Foundation..NS in C (AVFoundation..AV)->(CoreVideo, CoreMedia,
CoreAudio)

->CoreFoundation CF in Obj.C (Core Graphics..CG)

a foundation has no UI

when you use AVFoundation and want to display video use
QuartzCore is CoreAnimation, they can access the GPU

Metal allows you to send to the GPU shaders or programs (small pieces of
code)

Metal is like OpenGL

You can combine Metal with AVFoundation.

CoreImage is for image processing (e.g. to add filter effects on the captured
video)

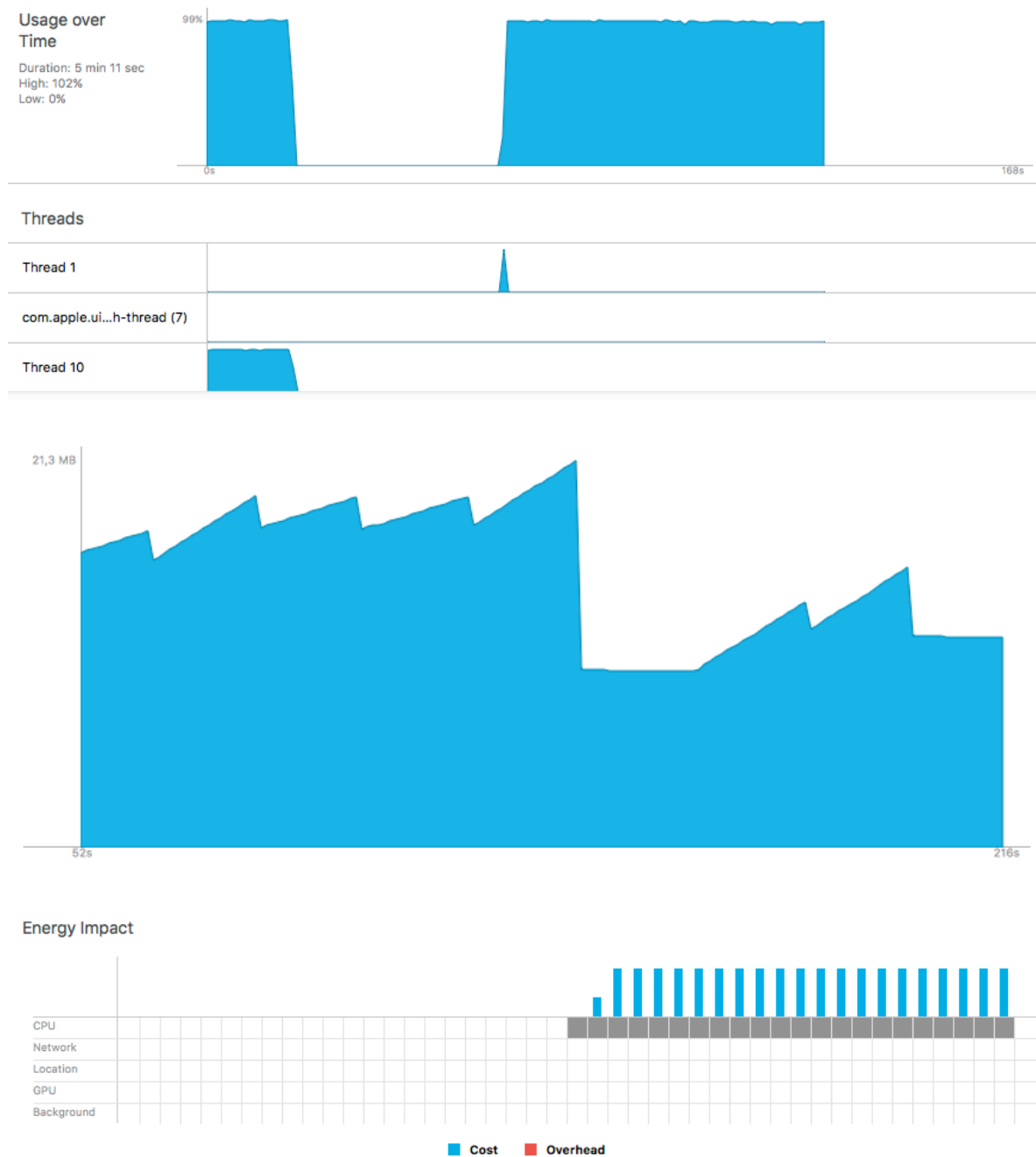
Accelerate is a framework that allows you to use Neon coprocessor inside the
ARM.

Accelerate works for the CPU, i.e. CoreImage, not Metal.

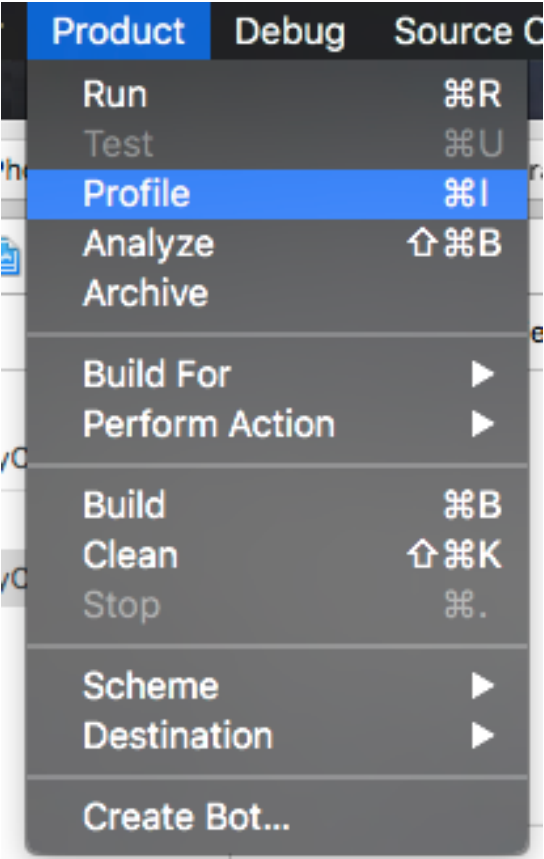
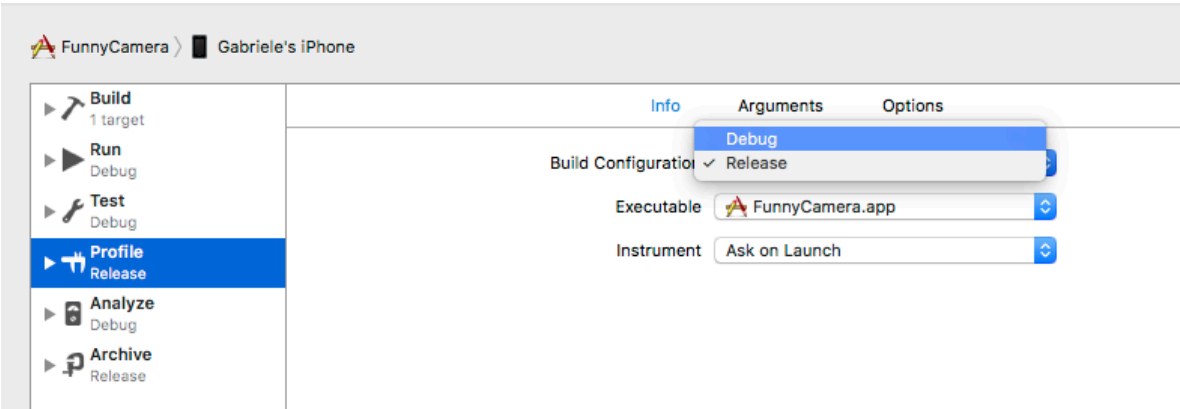
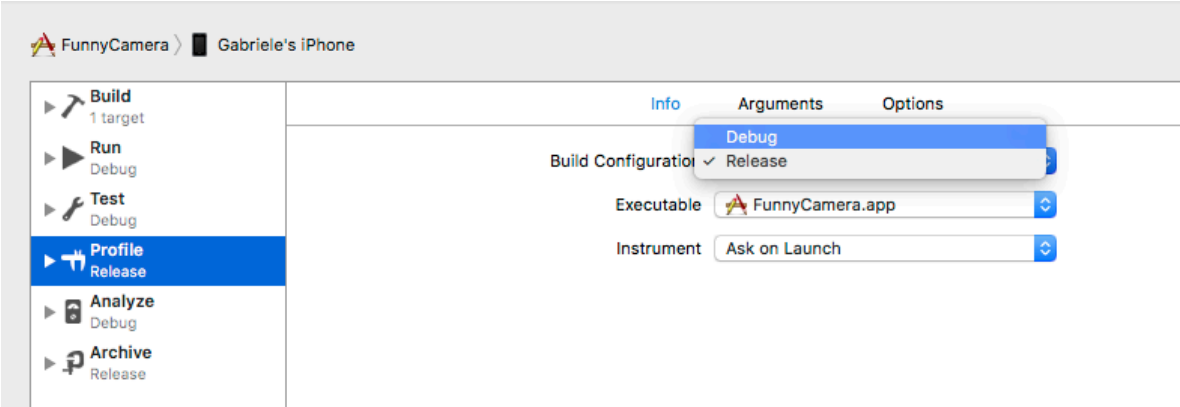
Instruments

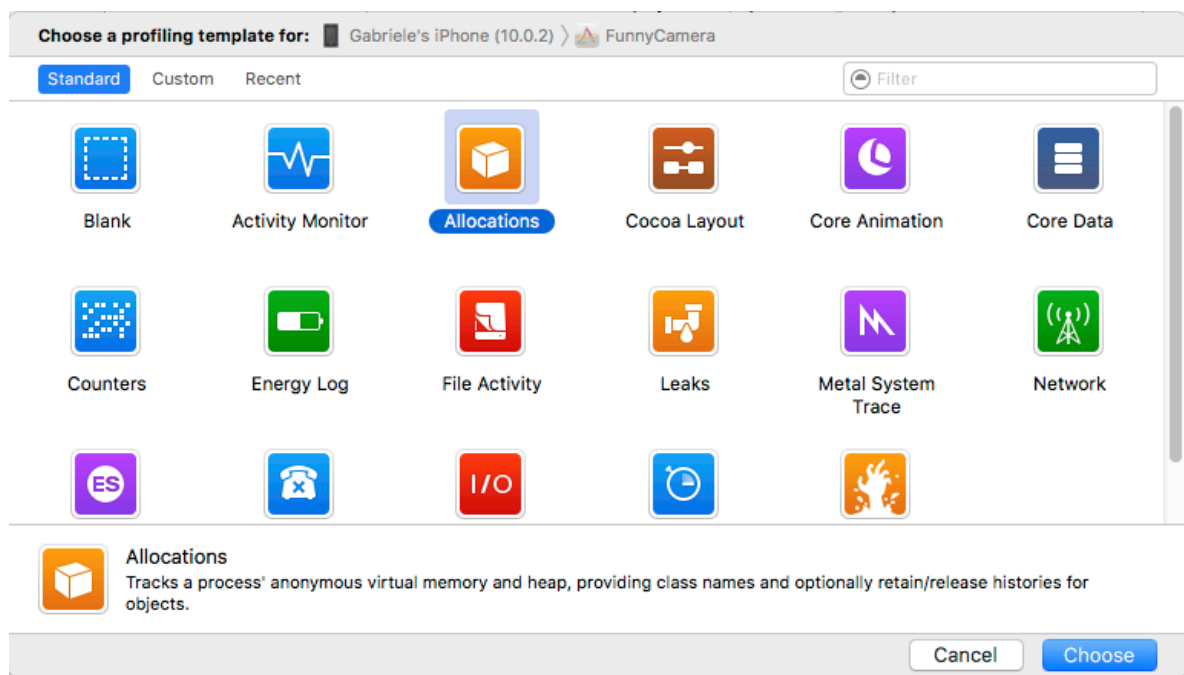
Use this tool often.

go to FunnyCammera app..

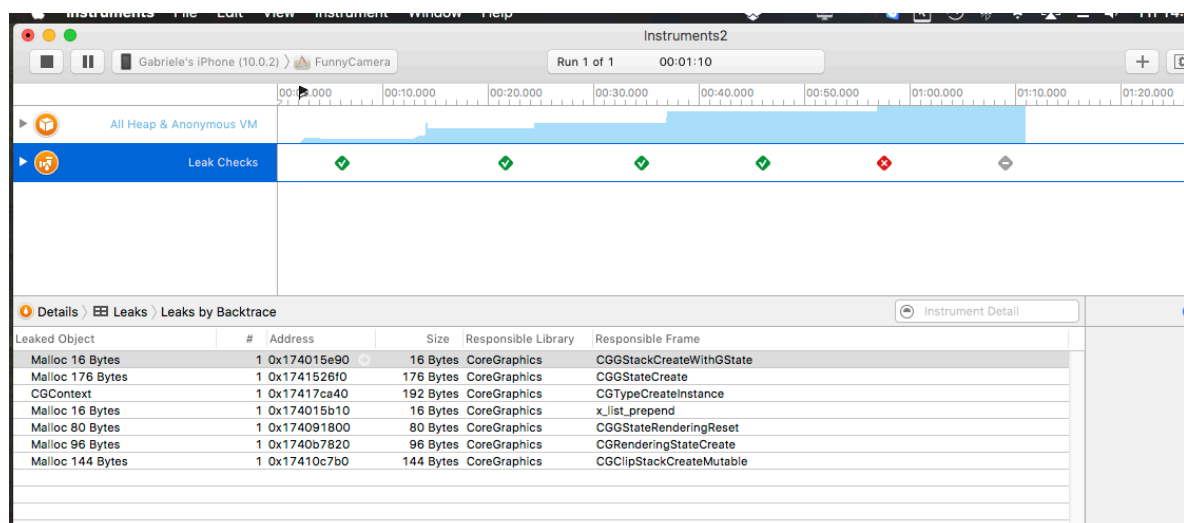


Stop the app,
run the app on the device in release mode.
For memory analysis we can run the debug mode





choose Leaks.
Unlock the device
Run



Stop.
Start the analysis

All Heap & Anonymous VM					
Leak Checks		<div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> </div>			

Details > Leaks > Leaks by Backtrace > 0x17417ca40 History						
#	Event Type	Δ RefCt	RefCt	Timestamp	Responsible Library	Responsible Caller
0	Malloc	+1	1	00:56.929.211	CoreGraphics	CGTypeCreateInstance

Enable the following check boxes

Stack Trace

0

malloc_zone_malloc

libsystem_malloc.dylib

CFRuntimeCreateInstance

CoreFoundation

CGTypeCreateInstance

CoreGraphics

CGContextCreateWithDelegat...

CoreGraphics

cgbitmap_context_create

CoreGraphics

CGBitmapContextCreateWith...

CoreGraphics

CGBitmapContextCreate

CoreGraphics

UIViewController captureOutp...

Users/gabrielefi... Controller.m:105

AVCaptureVideoDataOutput...

AVFoundation

47-[AVCaptureVideoDataOu...

AVFoundation

10

FigRemoteOperationReceiv...

✓ Invert Stack

✓ Source Location

✓ Library Name

✓ Frame #

✓ File Icon

Trace Call Duration

Lookup API Documentation

Copy Selected Frames

Locate dSYM...

Download dSYM

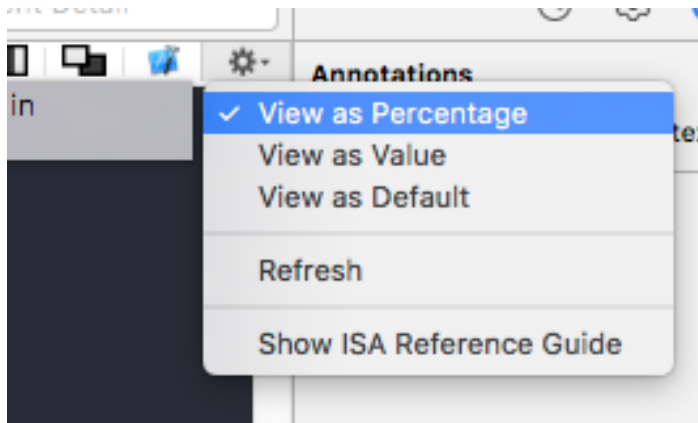
Load dSYM

```

100 const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
101 //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotFile);
102 processingImage(inImage, &outImage, 79, 79);
103
104 CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
105 CGContextRef context = CGBitmapContextCreate(outImage.data, width, height, 0, bytesPerRow, grayColorSpace,
106 kCGBitmapByteOrderDefault);
107 CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);
108
109 dispatch_sync(dispatch_get_main_queue(), ^{
110     [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
111 });

```

7 Bytes

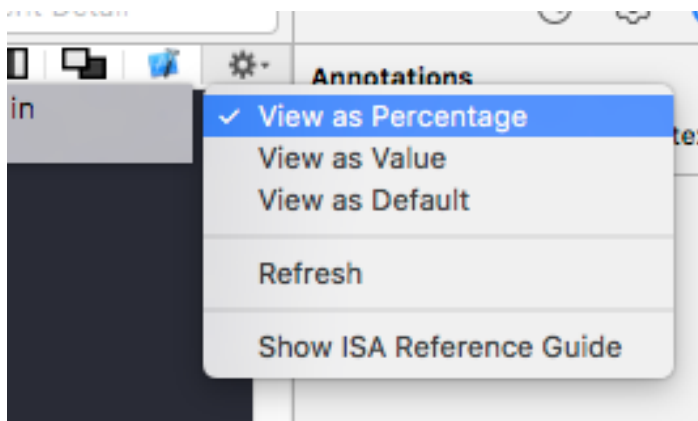


```

100 //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotFile);
101 processingImage(inImage, &outImage, 79, 79);
102
103 CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
104 CGContextRef context = CGBitmapContextCreate(outImage.data, width, height, 8, bytesPerRow, grayColorSpace,
105 kCGBitmapByteOrderDefault);
106 CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);
107
108 dispatch_sync(dispatch_get_main_queue(), ^{
109     [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
110 });

```

ARC doesn't understand C, only Obj-C and Swift. You have to release objects manually

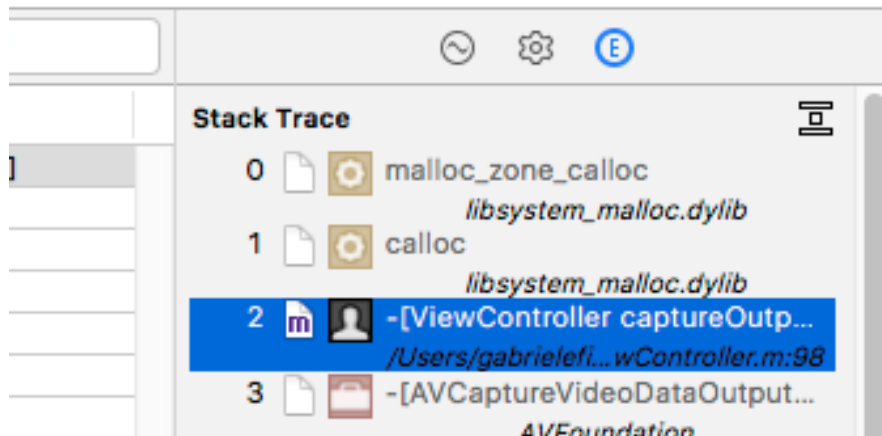


```

103
104 CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
105 CGContextRef context = CGBitmapContextCreate(outImage.data, width, height, 8, bytesPerRow, grayColorSpace,
106 kCGBitmapByteOrderDefault);
107 CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);
108
109 dispatch_sync(dispatch_get_main_queue(), ^{
110     [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
111 });
112 CGContextRelease(context);

```

Stop and restart the Instrument, and check there are other leaks. Repeat this until you don't have any leaks.



```

96     const vImage_Buffer inImage = { lumaBuffer, height, width, bytesPerRow };
97
98     Pixel_8 *outBuffer = (Pixel_8 *)calloc(width*height, sizeof(Pixel_8));
99     const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
100

```

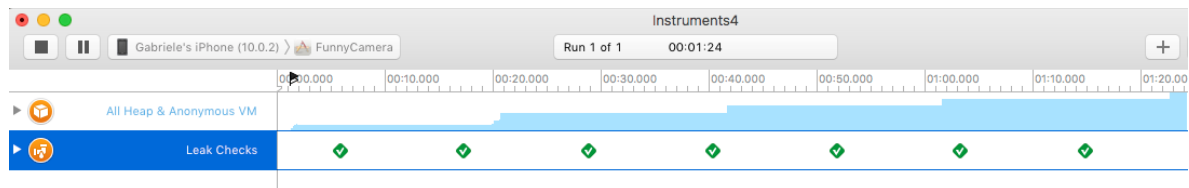
We have to free a memory allocation. Then add the free.

```

112     CGContextRelease(context);
113     free(outBuffer);
114

```

Quit and relaunch instrument.



No leaks!

We have two kind of memory problems.

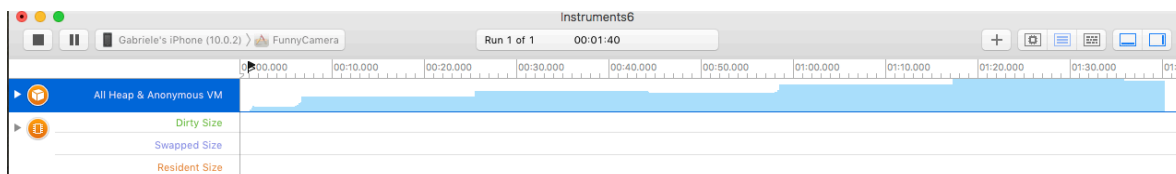
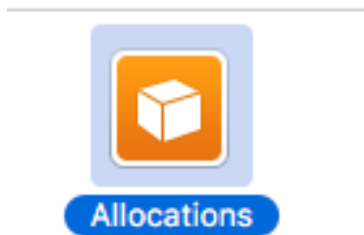
Memory leaks:

An app is a collection of connected object which live in memory. A memory leak is when you break a reference to an object, but you don't delete the object. see Retain Cycles!

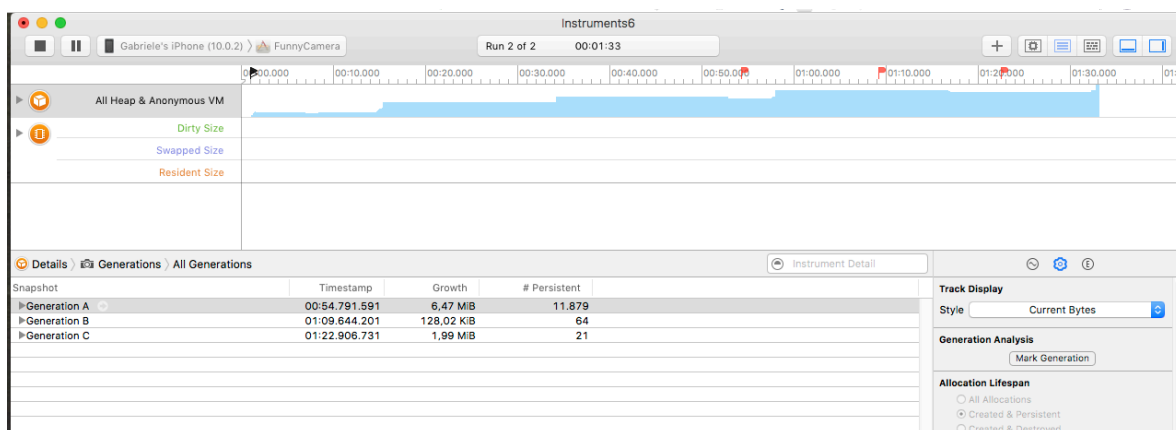
Abandoned memory:

the memory is still marked used, and the application cannot get anymore because the object was not released.

Another verification: after the memory usage doesn't return to the same level there are problems again



The tool works by getting the difference between snapshot of the heap memory.



Details > Generations > All Generations			
Snapshot	Timestamp	Growth	# Persistent
▶ Generation A	00:54.791.591	6,47 MiB	11.879
▶ Generation B	01:09.644.201	128,02 KiB	64
▶ Generation C	01:22.906.731	1,99 MiB	21
▶ Generation D	01:59.732.304	2,03 MiB	153

Details > Generations > All Generations > Generation D			
Bytes Used		Count	Symbol Name
2.00 MB	98.7%	9	▶ start_wqthread libsystem_pthread.dylib
25.55 KB	1.2%	140	▶ start libdyld.dylib
304 Bytes	0.0%	4	▶ thread_start libsystem_pthread.dylib

Details Generations All Generations Generation D			Involves Symbol	Heaviest Stack Trace	
Bytes Used	Count	Symbol Name			
2.00 MB 98.7%	9	start_wqthread libsystem_pthread.dylib	19	2.00 MB	start_wqthread
2.00 MB 98.7%	9	_pthread_wqthread libsystem_pthread.dylib	18	2.00 MB	_pthread_wqthread
2.00 MB 98.7%	9	_dispatch_worker_thread3 libdispatch.dylib	17	2.00 MB	_dispatch_worker_thread3
2.00 MB 98.7%	9	_dispatch_root_queue_drain libdispatch.dylib	16	2.00 MB	_dispatch_root_queue_drain
1.99 MB 97.9%	7	_dispatch_queue_invoke libdispatch.dylib	15	1.99 MB	_dispatch_queue_invoke
1.99 MB 97.9%	7	_dispatch_queue_serial_drain libdispatch.dylib	14	1.99 MB	_dispatch_queue_serial_drain
1.99 MB 97.9%	7	_dispatch_source_invoke libdispatch.dylib	13	1.99 MB	_dispatch_source_invoke
1.99 MB 97.9%	5	_dispatch_continuation_pop libdispatch.dylib	12	1.99 MB	_dispatch_continuation_pop
1.99 MB 97.9%	5	_dispatch_client_callout libdispatch.dylib	11	1.99 MB	_dispatch_client_callout
1.99 MB 97.9%	5	_FigRemoteQueueReceiverSetHandler_block_invoke.2 CoreMedia	10	1.99 MB	_FigRemoteQueueReceiverSetHandler_block_invoke.2
1.99 MB 97.9%	5	_FigRemoteOperationReceiverCreateMessageReceiver_block_invoke CoreMedia	9	1.99 MB	_FigRemoteOperationReceiverCreateMessageReceiver_block_invoke
1.99 MB 97.9%	5	_47-[AVCaptureVideoDataOutput _updateRemoteQueue:]_block_invoke CoreMedia	8	1.99 MB	_47-[AVCaptureVideoDataOutput _updateRemoteQueue:]_block_invoke
1.99 MB 97.9%	5	-[AVCaptureVideoDataOutput _handleRemoteQueueOperation:] AVFoundation	7	1.99 MB	-[AVCaptureVideoDataOutput _handleRemoteQueueOperation:]
1.99 MB 97.9%	5	-[ViewController captureOutput:didOutputSampleBuffer:fromConnection:] CoreGraphics	6	1.99 MB	-[ViewController captureOutput:didOutputSampleBuffer:fromConnection:]
64 Bytes 0.0%	2	_dispatch_continuation_pop libdispatch.dylib	5	1.99 MB	_dispatch_continuation_pop
17.50 KB 0.8%	2	_dispatch_queue_override_invoke libdispatch.dylib	4	1.99 MB	_dispatch_queue_override_invoke
25.55 KB 1.2%	140	_start libdyld.dylib	3	1.99 MB	_start
304 Bytes 0.0%	4	_thread_start libsystem_pthread.dylib	2	1.99 MB	_thread_start

```

96     const vImage_Buffer inImage = { lumaBuffer, height, width, bytesPerRow };
97     Pixel_8 *outBuffer = (Pixel_8 *)calloc(width*height, sizeof(Pixel_8));
98     const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
99
100     //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
101     processingImage(inImage, &outImage, 79, 79);
102
103     CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
104     CGContextRef context = CGBitmapContextCreate(outImage.data,
105                                                  width, height, 8, bytesPerRow, grayColorSpace, kCGBitmapByteOrderDefault);
106     CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);
107
108     dispatch_sync(dispatch_get_main_queue(), ^{
109         [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
110     });

```

```

112     CGContextRelease(context);
113     free(outBuffer);
114     CGImageRelease(dstImageFilter);

```

Done.

```

112     CGContextRelease(context);
113     free(outBuffer);
114     CGImageRelease(dstImageFilter);
115     CGColorSpaceRelease(grayColorSpace);

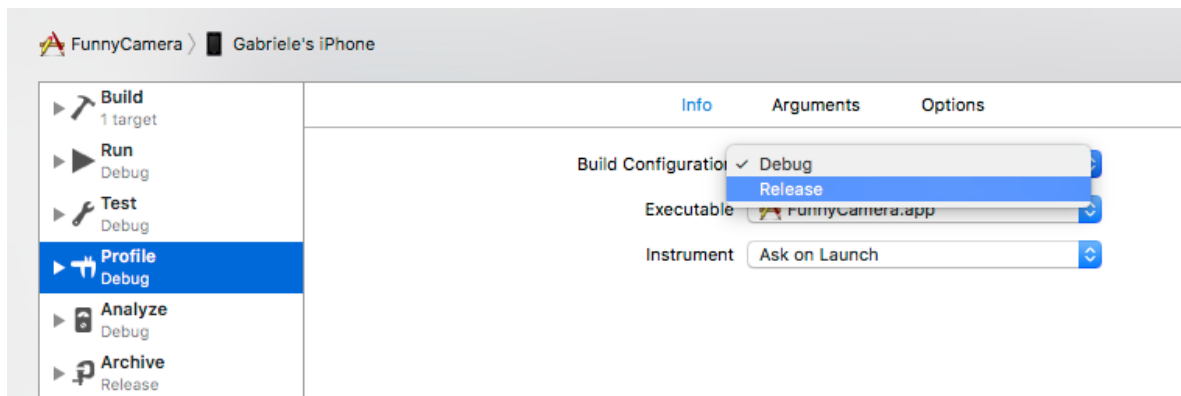
```

Memory

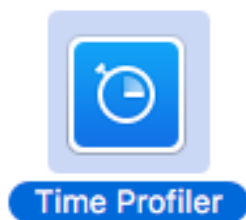
Duration: 1 min 39 sec
High: 11,4 MB
Low: Zero KB



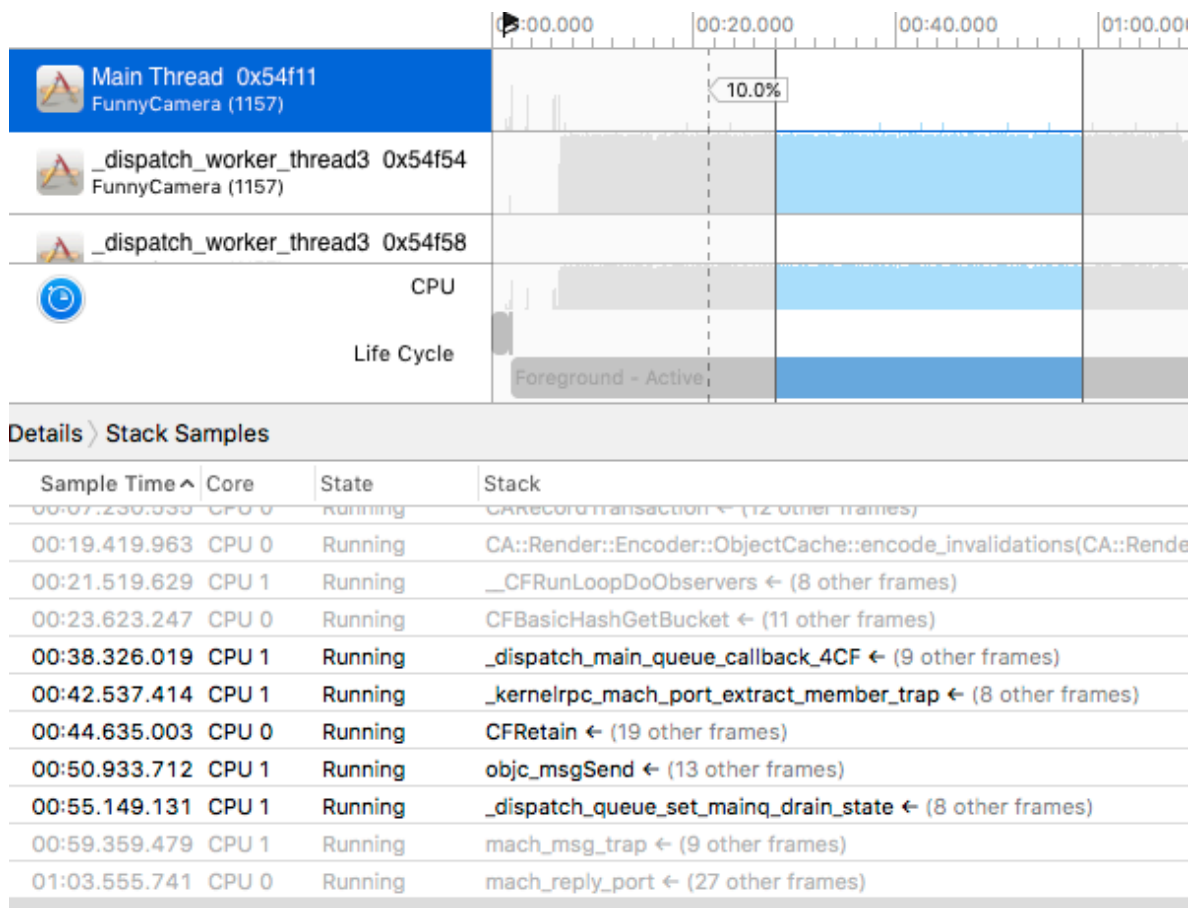
Now we have to fix CPU usage.
Set the project in release mode.



This time we use the Time Profiler



Every 10 ns instruments collects the stack trace and performs some computation.



After a minute or so, stop recording and find out the methods which is consuming more CPU time

Weight	Self Weight	Symbol Name
3.14 min 100.0%	0 s	▼FunnyCamera (1157) +
1.93 min 61.6%	0 s	▼_dispatch_worker_thread3 0x54f55 +
1.93 min 61.6%	0 s	▼_pthread_wqthread libsystem_pthread.dylib
1.93 min 61.6%	0 s	▼_dispatch_kevent_worker_thread libdispatch.dylib
1.93 min 61.6%	0 s	▼_dispatch_root_queue_drain_deferred_item libdispatch.dylib
1.93 min 61.6%	0 s	▼_dispatch_queue_invoke libdispatch.dylib
1.93 min 61.6%	0 s	▼_dispatch_queue_serial_drain libdispatch.dylib
1.93 min 61.6%	0 s	▼_dispatch_queue_invoke libdispatch.dylib
1.93 min 61.6%	0 s	▼_dispatch_queue_serial_drain libdispatch.dylib
1.93 min 61.6%	0 s	▶_dispatch_source_invoke libdispatch.dylib
1.00 ms 0.0%	0 s	▶_dispatch_client_callout libdispatch.dylib
1.00 ms 0.0%	0 s	▶_dispatch_source_invoke libdispatch.dylib
15.00 ms 0.0%	0 s	▶_dispatch_worker_thread3 libdispatch.dylib
1.20 min 38.2%	0 s	▶_dispatch_worker_thread3 0x54f54
296.00 ms 0.1%	0 s	▶Main Thread 0x54f11

In the Extended Details perspective you will find the reference to the source code


```

80 - (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer fromConnection:
    (AVCaptureConnection *)connection
81 {
82     if (!_readyForProcessing) {
83         _readyForProcessing = NO;
84
85         CVImageBufferRef imageBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);
86         CVPixelBufferLockBaseAddress(imageBuffer, 0);
87
88         // For the iOS the luma is contained in full plane (8-bit)
89         size_t width = CVPixelBufferGetWidthOfPlane(imageBuffer, 0);
90         size_t height = CVPixelBufferGetHeightOfPlane(imageBuffer, 0);
91         size_t bytesPerRow = CVPixelBufferGetBytesPerRowOfPlane(imageBuffer, 0);
92
93         Pixel_8 *lumaBuffer = (Pixel_8 *)CVPixelBufferGetBaseAddressOfPlane(imageBuffer, 0);
94
95         const vImage_Buffer inImage = { lumaBuffer, height, width, bytesPerRow };
96
97         Pixel_8 *outBuffer = (Pixel_8 *)calloc(width*height, sizeof(Pixel_8));
98         const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
99
100         //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
101         processingImage(inImage, &outImage, 79, 79);
102
103         CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
104         CGContextRef context = CGContextCreateFromData(outImage.data, width, height, 8, bytesPerRow, grayColorSpace);

```

You may discover the critical piece of code. In this case is a software convolution

```

134     sum = 0;
135     for (unsigned long x = 0; x < N; x++) {
136         for (unsigned long y = 0; y < M; y++) {
137             sum += *(bitmap + (i-x)*width + (j-y));
138         }
139     }
140     transformedBitmap[i*width+j] = sum;
141 }
142 }
143 }

```

There is a framework that make you able to access the NEON coprocessor to do such kind of calculation in hardware

<pre> AppDelegate.m RootViewController.h RootViewController.m RootViewController.xib ViewController.h ViewController.m </pre>	<pre> 9 #import <UIKit/UIKit.h> 10 #import <QuartzCore/QuartzCore.h> 11 #import <CoreMedia/CoreMedia.h> 12 #import <Accelerate/Accelerate.h> 13 #import <AVFoundation/AVFoundation.h> 14 15 @protocol ViewControllerDelegate; 16 17 18 @interface ViewController : UIViewController 19 @property (nonatomic, weak) id<ViewControllerDelegate> delegate; </pre>
---	---

Uncomment the NEON function

```

99     const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
100
101     //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
102     processingImage(inImage, &outImage, 79, 79);
103
104     CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();

```

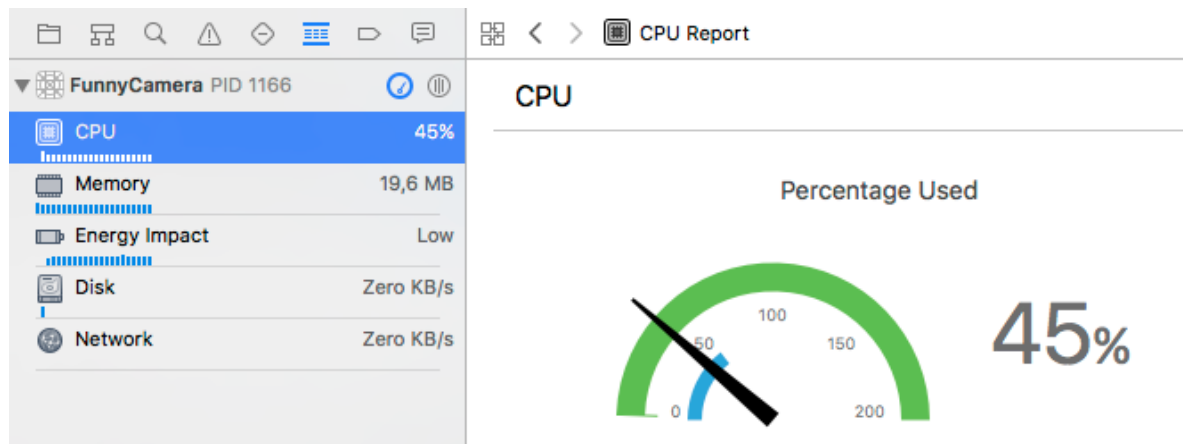
and comment out the software implementation

```

99     const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
100
101     vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
102     //processingImage(inImage, &outImage, 79, 79);
103
104     CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();

```

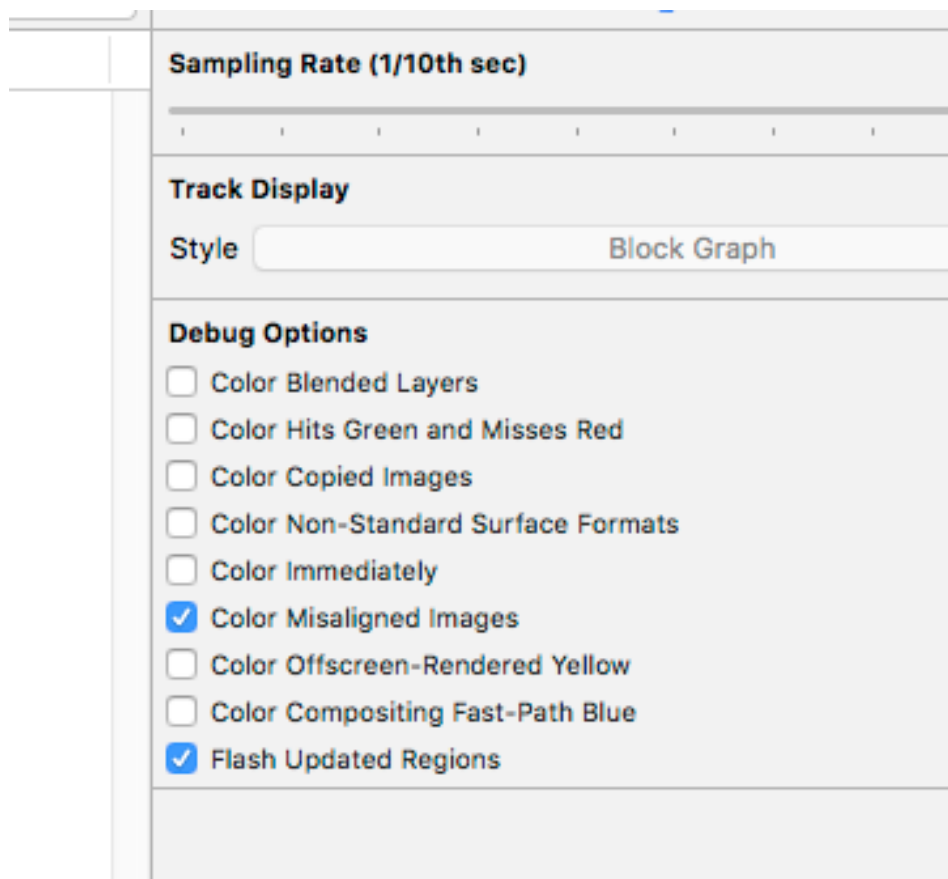
Now run the app again and you will see the difference!!



Tip: when you implement some new functionality in your app, run the Instrument to profile it.

Tip: you can run those instruments in the Continuous Integration. You can run this via the command line tool on a server.

Other tools in Instruments



Xcode, a review

Warning: to test an app on iOS 9.3 you cannot use Xcode 8.1 (shipped with

10.1). You have to use Xcode 7.3. Otherwise you will be not sure 100% it will work really.