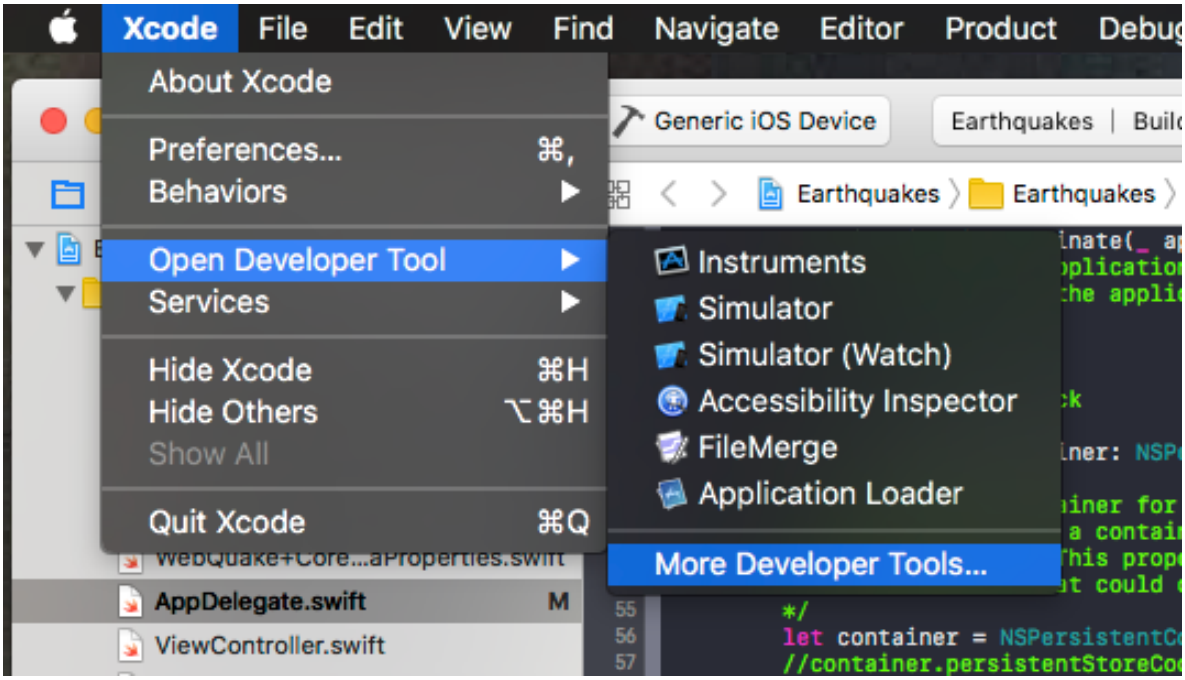
5th Day - iOS Swift Training in Barcelona

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, ..)
- Continous

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.

QuickTimeStramingServer 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
QuarzCore 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.

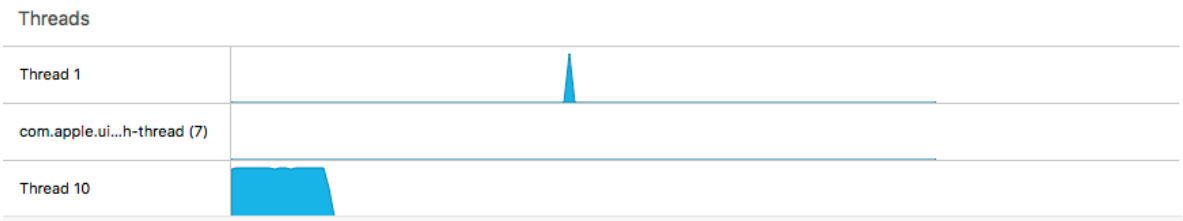
CoreImage is for image processing (e.g. to add filter effects on the captured video)

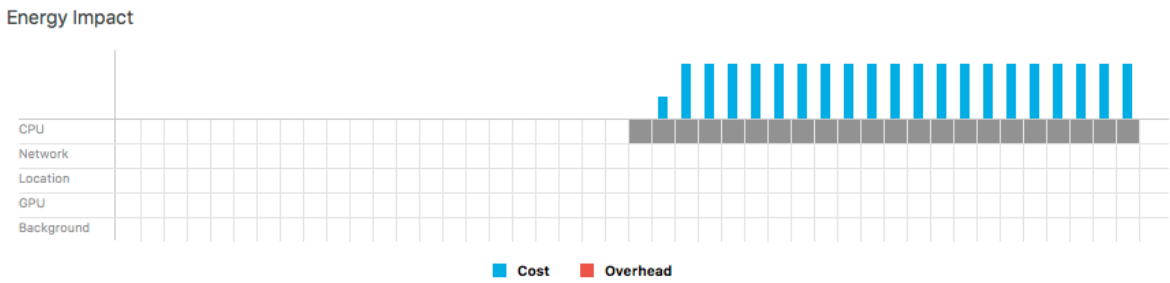
Accelerate is a framework that allows you to use Neon coprocessor inside the ARM.
Accelerate works for the CPU, i.e. CoreImage, not Metal.

Instruments

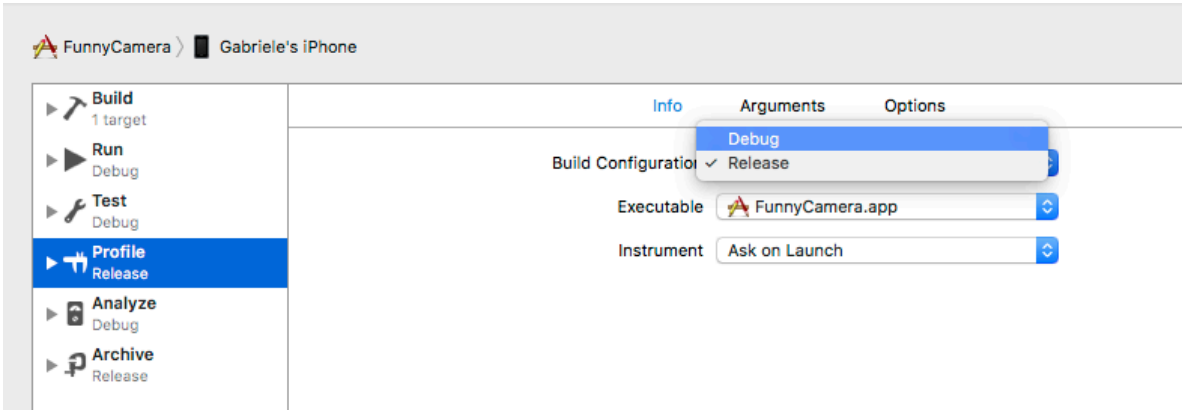
Use this tool often.

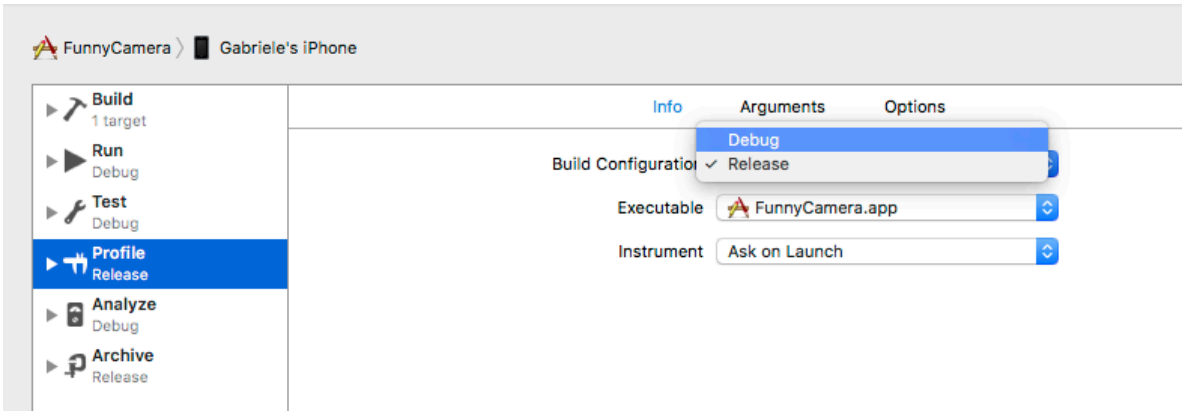
go to FunnyCammera app..

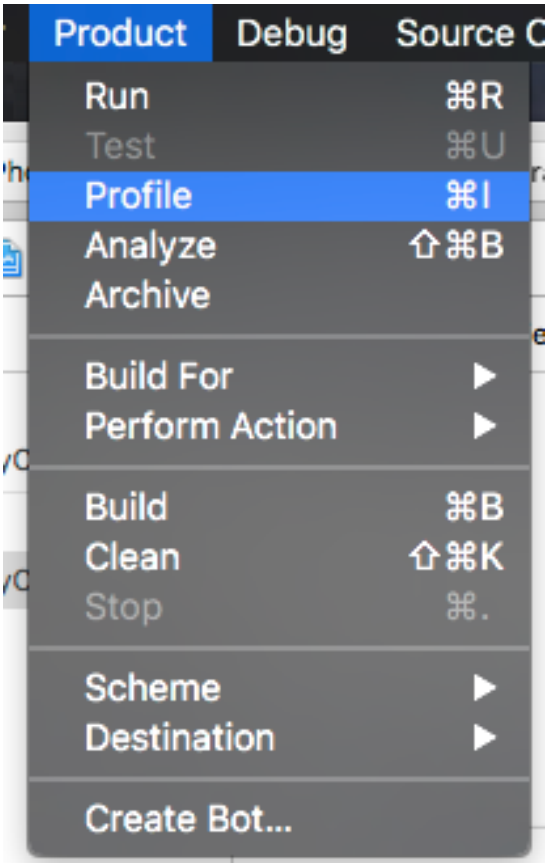


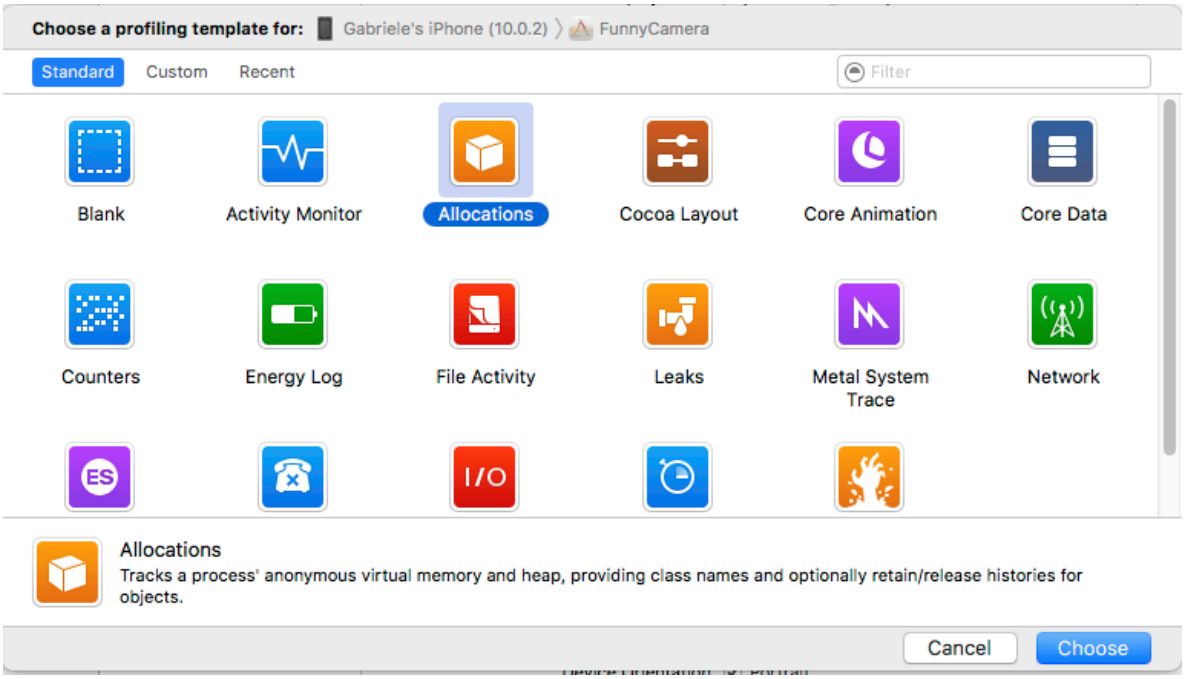


Stop the app,
run the app on the device in release mode.
For memory analysis we can run the debug mode

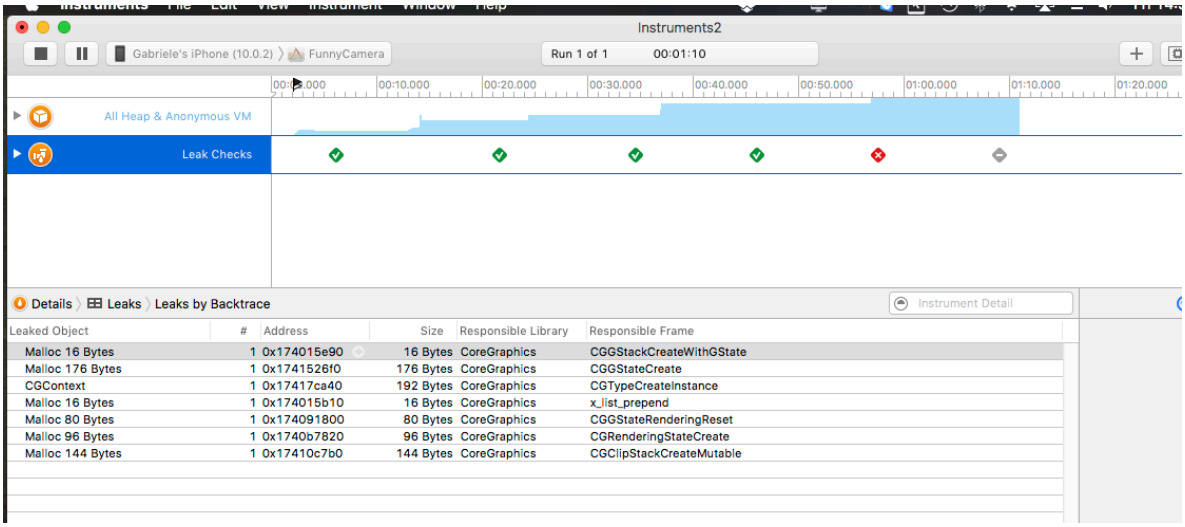

FunnyCamera
Gabriele's iPhone
Build
1 target
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Debug
Build Configuratio
Release
Executable
FunnyCamera.app
Instrument
Ask on Launch


FunnyCamera
Gabriele's iPhone
Build
1 target
Run
Debug
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Debug
Build Configuratio
Release
Executable
FunnyCamera.app
Instrument
Ask on Launch

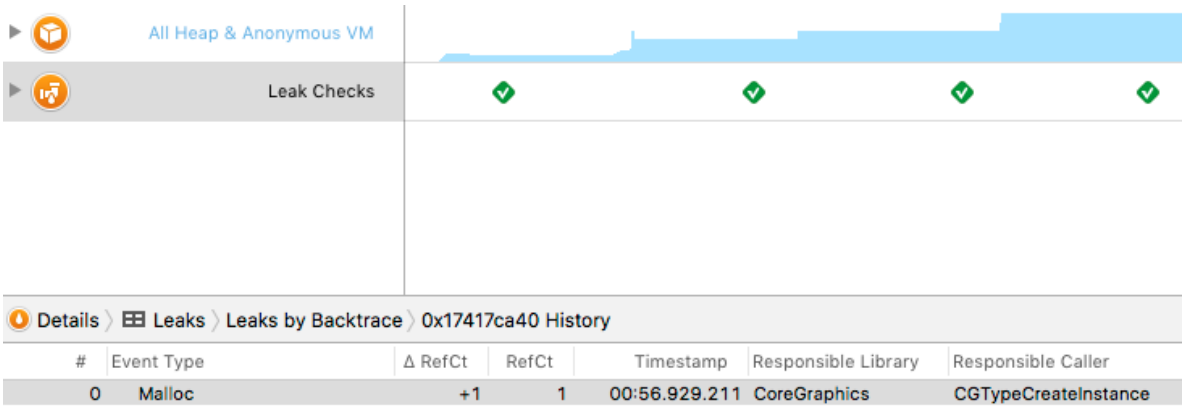

Product
Debug
Source C
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Scheme
Destination
Create Bot...



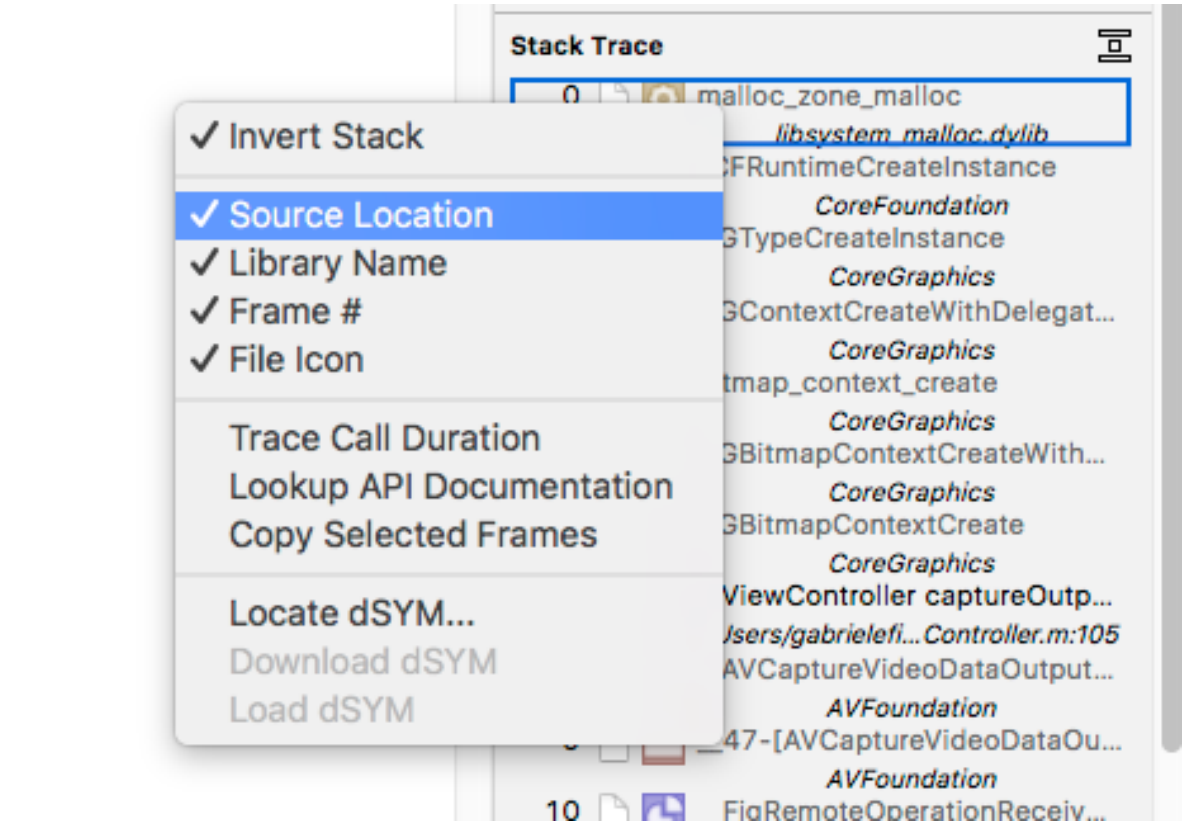
choose Leaks.
Unlock the device
Run

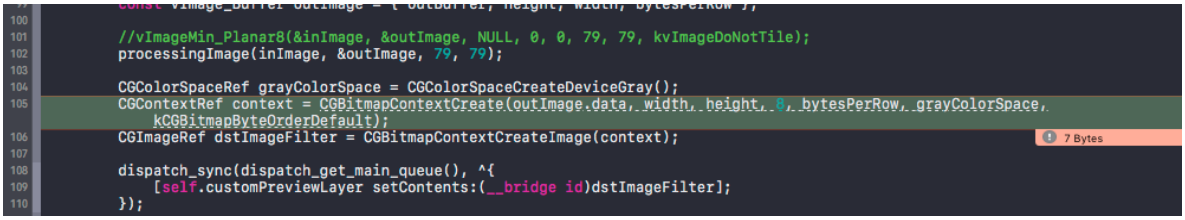


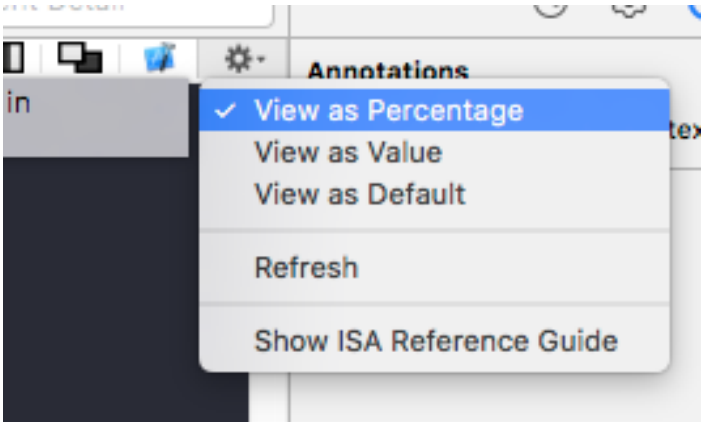
Stop.
Start the analysis



Enable the following check boxes

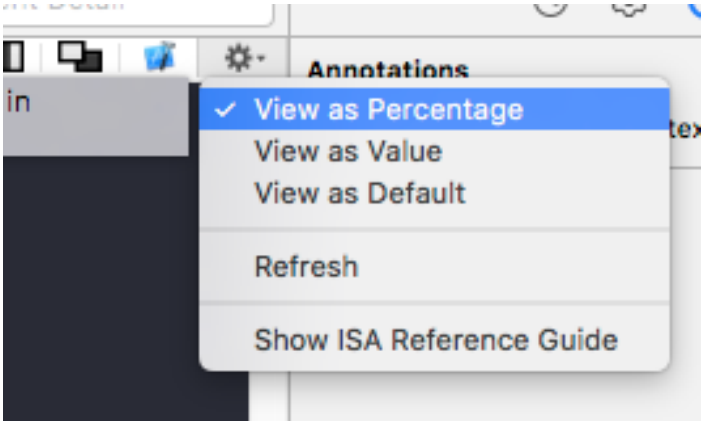






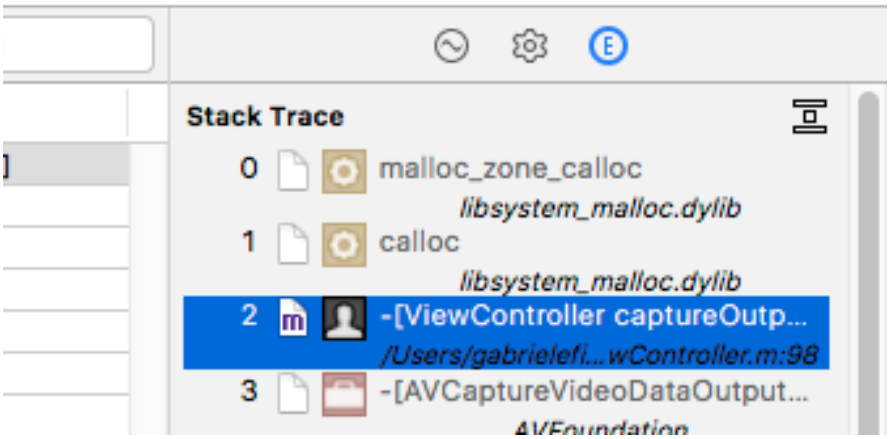

```
//vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
processingImage(inImage, &outImage, 79, 79);

CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
CGContextRef context = CGBitmapContextCreate(outImage.data, width, height, 8, bytesPerRow, grayColorSpace,
    kCGBitmapByteOrderDefault);
CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);

dispatch_sync(dispatch_get_main_queue(), ^{
    [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
});
```

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




```
CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
CGContextRef context = CGBitmapContextCreate(outImage.data, width, height, 8, bytesPerRow, grayColorSpace,
    kCGBitmapByteOrderDefault);
CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);

dispatch_sync(dispatch_get_main_queue(), ^{
    [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
});

CGContextRelease(context);
```

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

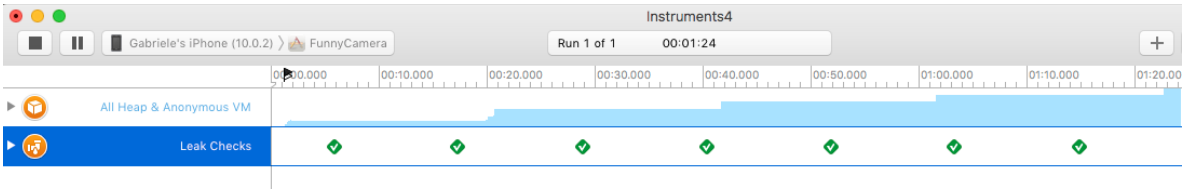



```
96      const vImage_Buffer inImage = { lumaBuffer, height, width, bytesPerRow };
97
98      Pixel_8 *outBuffer = (Pixel_8 *)calloc(width*height, sizeof(Pixel_8));   100.0%
99      const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
100
```

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

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

Quit and relaunch instrument.

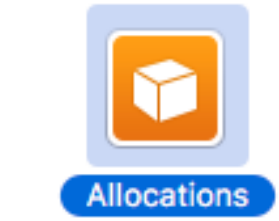


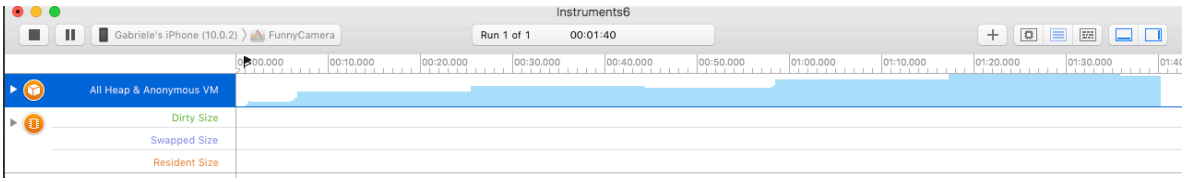
No leaks!
We have two kind of memory problems.

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!

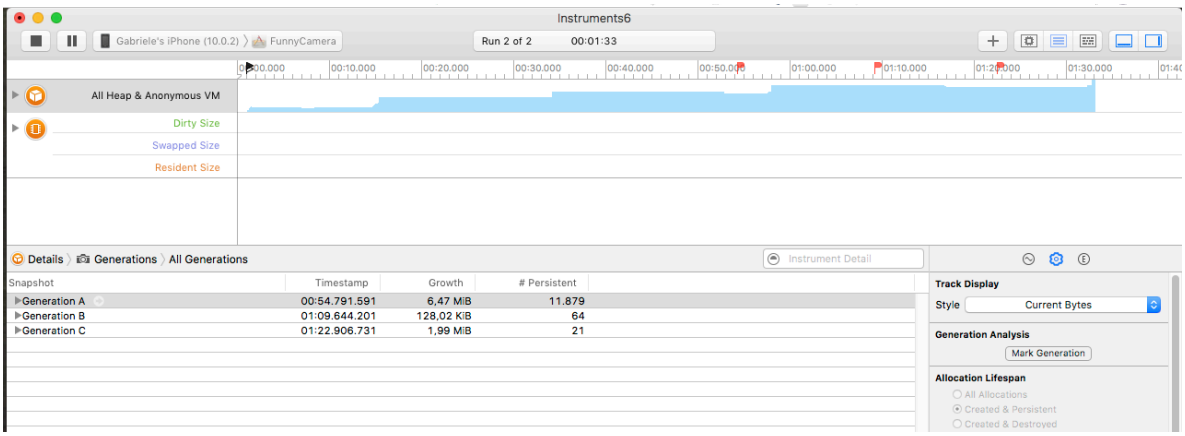
Abandoned memory:
the memory is still marked used, and the application cannot get anymore because the object was not released.

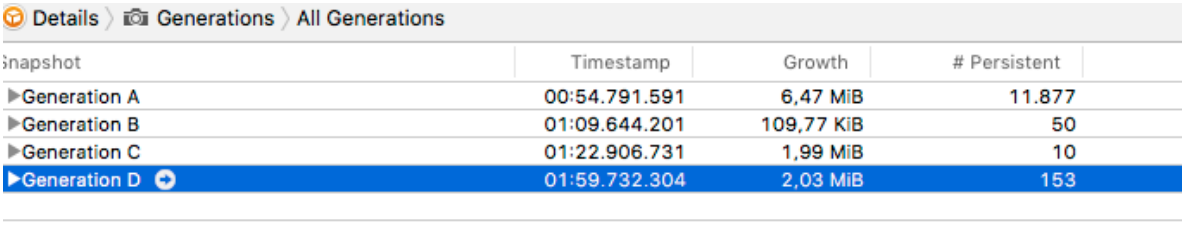
Another verification: after the memory usage doesn’t return to the same level there are problems again

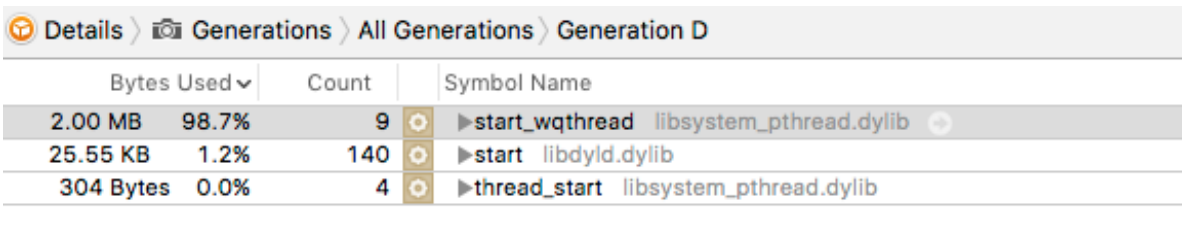


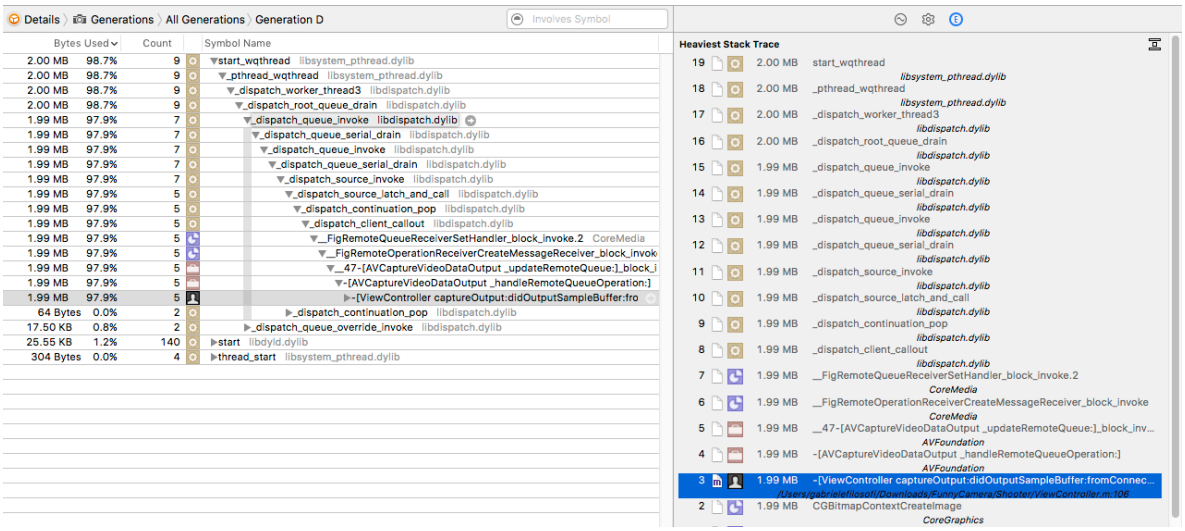


The tool works by getting the difference between snapshot of the heap memory.







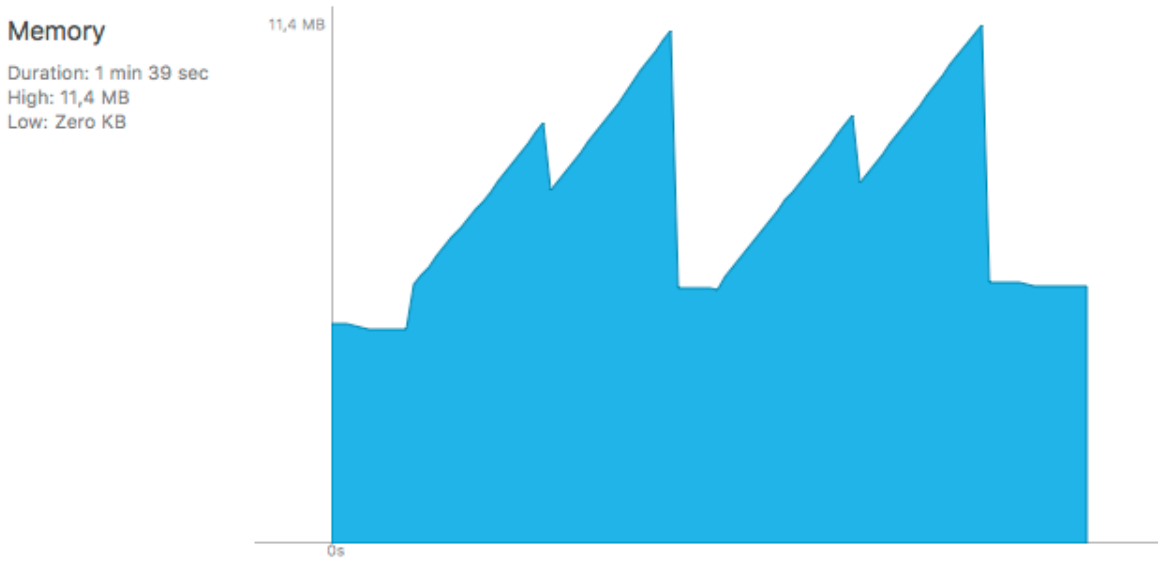



```
96          const vImage_Buffer inImage = { lumaBuffer, height, width, bytesPerRow };
97
98          Pixel_8 *outBuffer = (Pixel_8 *)calloc(width*height, sizeof(Pixel_8));   50.0%
99          const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };
100
101         //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
102         processingImage(inImage, &outImage, 79, 79);
103
104         CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
105         CGContextRef context = CGBitmapContextCreate(outImage.data,   0.1%
                width, height, 8, bytesPerRow, grayColorSpace, kCGBitmapByteOrderDefault);
106         CGImageRef dstImageFilter = CGBitmapContextCreateImage(context);   50.0%   CGBitmap
107
108         dispatch_sync(dispatch_get_main_queue(), ^{
109             [self.customPreviewLayer setContents:(__bridge id)dstImageFilter];
110         });
```

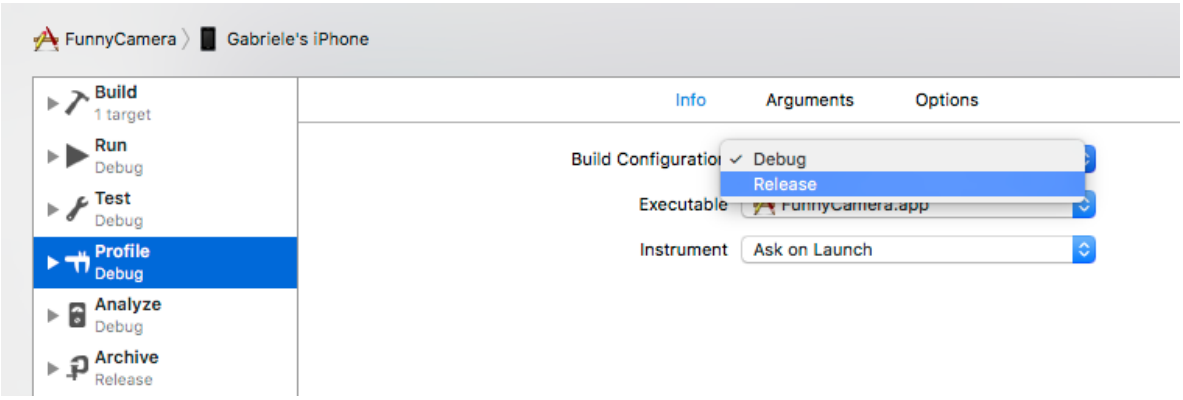
```
112         CGContextRelease(context);
113         free(outBuffer);
114         CGImageRelease(dstImageFilter);
```

Done.

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

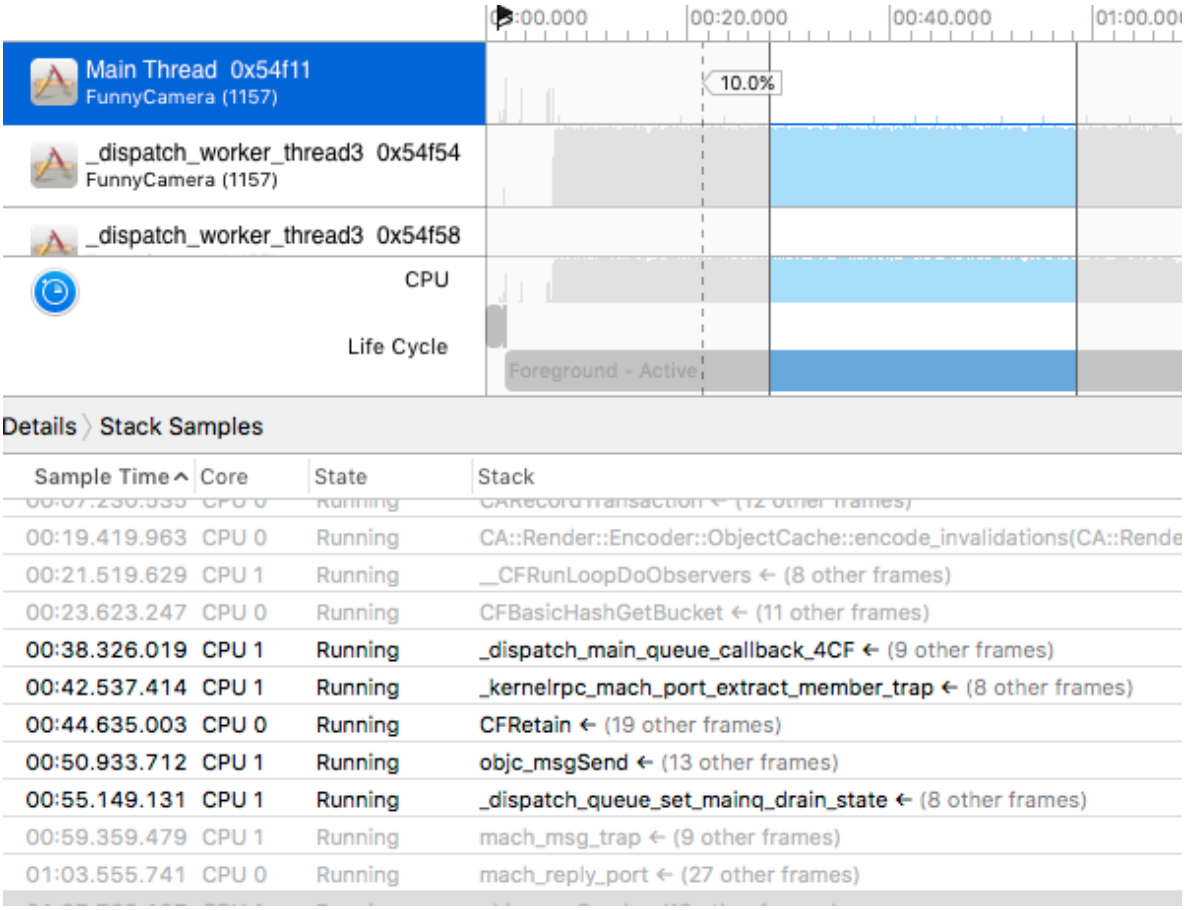


Now we have to fix CPU usage.
Set the project in release mode.

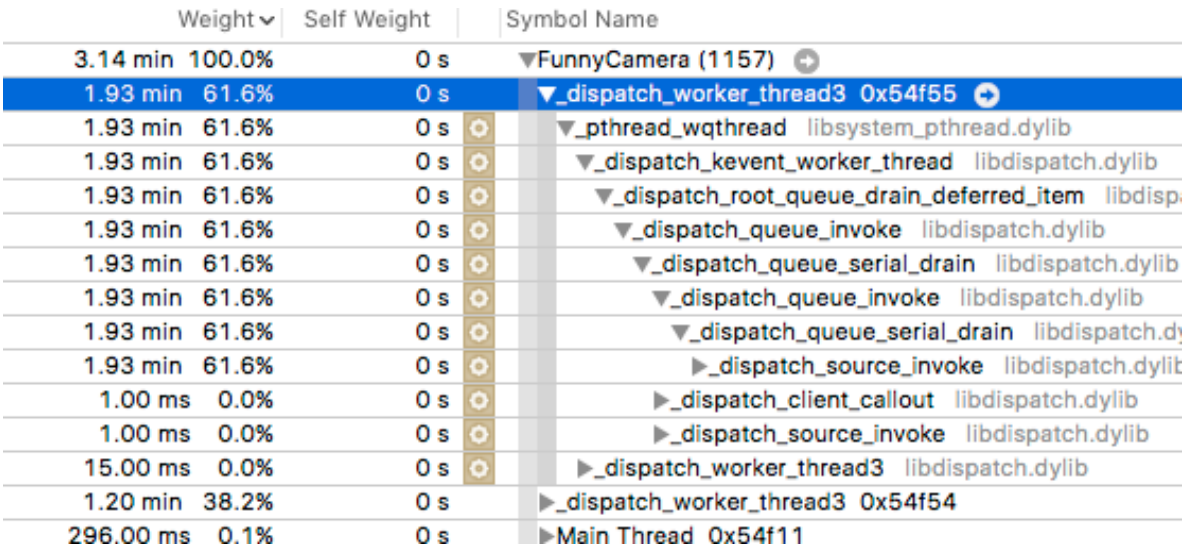


This time we use the Time Profiler

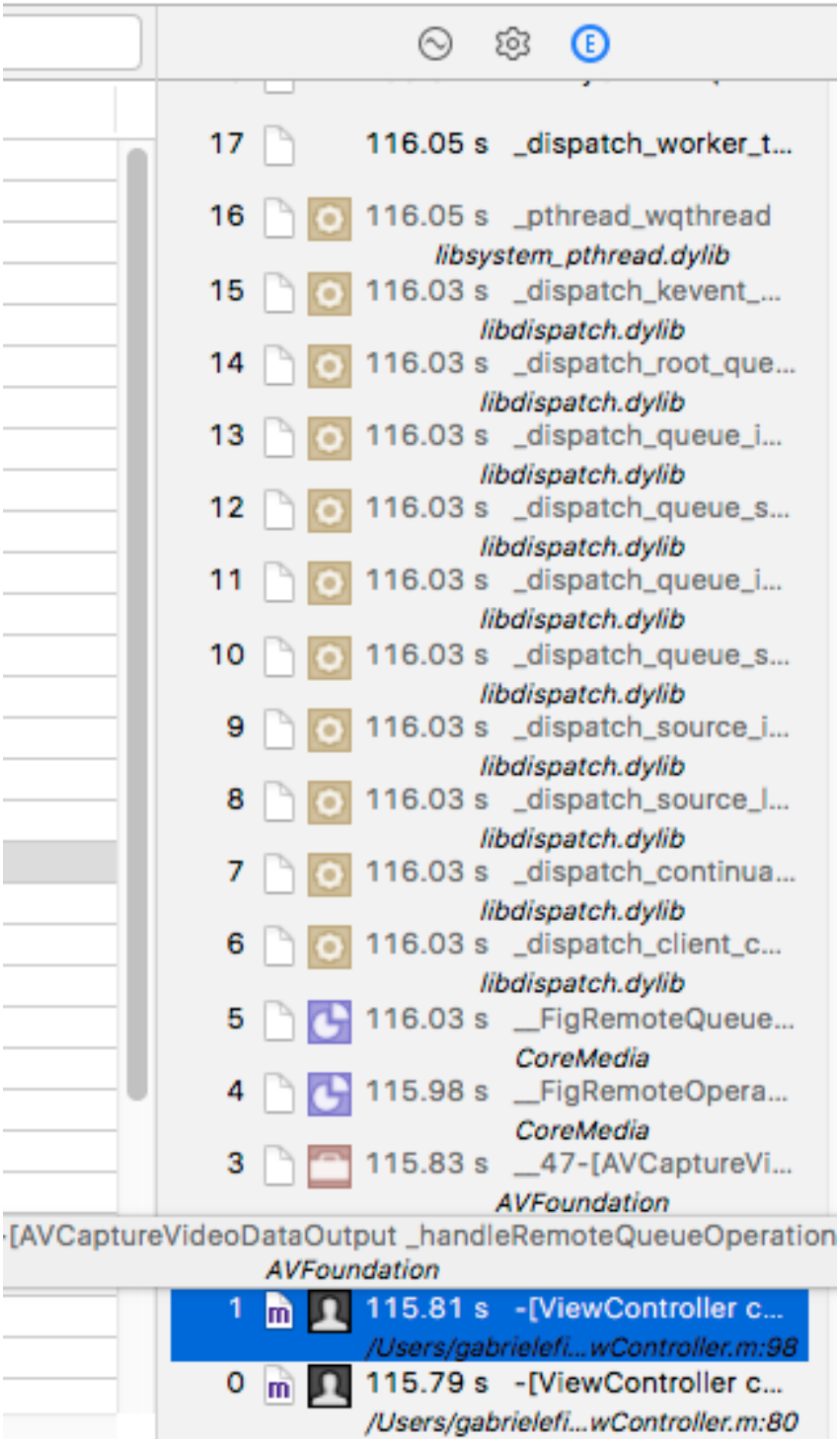
Every 10 ns instruments collects the stack trace and performs some computation.



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



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



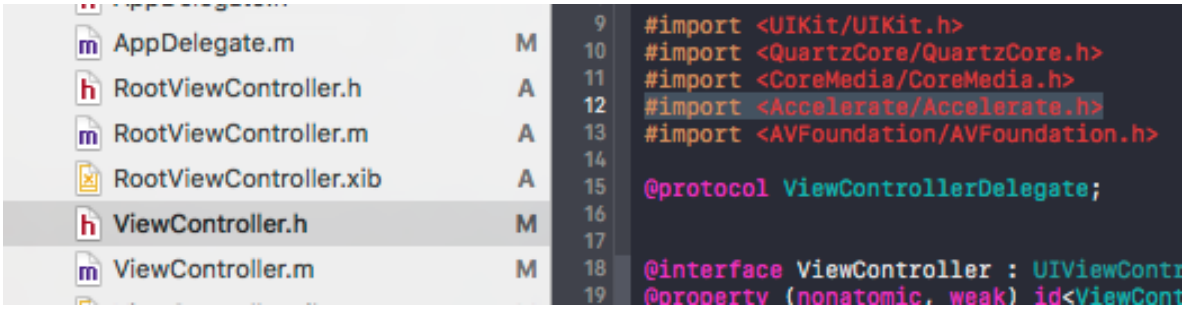
Double click it and you will get the line in source code

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer fromConnection:
    (AVCaptureConnection *)connection
{
    if (_readyForProcessing) {

        _readyForProcessing = NO;

        CVImageBufferRef imageBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);
        CVPixelBufferLockBaseAddress(imageBuffer, 0);

        // For the iOS the luma is contained in full plane (8-bit)
        size_t width = CVPixelBufferGetWidthOfPlane(imageBuffer, 0);
        size_t height = CVPixelBufferGetHeightOfPlane(imageBuffer, 0);
        size_t bytesPerRow = CVPixelBufferGetBytesPerRowOfPlane(imageBuffer, 0);

        Pixel_8 *lumaBuffer = (Pixel_8 *)CVPixelBufferGetBaseAddressOfPlane(imageBuffer, 0);

        const vImage_Buffer inImage = { lumaBuffer, height, width, bytesPerRow };

        Pixel_8 *outBuffer = (Pixel_8 *)calloc(width*height, sizeof(Pixel_8));
        const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };

        //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
        processingImage(inImage, &outImage, 79, 79);                                    74x

        CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
```

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

```
        sum = 0;
        for (unsigned long x = 0; x < N; x++) {                                         285x
            for (unsigned long y = 0; y < M; y++) {                                     28937x
                sum += *(bitmap + (i-x)*width + (j-y));                                 85394x
            }
        }
        transformedBitmap[i*width+j] = sum;                                             1096x
    }
}
```

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



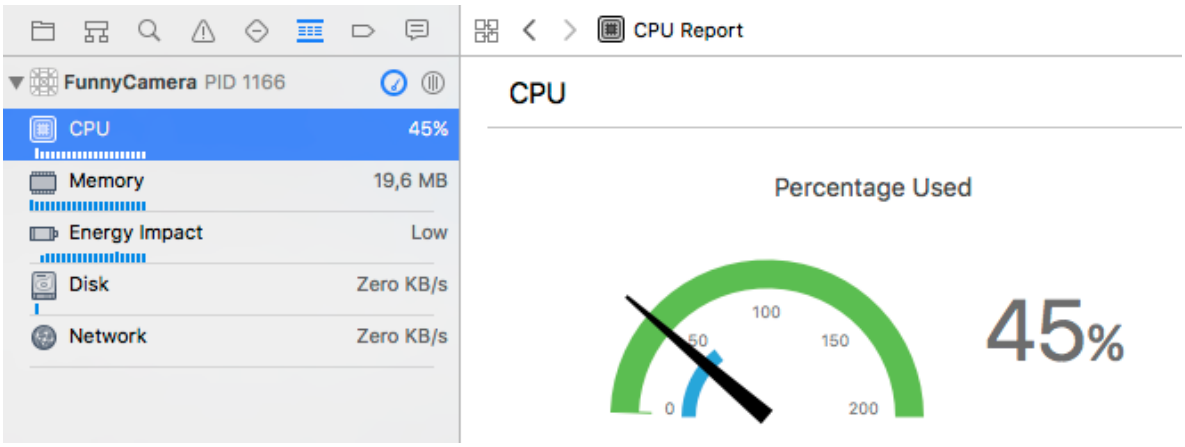
Uncomment the NEON function

```
        const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };

        //vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
        processingImage(inImage, &outImage, 79, 79);
```

and comment out the software implementation

```
        const vImage_Buffer outImage = { outBuffer, height, width, bytesPerRow };

        vImageMin_Planar8(&inImage, &outImage, NULL, 0, 0, 79, 79, kvImageDoNotTile);
        //processingImage(inImage, &outImage, 79, 79);

        CGColorSpaceRef grayColorSpace = CGColorSpaceCreateDeviceGray();
```

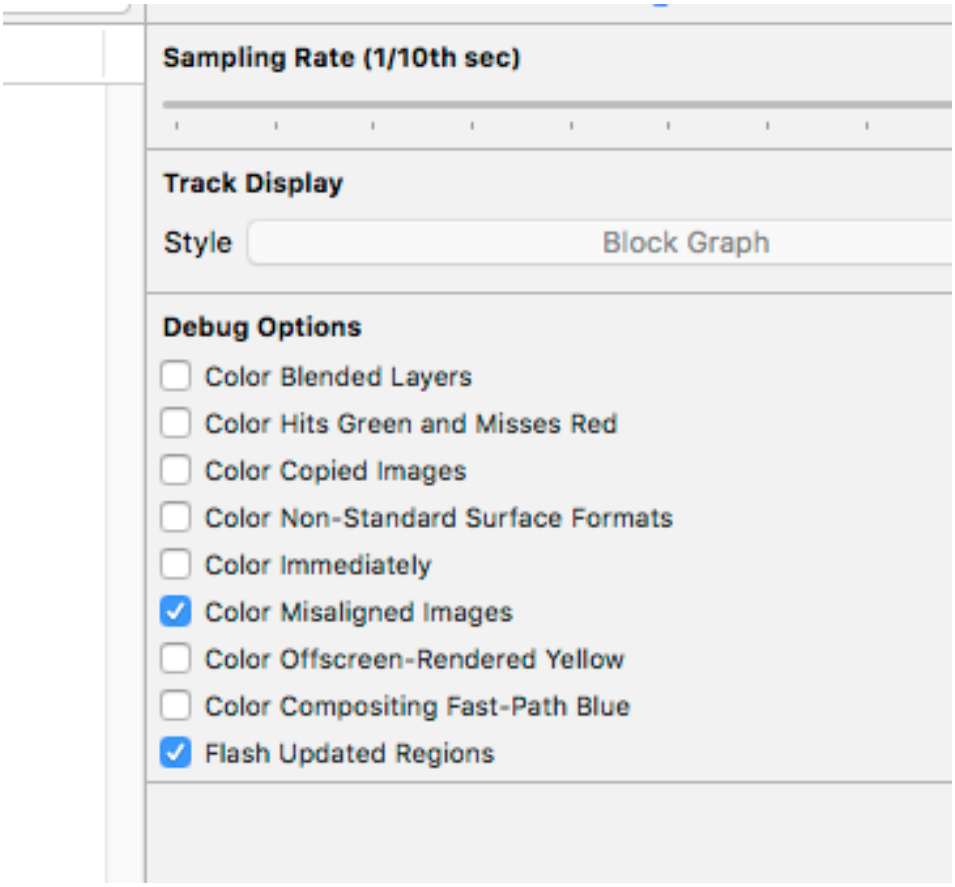
Now run the app again and you will see the difference!!



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.