

Advanced Graphics and Animations for iOS Apps

Session 419

Axel Wefers

iOS Software Engineer

Michael Ingrassia

iOS Software Engineer

What You Will Learn

Core Animation pipeline

Rendering concepts

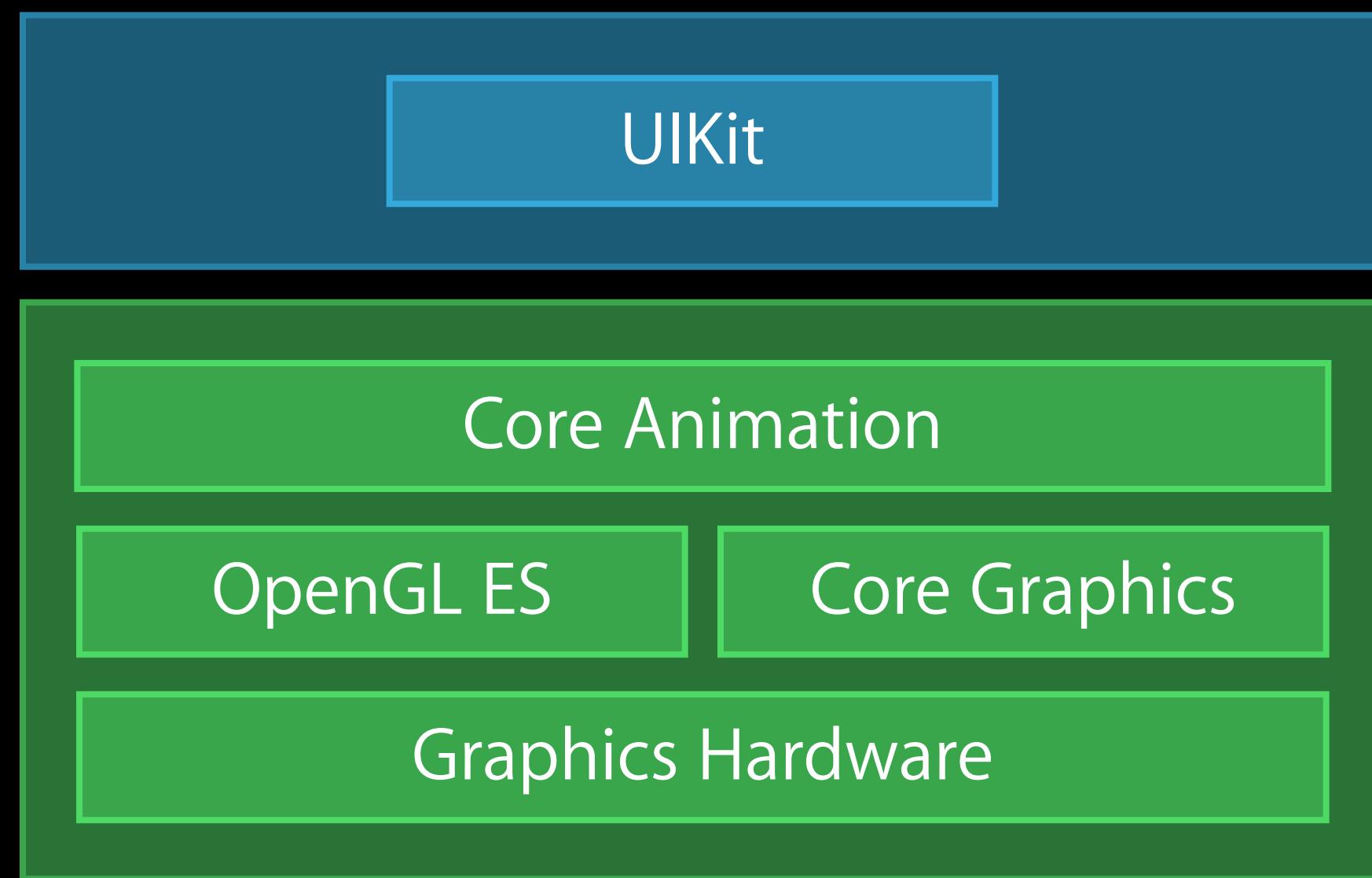
UIBlurEffect

UIVibrancyEffect

Profiling tools

Case studies

Technology Framework



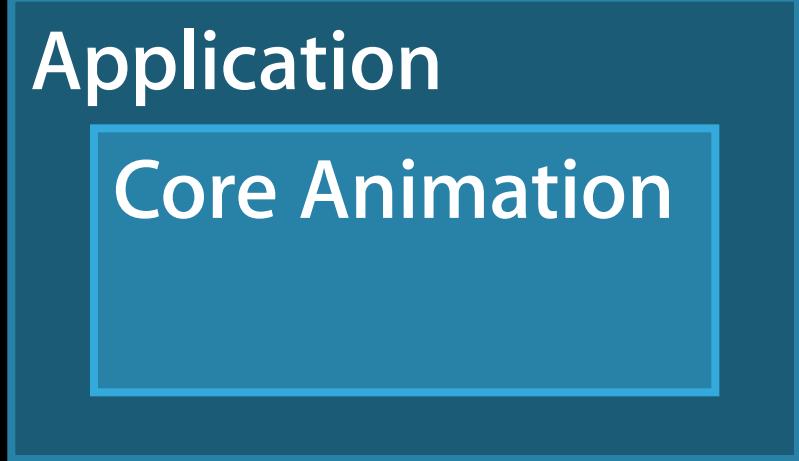
Core Animation Pipeline

Axel Wefers
iOS Software Engineer

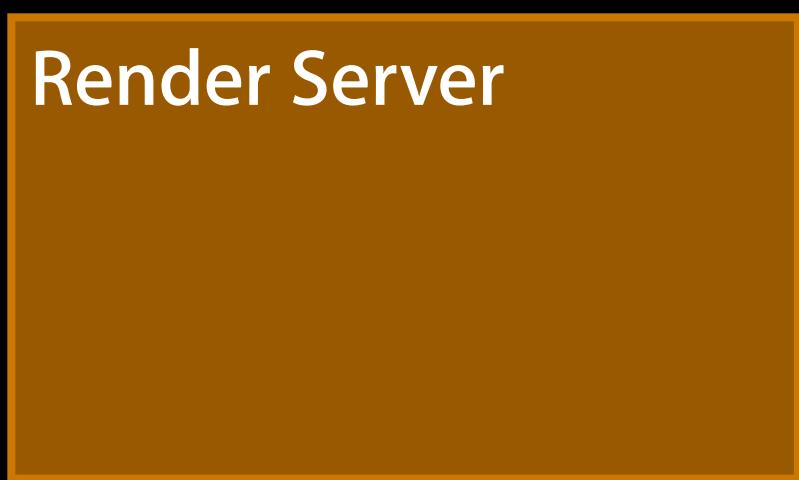
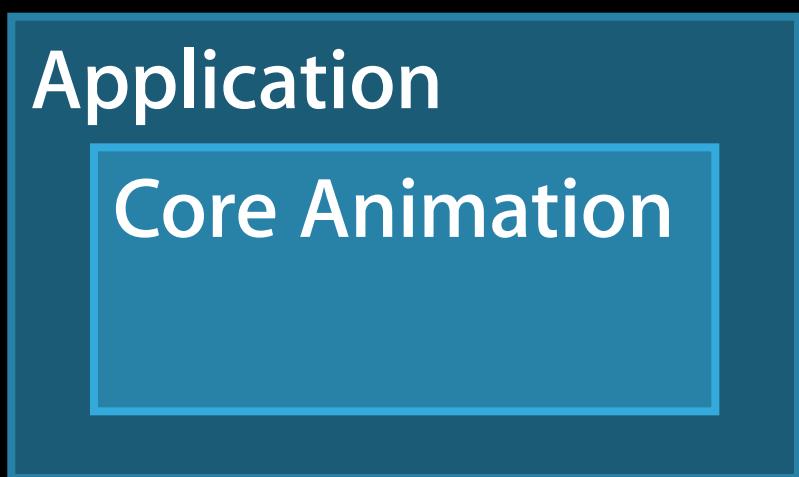
Core Animation Pipeline

Application

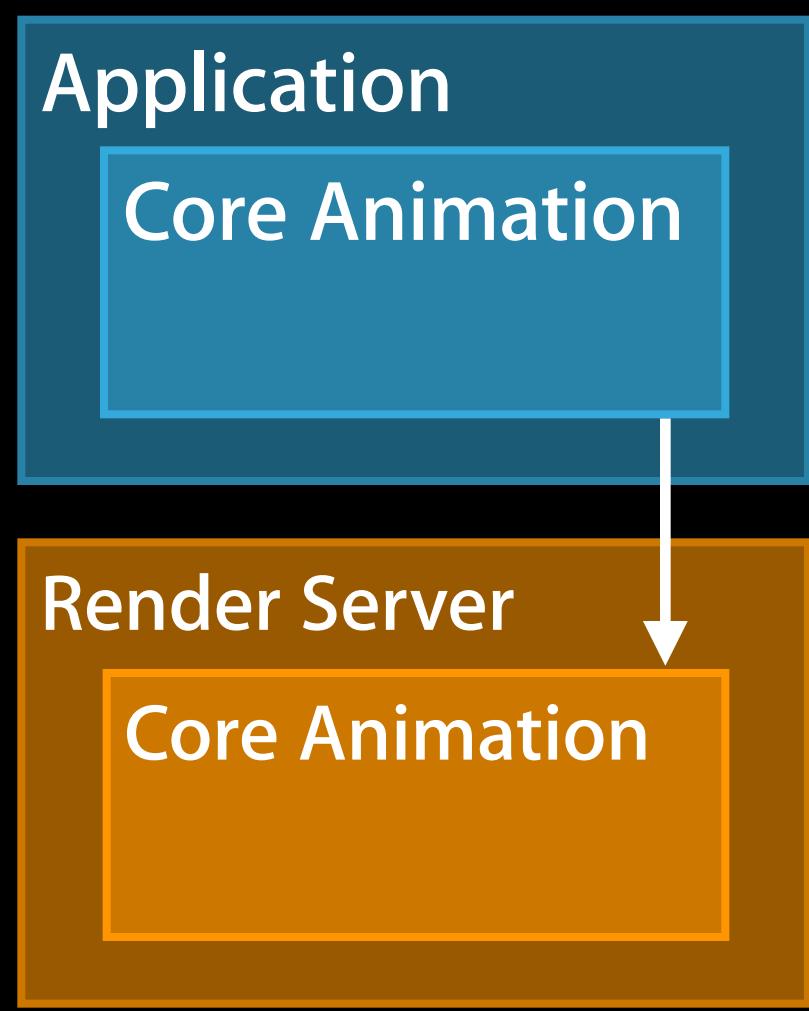
Core Animation Pipeline



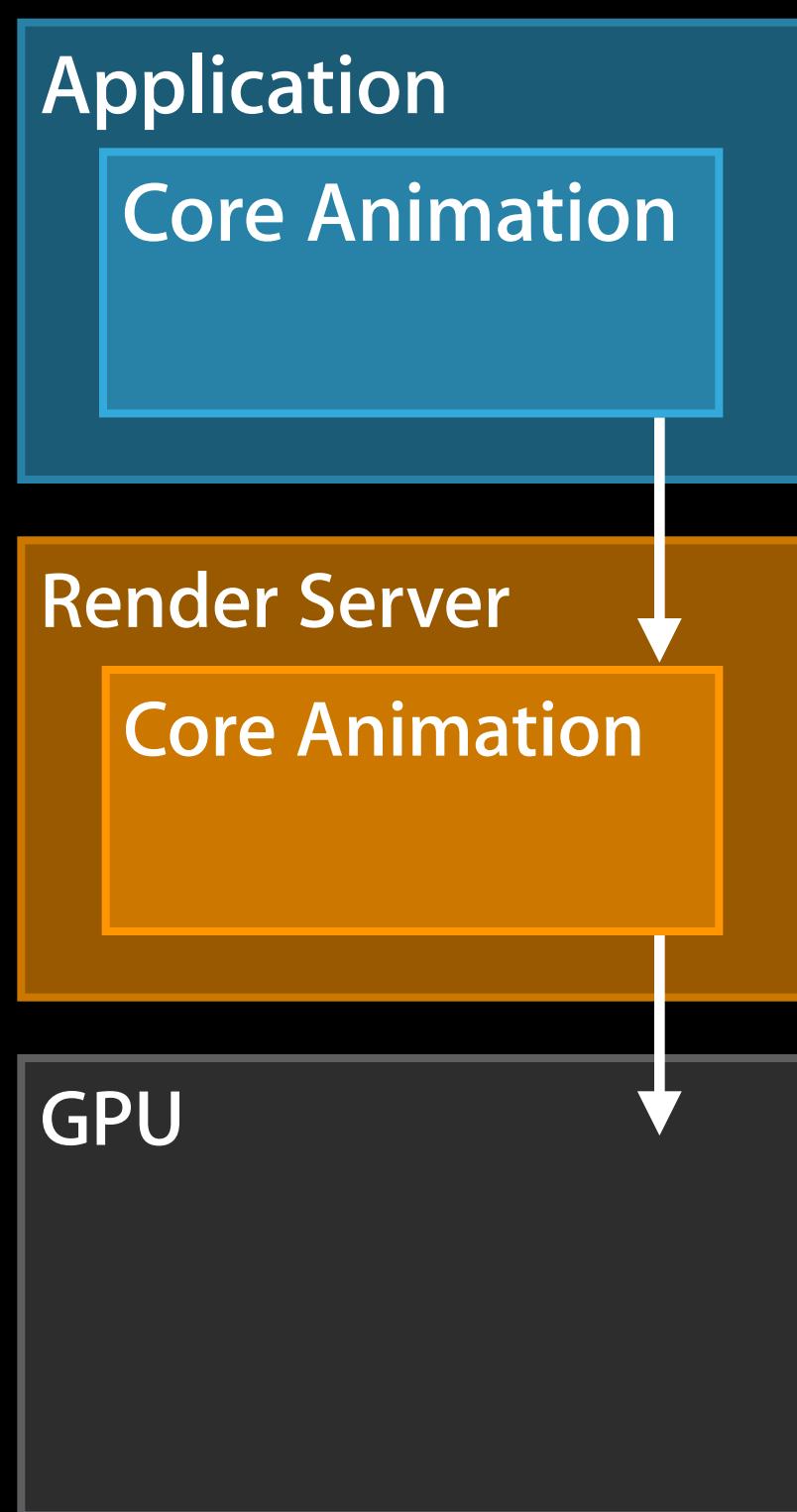
Core Animation Pipeline



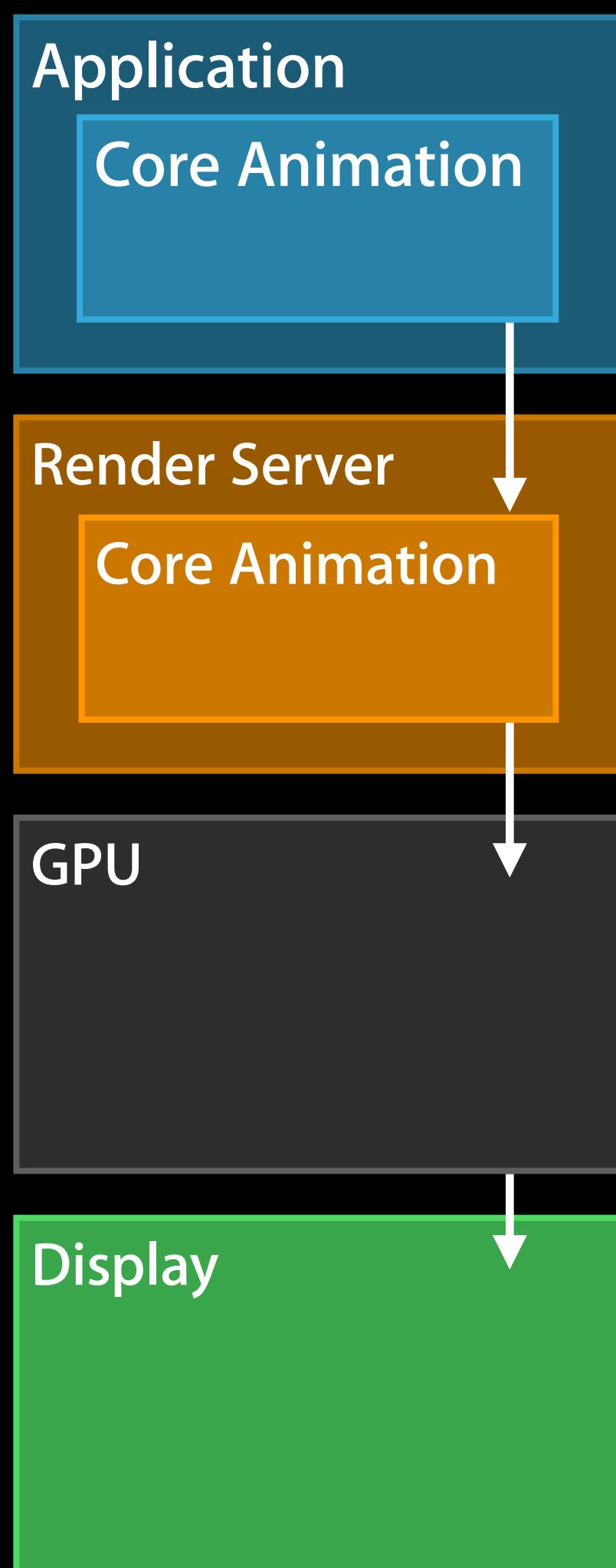
Core Animation Pipeline



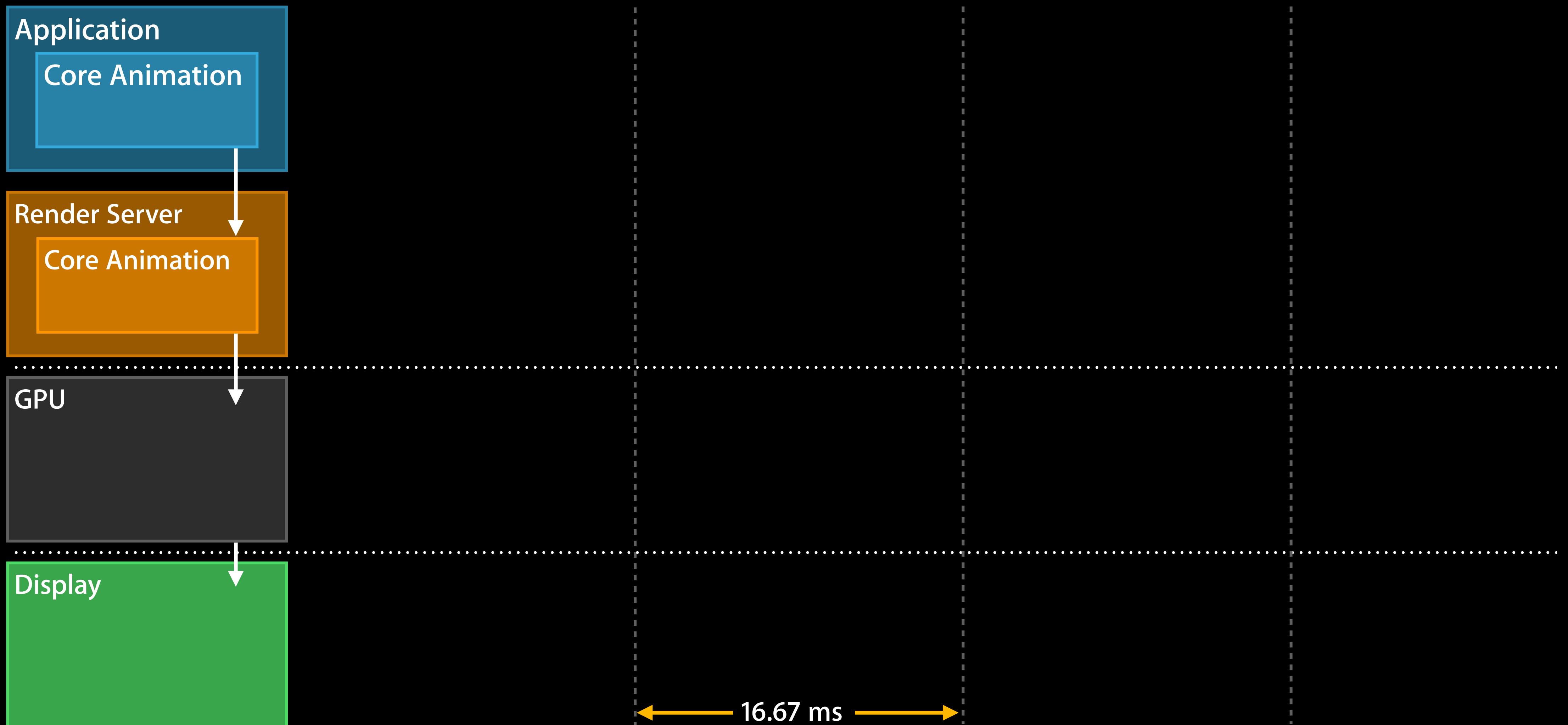
Core Animation Pipeline



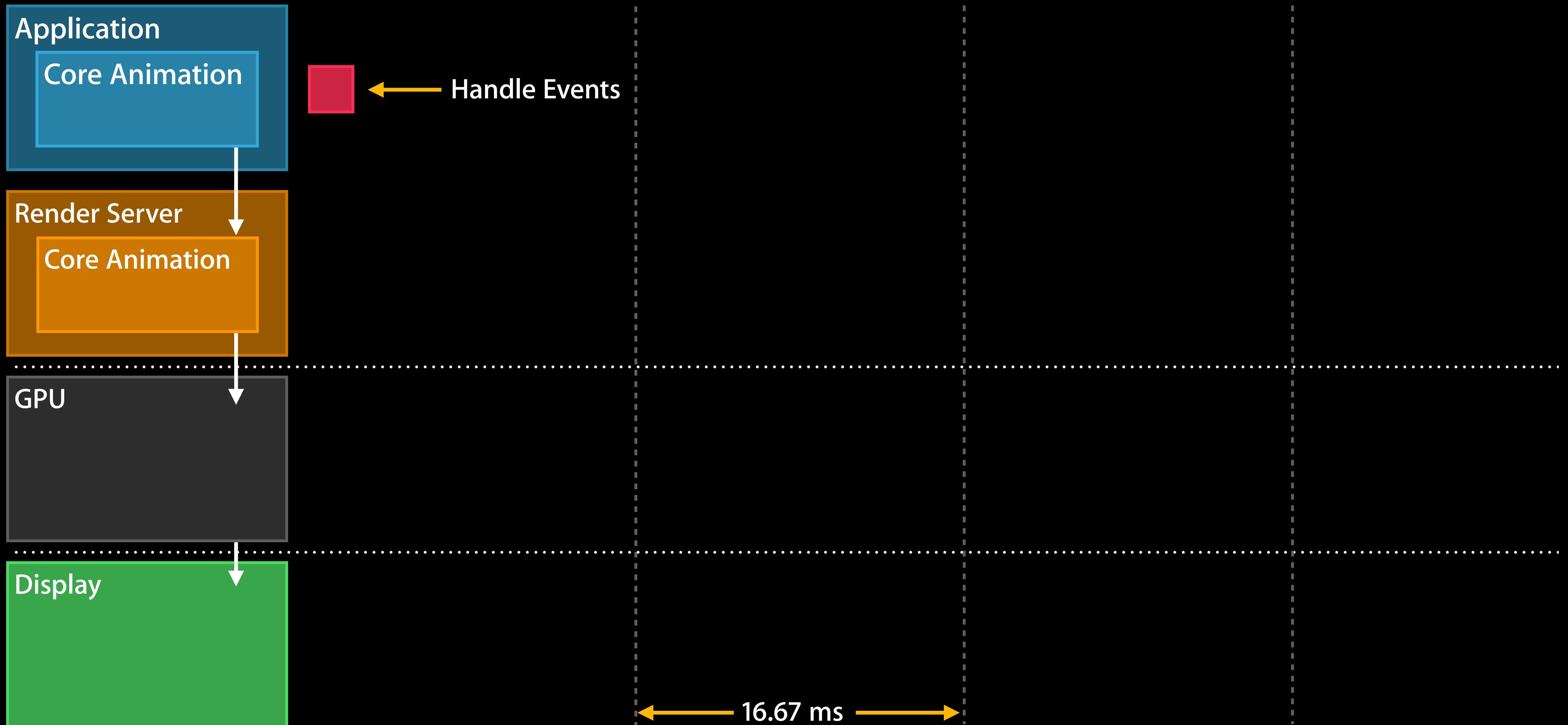
Core Animation Pipeline



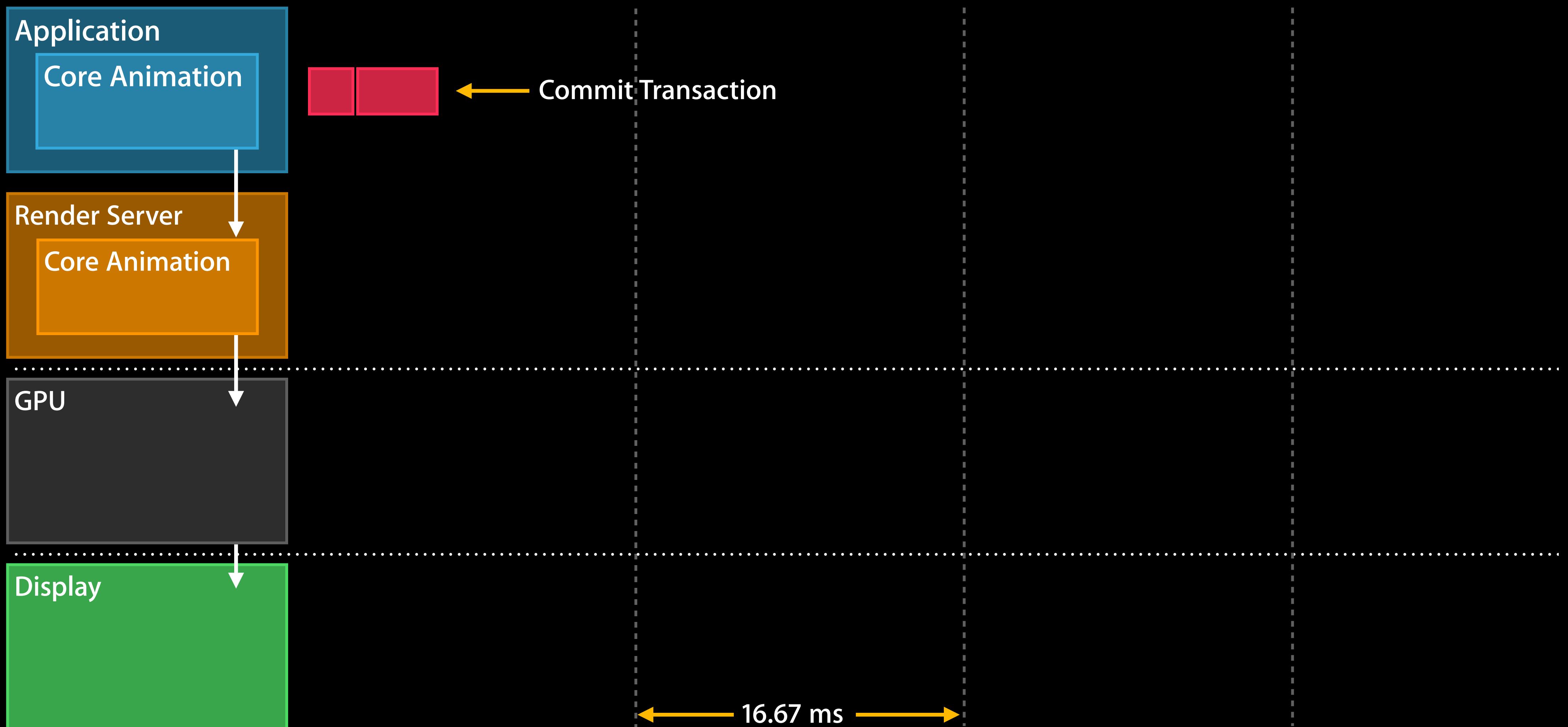
Core Animation Pipeline



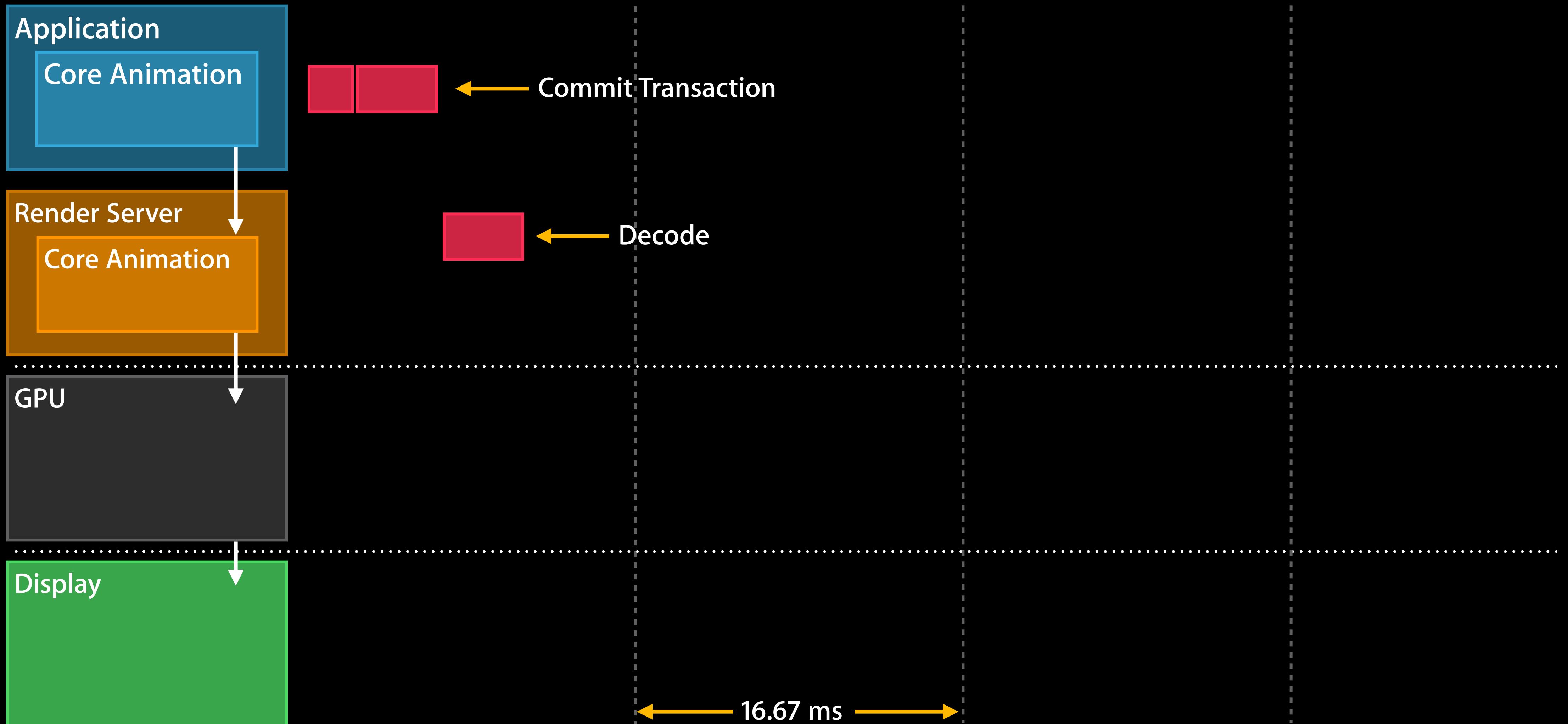
Core Animation Pipeline



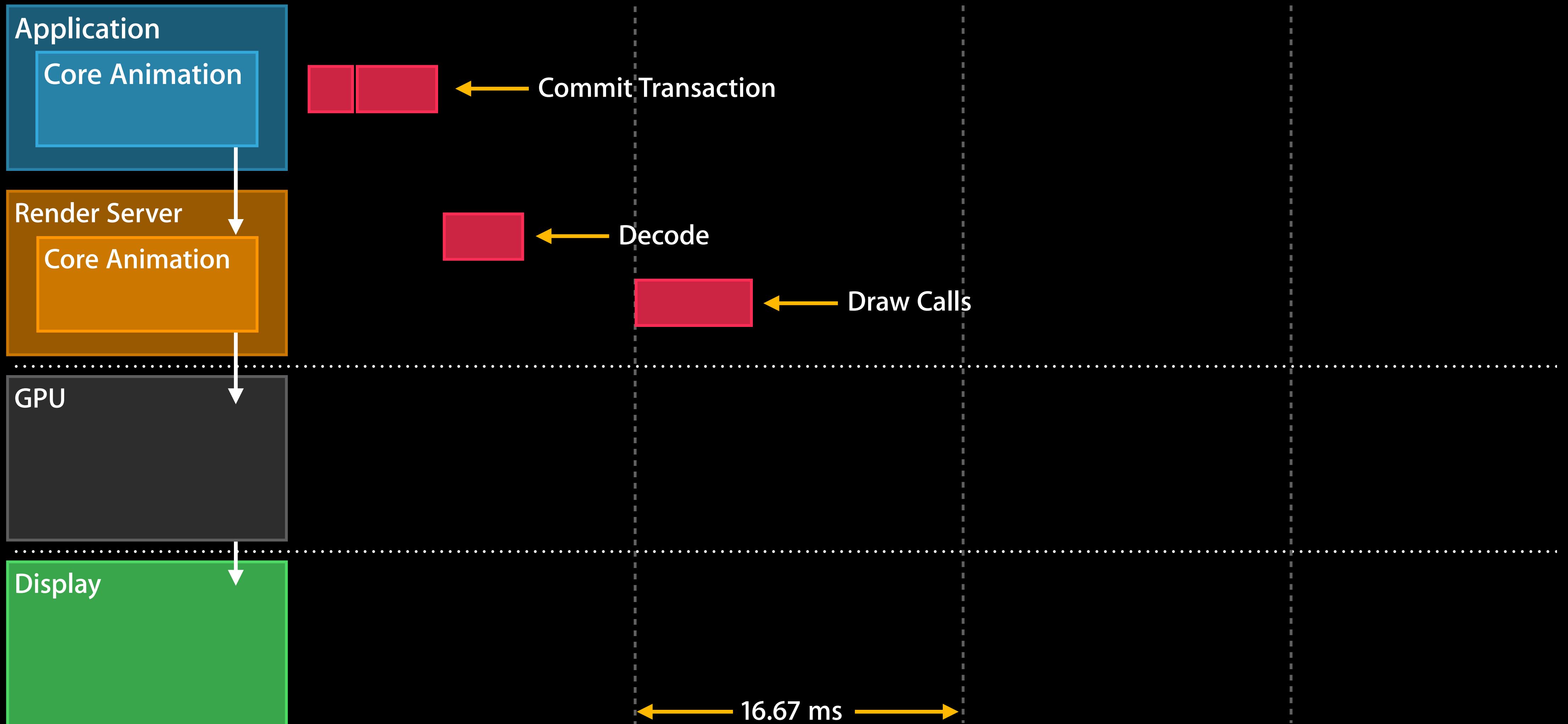
Core Animation Pipeline



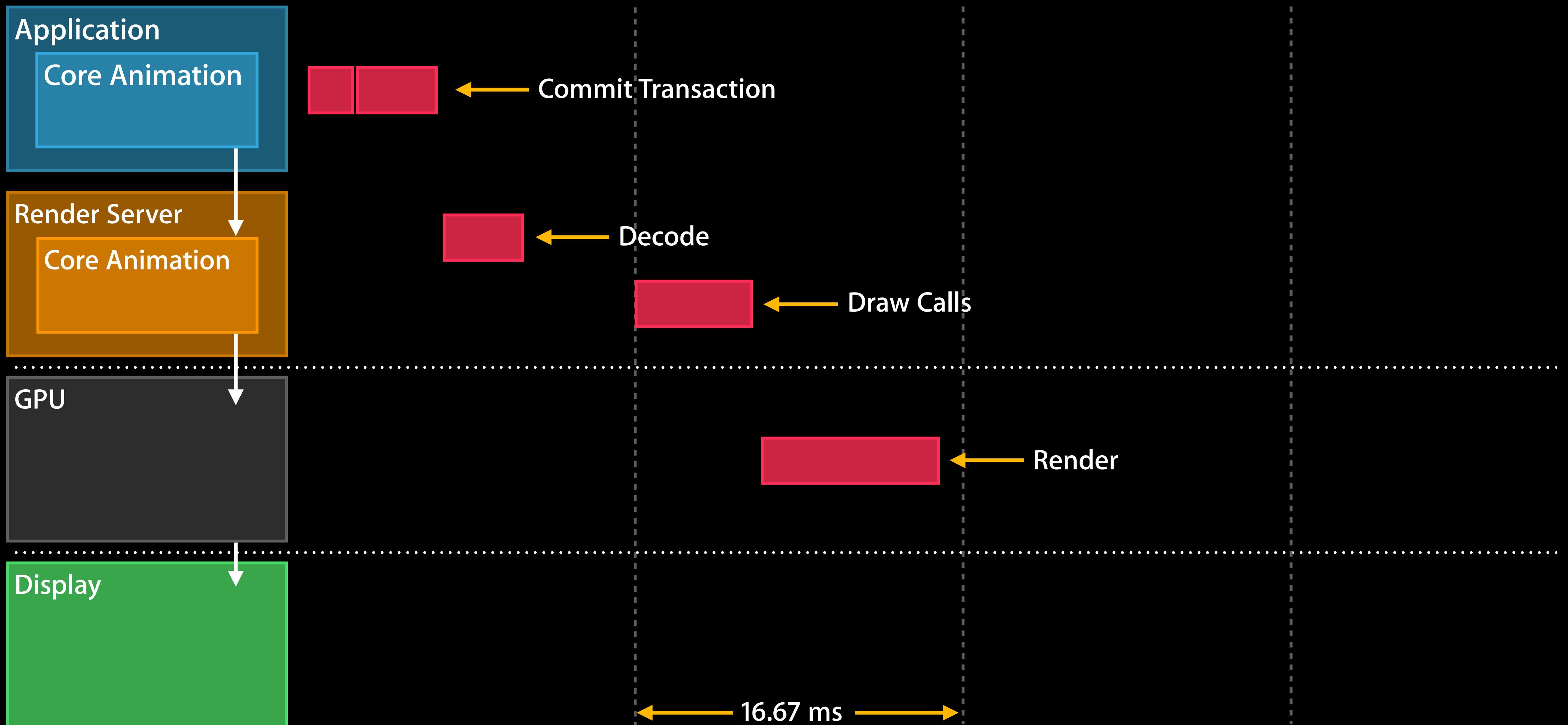
Core Animation Pipeline



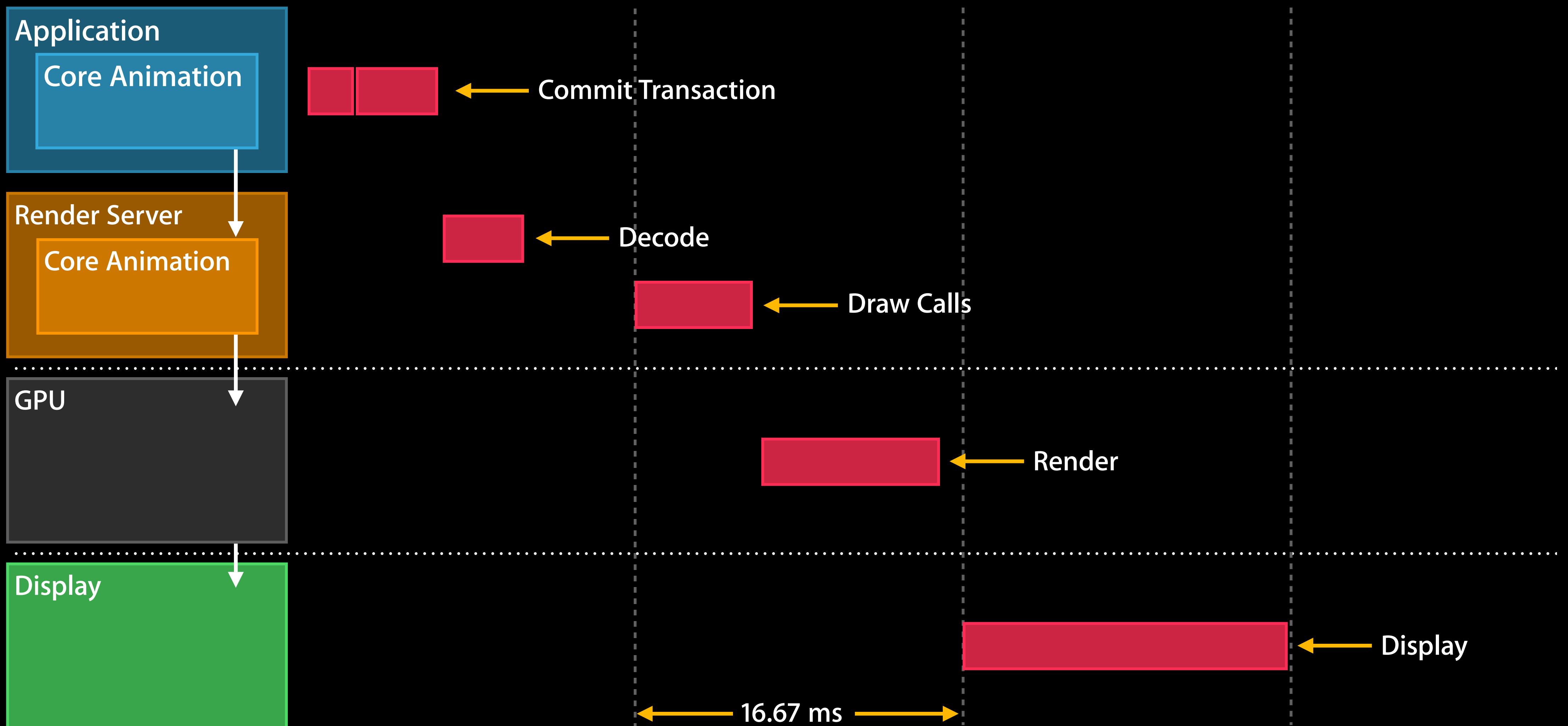
Core Animation Pipeline



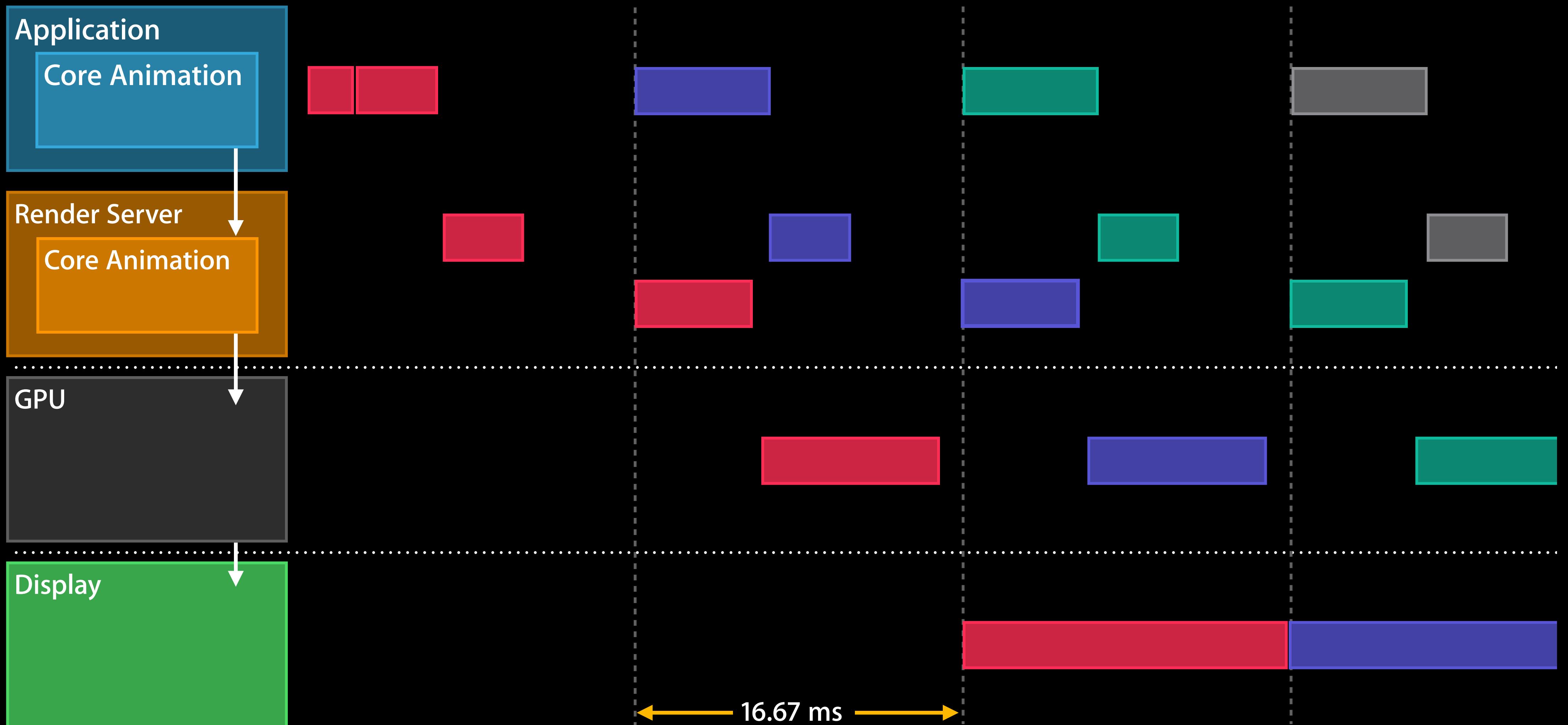
Core Animation Pipeline



Core Animation Pipeline



Core Animation Pipeline



Core Animation Pipeline



Commit Transaction

Commit Transaction

Layout

Set up the views

Commit Transaction

Layout

Display

Set up the views

Draw the views

Commit Transaction

Layout

Display

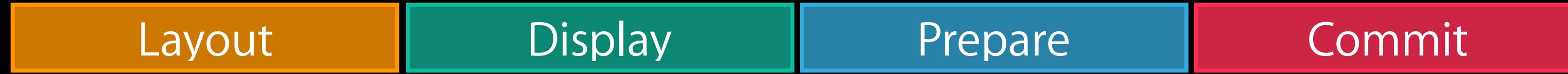
Prepare

Set up the views

Draw the views

Additional Core Animation work

Commit Transaction



Set up the views

Draw the views

Additional Core Animation work

Package up layers and send them to render server

Commit Transaction

Layout

Display

Prepare

Commit

Layout

Layout

Display

Prepare

Commit

`layoutSubviews` overrides are invoked

View creation, `addSubview:`

Populate content, database lookups

Usually CPU bound or I/O bound

Display

Layout

Display

Prepare

Commit

Draw contents via **drawRect**: if it is overridden

String drawing

Usually CPU or memory bound

Prepare Commit

Layout Display **Prepare** Commit

Image decoding

Image conversion

Commit

Layout

Display

Prepare

Commit

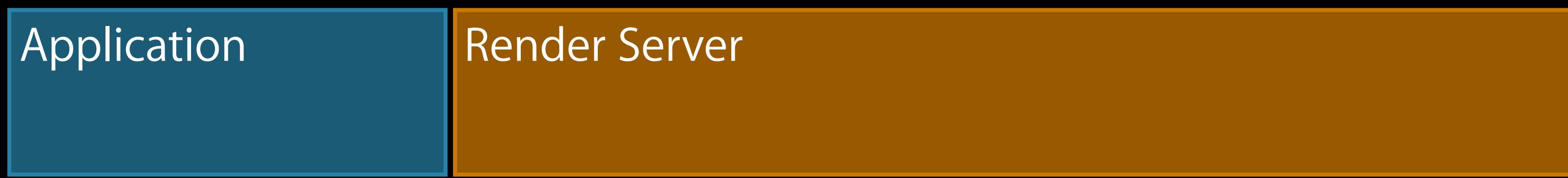
Package up layers and send to render server

Recursive

Expensive if layer tree is complex

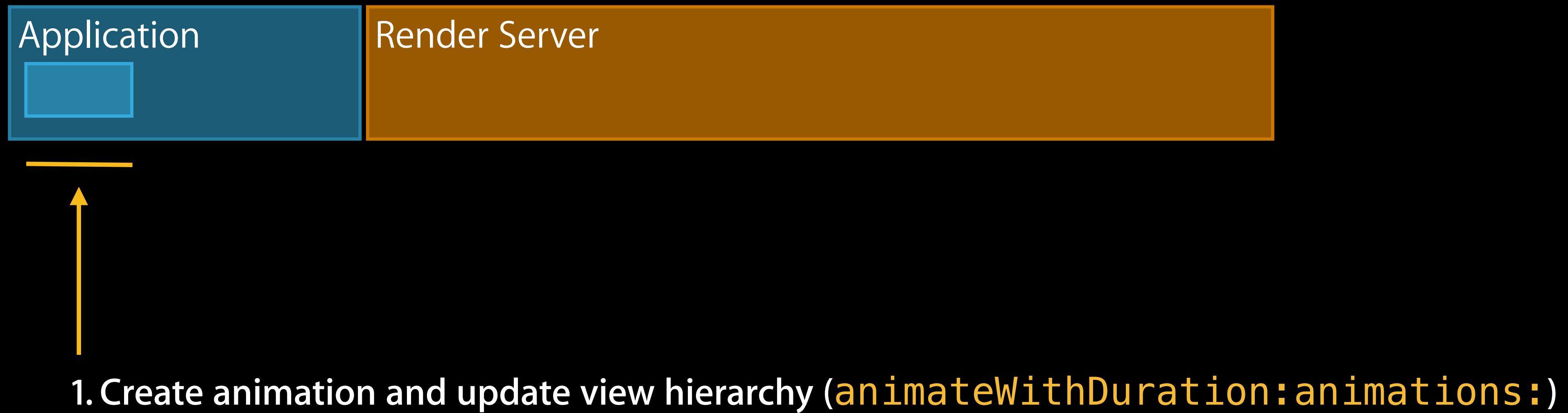
Animation

Three-stage process



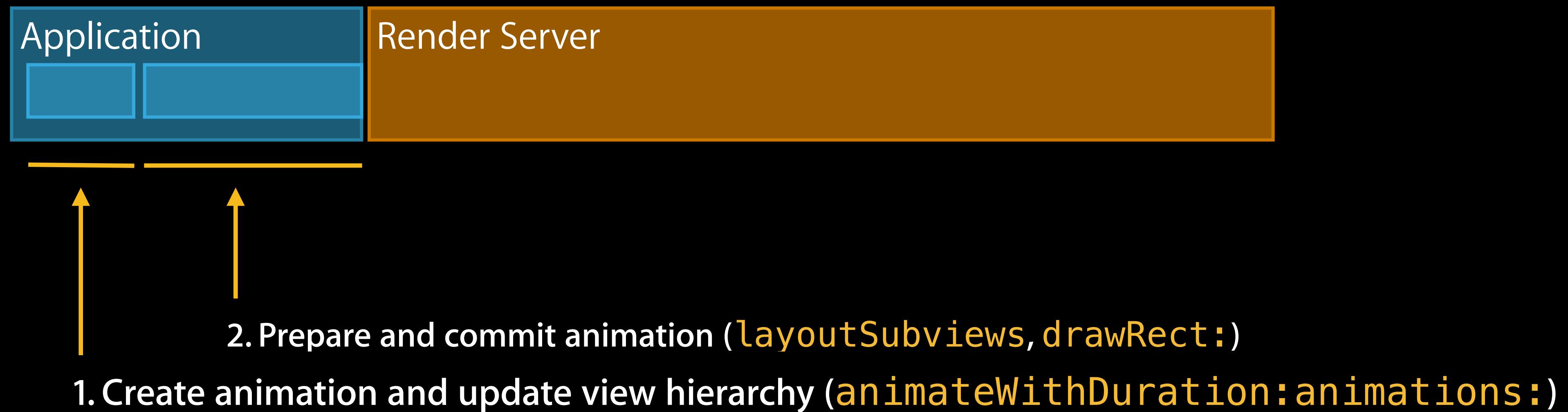
Animation

Three-stage process



Animation

Three-stage process



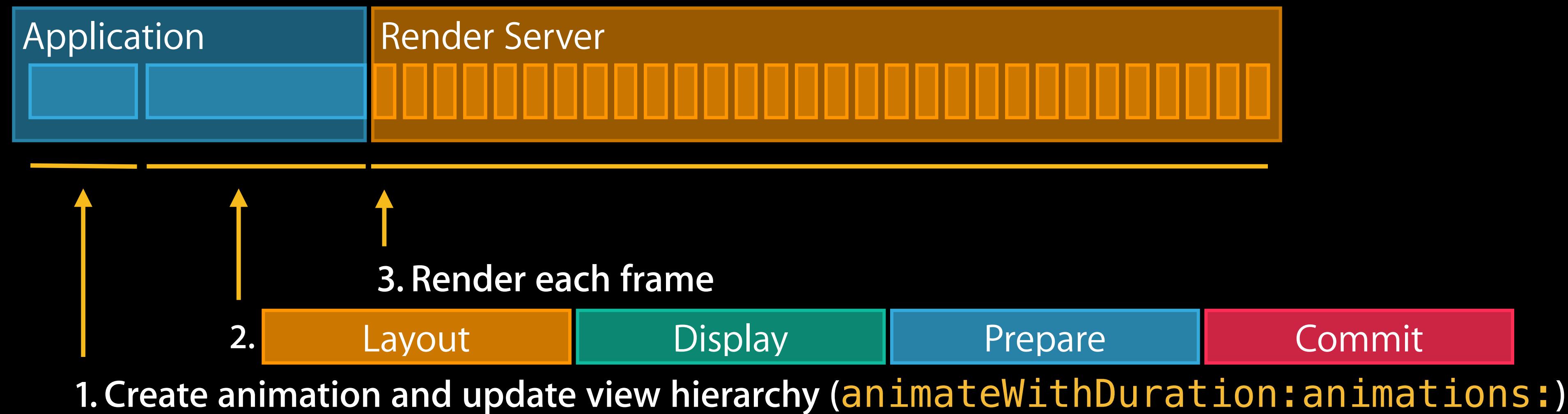
Animation

Three-stage process



Animation

Three-stage process



Rendering Concepts

Axel Wefers
iOS Software Engineer

Rendering Concepts

Tile based rendering

Render passes

Example masking

Tile Based Rendering

Screen is split into tiles of NxN pixels

Each tile fits into the SoC cache

Geometry is split in tile buckets

Rasterization can begin after all geometry
is submitted

Tile Based Rendering

Screen is split into tiles of NxN pixels

Each tile fits into the SoC cache

Geometry is split in tile buckets

Rasterization can begin after all geometry
is submitted



Tile Based Rendering

Screen is split into tiles of NxN pixels

Each tile fits into the SoC cache

Geometry is split in tile buckets

Rasterization can begin after all geometry
is submitted



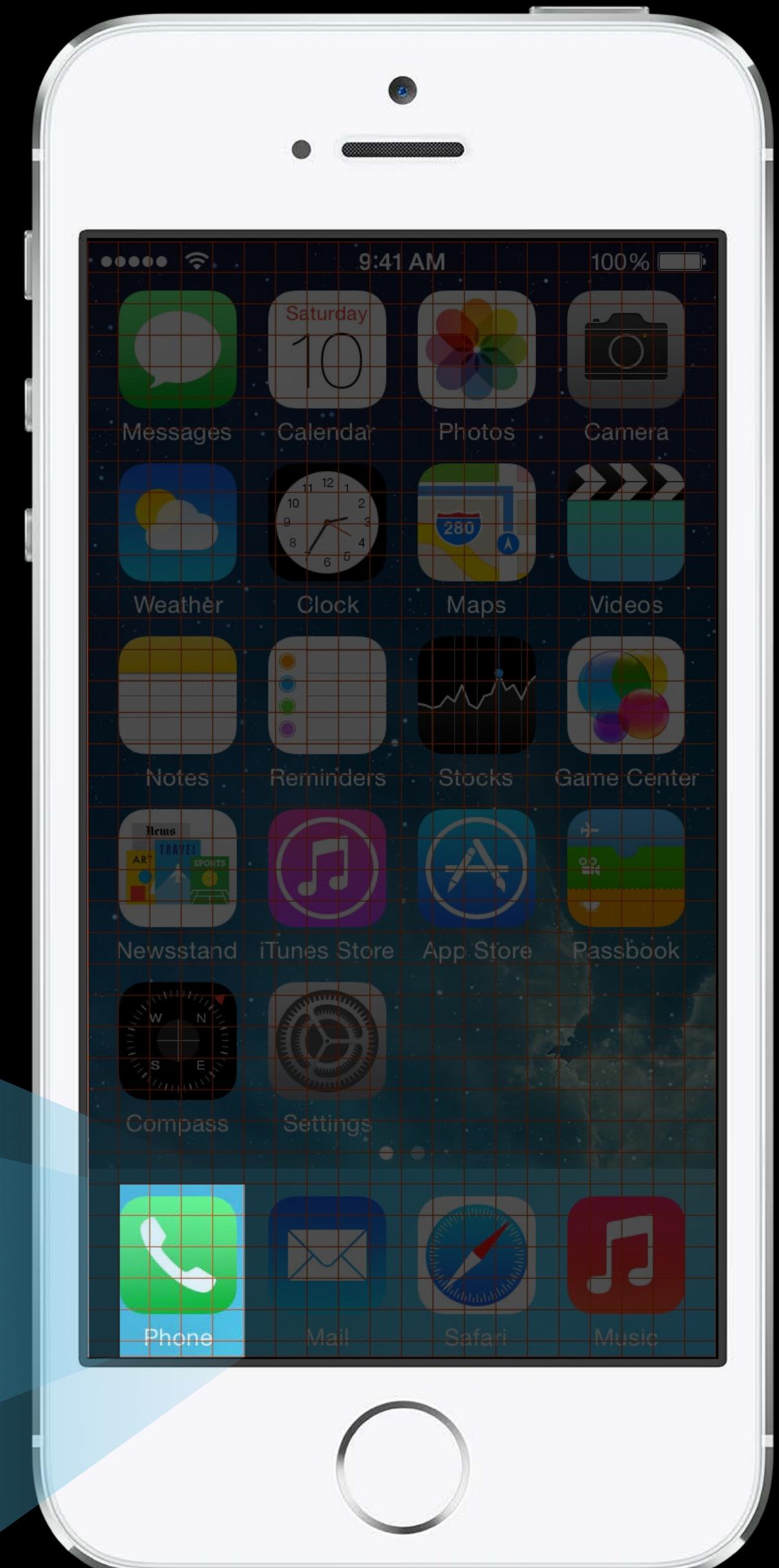
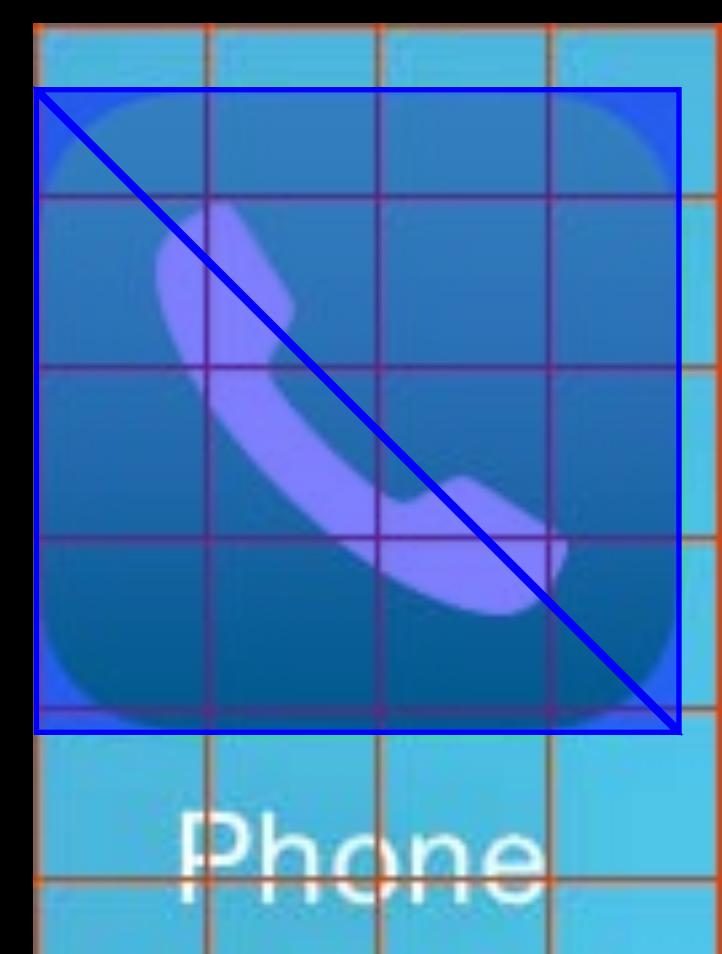
Tile Based Rendering

Screen is split into tiles of NxN pixels

Each tile fits into the SoC cache

Geometry is split in tile buckets

Rasterization can begin after all geometry
is submitted



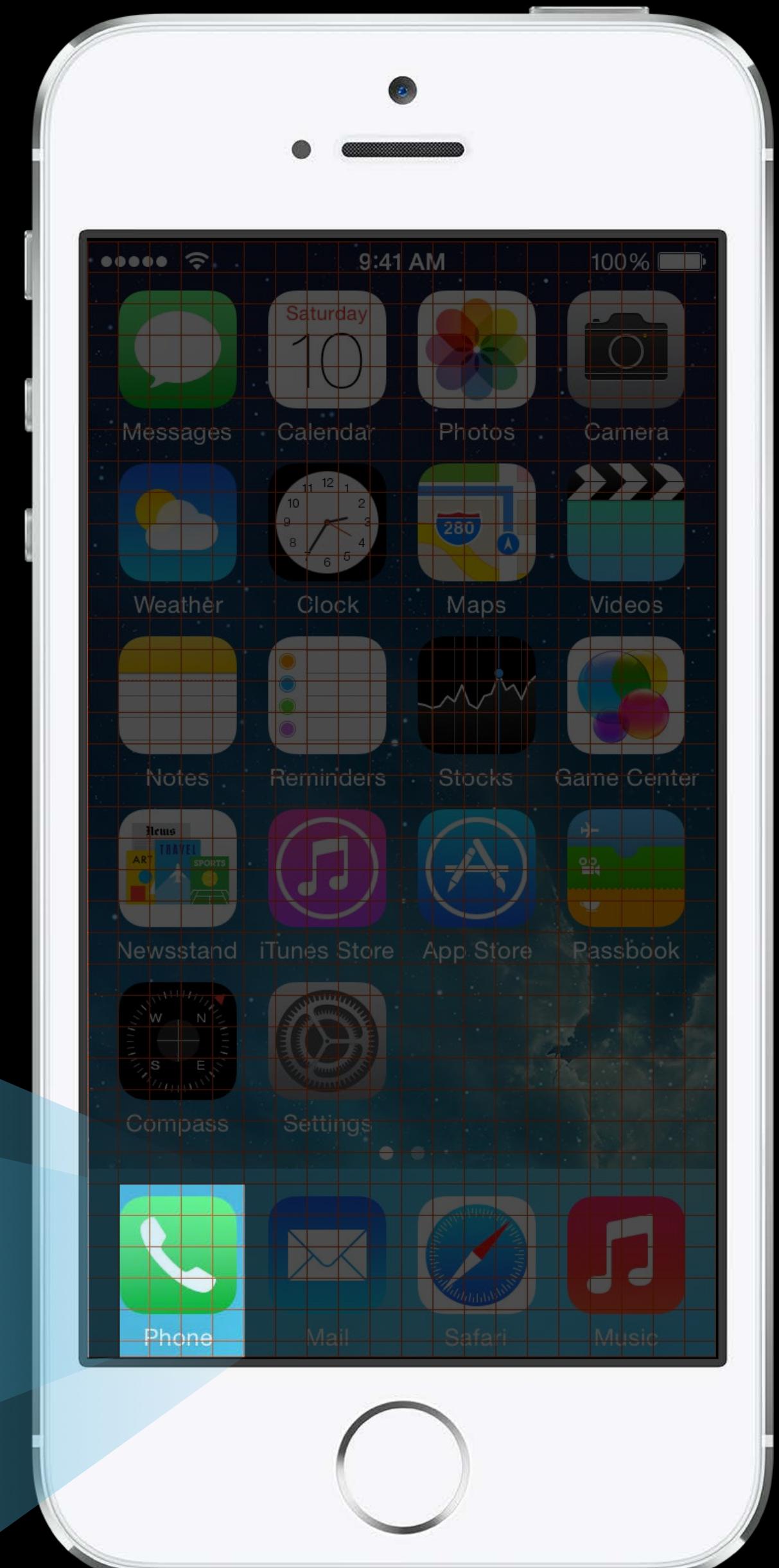
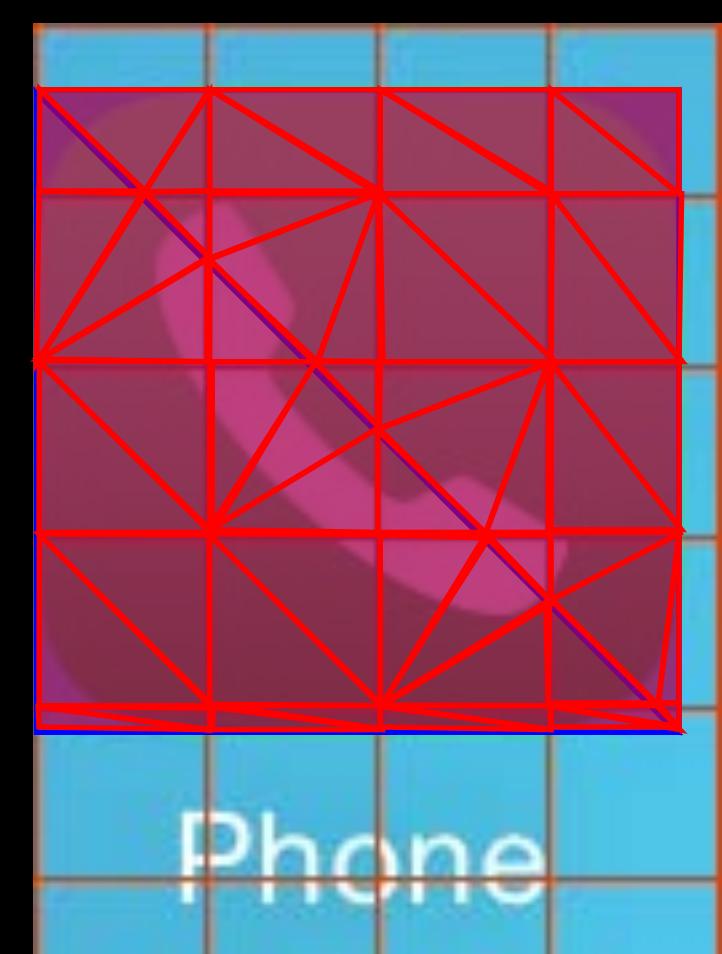
Tile Based Rendering

Screen is split into tiles of NxN pixels

Each tile fits into the SoC cache

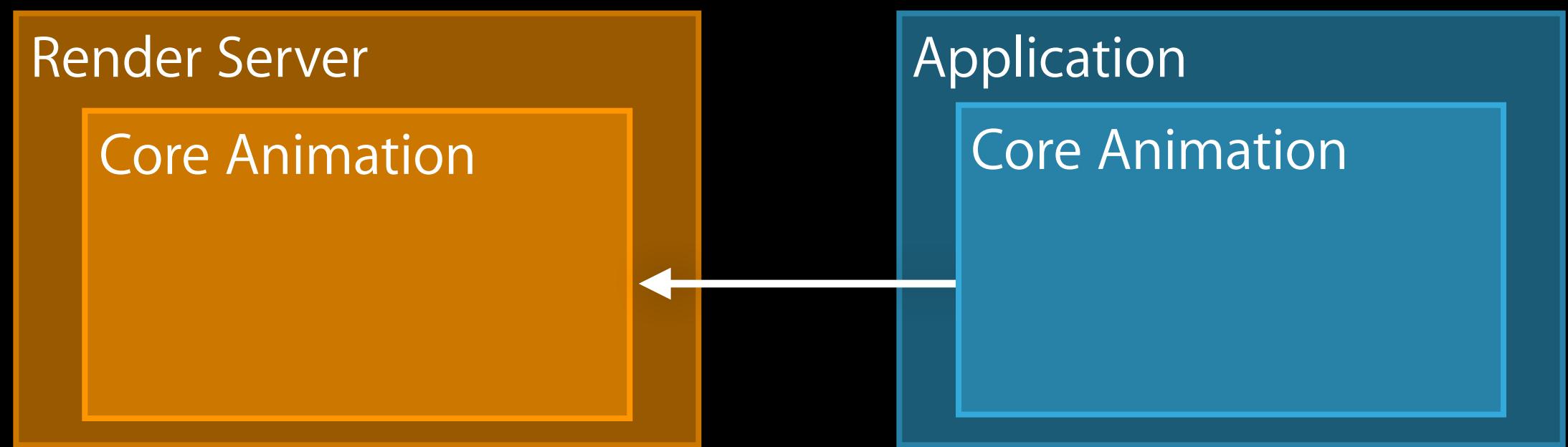
Geometry is split in tile buckets

Rasterization can begin after all geometry
is submitted



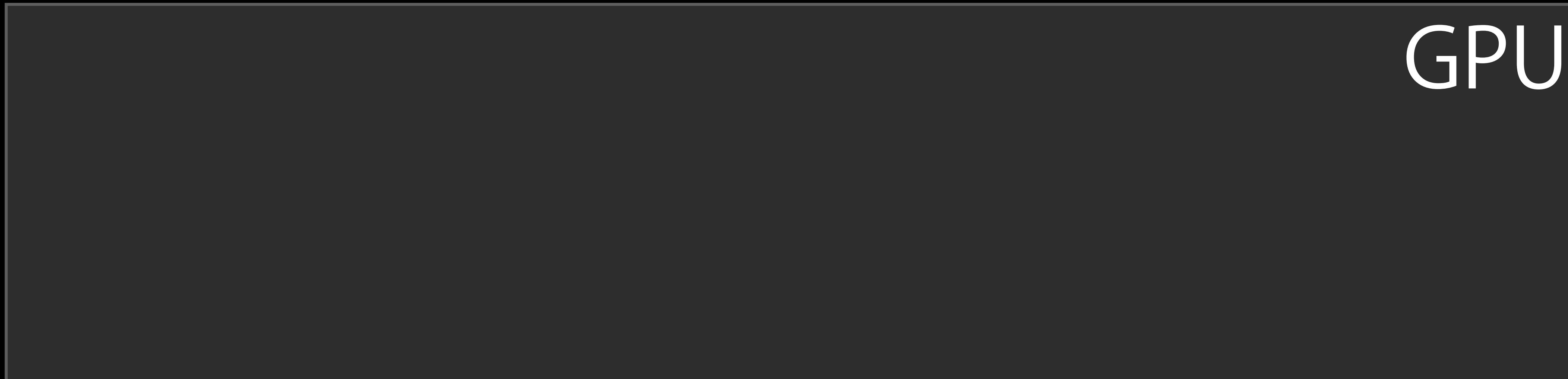
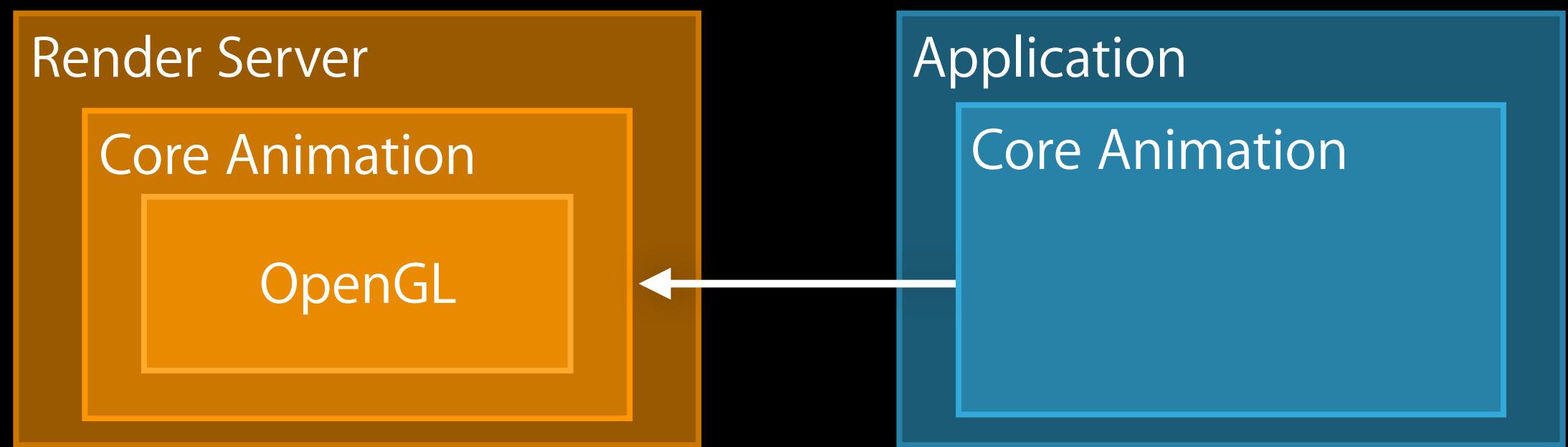
Tile Based Rendering

Rendering pass



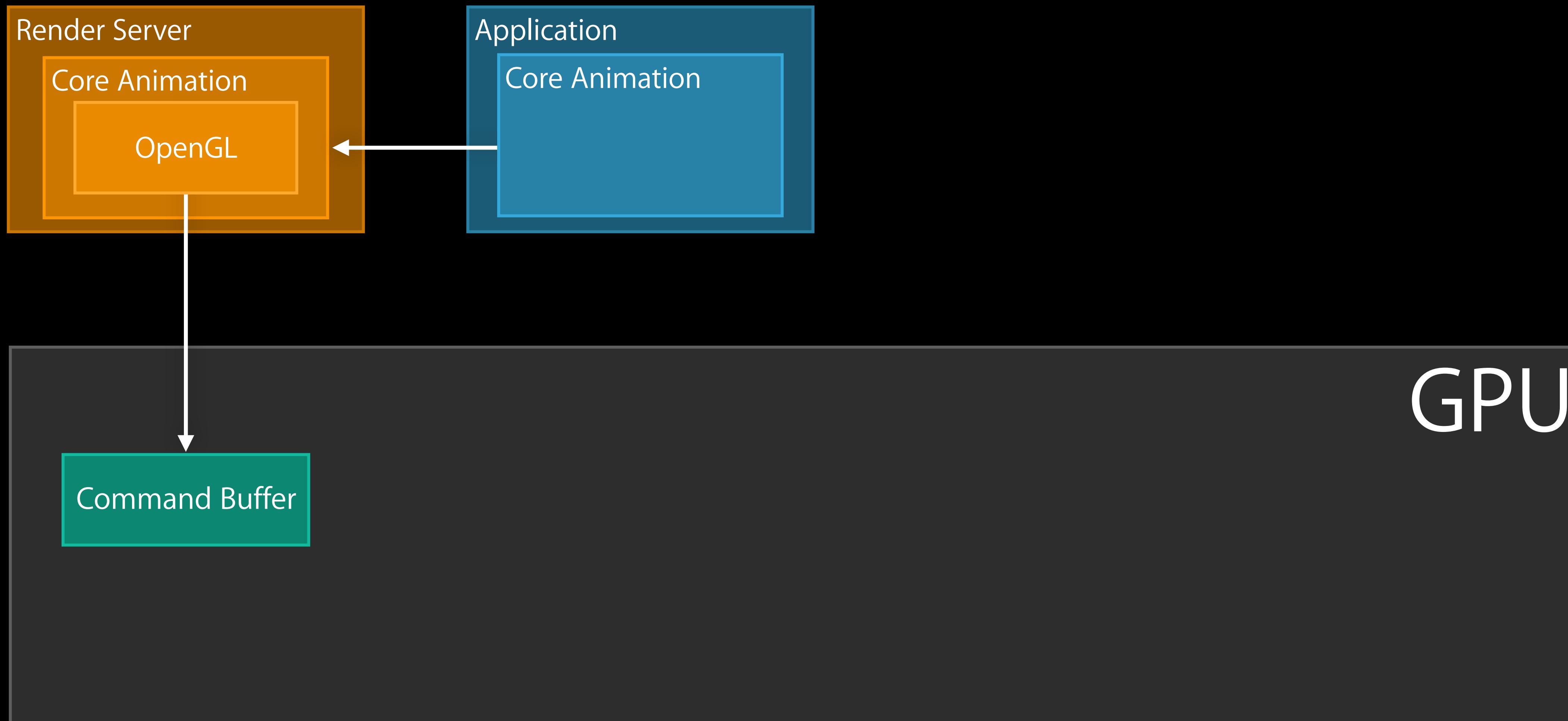
Tile Based Rendering

Rendering pass



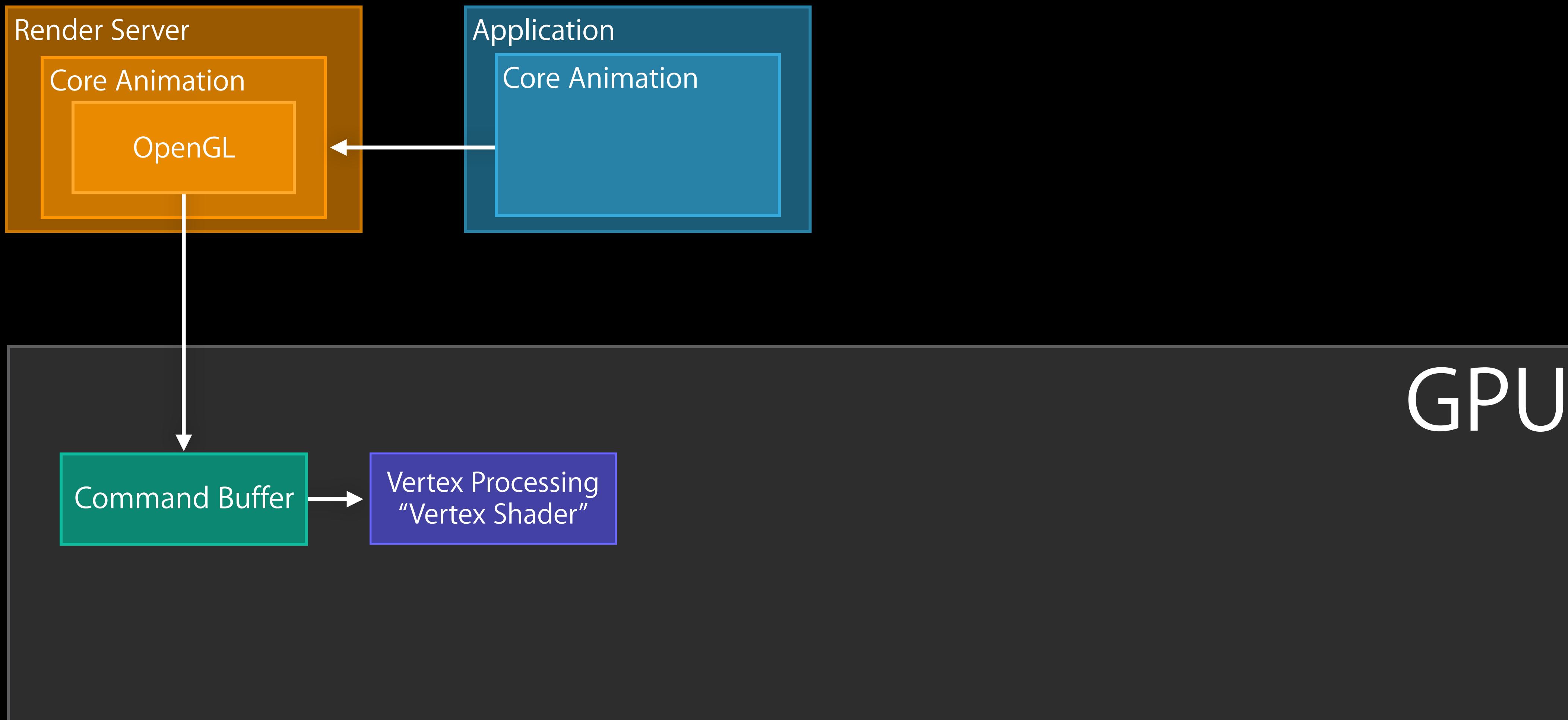
Tile Based Rendering

Rendering pass



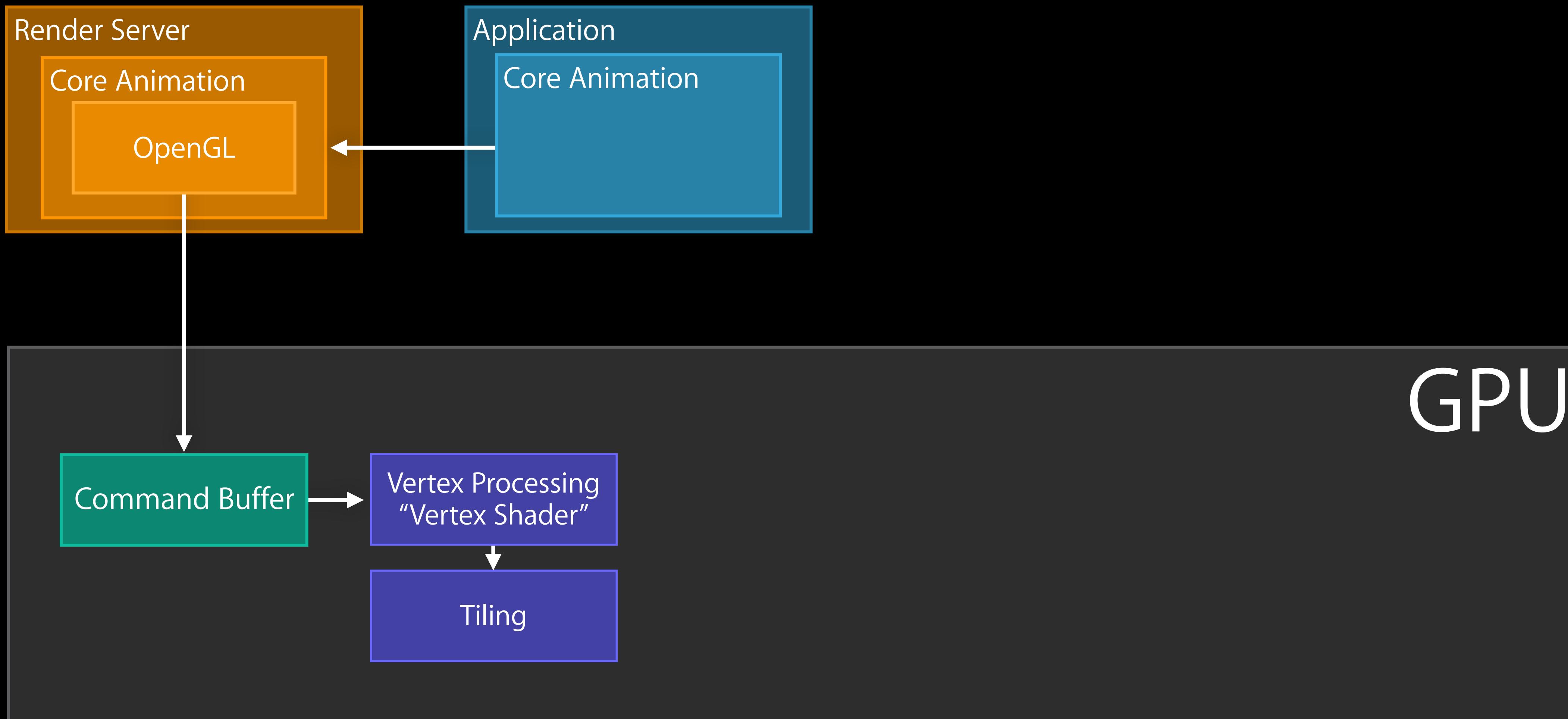
Tile Based Rendering

Rendering pass



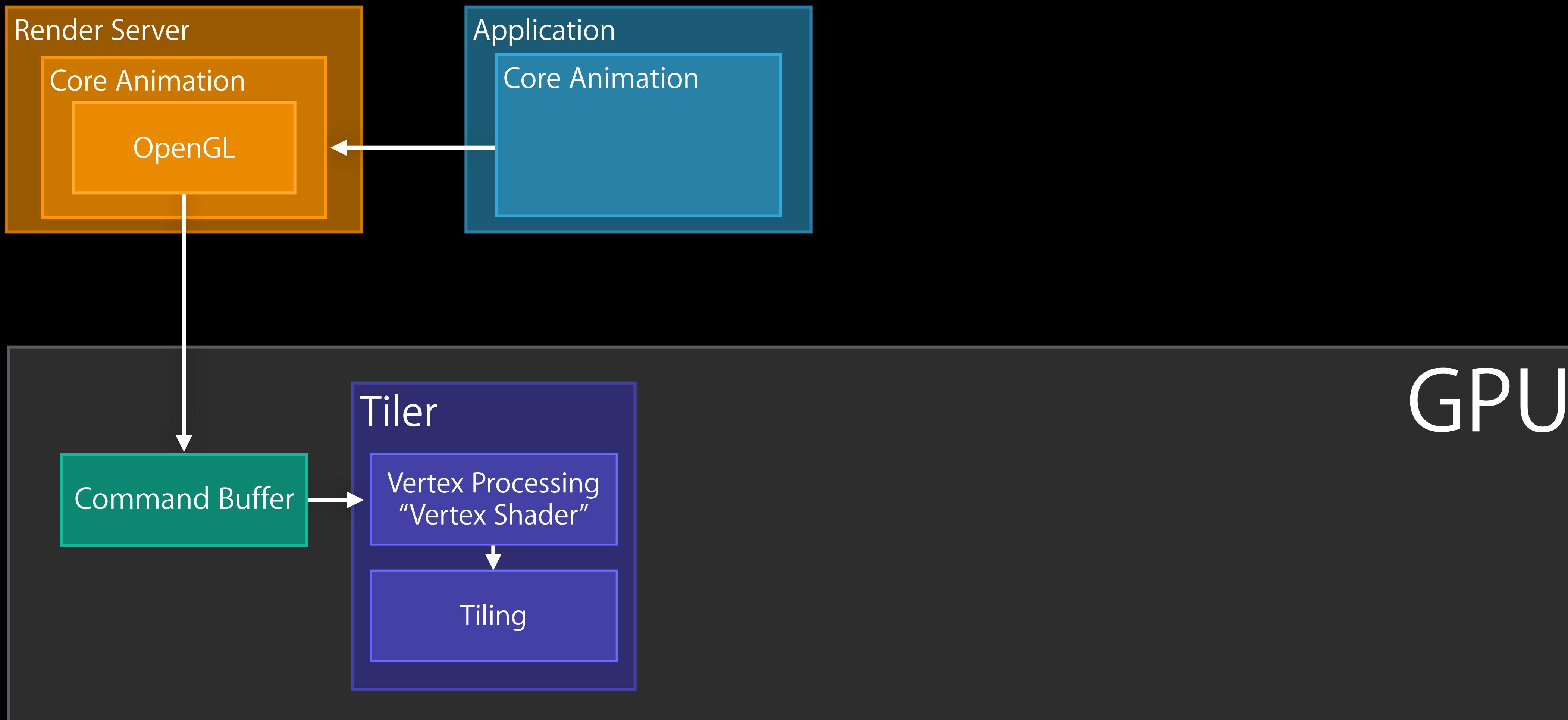
Tile Based Rendering

Rendering pass



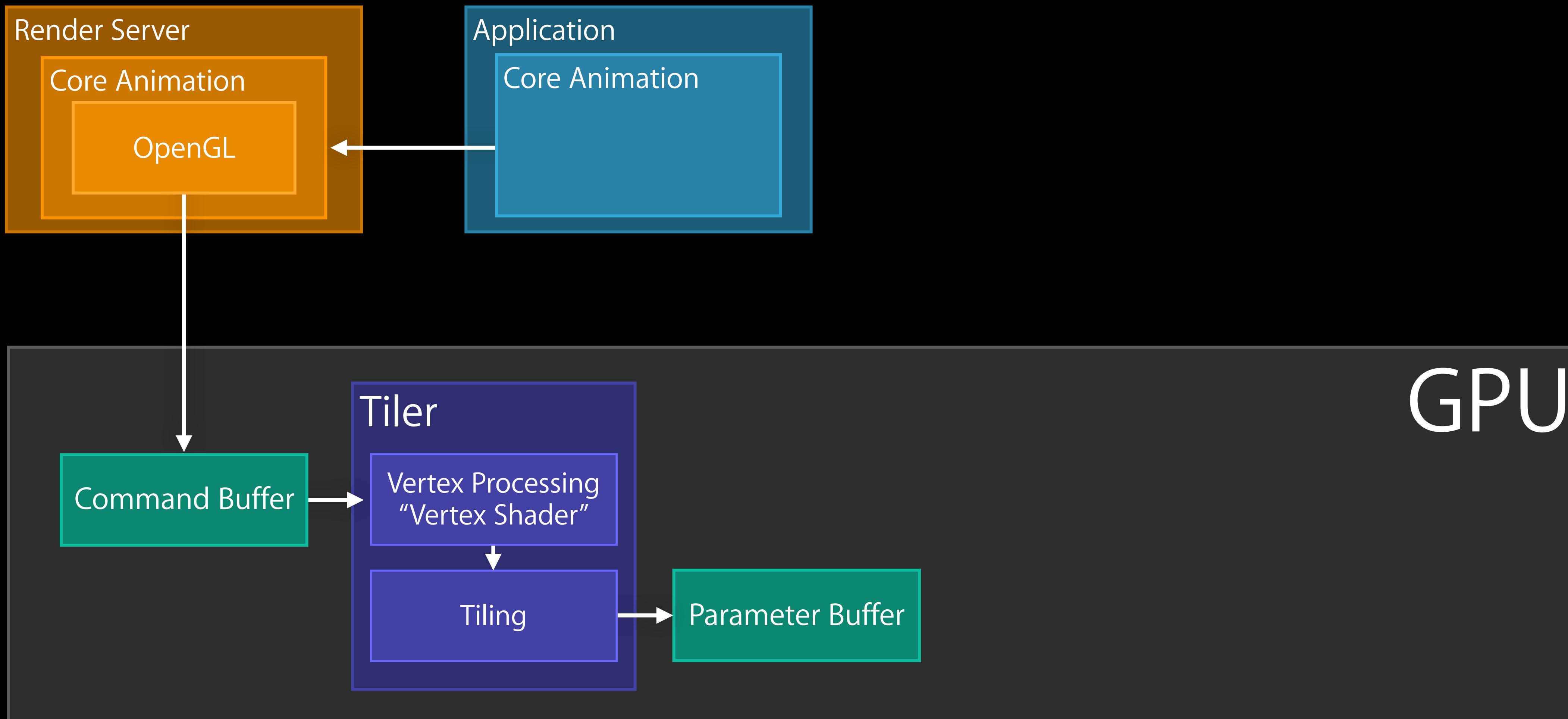
Tile Based Rendering

Rendering pass



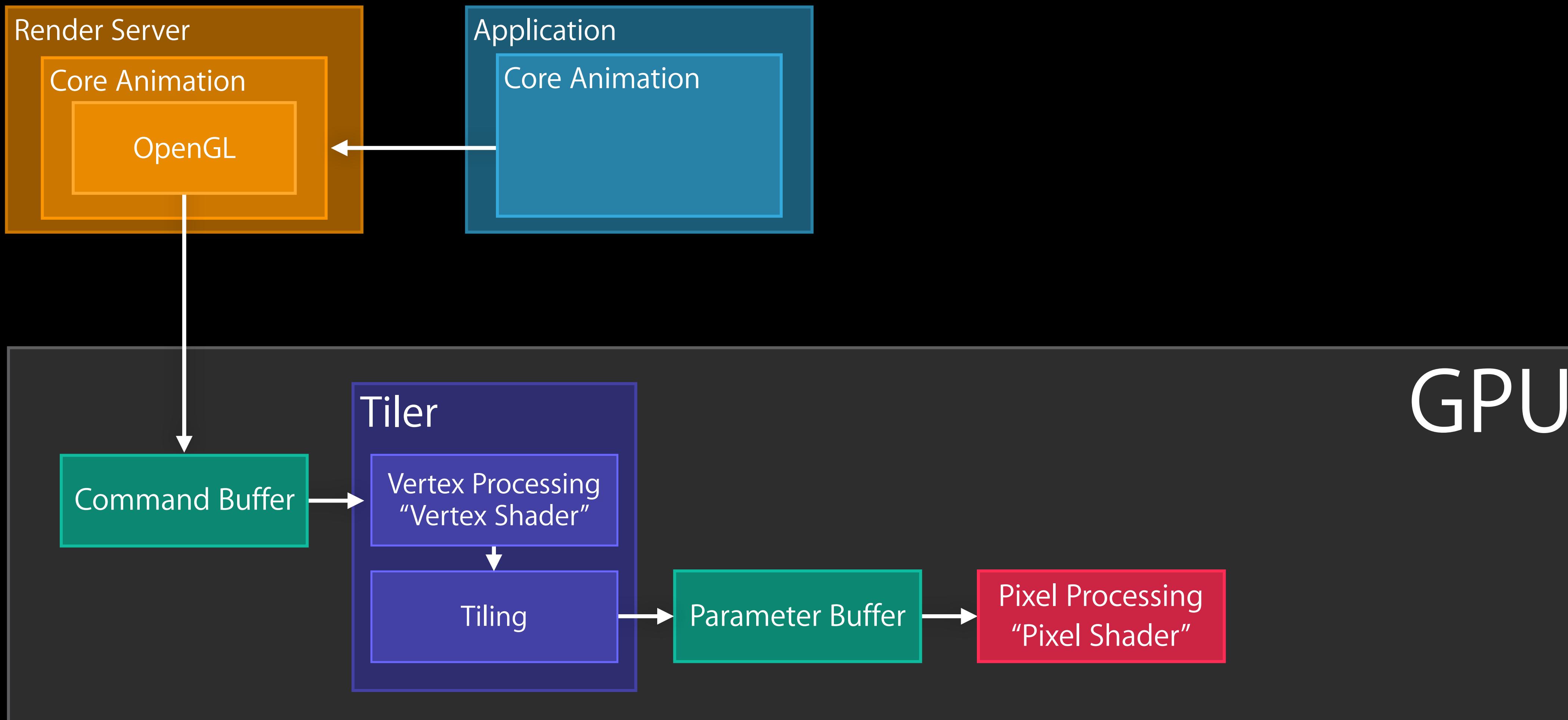
Tile Based Rendering

Rendering pass



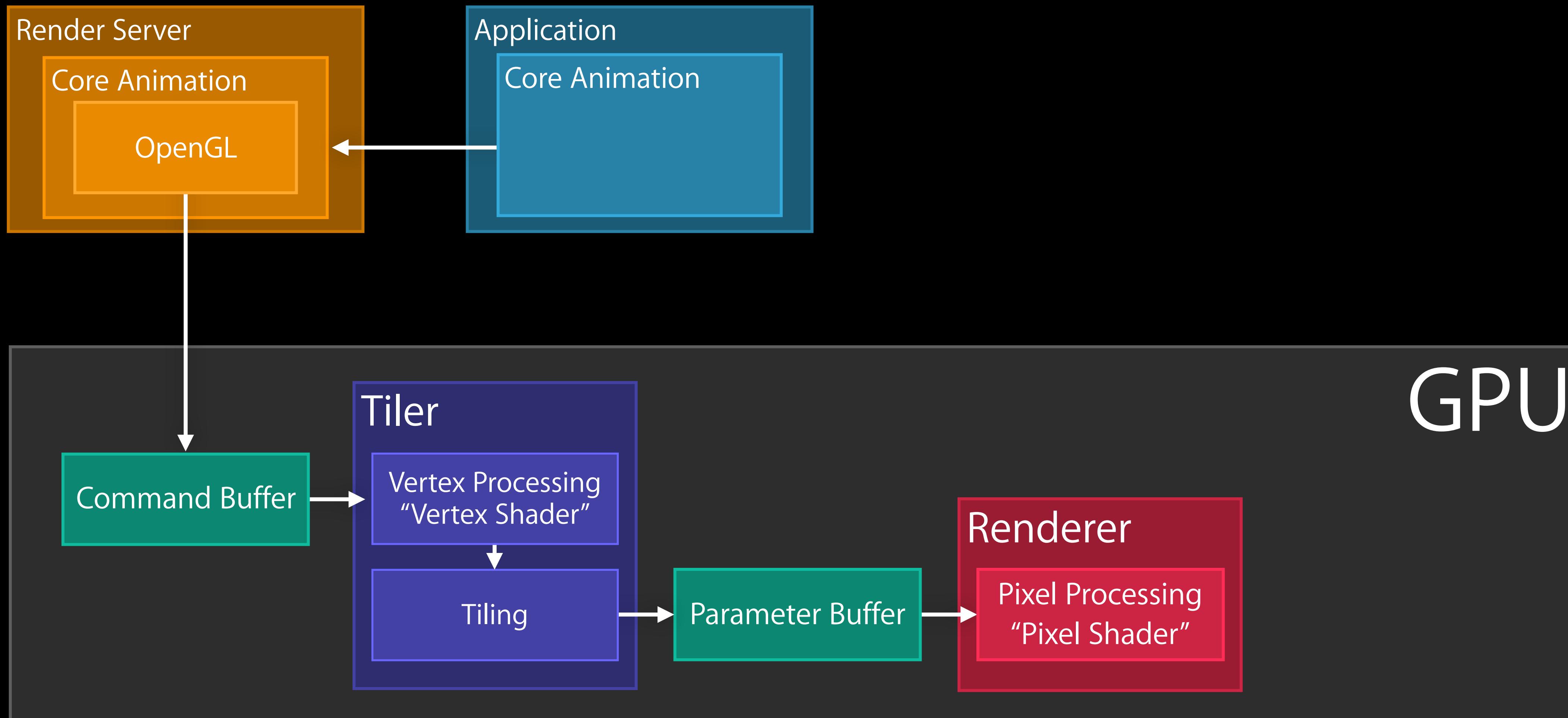
Tile Based Rendering

Rendering pass



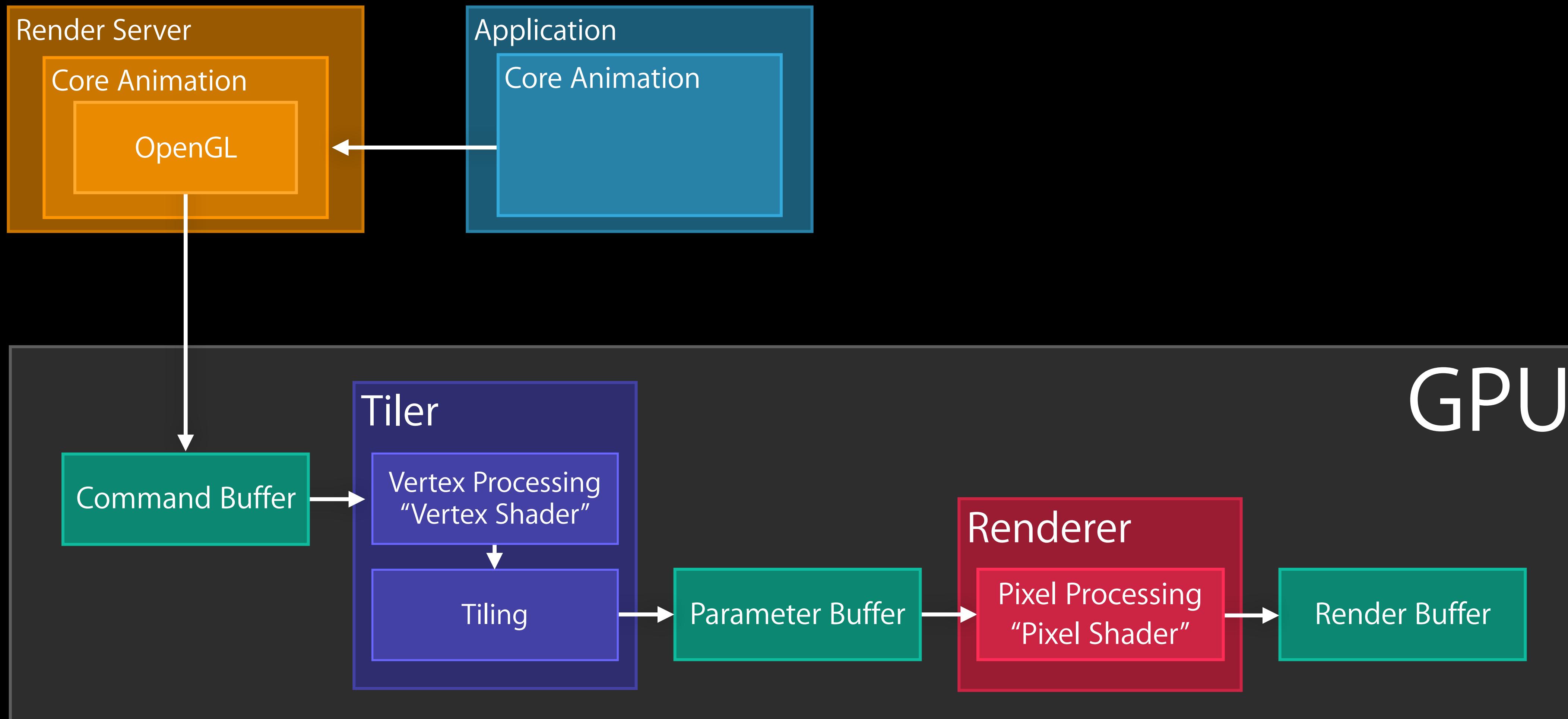
Tile Based Rendering

Rendering pass



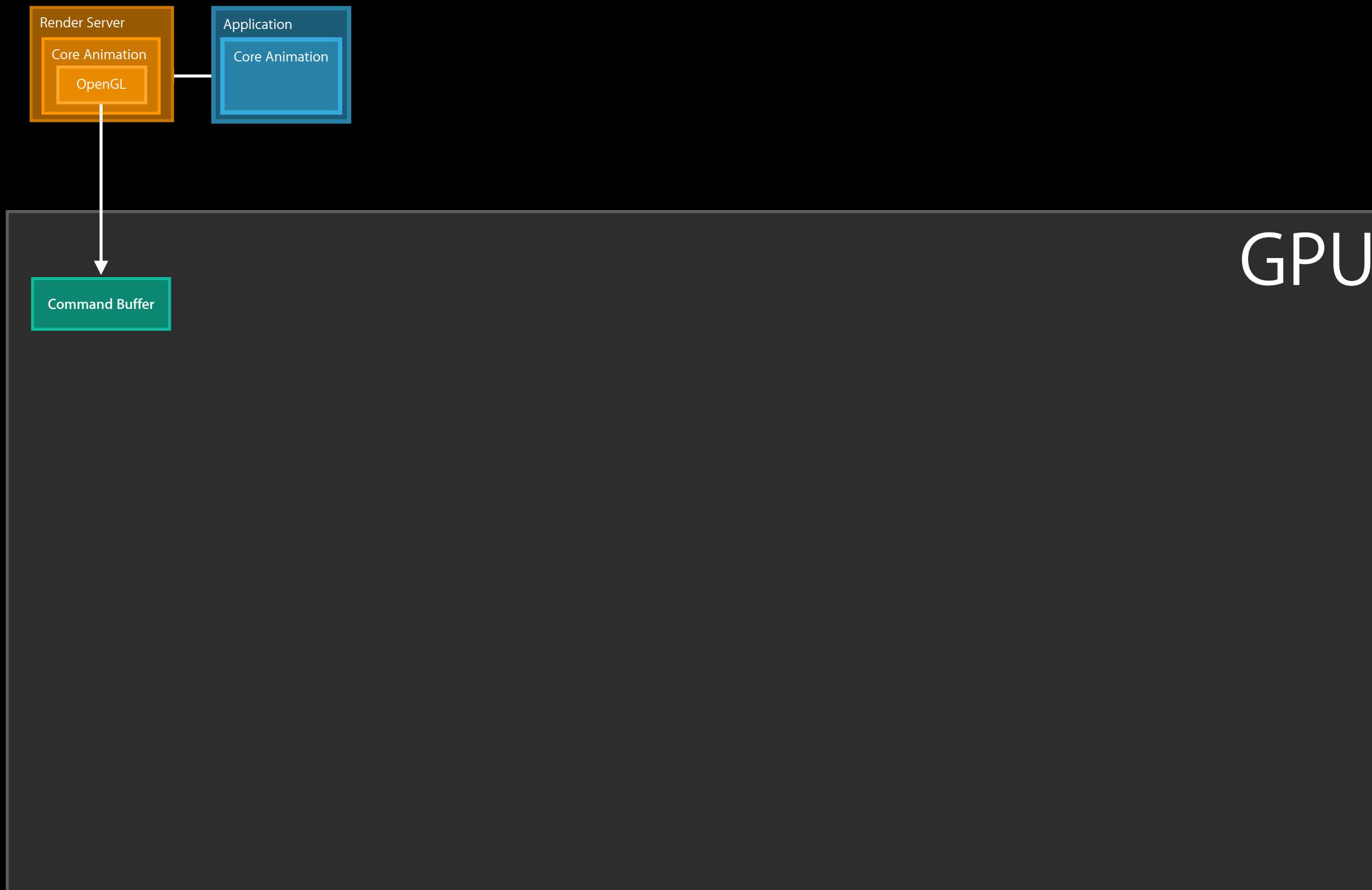
Tile Based Rendering

Rendering pass



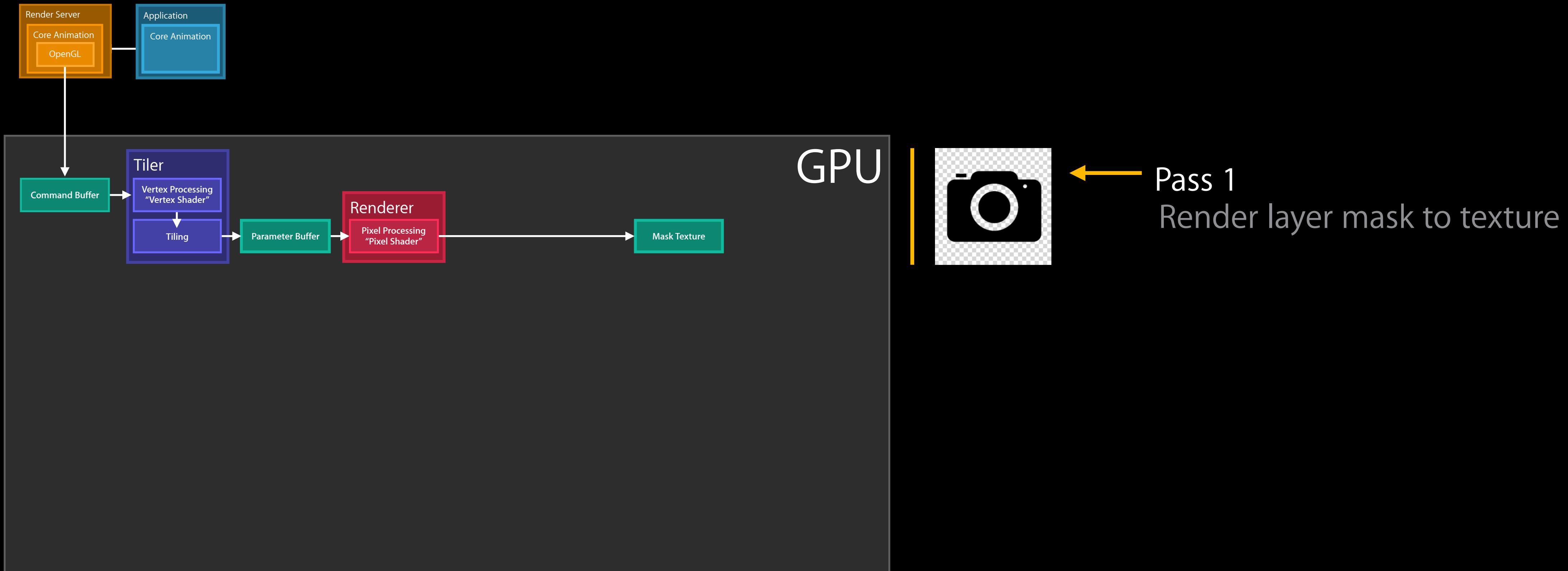
Masking

Rendering passes



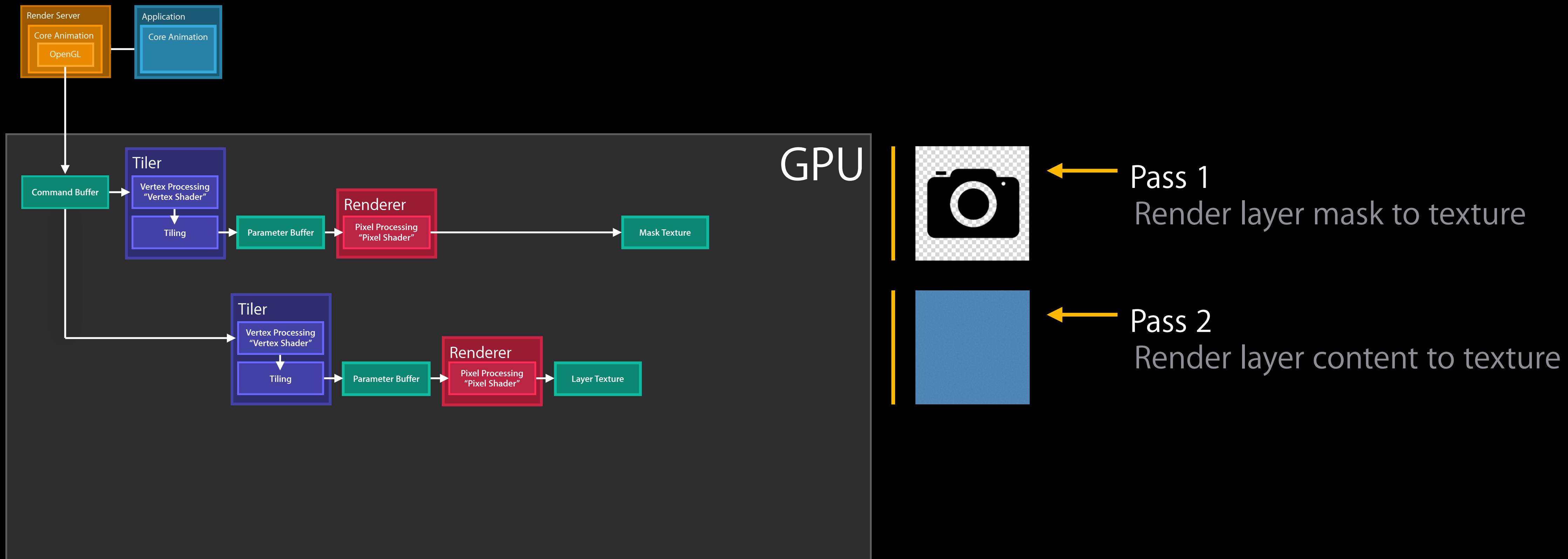
Masking

Rendering passes



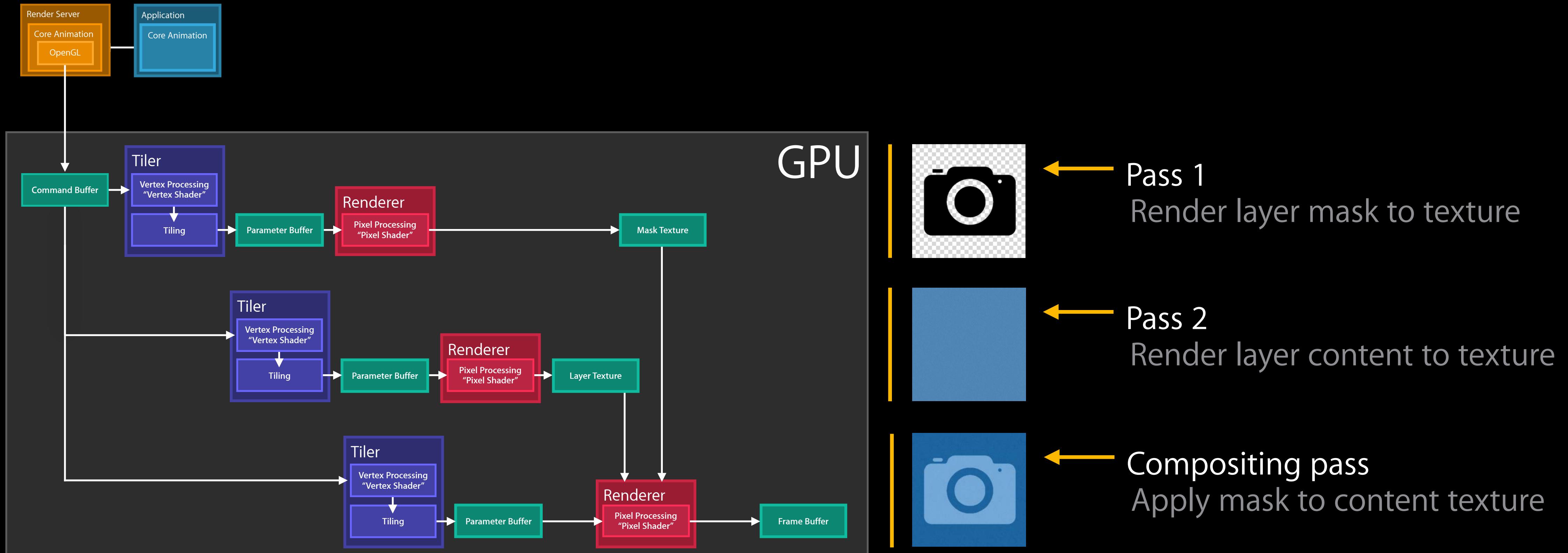
Masking

Rendering passes



Masking

Rendering passes



UIBlurEffect

Axel Wefers
iOS Software Engineer

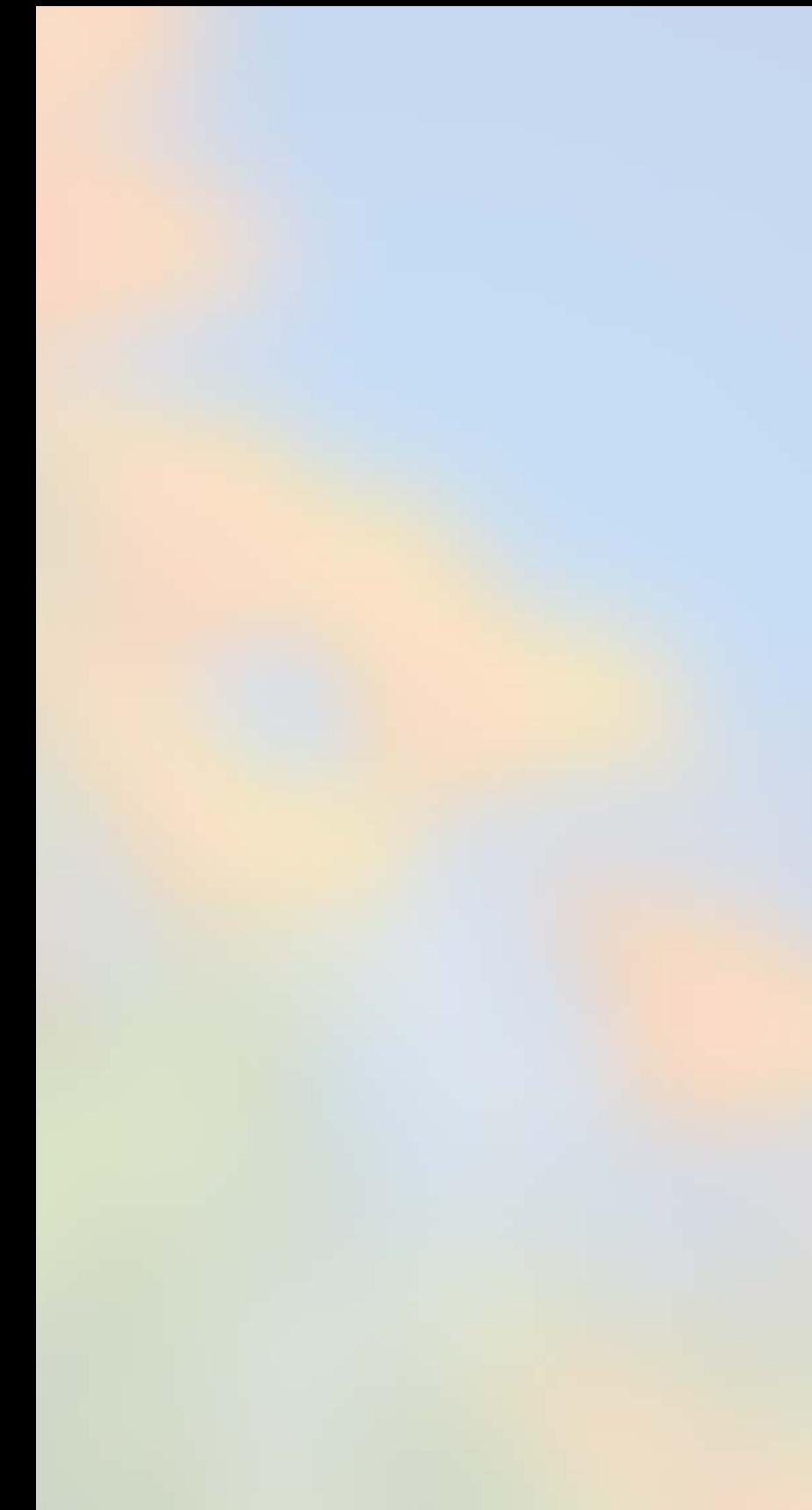
UIVisualEffectView with UIBlurEffect

UIBlurEffect styles

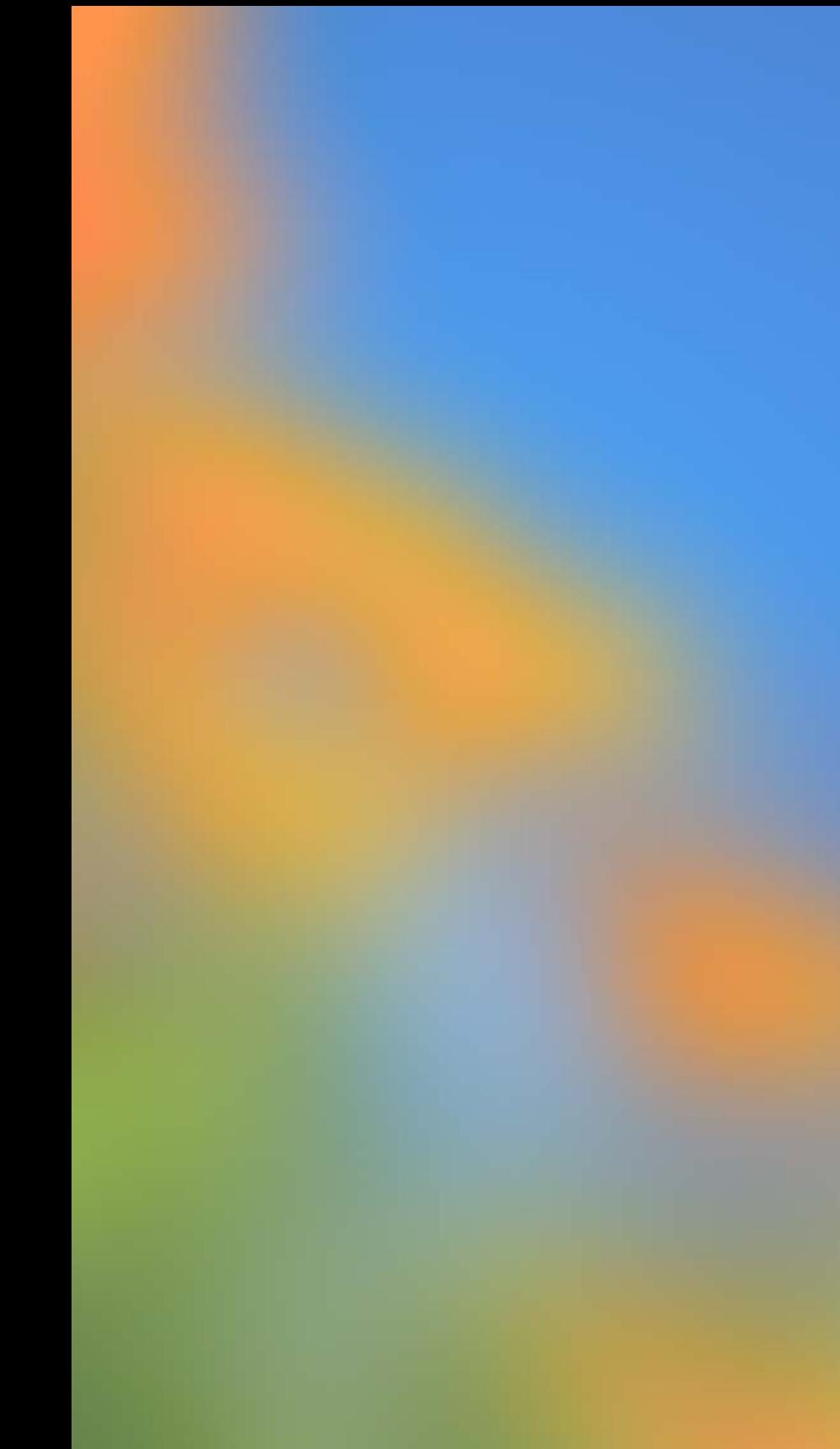
No effect



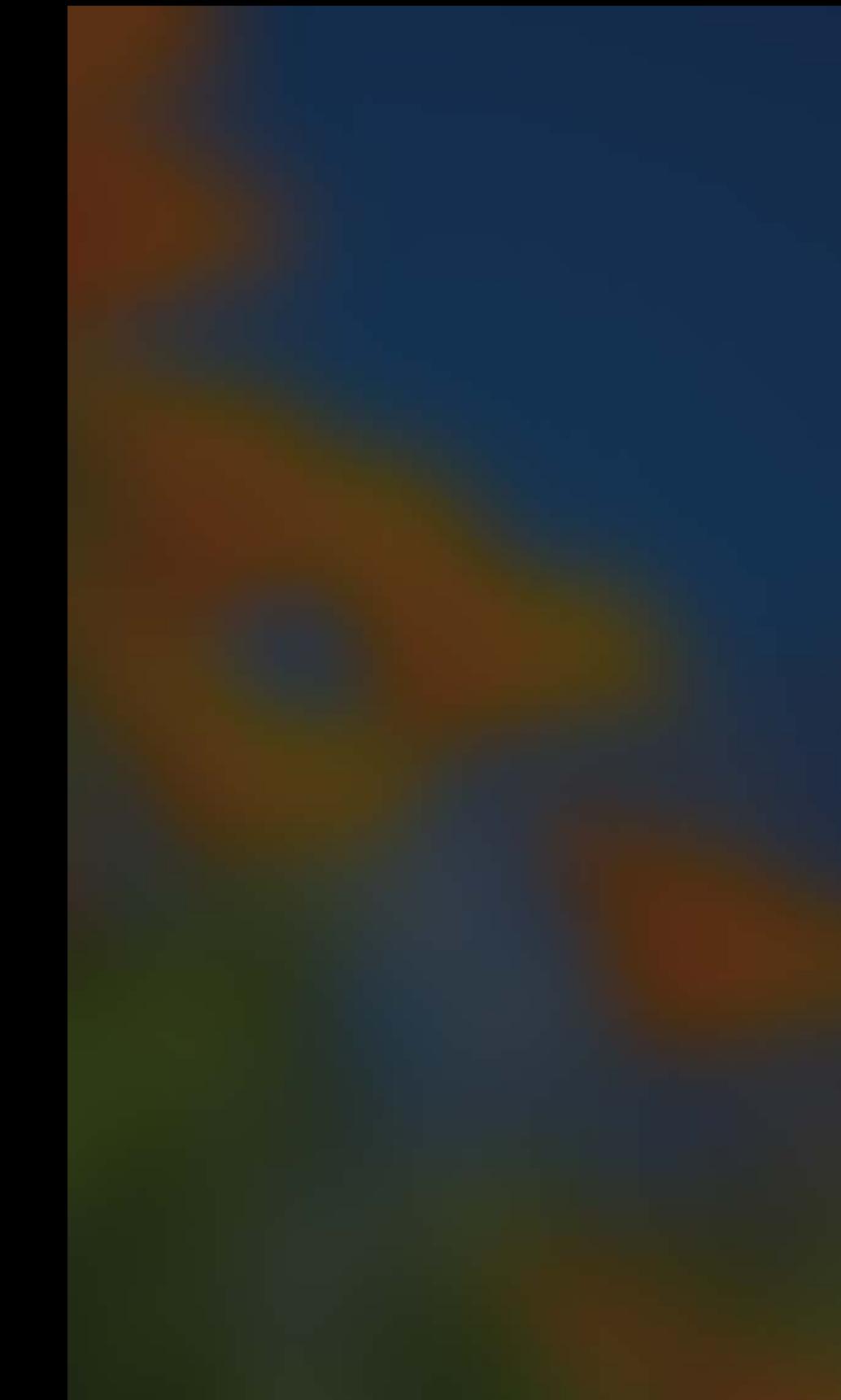
Extra light



Light

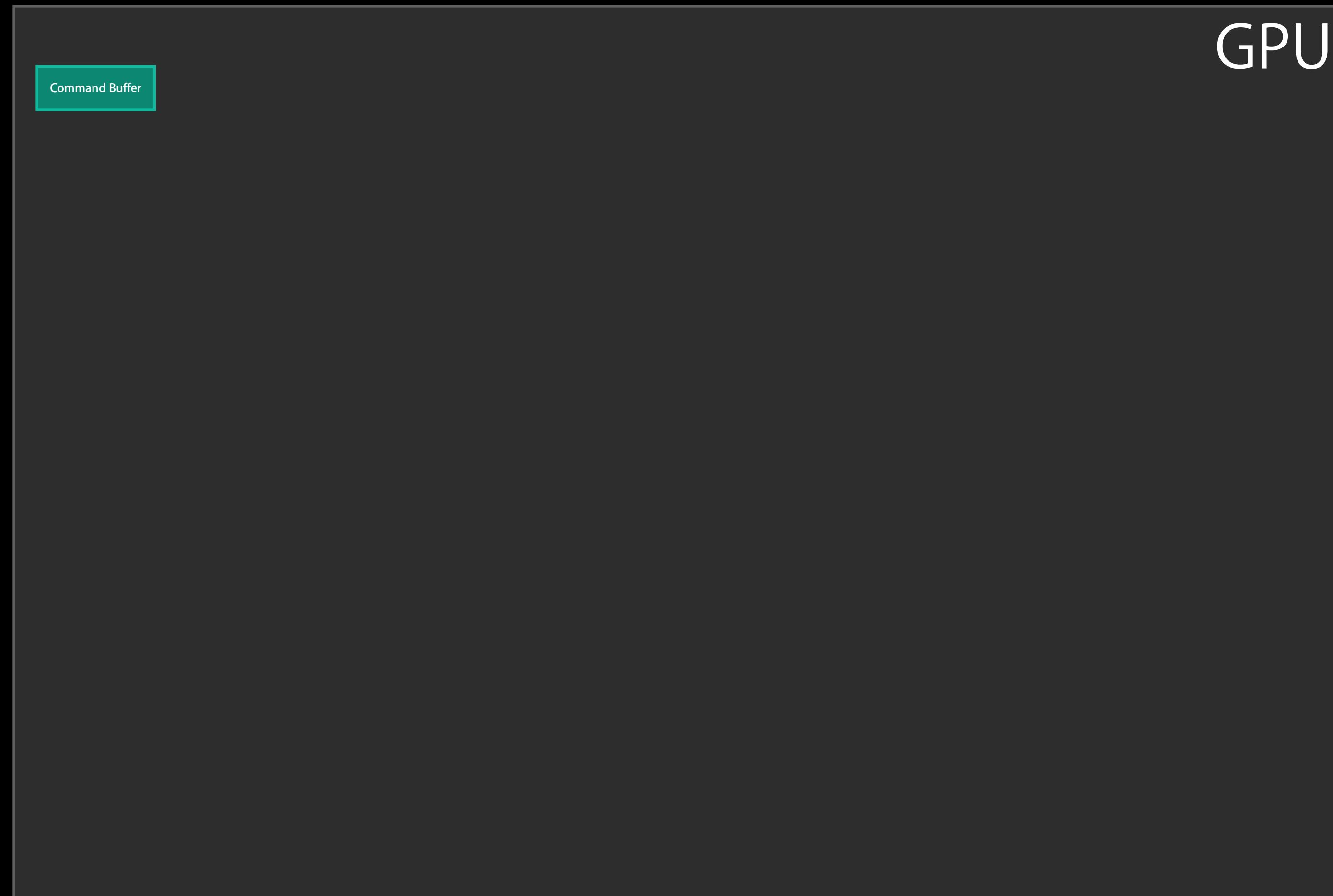


Dark



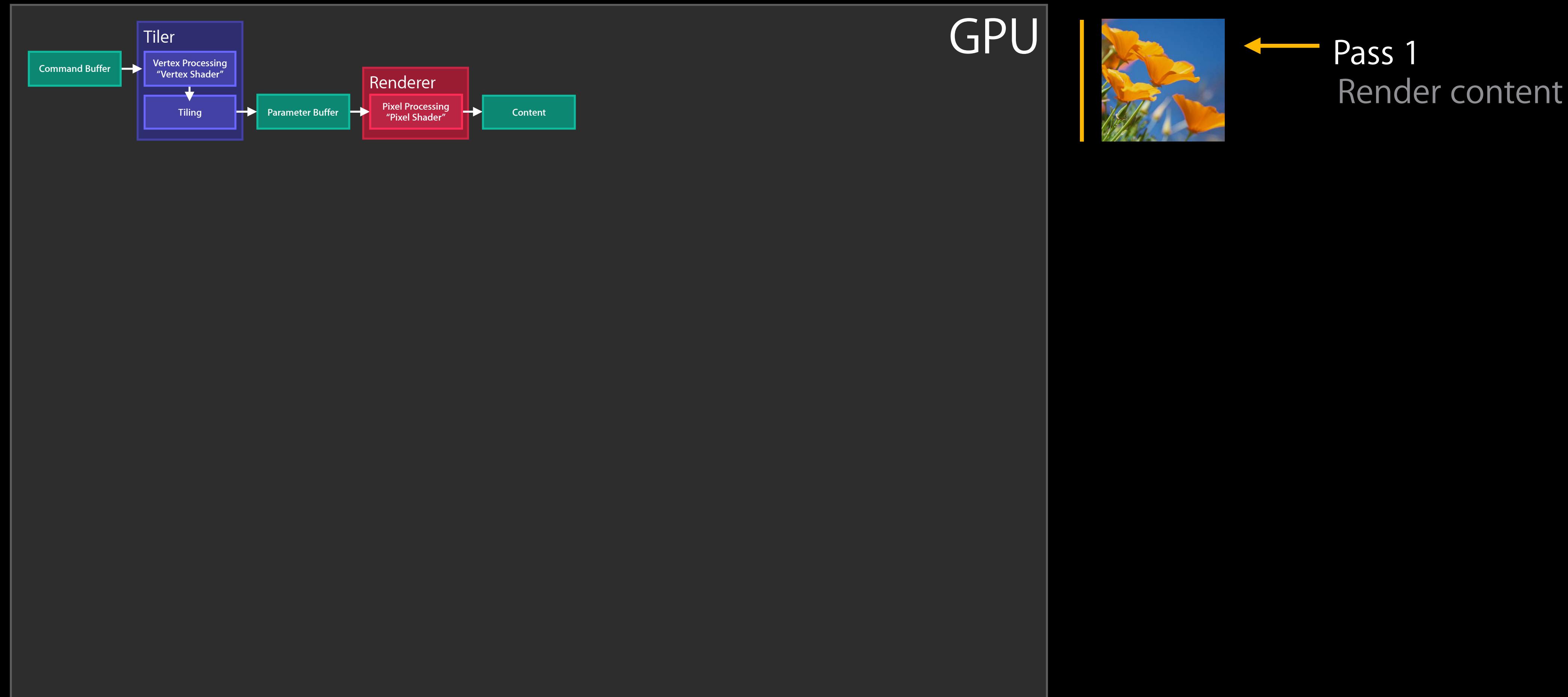
UIVisualEffectView with UIBlurEffect

Rendering passes (best case)



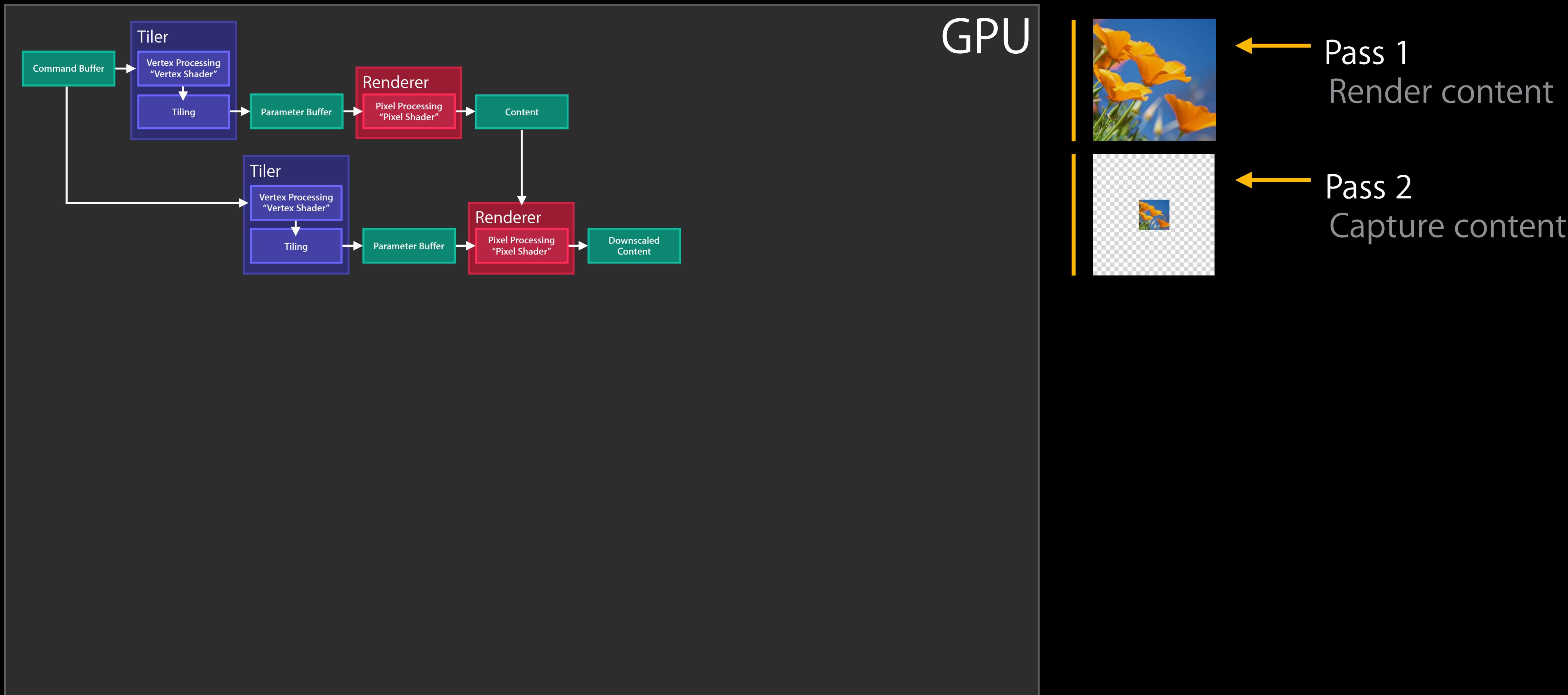
UIVisualEffectView with UIBlurEffect

Rendering passes (best case)



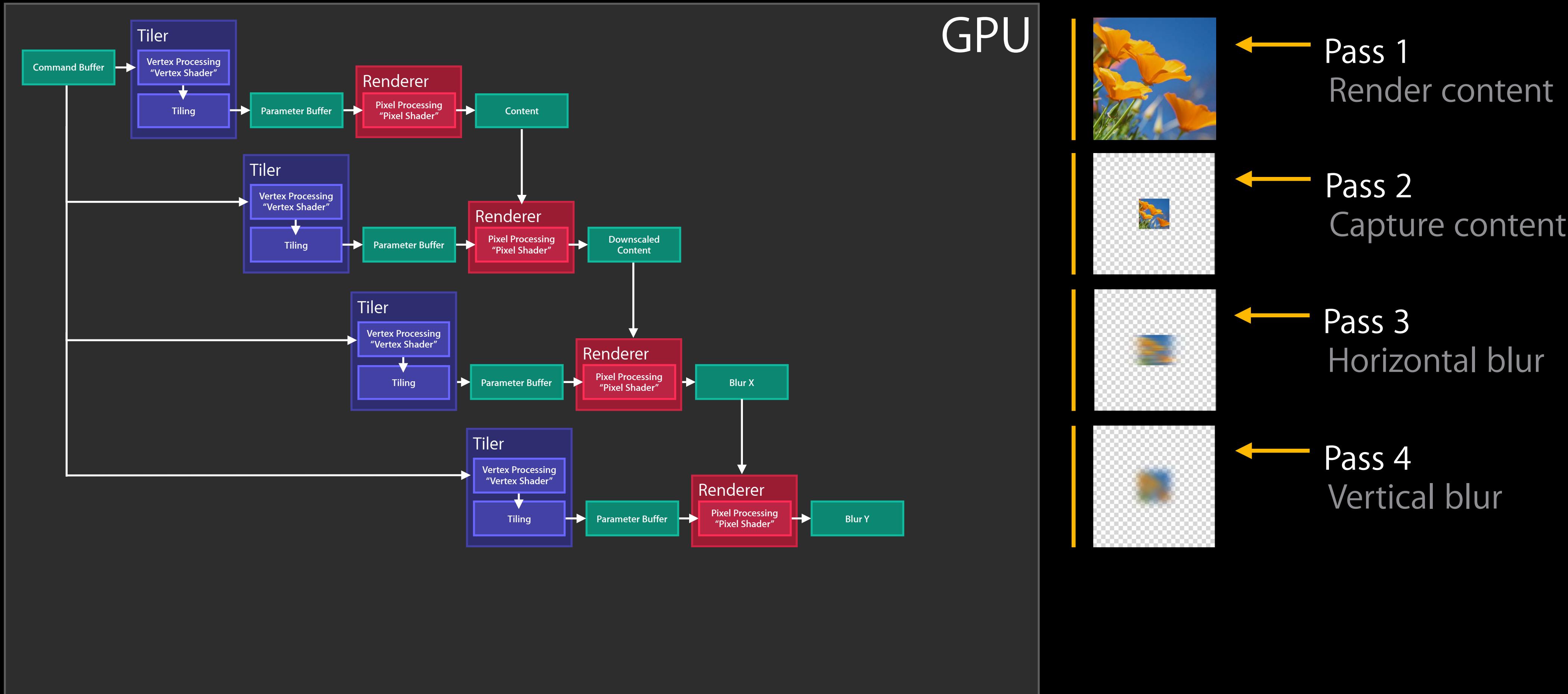
UIVisualEffectView with UIBlurEffect

Rendering passes (best case)



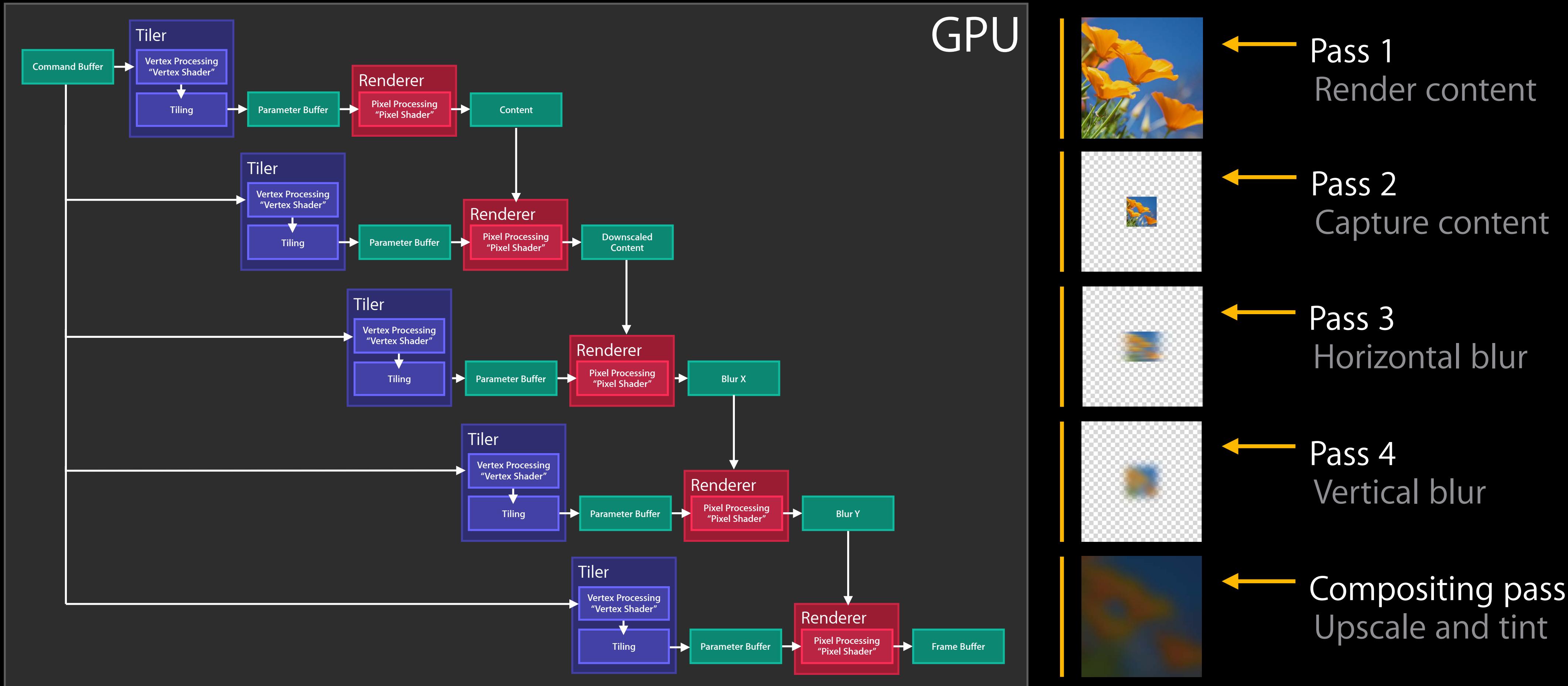
UIVisualEffectView with UIBlurEffect

Rendering passes (best case)



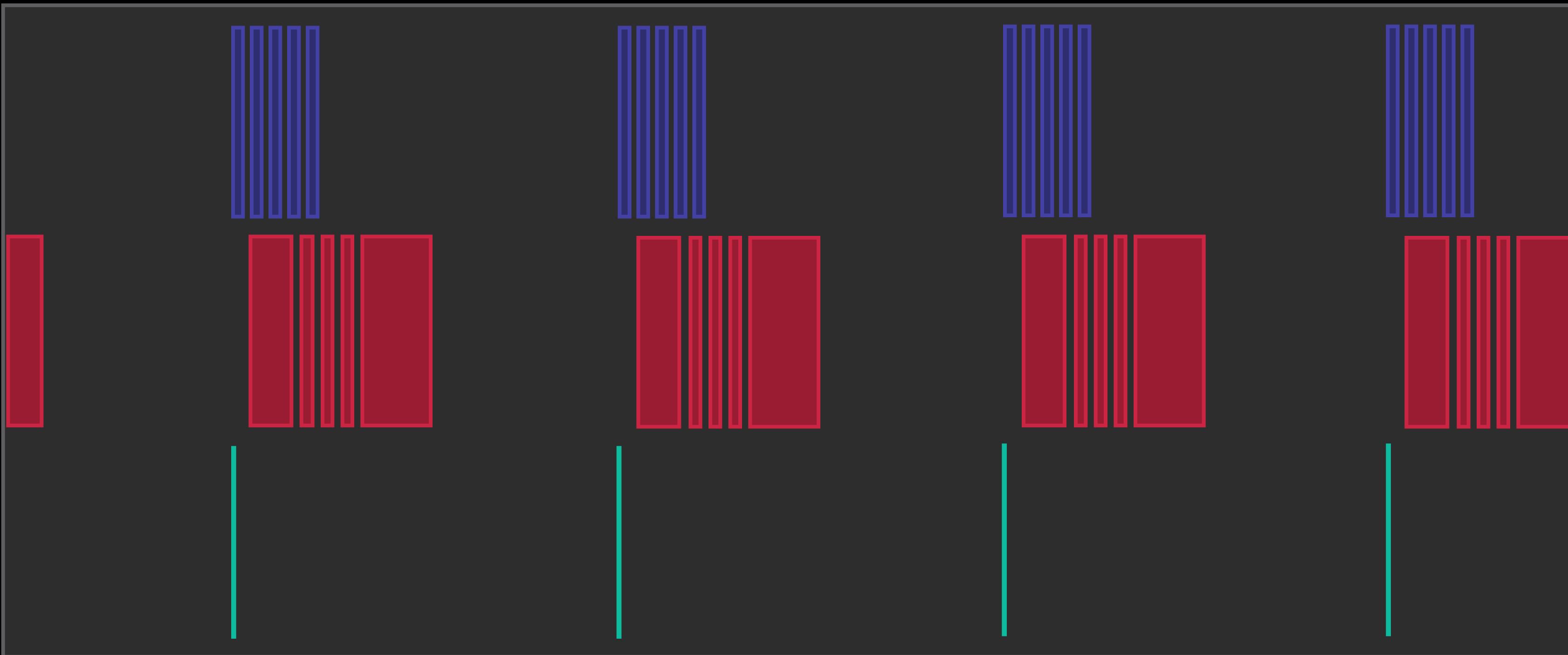
UIVisualEffectView with UIBlurEffect

Rendering passes (best case)



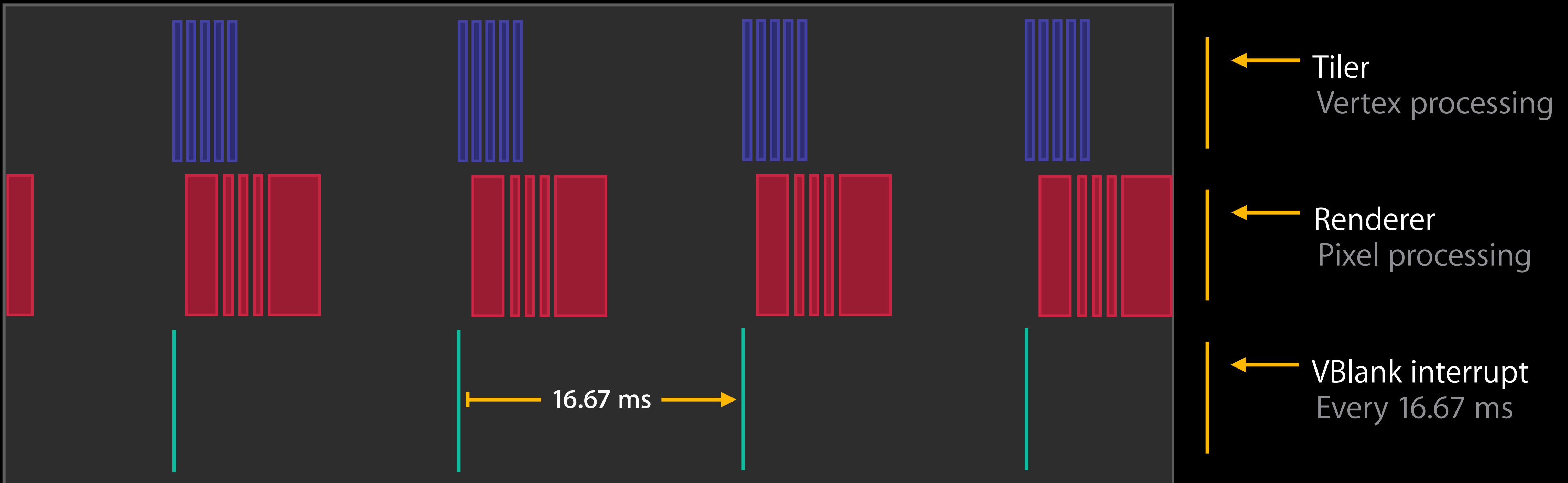
UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



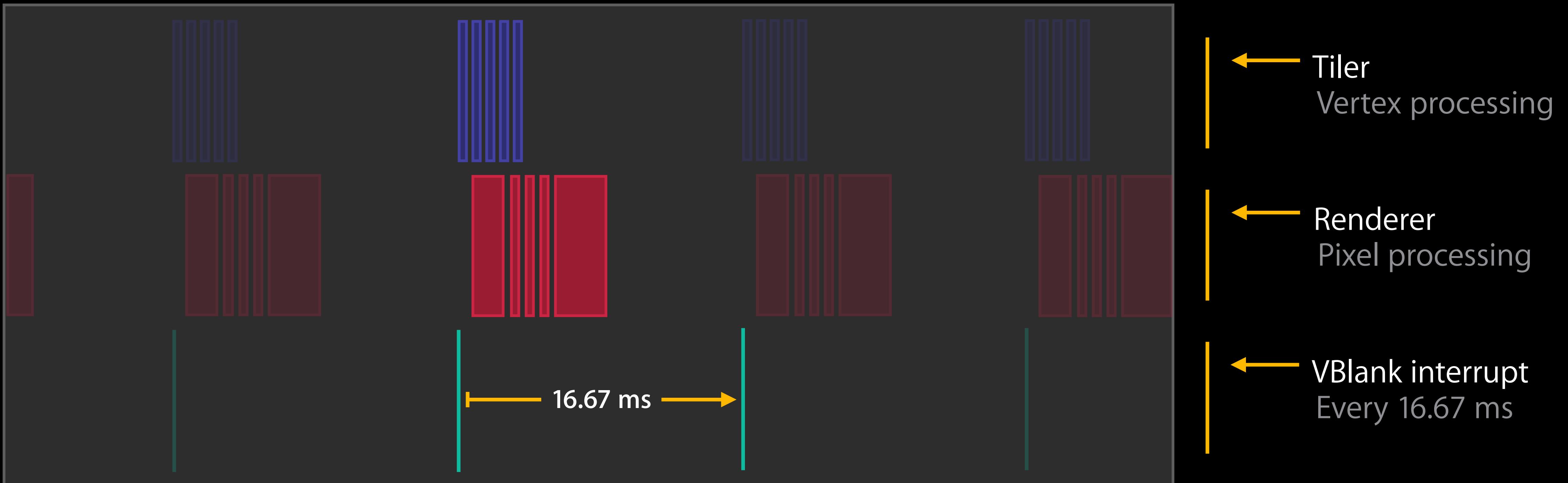
UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



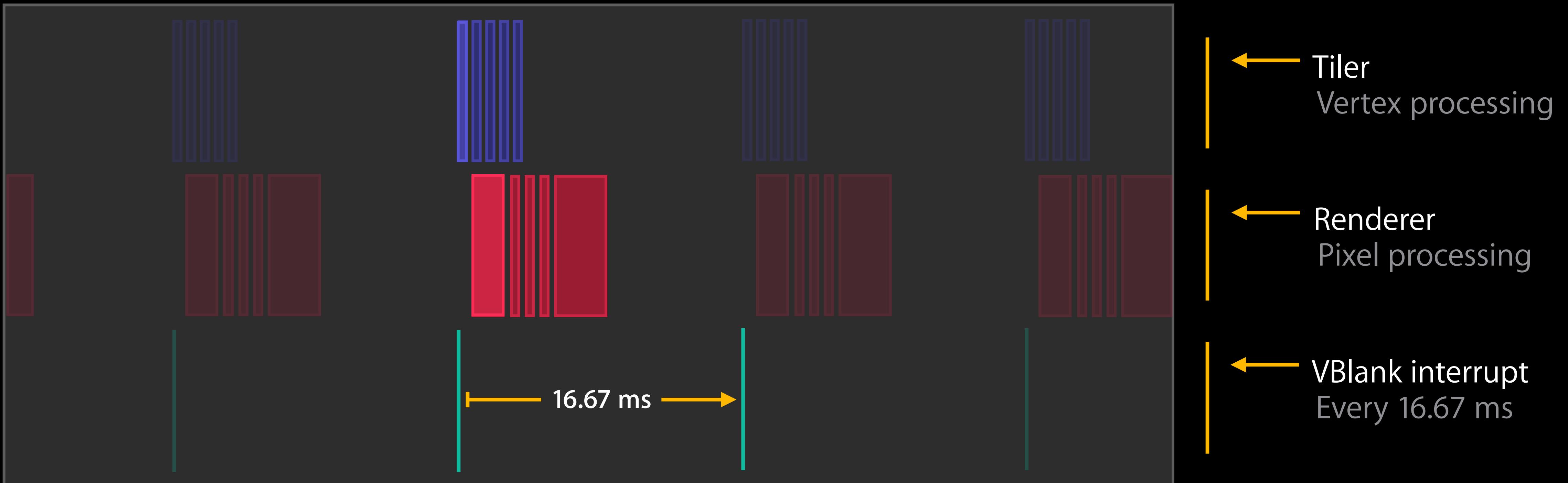
UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



UIVisualEffectView with UIBlurEffect

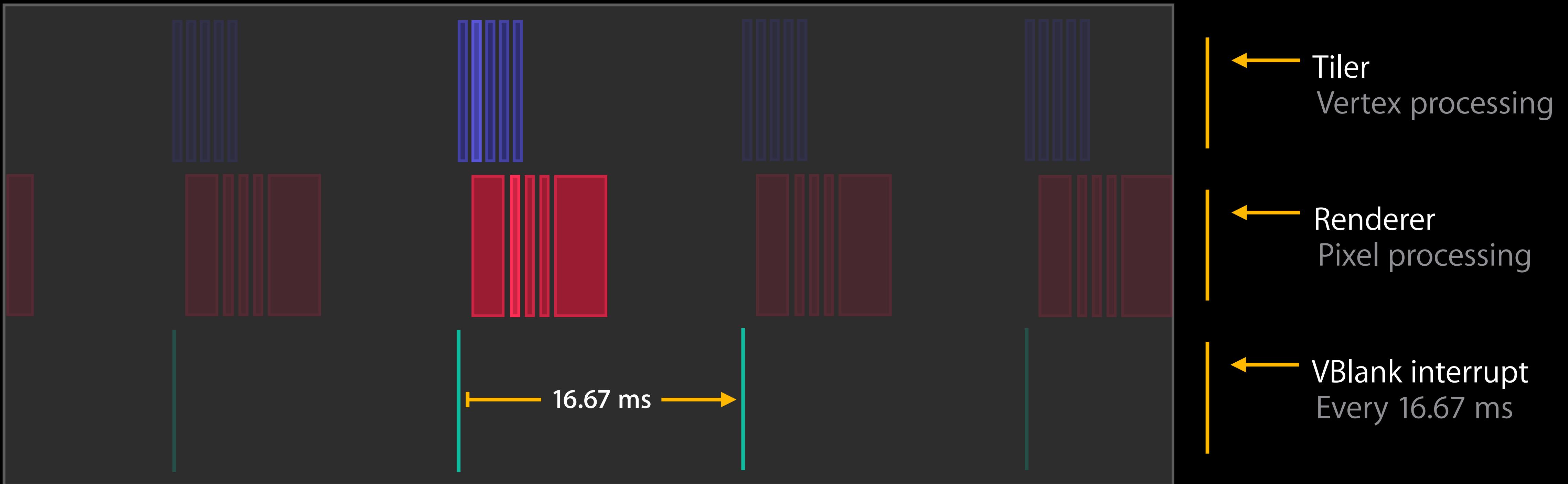
GPU utilization, fullscreen, iPad Air



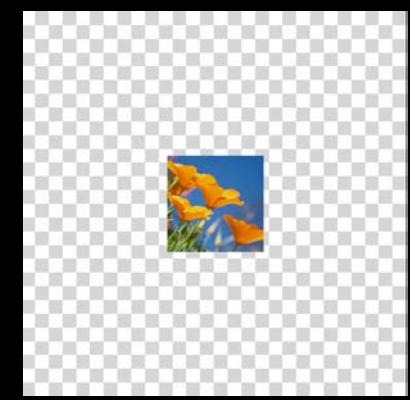
Pass 1

UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



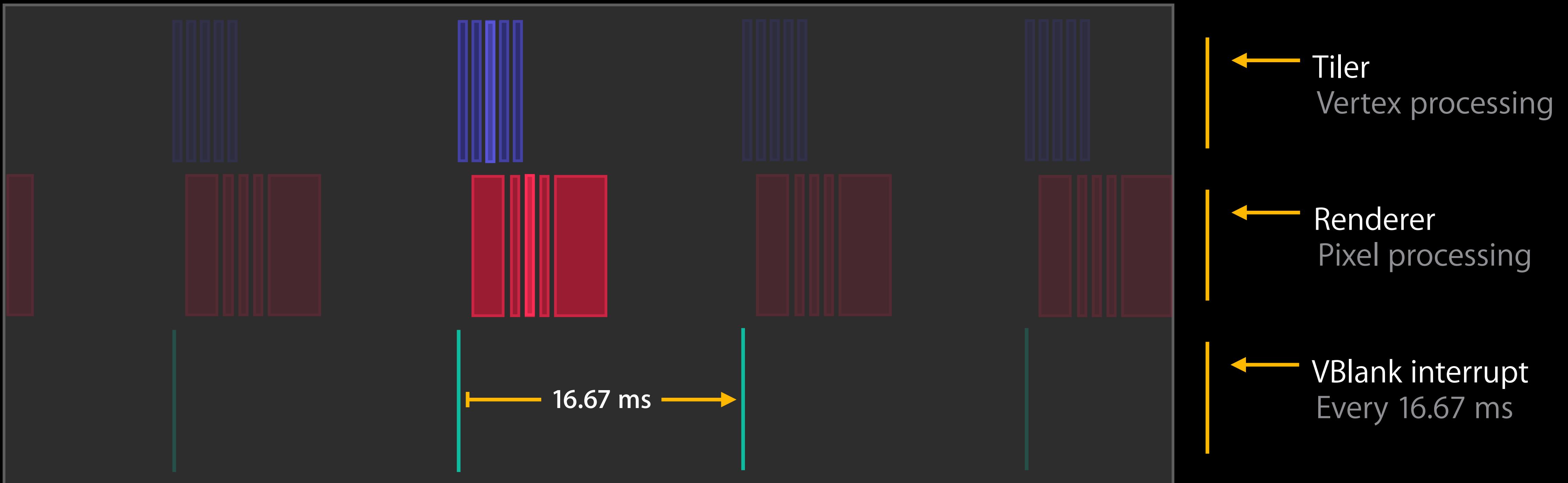
Pass 1



Pass 2

UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



Pass 1



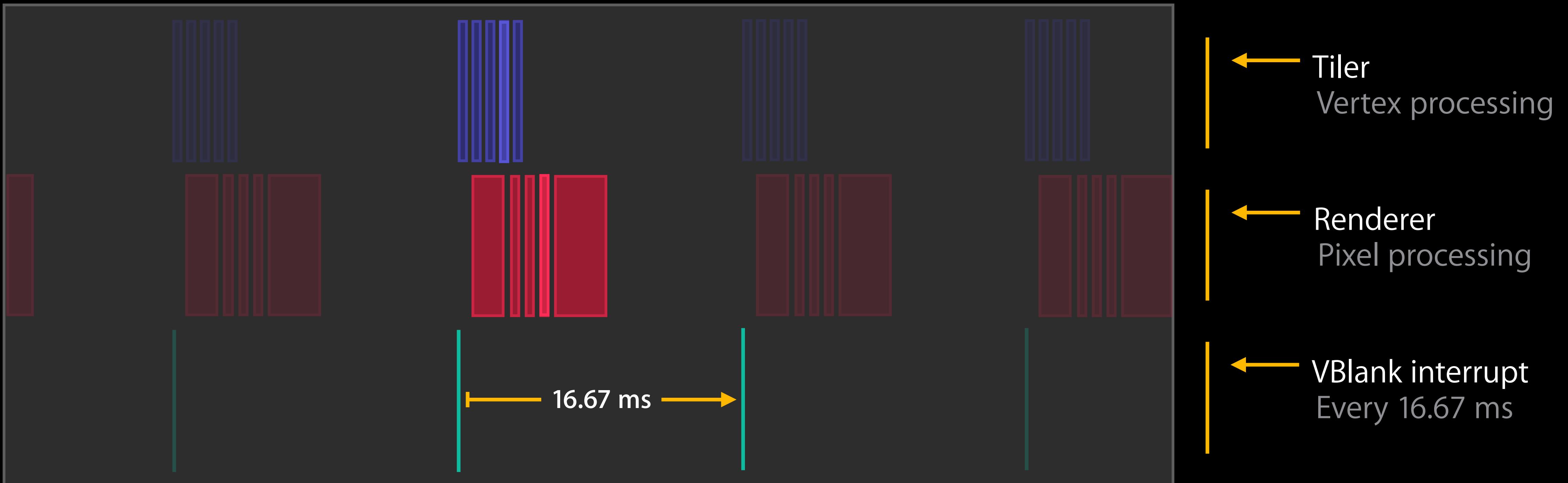
Pass 2



Pass 3

UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



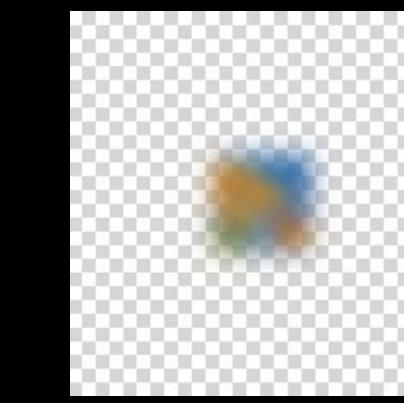
Pass 1



Pass 2



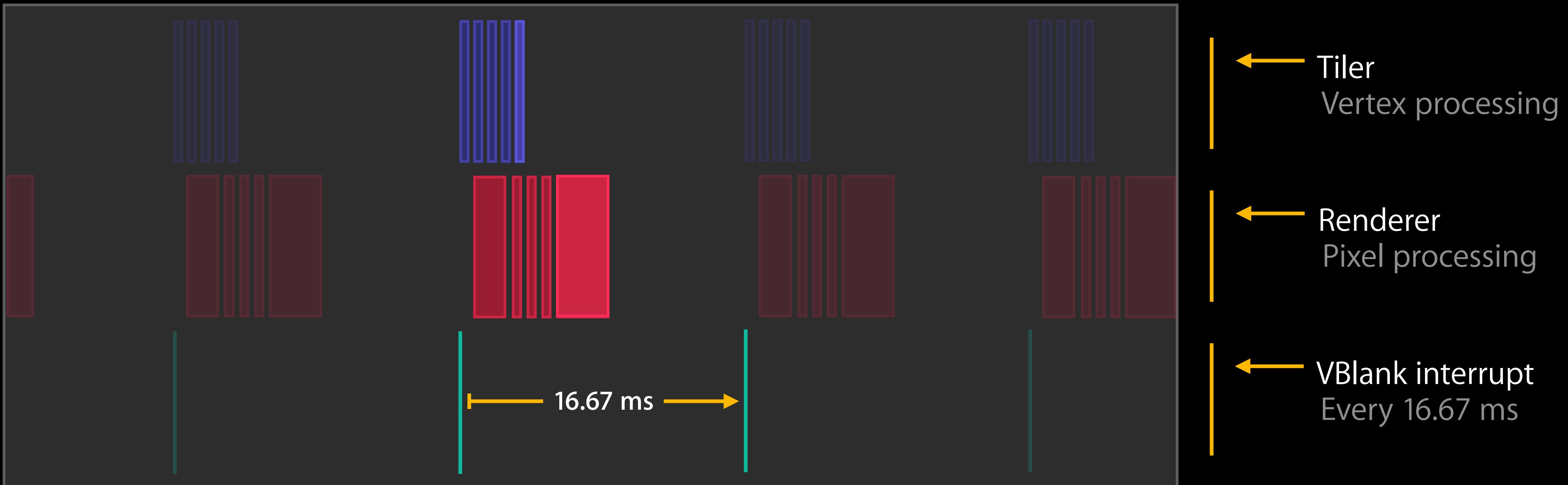
Pass 3



Pass 4

UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



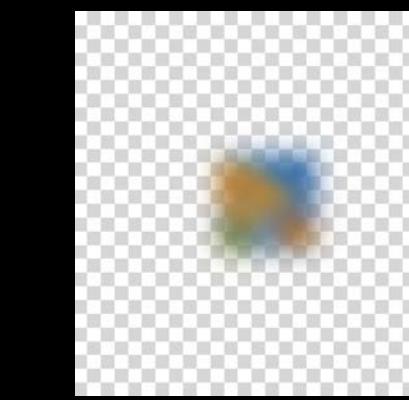
Pass 1



Pass 2



Pass 3



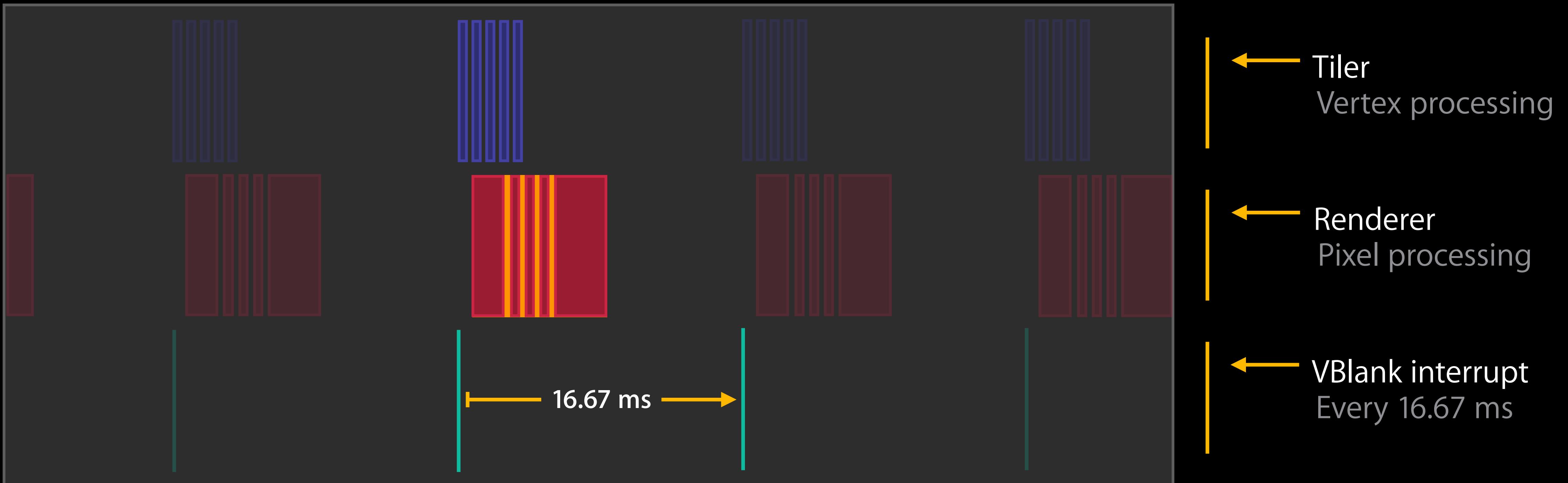
Pass 4



Pass 5

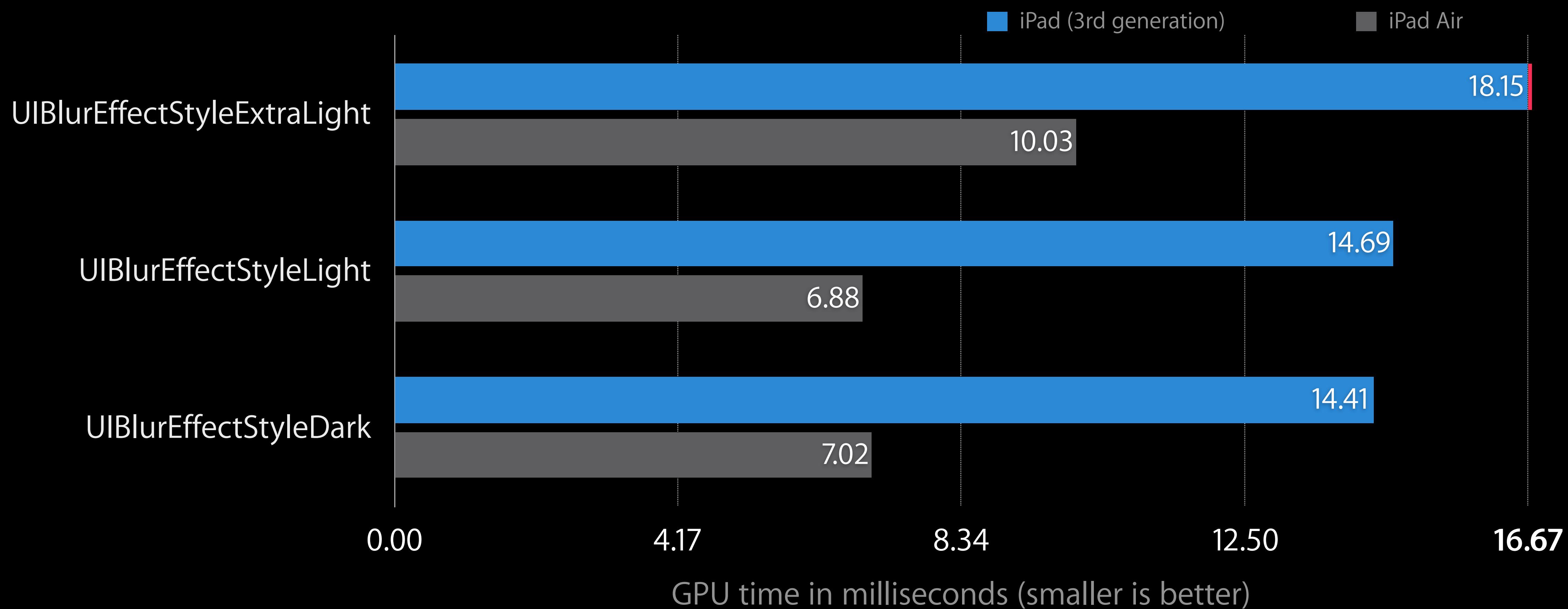
UIVisualEffectView with UIBlurEffect

GPU utilization, fullscreen, iPad Air



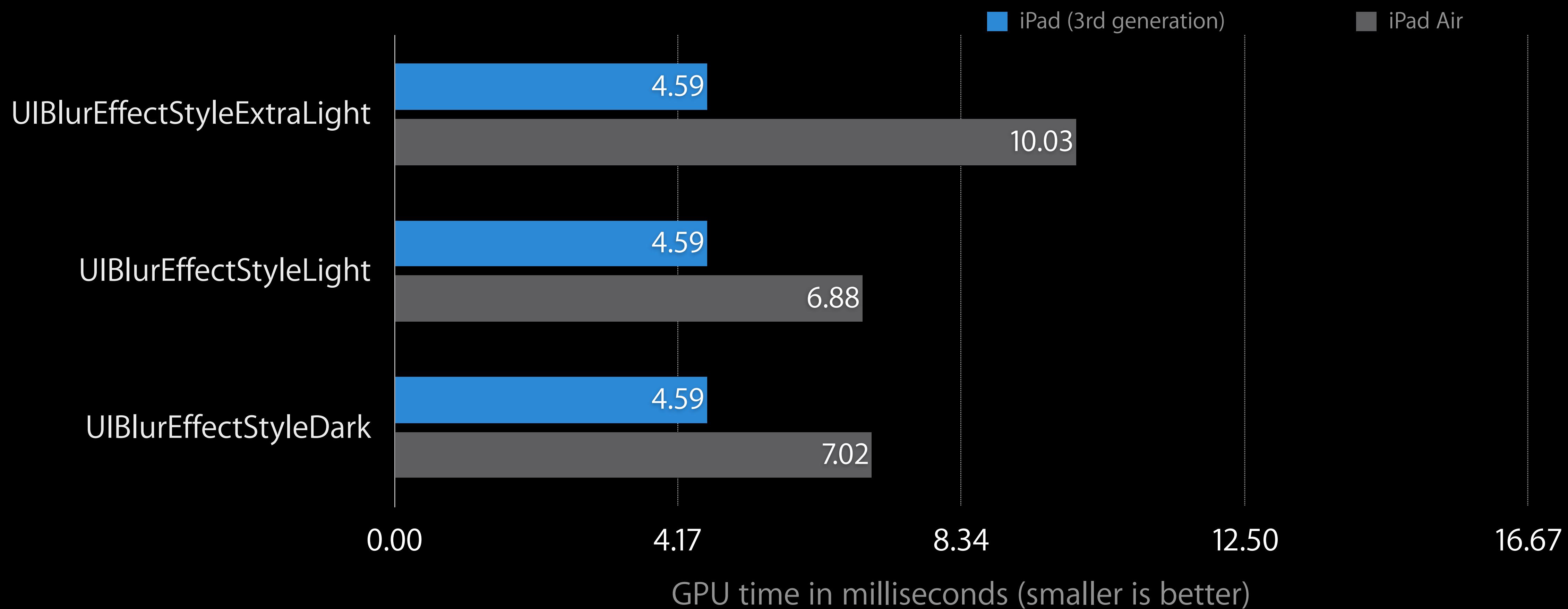
UIVisualEffectView with UIBlurEffect

FULLSCREEN performance



UIVisualEffectView with UIBlurEffect

FULLSCREEN performance



UIVisualEffectView with UIBlurEffect

UIBlurEffect support

Device	Blur	Tint
iPad 2	✗	✓
iPad (3rd generation)	✗	✓
iPad (4th generation)	✓	✓
iPad Air	✓	✓
iPad mini	✓	✓
iPad mini Retina display	✓	✓
All iPhones	✓	✓
iPod touch	✓	✓

UIVisualEffectView with UIBlurEffect

Performance considerations

UIBlurEffect adds multiple offscreen passes depending on style

Only dirty regions are redrawn

Effect is very costly

- UI can easily be GPU bound
- Keep bounds of view as small as possible
- Make sure to budget for effect

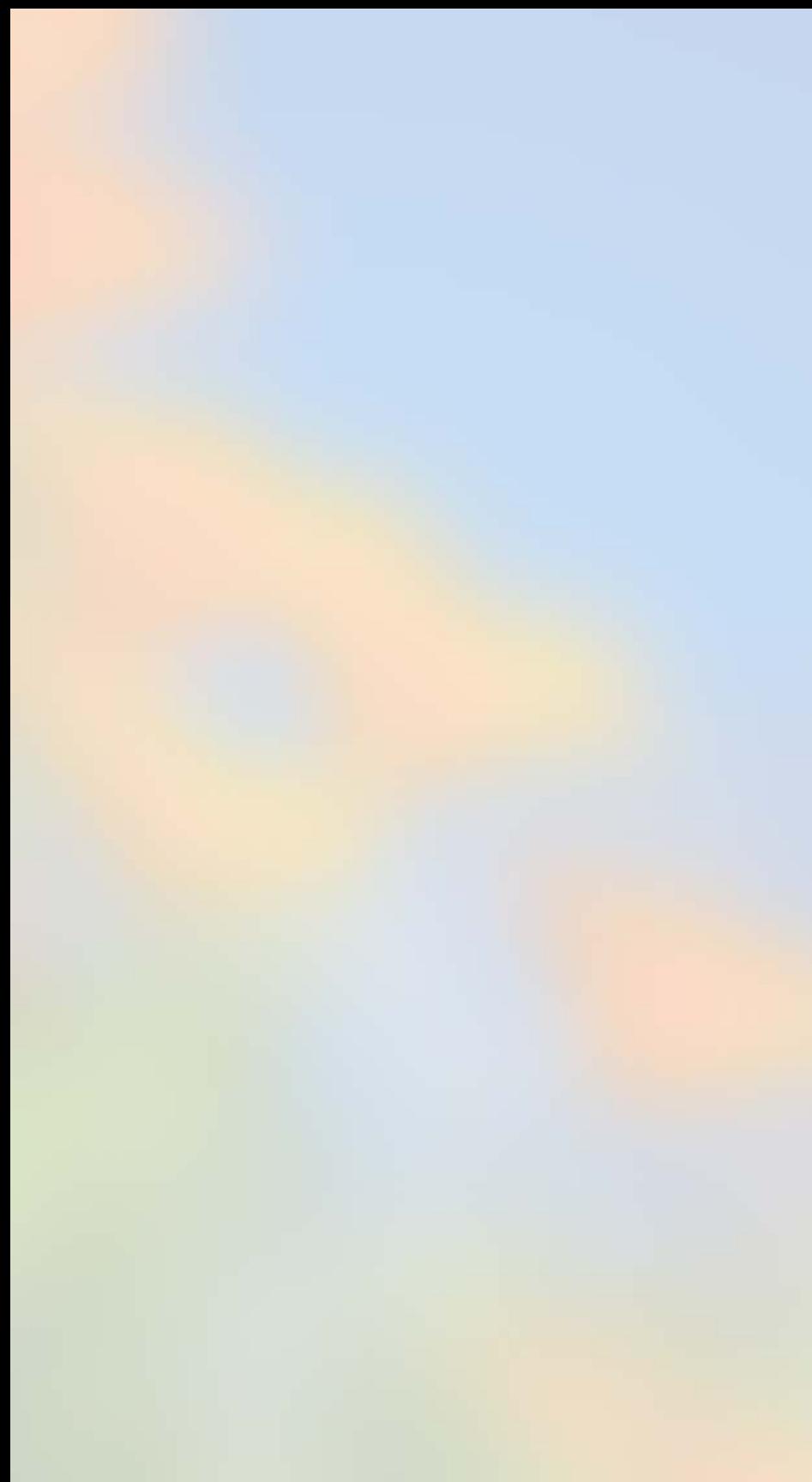
UIVibrancyEffect

Axel Wefers
iOS Software Engineer

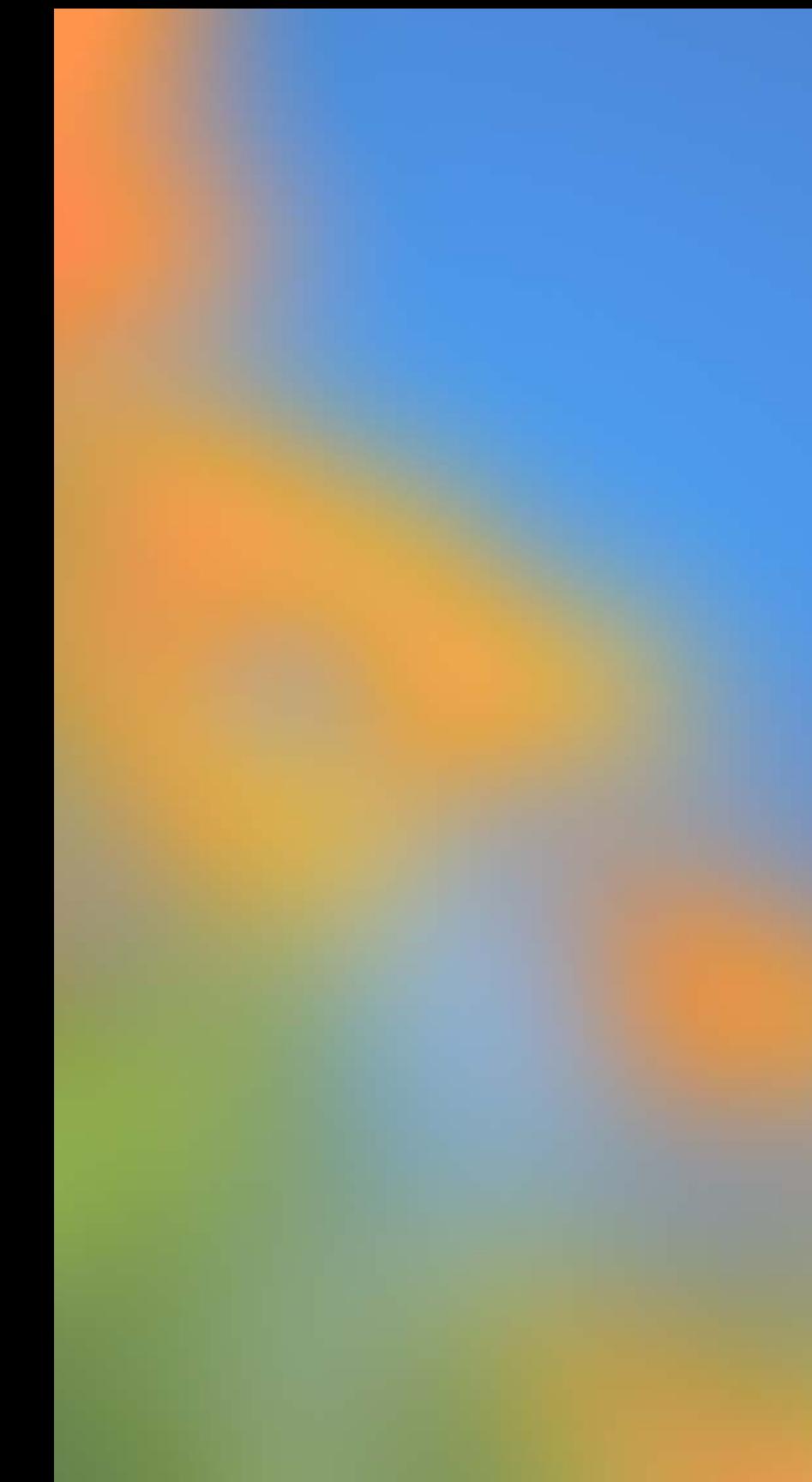
UIVisualEffectView with UIVibrancyEffect

UIVibrancyEffect styles

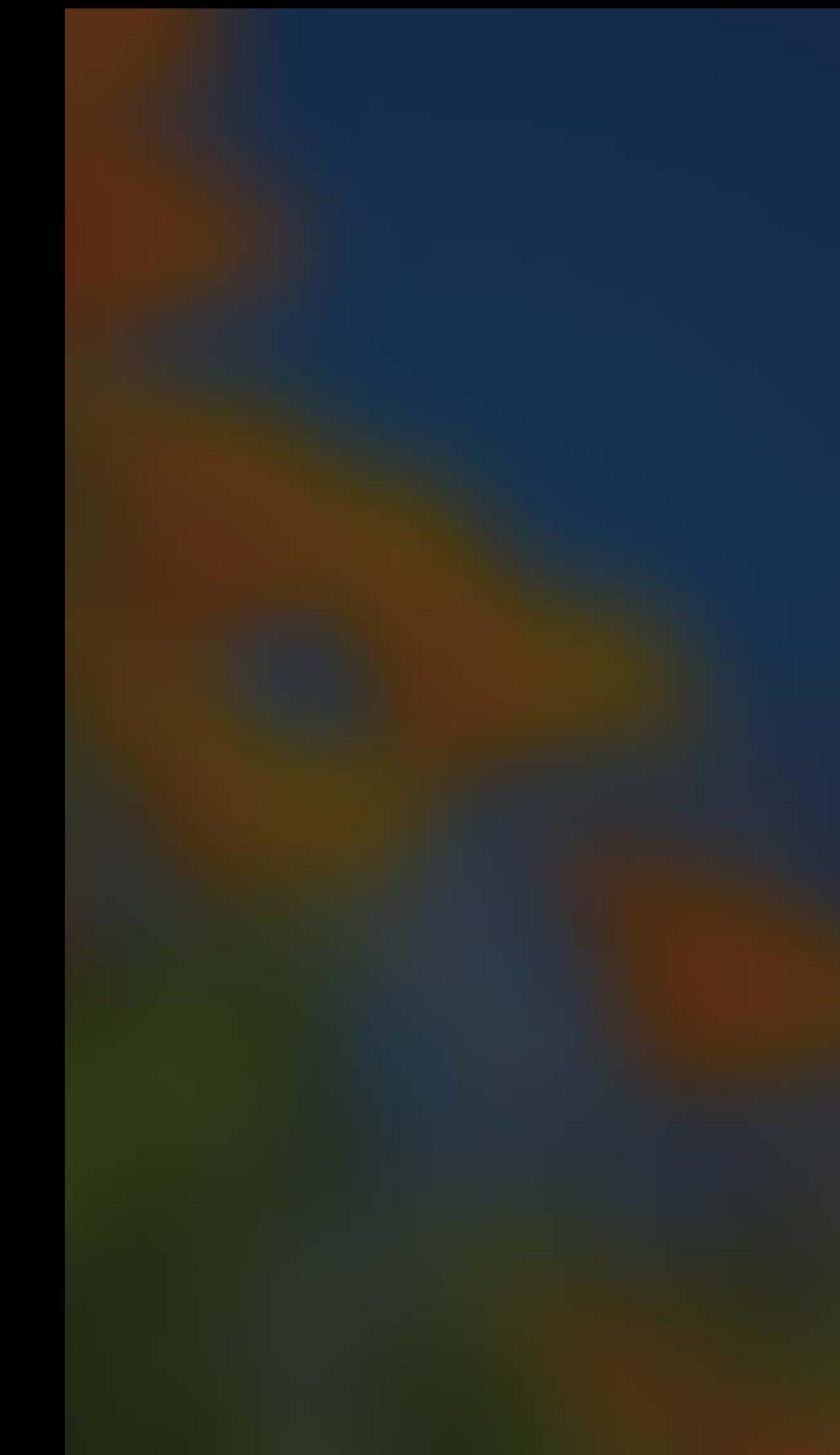
Extra light



Light



Dark



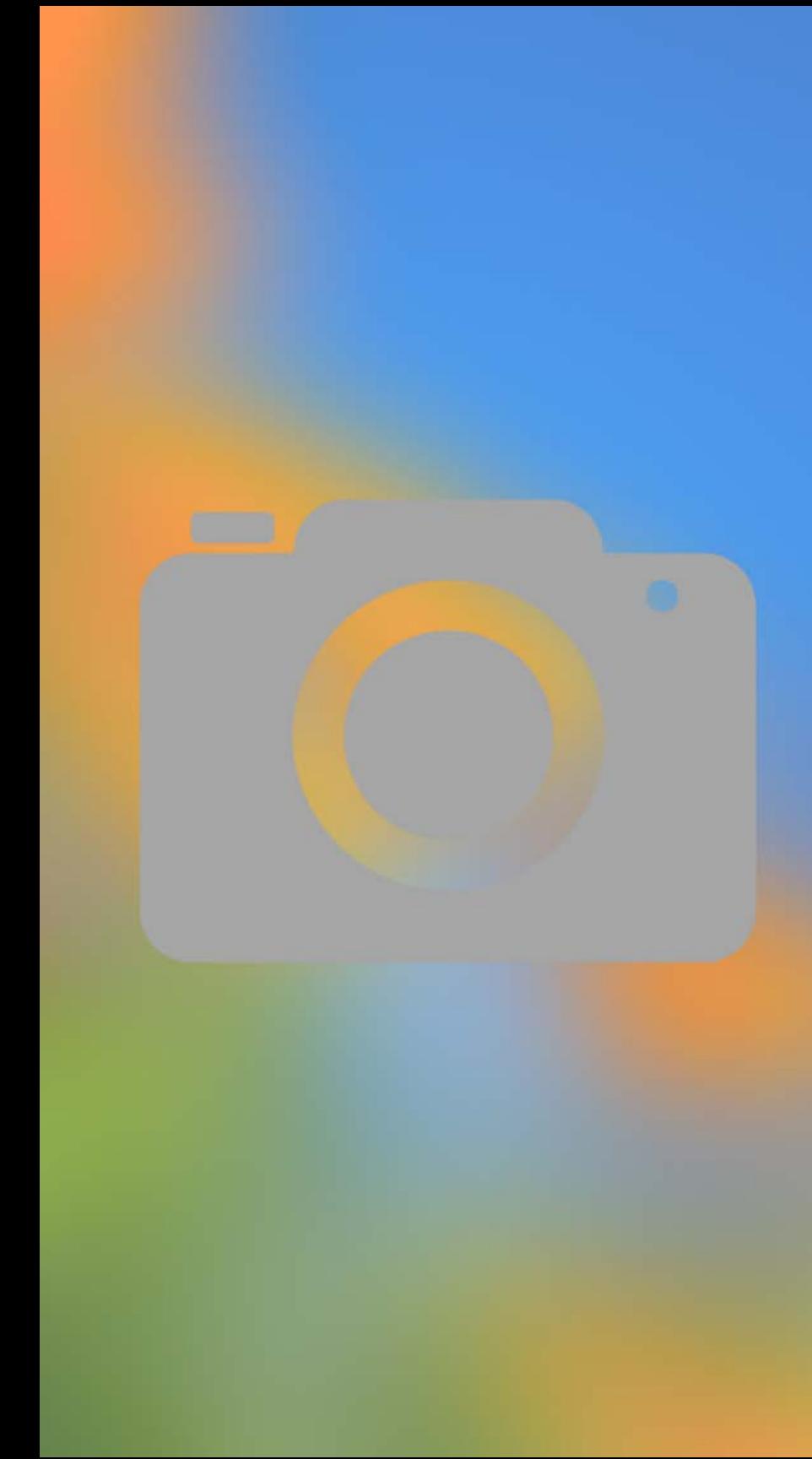
UIVisualEffectView with UVibrancyEffect

UVibrancyEffect styles

Extra light



Light



Dark



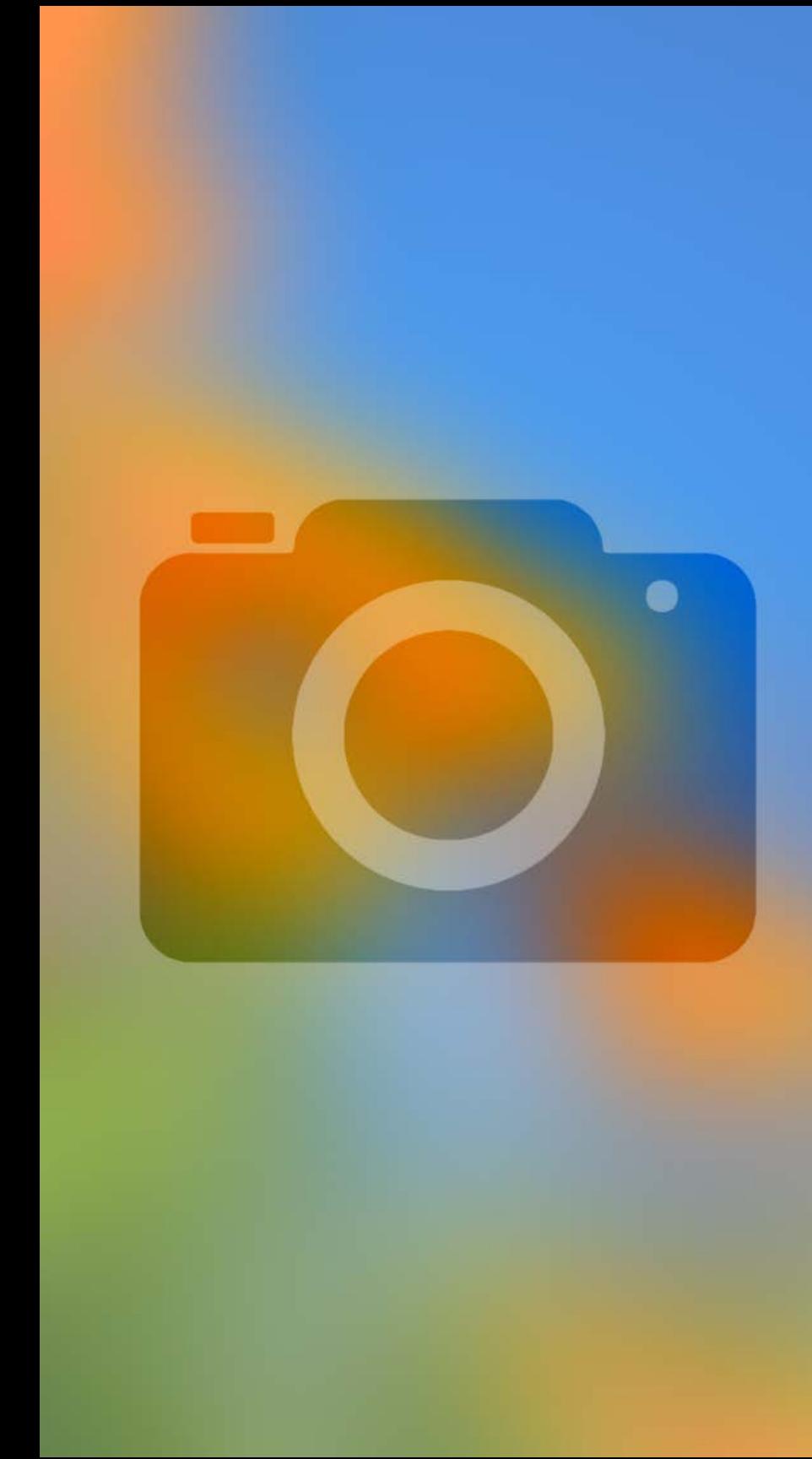
UIVisualEffectView with UVibrancyEffect

UVibrancyEffect styles

Extra light



Light

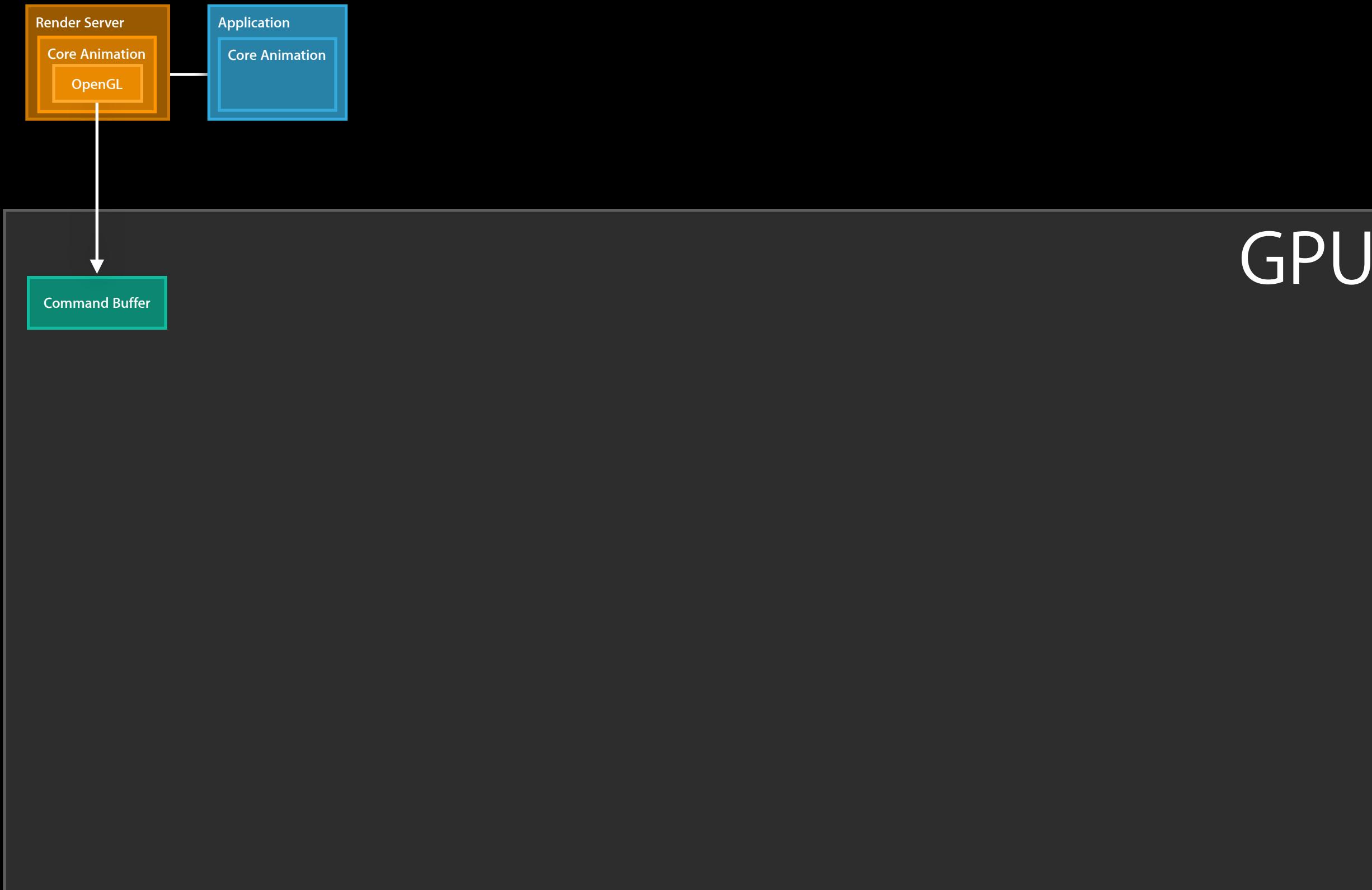


Dark



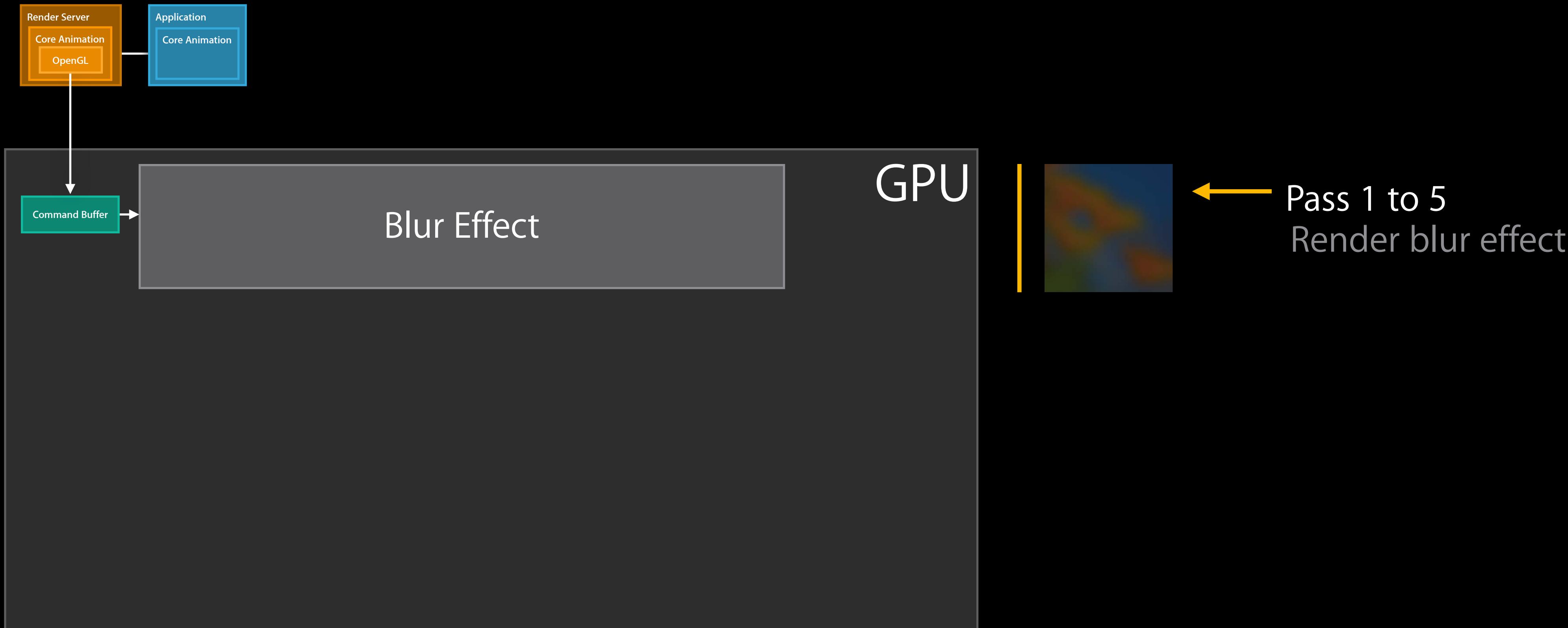
UIVisualEffectView with UIVibrancyEffect

Rendering passes



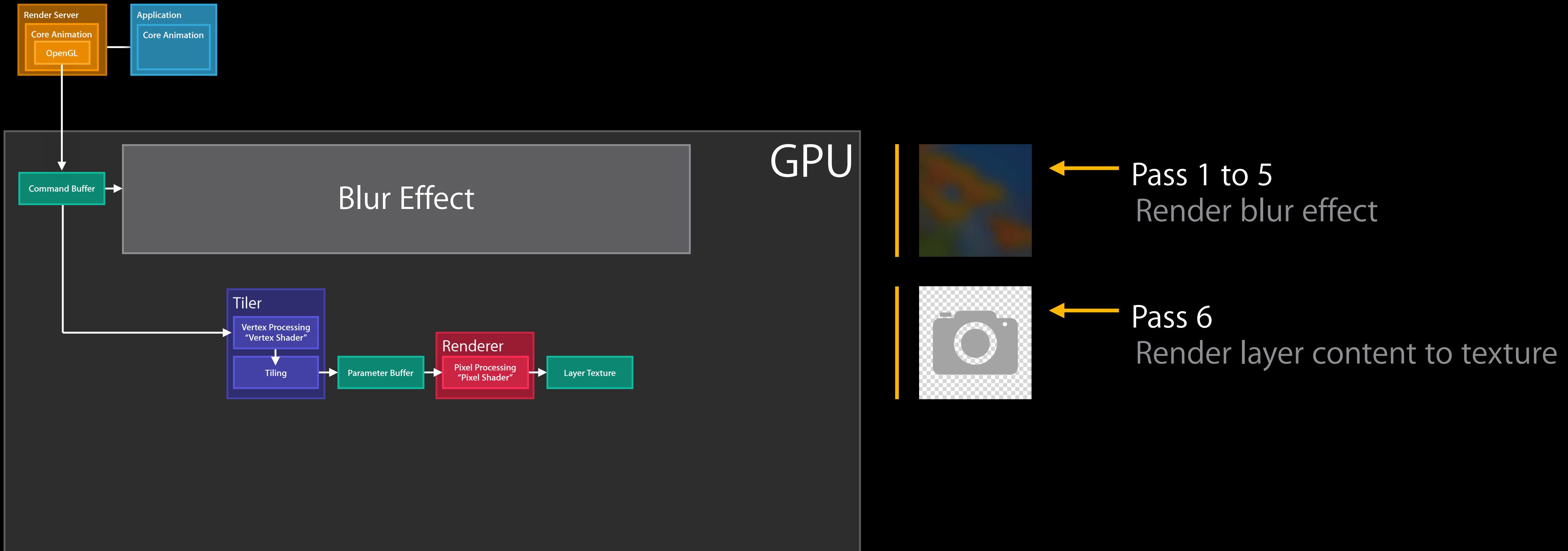
UIVisualEffectView with UIVibrancyEffect

Rendering passes



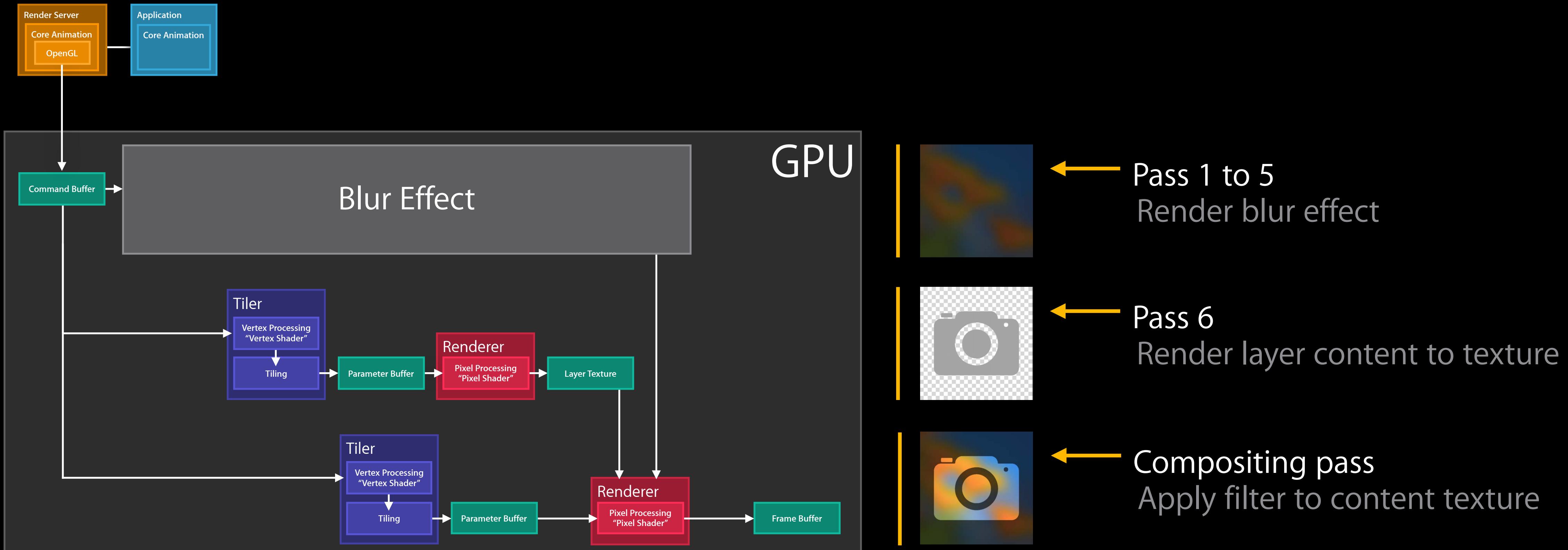
UIVisualEffectView with UIVibrancyEffect

Rendering passes



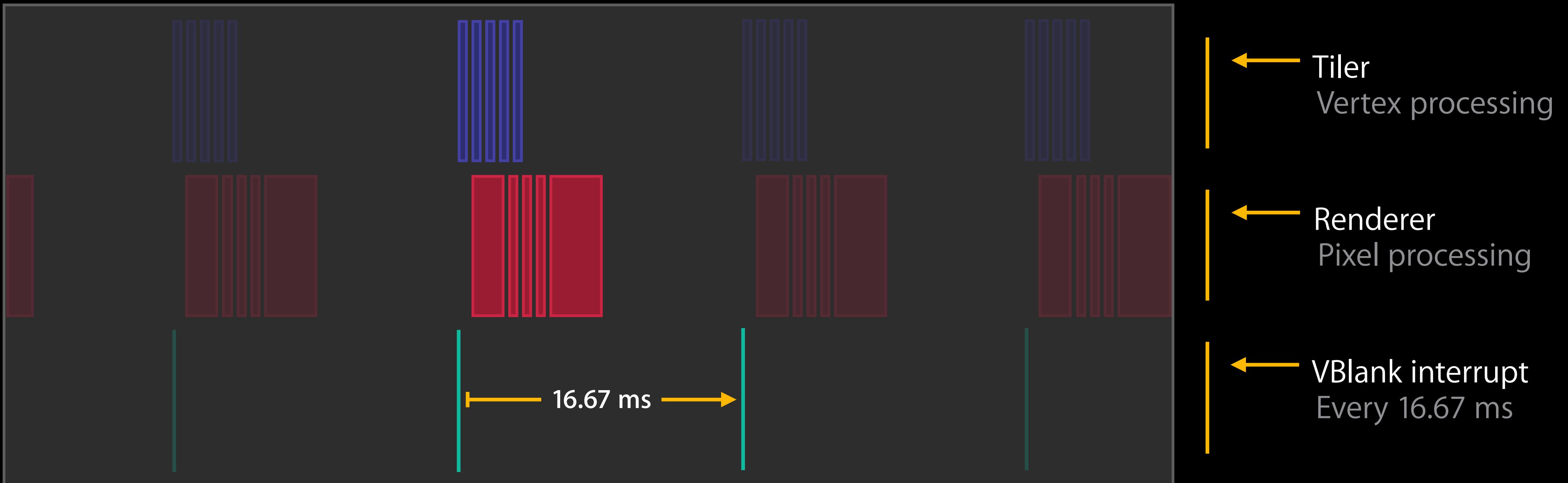
UIVisualEffectView with UIVibrancyEffect

Rendering passes



UIVisualEffectView with UIVibrancyEffect

GPU utilization, fullscreen, iPad Air



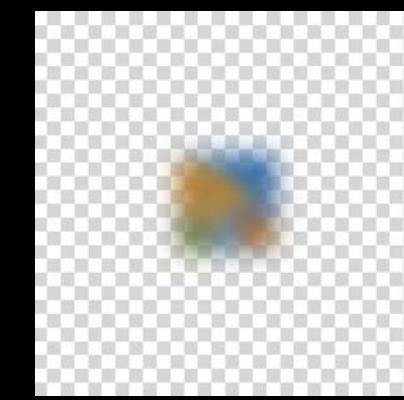
Pass 1



Pass 2



Pass 3



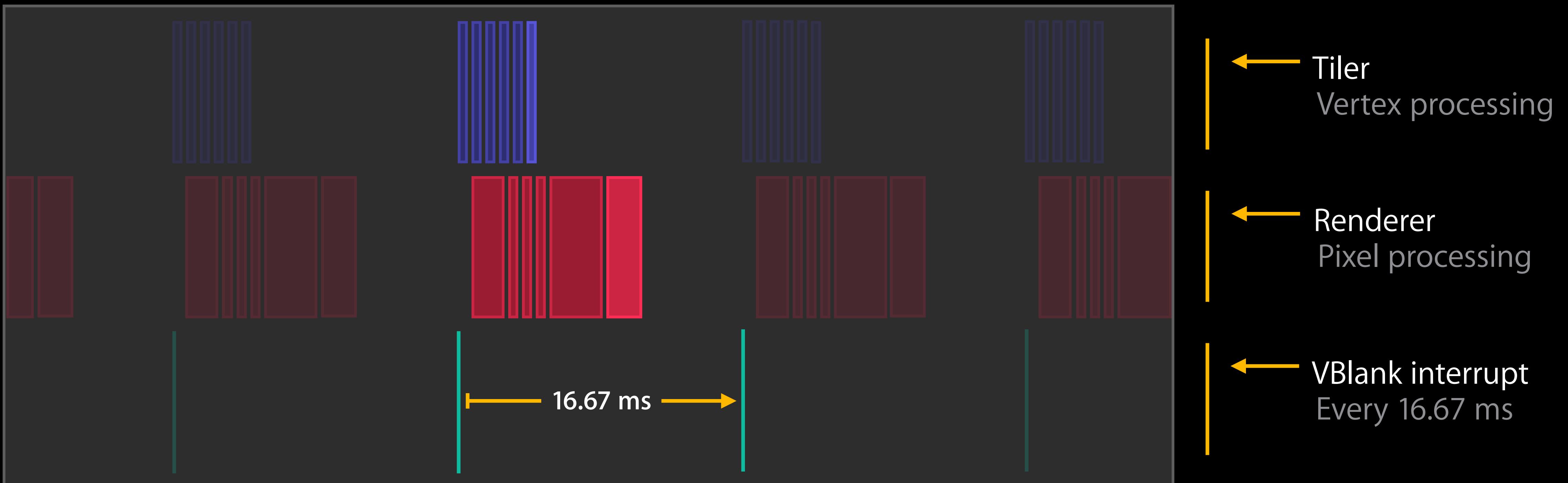
Pass 4



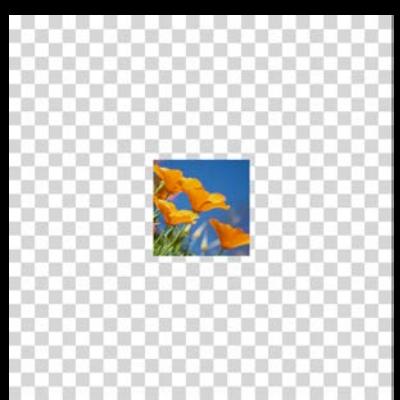
Pass 5

UIVisualEffectView with UIVibrancyEffect

GPU utilization, fullscreen, iPad Air



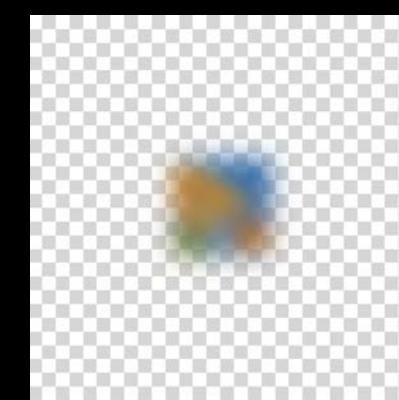
Pass 1



Pass 2



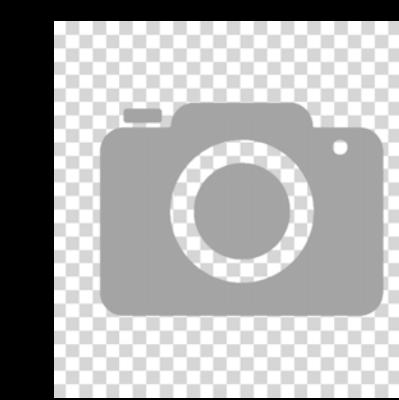
Pass 3



Pass 4



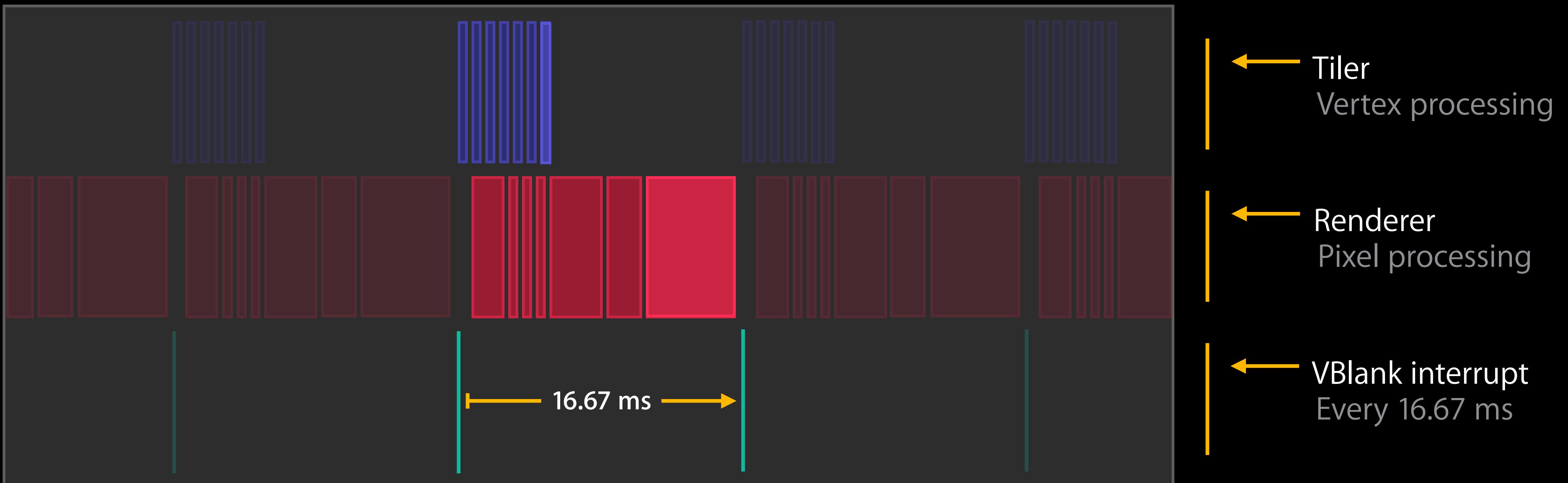
Pass 5



Pass 6

UIVisualEffectView with UIVibrancyEffect

GPU utilization, fullscreen, iPad Air



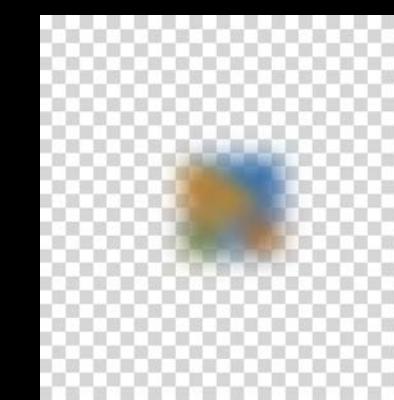
Pass 1



Pass 2



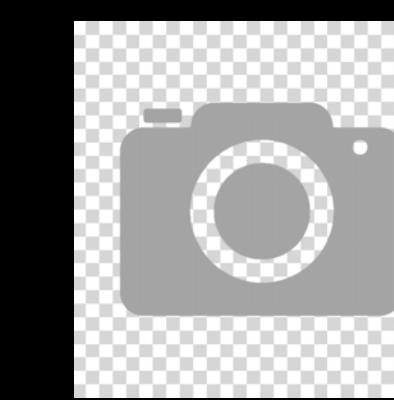
Pass 3



Pass 4



Pass 5



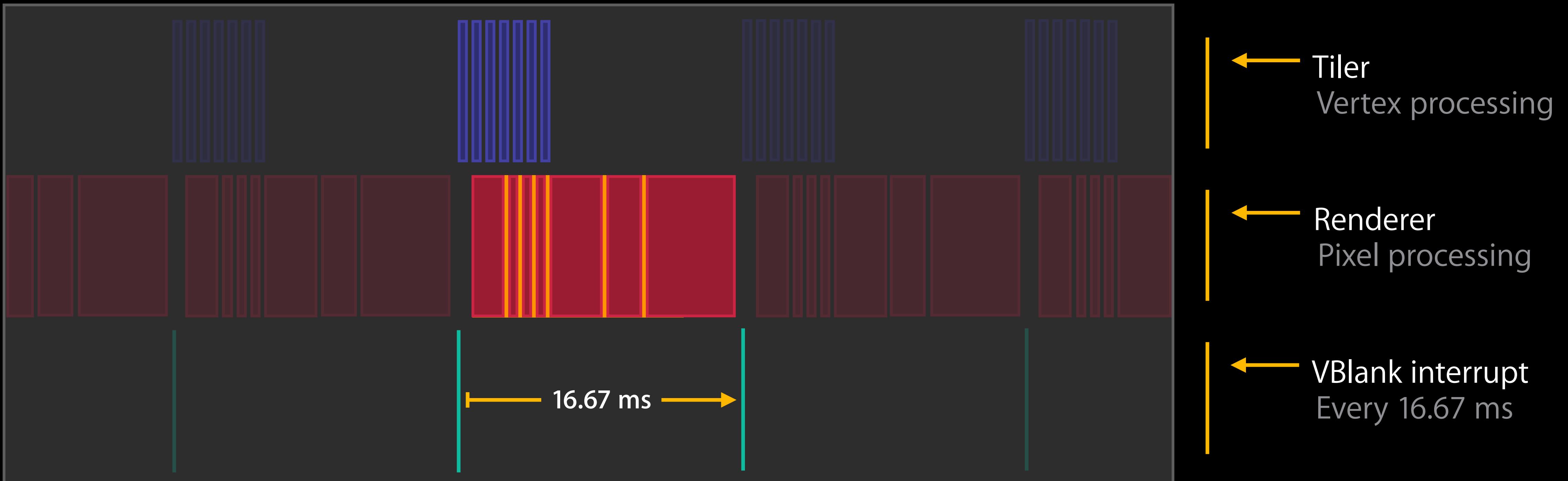
Pass 6



Pass 7

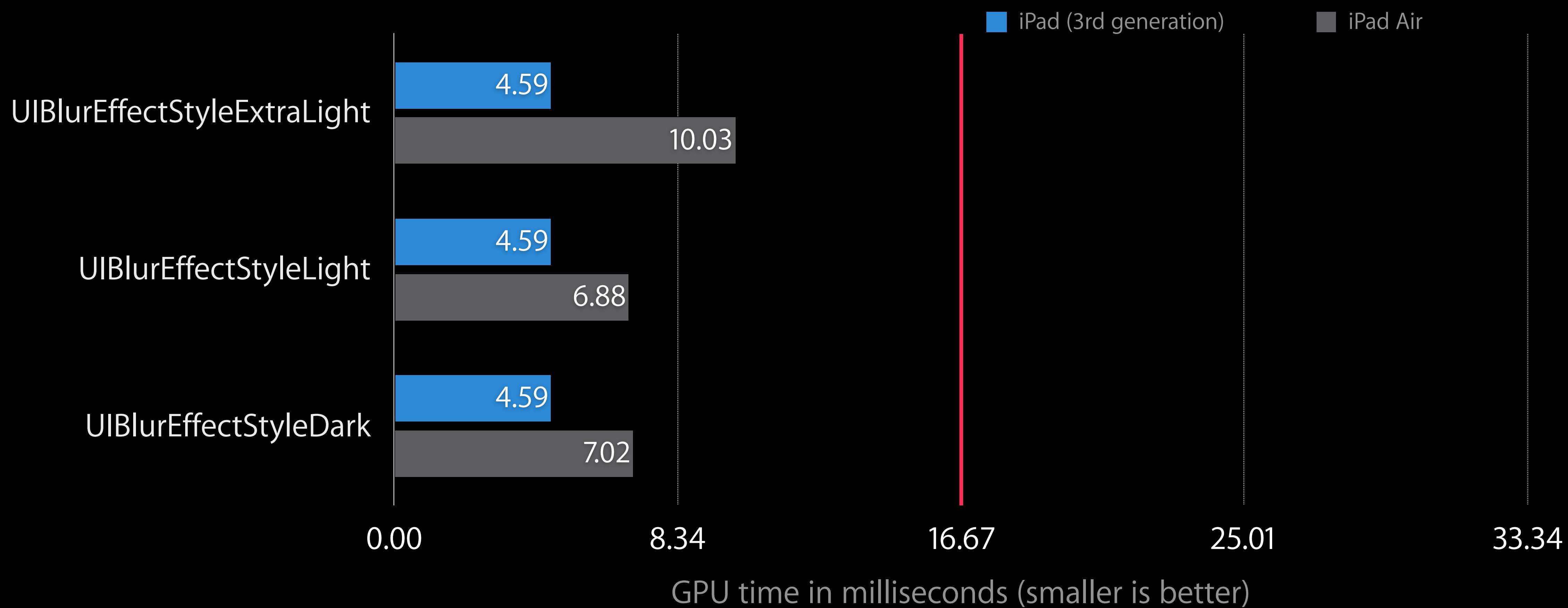
UIVisualEffectView with UIVibrancyEffect

GPU utilization, fullscreen, iPad Air



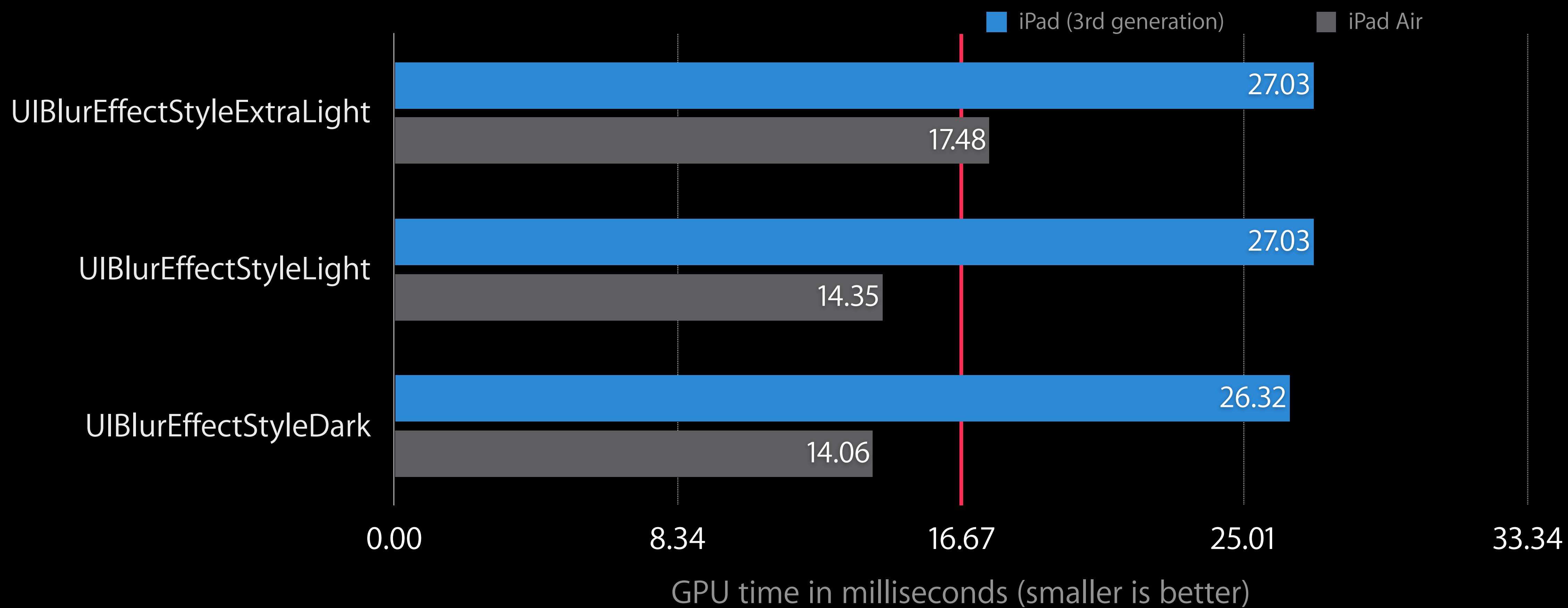
UIVisualEffectView with UIVibrancyEffect

FULLSCREEN performance



UIVisualEffectView with UIVibrancyEffect

FULLSCREEN performance



UIVisualEffectView with UIVibrancyEffect

Performance considerations

UIVibrancyEffect adds two offscreen passes

UIVibrancyEffect uses expensive compositing filter for content

Use UIVibrancyEffect on small regions

Only dirty regions are redrawn

UIVibrancyEffect is very costly on all devices

- UI can easily be GPU bound
- Keep bounds of view as small as possible
- Make sure to budget for effects

Rasterization

Performance considerations

Use to composite to image once with GPU

Enable with `shouldRasterize` property on `CALayer`

Extra offscreen passes when updating content

Do not overuse, cache size is limited to 2.5x of screen size

Rasterized images evicted from cache if unused for more than 100ms

Rasterization

Typical use cases

Avoid redrawing expensive effects for static content

Avoid redrawing of complex view hierarchies

Group Opacity

Performance considerations

Disable with `allowsGroupOpacity` property on `CALayer`

Will introduce offscreen passes:

- If layer is not opaque (`opacity != 1.0`)
- And if layer has nontrivial content (child layers or background image)
 - Sub view hierarchy needs to be composited before being blended

Always turn it off if not needed

Tools

Michael Ingrassia
iOS Software Engineer

Performance Investigation Mindset

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Too many offscreen passes?

Fewer is better

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Too many offscreen passes?

Fewer is better

Too much blending?

Less is better

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Too many offscreen passes?

Fewer is better

Too much blending?

Less is better

Any strange image formats or sizes? Avoid on-the-fly conversions or resizing

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Too many offscreen passes?

Fewer is better

Too much blending?

Less is better

Any strange image formats or sizes? Avoid on-the-fly conversions or resizing

Any expensive views or effects?

Understand the cost of what is in use

Performance Investigation Mindset

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Too many offscreen passes?

Fewer is better

Too much blending?

Less is better

Any strange image formats or sizes? Avoid on-the-fly conversions or resizing

Any expensive views or effects?

Understand the cost of what is in use

Anything unexpected in hierarchy?

Know the actual view hierarchy

Tools

Instruments

- Core Animation instrument
- OpenGL ES Driver instrument

Simulator

- Color debug options

Xcode

- View debugging

Instruments

Core Animation template



Choose a profiling template for: iPhone5s (v8.0) > All Processes

Standard Custom Recent Q Search

Blank	Activity Monitor	Allocations	Automation	Core Animation	Counters
Energy	Leaks	Network	OpenGL ES	OpenGL ES	System Trace
Core Animation This template measures application graphics performance as well as CPU usage of a process via time profiling.					

Open an Existing File... Cancel Choose

Instruments

Core Animation template



Choose a profiling template for: iPhone5s (v8.0) > All Processes

Standard Custom Recent Q Search

Blank Activity Monitor Allocations Automation Core Animation Counters

Energy Leaks Network OpenGL ES OpenGL ES System Trace

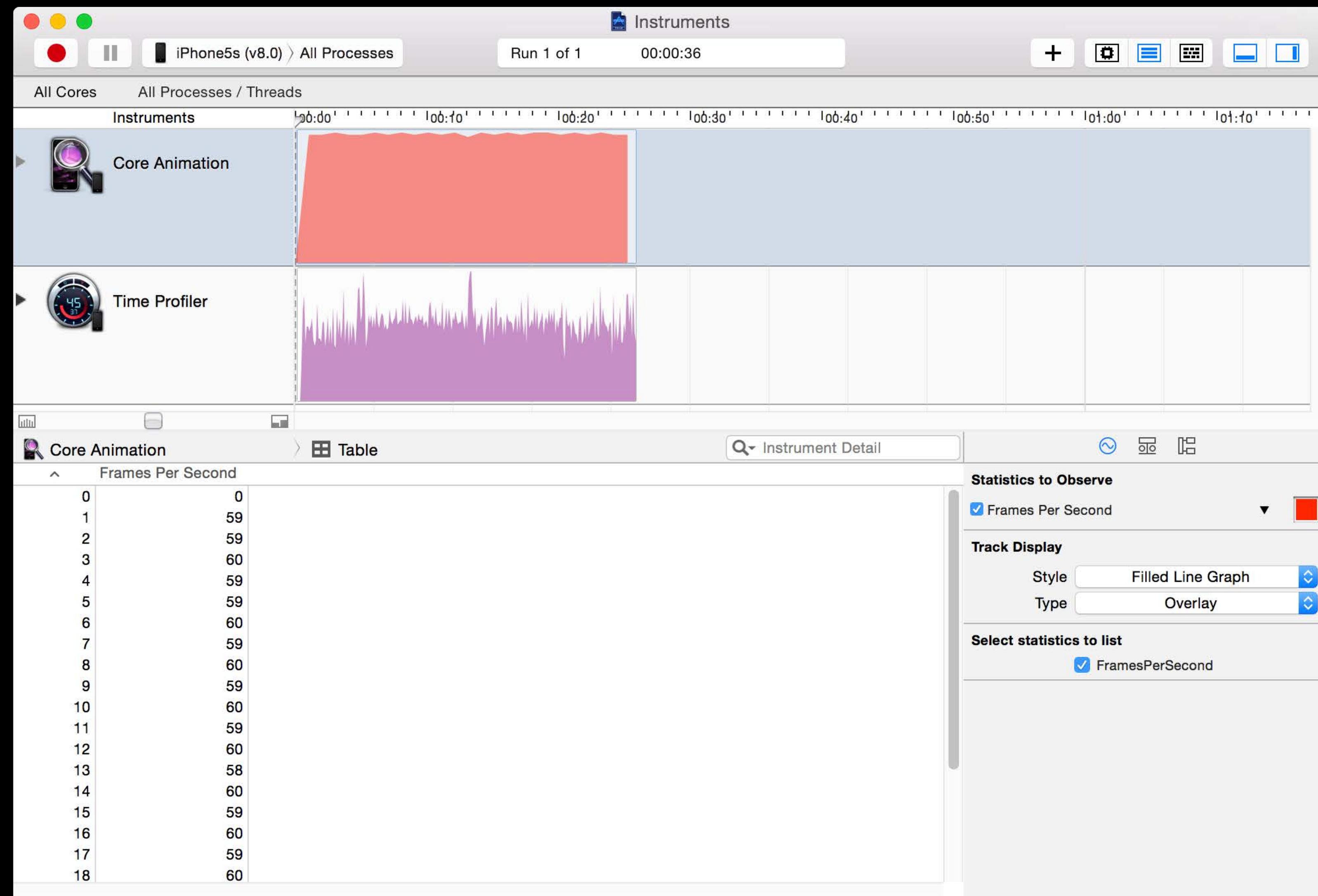
Core Animation
This template measures application graphics performance as well as CPU usage of a process via time profiling.

Open an Existing File... Cancel Choose

A screenshot of the Instruments application's 'Choose a profiling template' dialog. The dialog shows various profiling templates as icons: Standard (selected), Custom, Recent, a search bar, and icons for Blank, Activity Monitor, Allocations, Automation, Core Animation (which is highlighted with a yellow border), Counters, Energy, Leaks, Network, OpenGL ES, OpenGL ES, and System Trace. Below the Core Animation icon, there is a detailed description of the template: 'Core Animation' and 'This template measures application graphics performance as well as CPU usage of a process via time profiling.' At the bottom are buttons for 'Open an Existing File...', 'Cancel', and 'Choose'.

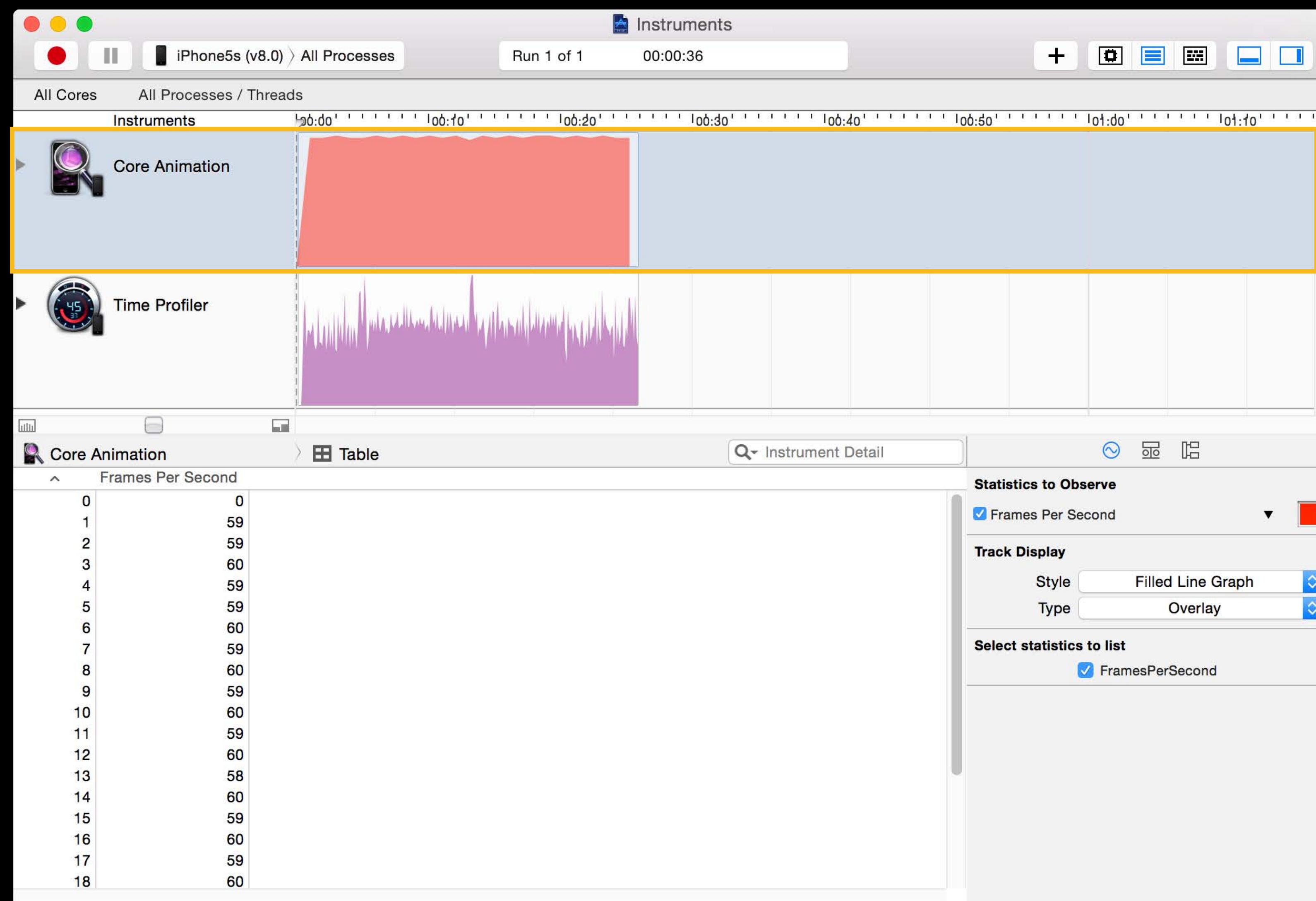
Core Animation Instrument

Measuring frame rate



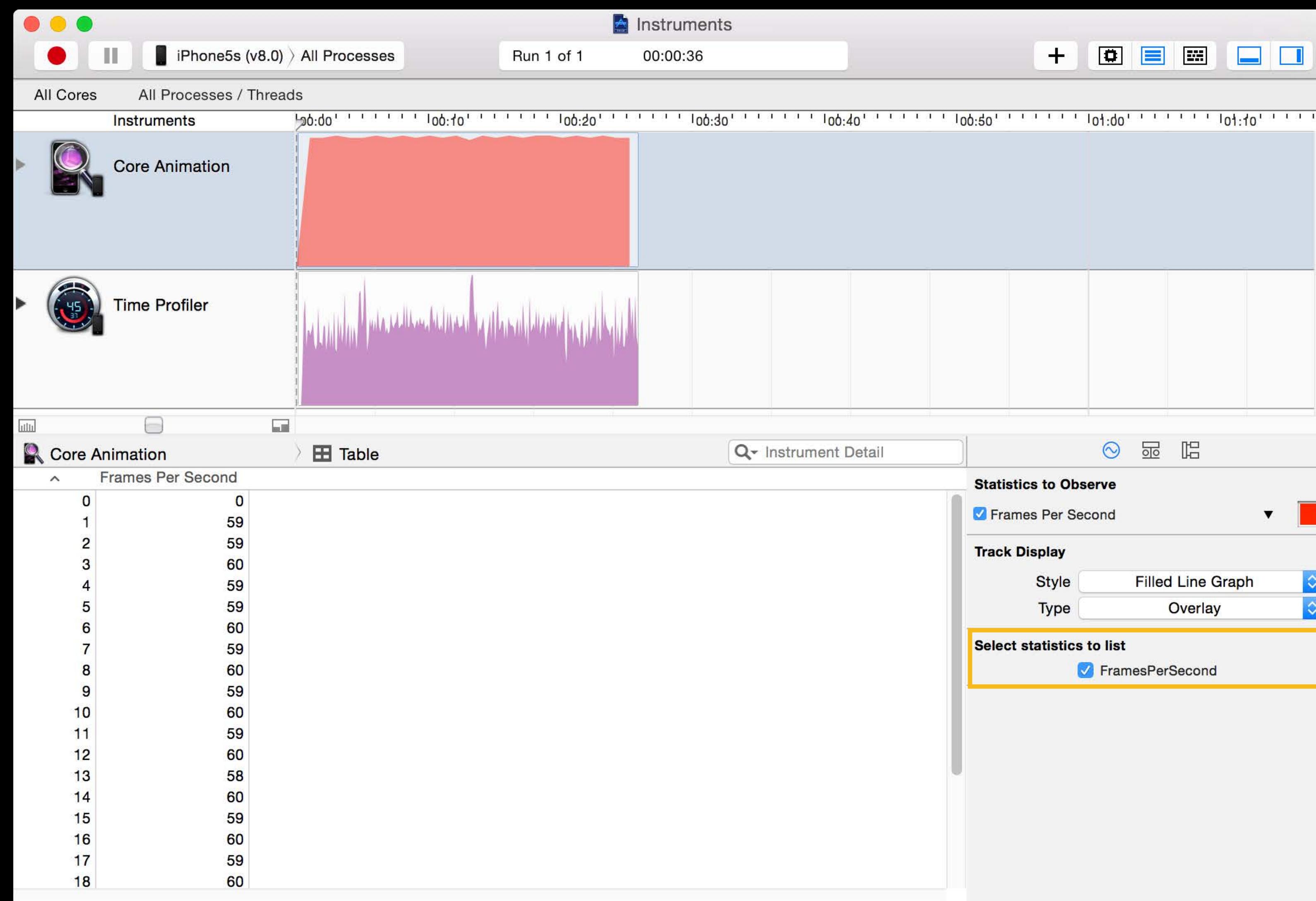
Core Animation Instrument

Measuring frame rate



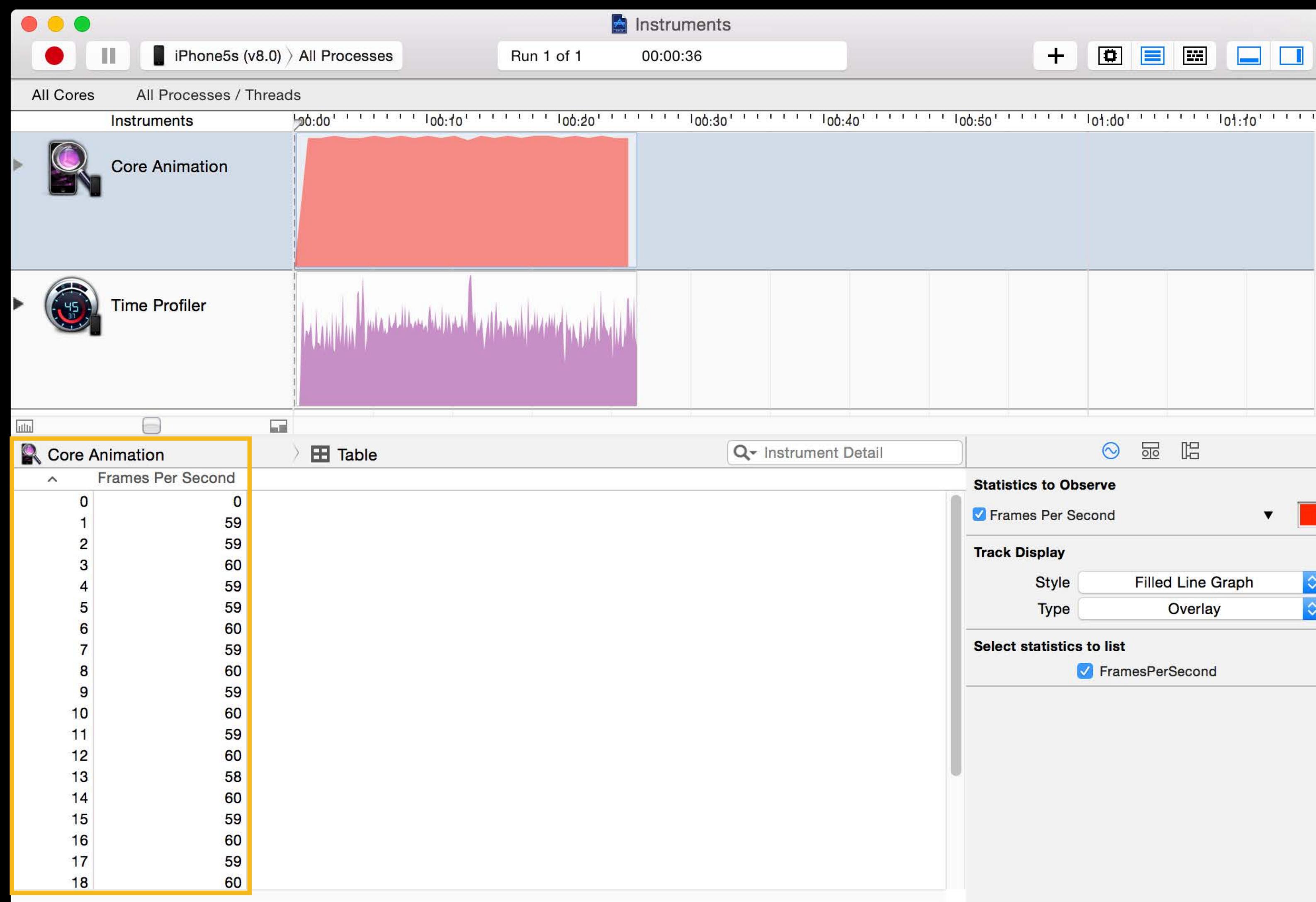
Core Animation Instrument

Measuring frame rate



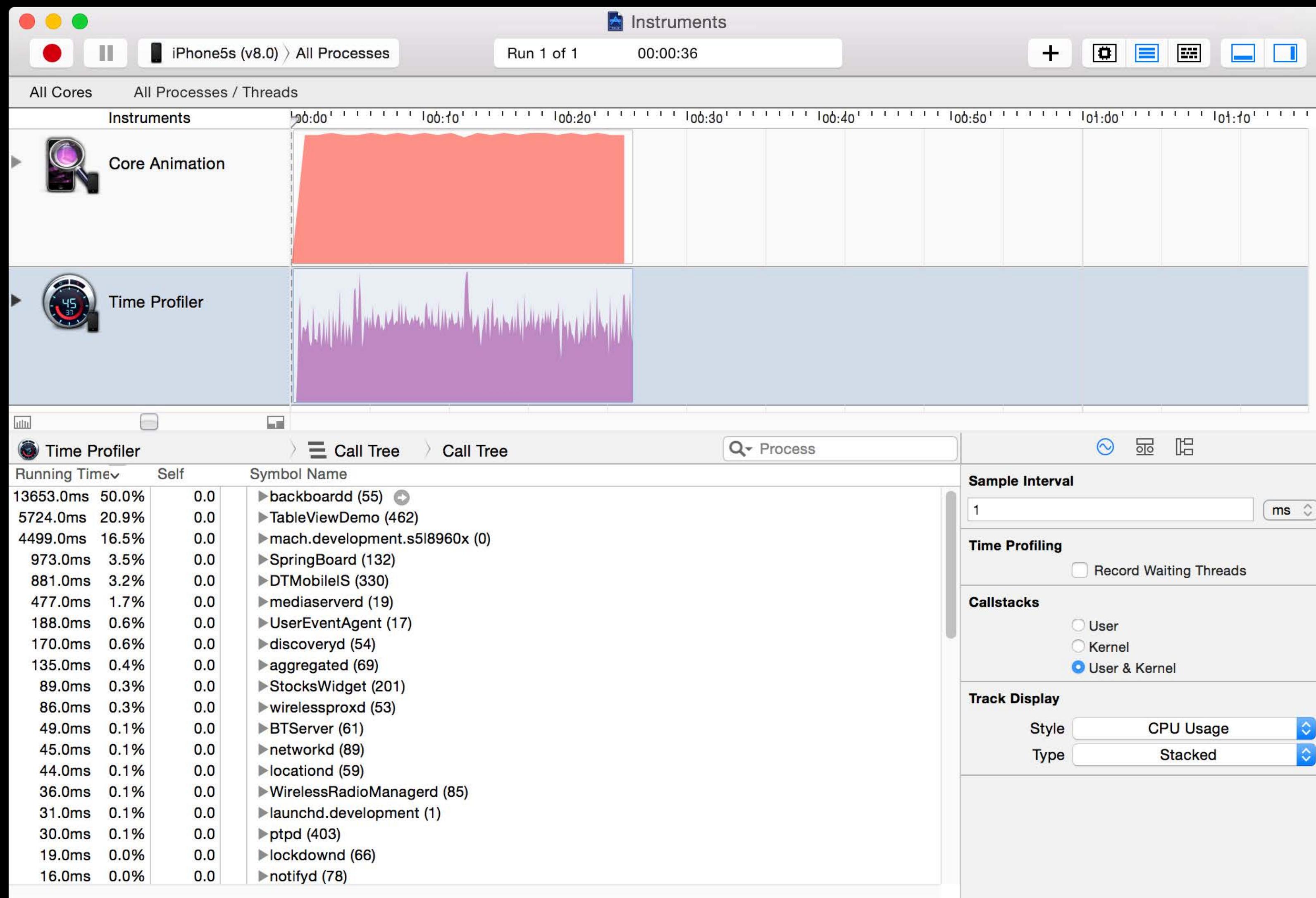
Core Animation Instrument

Measuring frame rate



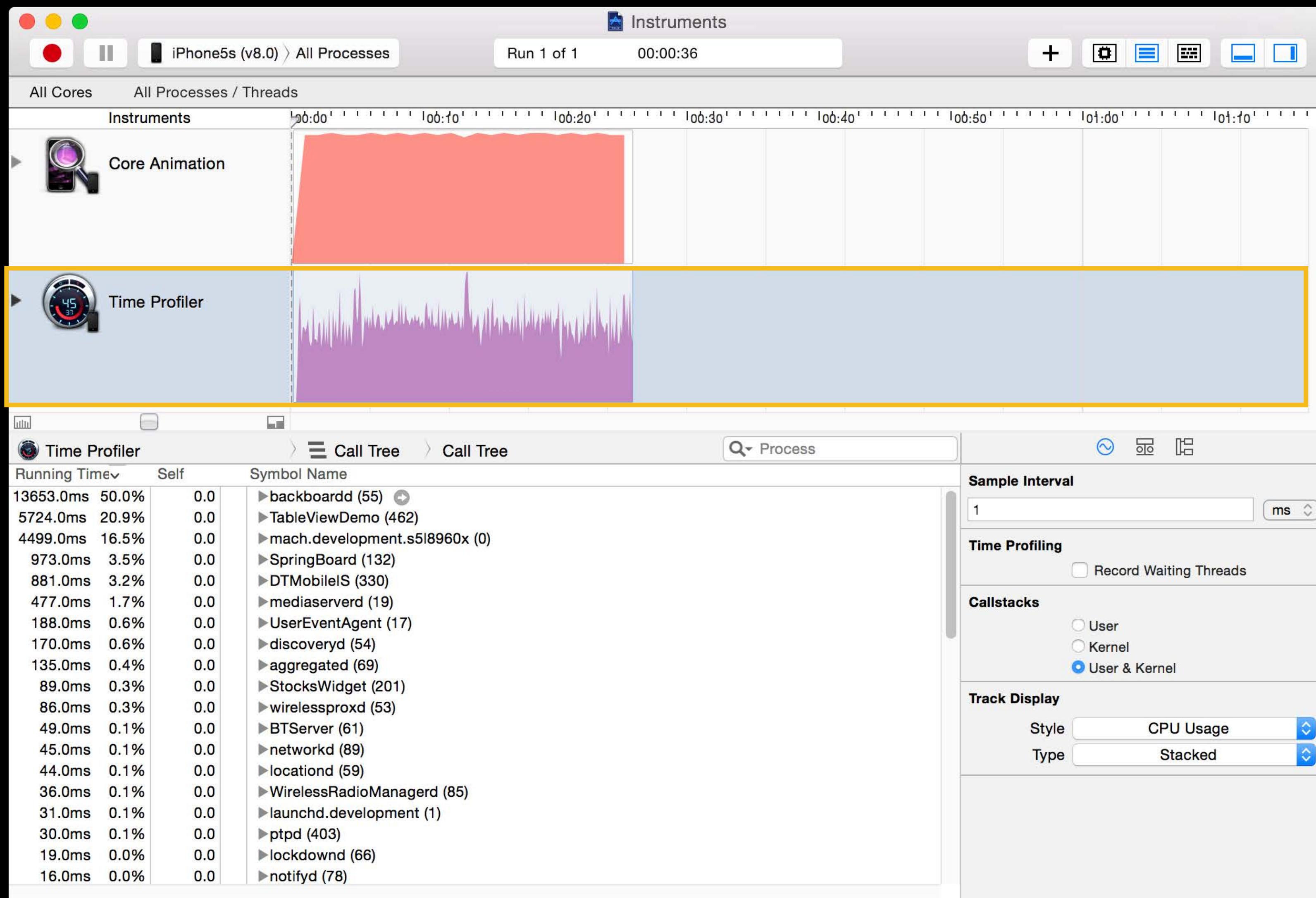
Time Profiler Instrument

CPU utilization



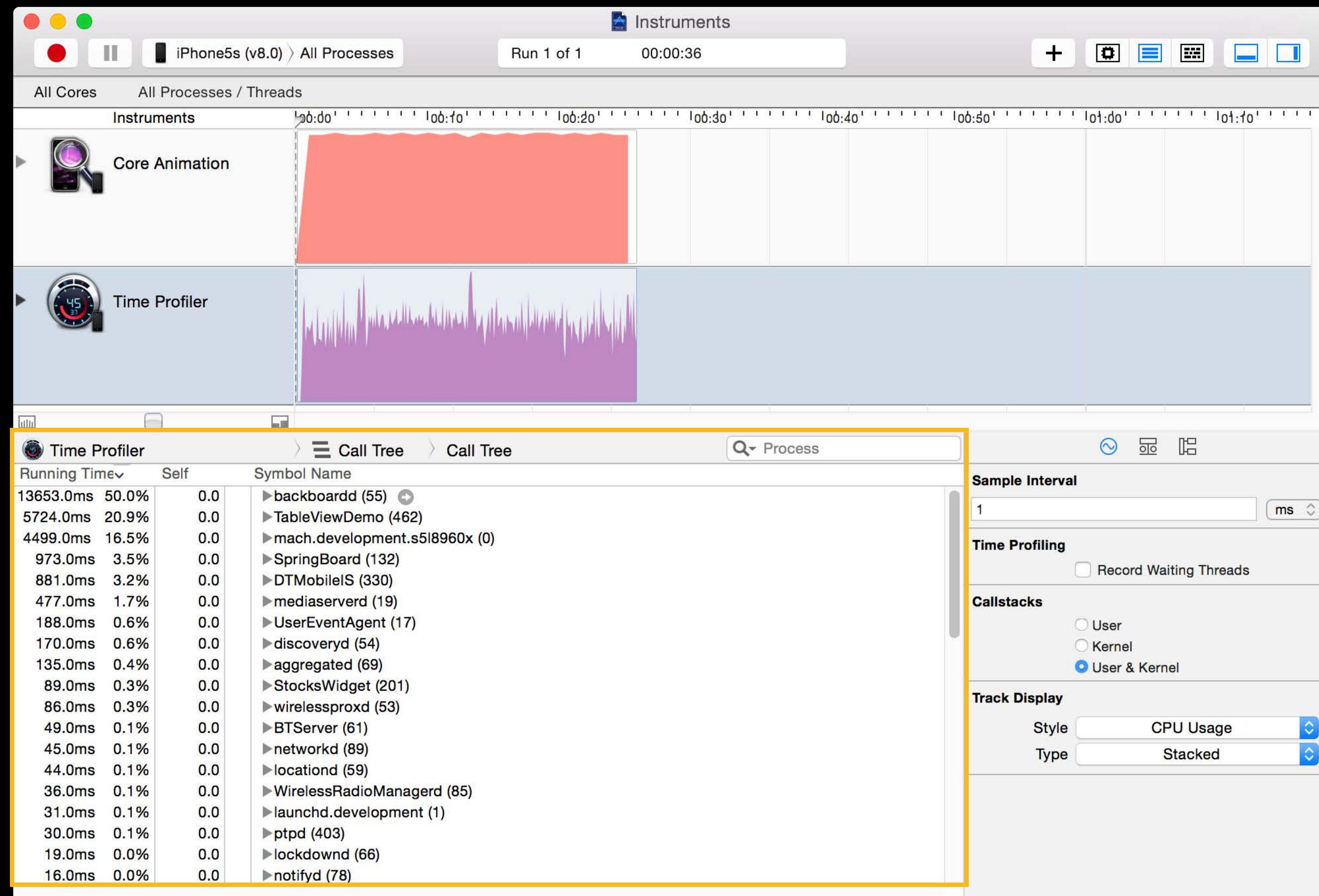
Time Profiler Instrument

CPU utilization



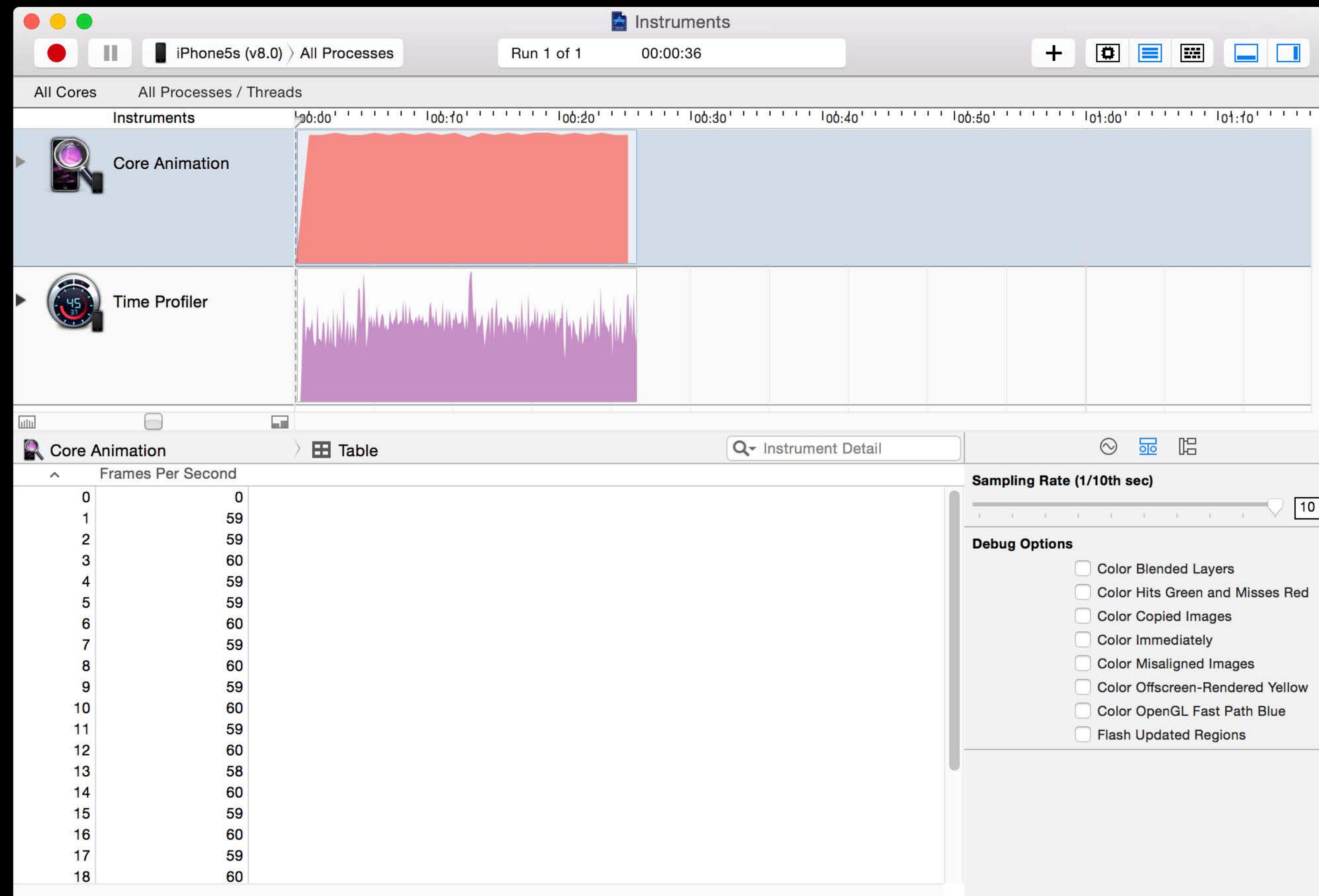
Time Profiler Instrument

CPU utilization



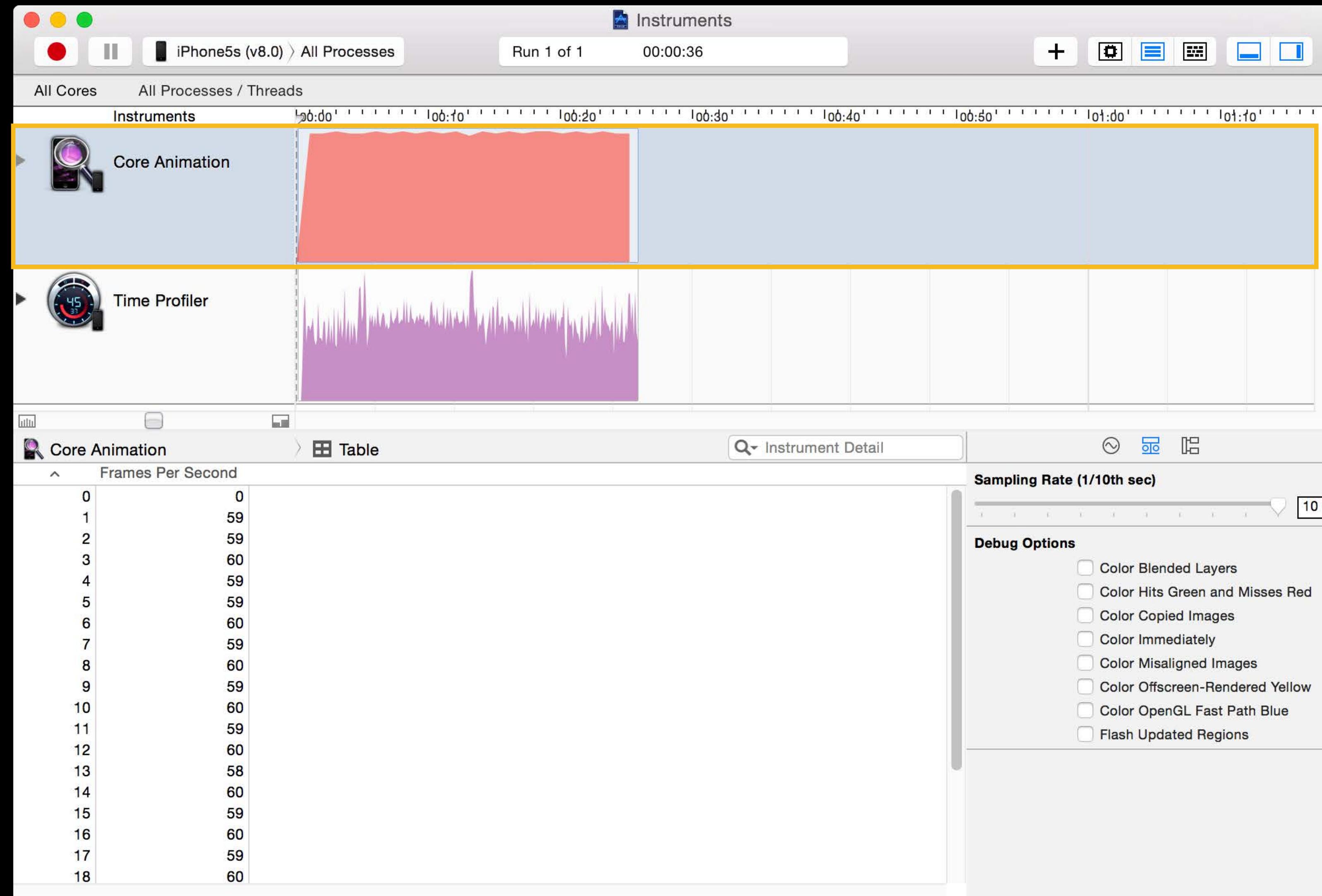
Core Animation Instrument

Color debug options



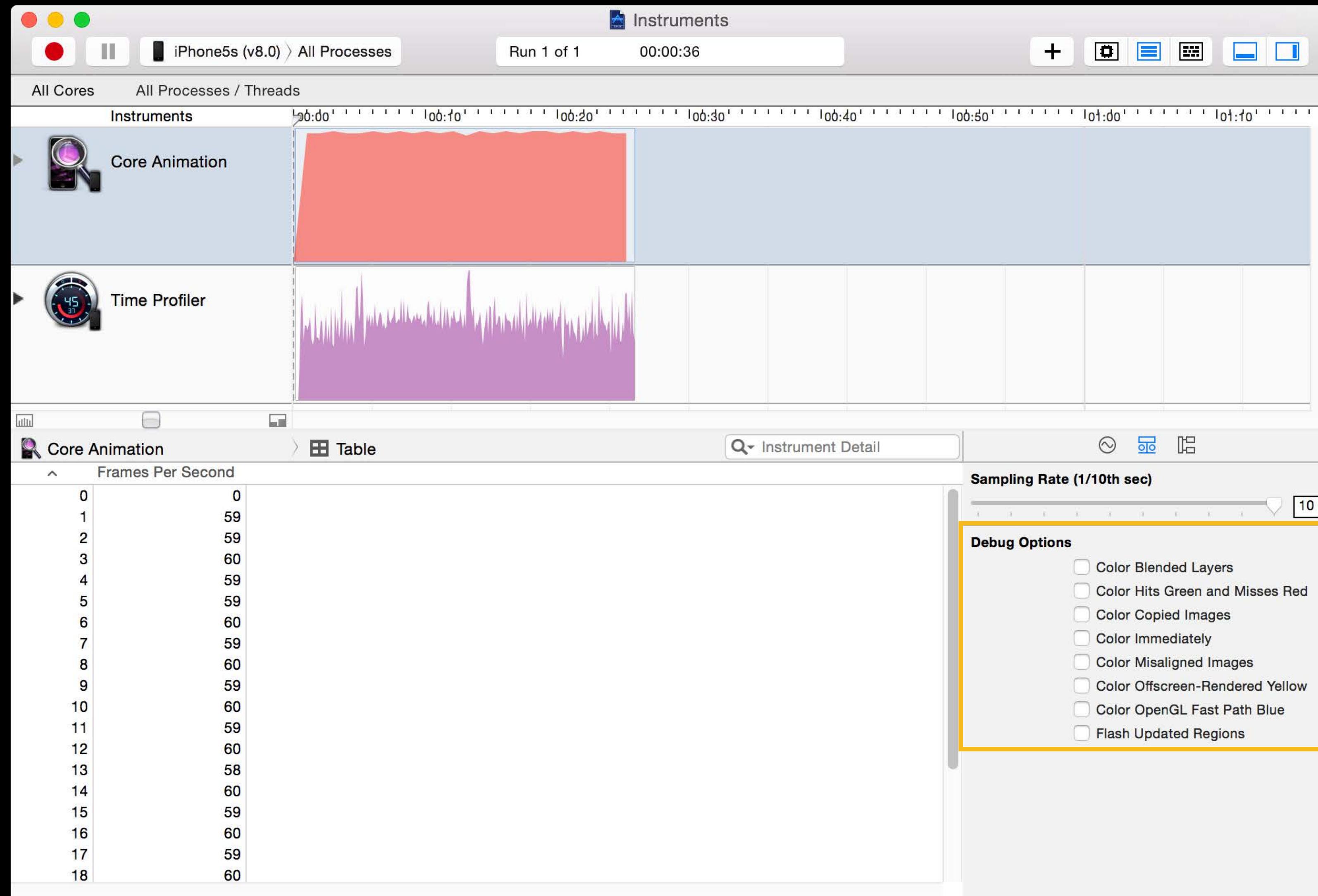
Core Animation Instrument

Color debug options



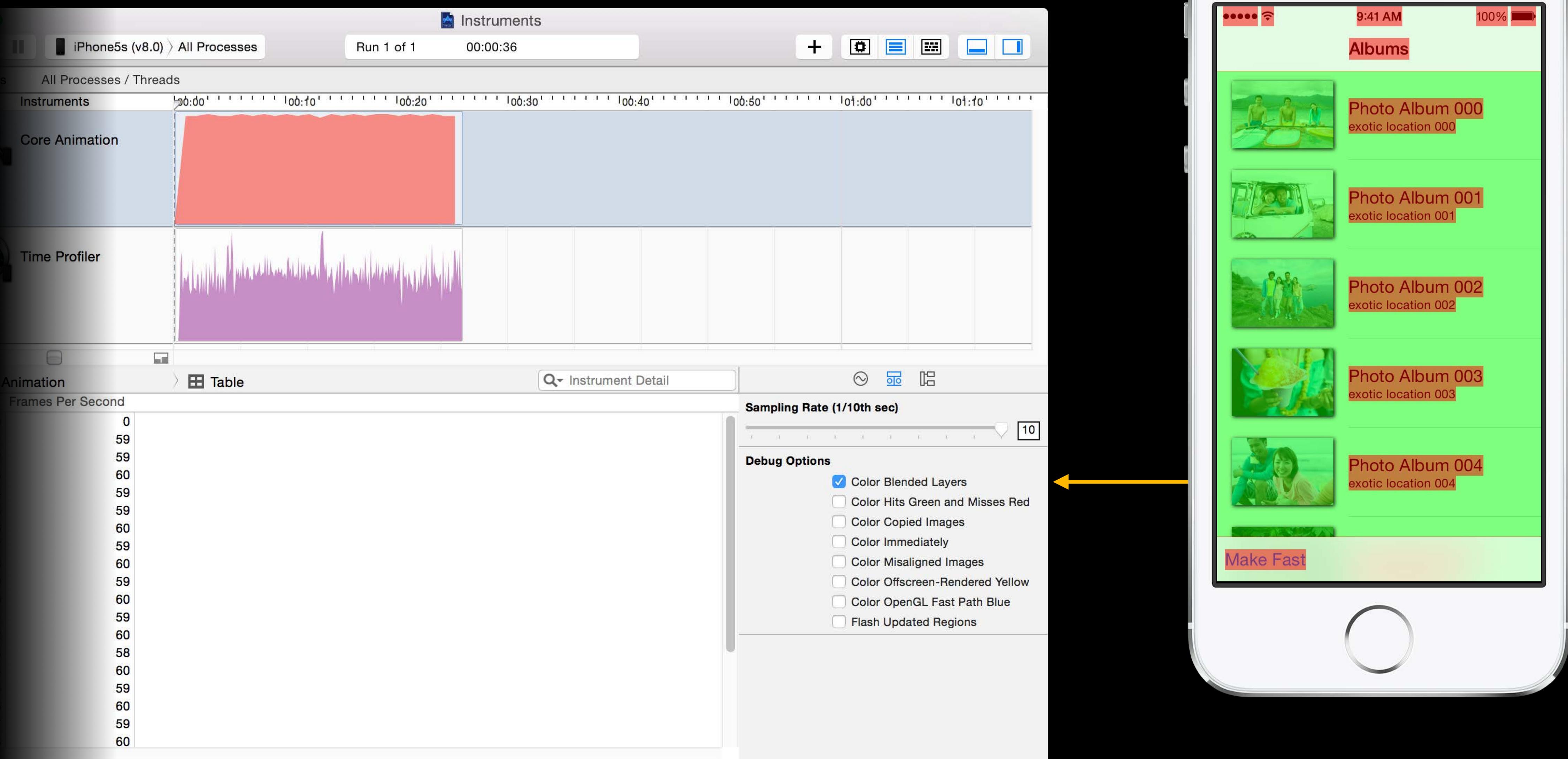
Core Animation Instrument

Color debug options



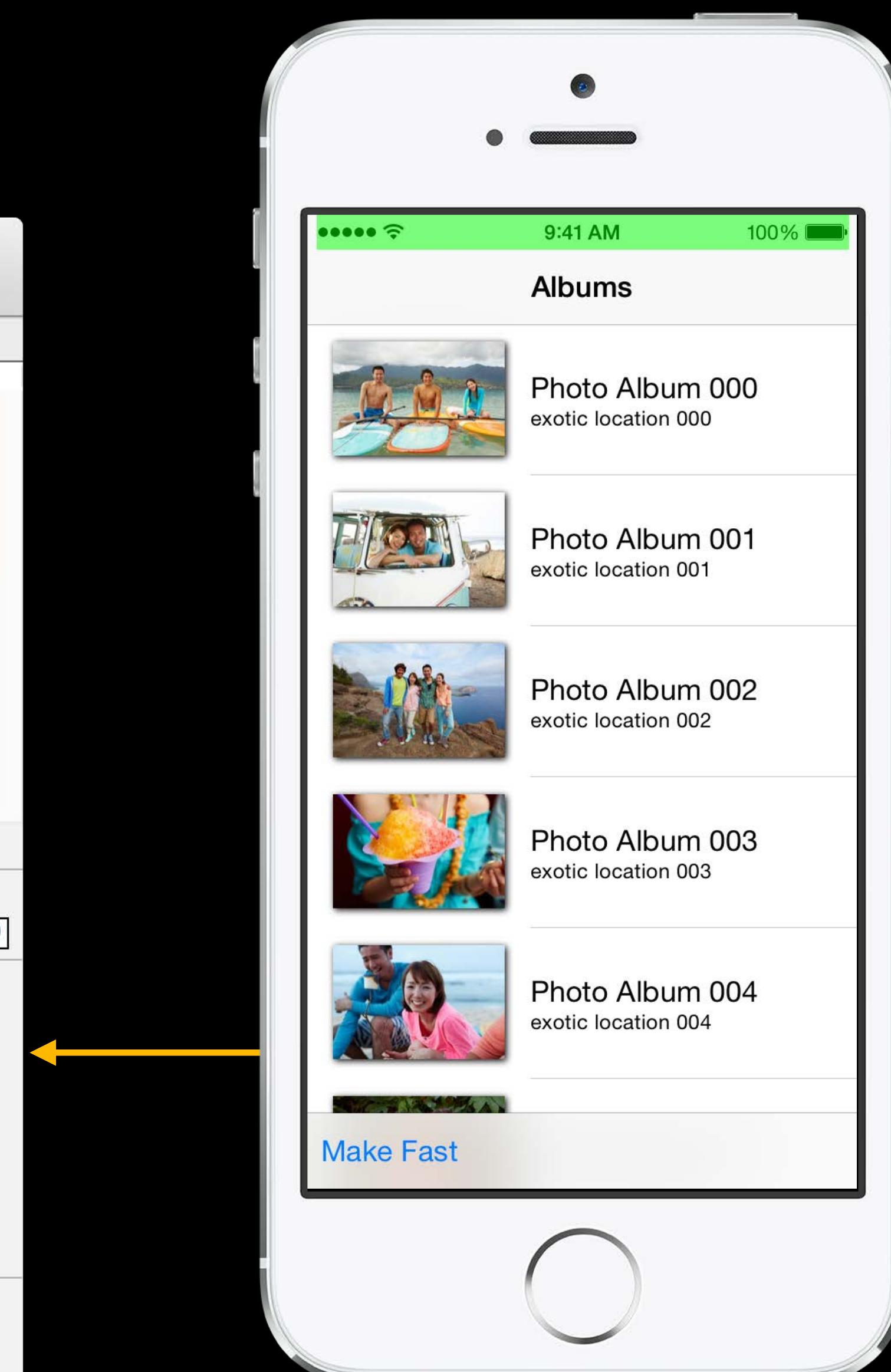
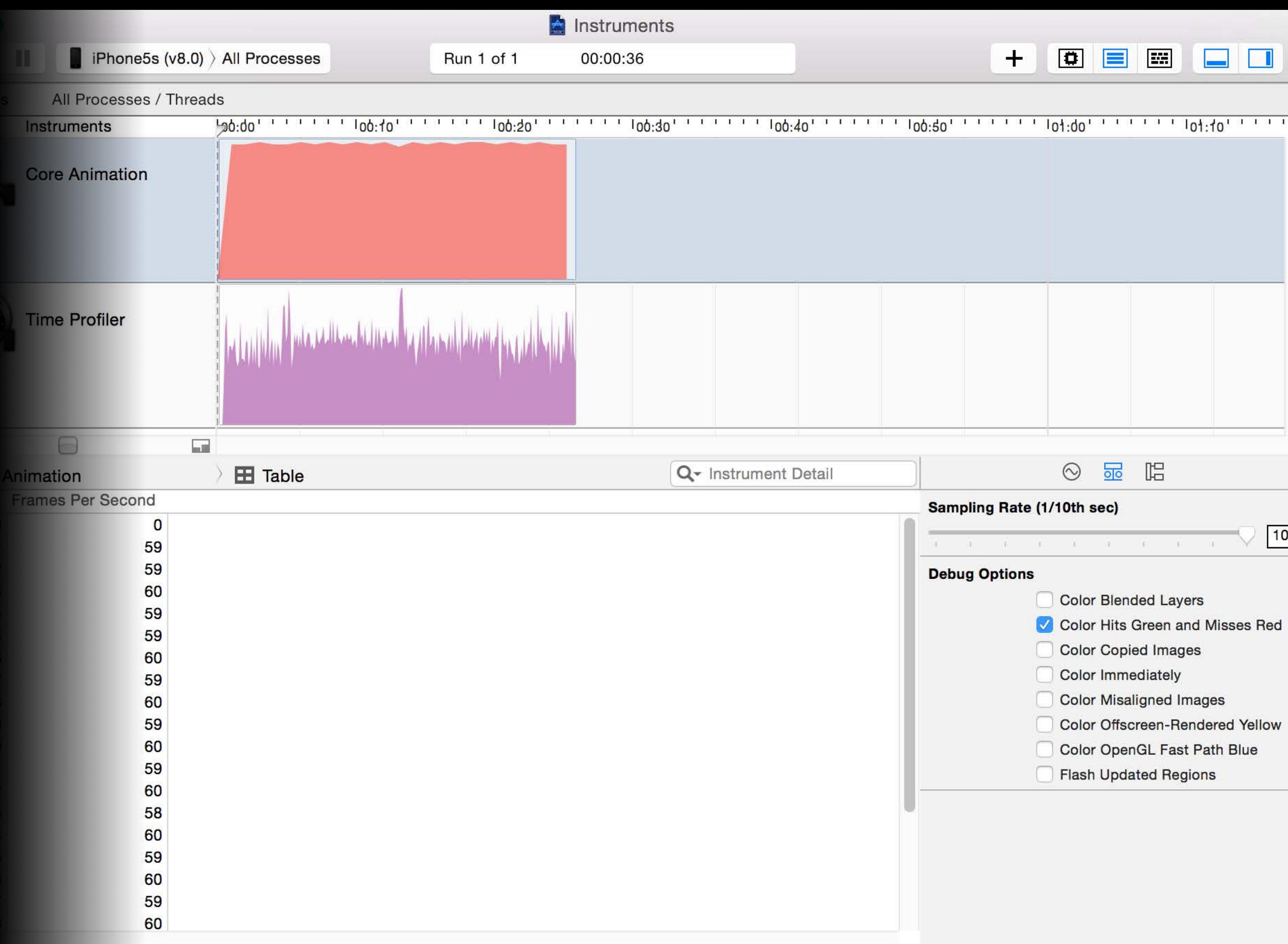
Core Animation Instrument

Color blended layers



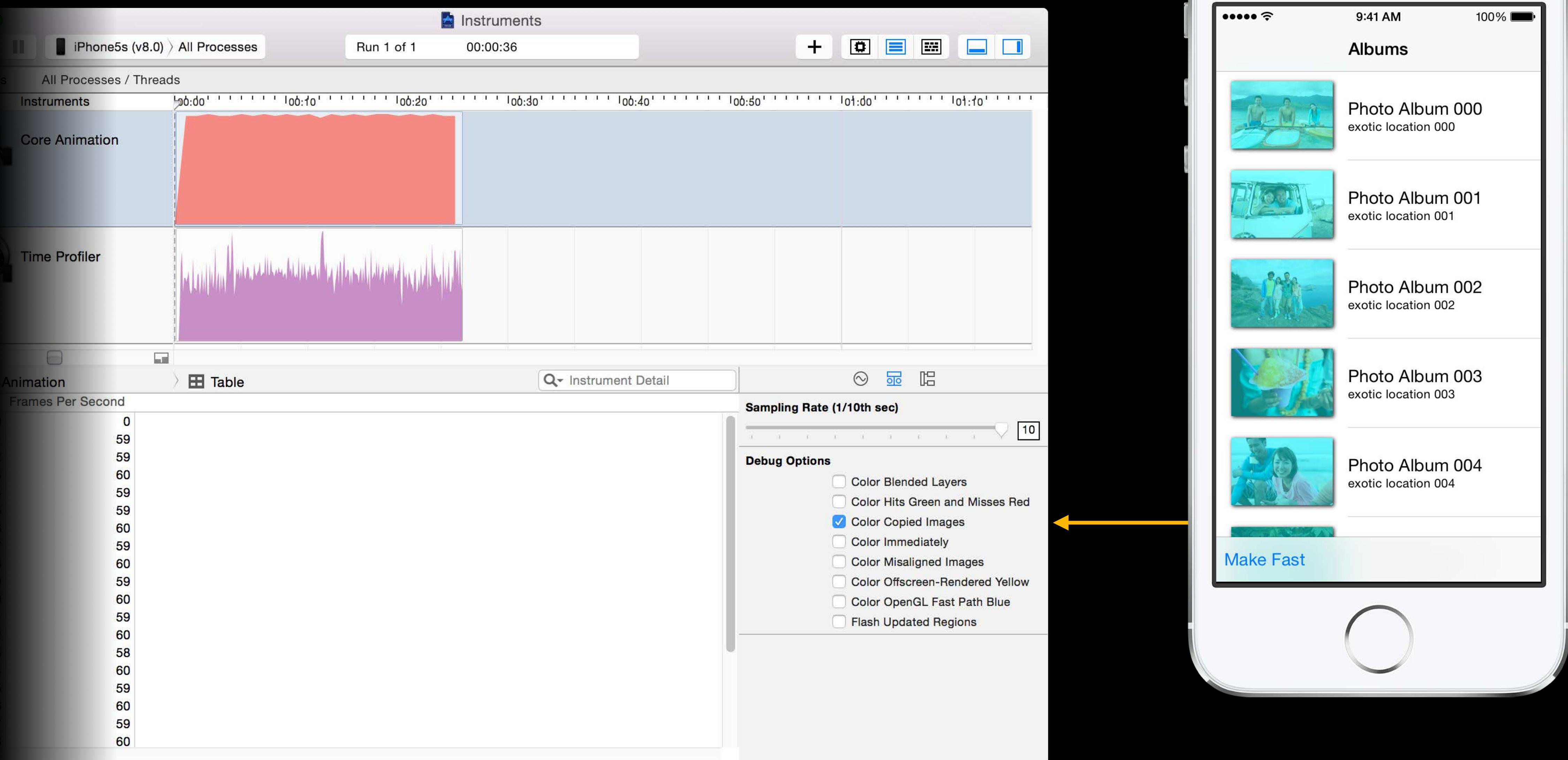
Core Animation Instrument

Color hits green and misses red



Core Animation Instrument

Color copied images



Core Animation Instrument

Color misaligned images

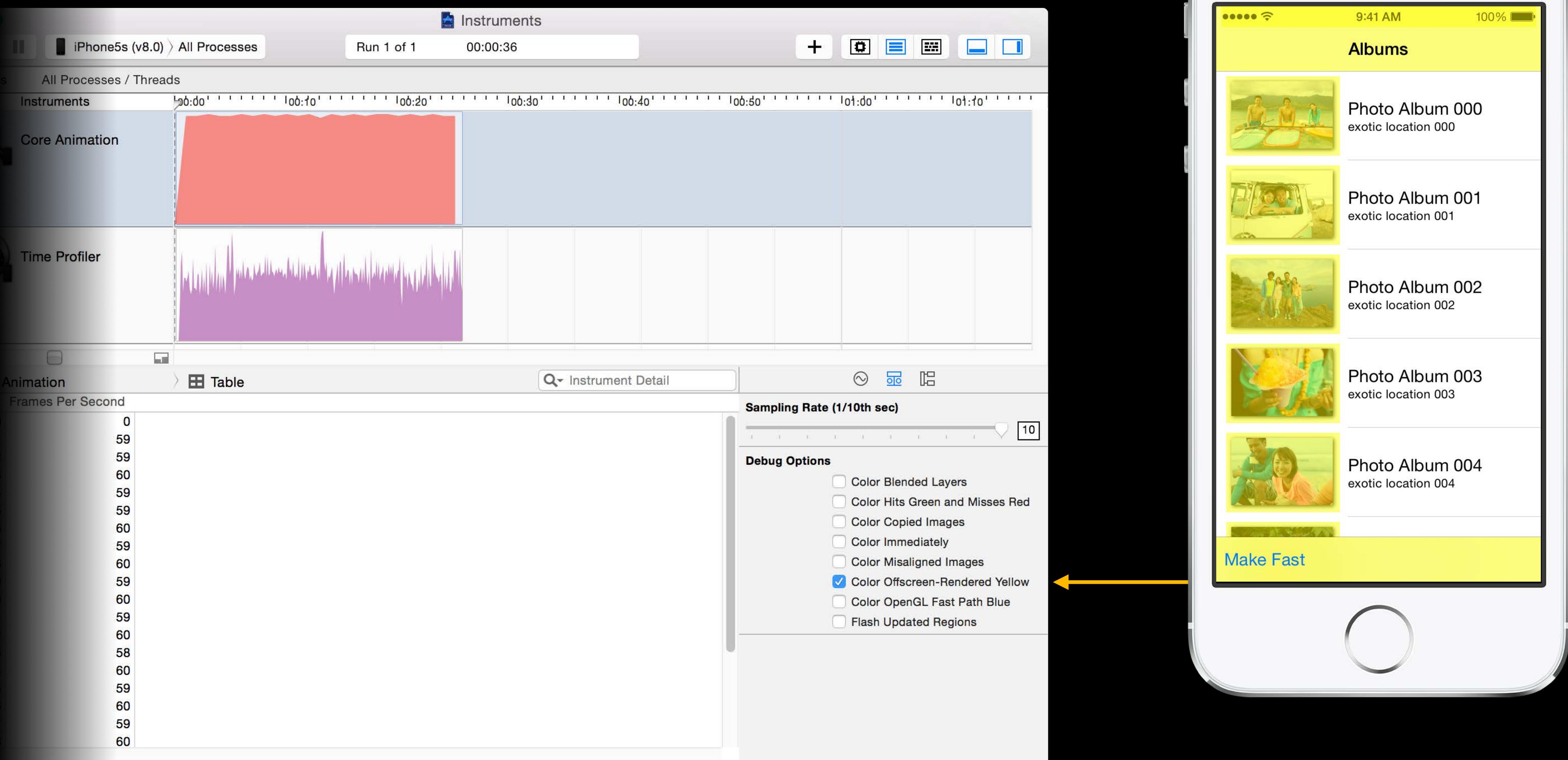
The image shows the Instruments application running on a Mac, connected to an iPhone 5s (v8.0). The Instruments interface displays two main tracks: 'Core Animation' and 'Time Profiler'. The 'Core Animation' track shows a large red rectangular area, indicating a performance issue. The 'Time Profiler' track shows a purple waveform with high-frequency oscillations, suggesting CPU contention or memory access issues.

The Instruments window also includes a 'Sampling Rate (1/10th sec)' slider set to 10, and a 'Debug Options' section with several checkboxes. One checkbox, 'Color Misaligned Images', is checked and highlighted with a yellow arrow pointing from the right towards the iPhone screen.

On the right side of the image is a screenshot of an iPhone displaying a 'Albums' screen. The screen lists five photo albums: 'Photo Album 000', 'Photo Album 001', 'Photo Album 002', 'Photo Album 003', and 'Photo Album 004'. Each album entry includes a small thumbnail image and a label 'exotic location 000' through '004'. At the bottom of the screen, there is a green button labeled 'Make Fast'.

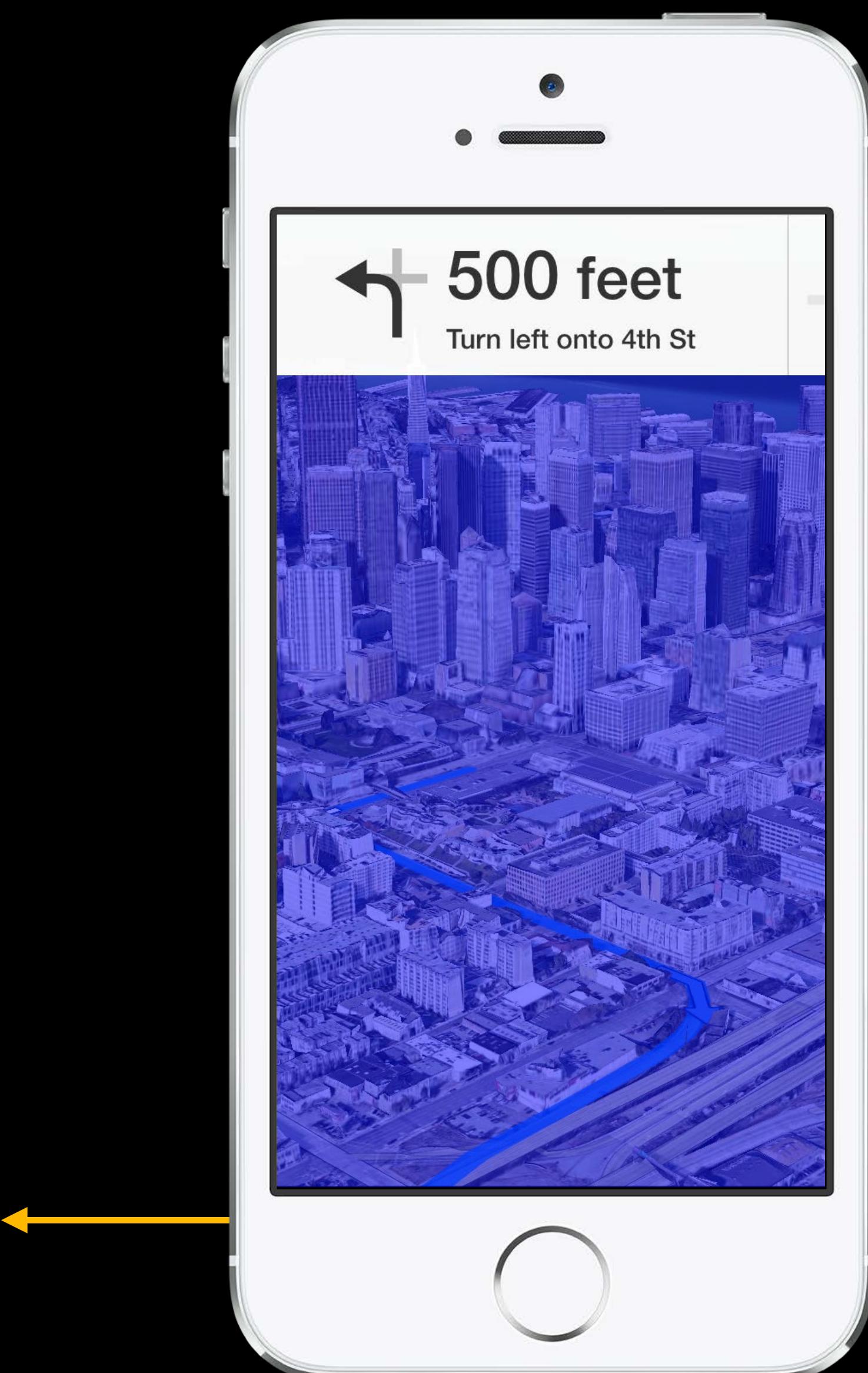
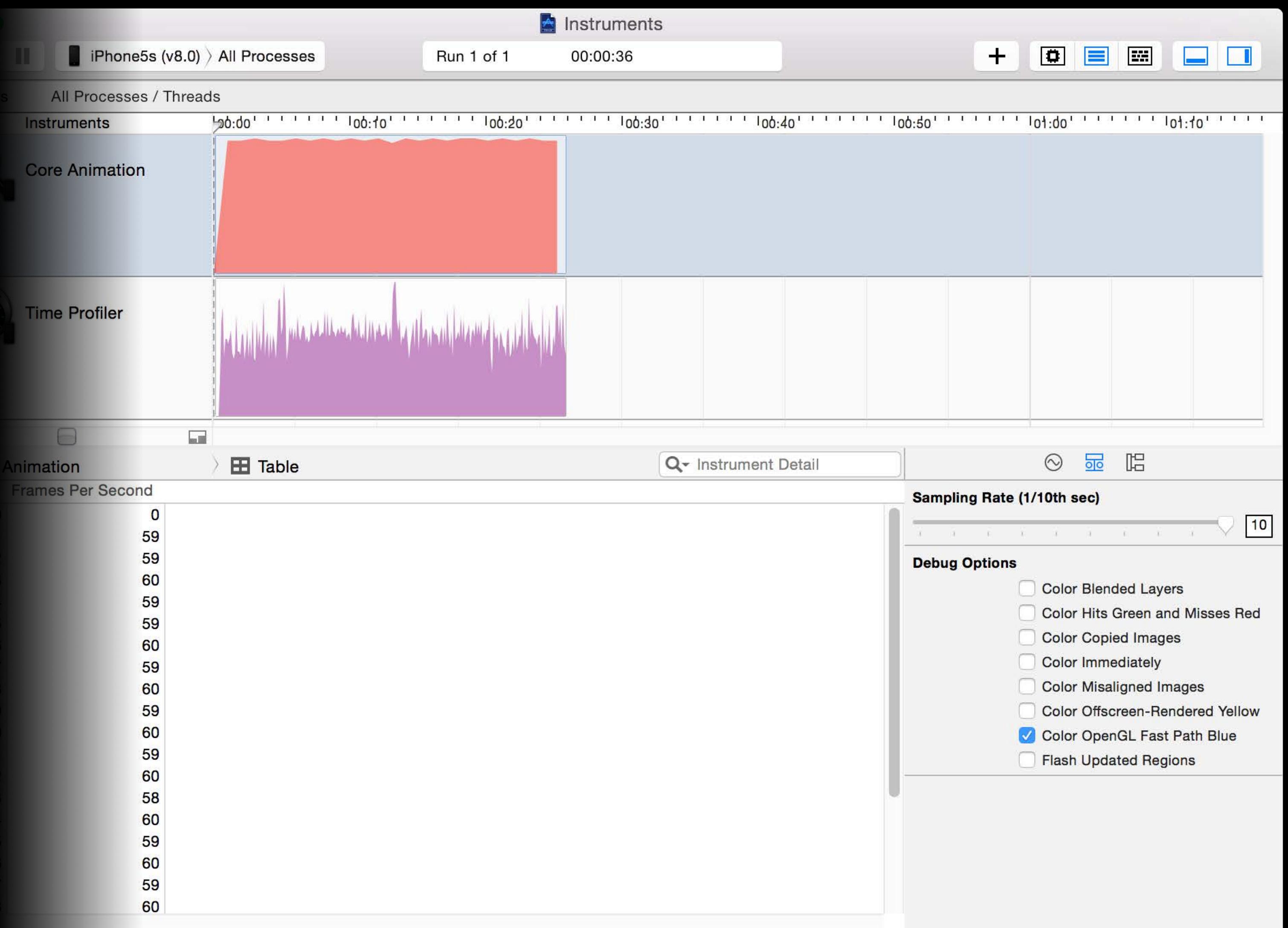
Core Animation Instrument

Color offscreen-rendered yellow



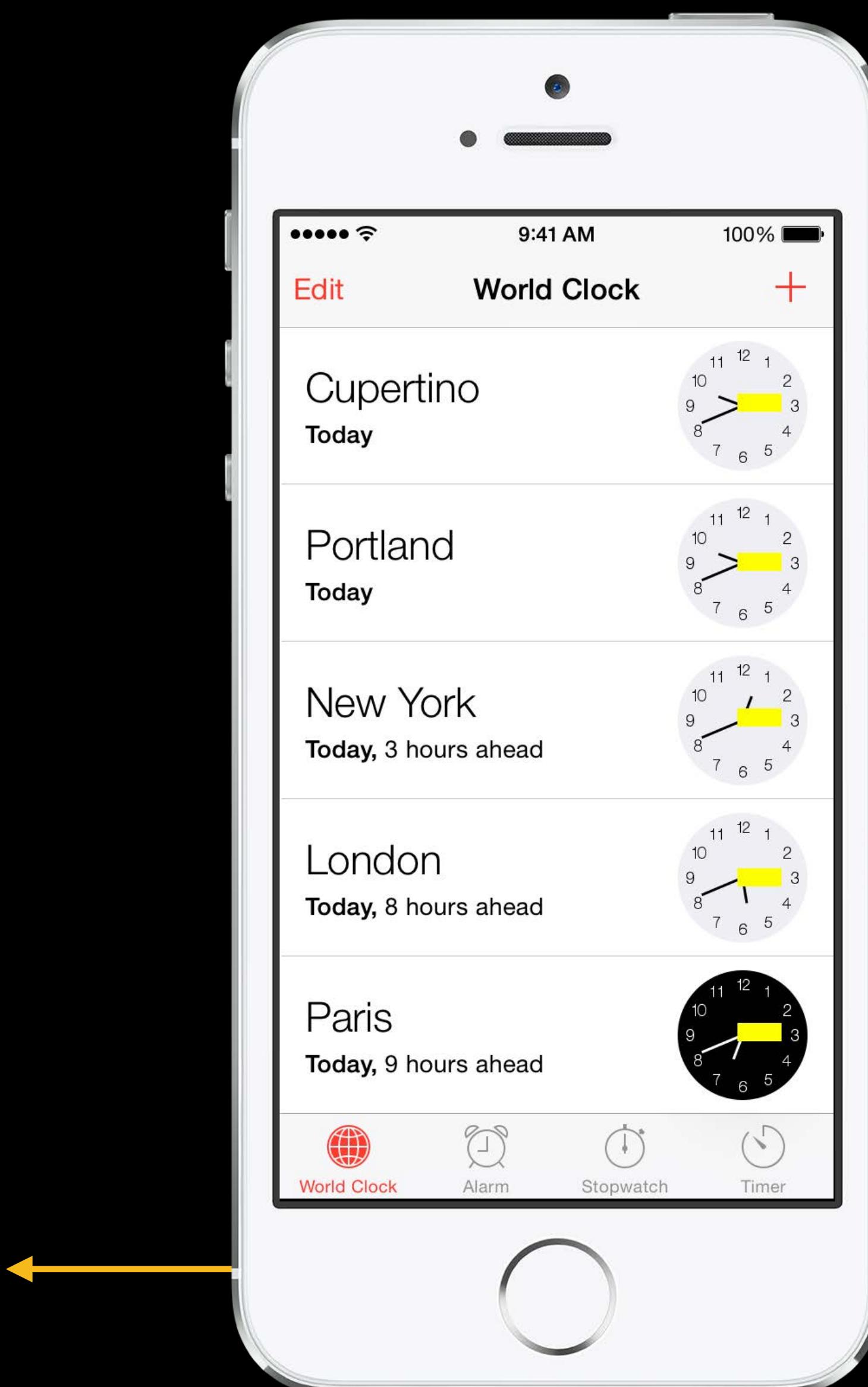
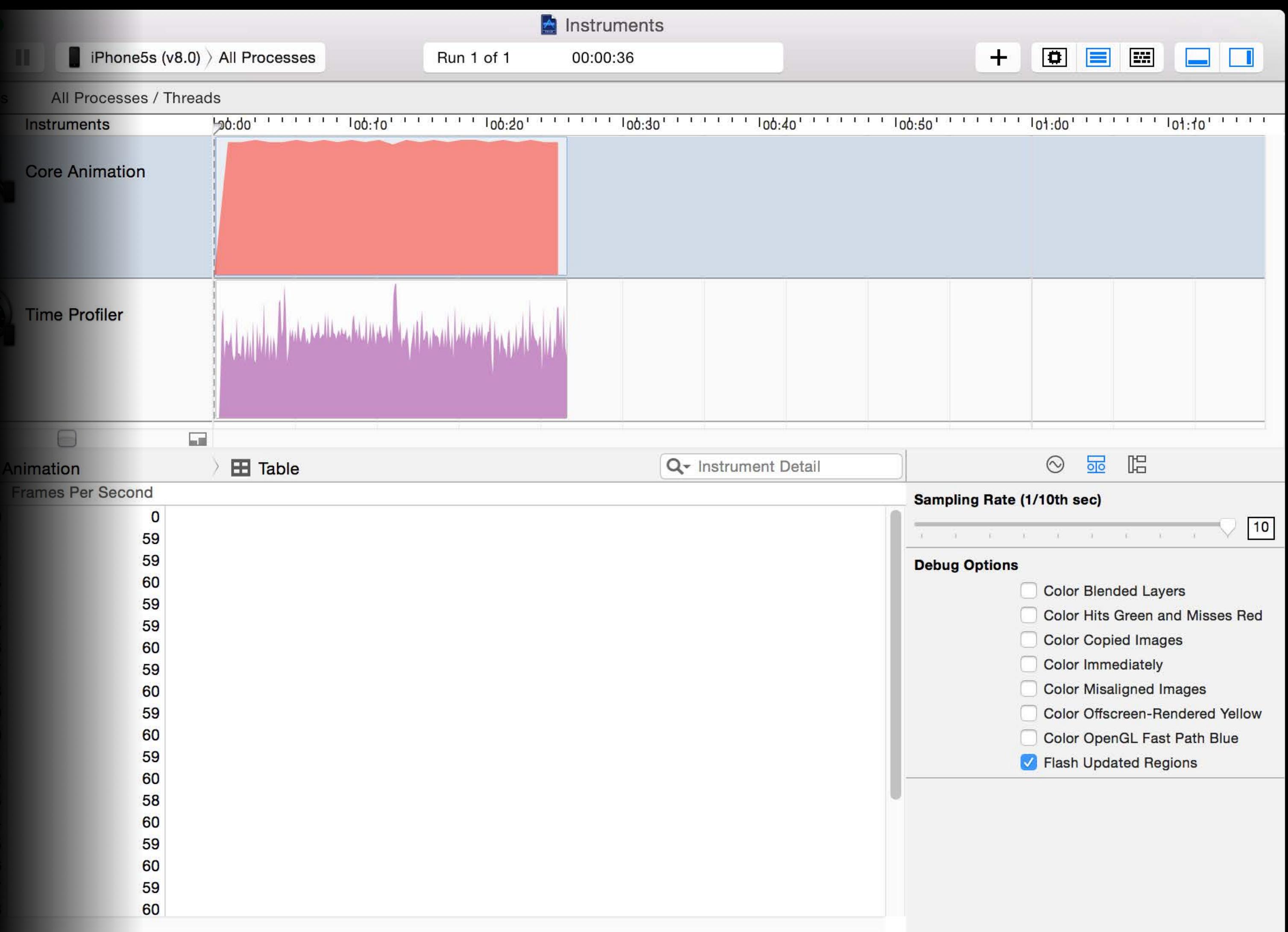
Core Animation Instrument

Color OpenGL fast path blue



Core Animation Instrument

Flash updated regions



Performance Investigation Mindset

Core Animation instrument summary

What is the frame rate?

Goal is always 60 frames per second

Any unnecessary CPU rendering?

GPU is desirable but know when CPU makes sense

Too many offscreen passes?

Fewer is better

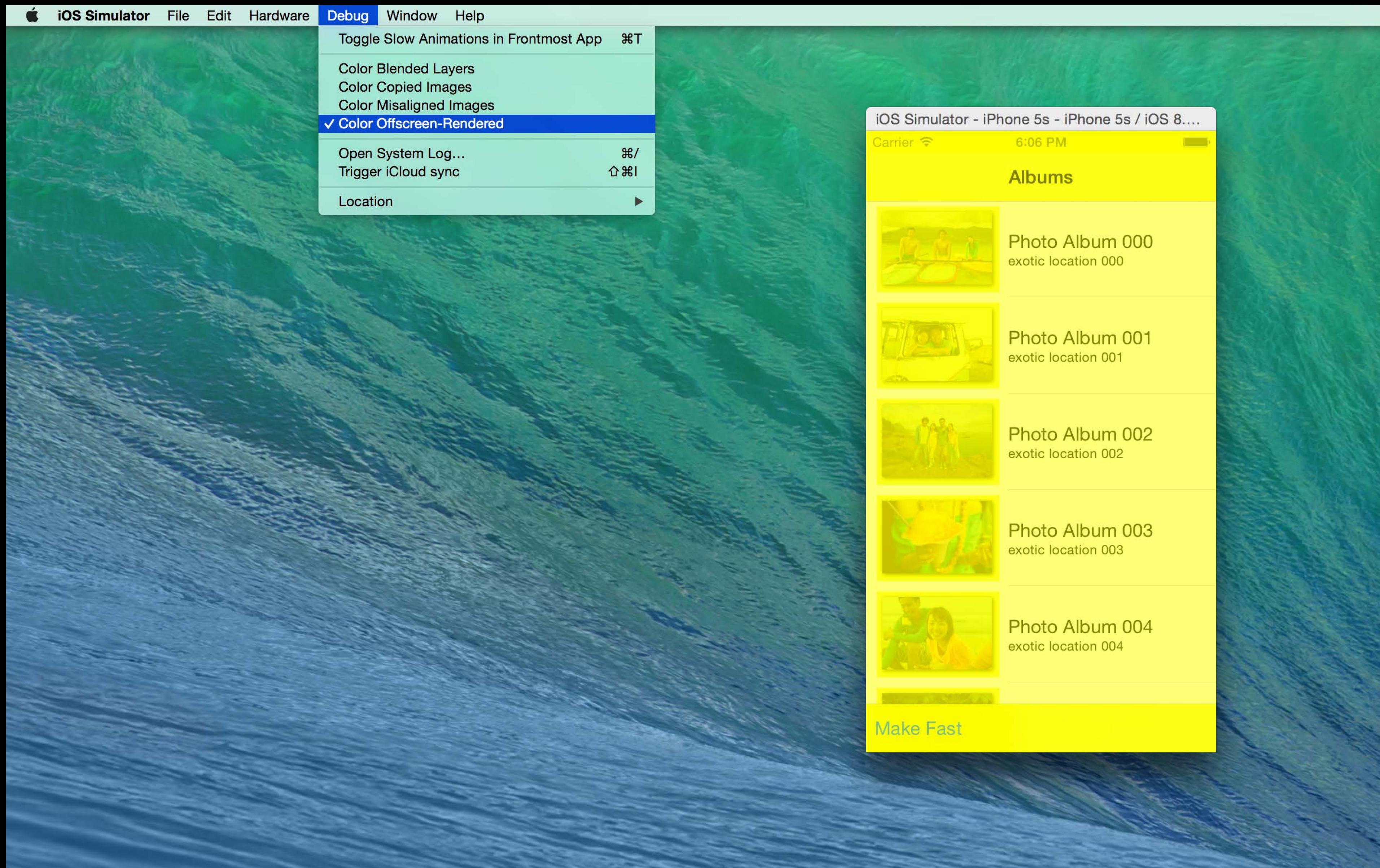
Too much blending?

Less is better

Any strange image formats or sizes?

Avoid on-the-fly conversions or resizing

iOS Simulator Coloring Options



Instruments

OpenGL ES Driver template



Choose a profiling template for: iPhone5s (v8.0) > All Processes

Standard Custom Recent Search

Energy Diagnostics Leaks Network OpenGL ES Analysis OpenGL ES Driver System Trace

System Usage Time Profiler Zombies

OpenGL ES Driver
This template measures OpenGL ES graphics performance as well as CPU usage of a process.

Open an Existing File... Cancel Choose

Instruments

OpenGL ES Driver template



Choose a profiling template for: iPhone5s (v8.0) > All Processes

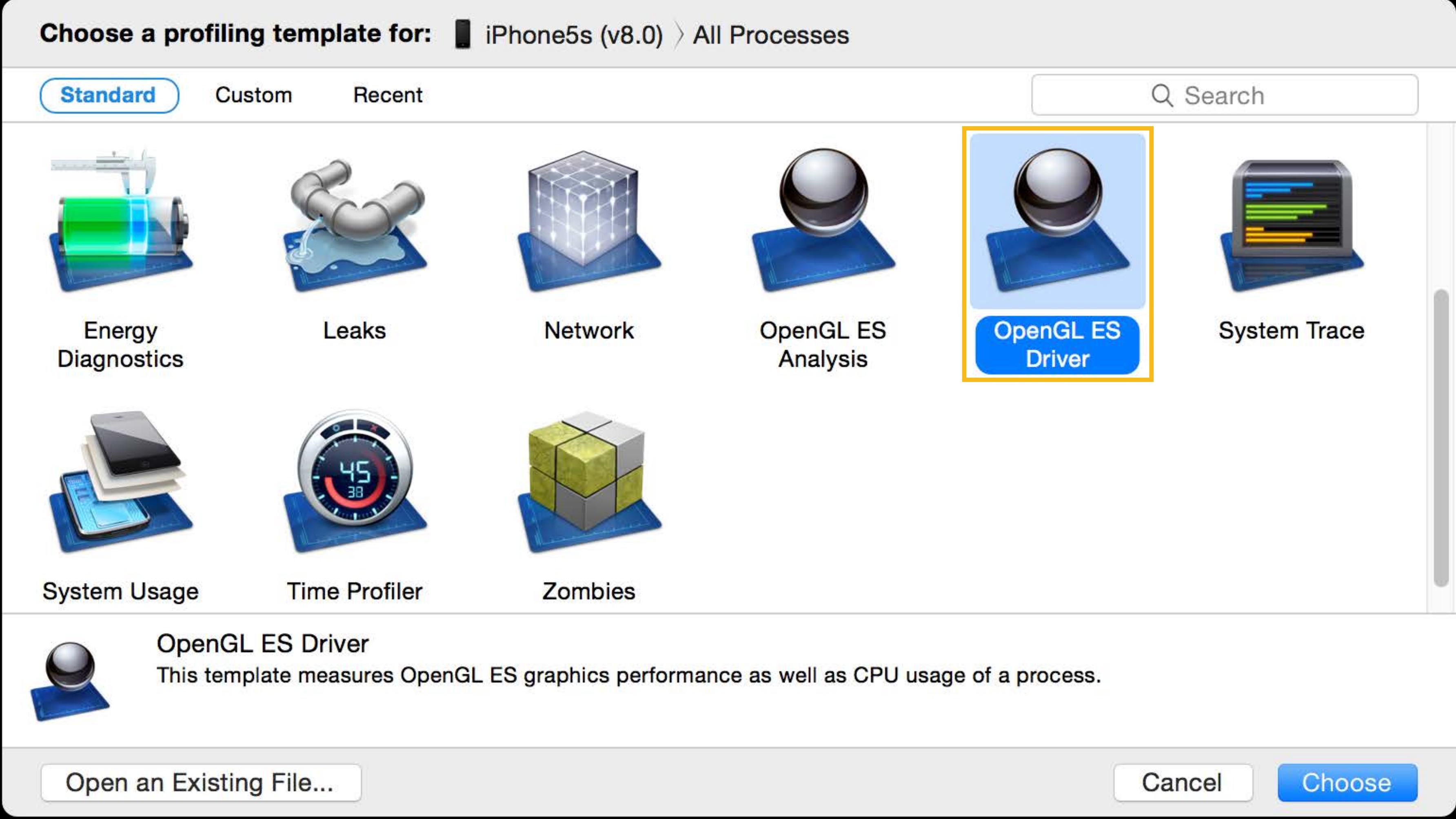
Standard Custom Recent Search

Energy Diagnostics Leaks Network OpenGL ES Analysis System Trace

System Usage Time Profiler Zombies

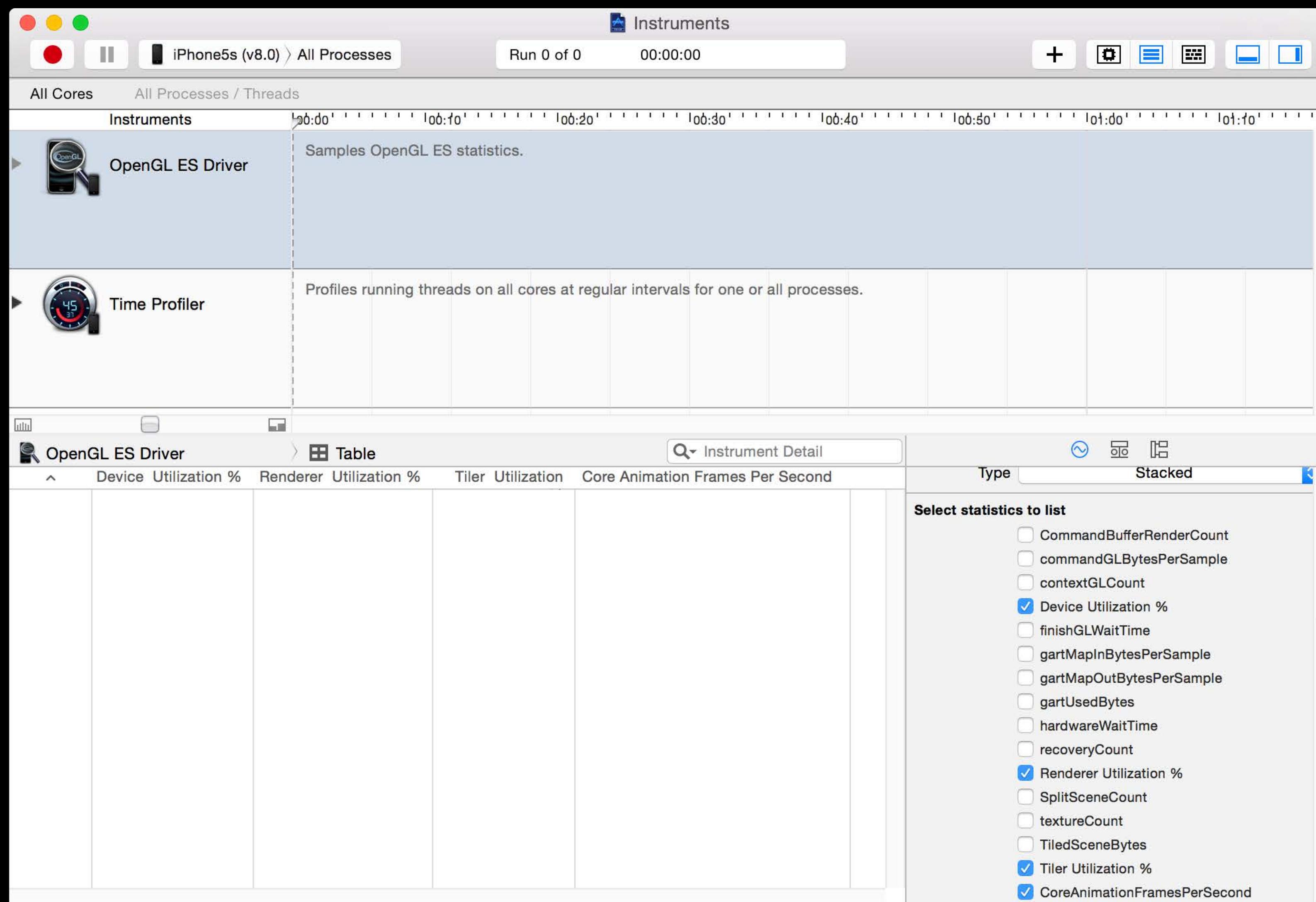
OpenGL ES Driver
This template measures OpenGL ES graphics performance as well as CPU usage of a process.

Open an Existing File... Cancel Choose



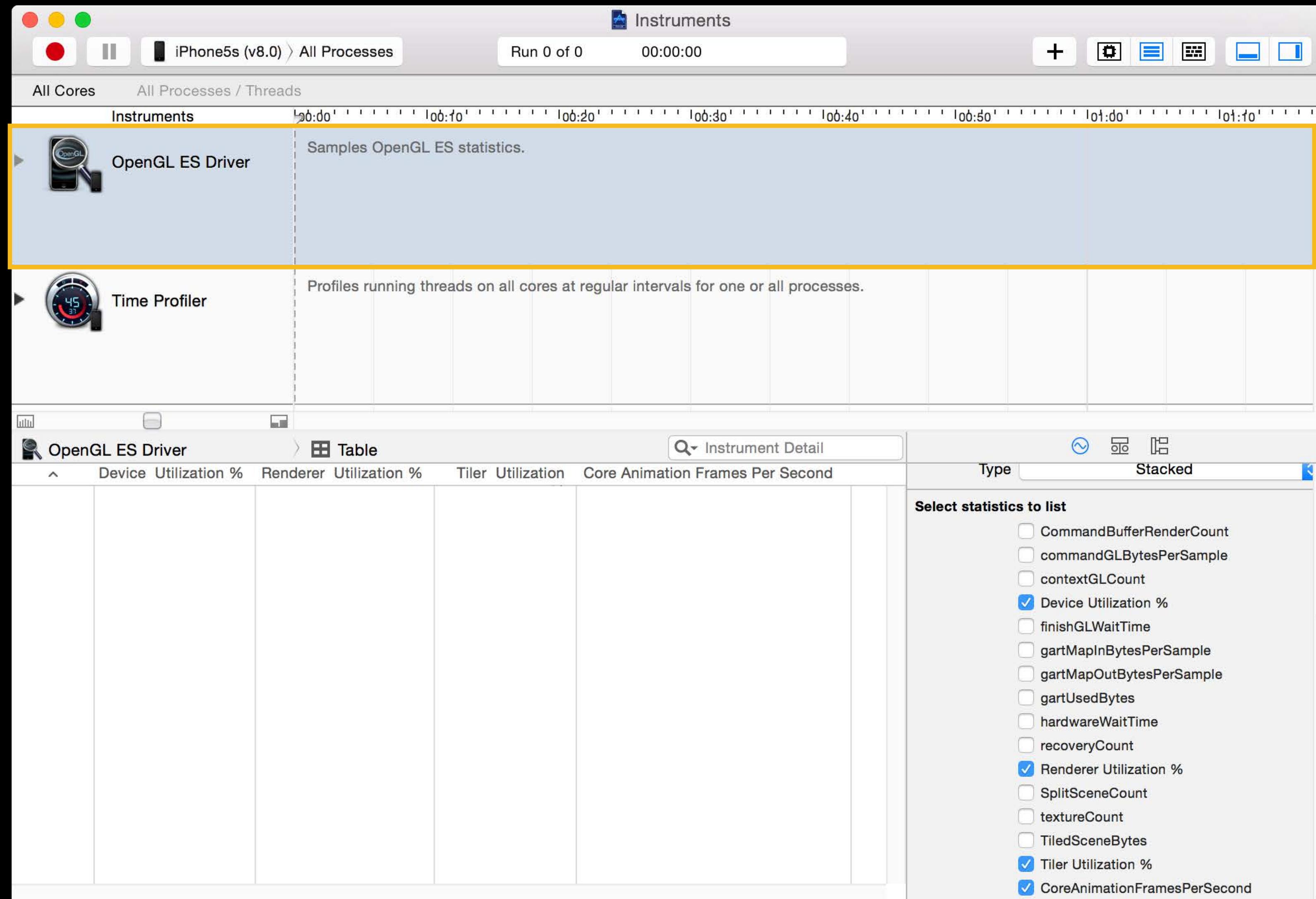
OpenGL ES Driver Instrument

Selecting statistics to list



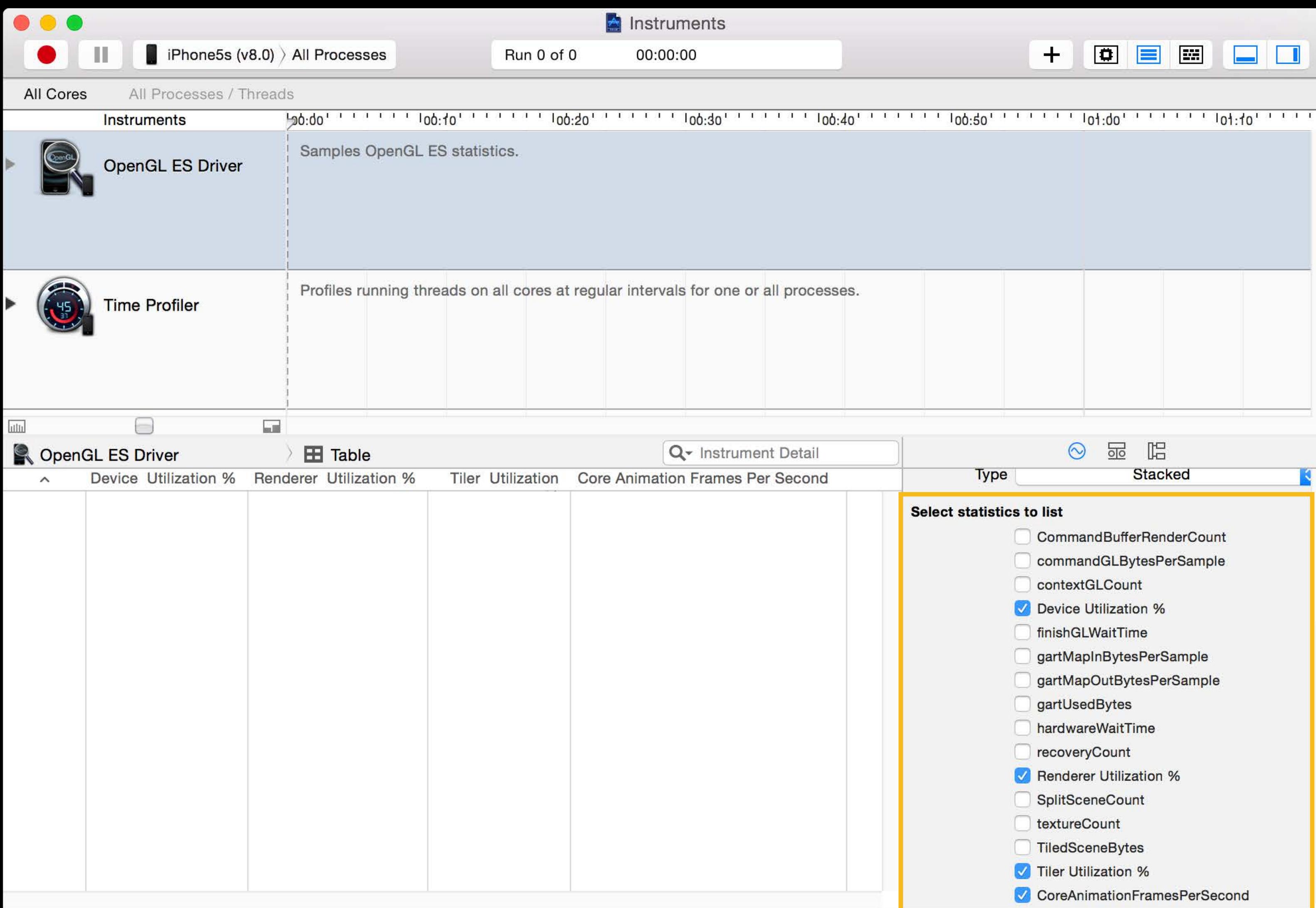
OpenGL ES Driver Instrument

Selecting statistics to list



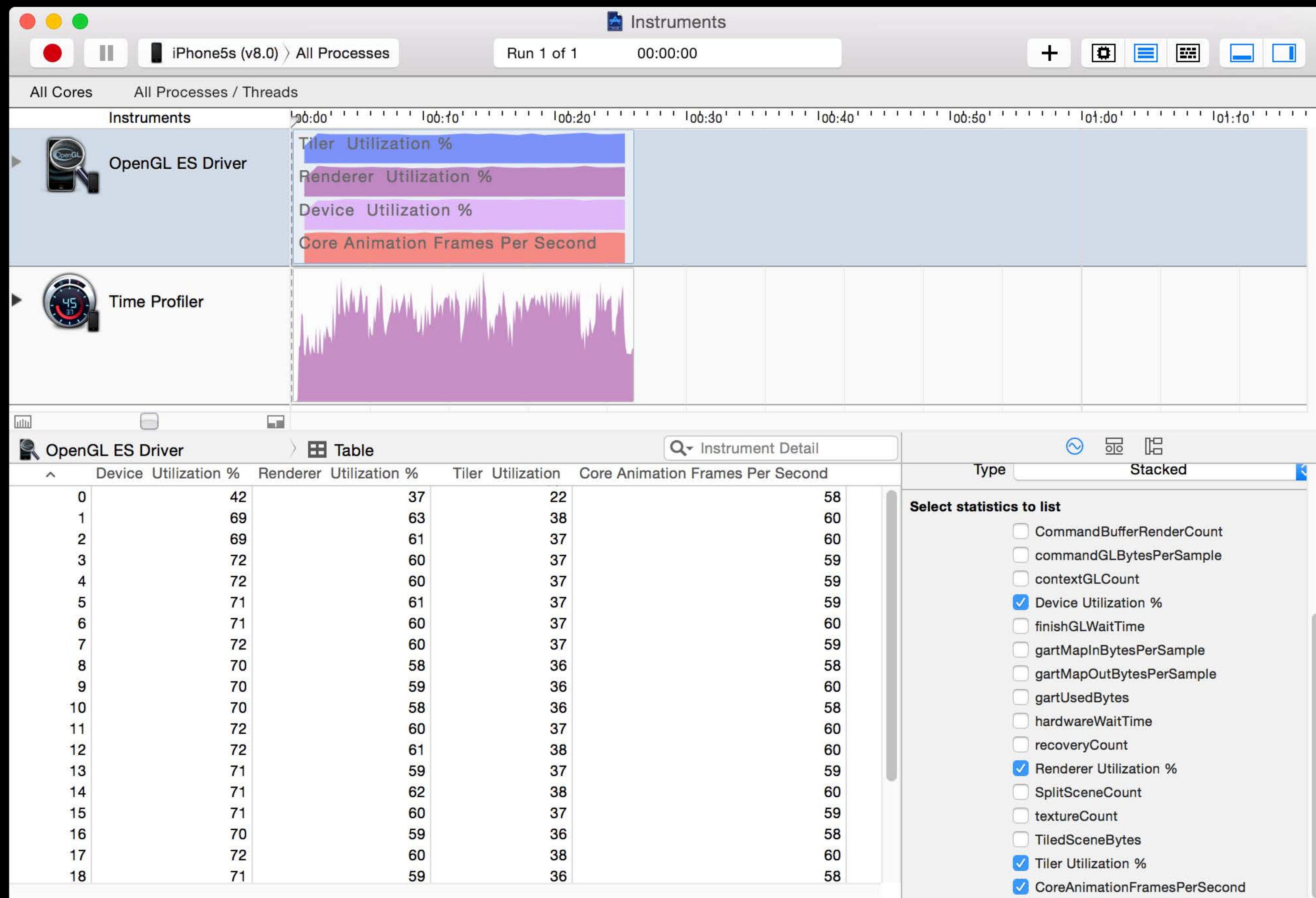
OpenGL ES Driver Instrument

Selecting statistics to list



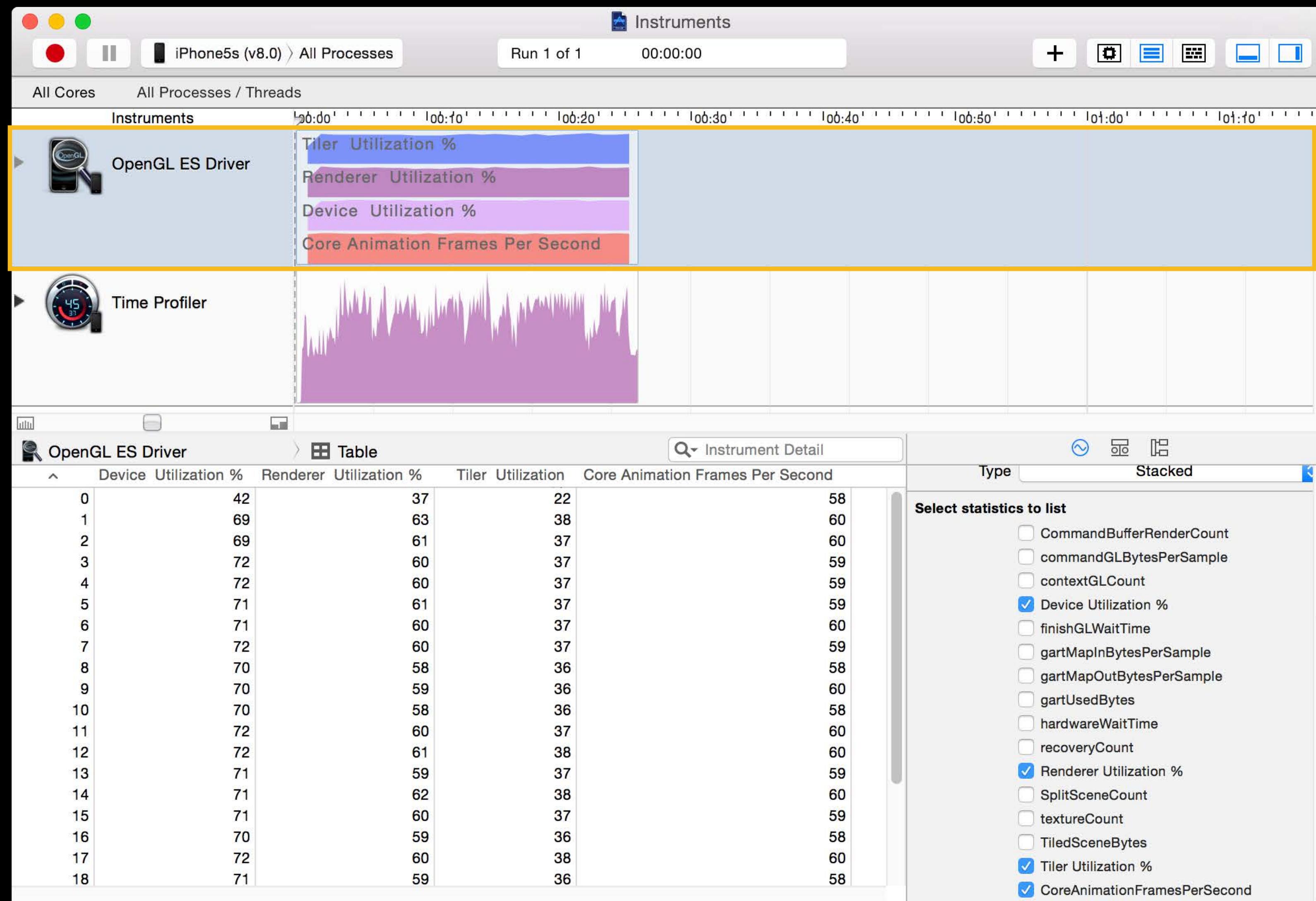
OpenGL ES Driver Instrument

GPU utilization



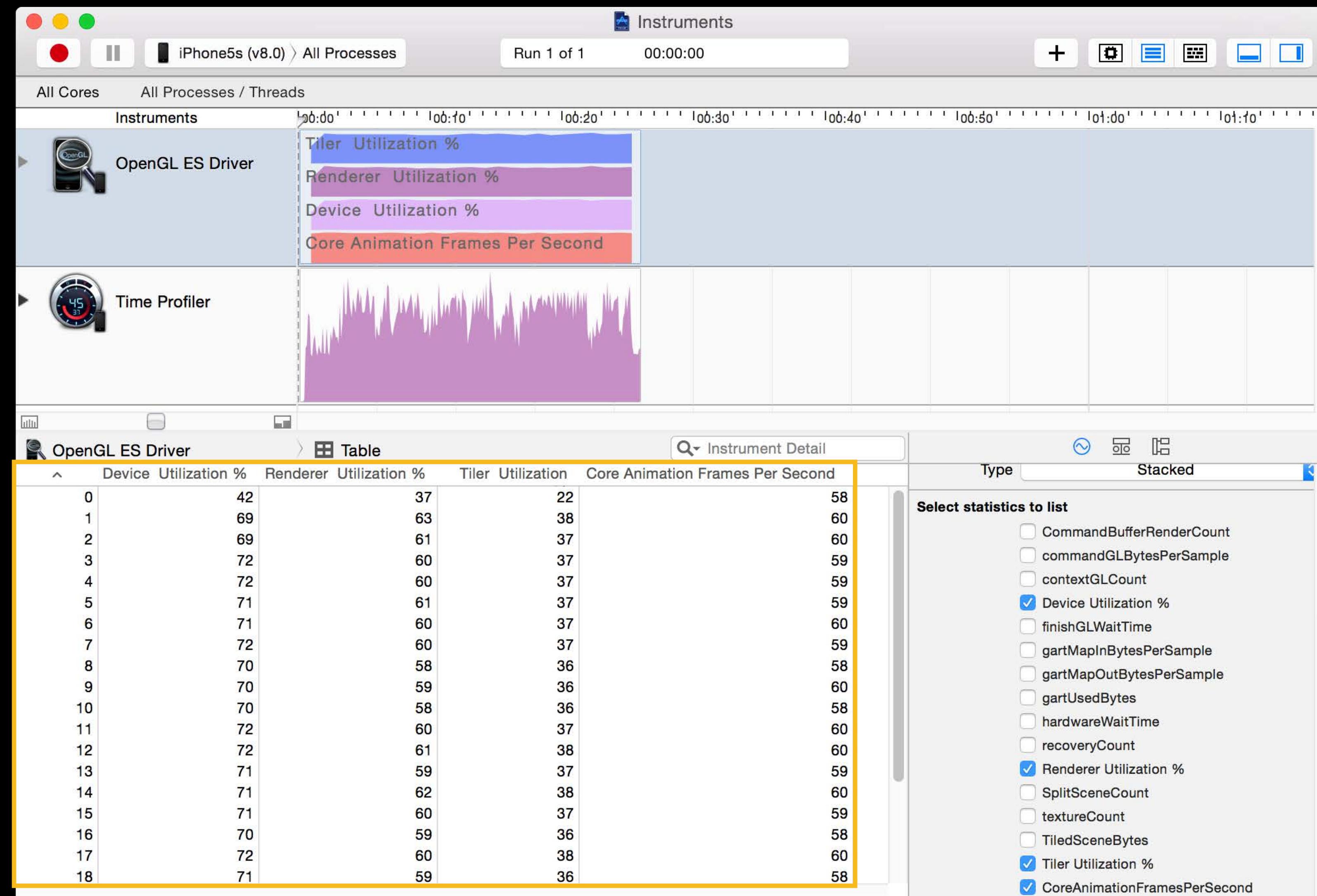
OpenGL ES Driver Instrument

GPU utilization



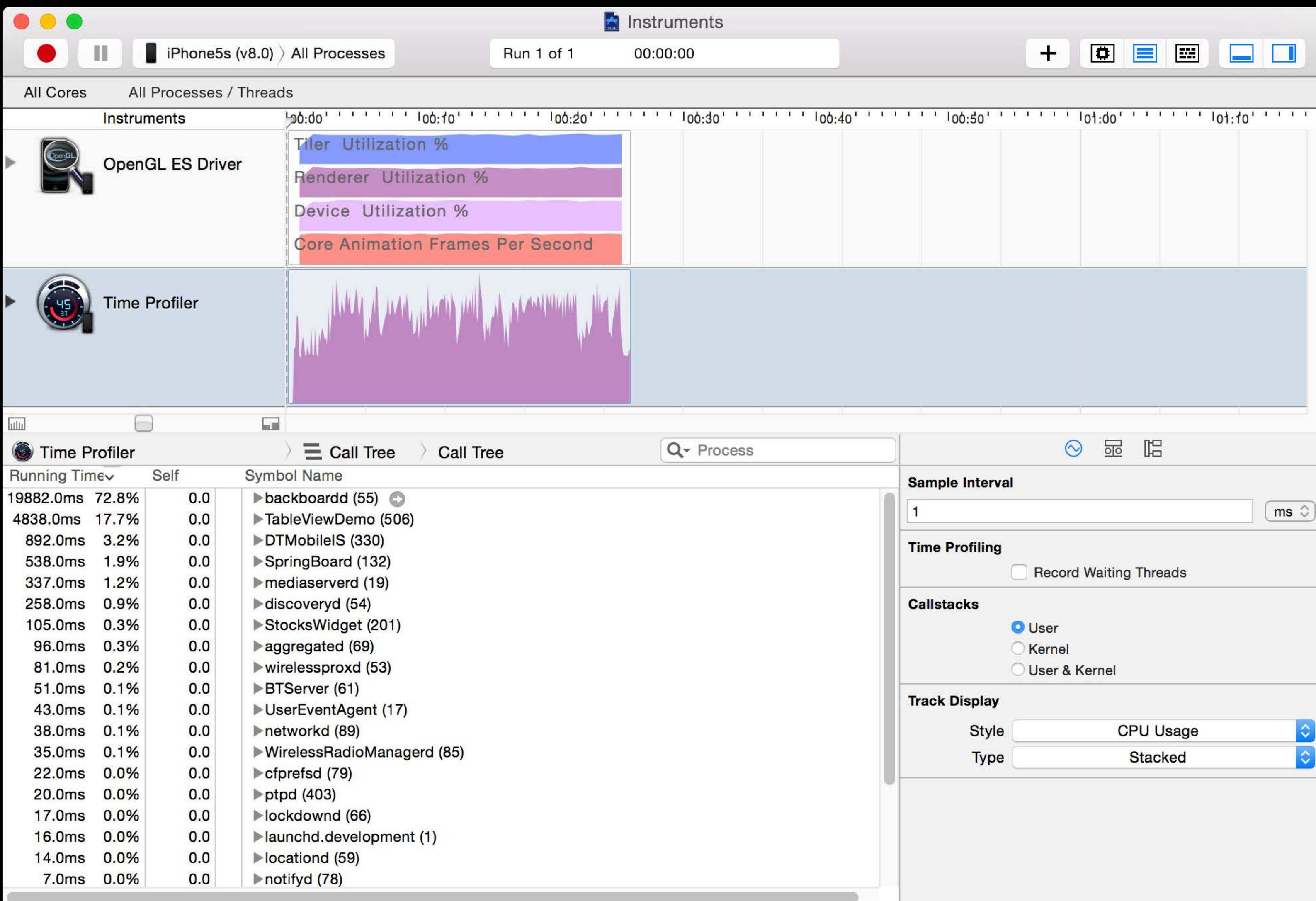
OpenGL ES Driver Instrument

GPU utilization



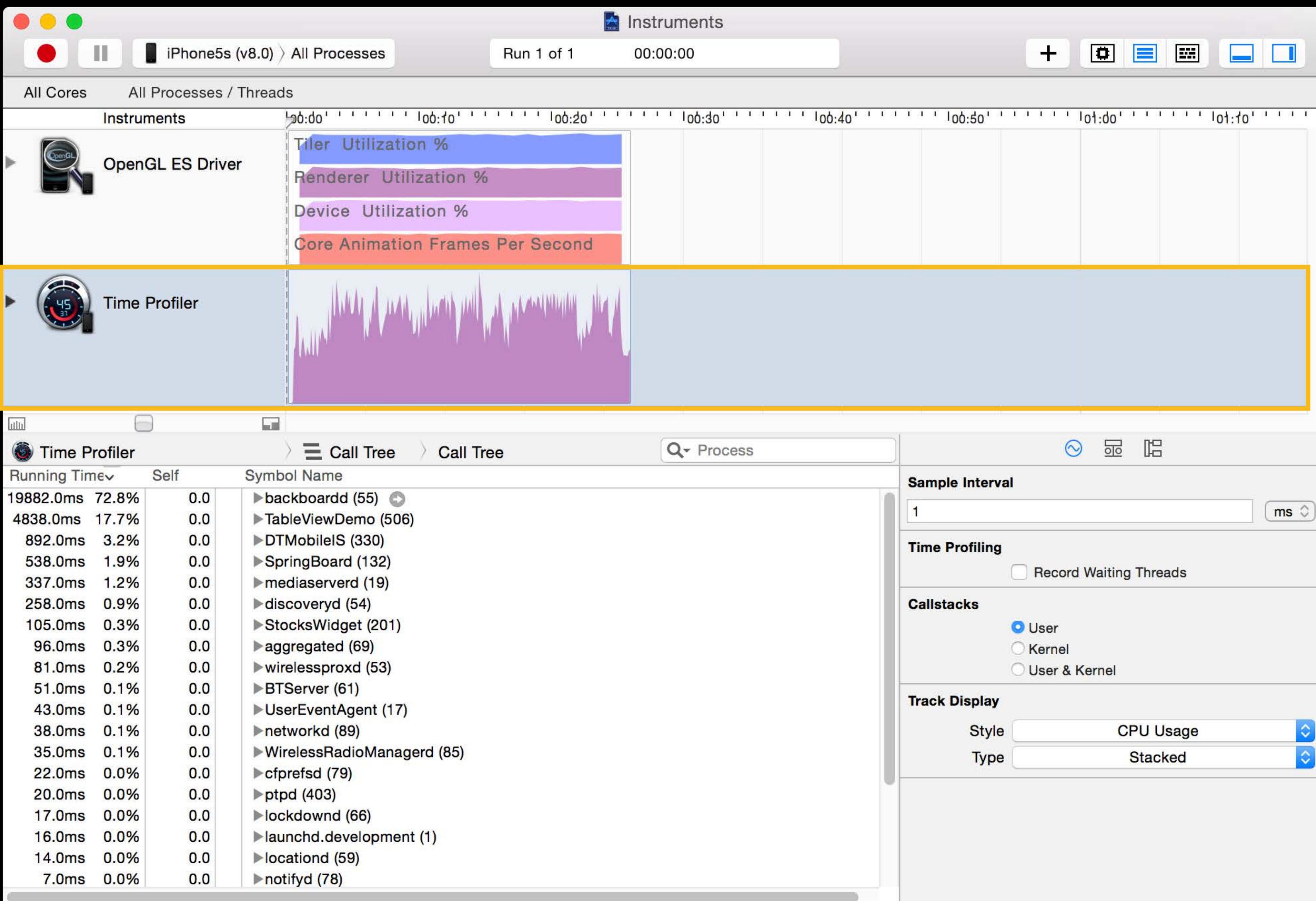
Time Profiler Instrument

CPU utilization



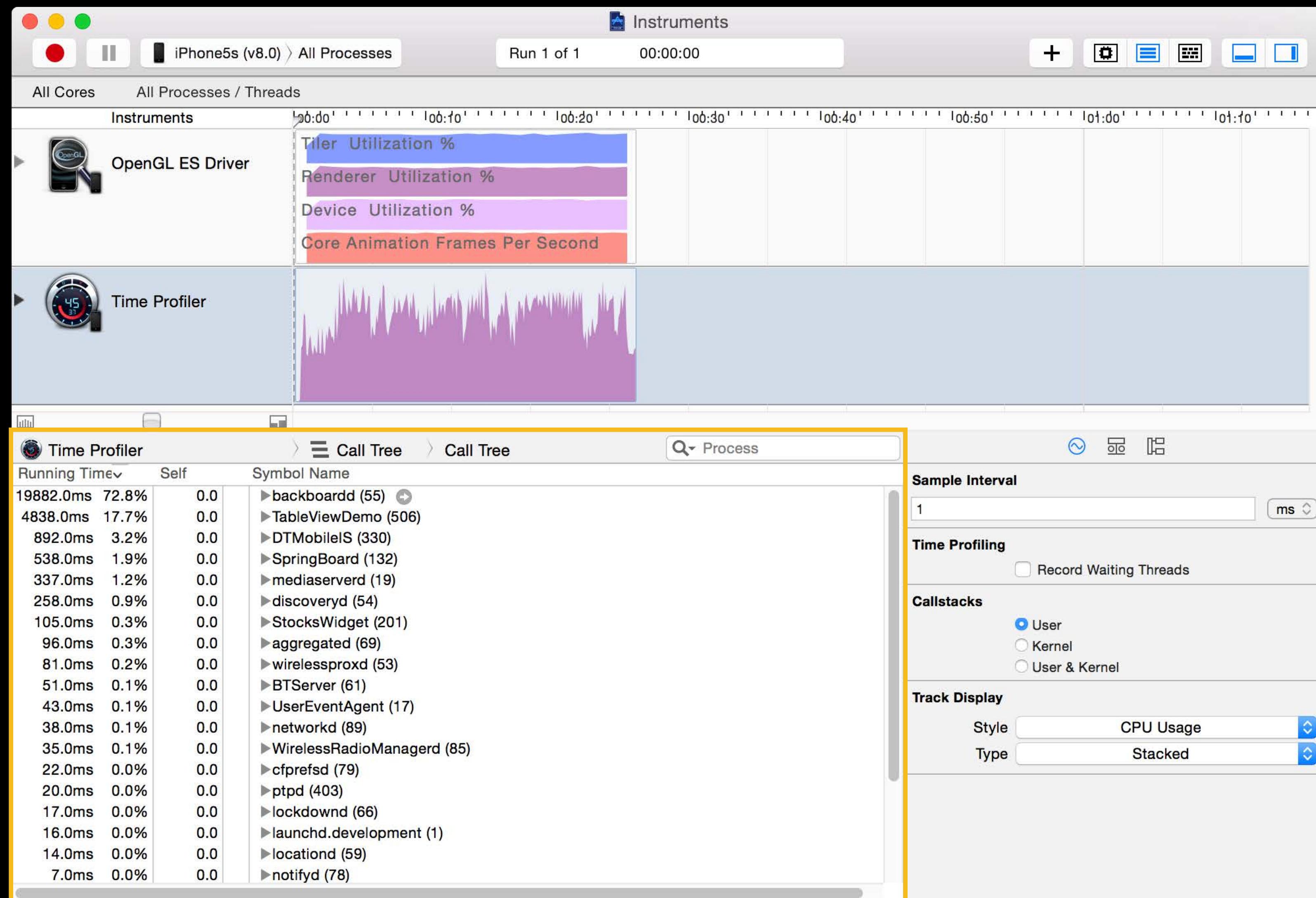
Time Profiler Instrument

CPU utilization



Time Profiler Instrument

CPU utilization



Performance Investigation Mindset

OpenGL ES Driver instrument summary

What is the frame rate?

Goal is always 60 frames per second

CPU or GPU bound?

Lower utilization is desired and saves battery

Any unnecessary CPU rendering?

GPU is desirable but know when CPU make sense

Xcode

View debugging



Screenshot of Xcode's View Debugger interface, showing a UITableView displaying photo albums.

The left sidebar shows the view hierarchy:

- TableViewDemo
- PID 791, Paused
- UIWindow
- UILayoutContainerView
- UINavigationTransitionView
- UIViewControllerWrapperView
- UIView
- UITableView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UIImageView
- UIImageView
- Constraints

The main area displays a table view with six rows, each representing a photo album:

Photo Album	Description
Photo Album 000	exotic location 000
Photo Album 001	exotic location 001
Photo Album 002	exotic location 002
Photo Album 003	exotic location 003
Photo Album 004	exotic location 004
Photo Album 005	exotic location 005

The right sidebar provides details about a selected UIImageView:

- Object**: UIImageView
- Class Name**: UIImageView
- Address**: 0x10b77ce70
- Image View**: Shows a thumbnail image of a couple.
- Highlighted**: No Image
- State**: Not Highlighted

Xcode

View debugging



Screenshot of Xcode's View Debugger interface, showing a UITableView displaying photo albums.

The left sidebar shows the view hierarchy:

- TableViewDemo
- PID 791, Paused
- UIWindow
- UILayoutContainerView
- UINavigationTransitionView
- UIViewControllerWrapperView
- UIView
- UITableView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UITableViewCell
- UITableViewCellContentView
- UILabel
- UITableViewCellLabel
- UIImageView
- _UITableViewCellSeparatorView
- UIImageView
- UIImageView
- Constraints

The main area displays a table view with six rows, each representing a photo album:

Photo Album	Description
Photo Album 000	exotic location 000
Photo Album 001	exotic location 001
Photo Album 002	exotic location 002
Photo Album 003	exotic location 003
Photo Album 004	exotic location 004
Photo Album 005	exotic location 005

The right sidebar provides details about a selected UIImageView:

- Object**: UIImageView
- Class Name**: UIImageView
- Address**: 0x10b77ce70
- Image View**: Shows a thumbnail image of a couple.
- Highlighted**: No Image
- State**: Not Highlighted

Xcode

View debugging



Running TableViewDemo on iPhone 5s

Object
Class Name UIImageView
Address 0x10b77ce70

Image View
Image
Highlighted No Image

State Not Highlighted

Albums

	Photo Album 000 exotic location 000
	Photo Album 001 exotic location 001
	Photo Album 002 exotic location 002
	Photo Album 003 exotic location 003
	Photo Album 004 exotic location 004
	Photo Album 005 exotic location 005

TableViewDemo PID 791, Paused

UIWindow

UILayoutContainerView

UINavigationTransitionView

UIViewControllerWrapperView

UIView

UITableView

UITableViewCell

UITableViewCellContentView

UILabel

UITextView

UIImageView

_UITableViewCellSeparatorView

UIImageView

UIImageView

Constraints

Constraints

Table View Demo

Thread 1

0 mach_msg_trap

Xcode

View debugging



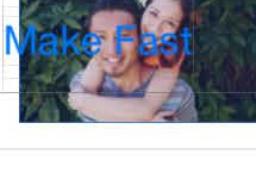
Running TableViewDemo on iPhone 5s

Object
Class Name UIImageView
Address 0x10b77ce70

Image View
Image 

Highlighted No Image
State Not Highlighted

Albums

	Photo Album 000 exotic location 000
	Photo Album 001 exotic location 001
	Photo Album 002 exotic location 002
	Photo Album 003 exotic location 003
	Photo Album 004 exotic location 004
	Photo Album 005 exotic location 005

Make Fast

TableViewDemo > Thread 1 > 0 mach_msg_trap

The screenshot shows the Xcode interface during view debugging. The left sidebar displays the view hierarchy for the 'TableViewDemo' project, specifically for the 'iPhone 5s' target. A yellow box highlights a UIImageView object located within a UITableViewCellContent view. The main canvas displays a table view with five visible cells, each containing a thumbnail image and a title like 'Photo Album 000'. The bottom status bar indicates the application is running and shows a stack trace with 'mach_msg_trap'.

Performance Investigation Mindset

Xcode view debugging summary

Any expensive views or effects?

Understand the cost of what is in use

Anything unexpected in hierarchy? Know the actual view hierarchy

Case Studies

Michael Ingrassia
iOS Software Engineer

Case Studies

Explore several scenarios

Measure performance on different devices

Keep the same appearance with better performance

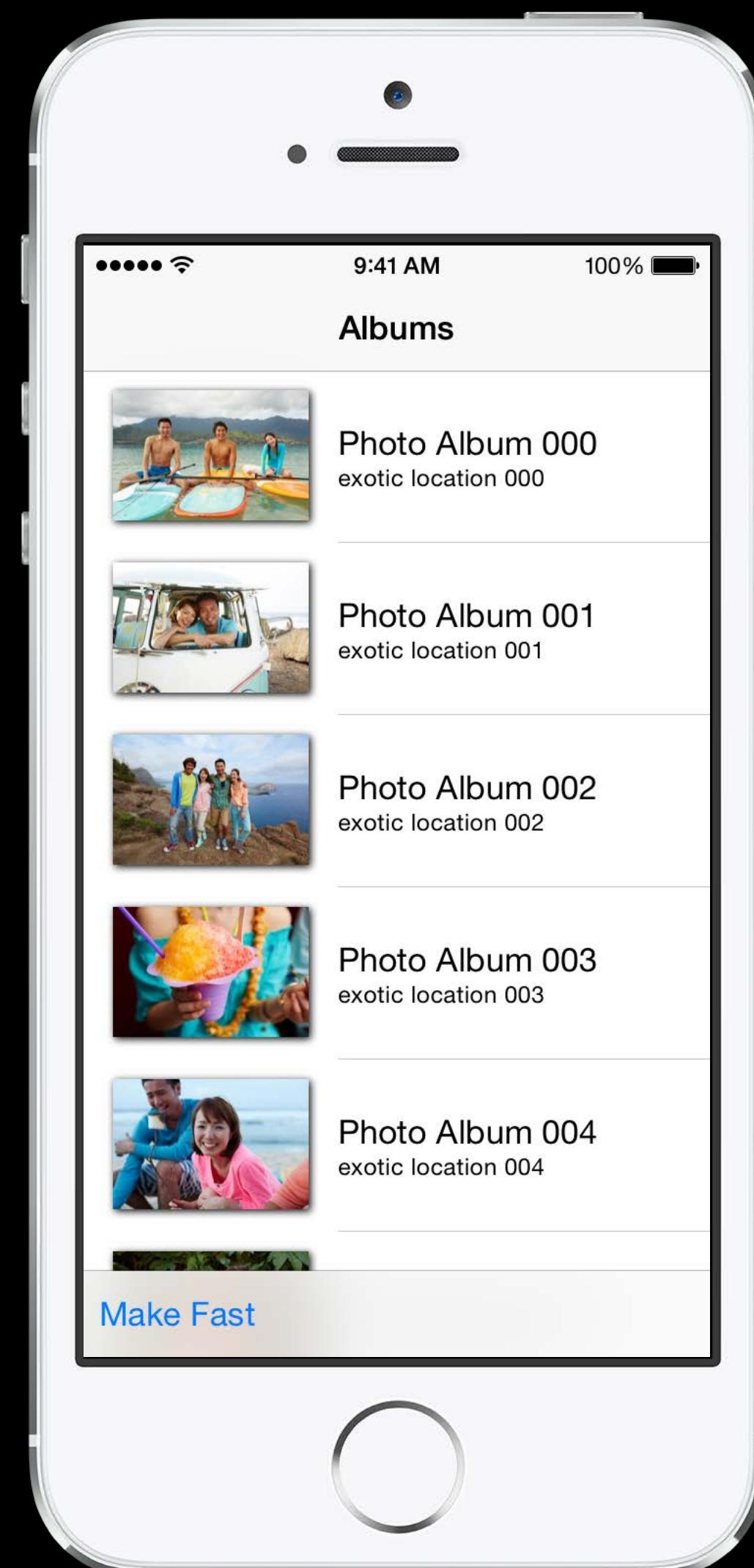
Fictitious Photo Application

Case study

Simple table view

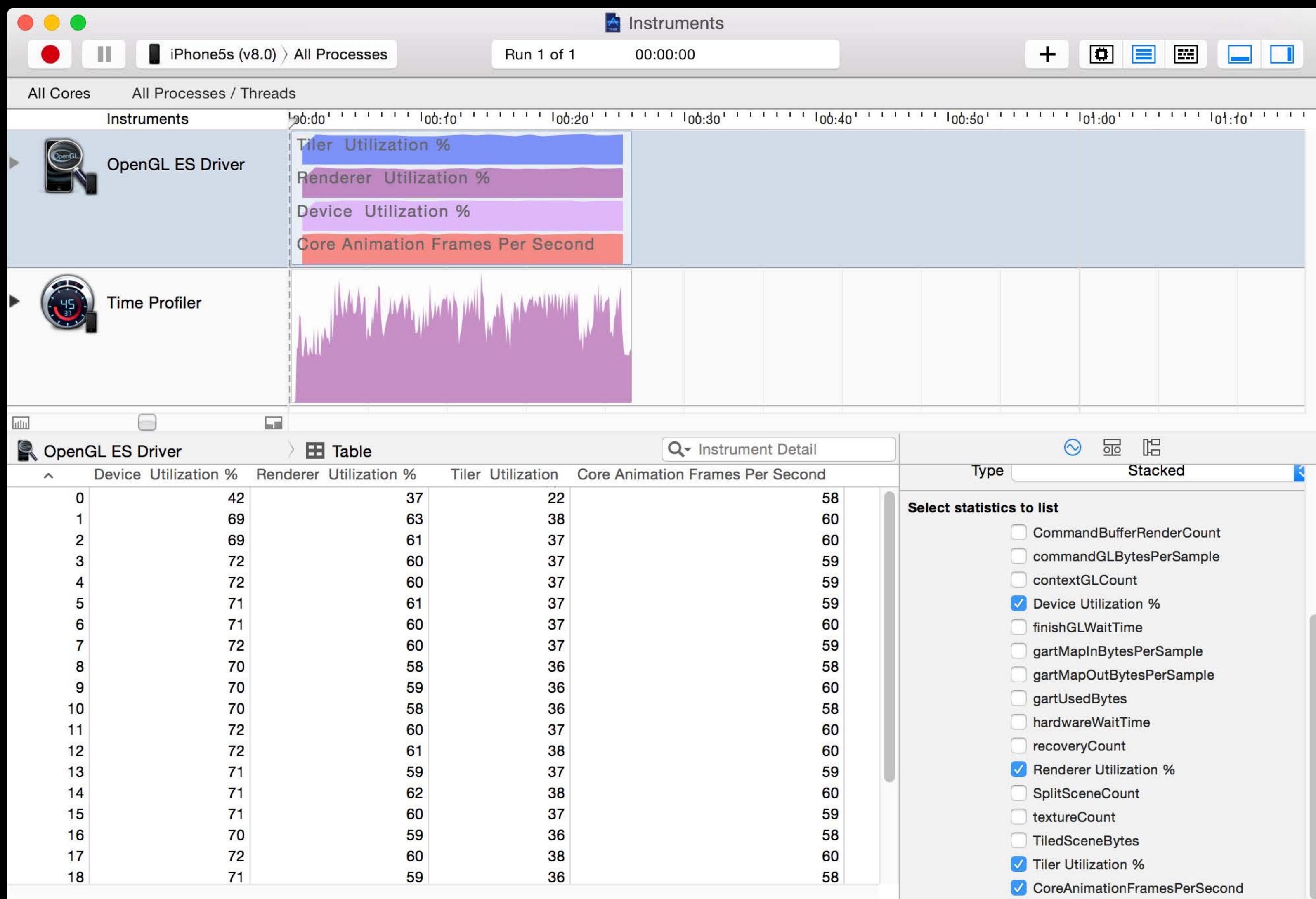
Each cell shows a photo thumbnail and some text

Each photo has a small shadow



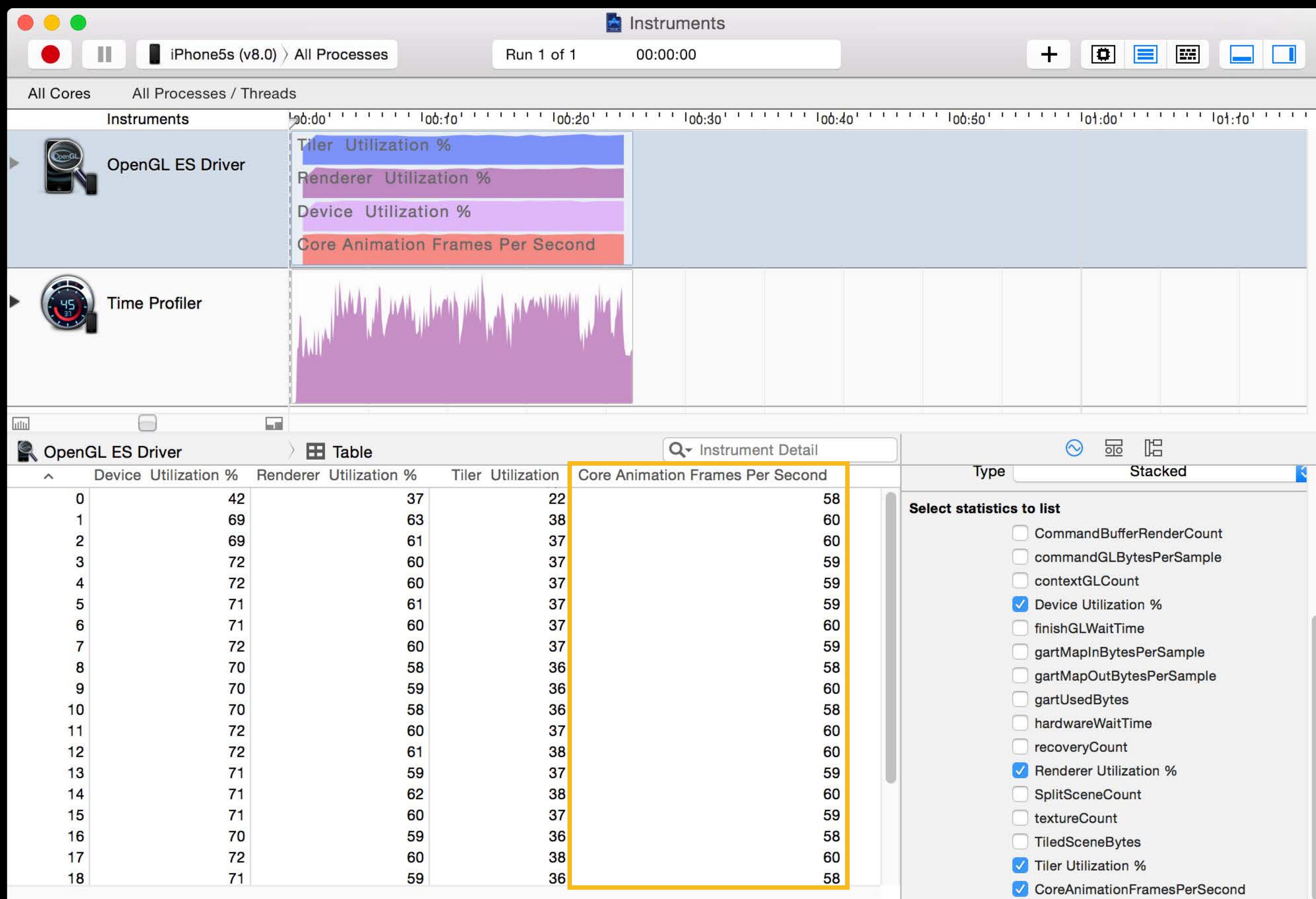
Measure Frame Rate on iPhone 5s

OpenGL ES Driver instrument



Measure Frame Rate on iPhone 5s

OpenGL ES Driver instrument

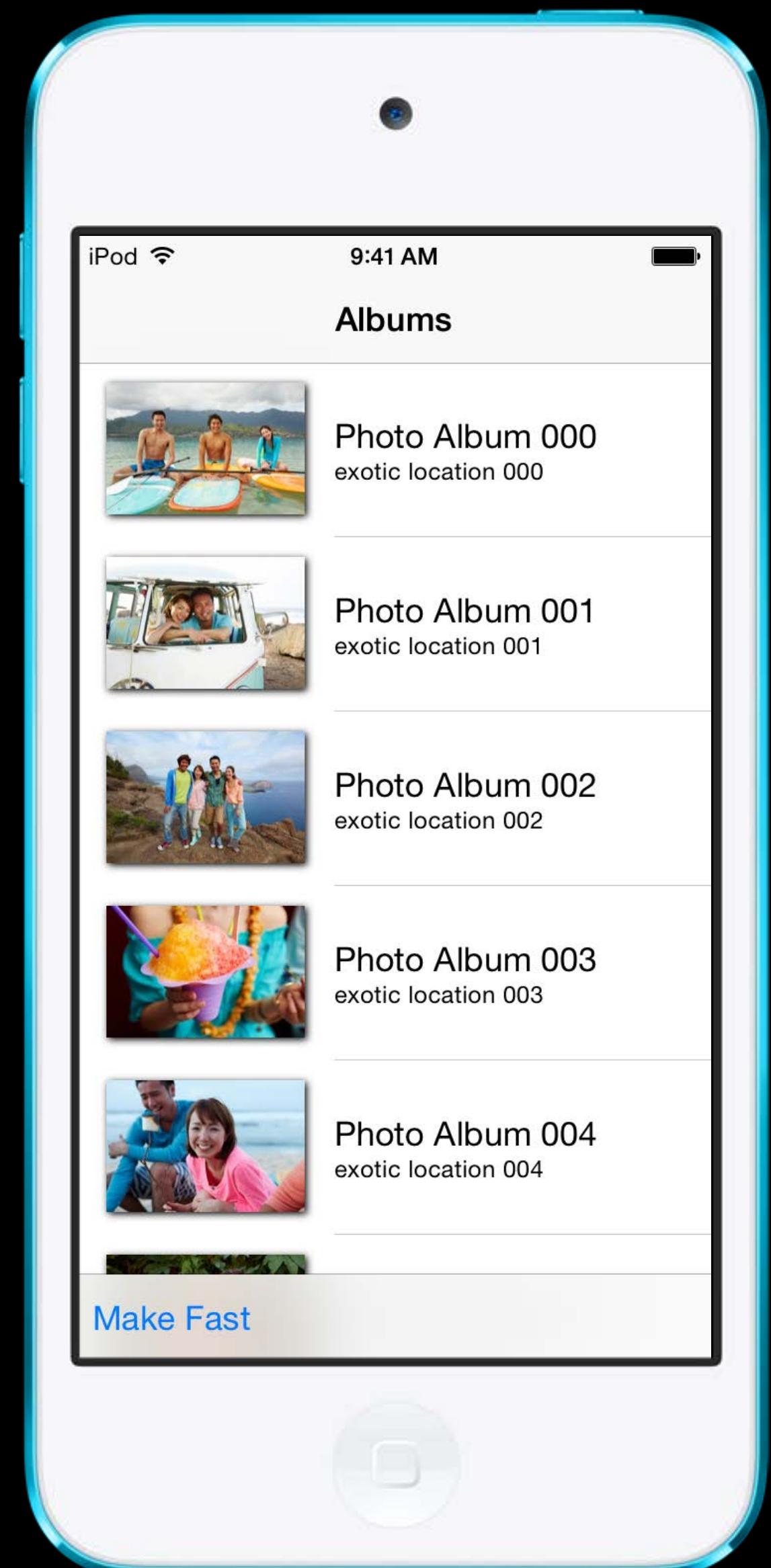


Awesome
Ship it?

Fictitious Photo Application

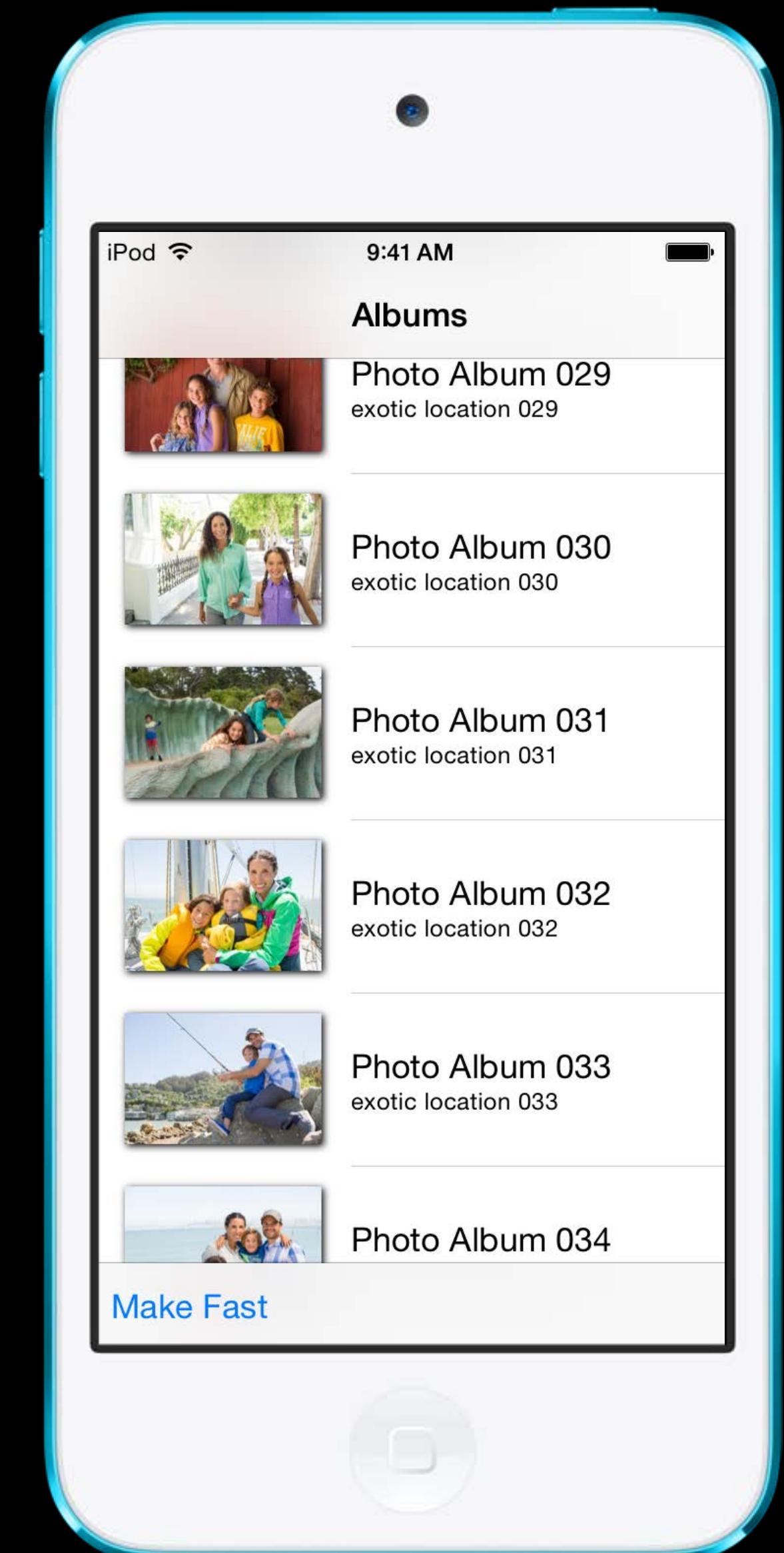
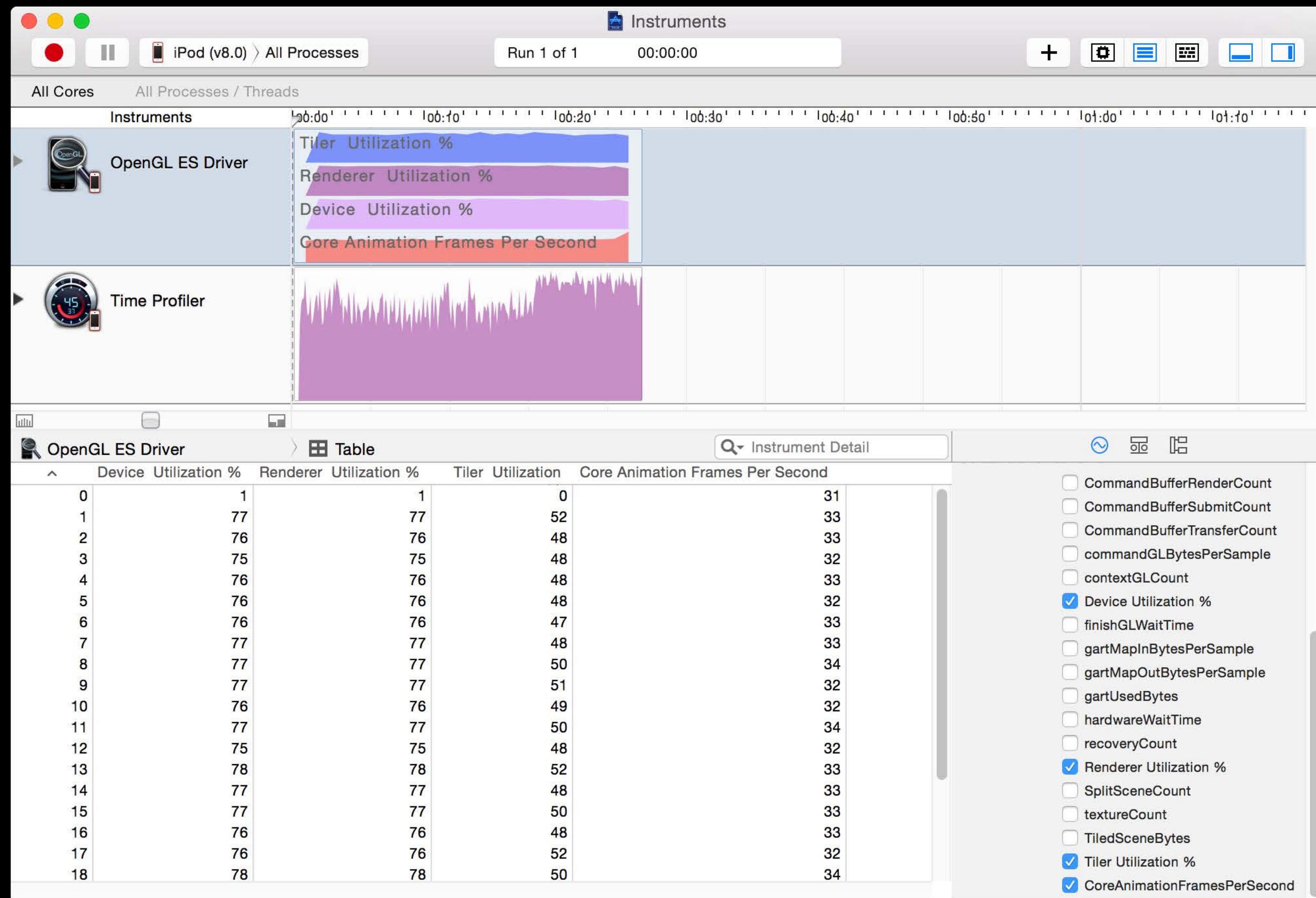
iPod touch scrolling performance

What about the performance on other devices?



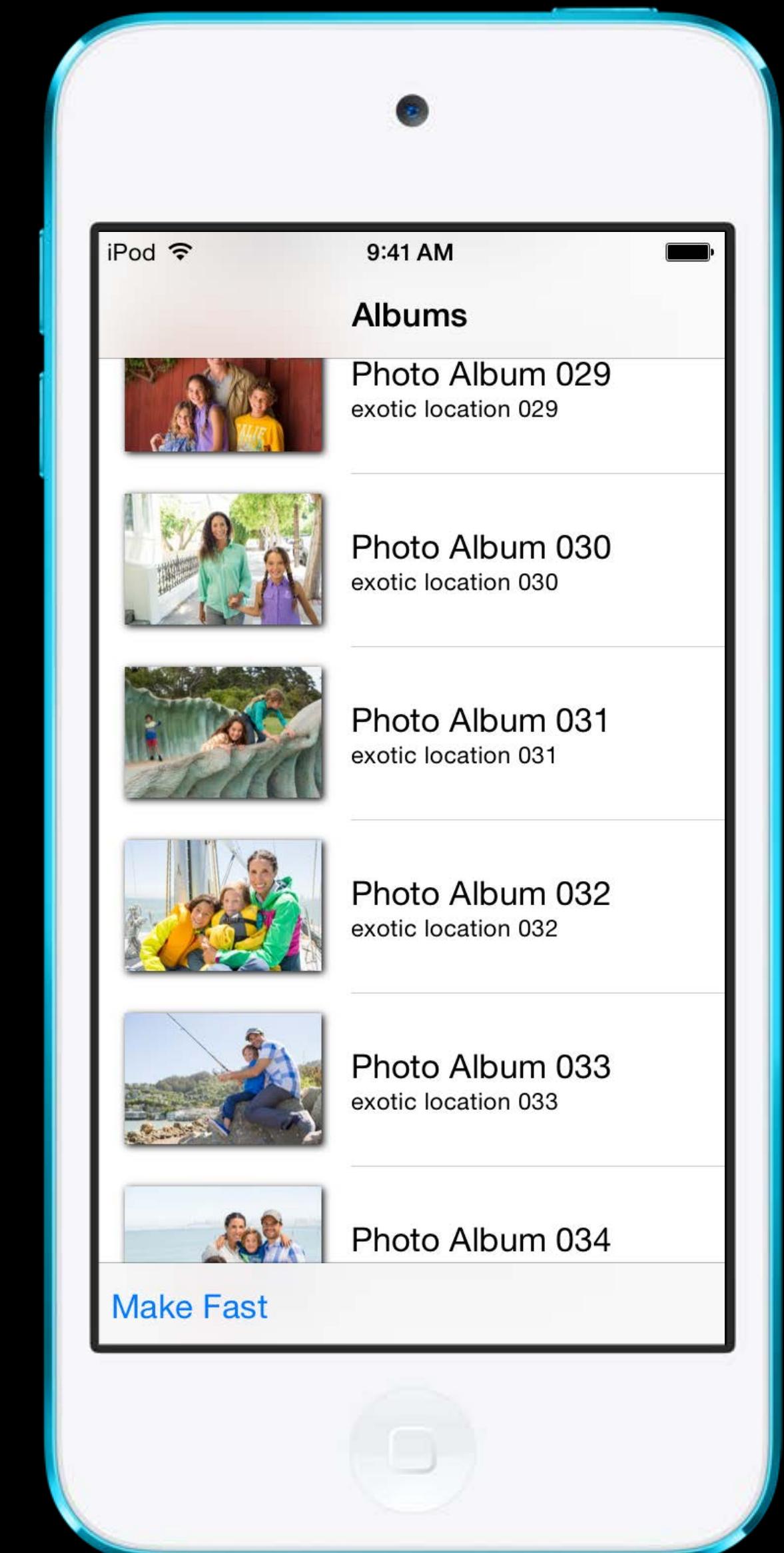
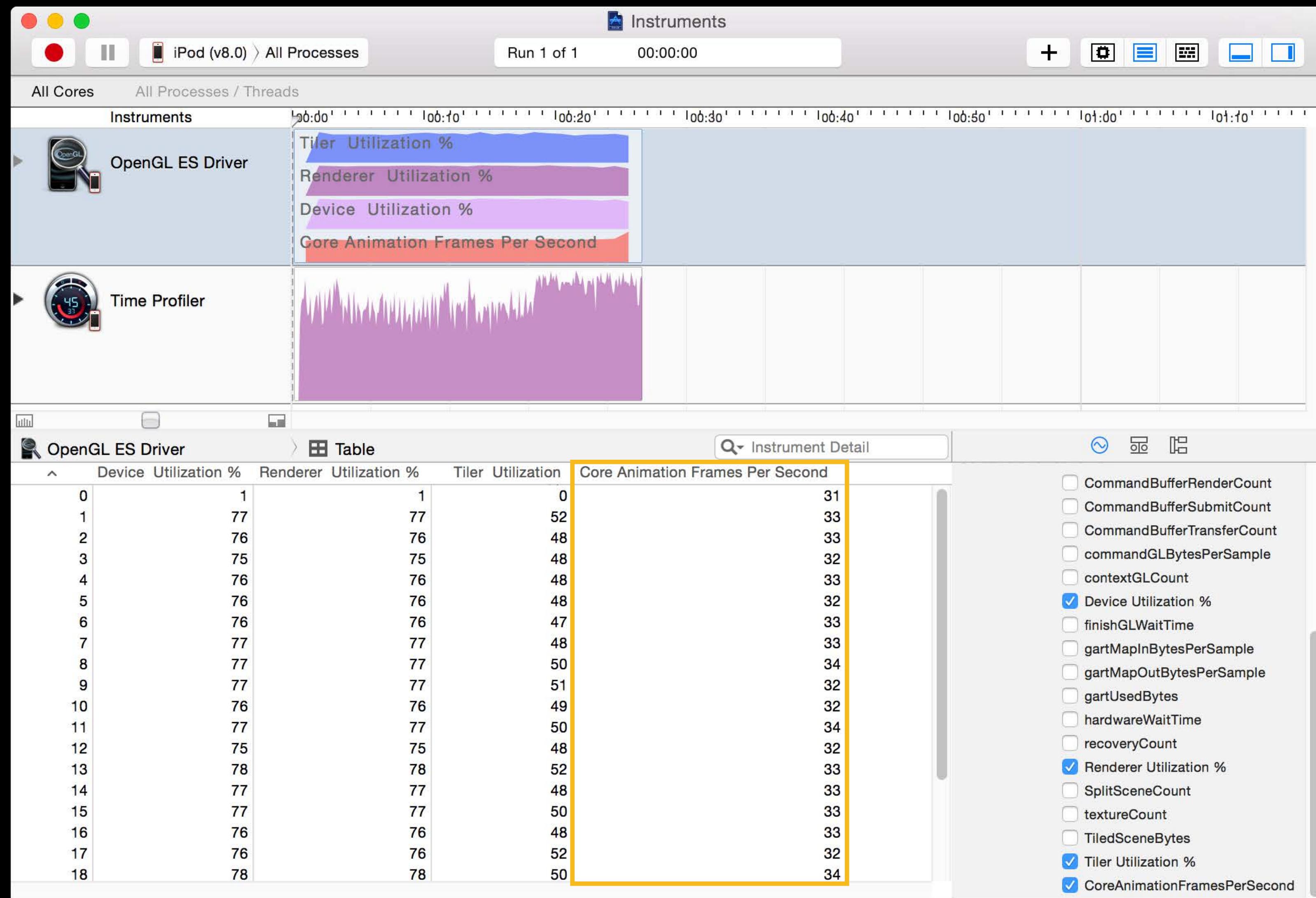
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



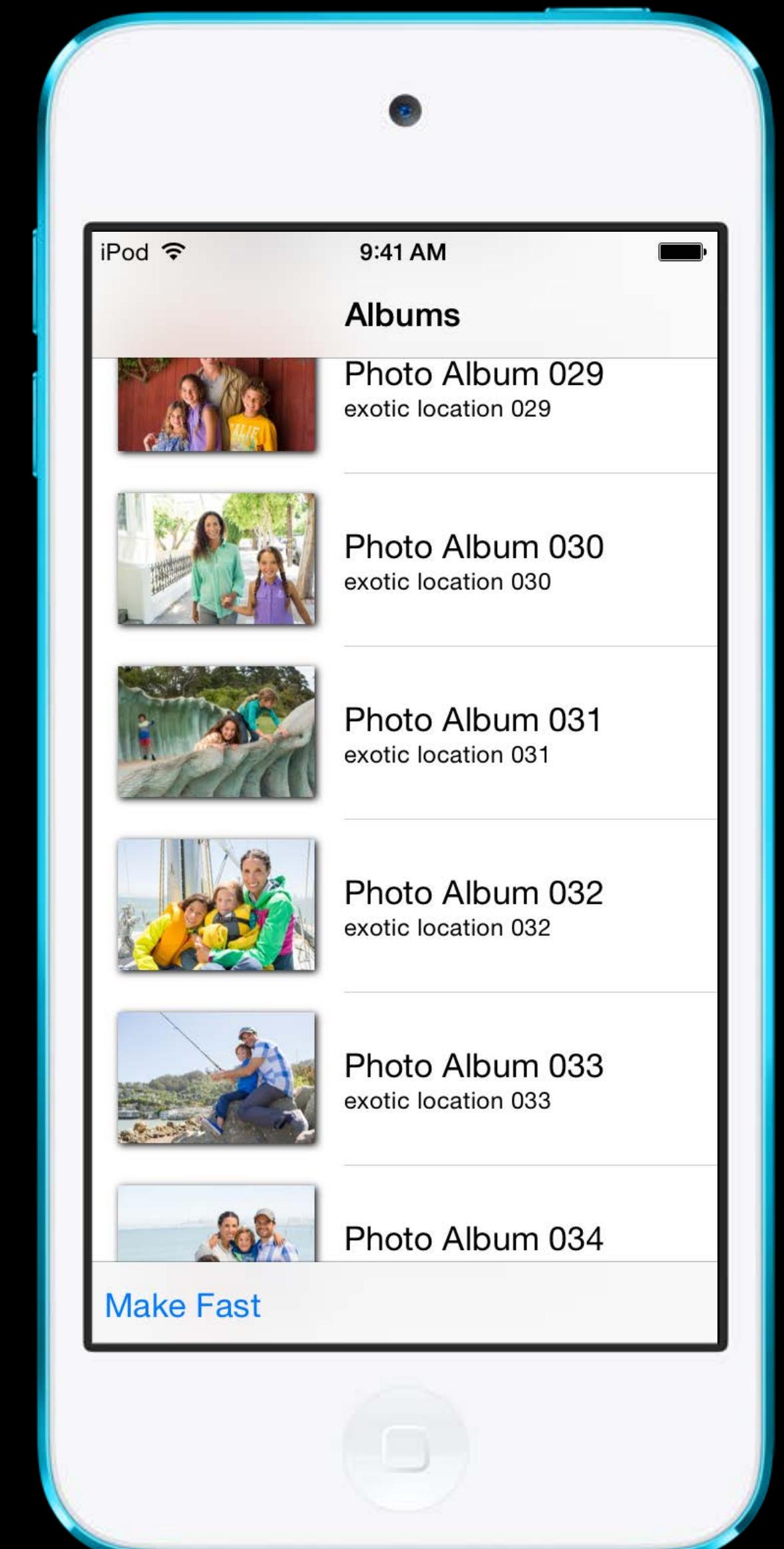
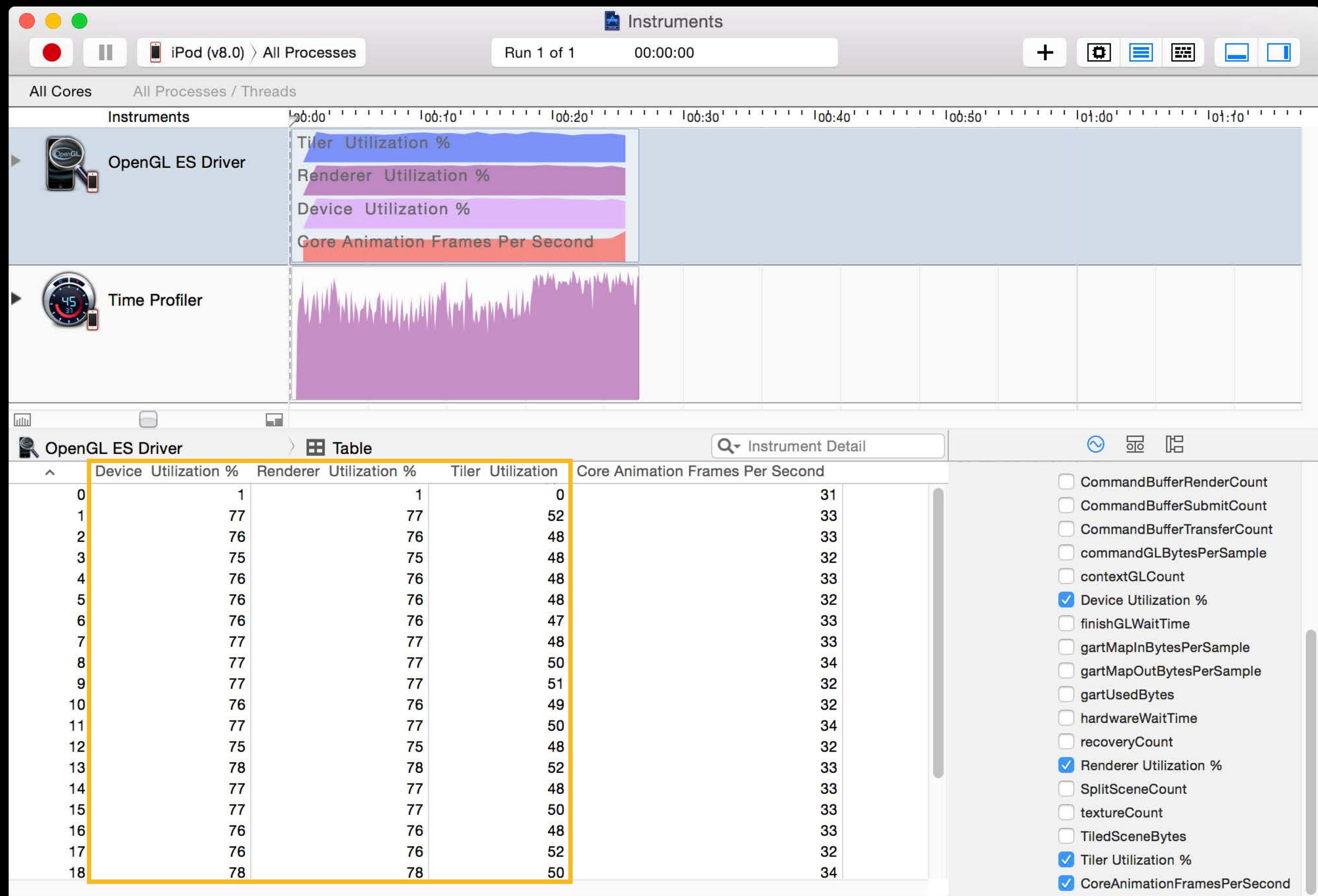
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



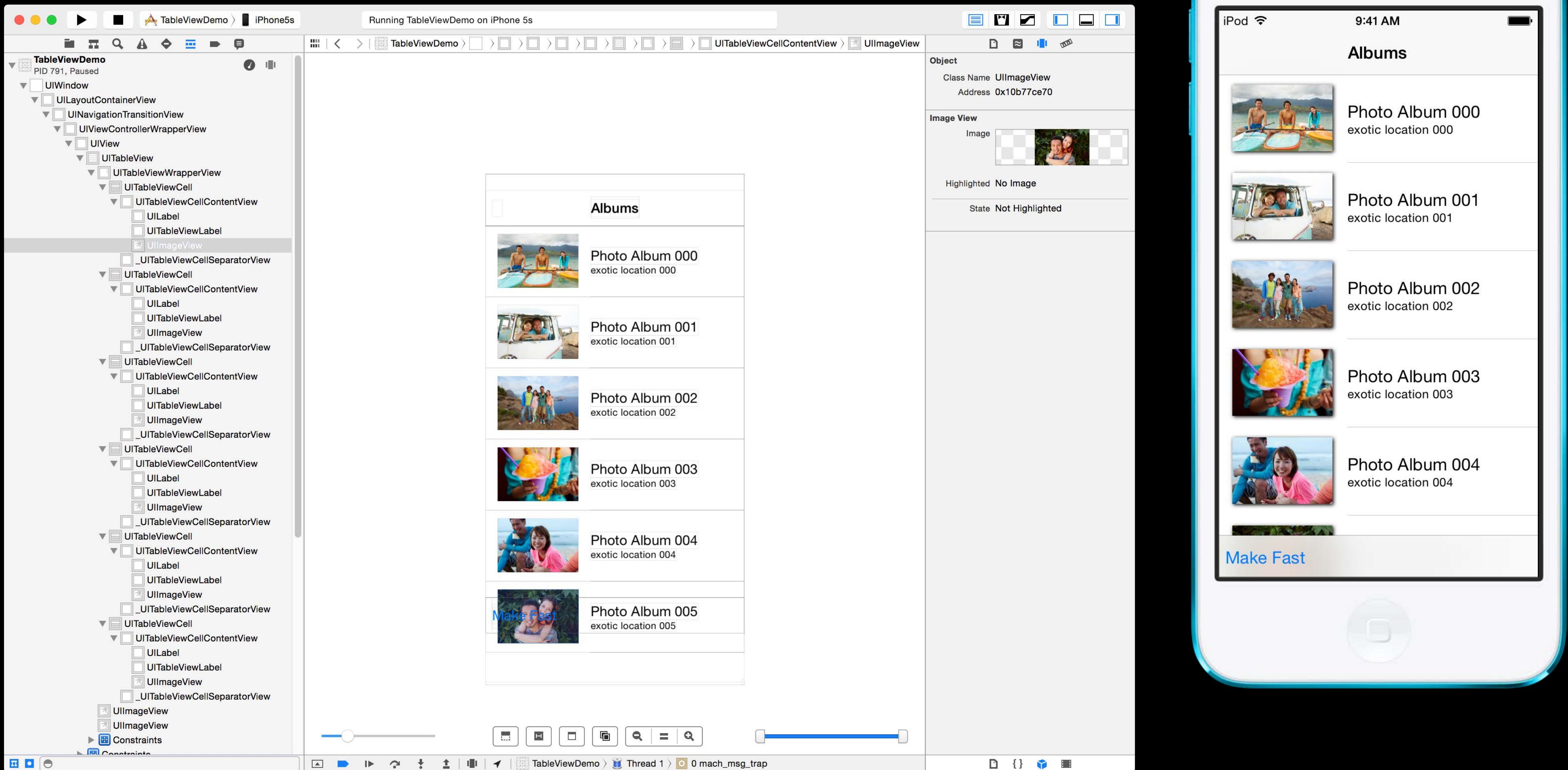
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



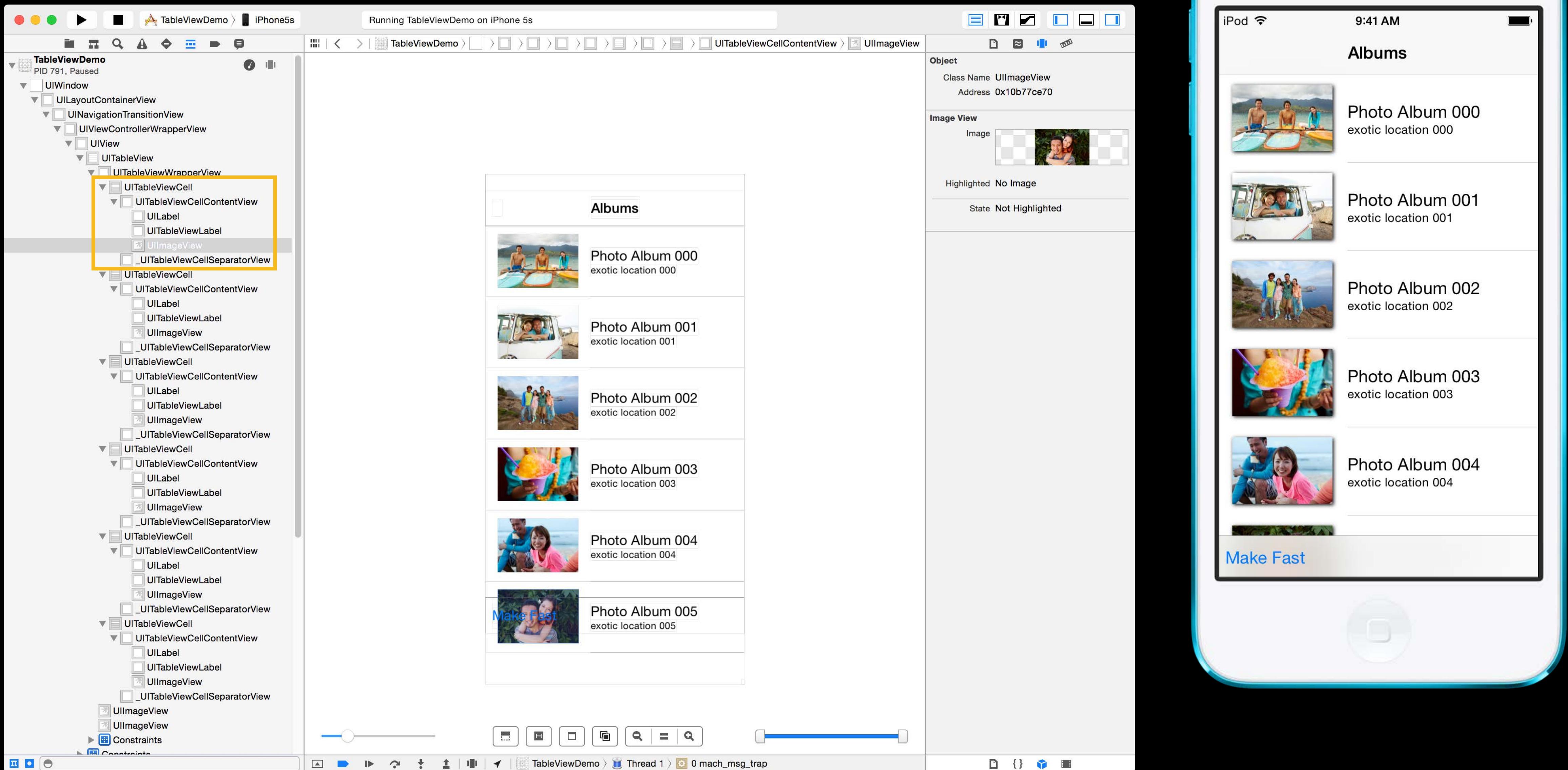
Analyzing View Hierarchy on iPod touch

Xcode view debugging



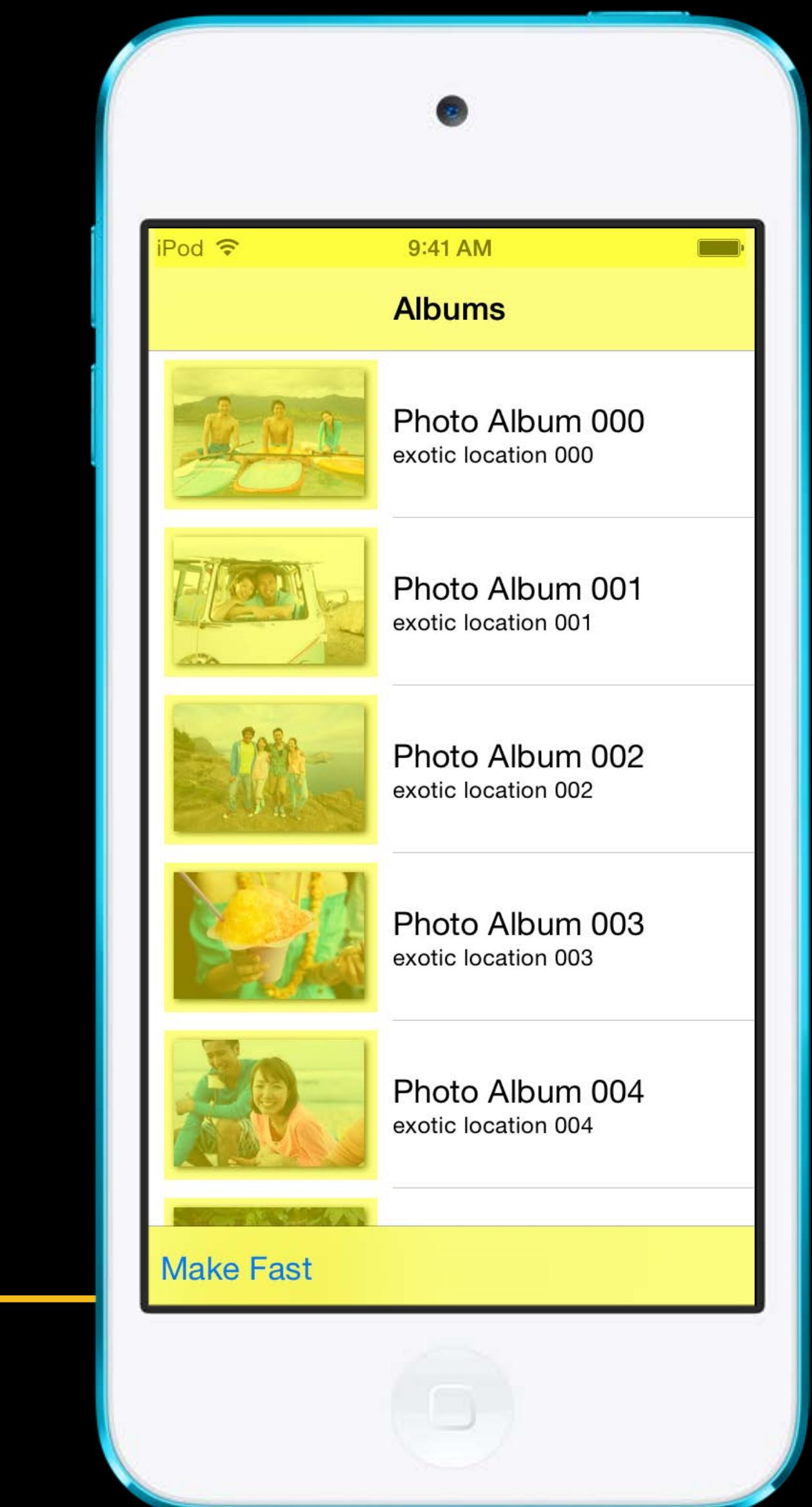
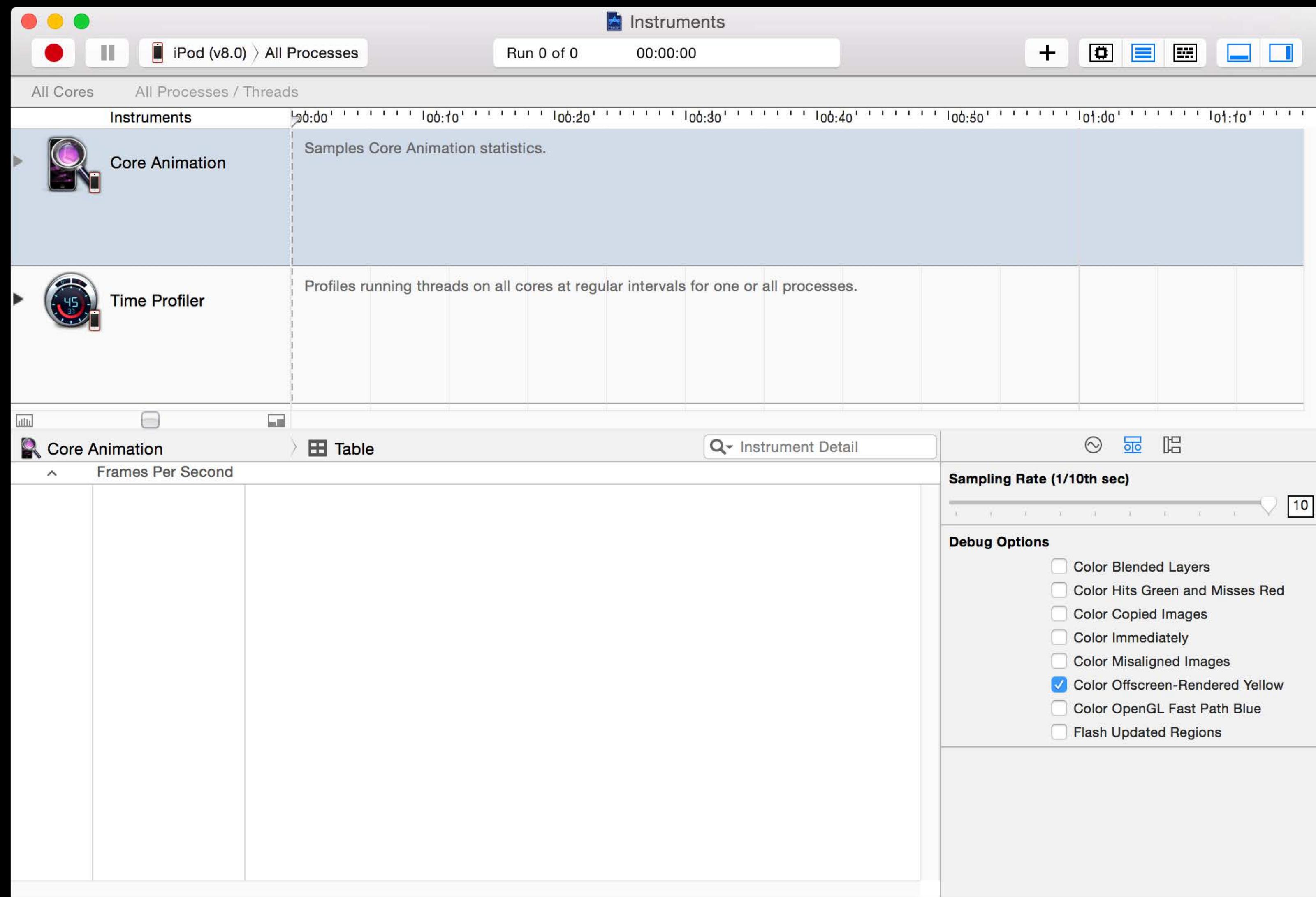
Analyzing View Hierarchy on iPod touch

Xcode view debugging



Color Offscreen-Rendered Yellow

Core Animation instrument



How Are We Setting up the Shadow?

How Are We Setting up the Shadow?

We are asking Core Animation to generate the shadow

```
CALayer *imageViewLayer = cell.imageView.layer;  
imageViewLayer.shadowColor = [UIColor blackColor].CGColor;  
imageViewLayer.shadowOpacity = 1.0;  
imageViewLayer.shadowRadius = 2.0;  
imageViewLayer.shadowOffset = CGSizeMake(1.0, 1.0);
```

How Are We Setting up the Shadow?

We are asking Core Animation to generate the shadow

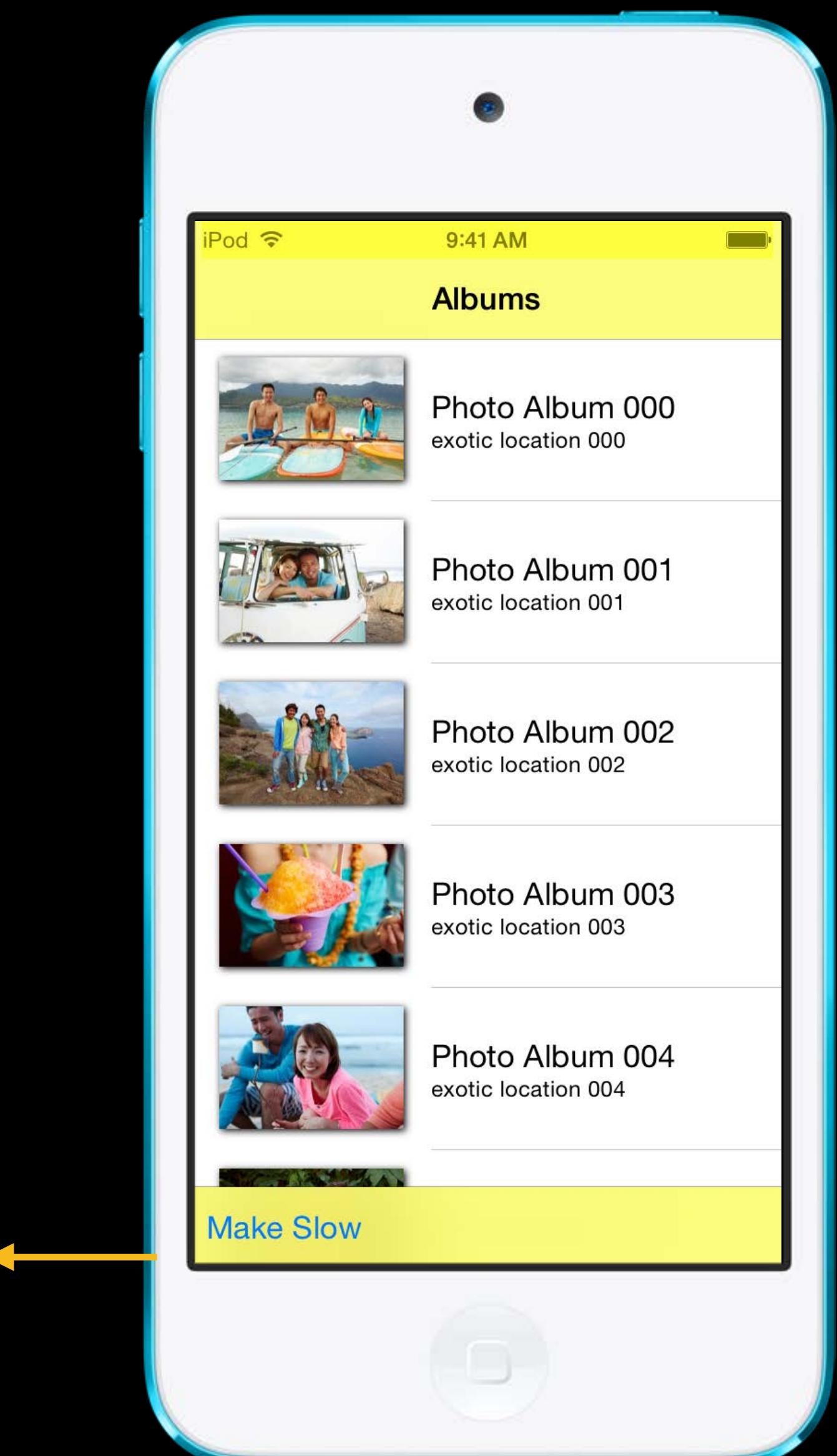
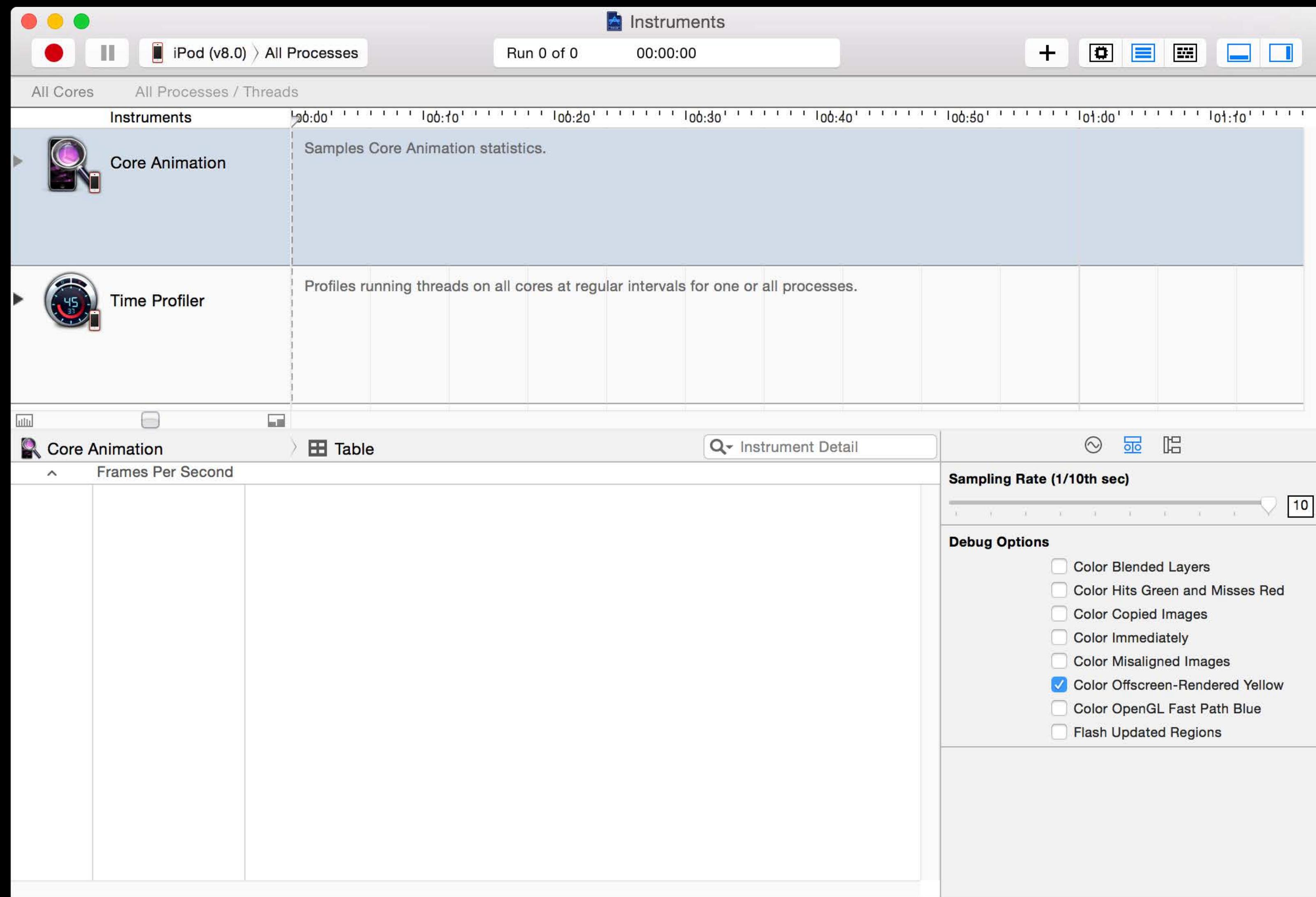
```
CALayer *imageViewLayer = cell.imageView.layer;  
imageViewLayer.shadowColor = [UIColor blackColor].CGColor;  
imageViewLayer.shadowOpacity = 1.0;  
imageViewLayer.shadowRadius = 2.0;  
imageViewLayer.shadowOffset = CGSizeMake(1.0, 1.0);
```

Perhaps there is a more efficient way

```
imageViewLayer.shadowPath = CGPathCreateWithRect(imageRect, NULL);
```

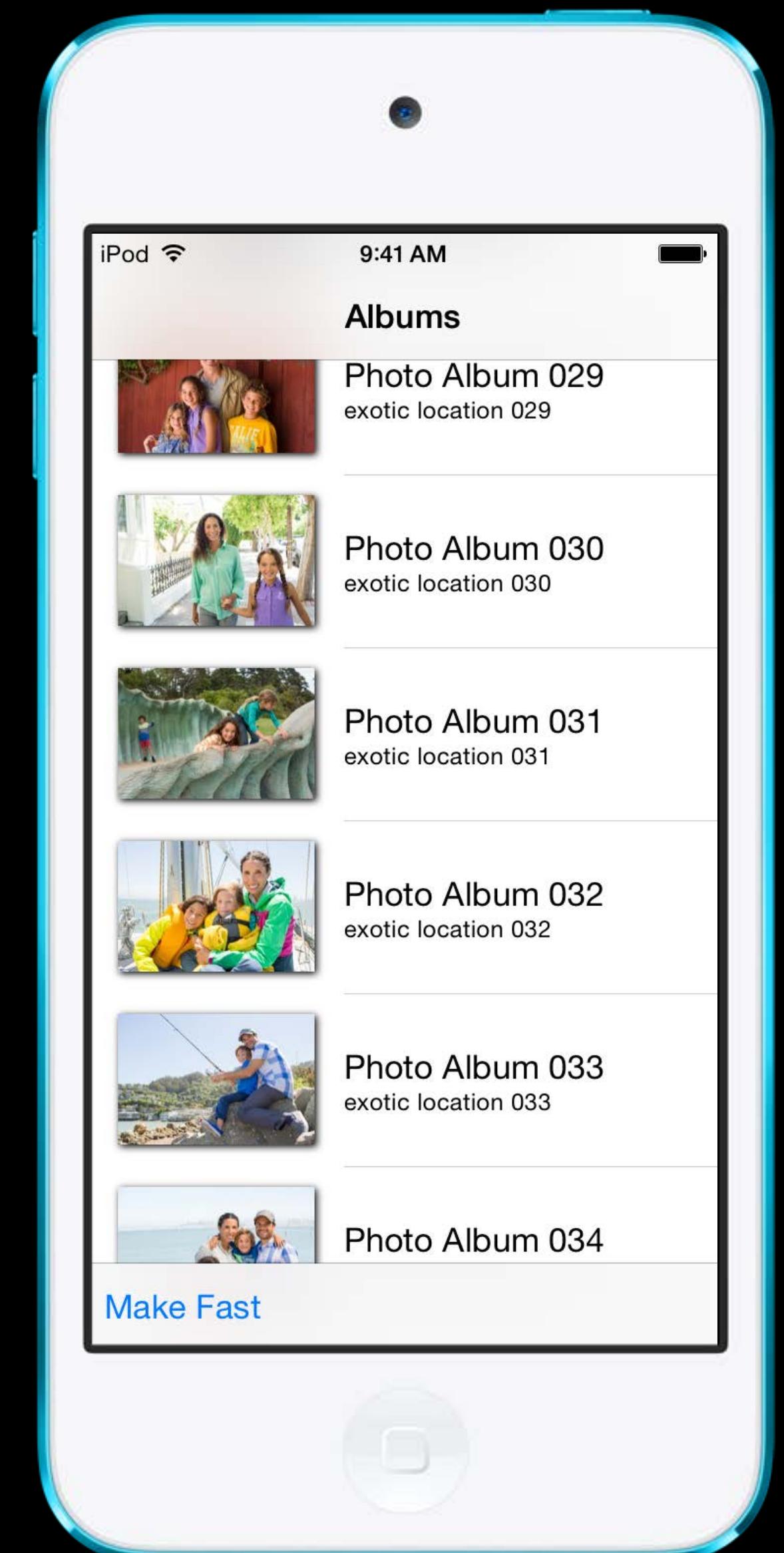
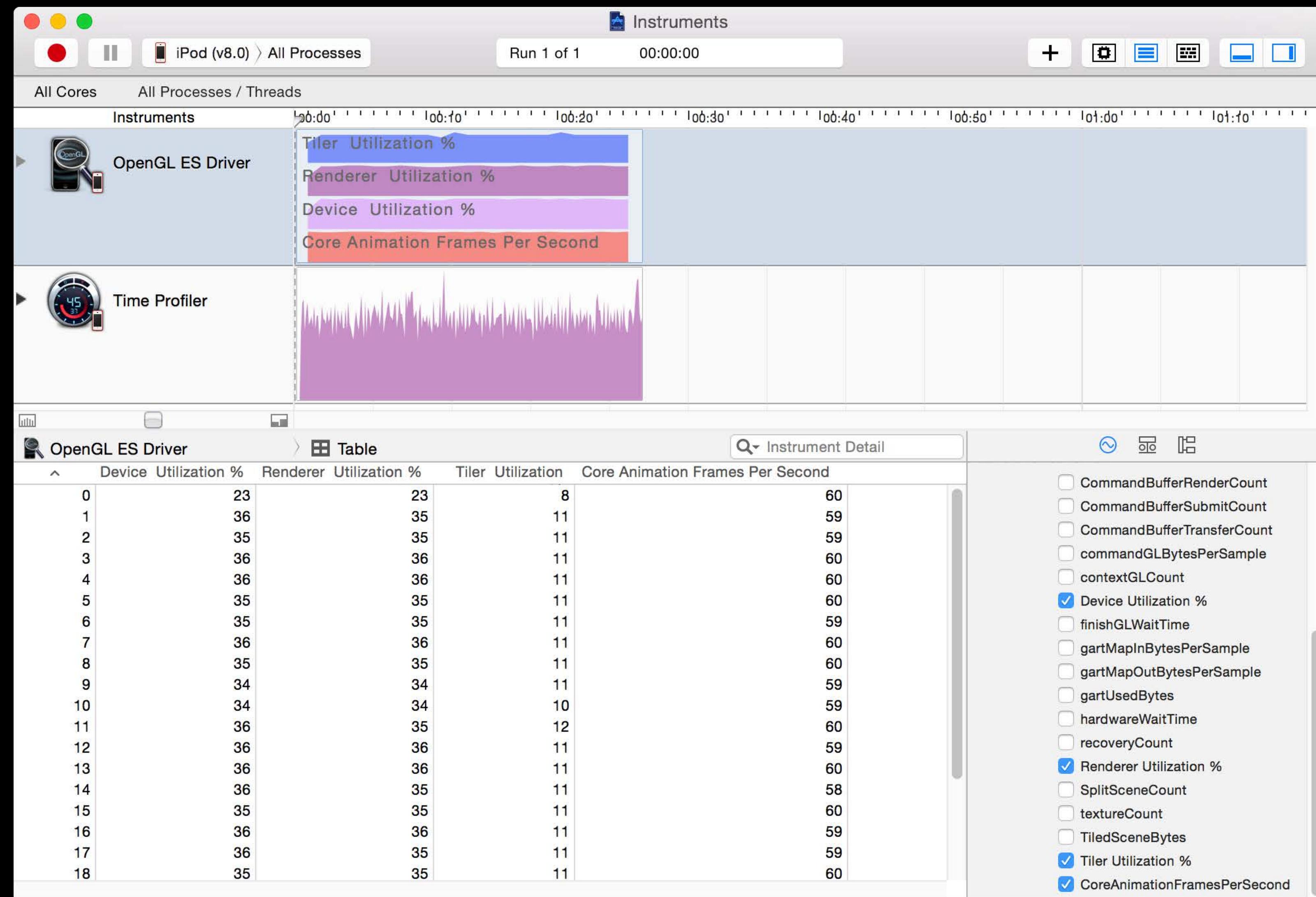
Color Offscreen-Rendered Yellow

Core Animation instrument



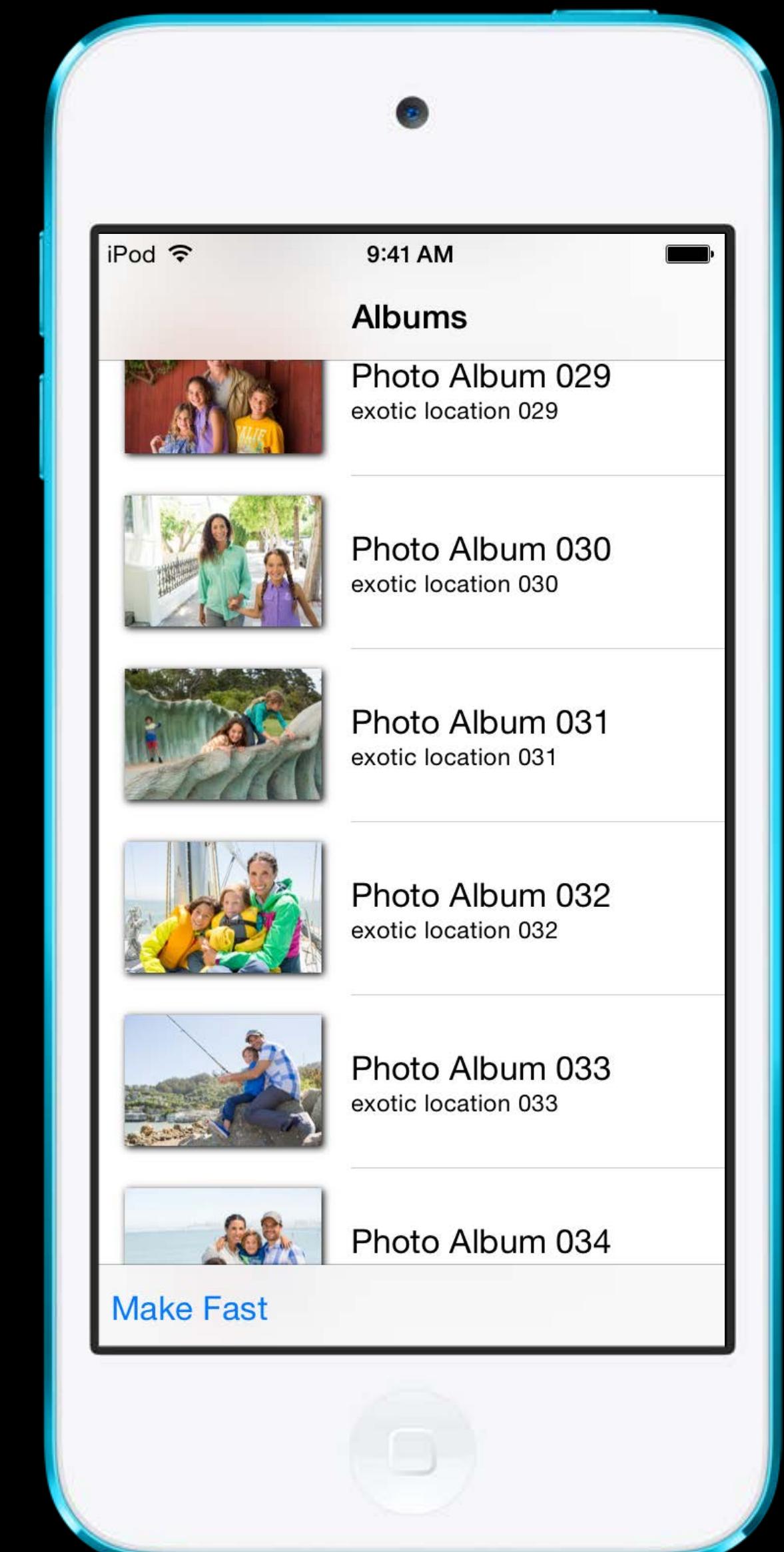
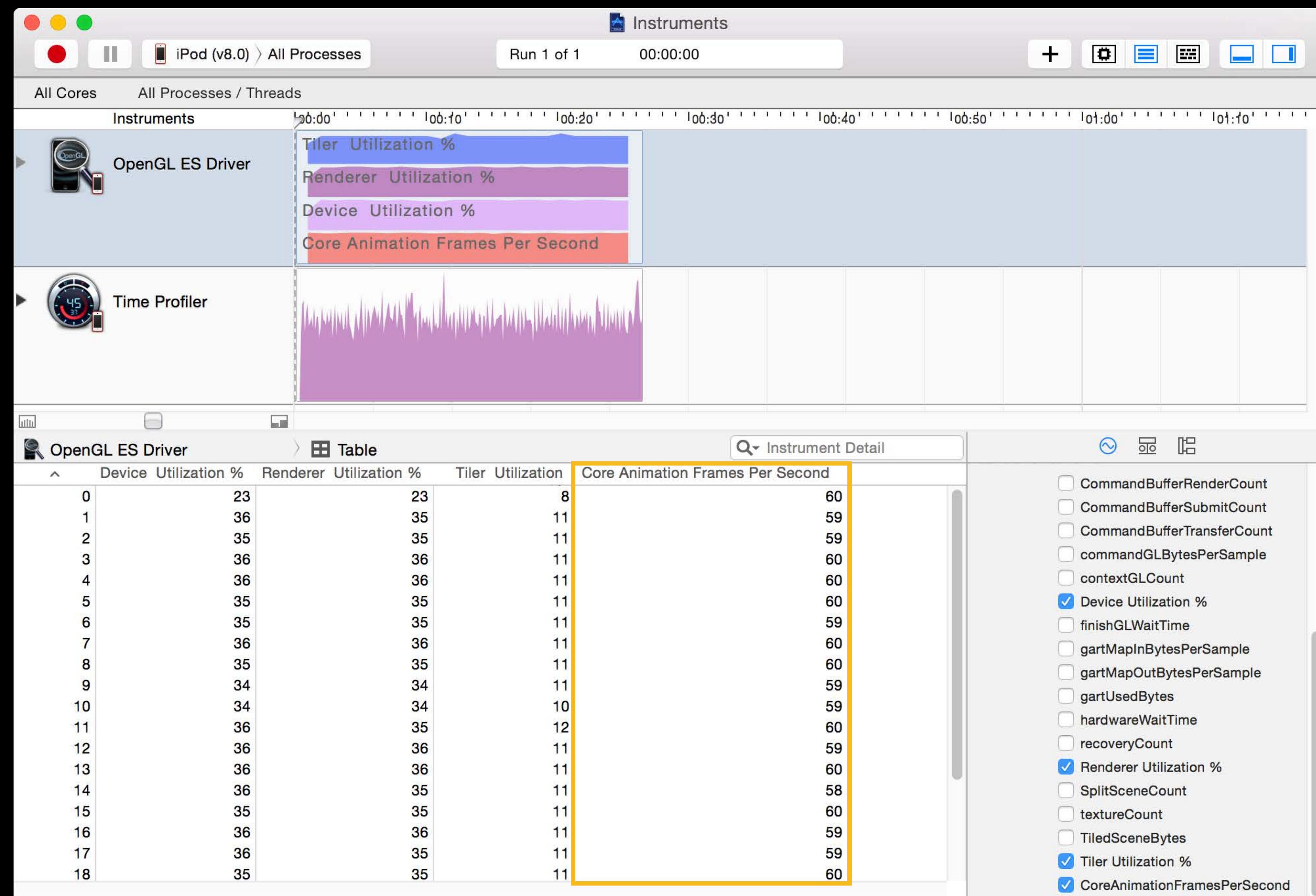
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



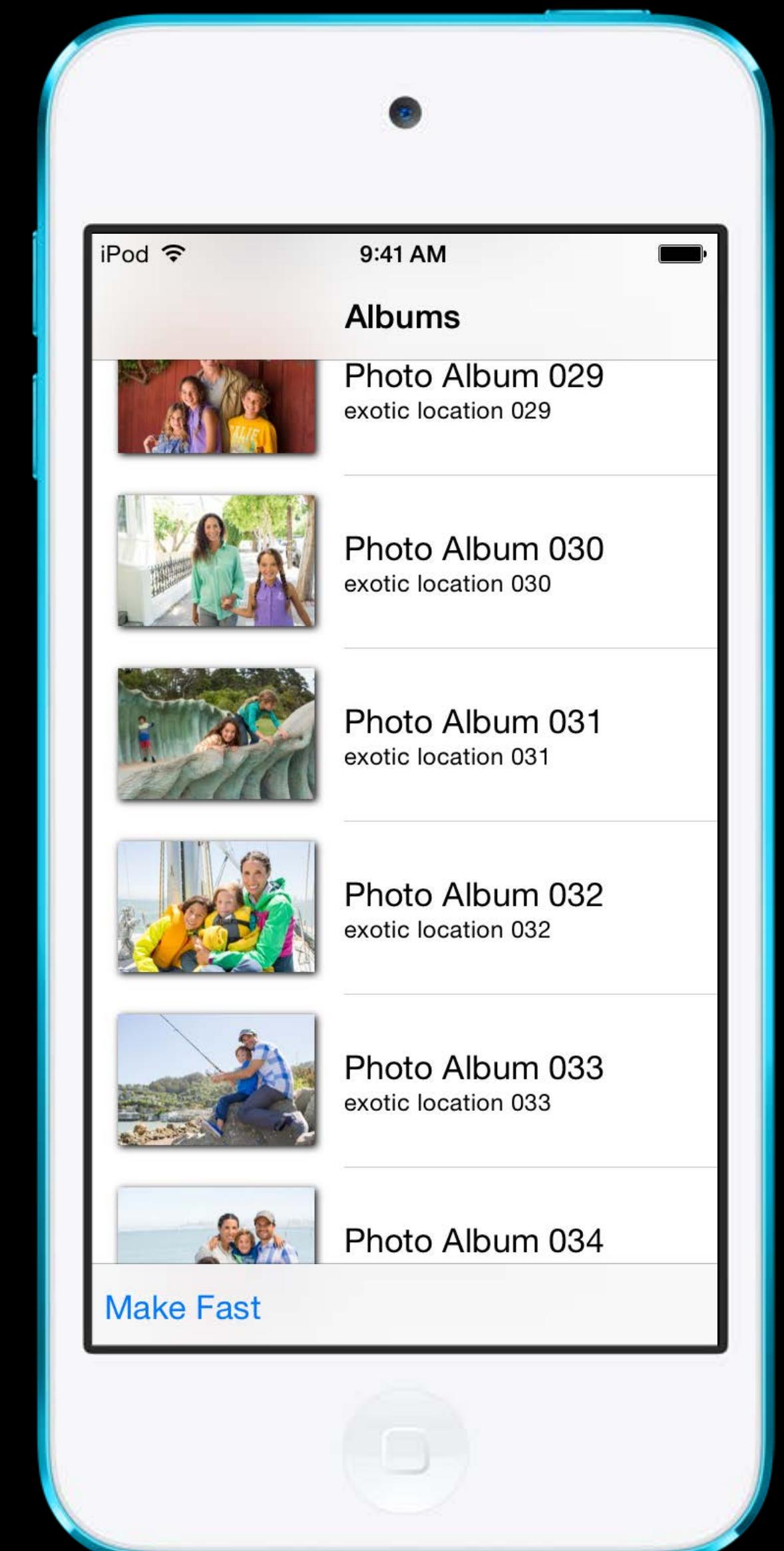
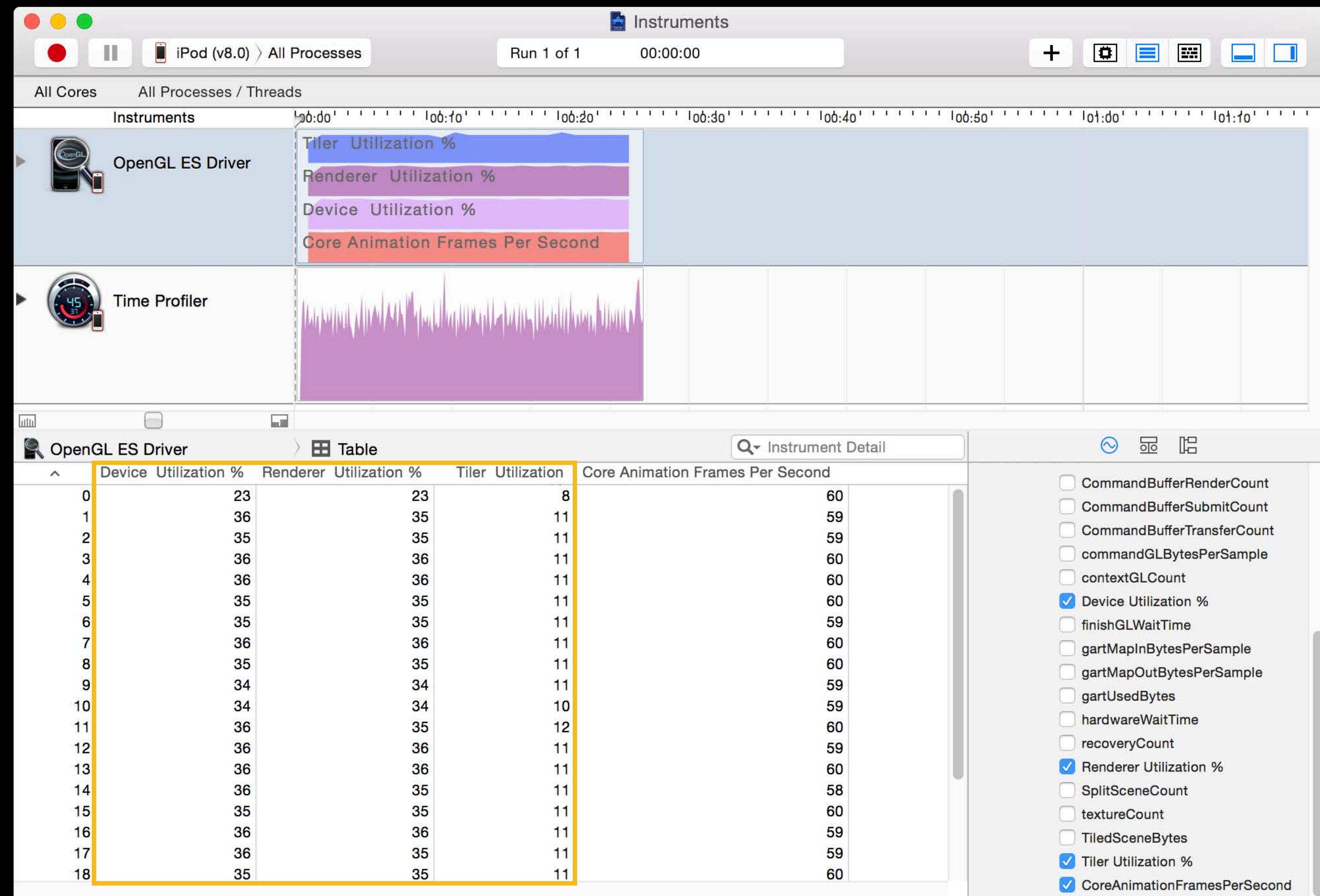
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



Measure Frame Rate on iPod touch

OpenGL ES Driver instrument

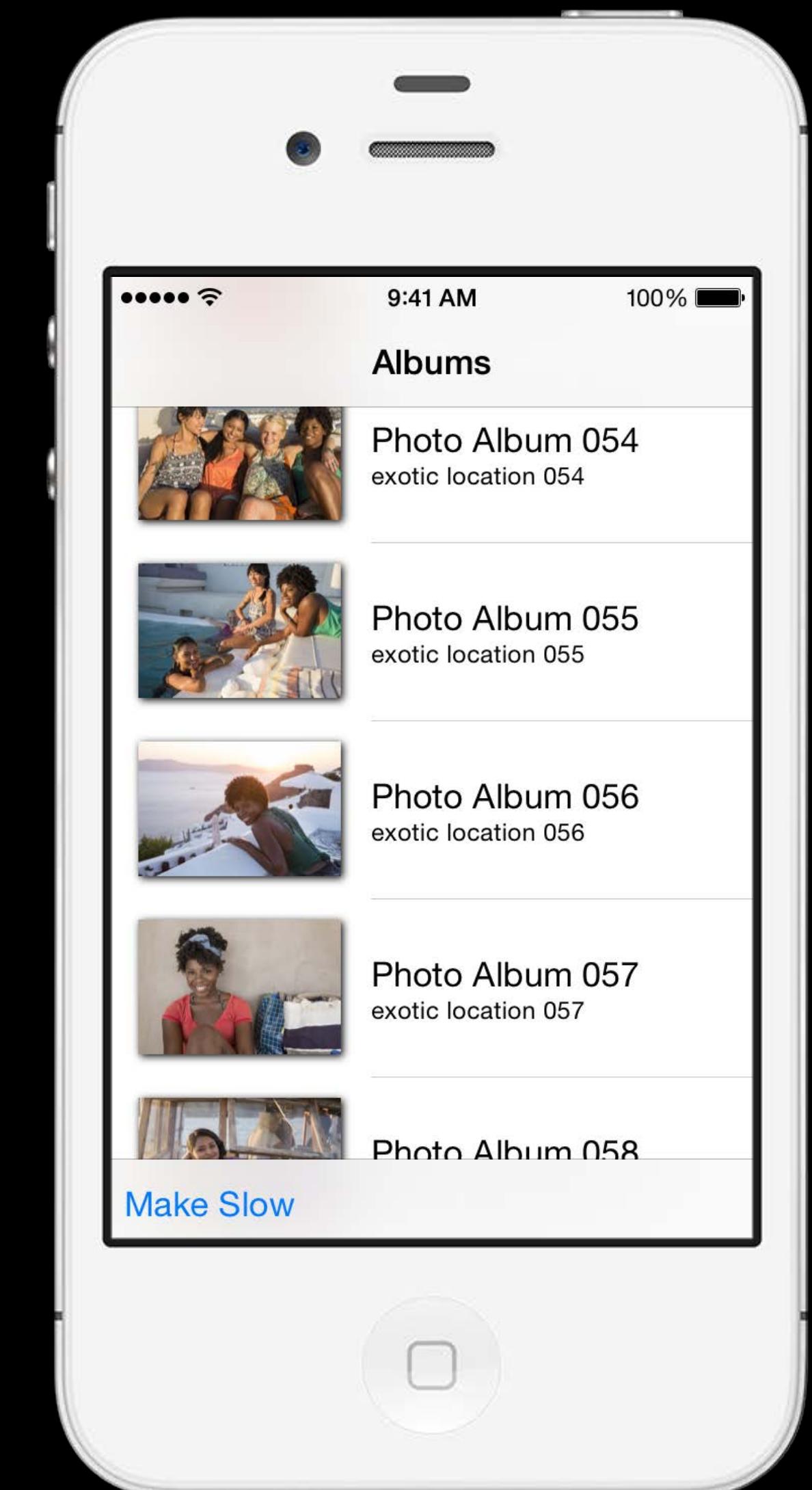
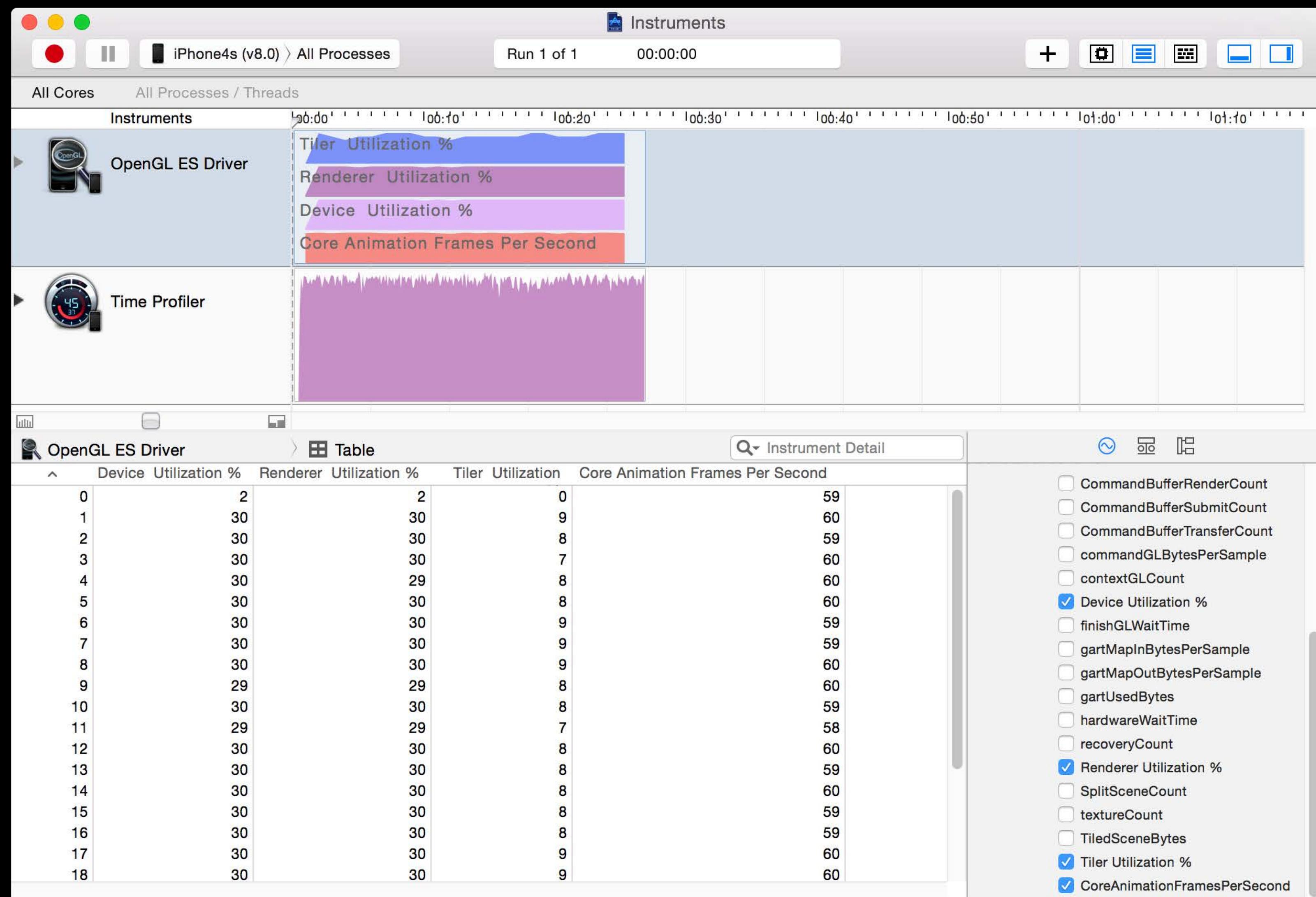


Awesome

Can we ship it now?

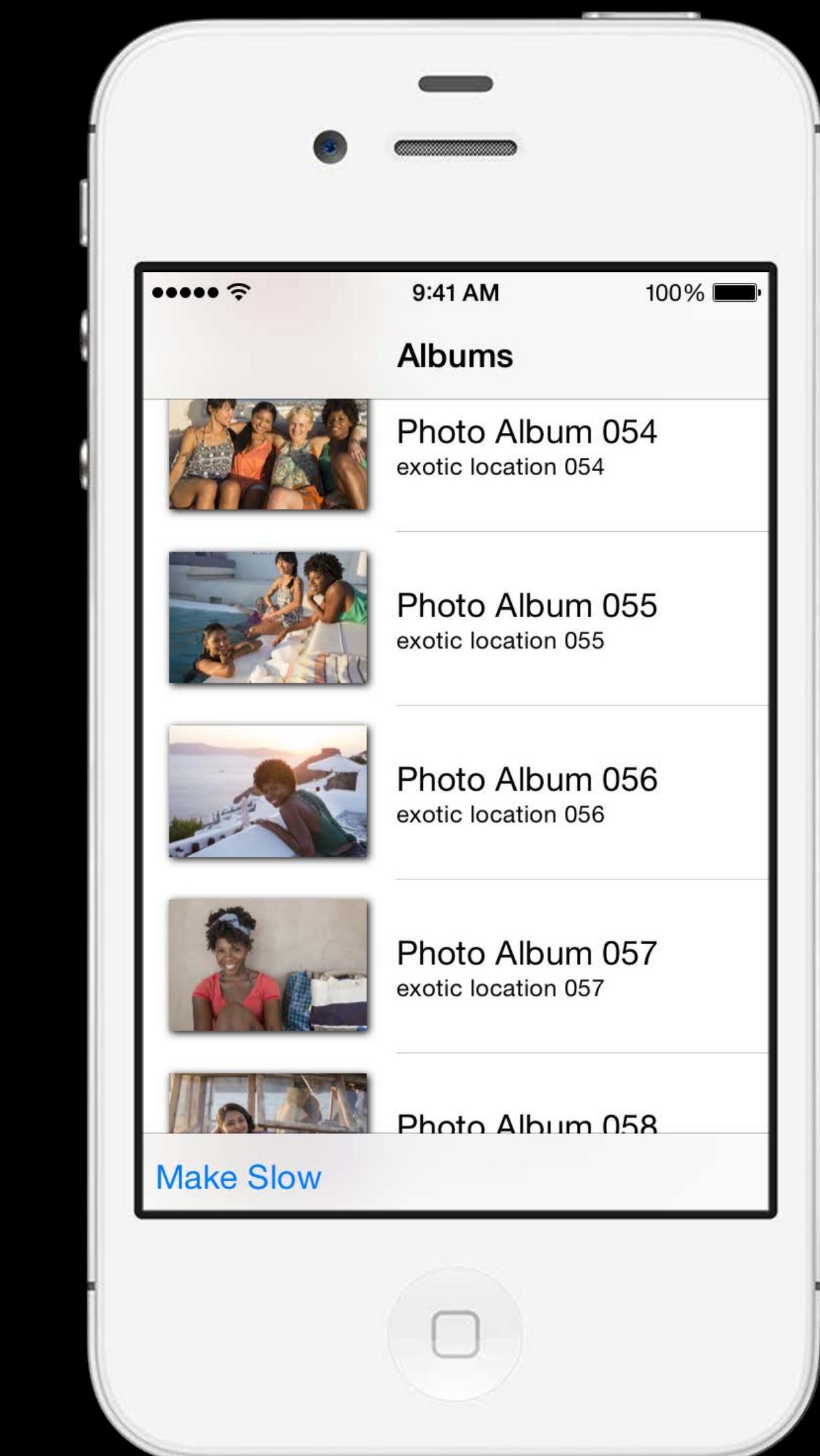
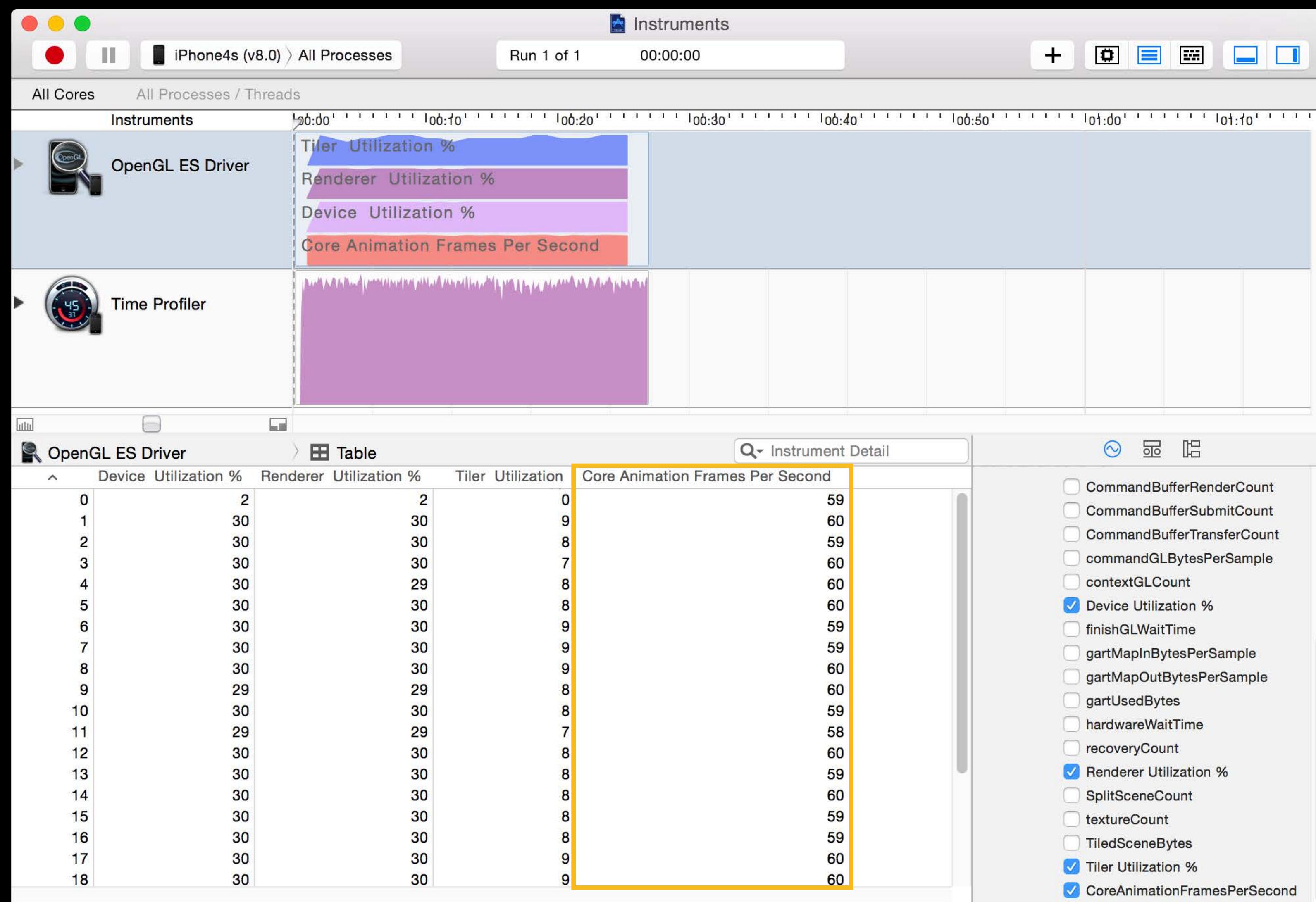
Measure Frame Rate on iPhone 4s

OpenGL ES Driver instrument



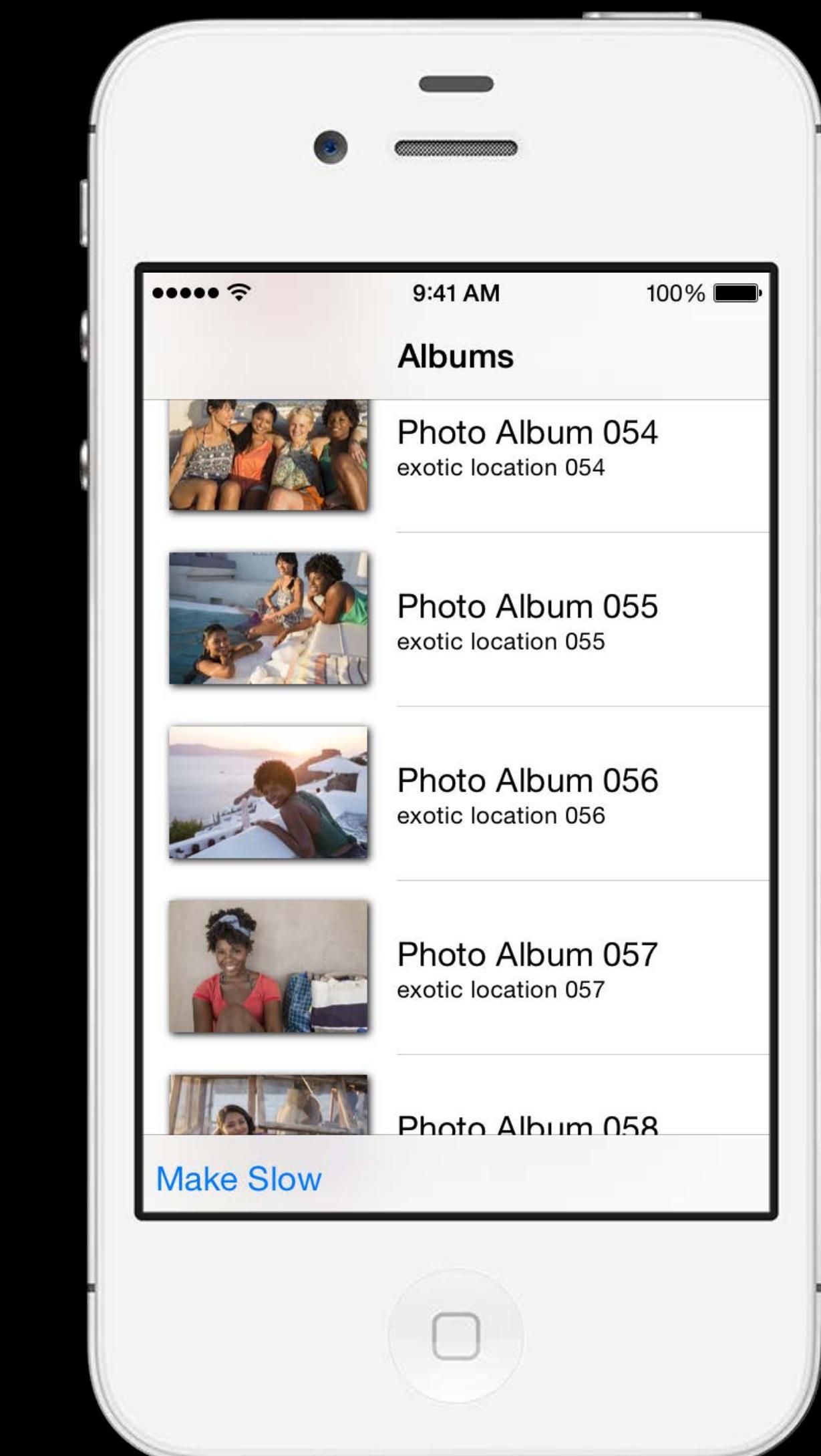
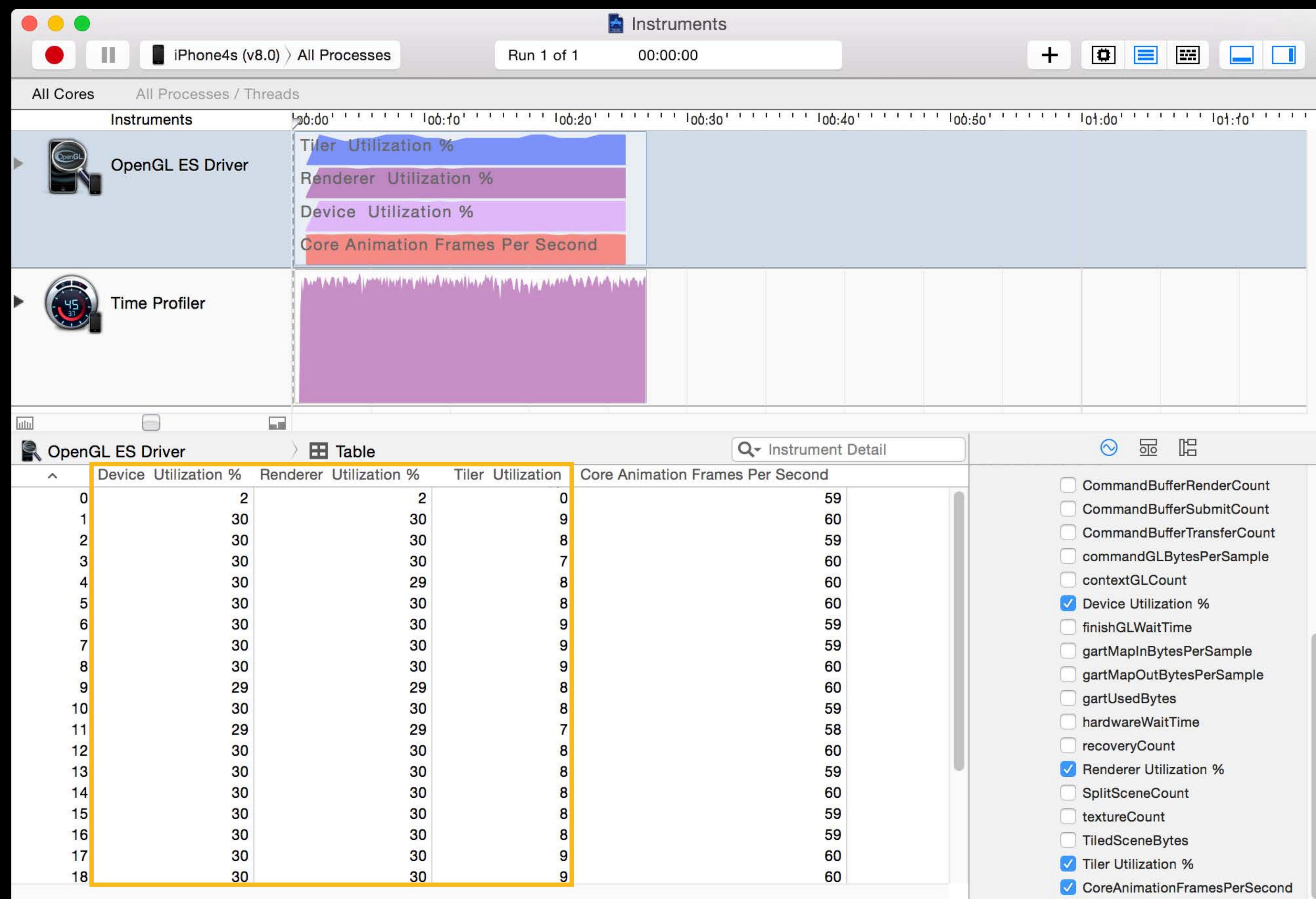
Measure Frame Rate on iPhone 4s

OpenGL ES Driver instrument



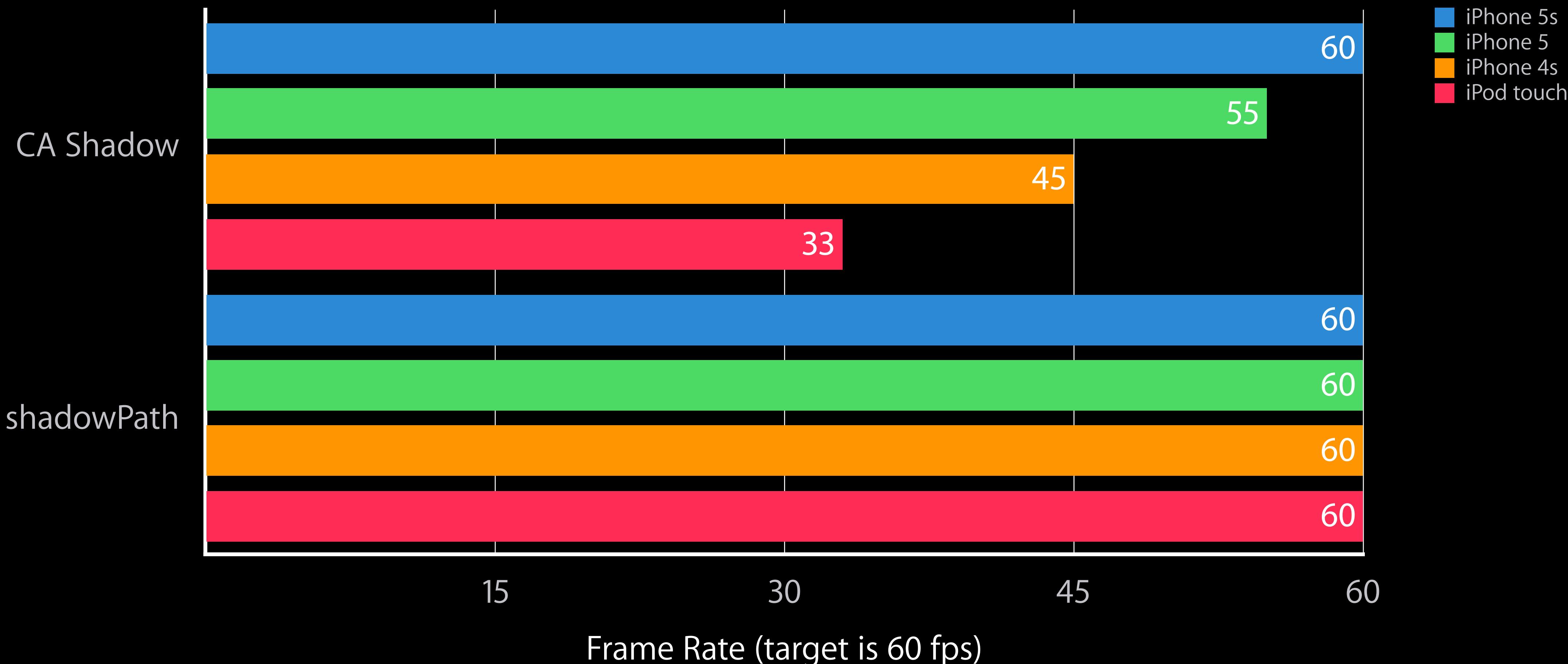
Measure Frame Rate on iPhone 4s

OpenGL ES Driver instrument



Fictitious Photo Application

Performance across devices



Awesome
Ship it!

Fictitious Photo Application

Summary

Offscreen passes are expensive

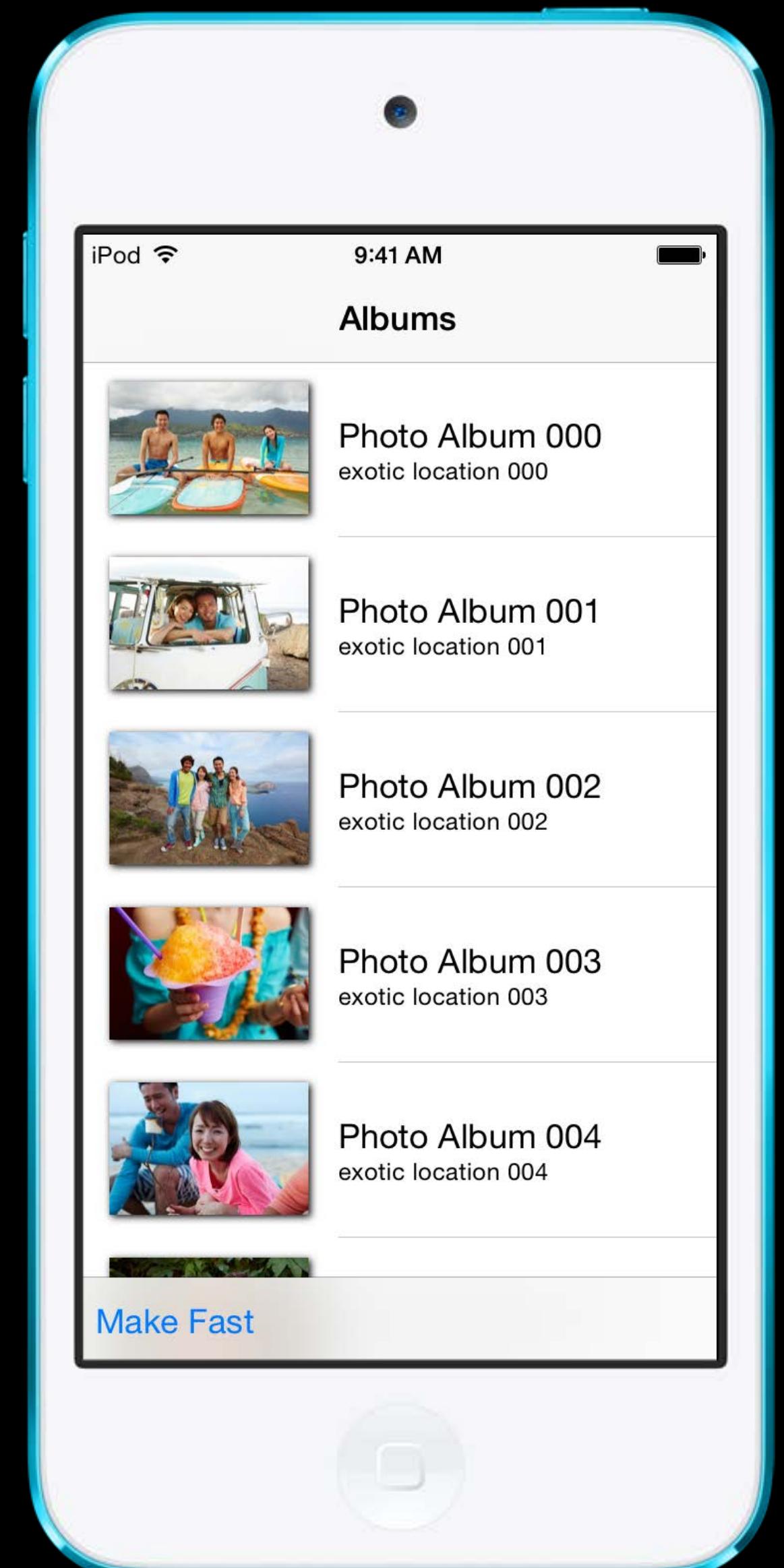
- Use Core Animation instrument to find them
- Know what you can do to avoid them

Measure performance across different devices

- Use OpenGL ES Driver instrument for GPU time
- Use Time Profiler instrument for CPU time

Know your view hierarchy and any hidden costs

- This is especially true for table cells and scrolling

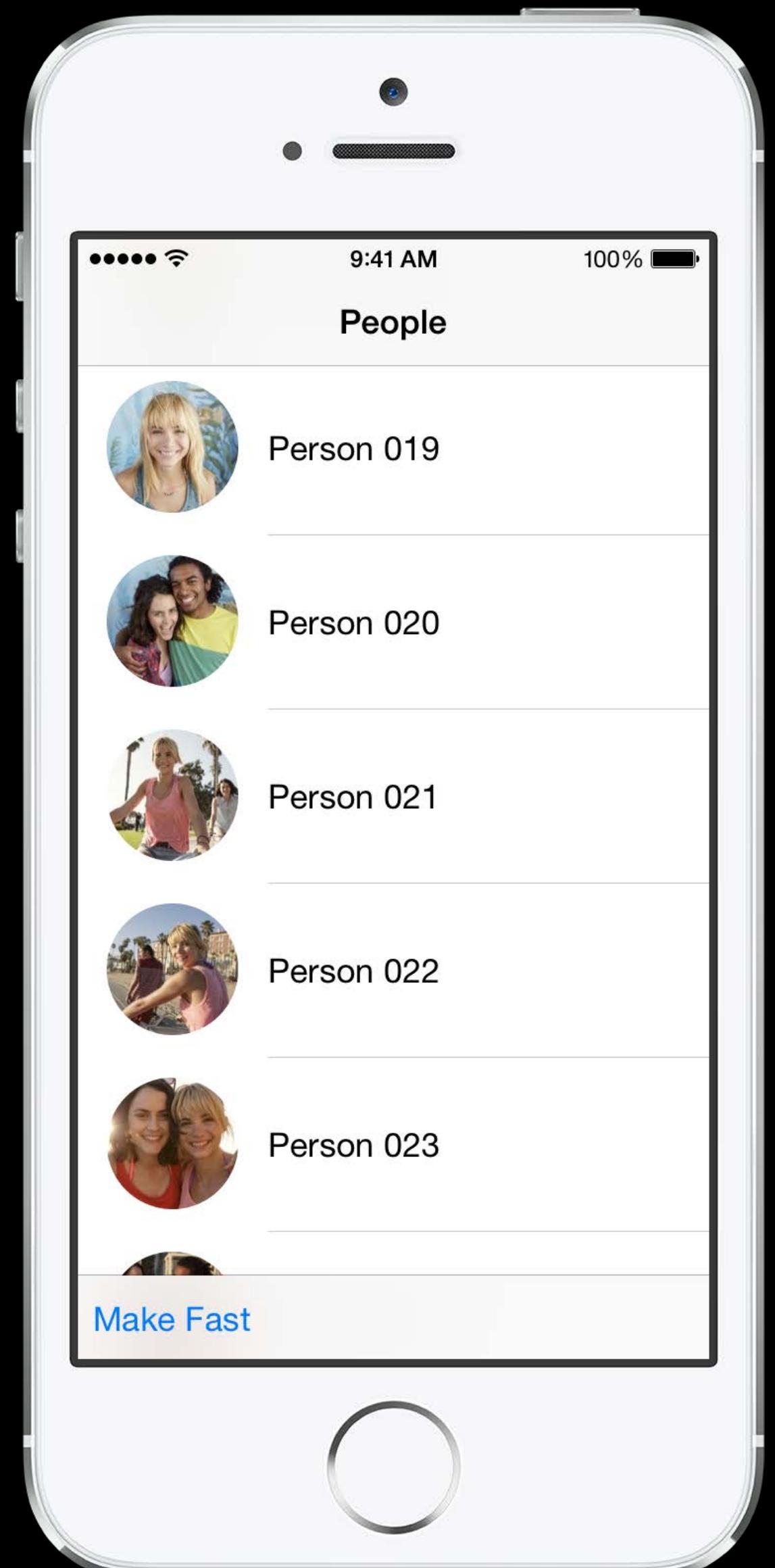


Fictitious Contacts Application

Case study

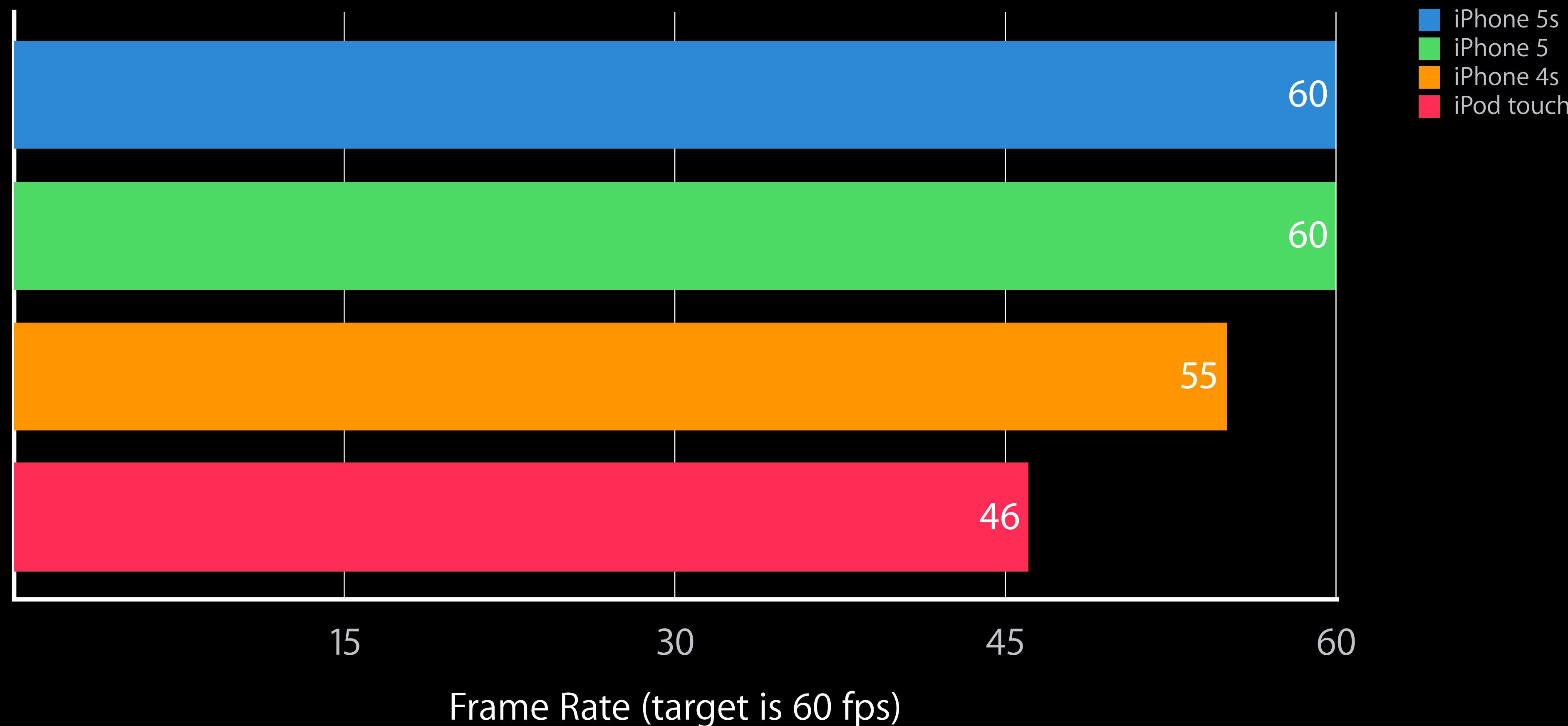
Simple table view

Each cell shows a round thumbnail and some text



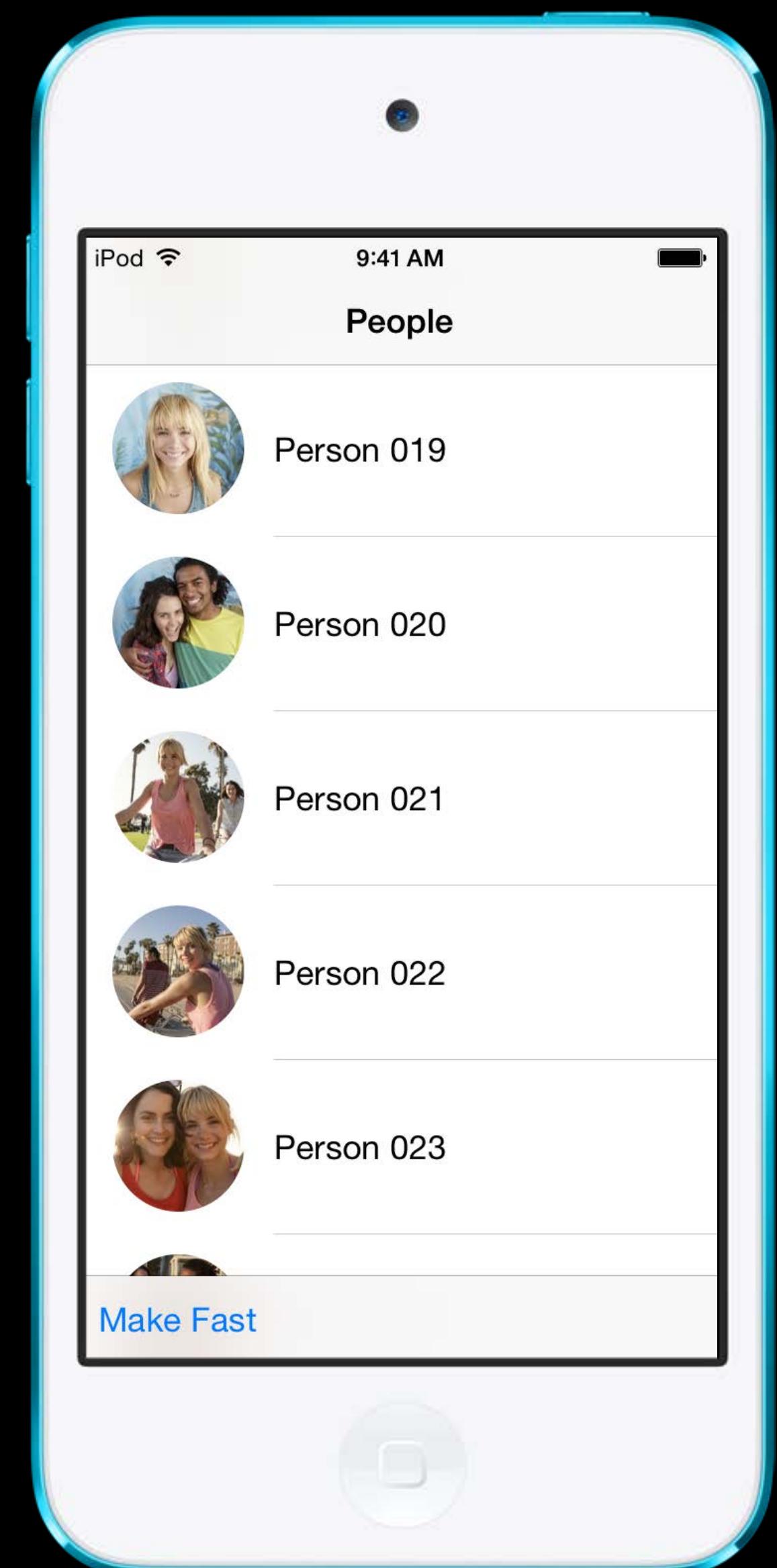
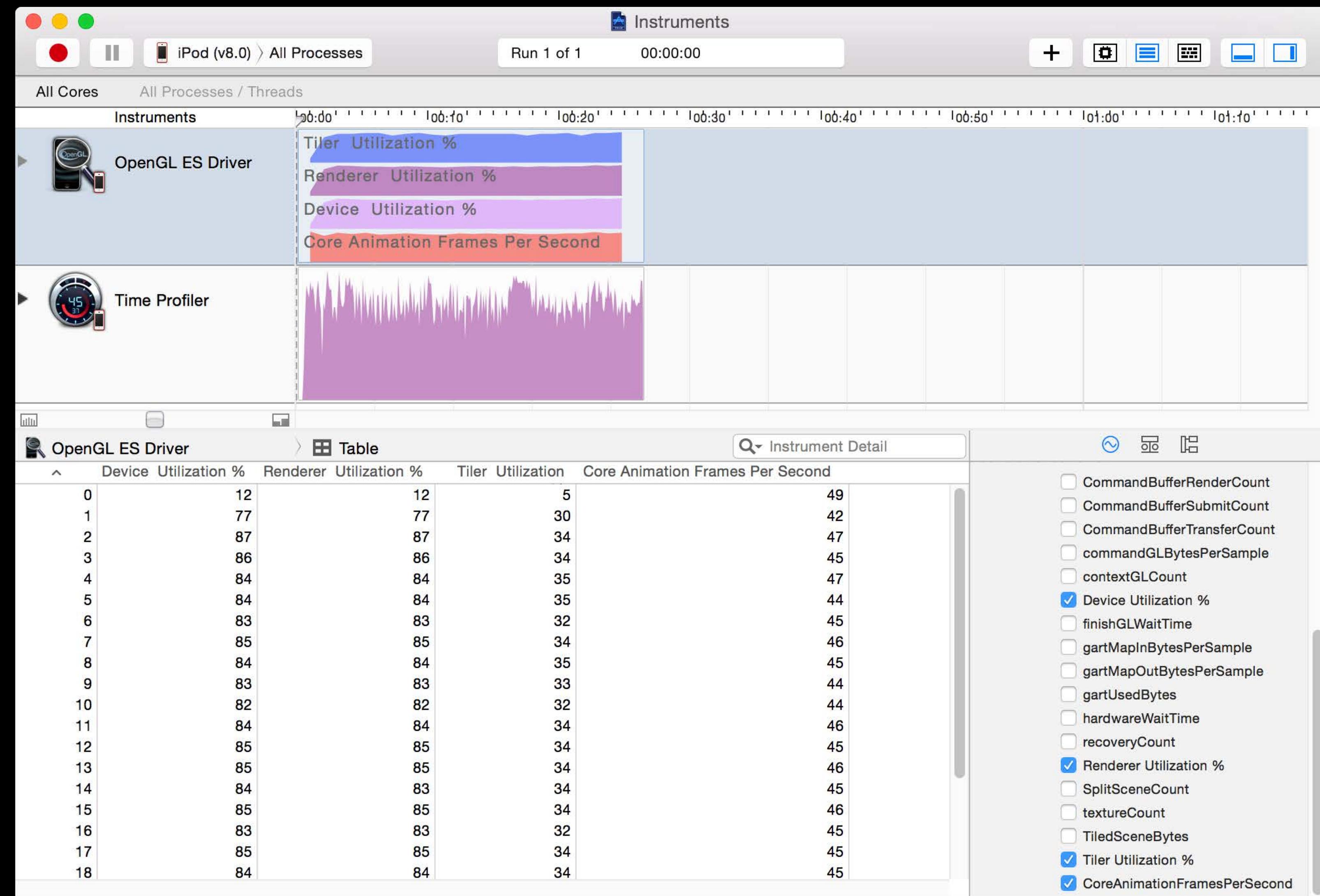
Fictitious Contacts Application

Performance across devices



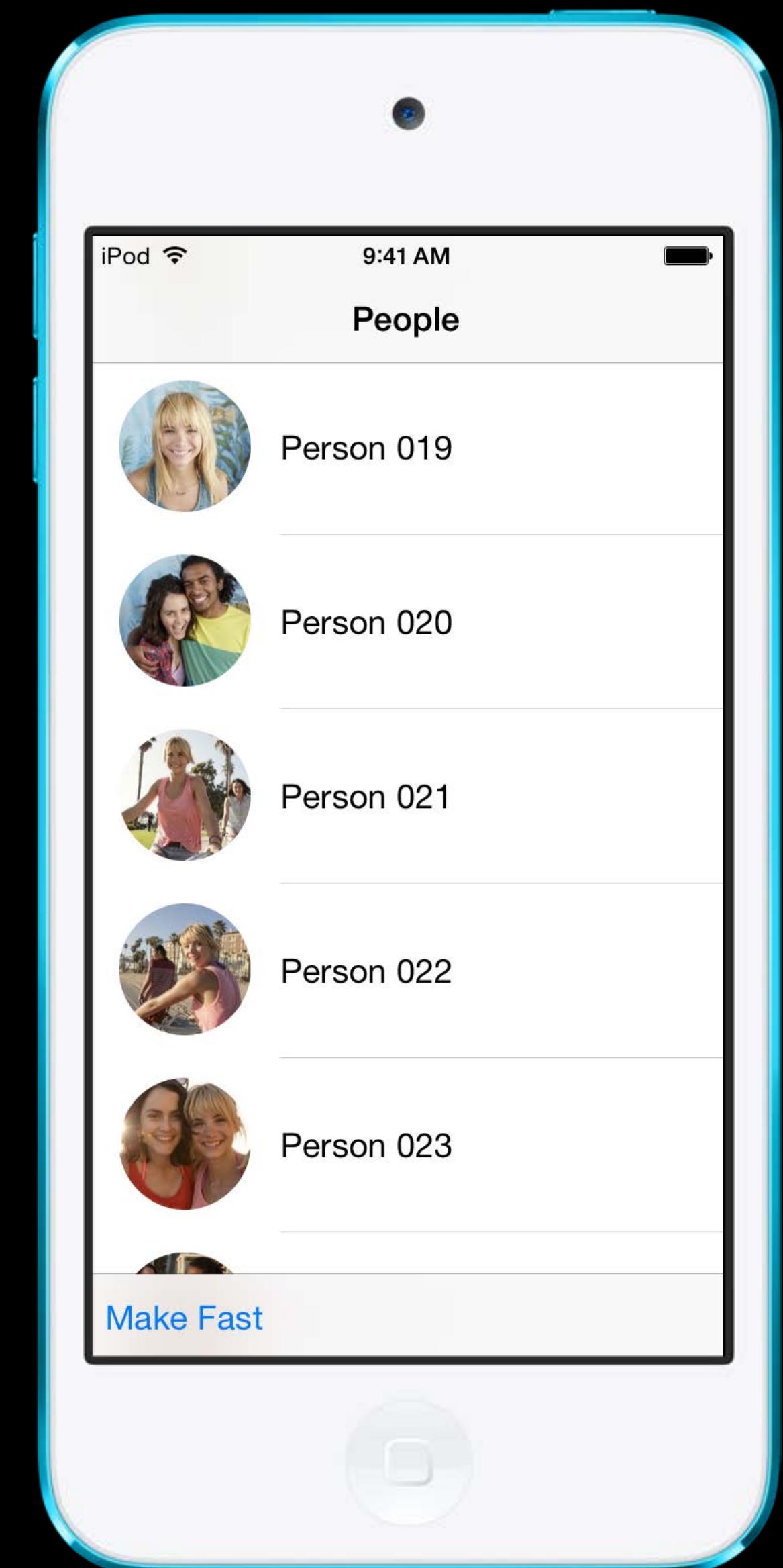
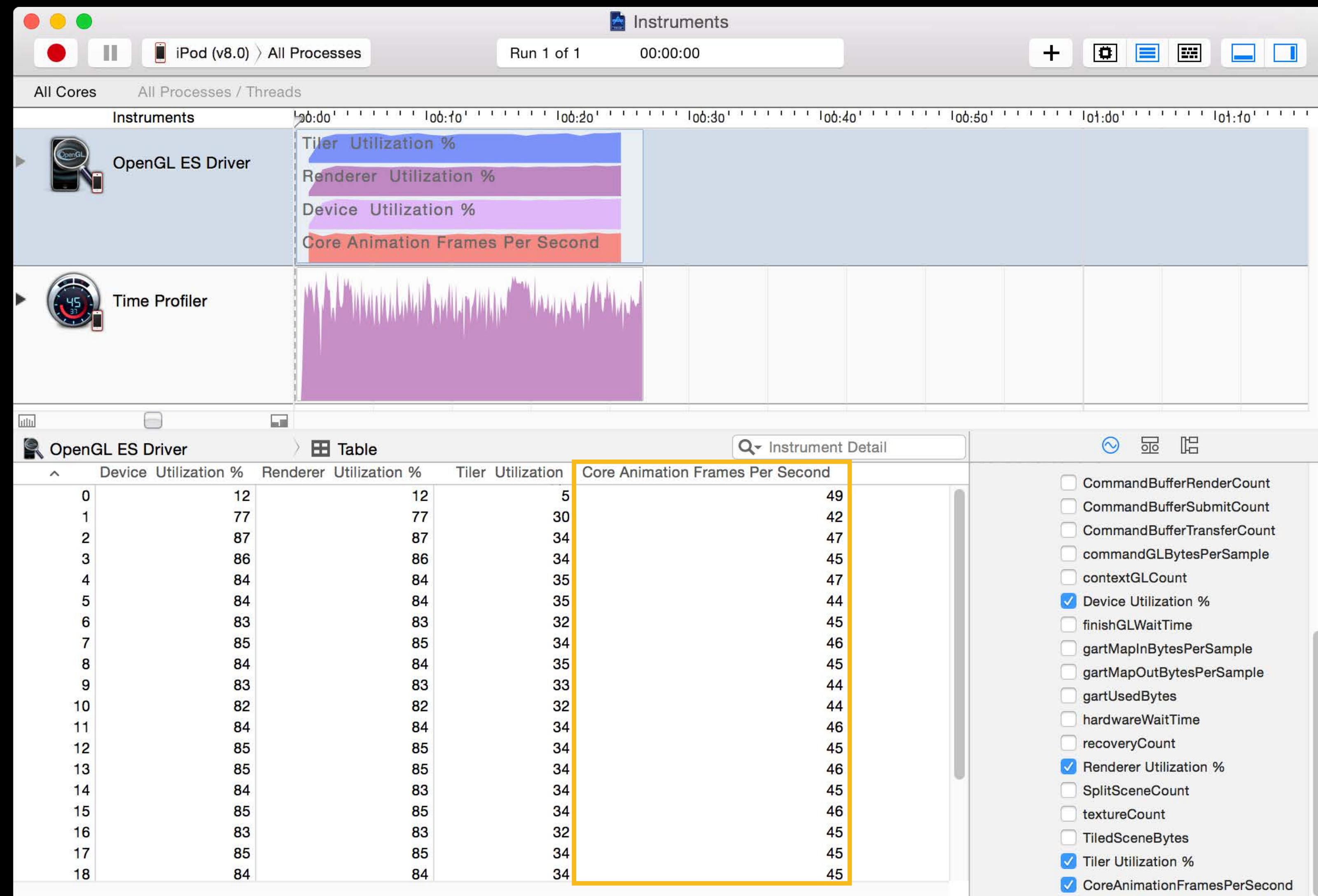
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



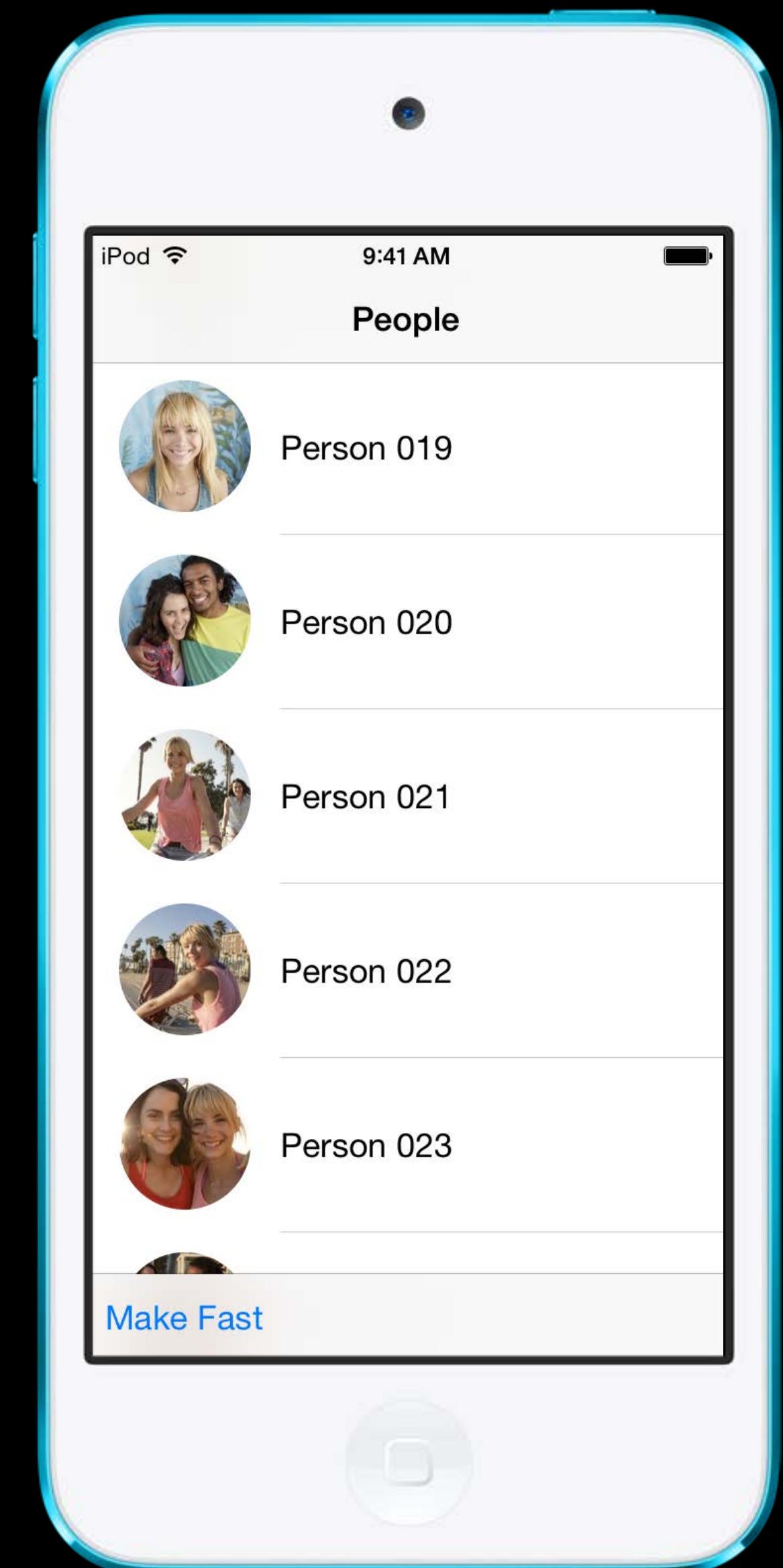
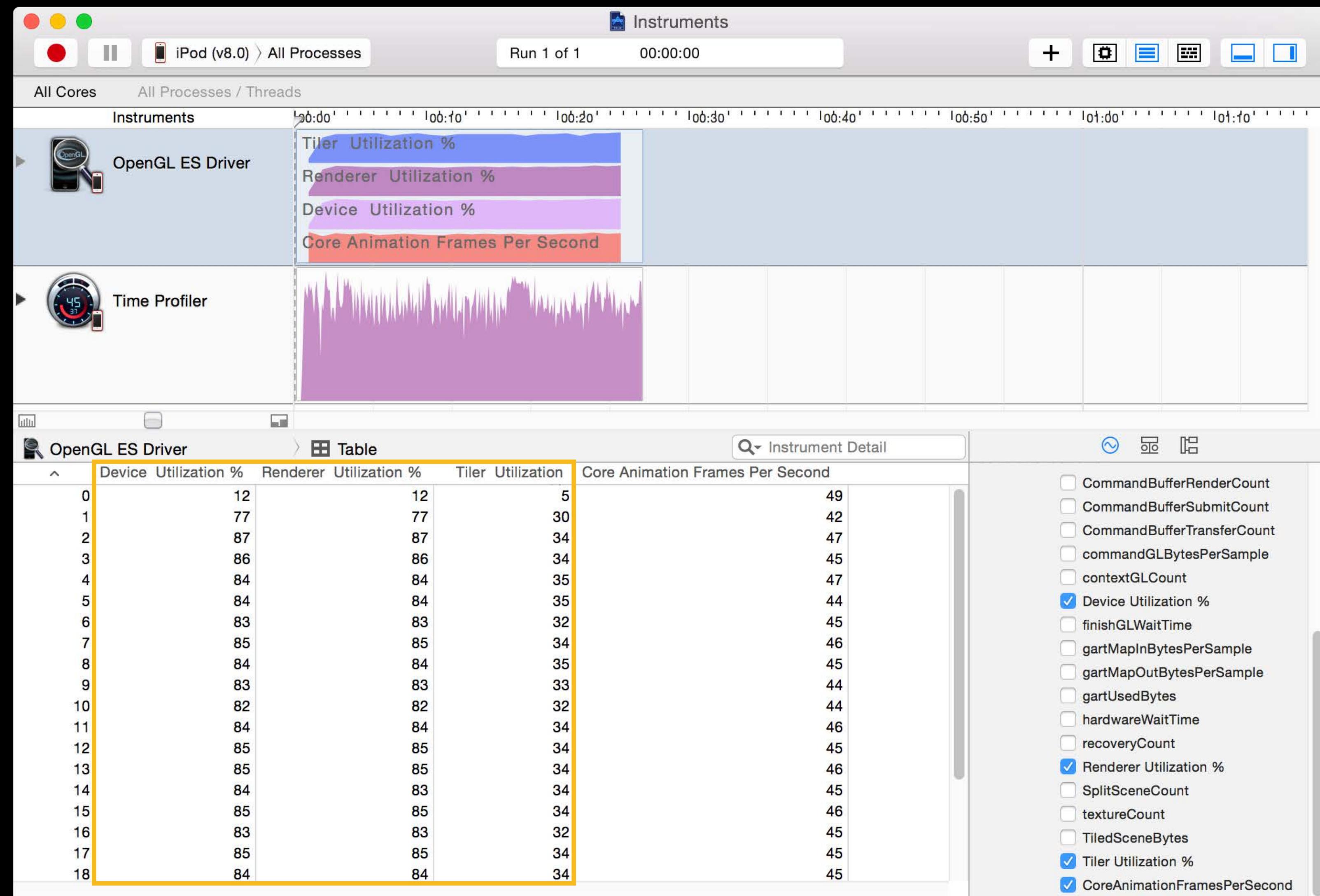
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



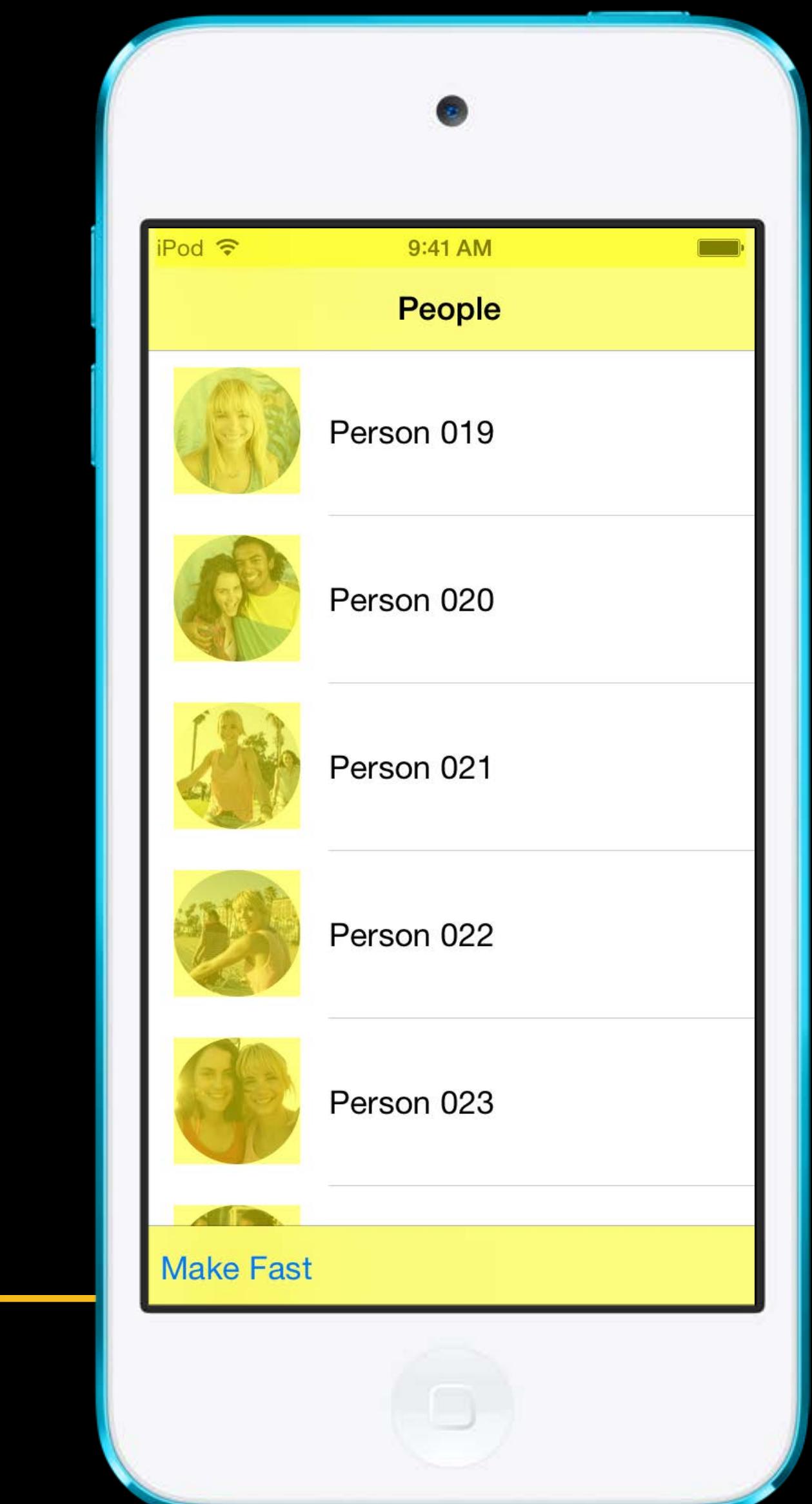
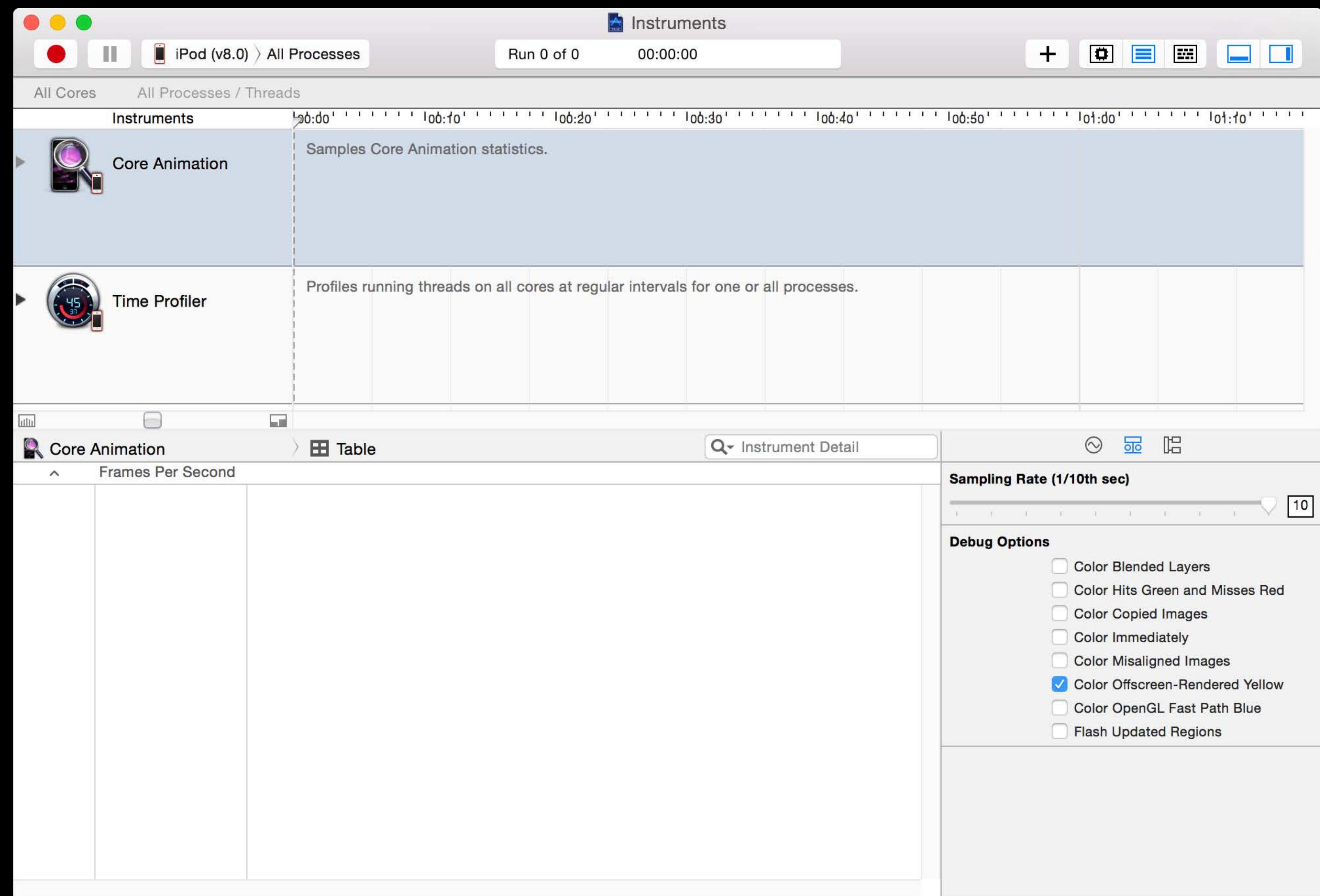
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



Color Offscreen-Rendered Yellow

Core Animation instrument



How Are We Achieving Round Thumbnails?

How Are We Achieving Round Thumbnails?

We are asking Core Animation to mask the image

```
CALayer *imageViewLayer = cell.imageView.layer;  
imageViewLayer.cornerRadius = imageHeight / 2.0;  
imageViewLayer.masksToBounds = YES;
```

How Are We Achieving Round Thumbnails?

We are asking Core Animation to mask the image

```
CALayer *imageViewLayer = cell.imageView.layer;  
imageViewLayer.cornerRadius = imageHeight / 2.0;  
imageViewLayer.masksToBounds = YES;
```

Perhaps there is a more efficient way

- Don't mask on the fly, pre-generate thumbnails as round, or

How Are We Achieving Round Thumbnails?

We are asking Core Animation to mask the image

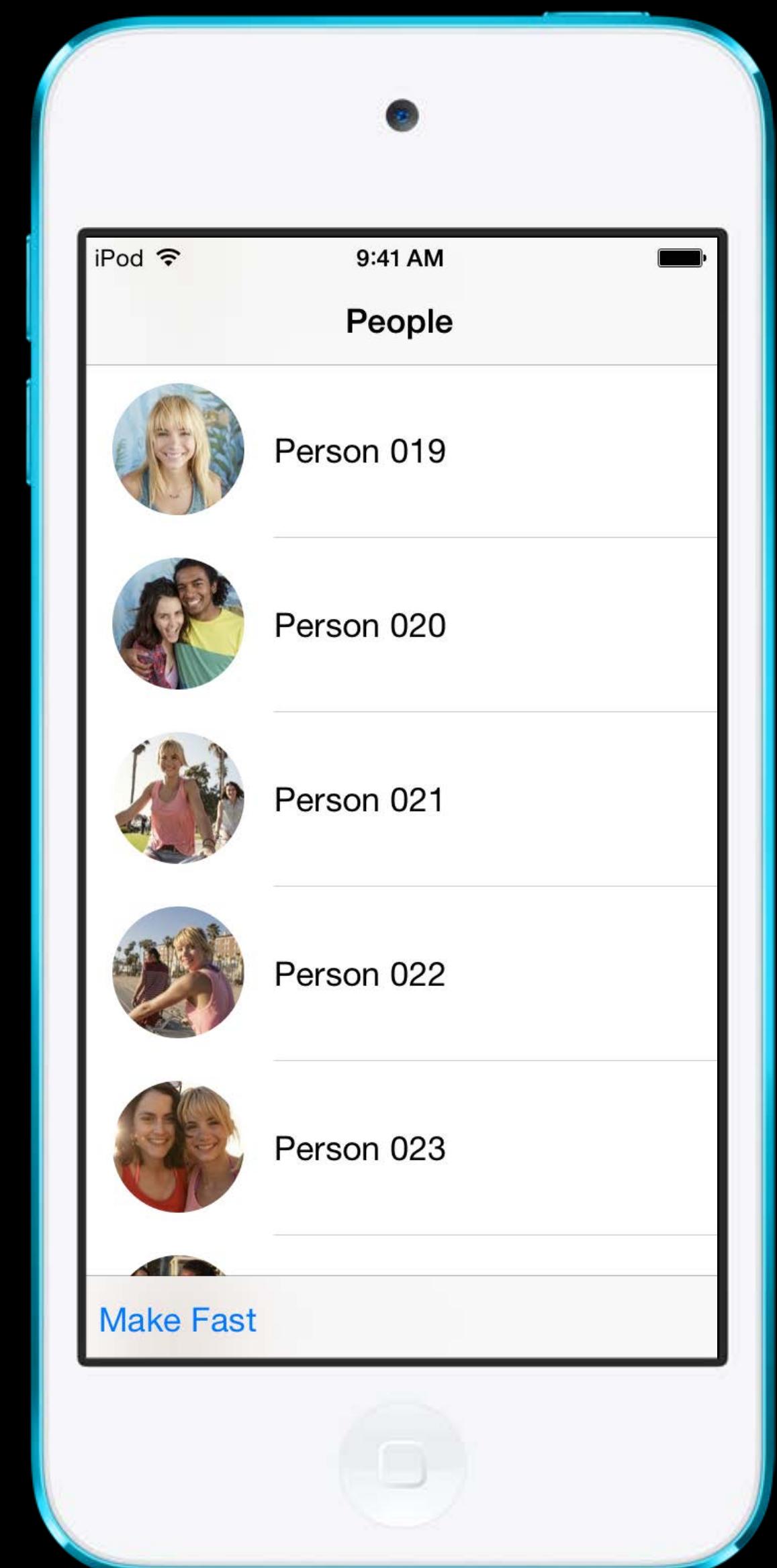
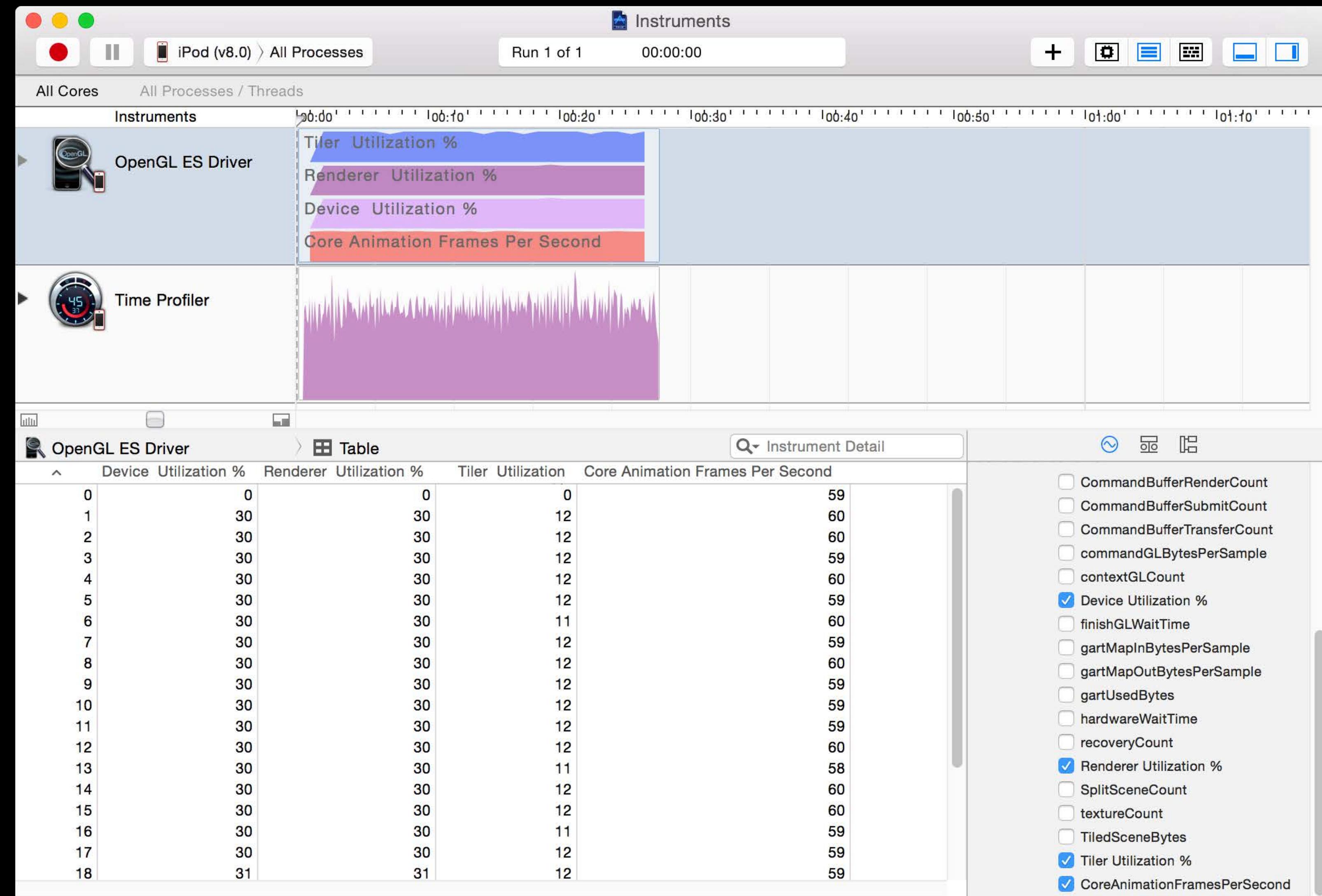
```
CALayer *imageViewLayer = cell.imageView.layer;  
imageViewLayer.cornerRadius = imageHeight / 2.0;  
imageViewLayer.masksToBounds = YES;
```

Perhaps there is a more efficient way

- Don't mask on the fly, pre-generate thumbnails as round, or
- If that is not possible, fake it
 - Table background is solid white
 - Render a white inverted circle on top of square thumbnail asset
 - Reducing offscreen passes but increasing blending, still a net performance win

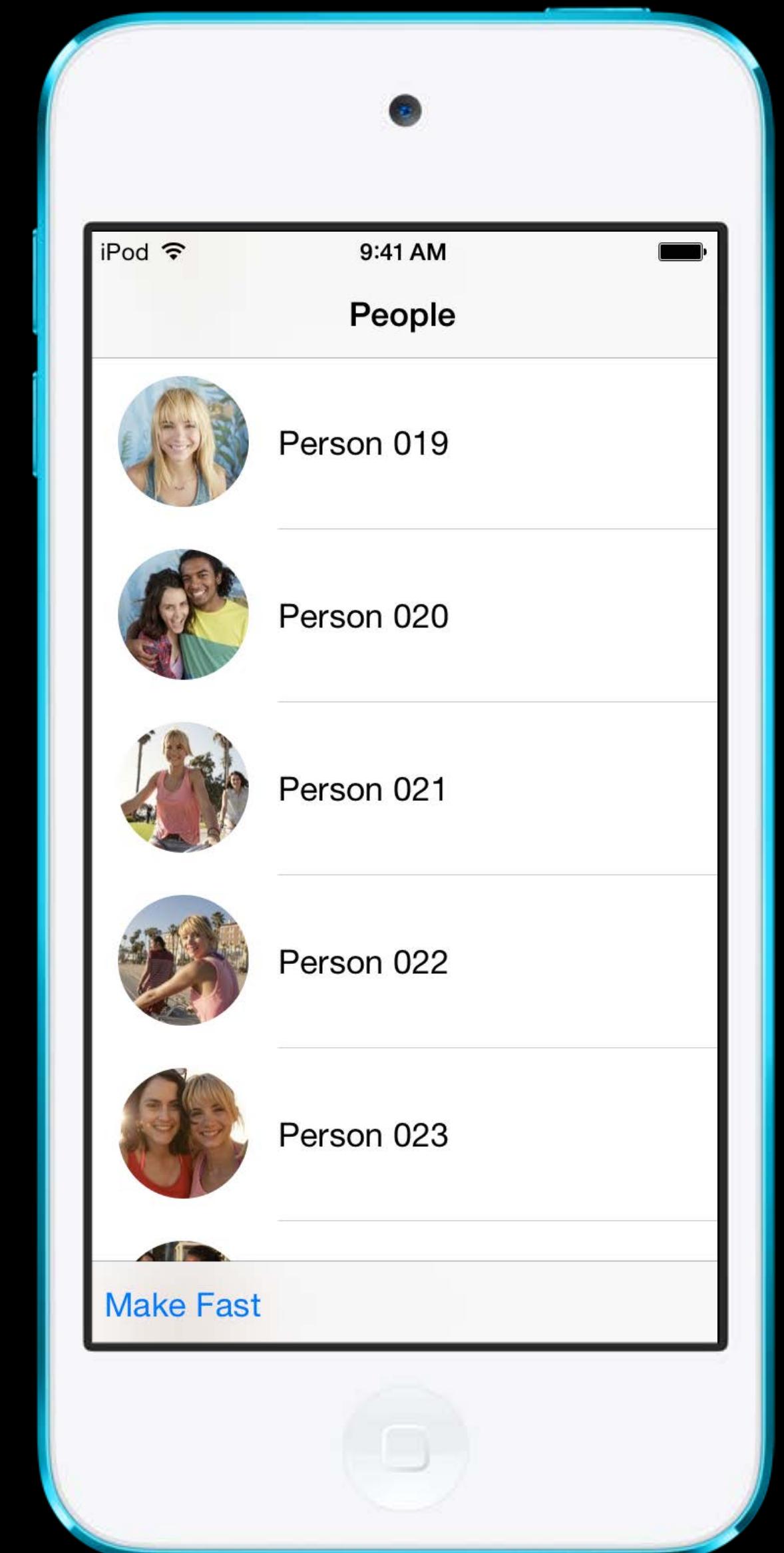
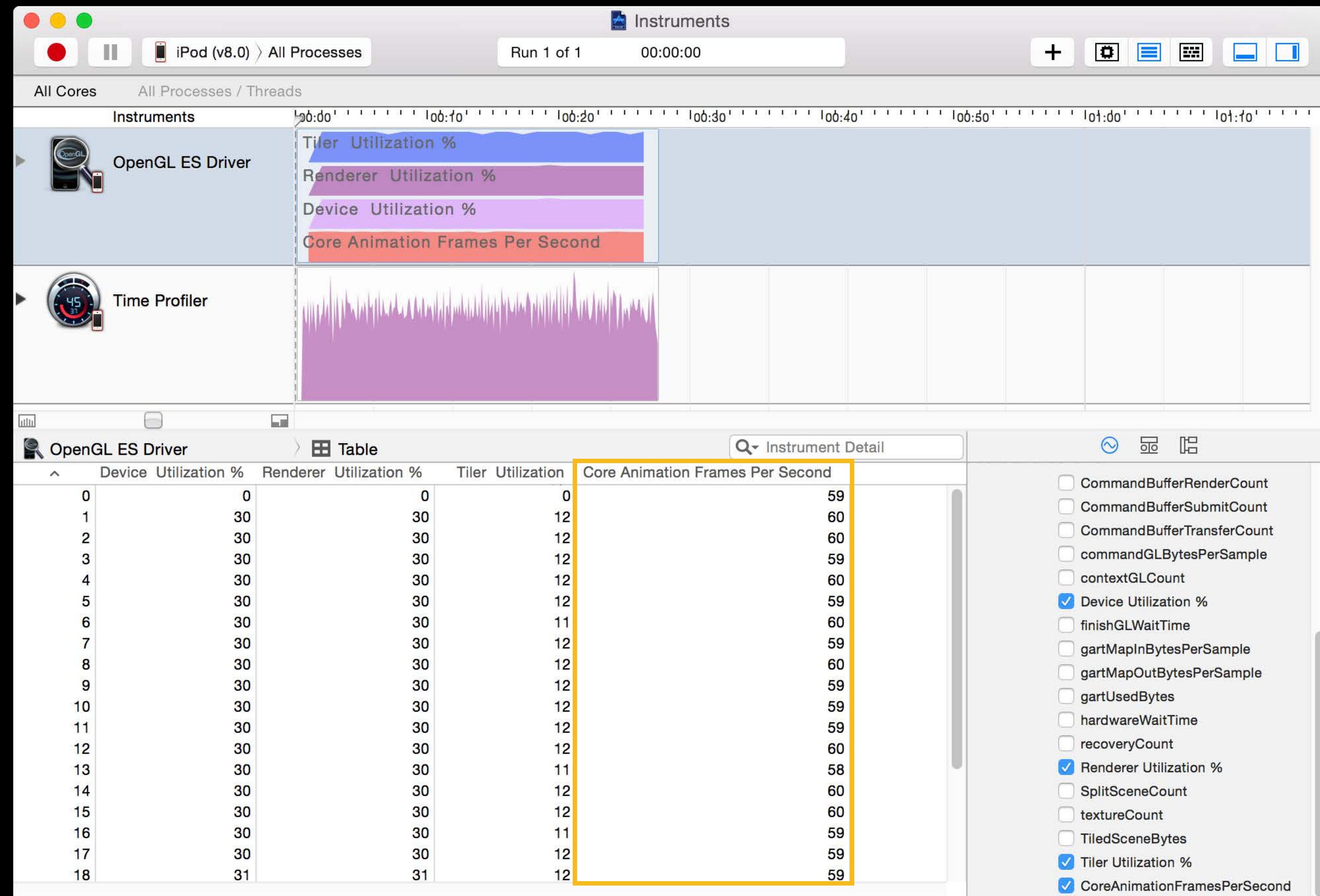
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



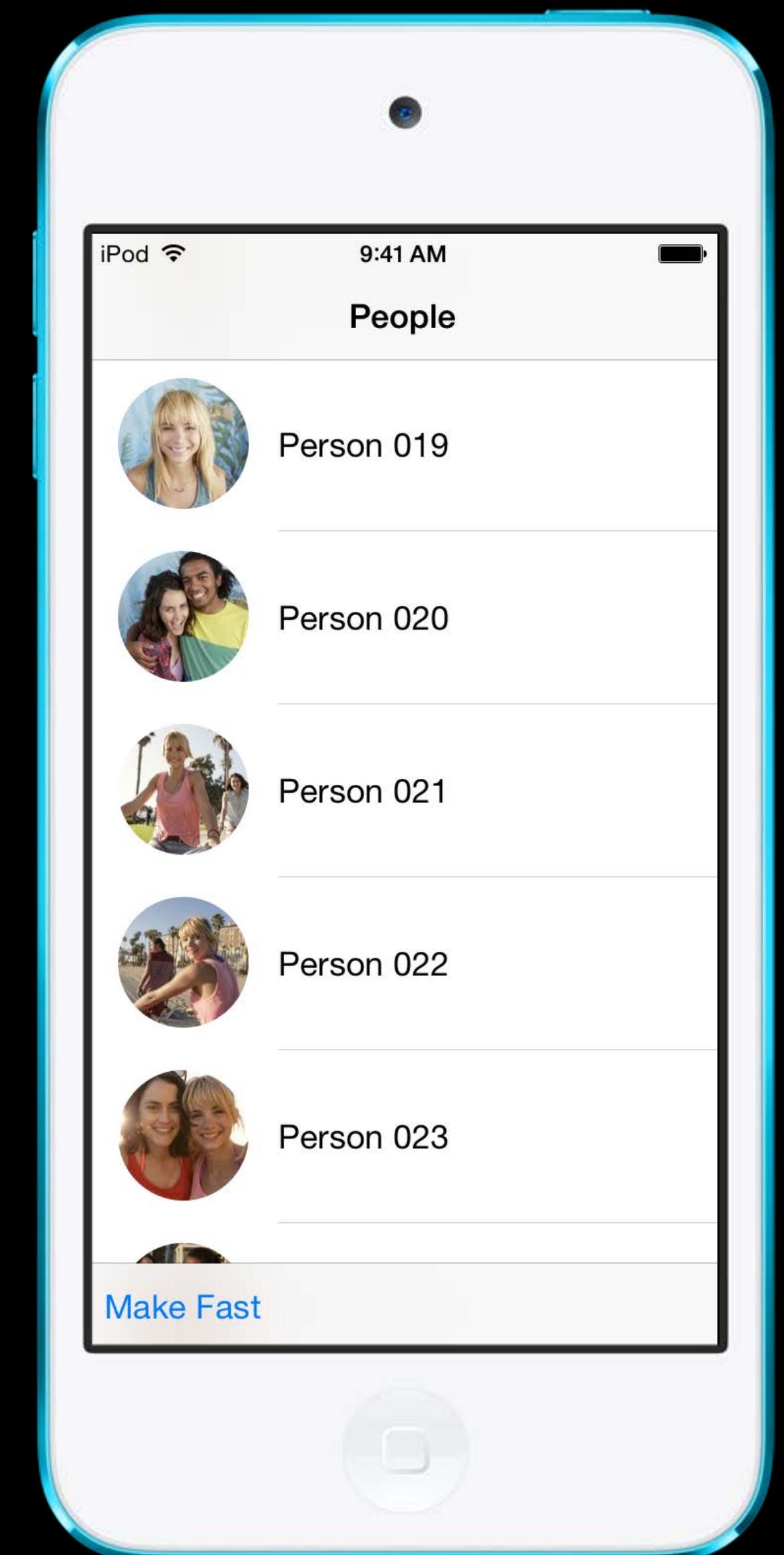
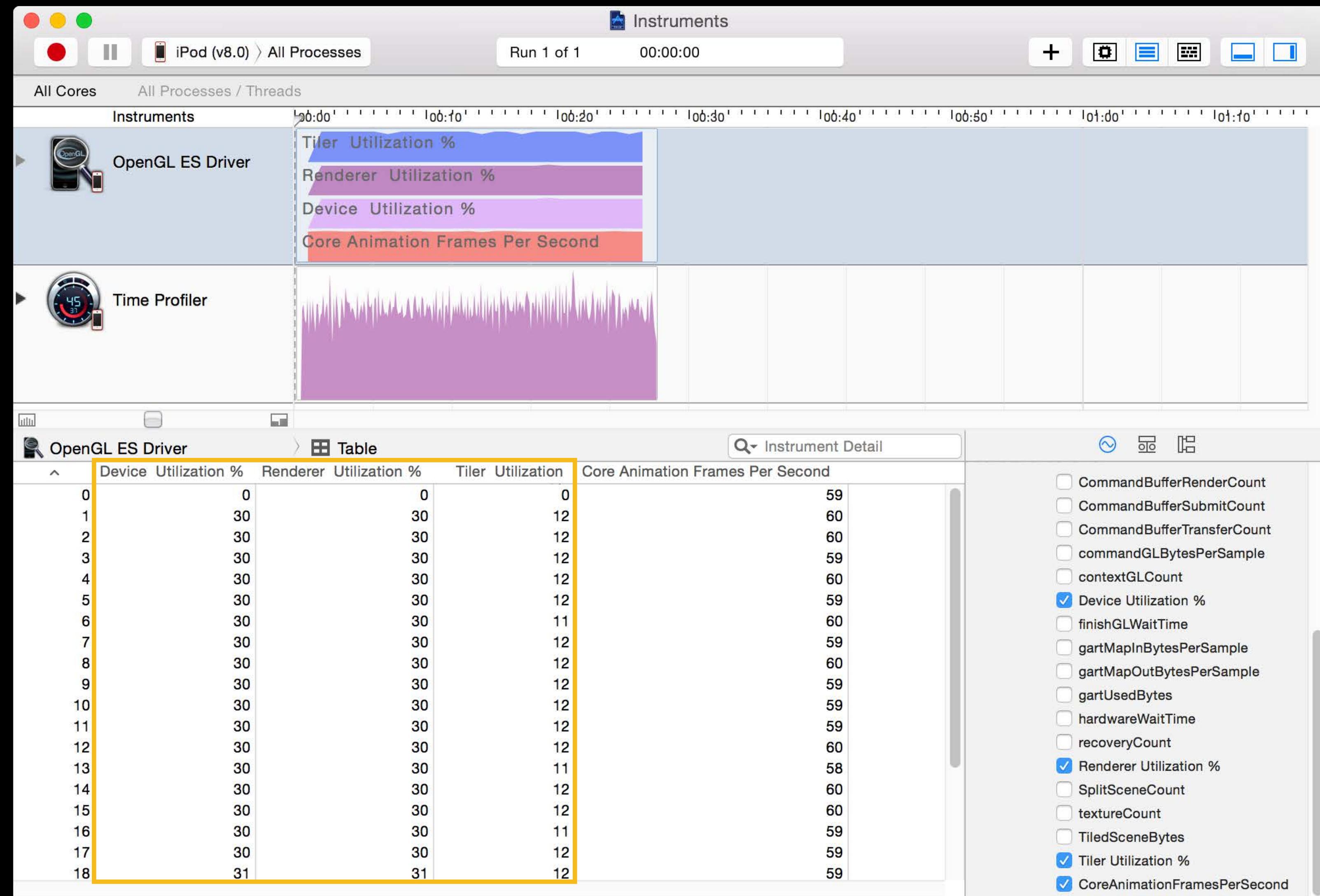
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



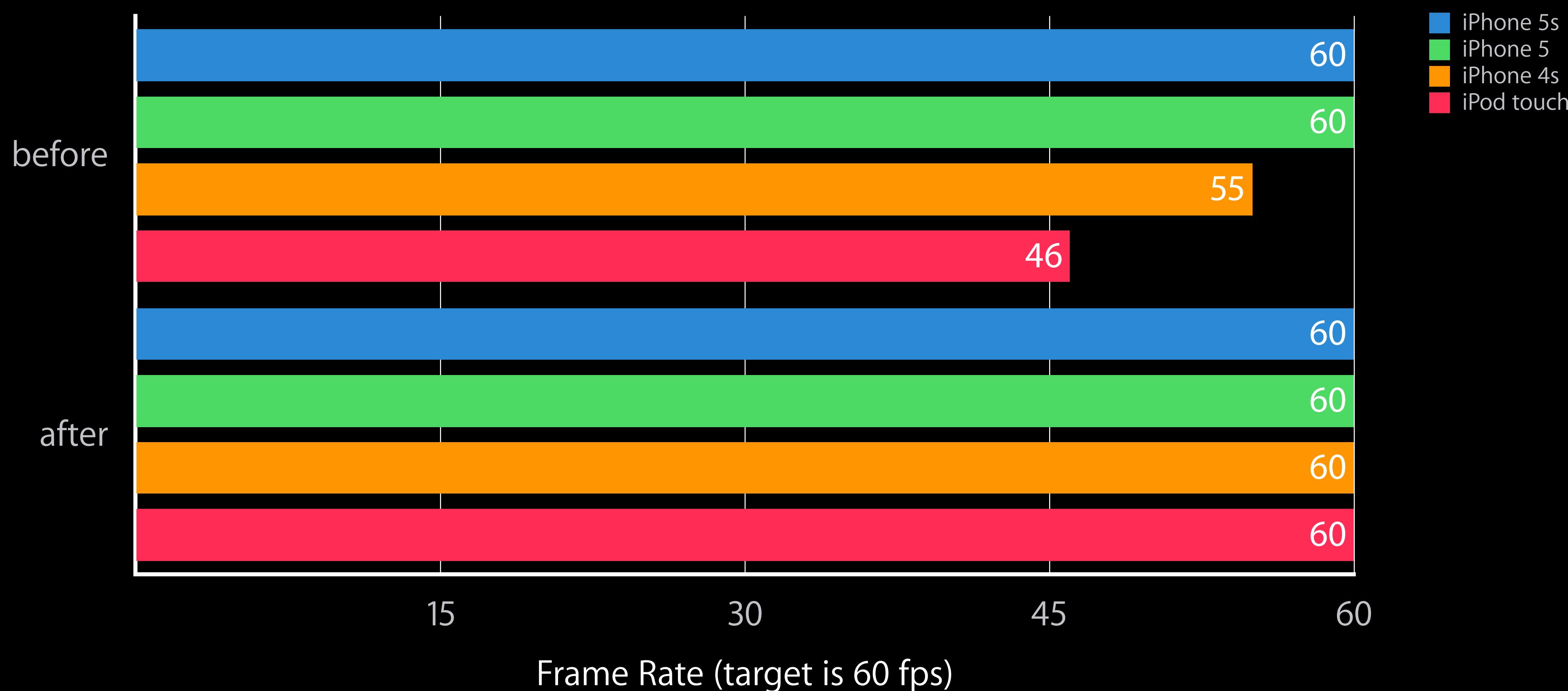
Measure Frame Rate on iPod touch

OpenGL ES Driver instrument



Fictitious Contacts Application

Performance across devices



Fictitious Contacts Application

Summary

Offscreen passes are expensive

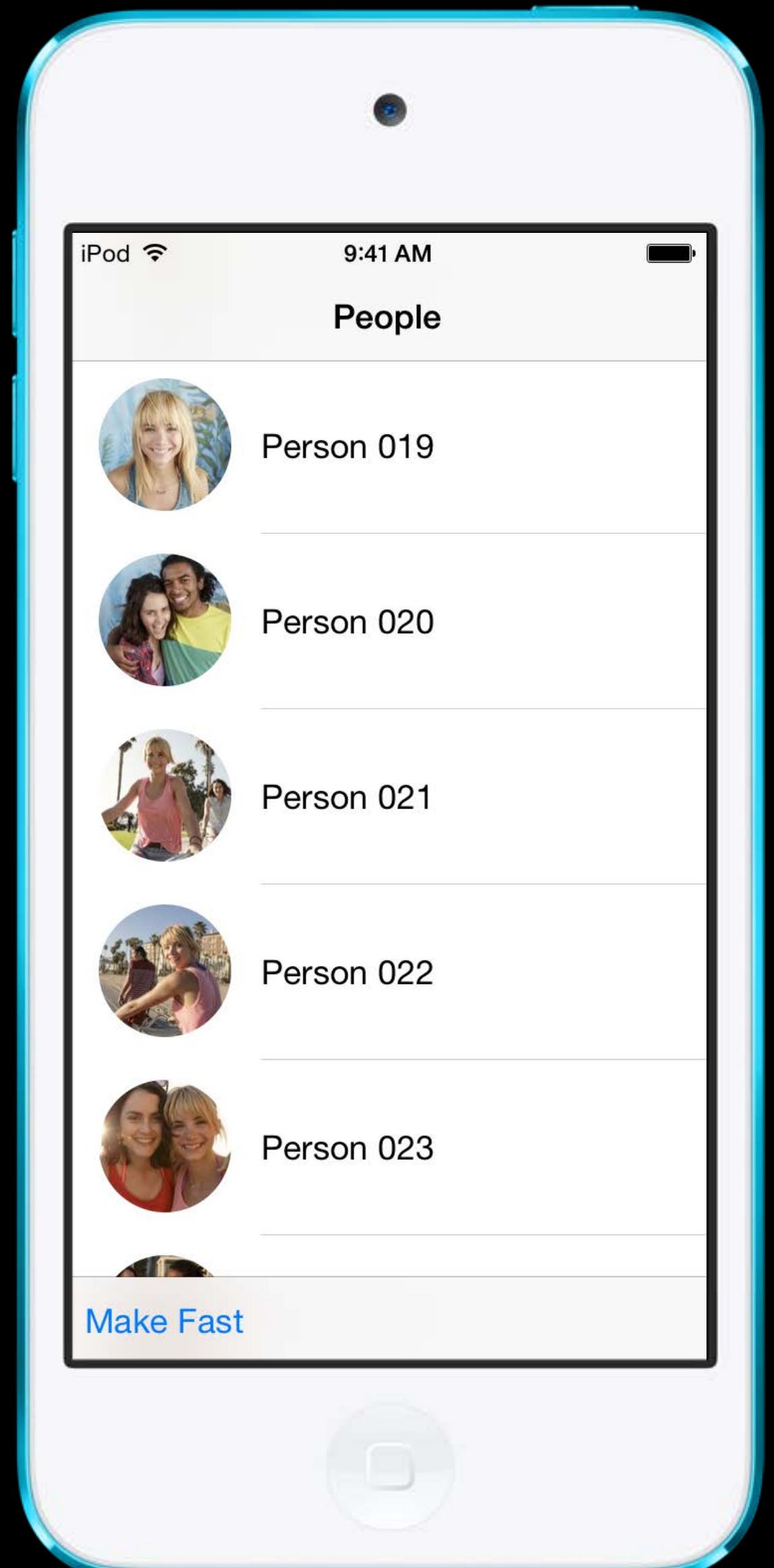
- Use Core Animation instrument to find them
- Know what you can do to avoid them

Measure performance across different devices

- Use OpenGL ES Driver instrument for GPU time
- Use Time Profiler instrument for CPU time

Know your view hierarchy and any hidden costs

- This is especially true for table cells and scrolling



Performance Investigation Mindset

Summary

What is the frame rate?		Core Animation or OpenGL ES Driver instrument
CPU or GPU bound?		OpenGL ES Driver and Time Profiler instrument
Any unnecessary CPU rendering?		Time Profiler instrument
Too many offscreen passes?		Core Animation instrument
Too much blending?		Core Animation instrument
Any strange image formats or sizes?		Core Animation instrument
Any expensive views or effects?		Xcode View Debugger
Anything unexpected in hierarchy?		Xcode View Debugger

Summary

Core Animation pipeline

Rendering concepts

UIBlurEffect

UIVibrancyEffect

Profiling tools

Case studies

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Dave DeLong
Developer Tools Evangelist
delong@apple.com

Documentation
Core Animation
http://developer.apple.com/library/IOs/documentation/Cocoa/Conceptual/CoreAnimation_guide/Introduction/Introduction.html

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

Related Sessions

-
- Improving Your App with Instruments Marina Tuesday 4:30PM
 - Debugging in Xcode 6 Marina Wednesday 10:15AM
 - Writing Energy Efficient Code, Part 1 Russian Hill Wednesday 10:15AM
 - Writing Energy Efficient Code, Part 2 Russian Hill Wednesday 11:30AM
 - Creating Custom iOS User Interfaces Marina Wednesday 3:15PM
 - Building Interruptible and Responsive Interactions Presidio Friday 11:30AM
-

Labs

- Core Animation and Quartz 2D Lab Graphics and Games Lab A Tuesday 2:00PM
 - Interface Builder and Live Views Lab Tools Lab C Wednesday 9:00AM
 - Power and Performance Lab Core OS Lab B Wednesday 2:00PM
 - Dynamics, View Animations, and Core Animation Lab Frameworks Lab A Thursday 9:00AM
 - Power and Performance Lab Core OS Lab A Thursday 3:15PM
 - Visual Effects and Appearance Customization Lab Frameworks Lab A Friday 9:00AM

