

# Visual Debugging with Xcode

Session 410

Chris Miles Xcode Debugger UI

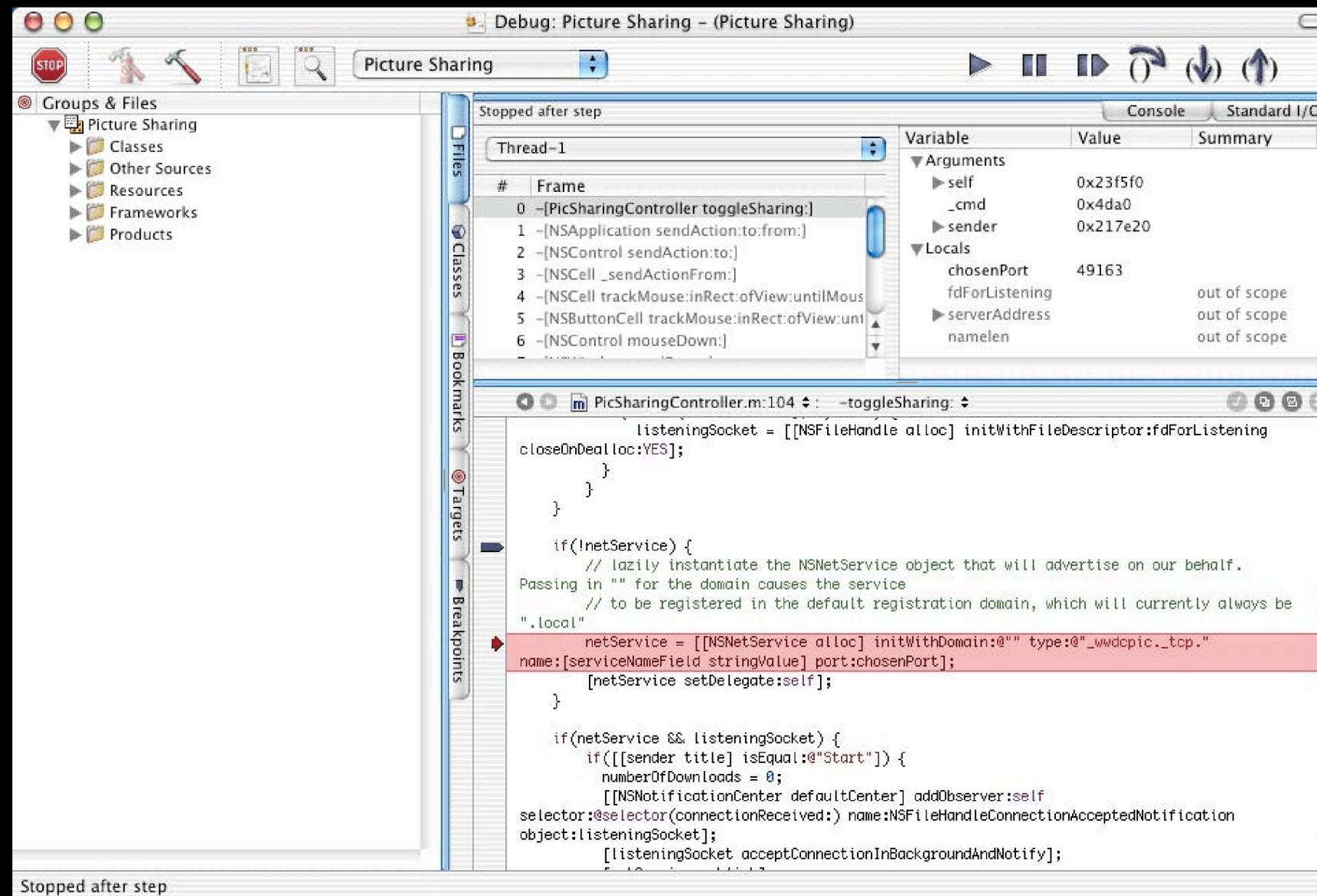
Tyler Casella Game Technologies

Daniel Delwood Software Radiologist



# Debugging

Later...



# Debugging Now...

The screenshot displays the Xcode development environment with the following components:

- Top Bar:** Shows the project name "Trailblazer" and the device "iPad Pro (9.7 inch)".
- Left Panel (Organizer):** Displays system metrics (CPU 0%, Memory 572.6 MB, Disk Zero KB/s, Network Zero KB/s) and a hierarchical view of the app's UI hierarchy. The selected element is an `UIImageView` within a `UIVisualEffectView`.
- Center Canvas:** A 3D perspective view of the app's interface, showing a landscape image, a text block titled "Regional Information", and a list of reviews.
- Right Panel (Property Inspector):** Shows the properties of the selected `UIImageView`.
  - Object:** Class Name: `UIImageView`, Address: `0x7fe06fc362b0`.
  - Image View:** Image: A small thumbnail of the landscape image. Highlighted: Empty Selection. State: Not Highlighted.
  - View:** Layer: `<CALayer: 0x7fe06fc48c60>`, Layer Class: `CALayer`.
  - Content Mode:** Aspect Fill.
  - Tag:** 0.
  - Interaction:** User Interaction Enabled Off, Multiple Touch Off.
  - Alpha:** 1.
  - Background:** White:0 Alpha:0.
  - Tint:** R:0.14 G:0.63 B:0.19 A:1.
  - Drawing:** Opaque On, Hidden Off, Clears Graphics Context On, Clip Subviews On, Autoresize Subviews On.
  - Stretching:** X: 0, Y: 0, Width: 1, Height: 1.
  - Accessibility:** Not Accessibility Element.
- Bottom Panel (Debug Console):** Shows a thread dump for `Thread 1` and a log entry: `0 TrailDetailsViewController.(viewDidAppear(Bool) -> ()).(closure #1)`. Below the log, the `lldb` command `po self.trail` is shown, returning an `Optional<Trail>` with a value of `<Trailblazer.Trail: 0x7fe071a02e10>`.

# Overview

Runtime issues

View debugging

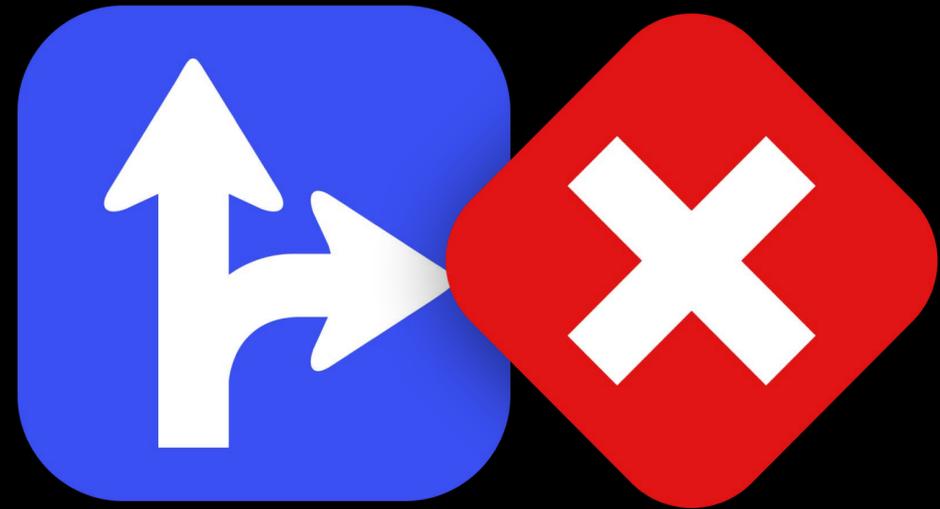
State machine Quick Looks

SpriteKit/SceneKit FPS gauge

Memory graph debugging

# Runtime Issues







Runtime Issues

# Runtime Issues

NEW

The screenshot displays the Xcode IDE with the following components:

- Toolbar:** Contains standard Xcode controls like play, stop, and zoom, along with a 'Trailblazer' icon.
- Left Sidebar:** Shows the 'Runtime' tab with a tree view of issues. The selected issue is 'Data race in Trailblazer.TrailDetailsViewController.resetUpdateState () -> () at updateFinished'. Below it, a stack trace shows the sequence of events: 'updateFinished' is a global variable, followed by a write of size 1 by thread 15, and then a write of size 1 by thread 16.
- Code Editor:** Displays the Swift code for TrailDetailsViewController. A red error banner highlights a data race in the `resetUpdateState()` function, specifically on the line `updateFinished = true`. The code includes methods like `fillDescriptionTextView()`, `loadInitialData()`, `routeEstimator?.requestUpdate()`, `resetUpdateState()`, `viewDidAppear()`, and `willTransition()`.
- Bottom Toolbar:** Includes a 'Filter' button and various navigation and tool icons.

# Runtime Issues

NEW

Trailblazer > iPhone SE

Running Trailblazer on iPhone SE

Trailblazer > Trailblazer > TrailDetailsViewController.swift

Runtime

```
fillDescriptionTextView()
updateUIForTraitCollection(self.navigationController!.traitCollecti

loadInitialData()

routeEstimator?.requestUpdate(completion: { success in
    self.updateDidFinish(withSuccess: success)
})
}

func resetUpdateState() {
    updateFinished = true
}
```

Data race in Trailblazer.TrailDetailsViewController.resetUpdateState(): 'updateFinished' is a global variable (0x1031d8b...)

by thread 15

ViewController.resetUpdateSta...

# Runtime Issues

NEW

Trailblazer > iPhone SE

Running Trailblazer on iPhone SE

Trailblazer > Trailblazer > TrailDetailsViewController.swift > TrailDetailsViewContro

Runtime

Issues

ized or destroyed mutex in

Trailblazer.TrailDetailsViewController  
( ) -> ( ) at updateFinished

' is a global variable (0x1031d8b

by thread 15

ViewController.resetUpdateSta...

```
fillDescriptionTextView()
updateUIForTraitCollection(self.navigationController!.traitCollecti

loadInitialData()

routeEstimator?.requestUpdate(completion: { success in
    self.updateDidFinish(withSuccess: success)
})
}

func resetUpdateState() {
    updateFinished = true
}
```

Data race in Trailblazer.TrailDetailsViewController.resetUp

# Runtime Issues

NEW

The screenshot shows the Xcode interface with the following details:

- Window Title:** Trailblazer > iPhone SE
- Toolbar:** Includes icons for folder, hierarchy, search, warning, zoom, list, and messages.
- Buildtime / Runtime:** The 'Runtime' tab is selected.
- Issue List:**
  - Trailblazer - 40710 2 issues
    - Threading Issues
      - Use of an uninitialized or destroyed mutex in CommitChange
      - Data race in Trailblazer.TrailDetailsViewController.resetUpdateState () -> () at updateFinished**
        - 'updateFinished' is a global variable (0x1031d8b68)
      - Write of size 1 by thread 15
        - 0 TrailDetailsViewController.resetUpdateSta...
        - 1 TrailDetailsViewController.notify() -> ()
        - 2 TrailDetailsViewController.updateDidFinis...
        - 3 TrailDetailsViewController.(viewDidLoad()...

The code editor on the right shows Swift code with a warning icon next to the `updateFinished =` line in the `resetUpdateState` function:

```
fillDescriptionT
updateUIForTrai

loadInitialData(

routeEstimator?
    self.updated
})
}

func resetUpdateStat
    updateFinished =
}

override func viewDi
    super.viewDidApp

// Scroll to mic
scrollView scroll
```

# Runtime Issues

NEW

Trailblazer > iPhone SE

Running Trailblazer on iPhone SE

Buildtime **Runtime**

Trailblazer - 40710 2 issues

- Threading Issues
  - Use of an uninitialized or destroyed mutex in CommitChange
  - Data race in Trailblazer.TrailDetailsViewController.resetUpdateState () -> () at updateFinished**
    - 'updateFinished' is a global variable (0x1031d8b68)
  - Write of size 1 by thread 15
    - 0 TrailDetailsViewController.resetUpdateSta...
    - 1 TrailDetailsViewController.notify() -> ()
    - 2 TrailDetailsViewController.updateDidFinis...
    - 3 TrailDetailsViewController.(viewDidLoad()...

```
fillDescription()
updateUIForTrail

loadInitialData()

routeEstimator?
    self.updated
})
}

func resetUpdateStat
updateFinished =
}

override func viewDi
super.viewDidApp

// Scroll to mic
scrollView scroll
```

# Runtime Issues

Participating tools

NEW

# Runtime Issues

Participating tools

NEW



Threads

# Runtime Issues

Participating tools

NEW



UI



Threads

# Runtime Issues

Participating tools

NEW



UI



Threads



Memory

# Runtime Issues

## Thread Sanitizer

NEW

Data races

Use of uninitialized mutexes

Unlock from wrong thread

Thread leaks

Unsafe calls in signal handlers

The screenshot shows the Thread Sanitizer console for a process named 'Trailblazer - 40710'. It lists two issues under 'Threading Issues'. The first issue is 'Use of an uninitialized or destroyed mutex in CommitChange'. The second issue, which is highlighted, is 'Data race in Trailblazer.TrailDetailsViewController.resetUpdateState () -> () at updateFinished'. Below this issue, it notes that 'updateFinished' is a global variable (0x1031d8b68). The console also shows two write operations: one by thread 15 and another by thread 16, both of size 1. The write by thread 15 is associated with several method calls, including 'resetUpdateSta...', 'notify() -> ()', 'updateDidFinis...', '(viewDidLoad())...', 'partial apply for TrailDetailsViewController...', and 'thunk'. The write by thread 16 is associated with 'writeResultAndReset', 'notifyWithResult', and 'UpdateDidFinish'.

# View Debugging

# View Debugging

```
(lldb) po [self.view recursiveDescription]
```

```
<UIView: 0x7fcc9f613f60; frame = (0 0; 375 667); autoresize = RM+BM; layer = <CALayer: 0x7fcc9f6140f0>>  
  <UIScrollView: 0x7fcc9f614270; frame = (0 0; 375 667); clipsToBounds = YES; autoresize = RM+BM; gestureRecognizers = <NSA  
    <UIView: 0x7fcc9f614400; frame = (166.5 24; 42 20.5); text = 'Label'; opaque = NO; autoresize = RM+BM; userInt  
    <UIButton: 0x7fcc9f614a10; frame = (164.5 52.5; 46 30); opaque = NO; autoresize = RM+BM; layer = <CALayer: 0x7f  
      <UIButtonLabel: 0x7fcc9f615080; frame = (0 6; 46 18); text = 'Button'; opaque = NO; userInteractionEnabled  
    <UISegmentedControl: 0x7fcc9f6158a0; frame = (61 0; 60 29); opaque = NO; layer = <CALayer: 0x7fcc9f615d70>>  
      <UISegment: 0x7fcc9f6161d0; frame = (7 6.5; 46 16); text = 'Second'; opaque = NO; userInteractio  
      <UIImageView: 0x7fcc9f6161d0; frame = (60 0; 1 29); alpha = 0; opaque = NO; autoresize = LM; userInte  
    <UISegment: 0x7fcc9f6161d0; frame = (0 0; 60 29); opaque = NO; layer = <CALayer: 0x7fcc9f6161d0>>  
      <UISegmentLabel: 0x7fcc9f6161d0; frame = (16.5 6.5; 27 16); text = 'First'; opaque = NO; userInteract  
      <UIImageView: 0x7fcc9f6161d0; frame = (60 0; 1 29); opaque = NO; autoresize = LM; userInteractionEnab  
    <UITextField: 0x7fcc9f6161d0; frame = (98.5 246; 178 30); text = ''; clipsToBounds = YES; opaque = NO; autoresi  
      <_UITextFieldRoundedRectBackgroundViewNeue: 0x7fcc9f6161d0; frame = (0 0; 178 30); opaque = NO; autoresize  
      <UITextFieldLabel: 0x7fcc9f6161d0; frame = (7 0.5; 164 27.5); text = 'This is a placeholder'; opaque = NO;  
    <UITextField: 0x7fcc9f6161d0; frame = (142.5 285; 90.5 24.5); text = 'This is text'; clipsToBounds = YES; opaqu  
      <UITextFieldLabel: 0x7fcc9f6161d0; frame = (2 2; 87 20.5); text = 'This is text'; opaque = NO; userInterac  
    <UIButton: 0x7fcc9f6161d0; frame = (226.5 56.5; 22 22); opaque = NO; autoresize = RM+BM; layer = <CALayer: 0x7f  
      <UIImageView: 0x7fcc9f6161d0; frame = (0 0; 22 22); clipsToBounds = YES; opaque = NO; userInteractionEnabl
```





# View Debugging



# View Debugging



# View Debugging

The screenshot displays the Xcode interface for debugging a mobile application. The central pane shows a 3D view of the app's UI hierarchy, with various view layers and images visible. The left sidebar shows the project structure, including the hierarchy of views and controllers. The right sidebar shows the property inspector for the selected `UIImageView` object.

**Object**

- Class Name: `UIImageView`
- Address: `0x7f9bcff0bb30`

**Image View**

- Image:
- Highlighted: Empty Selection
- State: Not Highlighted

**View**

- Layer: `<CALayer: 0x7f9bcff0a0a0>`
- Layer Class: `CALayer`
- Content Mode: Scale To Fill
- Tag: 0
- Interaction: User Interaction Enabled Off
- Multiple Touch Off
- Alpha: 1
- Background: `<nil color>`
- Tint: `R:0.14 G:0.63 B:0.19 A:1`
- Drawing: Opaque On
- Hidden: Off
- Clears Graphics Context On
- Clip Subviews: Off
- Autosize Subviews: On
- Stretching X: 0
- Stretching Y: 0
- Width: 1
- Height: 1

**Accessibility**

- Not Accessibility Element
- Value: `<null>`

# View Debugging

Better than ever

NEW

# View Debugging

Better than ever

NEW

Up to 70% faster snapshots

70%

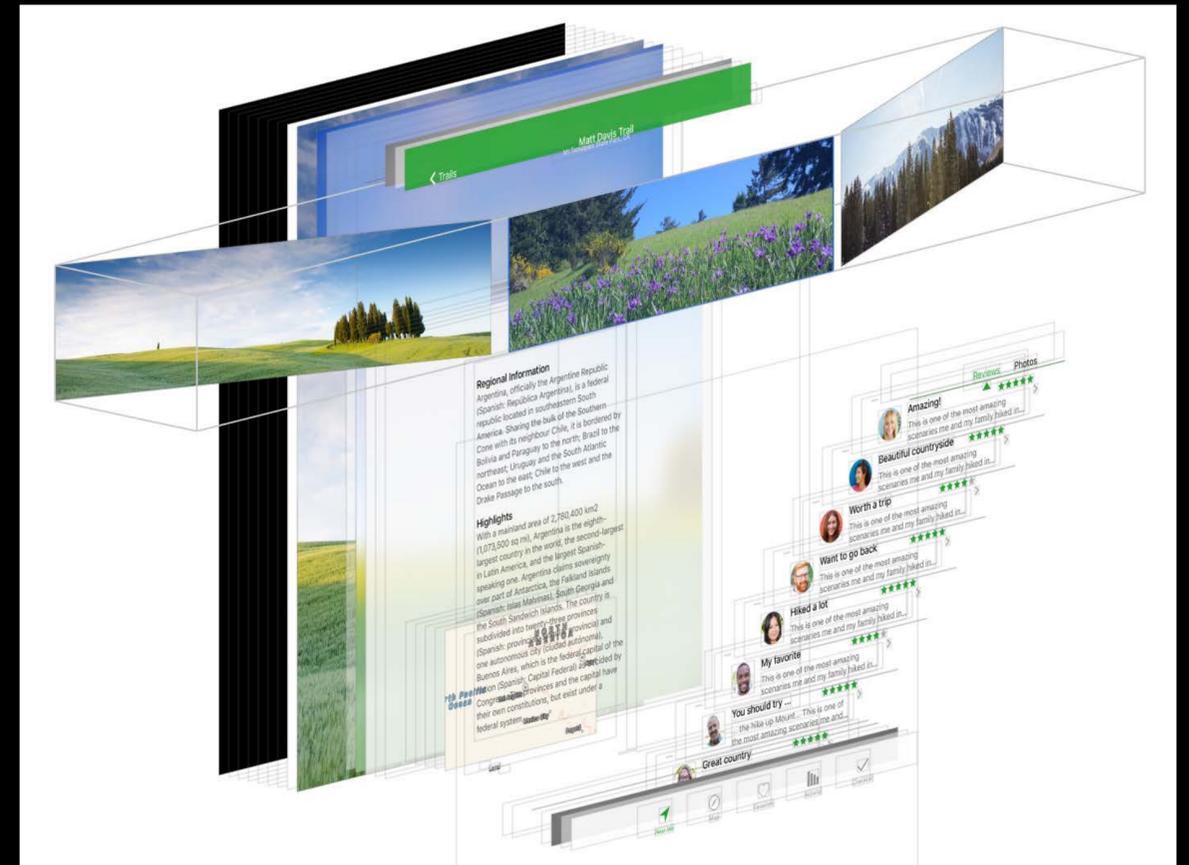
# View Debugging

## Better than ever

NEW

Up to 70% faster snapshots

Layout and transform accuracy



# View Debugging

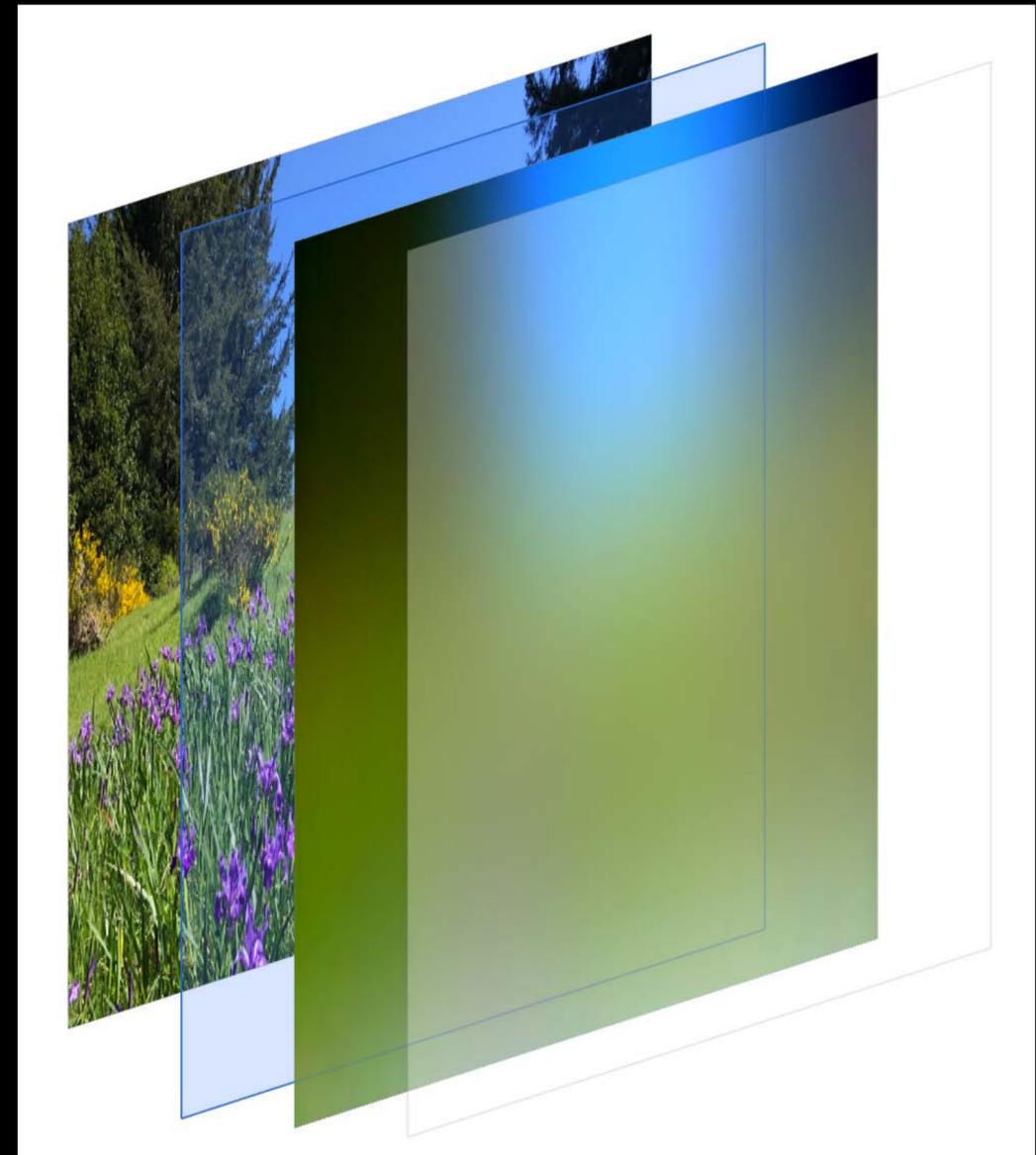
Better than ever

NEW

Up to 70% faster snapshots

Layout and transform accuracy

Blur rendering



# View Debugging

Better than ever

NEW

Up to 70% faster snapshots

Layout and transform accuracy

Blur rendering

Jump to class

**Object**

Class Name DemoBots.InstructionsLayoutView



# View Debugging

Better than ever

NEW

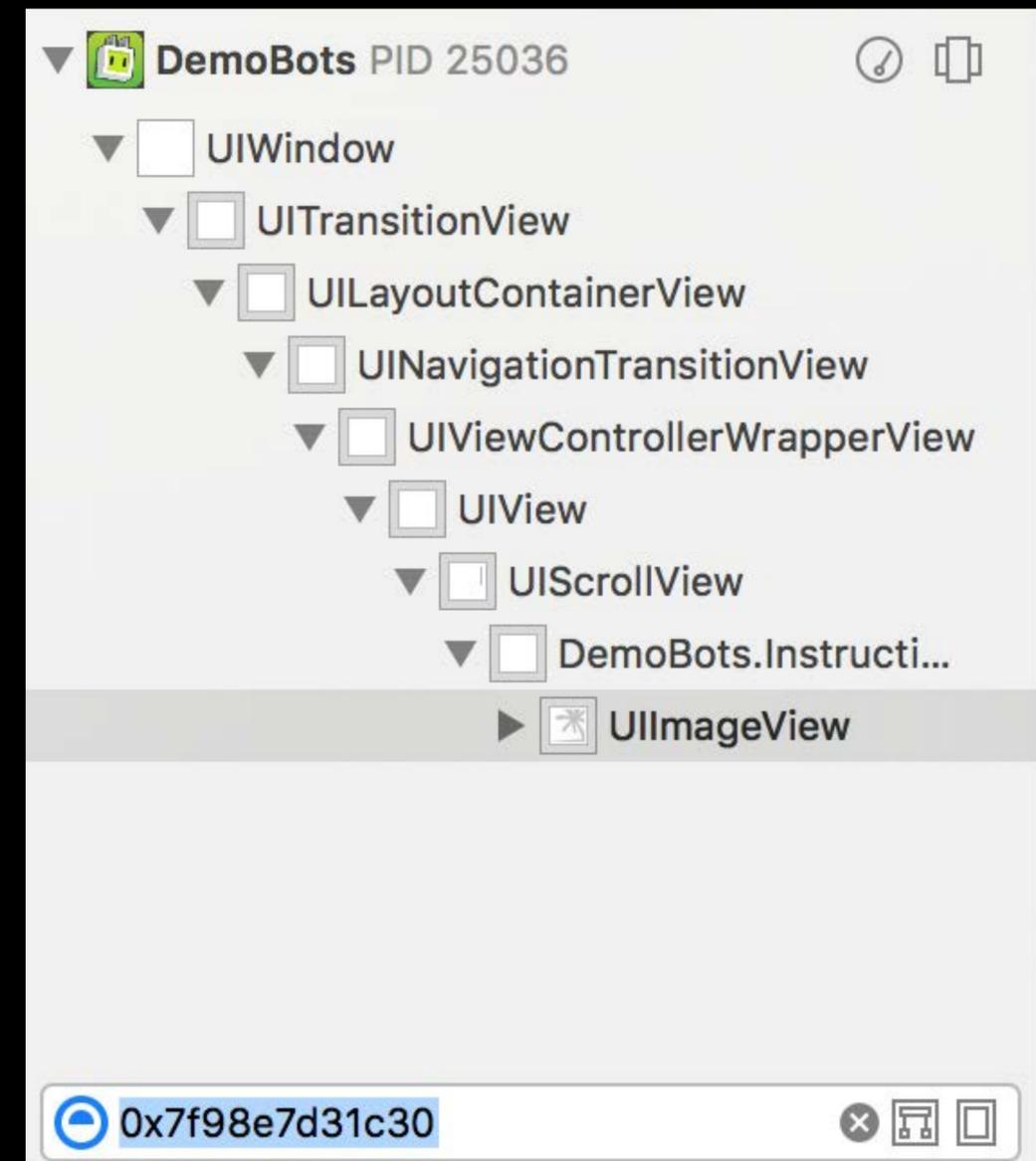
Up to 70% faster snapshots

Layout and transform accuracy

Blur rendering

Jump to class

Navigator filtering



# View Debugging

Better than ever

NEW

Up to 70% faster snapshots

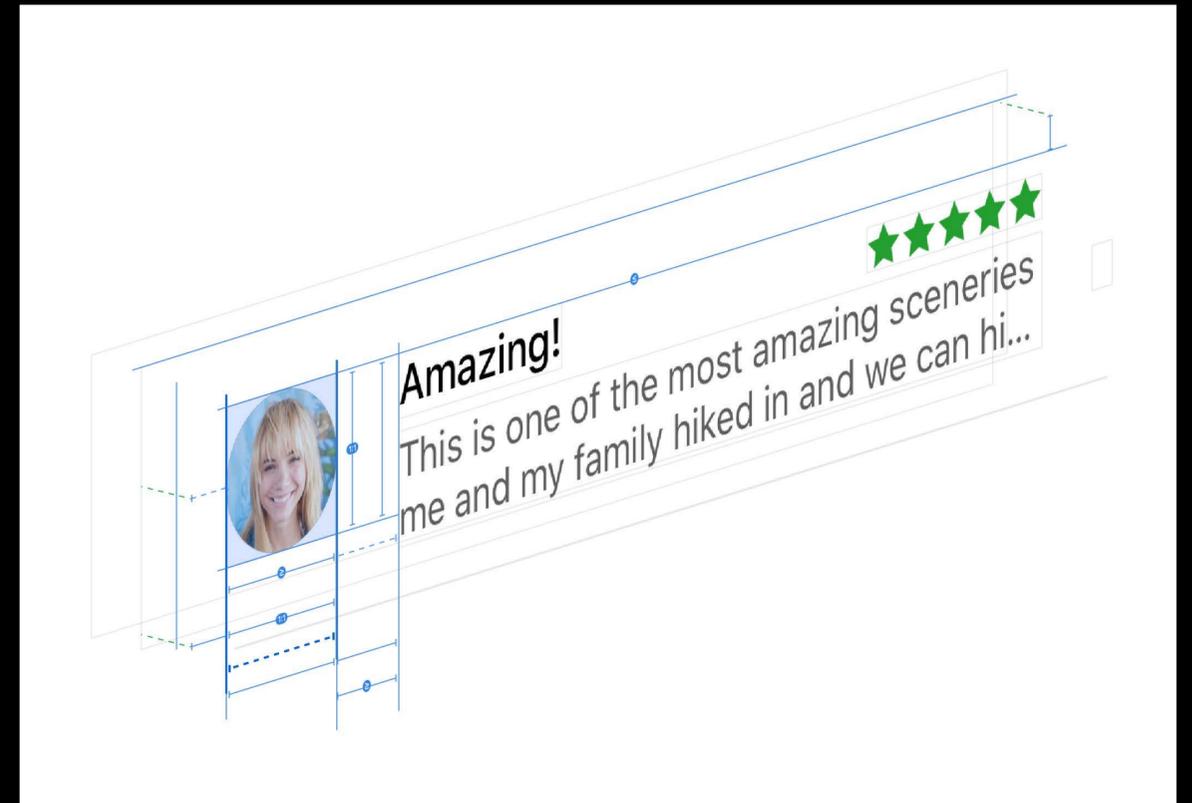
Layout and transform accuracy

Blur rendering

Jump to class

Navigator filtering

Auto Layout debugging



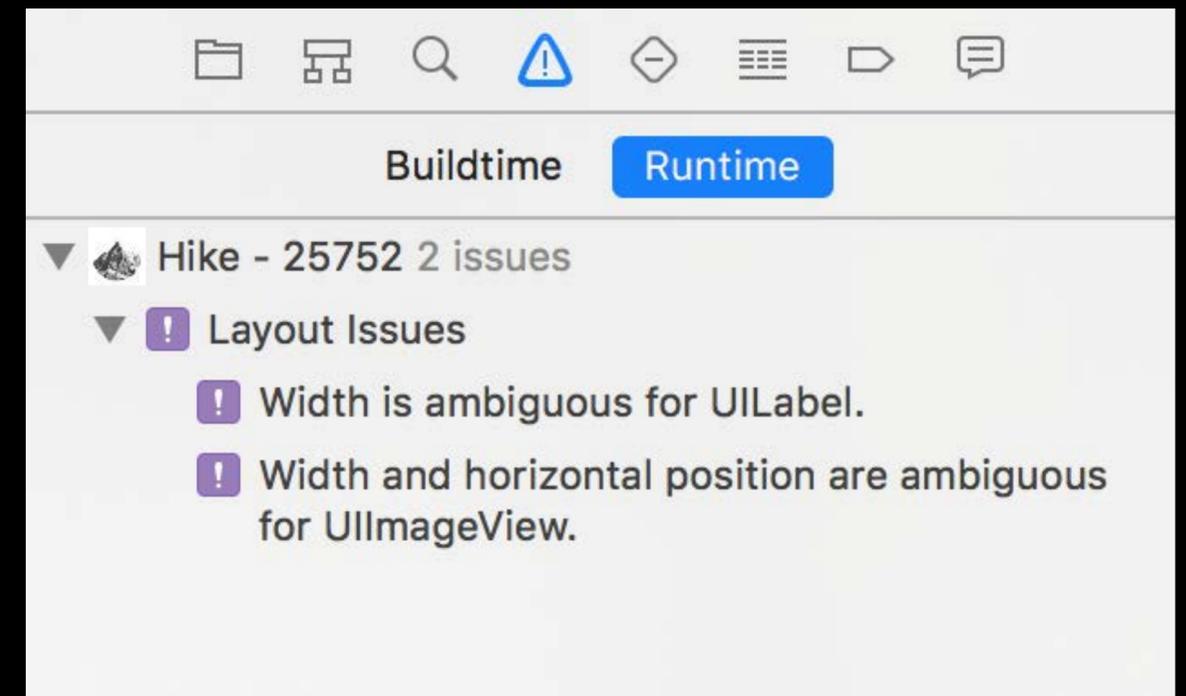
# View Debugging

## Ambiguous layout issues

NEW

Ambiguous layouts are reported as runtime issues

- Highlighted in the activity viewer
- Listed in the issue navigator

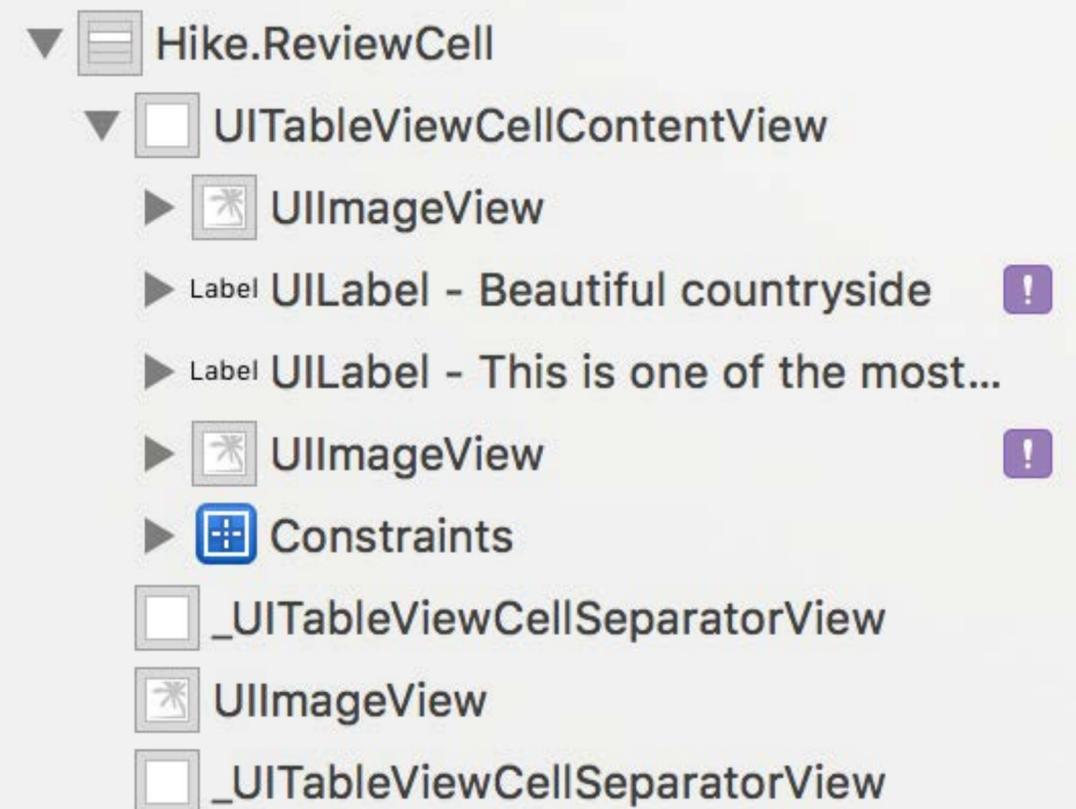


# View Debugging

## Ambiguous layout issues

NEW

Ambiguous layout issues are badged in the view hierarchy outline



# View Debugging

## Ambiguous layout issues

Ambiguous layout issues are explained in the view's size inspector

NEW

The screenshot displays the Xcode View Debugging Size Inspector for a view. The 'View' section shows the 'Frame' and 'Bounds' with their respective coordinates and dimensions. The 'Position' and 'Anchor Point' sections show their respective values. The 'Constraints' section highlights an ambiguous width and horizontal position, listing several constraints with their priorities. The 'Content Hugging Priority' and 'Content Compression Resistance Priority' sections show their respective values.

**View**

Frame X: 217  
Y: 19.5  
W: 79  
H: 14

Bounds X: 0  
Y: 0  
W: 79  
H: 14

Position X 256.5  
Y 26.5  
Z 0

Anchor Point X 0.5  
Y 0.5  
Z 0

**Constraints**

- ! Width and horizontal position are ambiguous.
- superview.trailing = self.trailing @ 1000
- self.width = 79 @ 1000 (content size)
- self.centerY = label.centerY @ 1000
- self.height = 14 @ 1000 (content size)
- self.leading ≥ label.trailing + 4 @ 1000
- label.trailing = self.trailing @ 1000

**Content Hugging Priority**

Horizontal 251  
Vertical 251

**Content Compression Resistance Priority**

Horizontal 751  
Vertical 751

# View Debugging

## Ambiguous layout issues

NEW

Ambiguous layout issues are explained in the view's size inspector

### Constraints

-  Width and horizontal position are ambiguous.
-  `superview.trailing = self.trailing @ 1000`
-  `self.width = 79 @ 1000 (content size)`
-  `self.centerY = label.centerY @ 1000`
-  `self.height = 14 @ 1000 (content size)`
-  `self.leading ≥ label.trailing + 4 @ 1000`
-  `label.trailing = self.trailing @ 1000`

*Demo*

Xcode view debugging

# Recap

Runtime issues

View debugging enhancements

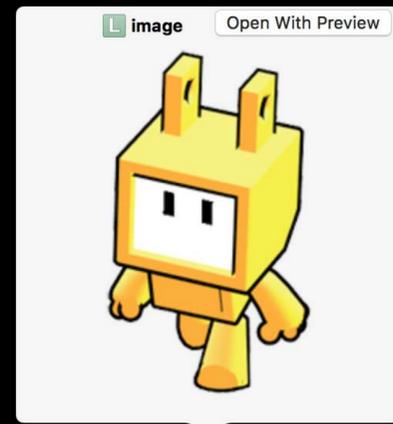
- Ambiguous layout issue reporting
- macOS, iOS, tvOS

# State Machine Quick Look

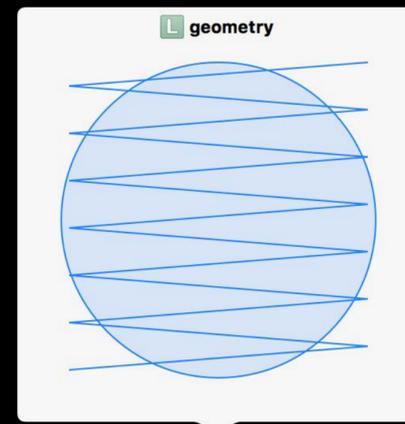
Tyler Casella Game Technologies

# Quick Look

Xcode 7



Images



Geometry



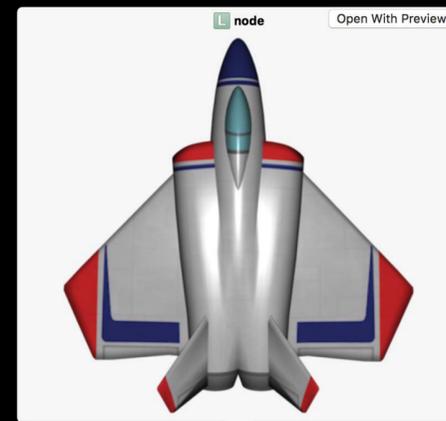
Views



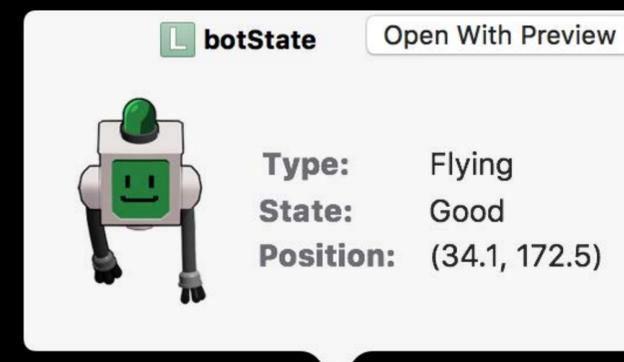
Colors



SpriteKit



SceneKit



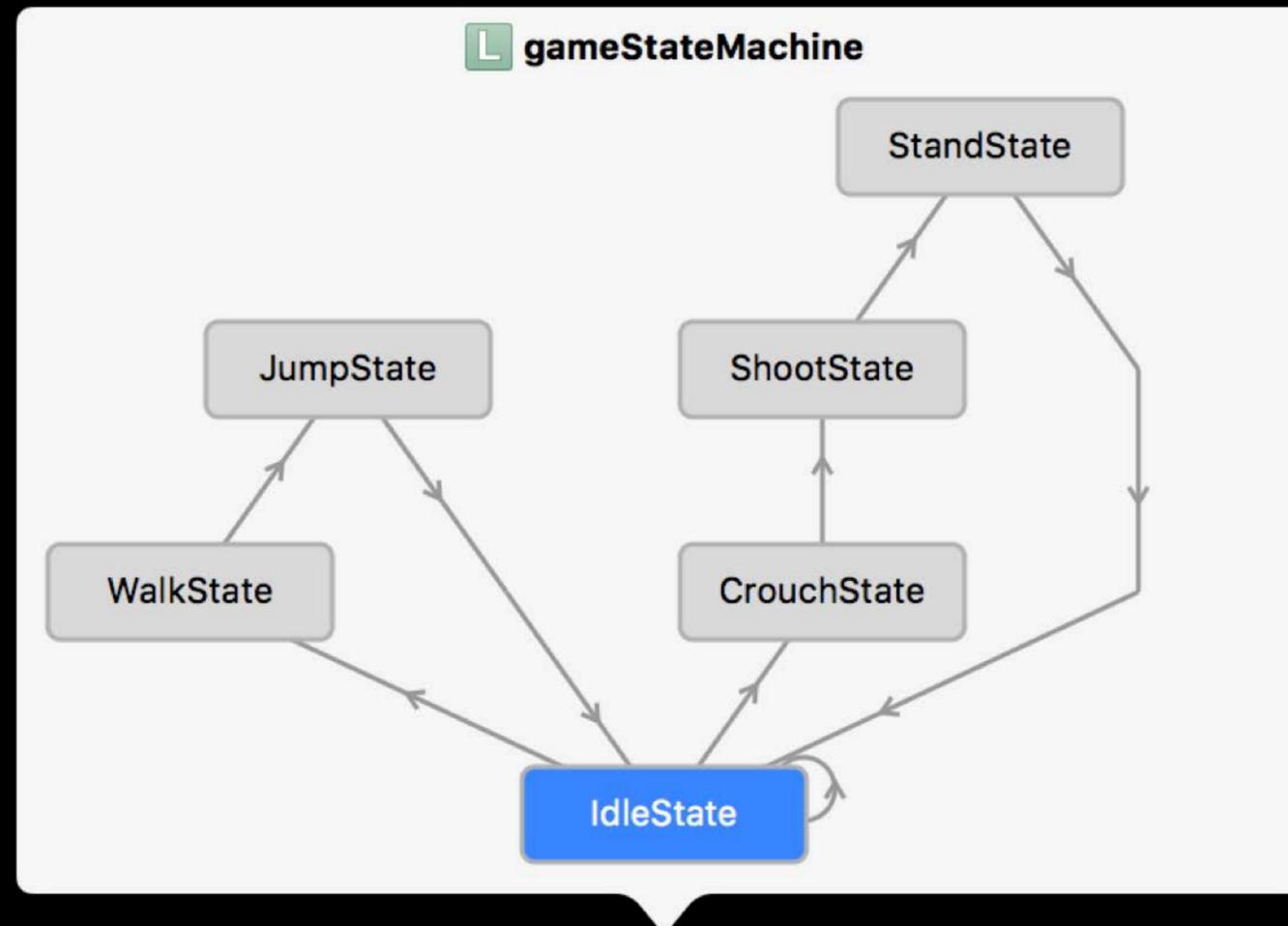
Custom

...and more!

# Quick Look

Xcode 8

NEW



State Machine

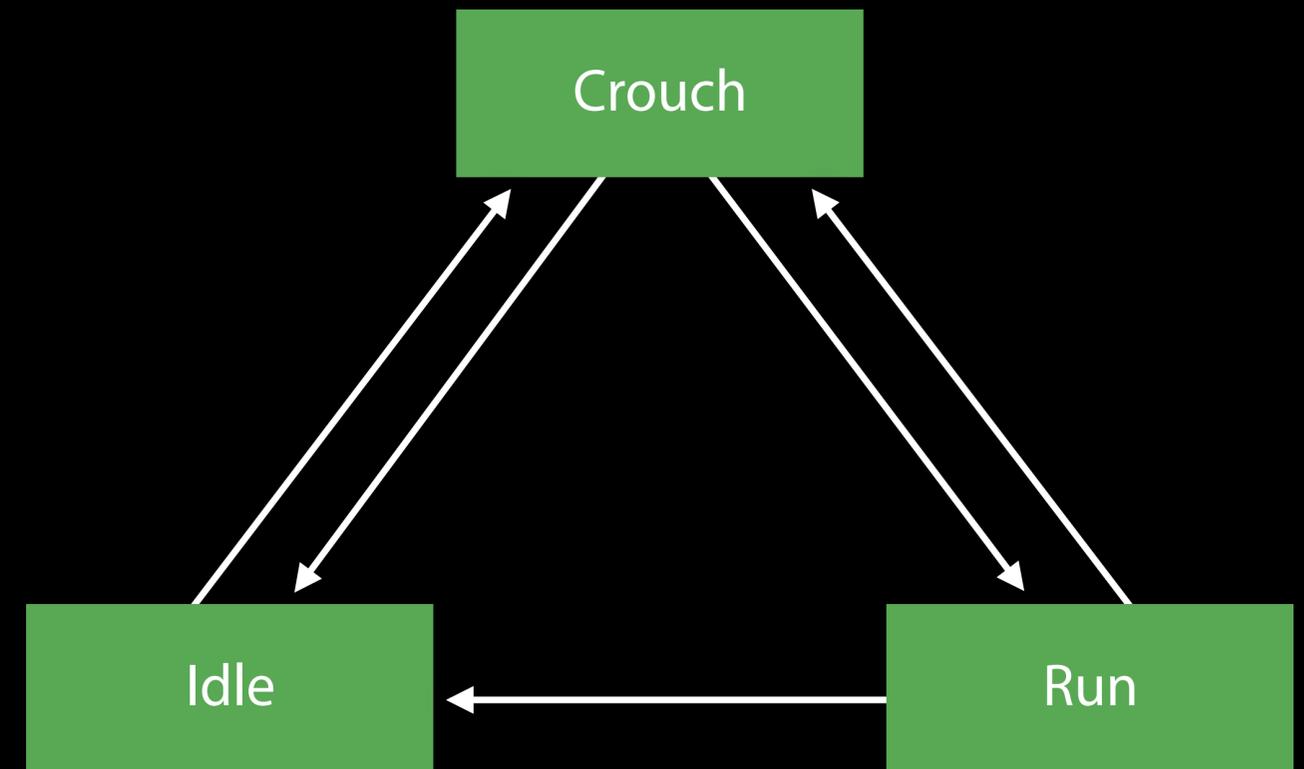
# State Machine Quick Look

## GKStateMachine

NEW

Available via GameplayKit

- macOS, iOS, tvOS



# State Machine Quick Look

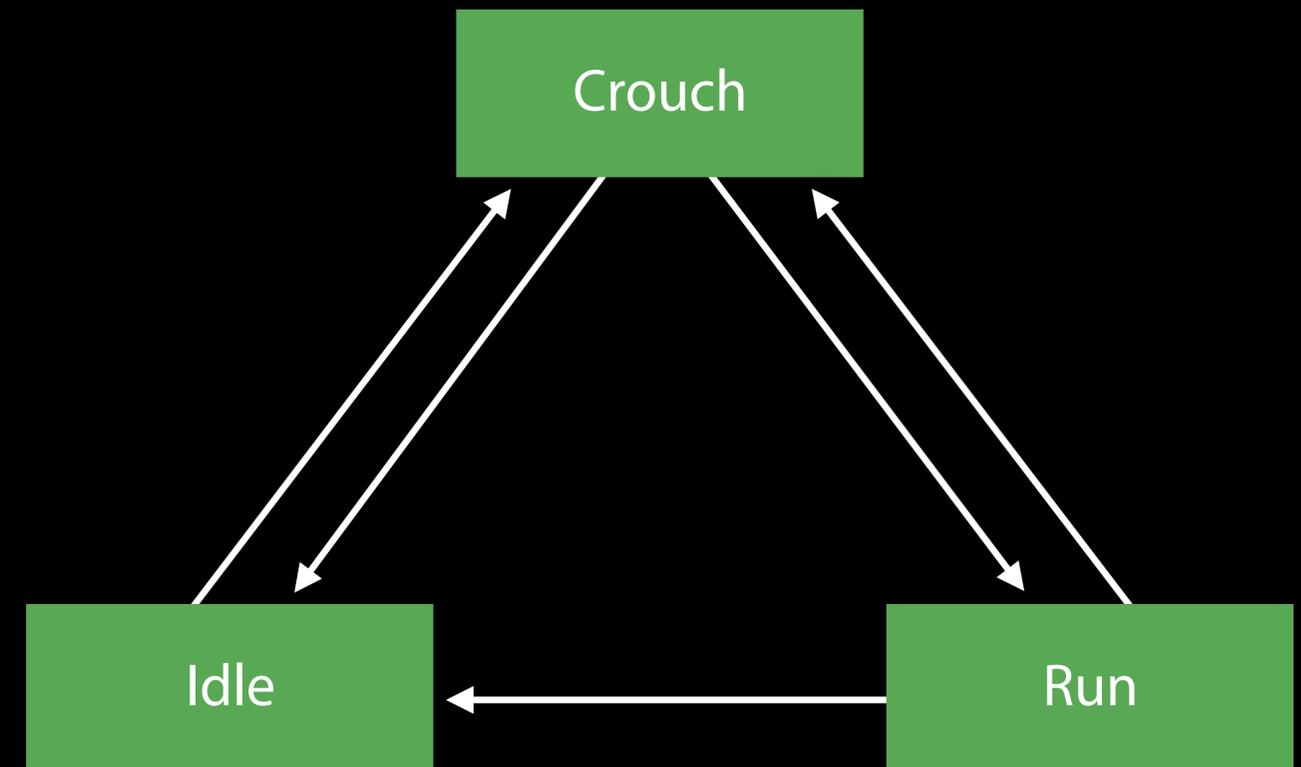
NEW

## GKStateMachine

Available via GameplayKit

- macOS, iOS, tvOS

Directed graph defining complex behavior



# State Machine Quick Look

NEW

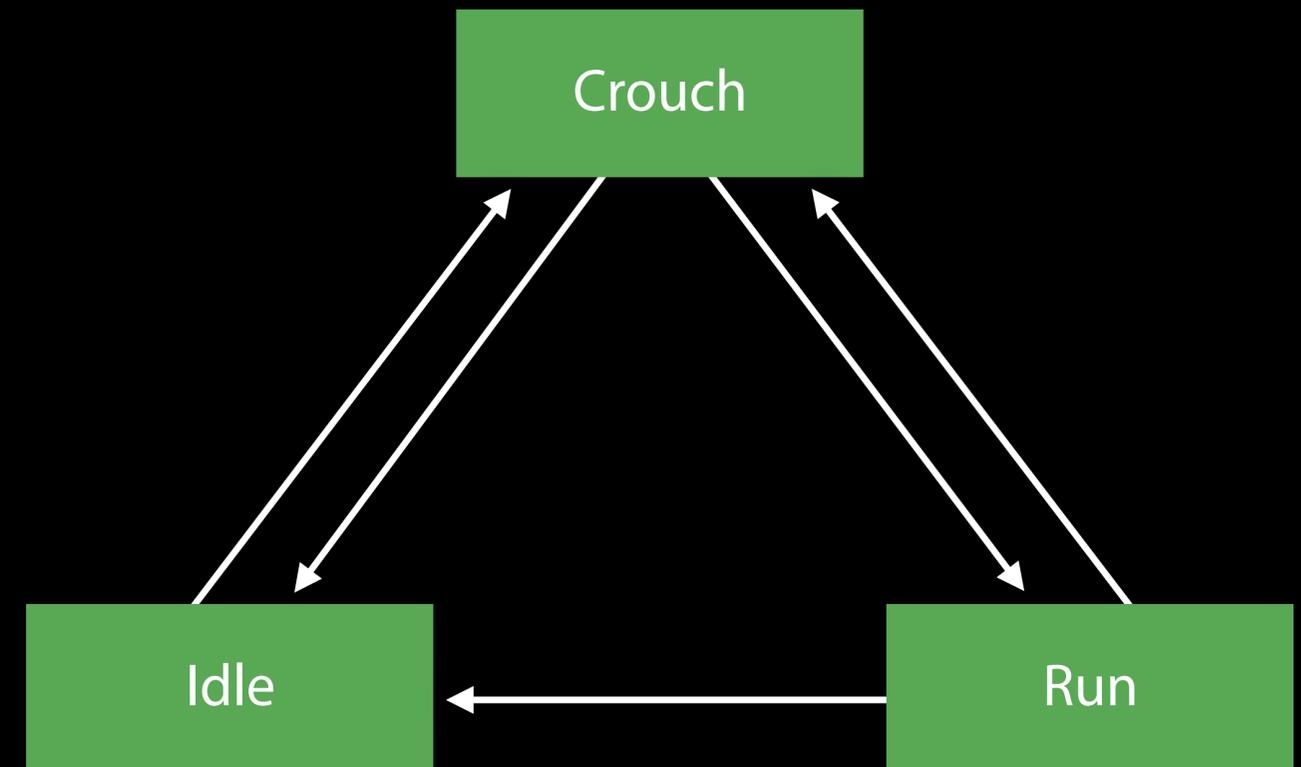
## GKStateMachine

Available via GameplayKit

- macOS, iOS, tvOS

Directed graph defining complex behavior

Provide discrete behavior per-state



# State Machine Quick Look

NEW

## GKStateMachine

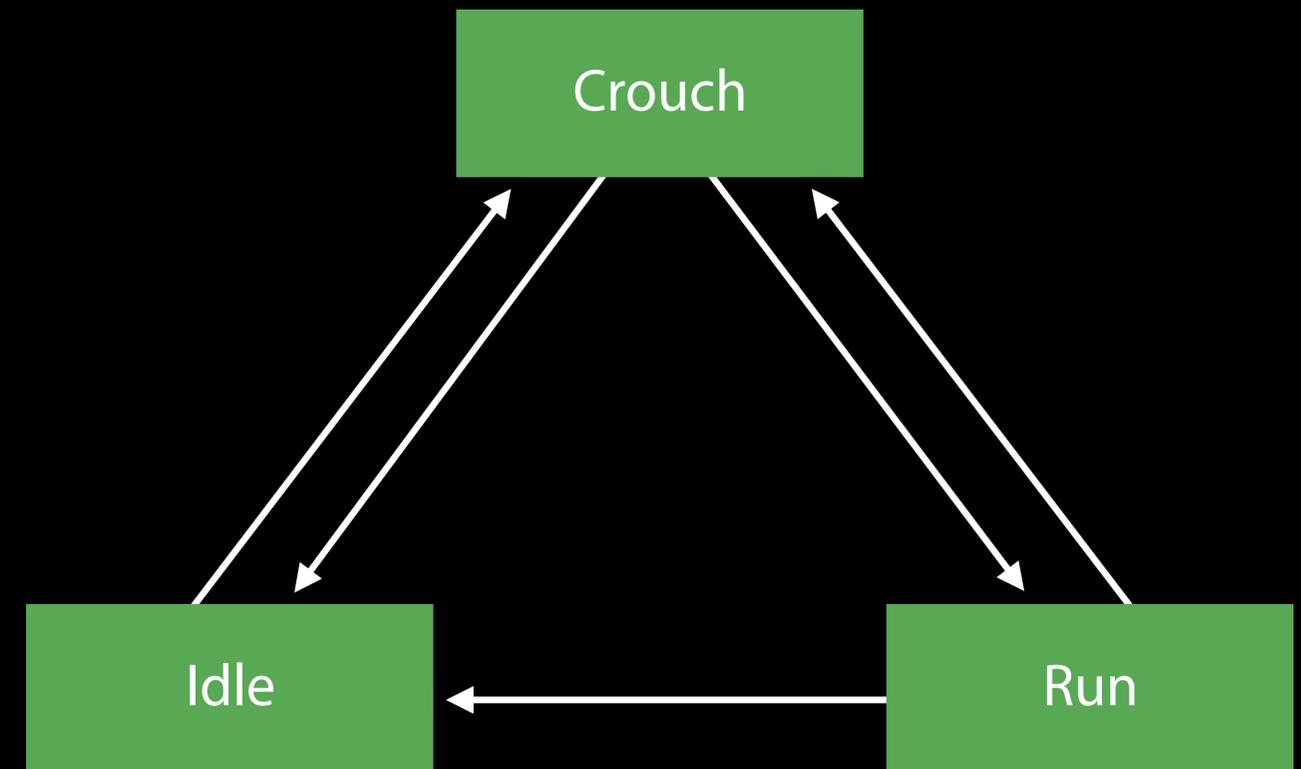
Available via GameplayKit

- macOS, iOS, tvOS

Directed graph defining complex behavior

Provide discrete behavior per-state

Define transitions between states



# State Machine Quick Look

NEW

## GKStateMachine

Available via GameplayKit

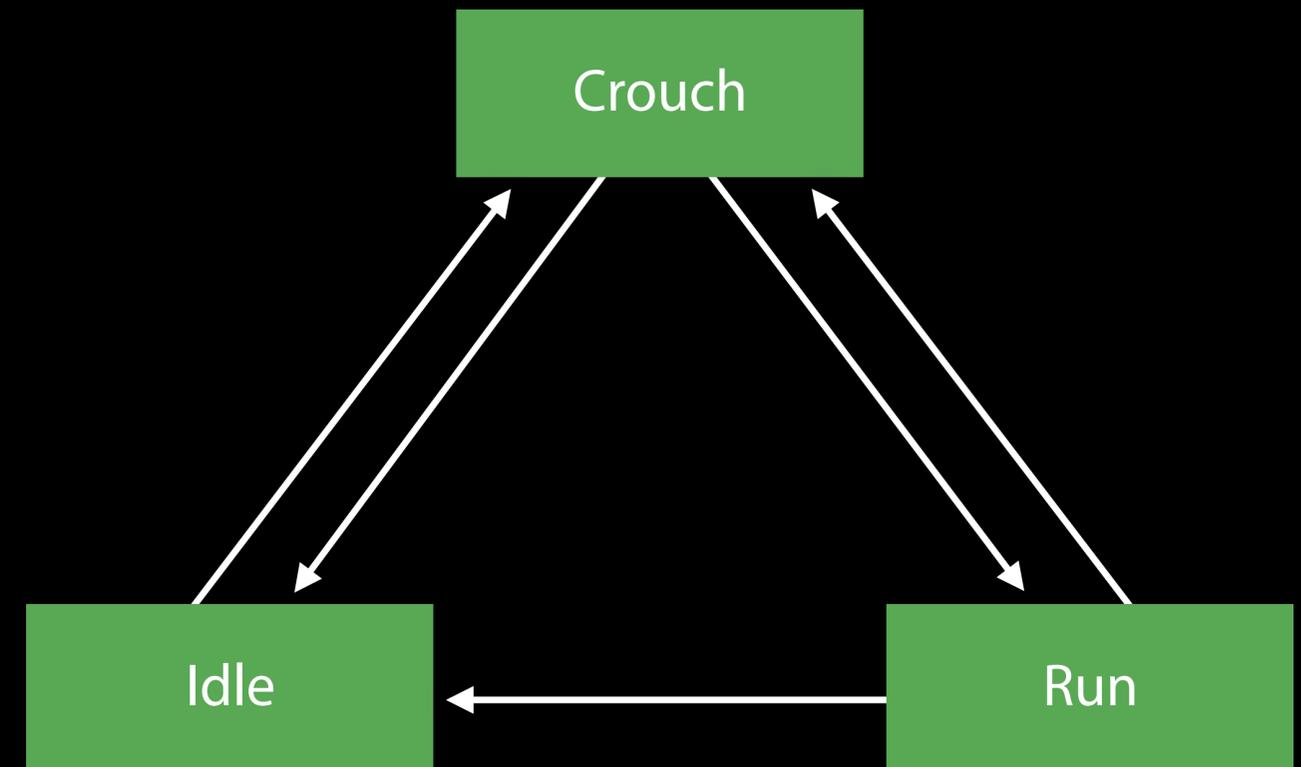
- macOS, iOS, tvOS

Directed graph defining complex behavior

Provide discrete behavior per-state

Define transitions between states

Difficult to visualize from code



# State Machine Quick Look

NEW

## GKStateMachine

Available via GameplayKit

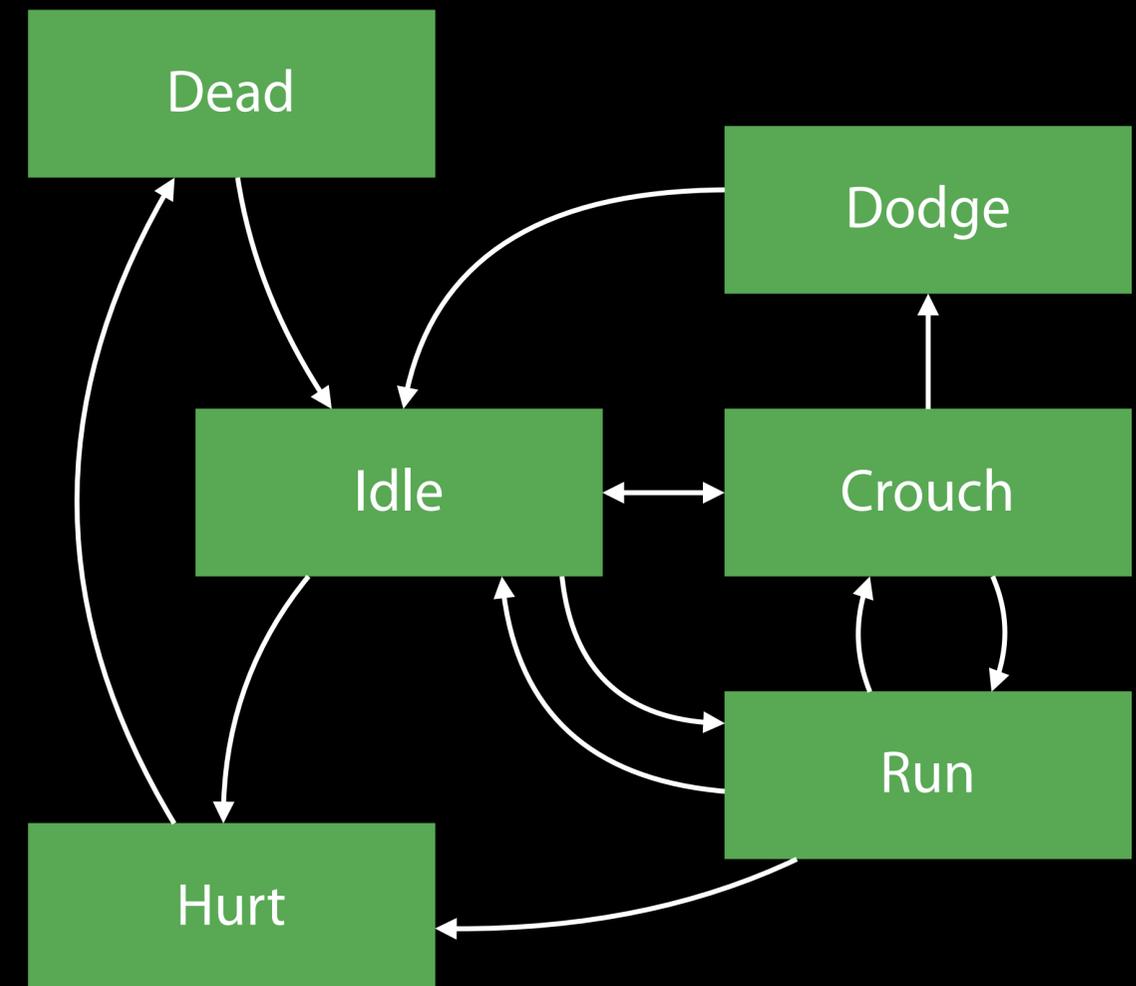
- macOS, iOS, tvOS

Directed graph defining complex behavior

Provide discrete behavior per-state

Define transitions between states

Difficult to visualize from code



# State Machine Quick Look

Then and now

```
(lldb) po machine.currentState
▽ Optional<GKState>
  ▽ some : <DemoBots.BeamIdleState: 0x174026c00>

(lldb) po machine.canEnterState(BeamIdleState)
false

(lldb) po machine.canEnterState(BeamFiringState)
true

(lldb) po machine.canEnterState(BeamCoolingState)
false

(lldb) po machine.canEnterState(BeamDisabledState)
false

(lldb) po machine.canEnterState(BeamChargingState)
false
```

Xcode 7.3

# State Machine Quick Look

Then and now

NEW

```
(lldb) po machine.currentState
▽ Optional<GKState>
  ▽ some : <DemoBots.BeamIdleState: 0x174026c00>

(lldb) po machine.canEnterState(BeamIdleState)
false

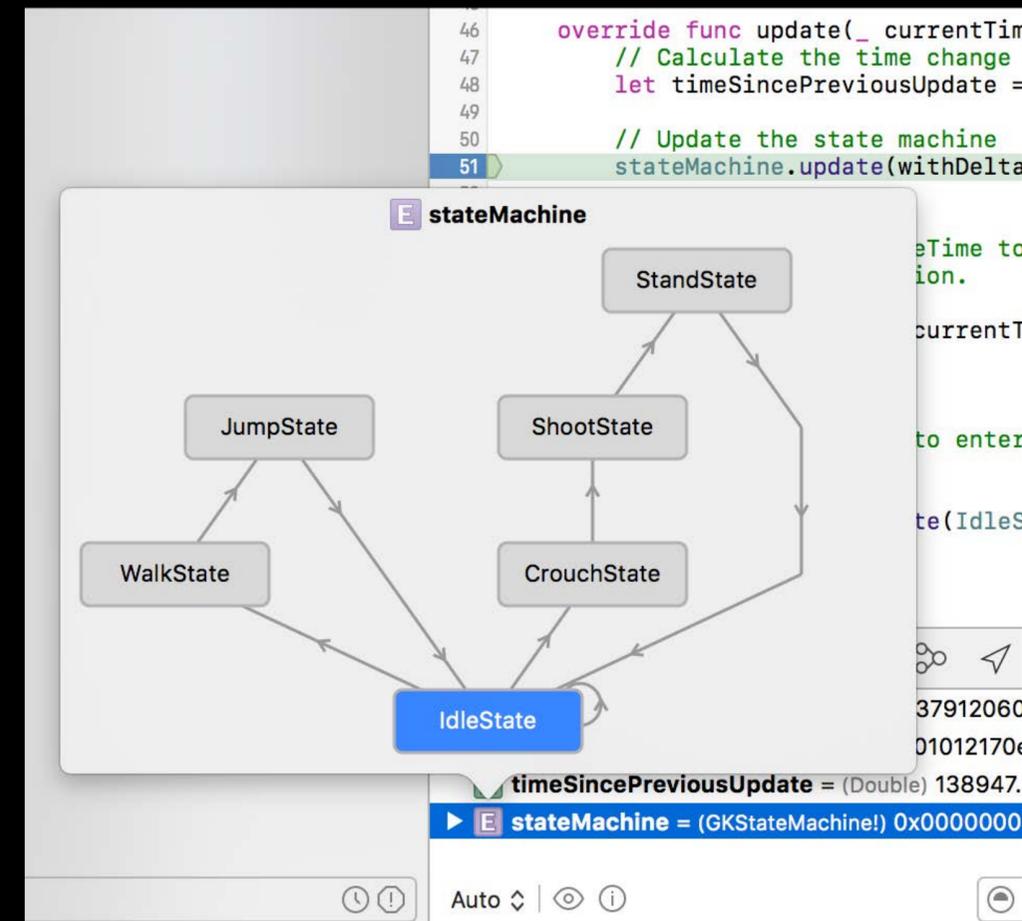
(lldb) po machine.canEnterState(BeamFiringState)
true

(lldb) po machine.canEnterState(BeamCoolingState)
false

(lldb) po machine.canEnterState(BeamDisabledState)
false

(lldb) po machine.canEnterState(BeamChargingState)
false
```

Xcode 7.3

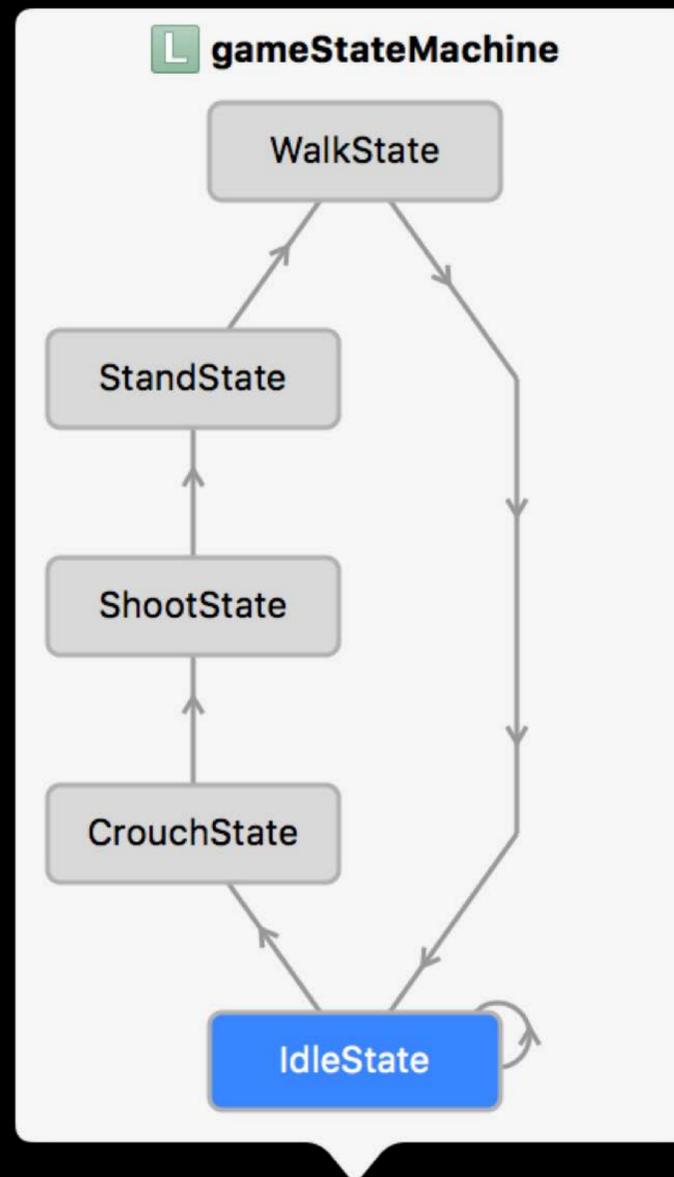


Xcode 8.0

# State Machine Quick Look

## Examples

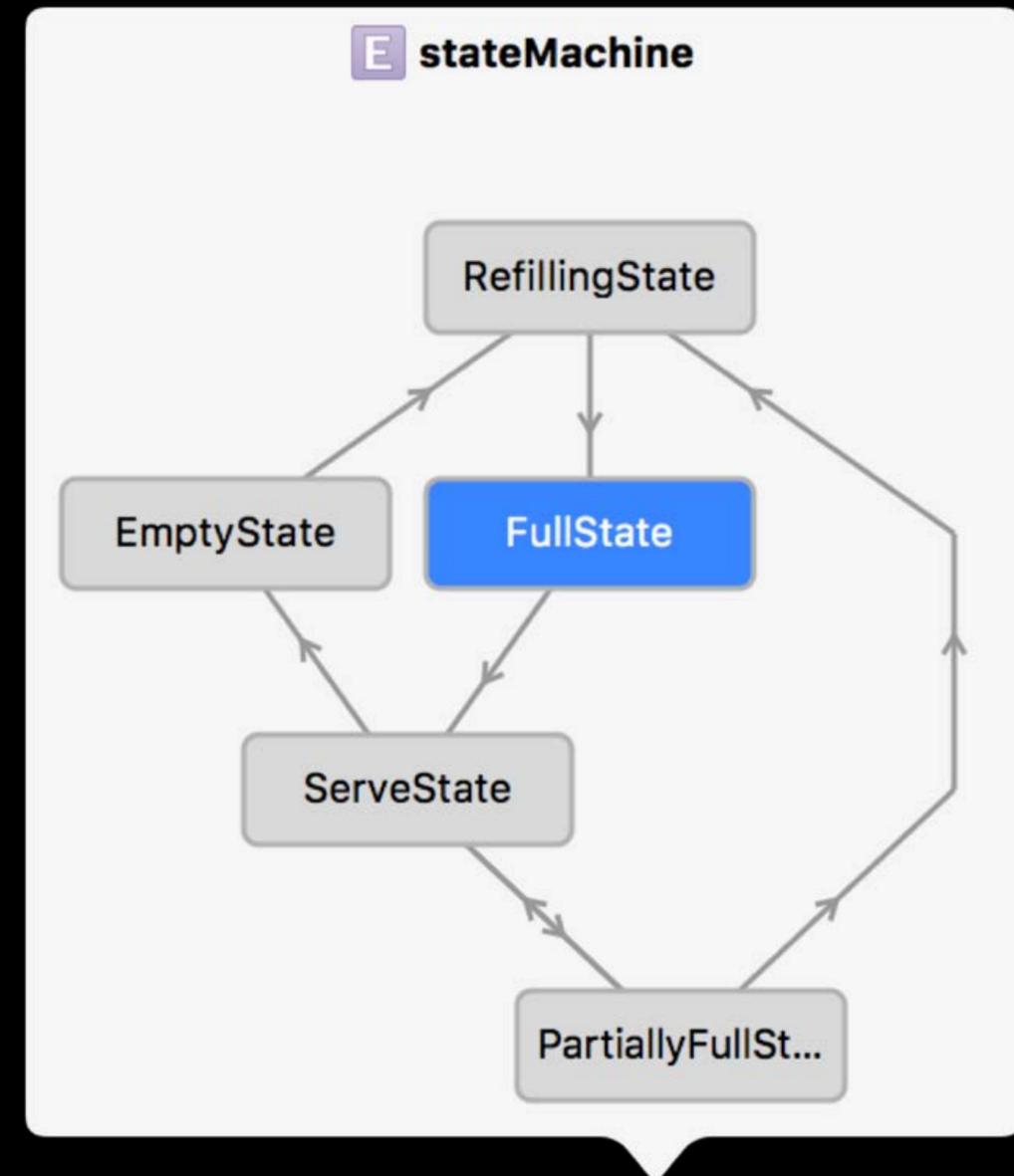
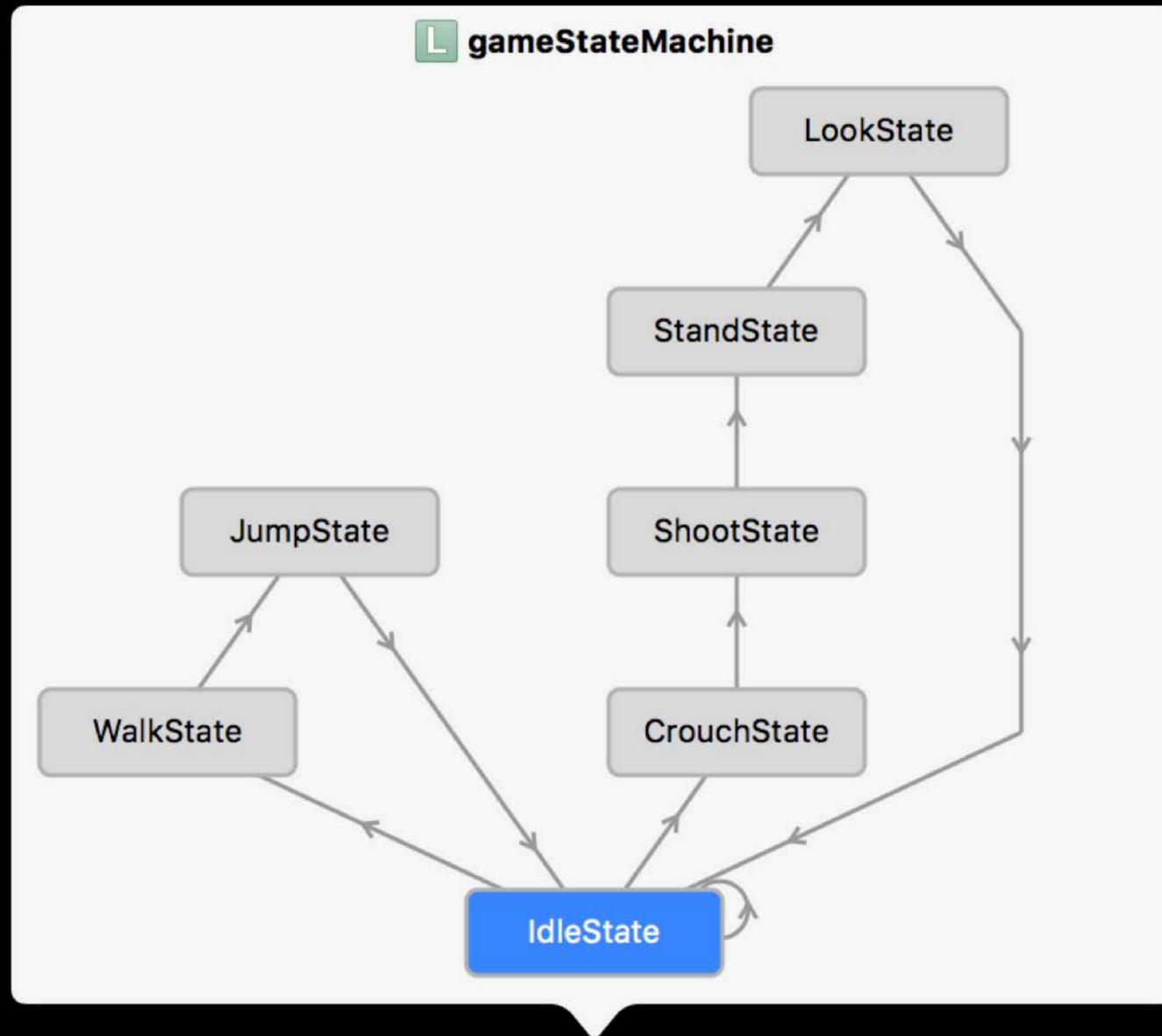
NEW



# State Machine Quick Look

NEW

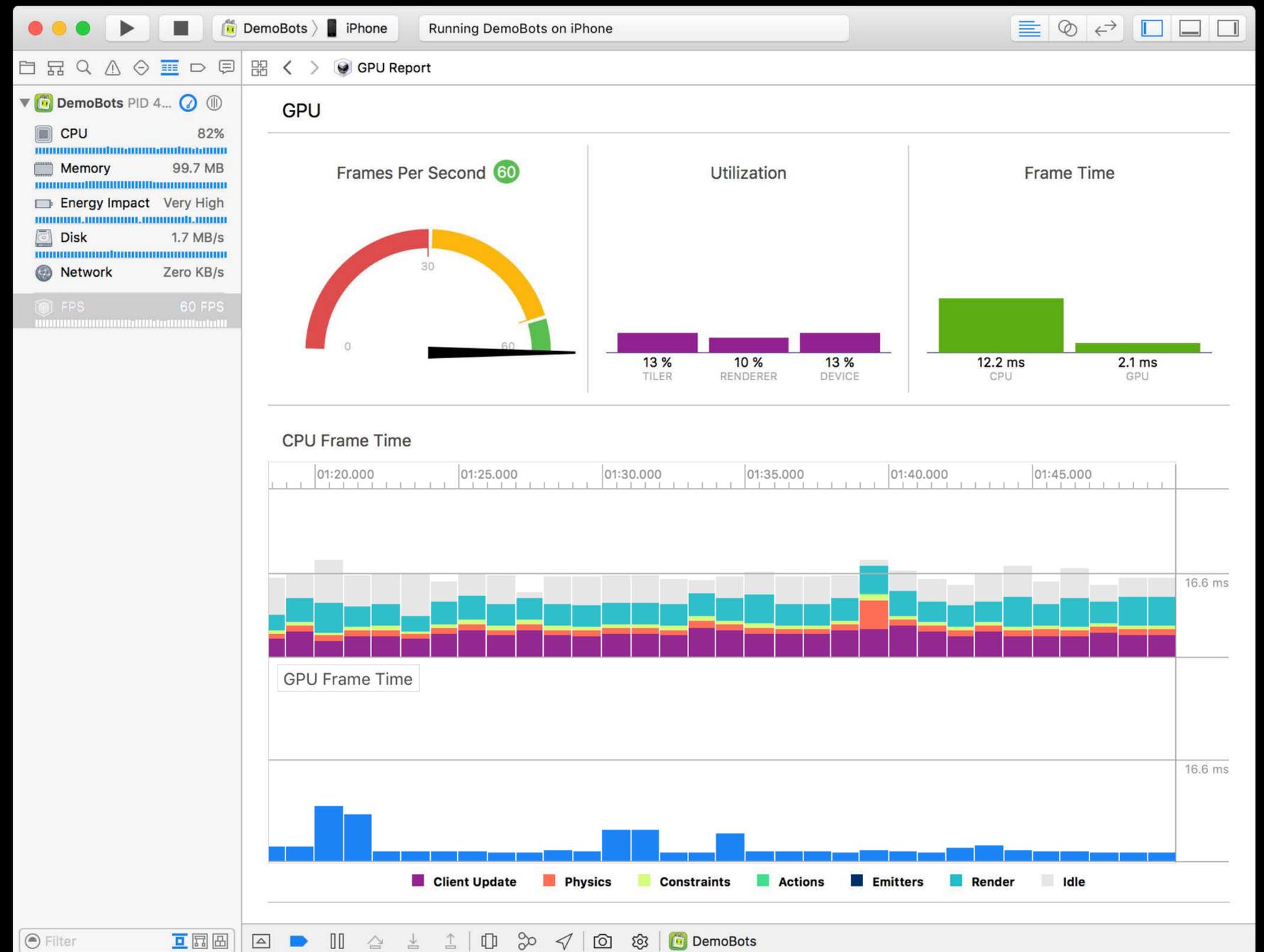
## Examples



# FPS Performance Gauge

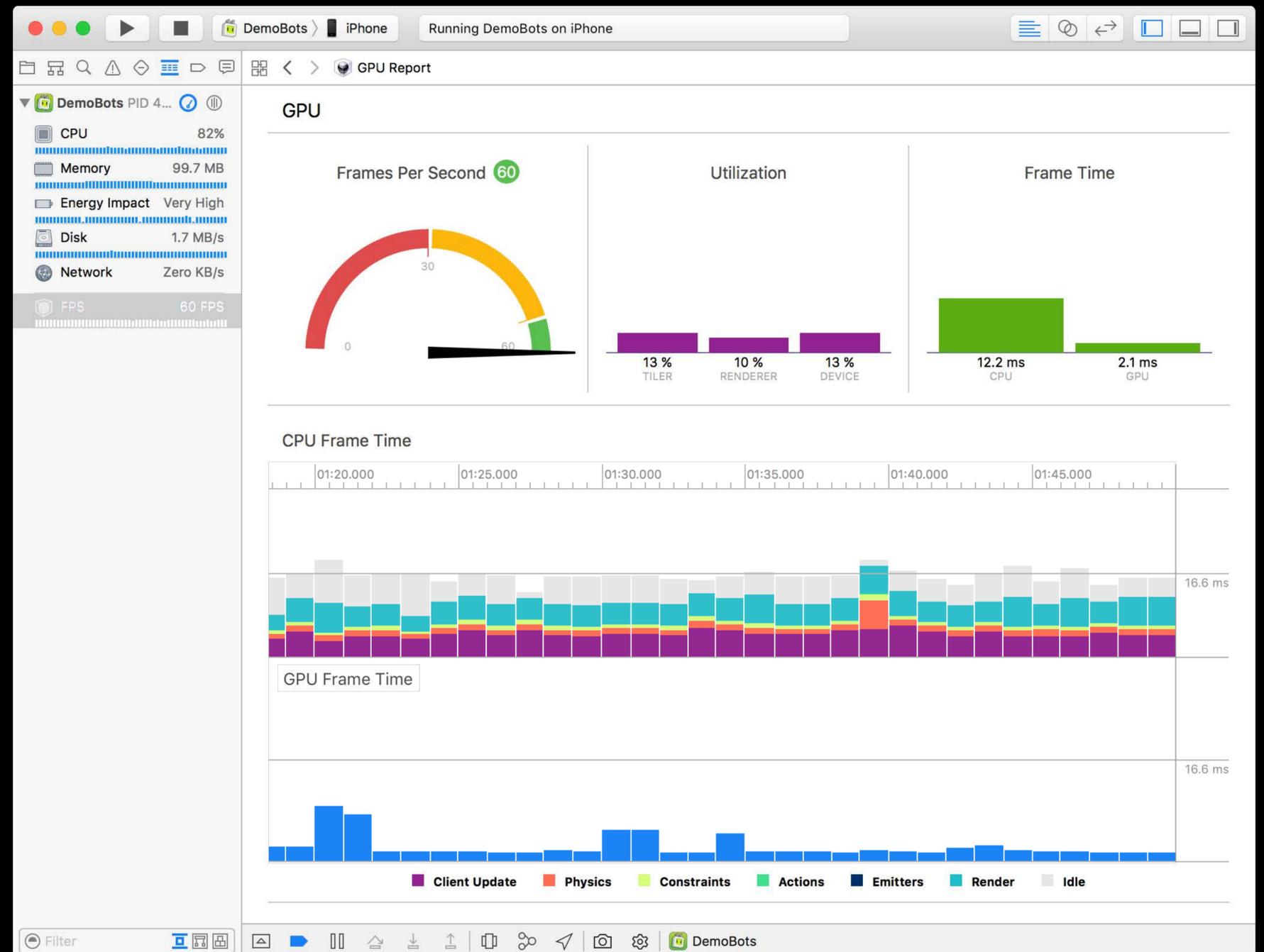
# FPS Performance Gauge

Real-time performance



# FPS Performance Gauge

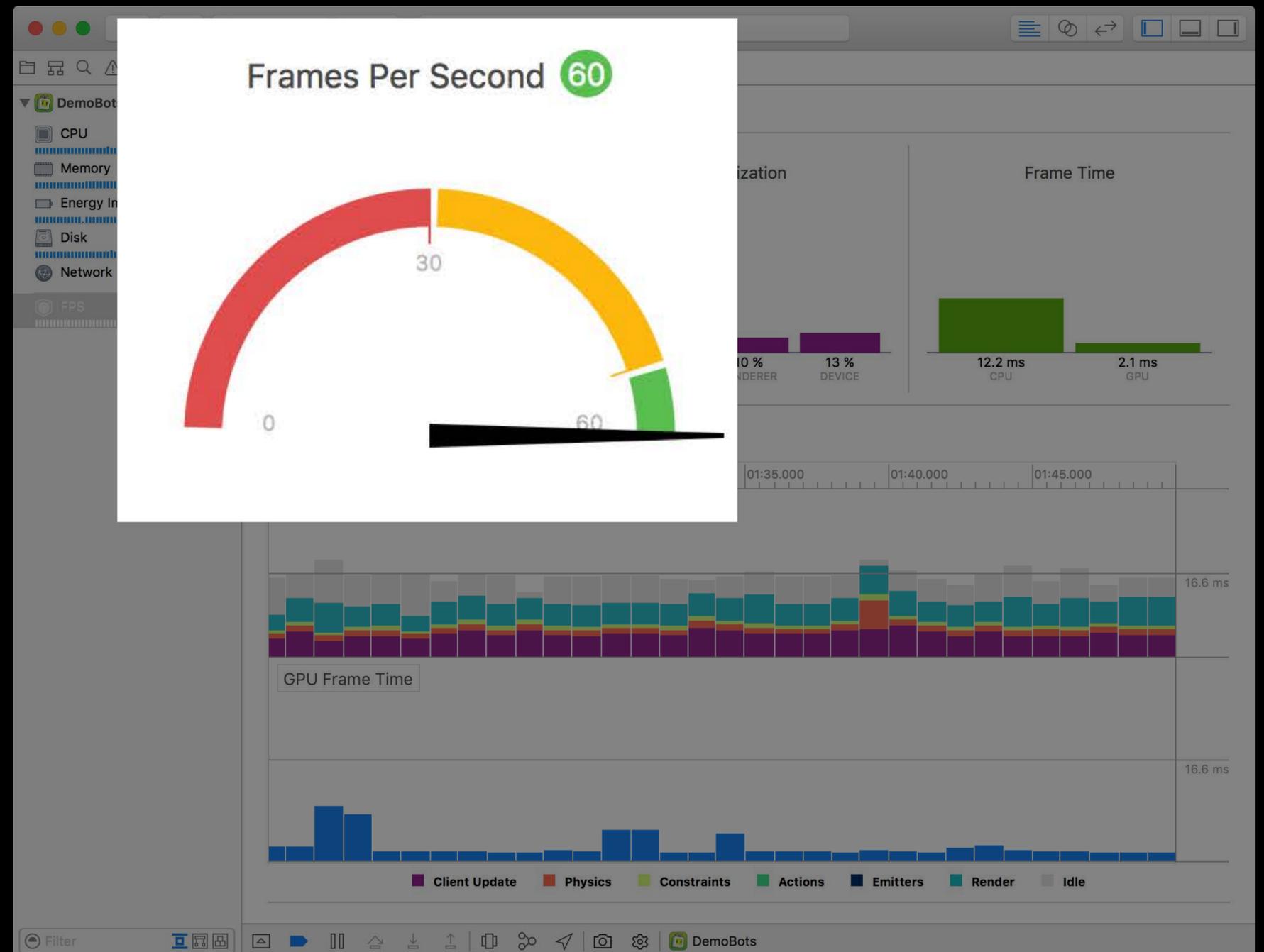
Real-time performance



# FPS Performance Gauge

Real-time performance

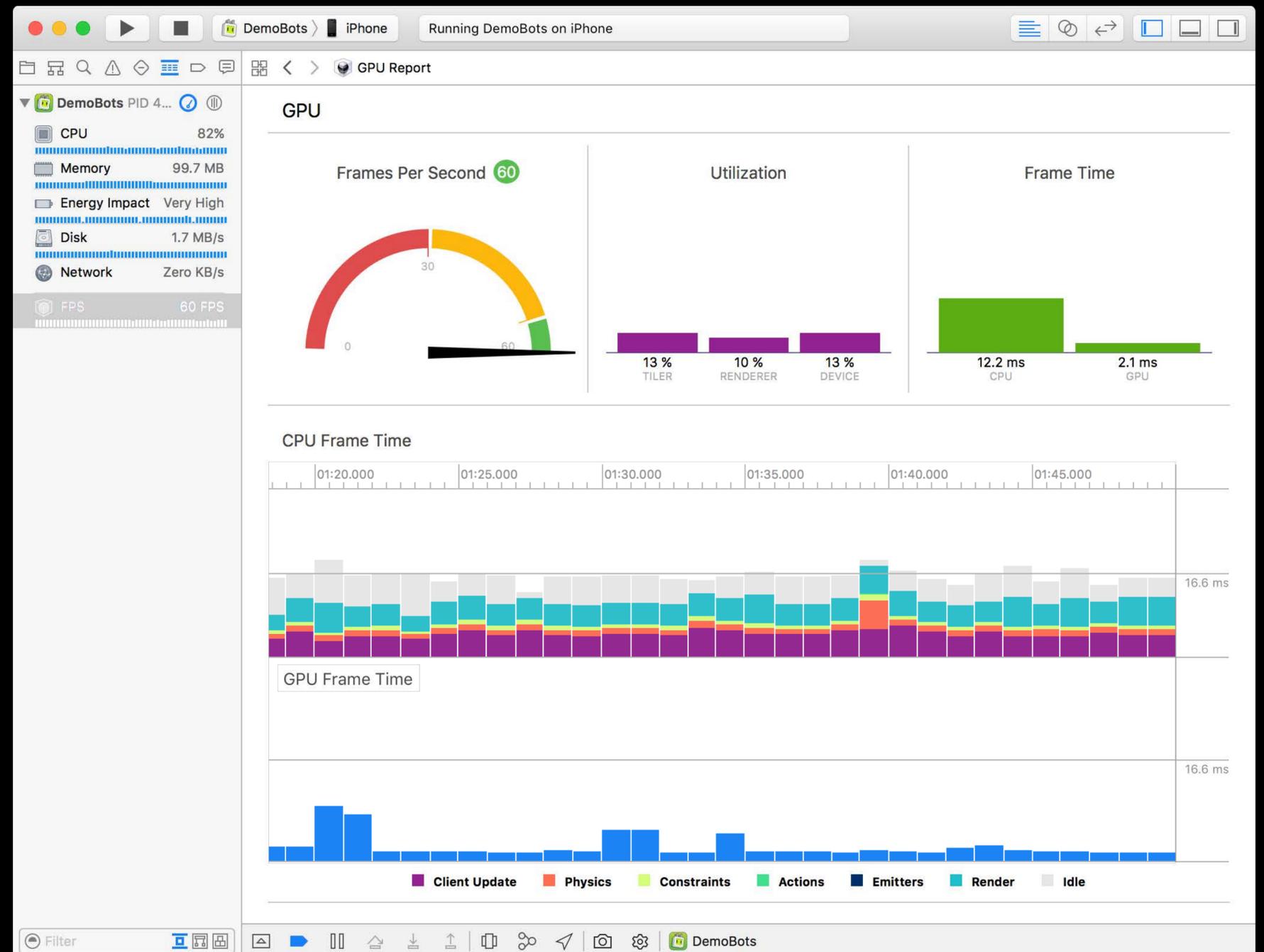
Frame rate



# FPS Performance Gauge

Real-time performance

Frame rate

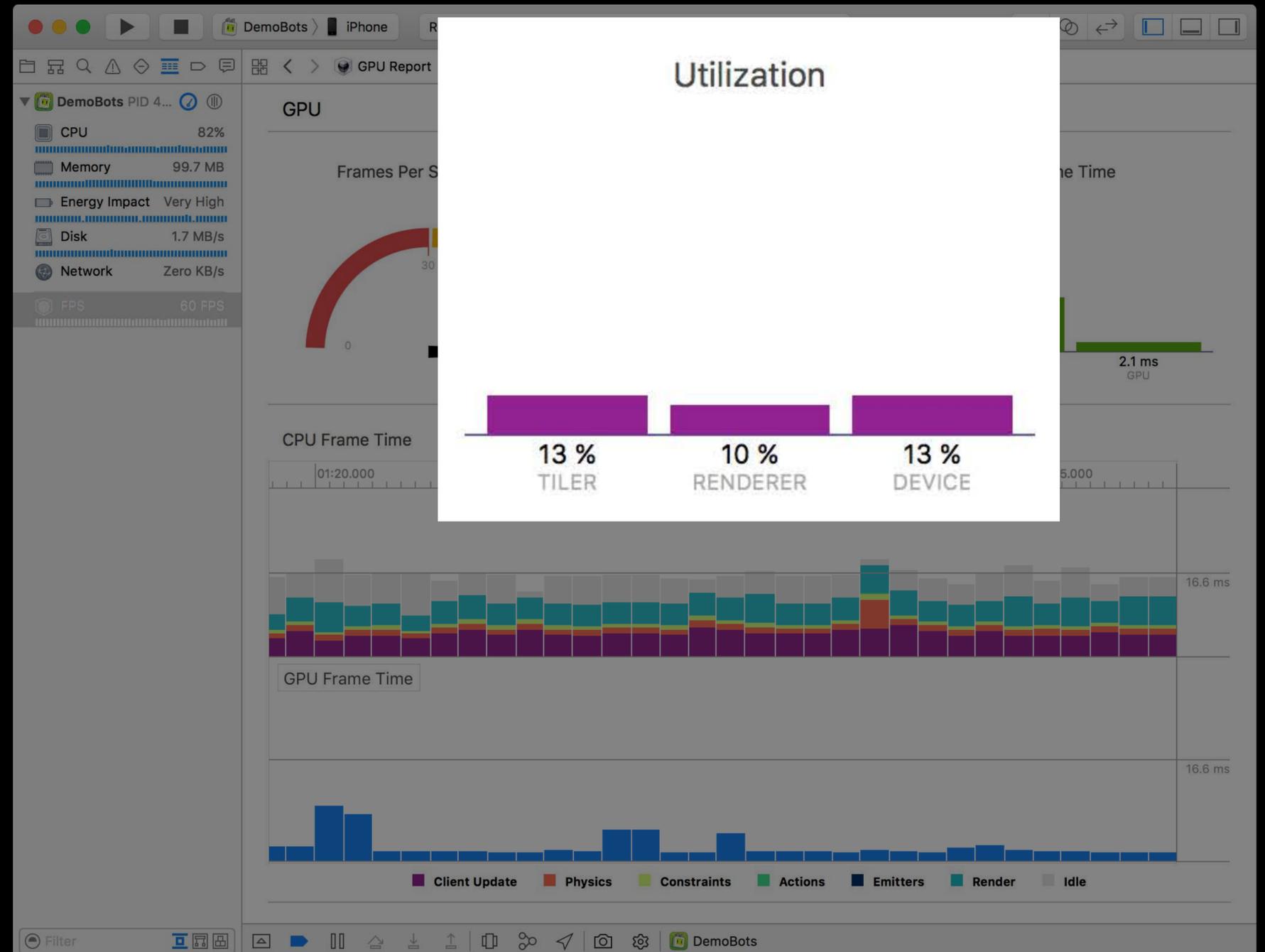


# FPS Performance Gauge

Real-time performance

Frame rate

GPU utilization

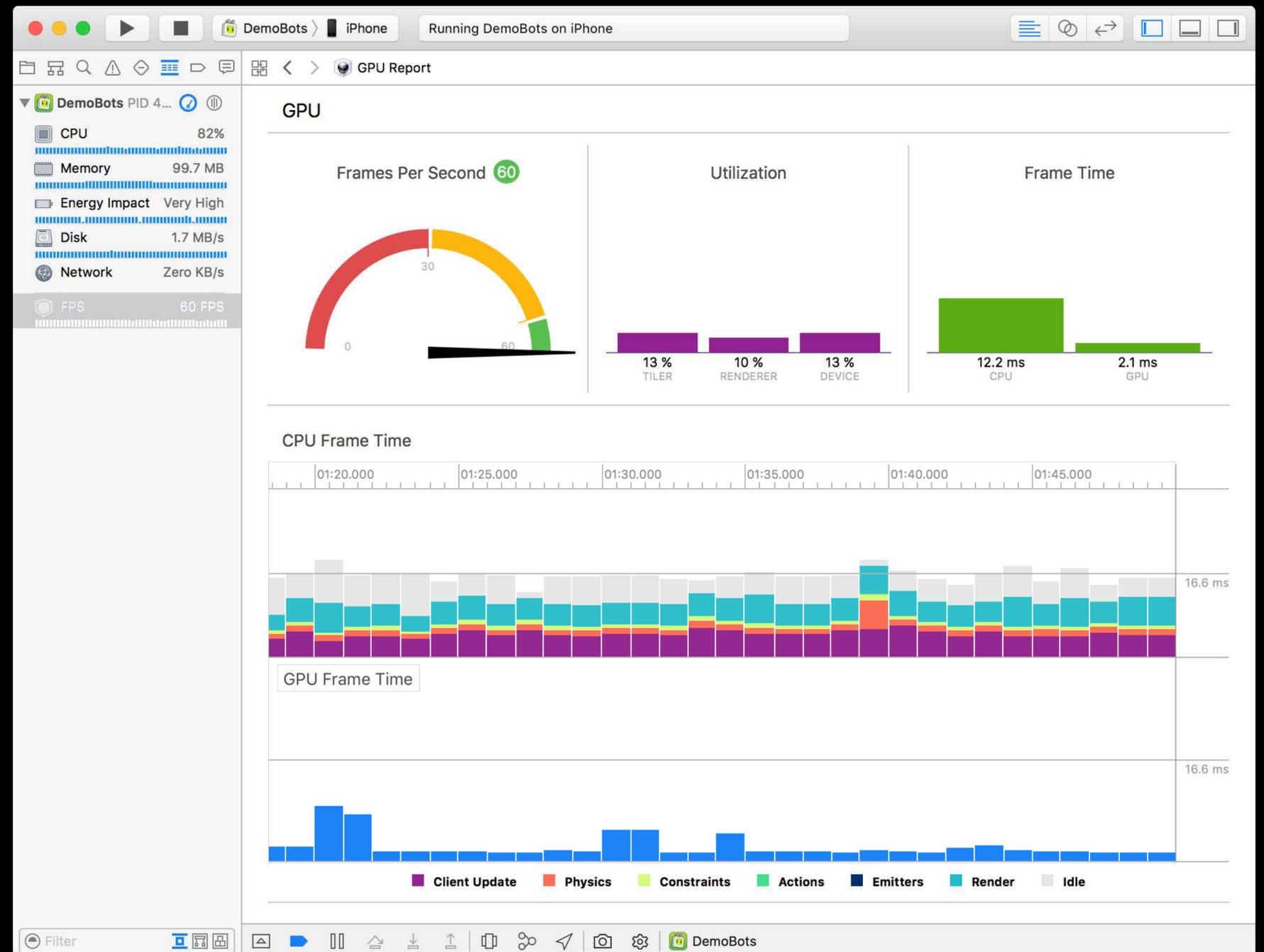


# FPS Performance Gauge

Real-time performance

Frame rate

GPU utilization



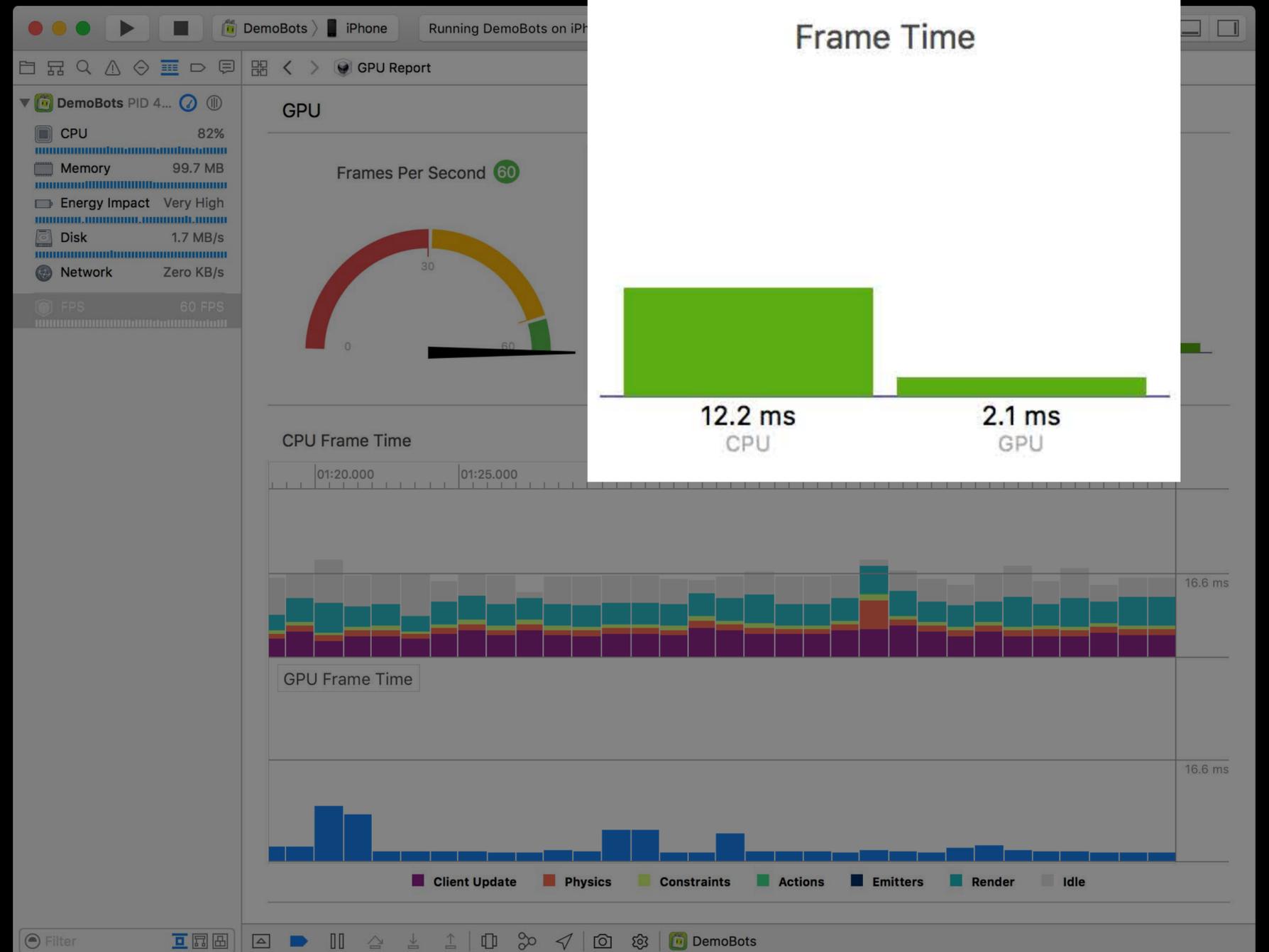
# FPS Performance Gauge

Real-time performance

Frame rate

GPU utilization

CPU / GPU frame time



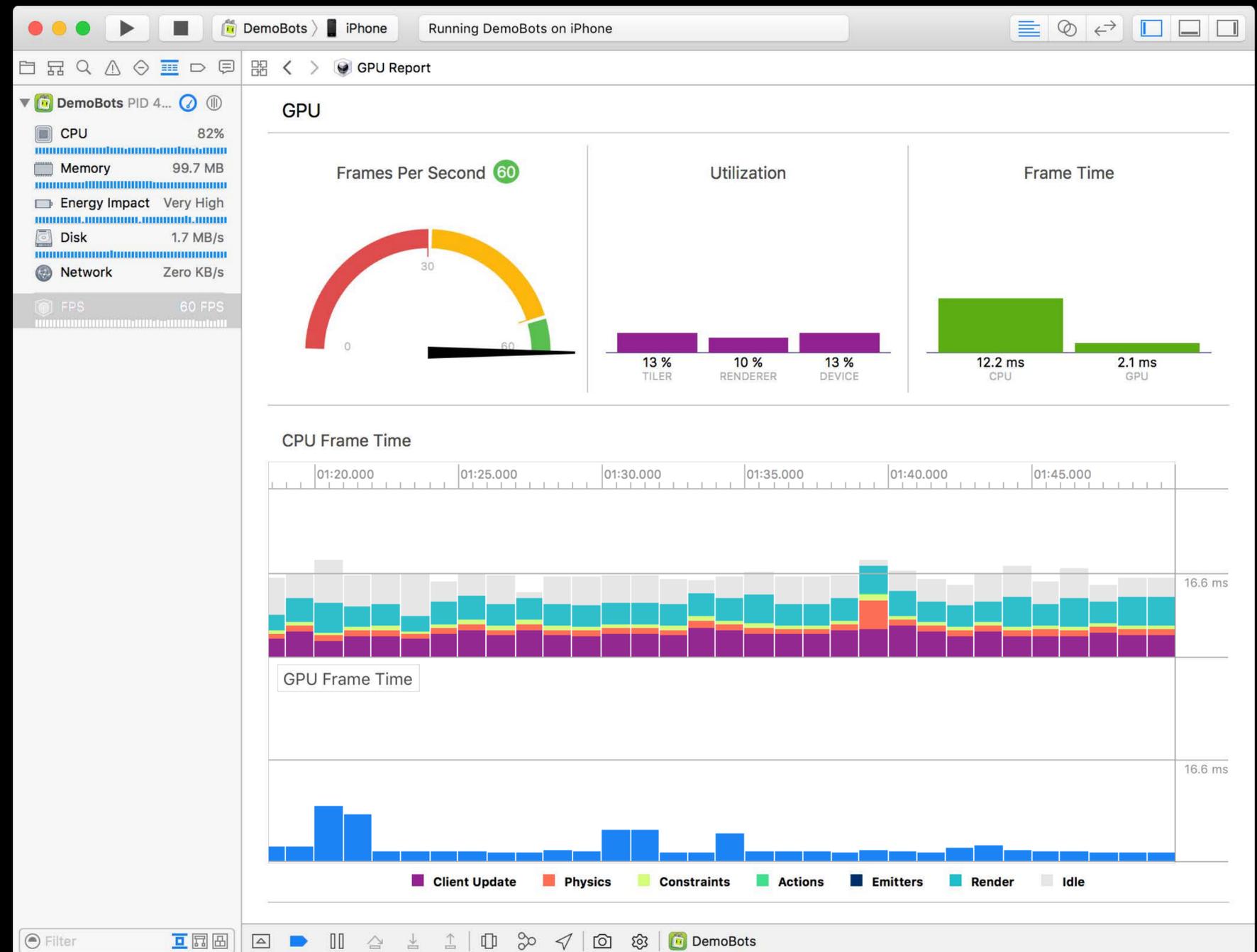
# FPS Performance Gauge

Real-time performance

Frame rate

GPU utilization

CPU / GPU frame time



# FPS Performance Gauge

NEW

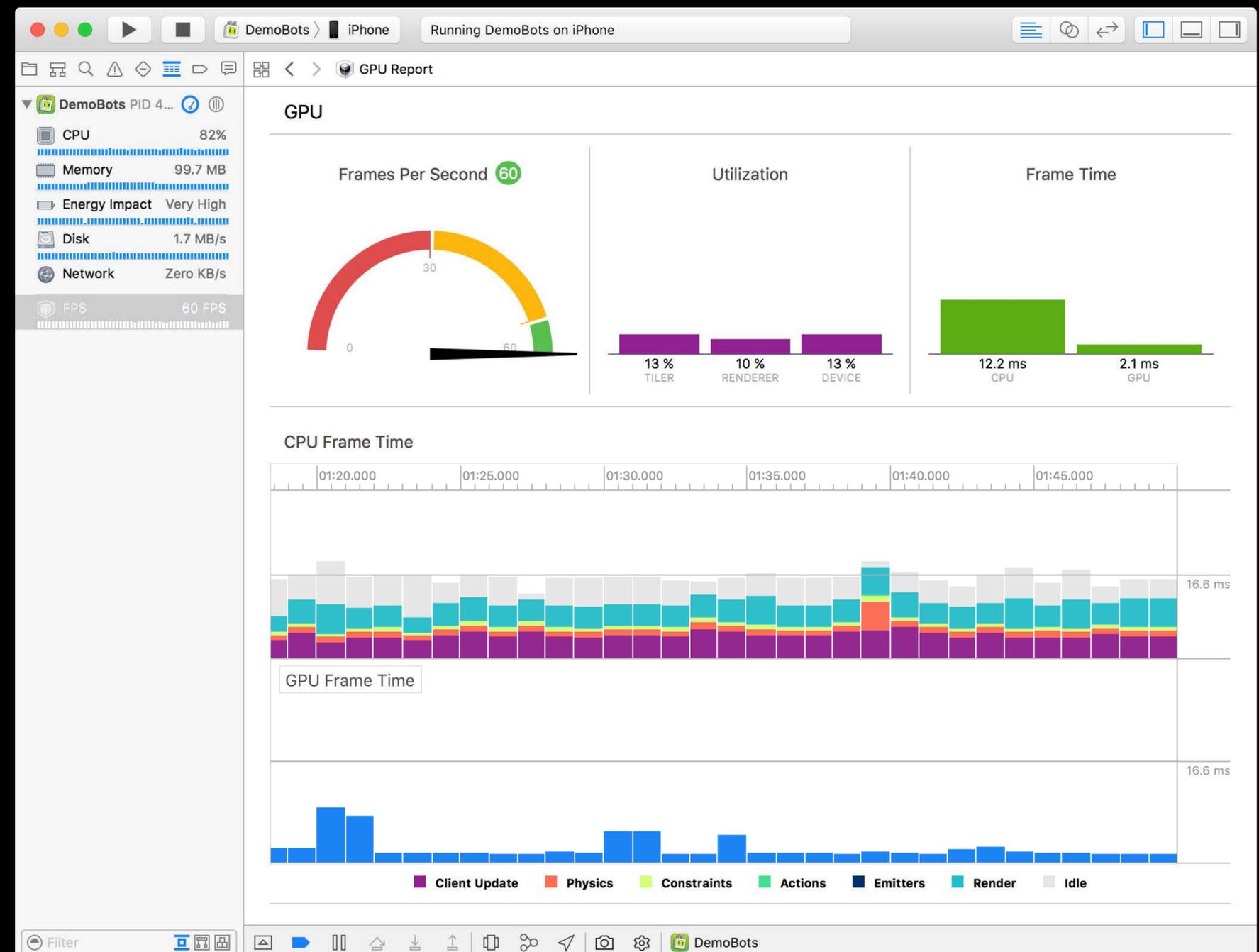
Real-time performance

Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# FPS Performance Gauge

NEW

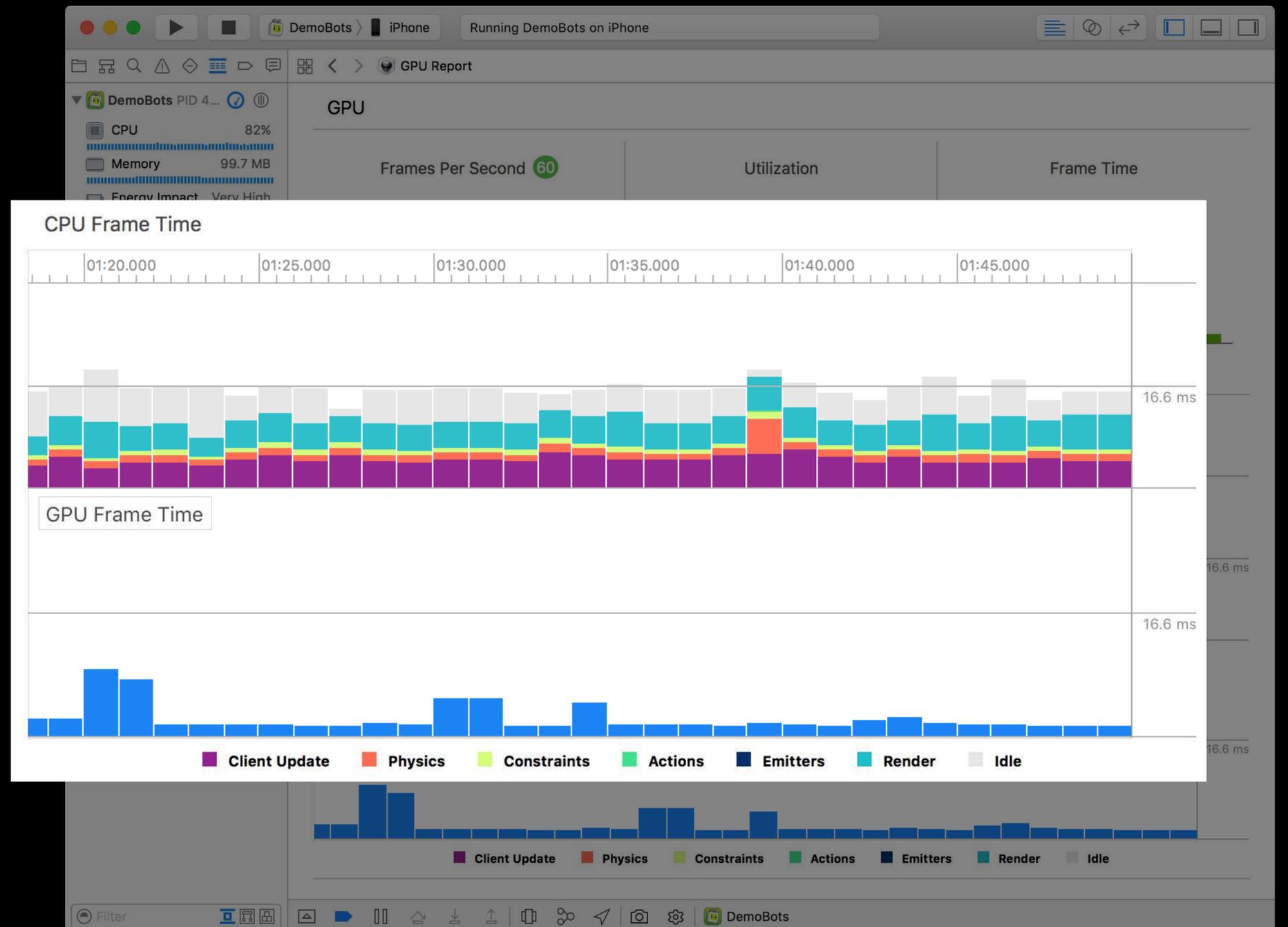
Real-time performance

Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# FPS Performance Gauge

Real-time performance

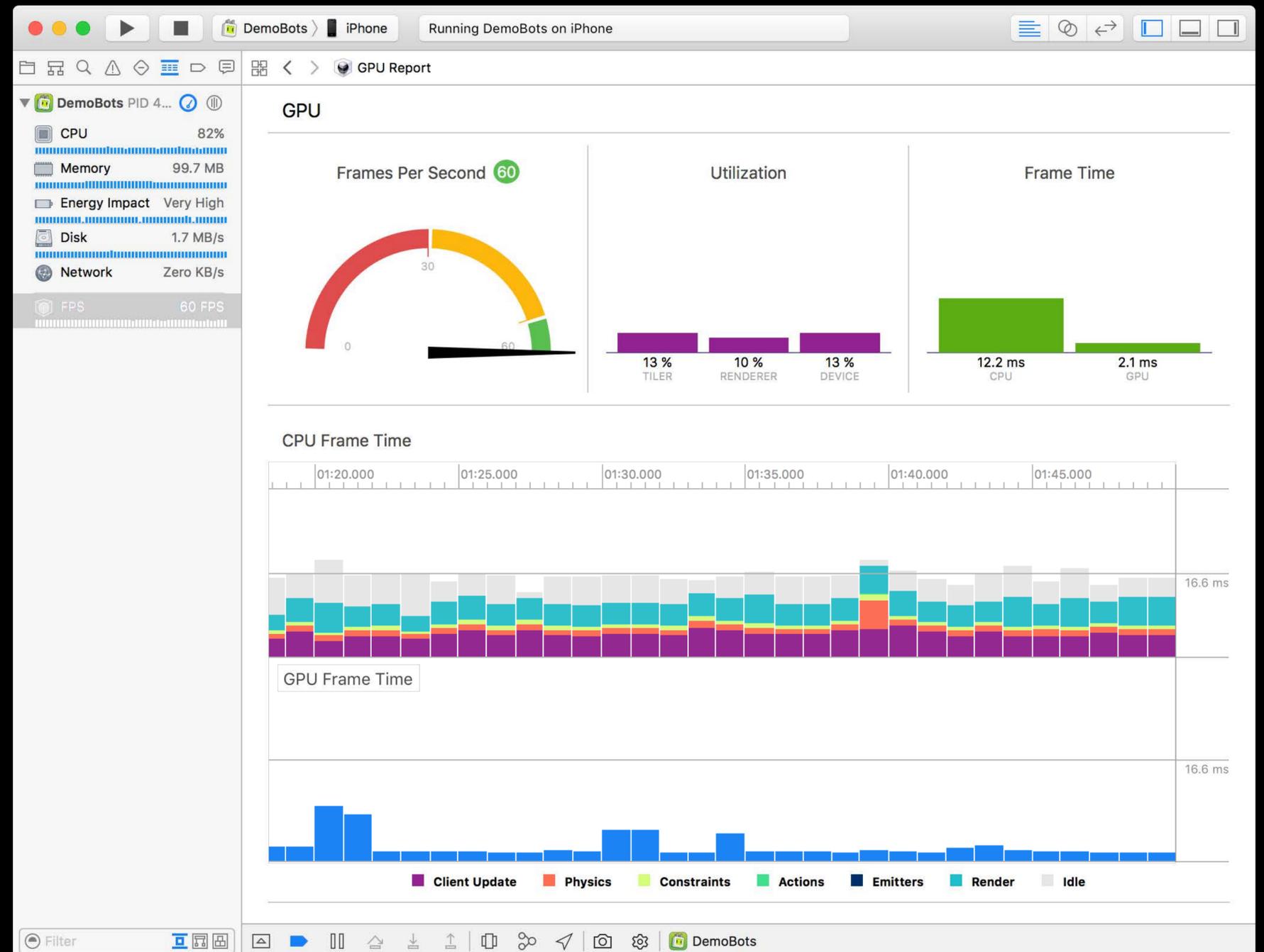
NEW

Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# FPS Performance Gauge

NEW

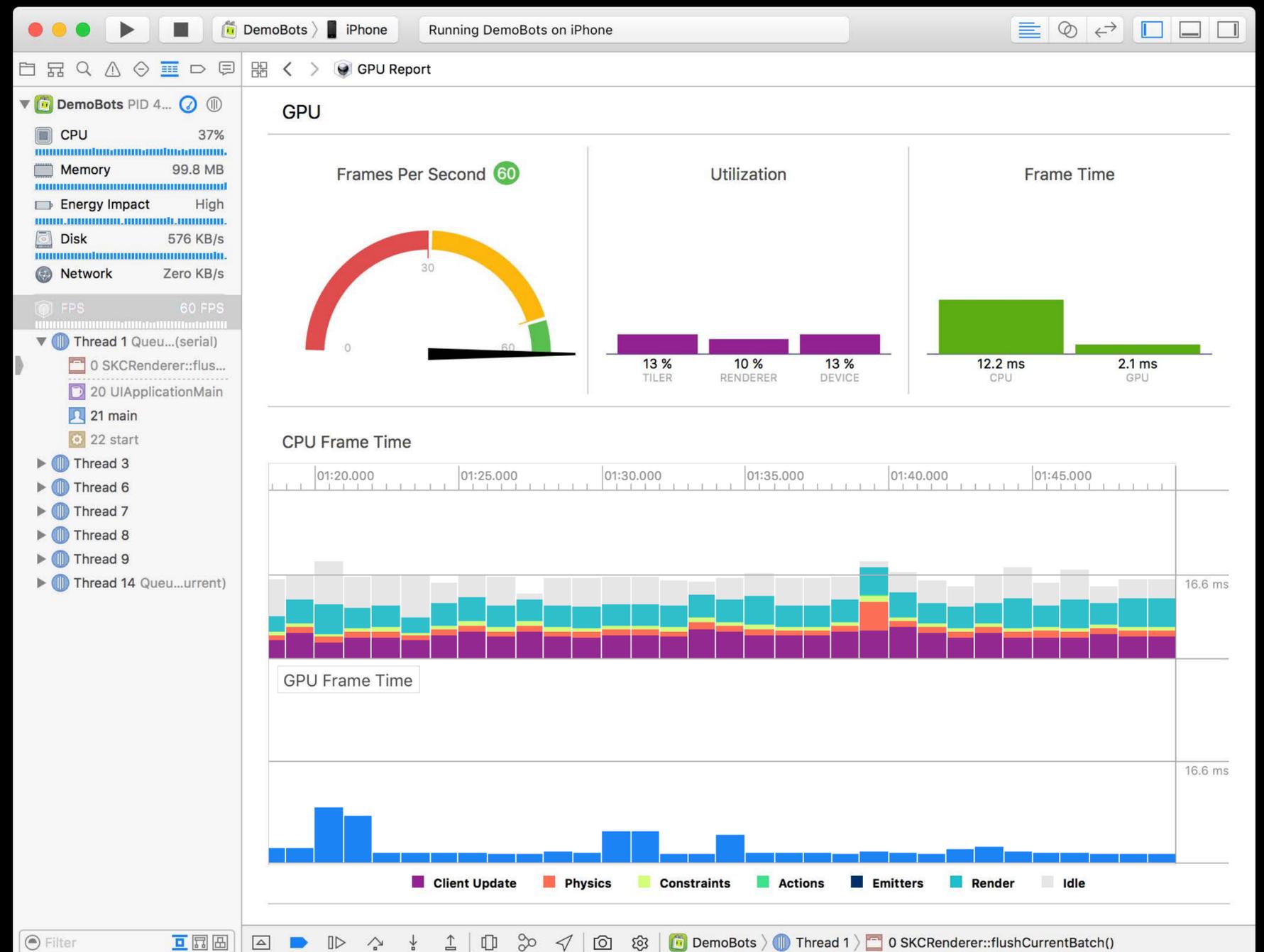
## Real-time performance

### Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# FPS Performance Gauge

NEW

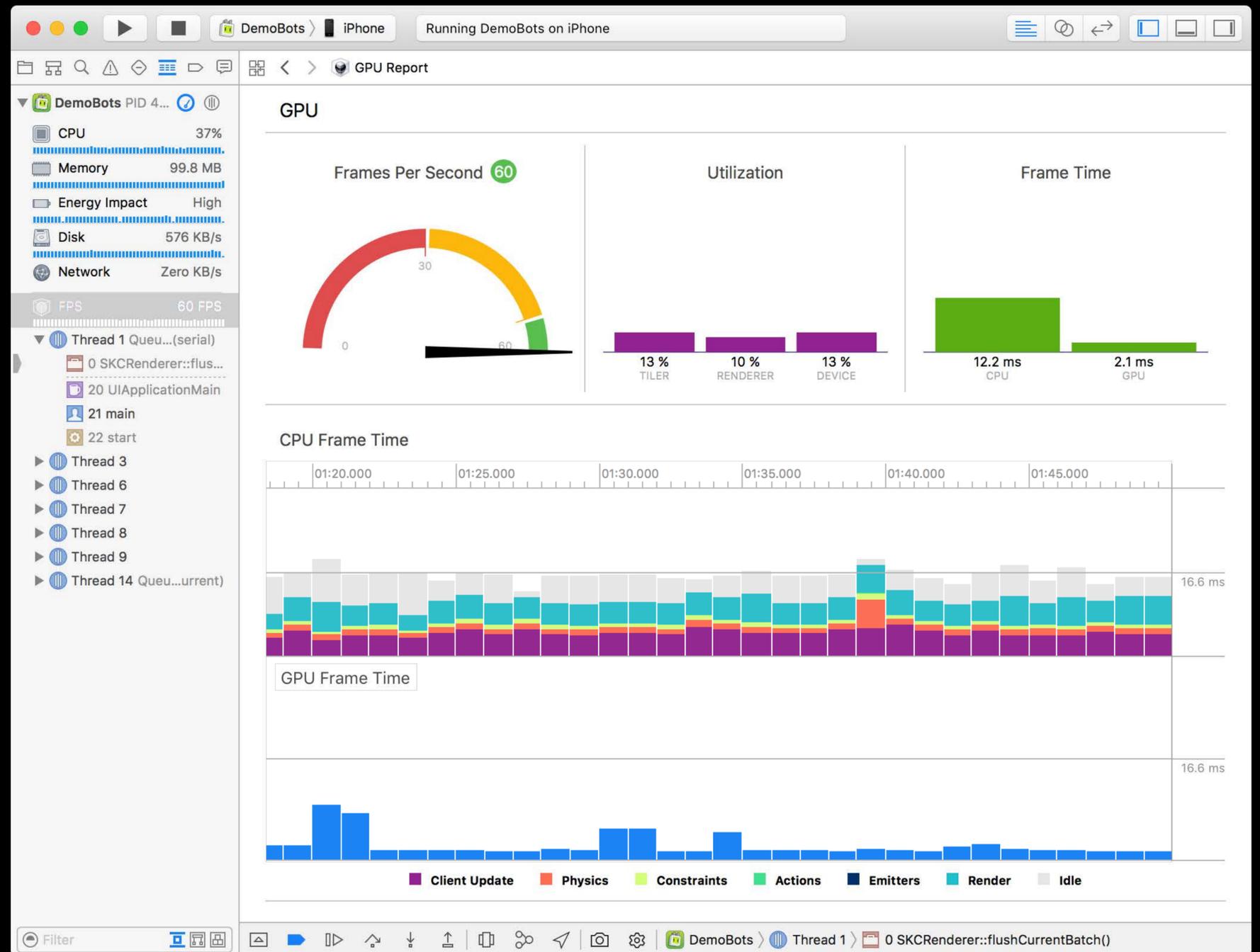
Real-time performance

Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# FPS Performance Gauge

Real-time performance

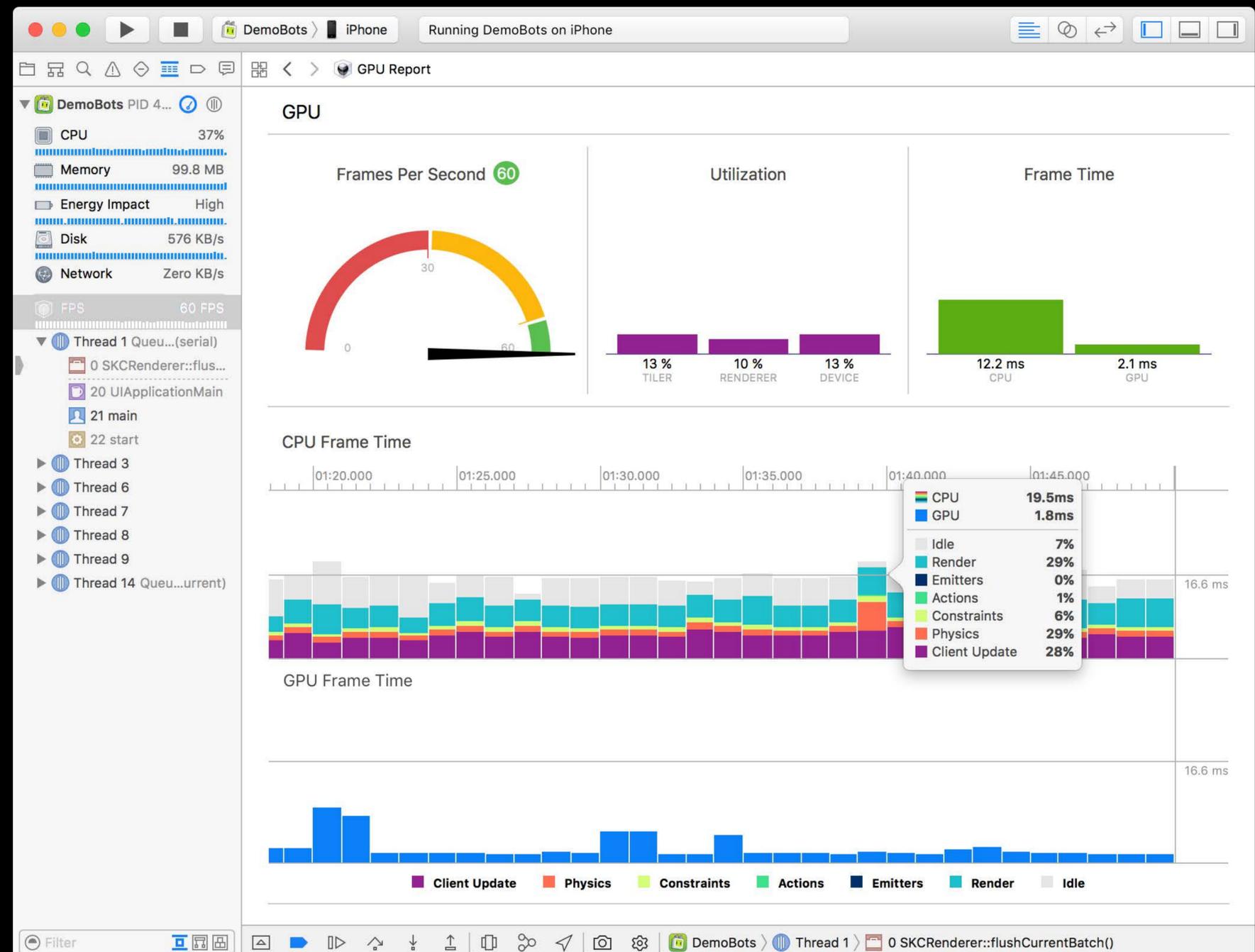
NEW

Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# FPS Performance Gauge

NEW

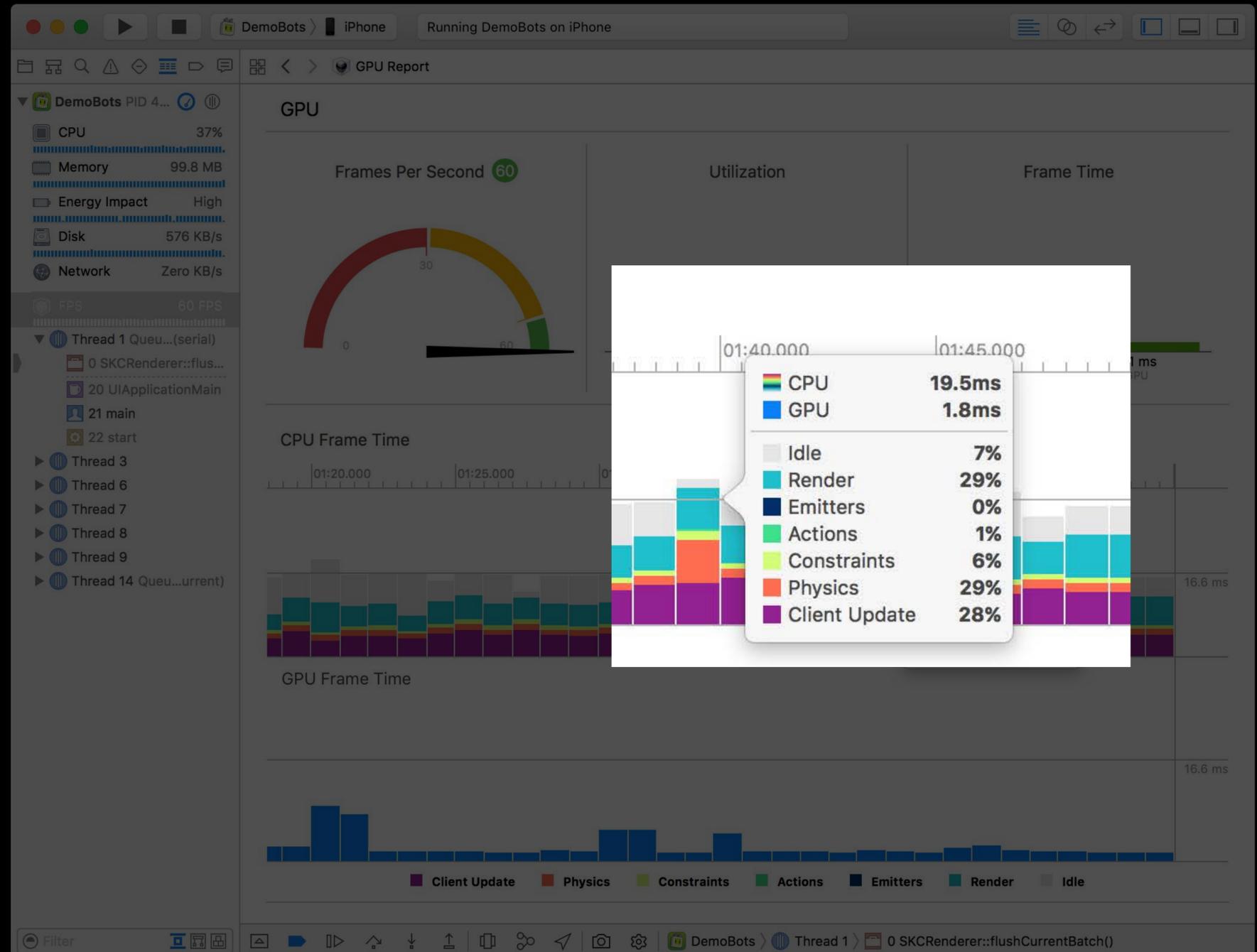
## Real-time performance

### Breakdown of update loop

- Render
- Client update
- Actions
- Physics

Easy to identify bottlenecks

Available on iOS and watchOS



# *Demo*

State machine Quick Look and FPS performance gauge

# Recap

State machine Quick Look

FPS performance gauge

# Memory Graph Debugging

Daniel Delwood Software Radiologist

*"Why does this object exist?"*

# Memory Graph Debugging

```
$ heap DemoBots [--addresses='.*Action']
```

```
Zone DefaultMallocZone_0x121016000: 61176 nodes (68571200 bytes)
```

COUNT	BYTES	AVG	CLASS_NAME	TYPE	BINARY
=====	=====	===	=====	=====	=====
22875	63015952	2754.8	non-object		
8898	429360	48.3	__NSCFString	ObjC	CoreFoundation
2402	576480	240.0	SKTexture	ObjC	SpriteKit
2183	209568	96.0	CUIRenditionKey	ObjC	CoreUI
1124	71936	64.0	NSConcreteData	ObjC	Foundation
1085	260400	240.0	CGImage	CFTYPE	CoreGraphics
1080	171504	158.8	CGImageProvider	CFTYPE	CoreGraphics
1075	172000	160.0	SKTextureCache	ObjC	SpriteKit
1070	256800	240.0	CGDataProvider	CFTYPE	CoreGraphics
1065	187440	176.0	CUIRenditionMetrics	ObjC	CoreUI
1059	84720	80.0	CUIRenditionLayerReference	ObjC	CoreUI
1053	1027728	976.0	_CUIInternalLinkRendition	ObjC	CoreUI
1052	70160	66.7	NSPathStore2	ObjC	Foundation
814	39072	48.0	NSMutableArray	ObjC	CoreFoundation
582	64240	110.4	NSArray (Object Storage)	C	CoreFoundation
470	22560	48.0	NSMutableDictionary	ObjC	CoreFoundation
451	7216	16.0	NSArray	ObjC	CoreFoundation
444	21312	48.0	Swift._NSContiguousString	Swift	libswiftCore.dylib
411	22048	53.6	__NSMallocBlock__	ObjC	CoreFoundation

# Memory Graph Debugging

```
$ leaks DemoBots --trace=0x7fc2e9e83c90
```

```
Region __DATA 0x110d6f000-0x110d75000[24K] rw-/rwx System/Library/Frameworks/UIKit.framework/UIKit
__DATA __bss: '_UIKeyWindow' 0x110d6f730 --> <UIWindow 0x7fc2e9c2ebd0> [816]
+584: __strong _rootViewController 0x7fc2e9c2ee18 --> <DemoBots.GameViewController 0x7fc2e9c1ddc0> [768]
+24: __strong _view 0x7fc2e9c1ddd8 --> <SKView 0x7fc2ec049a00> [1536]
+1144: __strong _scene 0x7fc2ec049e78 --> <DemoBots.LevelScene 0x7fc2f63daad0> [560]
+392: playerBot 0x7fc2f63dac58 --> <DemoBots.PlayerBot 0x7fc2e9d69600> [64]
+8: __strong _components 0x7fc2e9d69608 --> <NSMutableDictionary 0x7fc2e9d1d340> [48] item count: 10
+40: __strong _keys 0x7fc2e9d1d368 --> <NSDictionary (Key Storage) 0x7fc2e9df0820> [208]
+112: __strong 0x7fc2e9df0890 --> <DemoBots.RenderComponent 0x7fc2f4071120> [48]
+40: node 0x7fc2f4071148 --> <SKNode 0x7fc2ef712970> [128]
+40: __strong _actions 0x7fc2ef712998 --> <NSMutableArray 0x7fc2f640ec00> [48] item count: 23
+40: __strong _list 0x7fc2f640ec28 --> <NSArray (Object Storage) 0x7fc2e9ed8150> [208]
+120: __strong 0x7fc2e9ed81c8 --> <SKRepeat 0x7fc2e9ed0db0> [32]
+24: __strong _repeatedAction 0x7fc2e9ed0dc8 --> <SKSequence 0x7fc2e9e9eb40> [48]
+24: __strong _actions 0x7fc2e9e9eb58 --> <NSArray 0x7fc2f51a3ee0> [32] item count: 2
+24: __strong 0x7fc2f51a3ef8 --> <SKRunBlock 0x7fc2e9e84760> [32]
+8: _caction 0x7fc2e9e84768 --> <SKCAction 0x7fc2e9e83c90> [112]

Region __DATA 0x1162e0000-0x1162e6000[24K] rw-/rwx /System/Library/PrivateFrameworks/FrontBoardServices.framework/FrontBoardServices
__DATA __bss: '__instance' 0x1162e50e8 --> <FBSUIApplicationWorkspace 0x7fc2e9e0c9e0> [80]
+16: _delegate 0x7fc2e9e0c9f0 --> <UIApplication 0x7fc2e9d12290> [576]
+112: __strong _statusBarWindow 0x7fc2e9d12300 --> <UIStatusBarWindow 0x7fc2e9c13ec0> [912]
+416: __strong _scene 0x7fc2e9c14060 --> <FBSSceneImpl 0x7fc2e9f12990> [144]
```

# Memory Graph Debugging

```
$ malloc_history DemoBots 0x7fc2e9e83c90
```

```
ALLOC 0x7fc2e9e83c90-0x7fc2e9e83cff [size=112]: thread_112f48000 | start | main | UIApplicationMain | GSEventRunModal |  
CFRunLoopRunSpecific | __CFRunLoopRun | __CFRunLoopDoTimers | __CFRunLoopDoTimer |  
__CFRUNLOOP_IS_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ | CA::Display::DisplayLink::dispatch_items(unsigned long long, unsigned long  
long, unsigned long long) | CA::Display::DisplayLinkItem::dispatch() | -[SKDisplayLink _callbackForNextFrame:] | __29-[SKView  
setUpRenderCallback]_block_invoke | -[SKView _vsyncRenderForTime:preRender:postRender:] | __51-[SKView  
_vsyncRenderForTime:preRender:postRender:]_block_invoke.312 | -[SKView _update:] | -[SKScene _update:] | @objc  
DemoBots.LevelScene.update (Swift.Double) -> () | DemoBots.LevelScene.update (Swift.Double) -> () | DemoBots.PlayerBot.update  
(withDeltaTime : Swift.Double) -> () | -[SKNode runAction:] | -[SKRepeat copyWithZone:] | +[SKRepeat repeatActionForever:] | -  
[SKSequence copyWithZone:] | +[SKSequence sequenceWithActions:] | -[SKAction copyWithZone:] | -[SKWait init] | -[SKAction init] |  
operator new(unsigned long) | malloc
```

```
----  
FREE 0x7fc2e9e83c90-0x7fc2e9e83cff [size=112]: thread_112f48000 | start | main | UIApplicationMain | GSEventRunModal |  
CFRunLoopRunSpecific | __CFRunLoopRun | __CFRunLoopDoTimers | __CFRunLoopDoTimer |  
__CFRUNLOOP_IS_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ | CA::Display::DisplayLink::dispatch_items(unsigned long long, unsigned long  
long, unsigned long long) | CA::Display::DisplayLinkItem::dispatch() | -[SKDisplayLink _callbackForNextFrame:] | __29-[SKView  
setUpRenderCallback]_block_invoke | -[SKView _vsyncRenderForTime:preRender:postRender:] | __51-[SKView  
_vsyncRenderForTime:preRender:postRender:]_block_invoke.312 | -[SKView _update:] | -[SKScene _update:] | @objc  
DemoBots.LevelScene.update (Swift.Double) -> () | DemoBots.LevelScene.update (Swift.Double) -> () | DemoBots.PlayerBot.update  
(withDeltaTime : Swift.Double) -> () | -[SKNode runAction:] | -[SKRepeat copyWithZone:] | +[SKRepeat repeatActionForever:] | -  
[SKSequence copyWithZone:] | +[SKSequence sequenceWithActions:] | -[SKAction copyWithZone:] | -[SKWait init] | -[SKAction(Intern  
al)  
setCppAction:] | operator delete(void*)
```

```
ALLOC 0x7fc2e9e83c90-0x7fc2e9e83cff [size=112]: thread_112f48000 | start | main | UIApplicationMain | GSEventRunModal |  
CFRunLoopRunSpecific | __CFRunLoopRun | __CFRunLoopDoTimers | __CFRunLoopDoTimer |  
__CFRUNLOOP_IS_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ | CA::Display::DisplayLink::dispatch_items(unsigned long long, unsigned long  
long, unsigned long long) | CA::Display::DisplayLinkItem::dispatch() | -[SKDisplayLink _callbackForNextFrame:] | __29-[SKView  
setUpRenderCallback]_block_invoke | -[SKView _vsyncRenderForTime:preRender:postRender:] | __51-[SKView  
_vsyncRenderForTime:preRender:postRender:]_block_invoke.312 | -[SKView _update:] | -[SKScene _update:] | @objc  
DemoBots.LevelScene.update (Swift.Double) -> () | DemoBots.LevelScene.update (Swift.Double) -> () | DemoBots.PlayerBot.update  
(withDeltaTime : Swift.Double) -> () | -[SKNode runAction:] | -[SKRepeat copyWithZone:] | +[SKRepeat repeatActionForever:] | -  
[SKSequence copyWithZone:] | +[SKSequence sequenceWithActions:] | -[SKRunBlock copyWithZone:] | +[SKRunBlock runBlock:queue:] | -  
[SKRunBlock init] | -[SKAction init] | operator new(unsigned long) | malloc
```

# Memory Graph Debugging

NEW

The screenshot displays the Xcode Memory Graph Debugger interface. The main window shows a memory graph for a `PlayerBot` object. The graph consists of several nodes and their connections:

- `LevelScene` points to `PlayerBot`.
- `EntitySnapshot` points to `PlayerBot`.
- `ChargeComponent` points to `PlayerBot`.
- `EntitySnapshot` points to `PlayerBot`.
- `BadTaskBo...geHighRule` points to `PlayerBot`.
- `malloc(144)` points to `PlayerBot`.
- `_NativeSet...<GKEntity>` points to `PlayerBot`.
- `_NativeSet...<GKEntity>` points to `PlayerBot`.
- `SKCRepeat` points to `SKCAnimate`, which points to `malloc(144)`.
- `malloc(64)` points to `malloc(64)`.
- `malloc(64)` points to `BadTaskBo...geHighRule`.
- `NSArray (Object Storage)` points to `ChargeComponent`.
- `NSMutableArray` points to `NSArray (Object Storage)`.
- `GKComponentSystem` points to `NSMutableArray`.
- `EntitySnapshot` points to `ChargeComponent`.
- `EntitySnapshot` points to `ChargeComponent`.
- `PlayerBotNearRule` points to `EntitySnapshot`.
- `NSArray (Object Storage)` points to `PlayerBotNearRule`.
- `NSArray (Object Storage)` points to `PlayerBotNearRule`.

The right-hand pane shows the `Memory Details` for the selected `PlayerBot` object:

- Class: `PlayerBot`
- Kind: `Swift`
- Address: `0x110abcab0`
- Size: `64 bytes`

The `Hierarchy` pane shows the object's parentage:

- `PlayerBot`
- `GKEntity`
- `NSObject`

The `Backtrace` pane shows the stack of function calls that led to the object's creation:

- 0 `malloc_zone_malloc`
- 1 `calloc`
- 2 `class_createInstance`
- 3 `+[NSObject allocWithZone:]`
- 4 `DemoBots.PlayerBot._allocating_i...`
- 5 `DemoBots.LevelScene.init (coder :...`
- 6 `@objc DemoBots.LevelScene.init (...`
- 7 `_decodeObjectBinary`
- 8 `_decodeObject`
- 9 `+[SKNode nodeWithFileNamed:]`
- 10 `@nonobjc __ObjC.SKScene._allo...`
- 11 `DemoBots.LoadSceneOperation.s...`
- 12 `@objc DemoBots.LoadSceneOper...`
- 13 `_NSOQSchedule_f`
- 14 `_dispatch_client_callout`
- 15 `_dispatch_queue_serial_drain`
- 16 `_dispatch_queue_invoke`
- 17 `_dispatch_root_queue_drain`
- 18 `_dispatch_worker_thread3`
- 19 `_pthread_wqthread`
- 20 `start_wqthread`

The left-hand pane shows the `DemoBots` process with various system metrics (CPU, Memory, Energy Impact, Disk, Network, FPS) and a list of objects in memory, including `PlayerBot` objects and their states.

# Memory Graph Debugging

NEW

The screenshot displays the Xcode Memory Graph Debugger interface. On the left, the 'DemoBots' hierarchy is shown with the 'PlayerBot' object selected. The main area shows a memory graph for the selected object, with a 'PlayerBot' object highlighted. A preview window shows a yellow robot character. On the right, the 'Memory Details' panel shows the class 'PlayerBot', kind 'Swift', address '0x110abcab0', and size '64 bytes'. The 'Hierarchy' panel shows the object's parent chain: 'PlayerBot', 'GKEntity', and 'NSObject'. The 'Backtrace' panel shows the stack of function calls leading to the object's creation.

**Memory Details**

- Class PlayerBot
- Kind Swift
- Address 0x110abcab0
- Size 64 bytes

**Hierarchy**

- PlayerBot
- GKEntity
- NSObject

**Backtrace**

- 0 malloc\_zone\_malloc
- 1 calloc
- 2 class\_createInstance
- 3 +[NSObject allocWithZone:]
- 4 DemoBots.PlayerBot.\_\_allocating\_i...
- 5 DemoBots.LevelScene.init (coder :...
- 6 @objc DemoBots.LevelScene.init (...
- 7 \_decodeObjectBinary
- 8 \_decodeObject
- 9 +[SKNode nodeWithFileNamed:]
- 10 @nonobjc \_\_ObjC.SKScene.\_\_allo...
- 11 DemoBots.LoadSceneOperation.s...
- 12 @objc DemoBots.LoadSceneOper...
- 13 \_NSOQSchedule\_f
- 14 \_dispatch\_client\_callout
- 15 \_dispatch\_queue\_serial\_drain
- 16 \_dispatch\_queue\_invoke
- 17 \_dispatch\_root\_queue\_drain
- 18 \_dispatch\_worker\_thread3
- 19 \_pthread\_wqthread
- 20 start\_wqthread

# Memory Graph Debugging

NEW

The screenshot displays the Xcode interface for memory graph debugging. On the left, the 'DemoBots' process is selected, showing system metrics like CPU (0%), Memory (107.5 MB), and FPS (60). The central pane shows a memory graph for a 'PlayerBot' object (address 0x110abcab0). The graph illustrates the following structure:

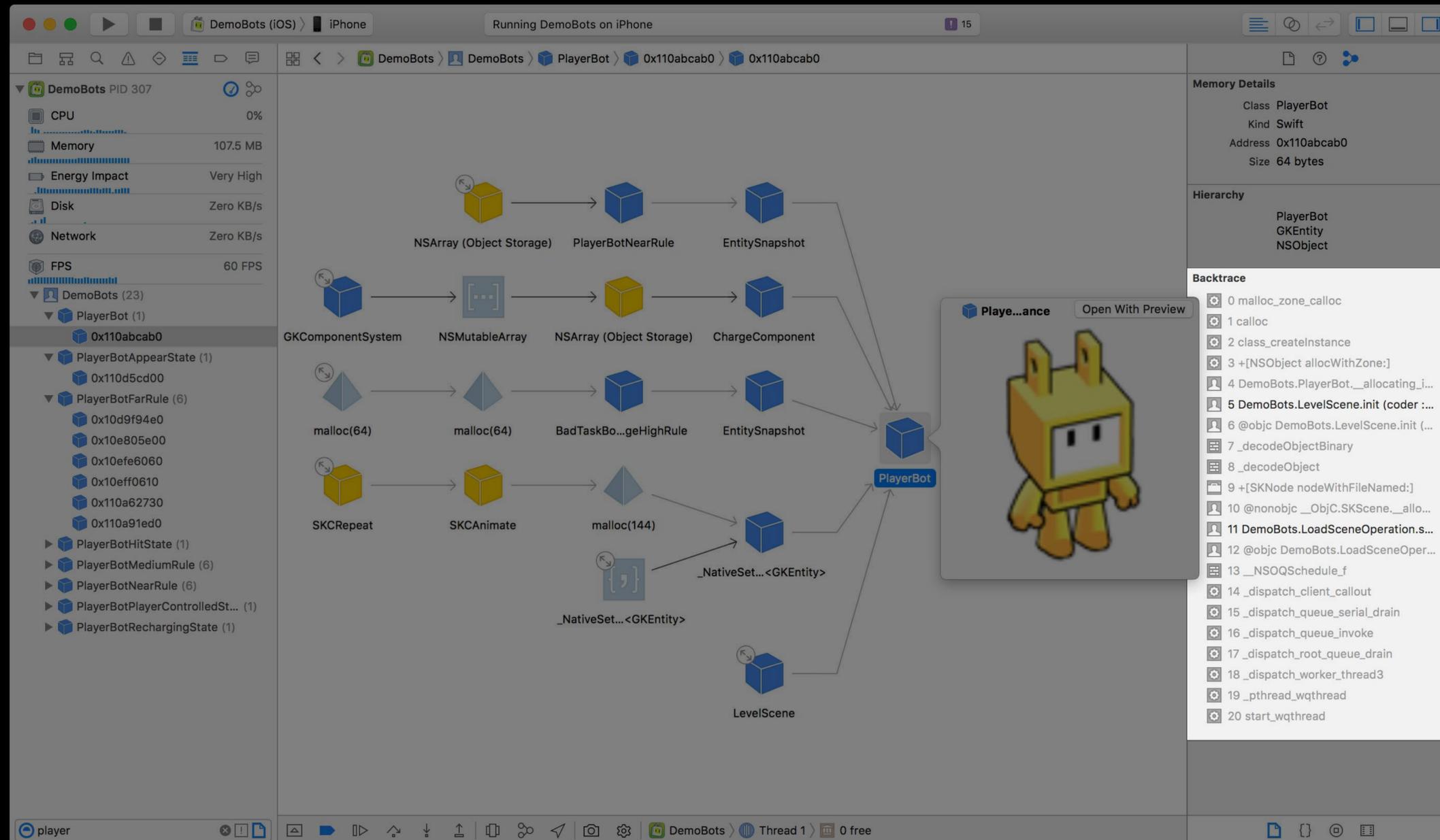
- NSArray (Object Storage)** (yellow cube) points to **PlayerBotNearRule** (blue cube), which points to **EntitySnapshot** (blue cube).
- GKComponentSystem** (blue cube) points to **NSMutableArray** (blue square), which points to **NSArray (Object Storage)** (yellow cube), which points to **ChargeComponent** (blue cube).
- malloc(64)** (blue triangle) points to **malloc(64)** (blue triangle), which points to **BadTaskBo...geHighRule** (blue cube), which points to **EntitySnapshot** (blue cube).
- SKCRepeat** (yellow cube) points to **SKCAnimate** (yellow cube), which points to **malloc(144)** (blue triangle), which points to **\_NativeSet...<GKEntity>** (blue cube).
- \_NativeSet...<GKEntity>** (blue cube) points to **\_NativeSet...<GKEntity>** (blue cube).
- LevelScene** (blue cube) points to **PlayerBot** (blue cube).

The **PlayerBot** object (blue cube) is highlighted, and a preview window shows a yellow robot character. The right sidebar provides the following details:

- Memory Details:** Class: PlayerBot, Kind: Swift, Address: 0x110abcab0, Size: 64 bytes.
- Hierarchy:** PlayerBot, GKEntity, NSObject.
- Backtrace:** A list of 21 stack frames, including `0 malloc_zone_malloc`, `1 calloc`, `2 class_createInstance`, and `5 DemoBots.LevelScene.init`.

# Memory Graph Debugging

NEW



The screenshot displays the Xcode Memory Graph Debugger interface. The central pane shows a memory graph where nodes are represented by colored boxes and arrows indicate memory references. The selected node is a **PlayerBot** object (address 0x110abcab0). A preview window for the selected object shows a yellow robot character.

**Memory Details**

- Class: PlayerBot
- Kind: Swift
- Address: 0x110abcab0
- Size: 64 bytes

**Hierarchy**

- PlayerBot
- GKEntity
- NSObject

**Backtrace**

```
0 malloc_zone_malloc
1 calloc
2 class_createInstance
3 +[NSObject allocWithZone:]
4 DemoBots.PlayerBot._allocating_i...
5 DemoBots.LevelScene.init (coder :...
6 @objc DemoBots.LevelScene.init (...
7 _decodeObjectBinary
8 _decodeObject
9 +[SKNode nodeWithFileNamed:]
10 @nonobjc __ObjC.SKScene._allo...
11 DemoBots.LoadSceneOperation.s...
12 @objc DemoBots.LoadSceneOper...
13 _NSOQSchedule_f
14 _dispatch_client_callout
15 _dispatch_queue_serial_drain
16 _dispatch_queue_invoke
17 _dispatch_root_queue_drain
18 _dispatch_worker_thread3
19 _pthread_wqthread
20 start_wqthread
```

**Graph Nodes and Edges:**

- PlayerBot (selected) is pointed to by LevelScene, GKComponentSystem, EntitySnapshot, ChargeComponent, and another EntitySnapshot.
- LevelScene points to \_NativeSet...<GKEntity>.
- GKComponentSystem points to NSMutableArray.
- NSMutableArray points to NSArray (Object Storage).
- ChargeComponent points to NSArray (Object Storage).
- EntitySnapshot (top) points to PlayerBotNearRule.
- EntitySnapshot (middle) points to BadTaskBo...geHighRule.
- EntitySnapshot (bottom) points to SKCRepeat.
- BadTaskBo...geHighRule points to malloc(144).
- SKCRepeat points to SKCAnimate.
- malloc(144) points to \_NativeSet...<GKEntity>.
- malloc(64) (top) points to NSArray (Object Storage).
- malloc(64) (bottom) points to malloc(144).
- malloc(64) (left) points to GKComponentSystem.

*Demo*

Memory graph debugging

# Memory Graph Debugging

Leaked and abandoned memory

# Memory Graph Debugging

Leaked and abandoned memory

Debugger mode, pauses to inspect app

- Available on macOS, iOS 10, tvOS 10, watchOS 3

# Memory Graph Debugging

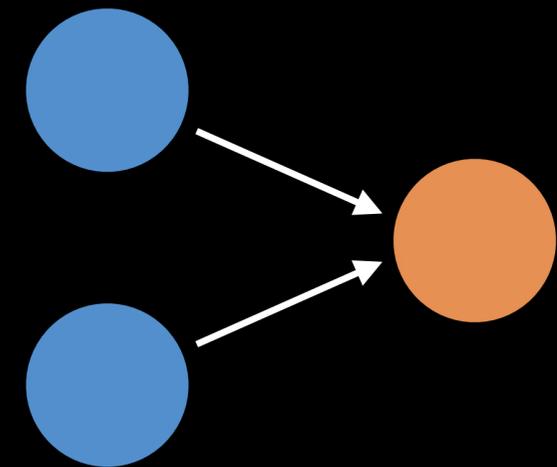
Leaked and abandoned memory

Debugger mode, pauses to inspect app

- Available on macOS, iOS 10, tvOS 10, watchOS 3

Two graph styles:

- Root paths
  - Referenced memory
  - How is the memory held by globals/stacks?



# Memory Graph Debugging

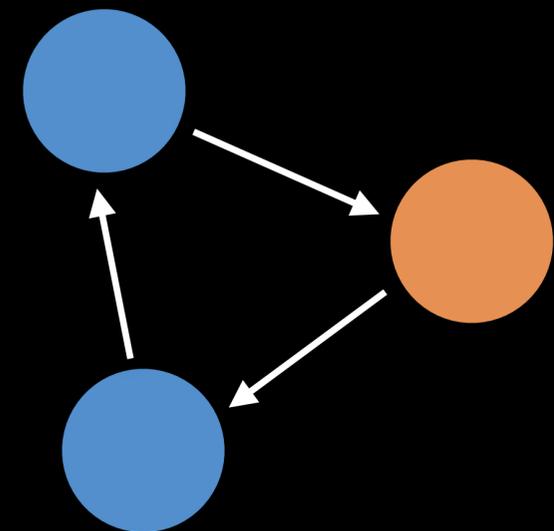
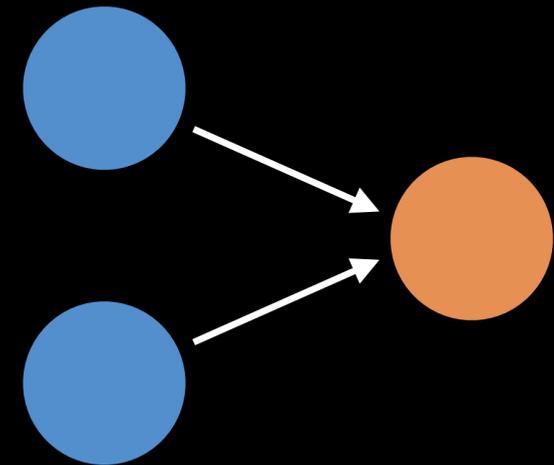
Leaked and abandoned memory

Debugger mode, pauses to inspect app

- Available on macOS, iOS 10, tvOS 10, watchOS 3

Two graph styles:

- Root paths
  - Referenced memory
  - How is the memory held by globals/stacks?
- Cycles
  - Leaked memory
  - How does the leak reference other leaks?



# Memory Graph Debugging

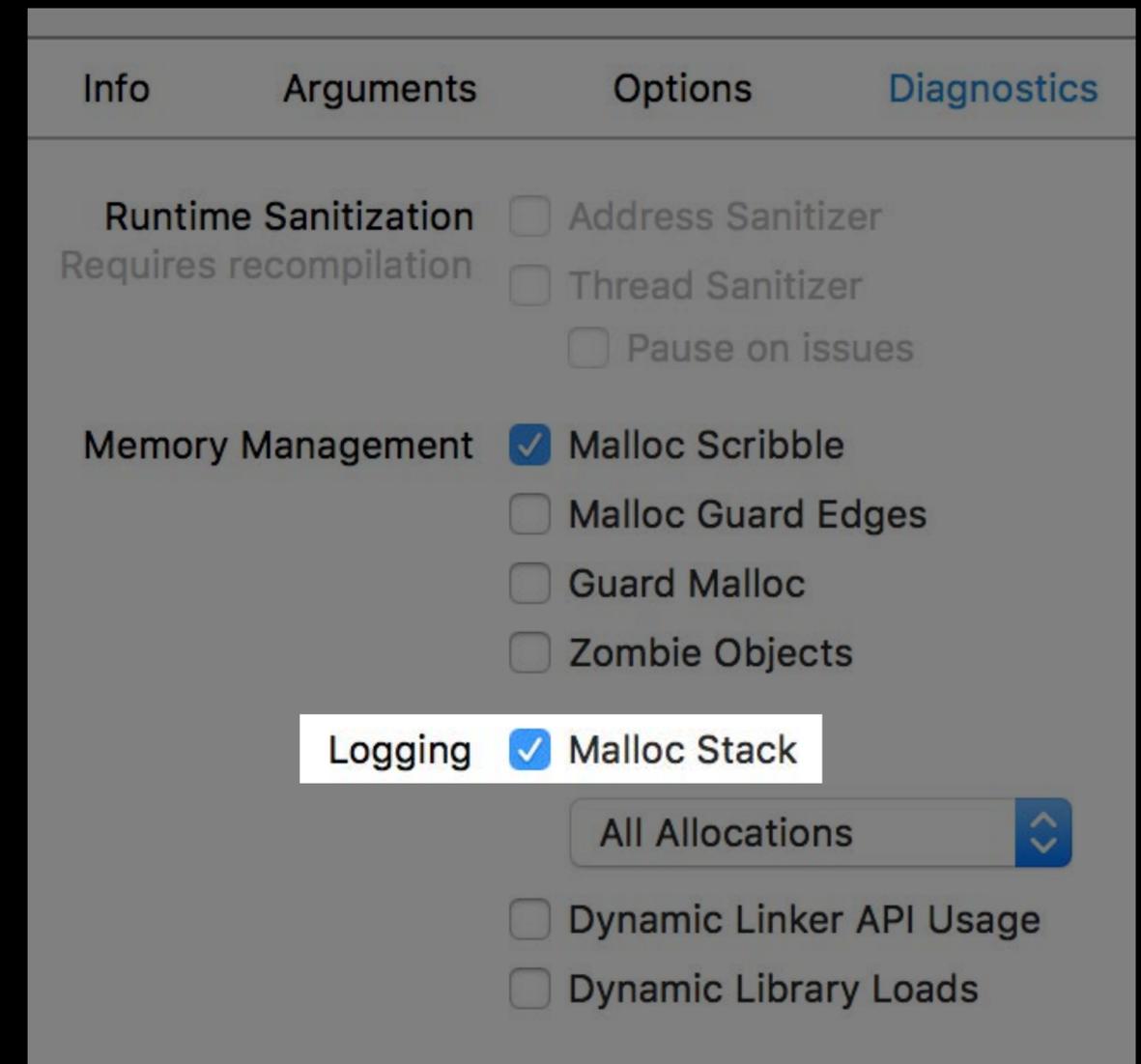
Stack logging integration

# Memory Graph Debugging

## Stack logging integration

Opt-in via Diagnostics scheme tab

- All Allocations
  - MallocStackLogging=1



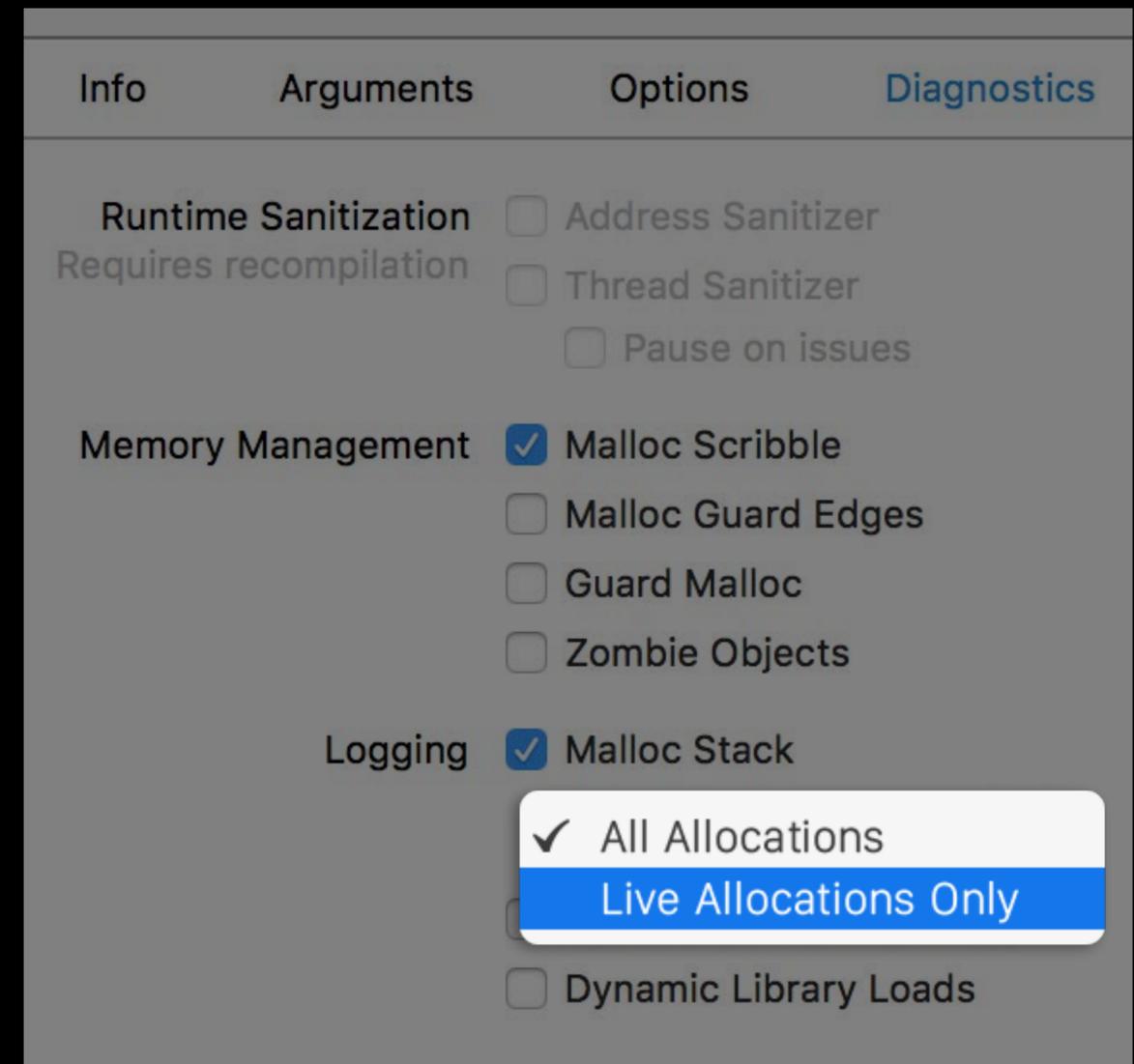
# Memory Graph Debugging

NEW

## Stack logging integration

Opt-in via Diagnostics scheme tab

- All Allocations
  - MallocStackLogging=1
- Live Allocations Only
  - Less memory/disk overhead
  - MallocStackLogging=lite



# Memory Graph Debugging

Introducing .memgraph

NEW



# Memory Graph Debugging

## Introducing .memgraph

NEW

Within Xcode:

- Save: File → “Export Memory Graph...”
- Load: double-click or drag to Xcode
  - No process in debugger — no backtraces, Quick Look, ‘po’



# Memory Graph Debugging

NEW

## Introducing .memgraph

Within Xcode:

- Save: File → “Export Memory Graph...”
- Load: double-click or drag to Xcode
  - No process in debugger — no backtraces, Quick Look, ‘po’

From command-line:

```
$ leaks --outputGraph=<path> <process> # creates .memgraph file
$ {leaks|vmmap|heap} <path/to/file.memgraph> [options] # operates on .memgraph file
```



# Memory Graph Debugging

Usage tips

# Memory Graph Debugging

Usage tips

Graph is conservative

# Memory Graph Debugging

## Usage tips

Graph is conservative

- Avoids 'leaks' false-positives, but there may be extraneous references

# Memory Graph Debugging

## Usage tips

Graph is conservative

- Avoids 'leaks' false-positives, but there may be extraneous references
- Gray references are unknown, may be stale pointer or not strong

# Memory Graph Debugging

## Usage tips

Graph is conservative

- Avoids 'leaks' false-positives, but there may be extraneous references
- Gray references are unknown, may be stale pointer or not strong
  - Enabling Malloc Scribble may improve accuracy

# Memory Graph Debugging

## Usage tips

### Graph is conservative

- Avoids 'leaks' false-positives, but there may be extraneous references
- Gray references are unknown, may be stale pointer or not strong
  - Enabling Malloc Scribble may improve accuracy
- Bold references are known to be strong

# Memory Graph Debugging

## Usage tips

### Graph is conservative

- Avoids 'leaks' false-positives, but there may be extraneous references
- Gray references are unknown, may be stale pointer or not strong
  - Enabling Malloc Scribble may improve accuracy
- Bold references are known to be strong
  - Swift 3's reflection data more accurate

# Memory Graph Debugging

## Usage tips

Graph is conservative

- Avoids 'leaks' false-positives, but there may be extraneous references
- Gray references are unknown, may be stale pointer or not strong
  - Enabling Malloc Scribble may improve accuracy
- Bold references are known to be strong
  - Swift 3's reflection data more accurate

Requires turning off sanitizers

# Memory Graph Debugging

Where to start

# Memory Graph Debugging

Where to start

Validate your expectations

- Are there more objects of your types than you expect?
- Are objects deallocated when they're no longer necessary?

# Memory Graph Debugging

## Where to start

### Validate your expectations

- Are there more objects of your types than you expect?
- Are objects deallocated when they're no longer necessary?

### Find the path that shouldn't be holding your object

- Strong captures from blocks and closures
- Back-references that should be weak/unowned

# Summary

New and improved visual tools in Xcode 8  
Built right into your debugging workflow  
Try them out, improve your App today!



More Information

<https://developer.apple.com/wwdc16/410>

# Related Sessions

---

System Trace in Depth

Nob Hill

Thursday 9:00AM

---

Thread Sanitizer and Static Analysis

Nob Hill

Thursday 10:00AM

---

Debugging Tips and Tricks

Pacific Heights

Friday 1:40PM

---

Using Time Profiler in Instruments

Nob Hill

Friday 3:00PM

---

# Labs

---

GameplayKit Lab	Graphic, Games and Media Lab B	Tuesday 10:10AM
-----------------	--------------------------------	-----------------

---

Profiling and Debugging Lab	Tools Lab C	Thursday 3:00PM
-----------------------------	-------------	-----------------

---

SceneKit Lab	Graphic, Games and Media Lab A	Thursday 3:00PM
--------------	--------------------------------	-----------------

---

SpriteKit Lab	Graphic, Games and Media Lab B	Friday 12:00PM
---------------	--------------------------------	----------------

---



W

W

D

C

1

6