Getting the Most Out of Simulator

Russ Bishop, Simulator and Devices Engineer
Tracy Popenhagen, Engineer Development Technologies
Katelyn Hicks, GPU Software Engineer
What is Simulator?

Burning questions and answers

Simulator from the Command Line

Metal in Simulator
What Is Simulator?
What Is Simulator?
What Is Simulator?
What Is Simulator?

macOS

Kernel
What Is Simulator?

macOS

launchd

Kernel
What Is Simulator?

macOS

cfprefsd
distnoted
launchd

Kernel
What Is Simulator?

macOS

- AppKit
- Foundation
- cfprefsd
- distnoted
- launchd

Kernel
What Is Simulator?

macOS

- Application
- AppKit
- cfprefsd
- launchd

- Application
- Foundation
- distnoted

Kernel
What Is Simulator?
What Is Simulator?

macOS
- Application
- AppKit
- cfprefsd
- launchd
- Foundation
- distnoted

iOS Simulator
- Application
- UIKit
- cfprefsd
- distnoted
- launchd_sim
- Foundation

watchOS Simulator
- Application
- WatchKit
- cfprefsd
- distnoted
- launchd_sim
- Foundation

Kernel
What Is Simulator?
What Is Simulator?

The iOS, watchOS, or tvOS userspace running on the macOS kernel
What Is Simulator?

The iOS, watchOS, or tvOS userspace running on the macOS kernel

Separate launchd, separate daemons, separate darwin notifications, separate URL sessions, and separate mach bootstrap
What Is Simulator?

The iOS, watchOS, or tvOS userspace running on the macOS kernel

Separate launchd, separate daemons, separate darwin notifications, separate URL sessions, and separate mach bootstrap

• Same filesystem, separate $HOME
What Is Simulator?

The iOS, watchOS, or tvOS userspace running on the macOS kernel

Separate launchd, separate daemons, separate darwin notifications, separate URL sessions, and separate mach bootstrap

• Same filesystem, separate $HOME

From libSystem up: built for iOS, watchOS, or tvOS
What Is Simulator?

The iOS, watchOS, or tvOS userspace running on the macOS kernel

Separate launchd, separate daemons, separate darwin notifications, separate URL sessions, and separate mach bootstrap

• Same filesystem, separate $HOME

From libSystem up: built for iOS, watchOS, or tvOS

Uses iOS, watchOS, or tvOS ABI
What Is Simulator?

The iOS, watchOS, or tvOS userspace running on the macOS kernel

Separate launchd, separate daemons, separate darwin notifications, separate URL sessions, and separate mach bootstrap

• Same filesystem, separate $HOME

From libSystem up: built for iOS, watchOS, or tvOS

Uses iOS, watchOS, or tvOS ABI

Built natively for x86 (not an emulator)
Simulator Details

Memory and CPU limits are not simulated
Simulator Details

Memory and CPU limits are not simulated

Different core counts, different threading behaviors
Simulator Details

Memory and CPU limits are not simulated

Different core counts, different threading behaviors

Application Sandbox is not enforced
Simulator Details

Memory and CPU limits are not simulated

Different core counts, different threading behaviors

Application Sandbox is not enforced

Simulates case-sensitive filesystem
Simulator Details

Memory and CPU limits are not simulated

Different core counts, different threading behaviors

Application Sandbox is not enforced

Simulates case-sensitive filesystem

Thread Sanitizer supported
Burning Questions and Answers
Features that may have been hidden in plain sight

Tracy Popenhagen, Engineer Development Technologies
Testing to Older OS?
Can I zoom in on Simulator?
Can I change which audio device Simulator uses for input and output?
Does iCloud work in Simulator?
Can I use Shake Gestures?
How do I get other simulators to show as deployment targets?
Welcome to Xcode

Version 11.0 beta (11M336w)

Get started with a playground
Explore new ideas quickly and easily.

Create a new Xcode project
Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

Clone an existing project
Start working on something from a Git repository.

Devices and Simulators

Open another project...
Welcome to Xcode
Version 11.0 beta (11M336w)

- Get started with a playground
  Explore new ideas quickly and easily.

- Create a new Xcode project
  Create an app for iPhone, iPad, Mac, Apple Watch, or Apple TV.

- Clone an existing project
  Start working on something from a Git repository.

Open another project...
Device
- iPhone

Build Only Device
- Generic iOS Device

iOS Simulators
- iPad Air (3rd generation)
- iPad Pro (9.7-inch)
- iPad Pro (11-inch)
- iPad Pro (12.9-inch) (3rd generation)
- iPhone 8
- iPhone 8 Plus
- iPhone Xs
- iPhone Xs Max
- iPhone XR

Add Additional Simulators...
iPhone

iOS 13.0 (17A5492t)
Model: iPhone XR
Capacity: 113.94 GB (108.81 GB available)
Serial Number: C7CX600KKWR6
Identifier: 00008020-00...0D993600013A

Show as run destination
Connect via network
Take Screenshot
View Device Logs
Open Console

PAIRED WATCHES

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Watch</td>
<td>Apple Watch Series 3 (GPS +..</td>
<td>6.0 (17R5491t) 7f168a1dd04078...</td>
<td></td>
</tr>
</tbody>
</table>

INSTALLED APPS

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No apps installed

DEVICE CONDITIONS
**iPhone Xs Max**

iOS 13.0 (17A5492t)
Model: iPhone Xs Max
Identifier: 4004E59C-C2...20EC8CE70192

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Watch Series 4</td>
<td>Apple Watch Series 4...</td>
<td>6.0</td>
<td>17R54917 987D919A-46F2...</td>
</tr>
</tbody>
</table>

**PAIRED WATCHES**
Create a new simulator:

- **Simulator Name:** [blank]
- **Device Type:** iPhone XR
- **OS Version:** iOS 13.0
- [ ] Paired Apple Watch

[Options: Cancel, Previous, Create]
Create a new simulator:

- **Simulator Name:** Tracy iPad mini
- **Device Type:** iPad mini (5th generation)
- **OS Version:** iOS 13.0

[Options: Cancel, Previous, Create]
override func viewDidLoad() {
    super.viewDidLoad()

    // create a new scene
    let scene = SCNScene(name: "art.scnassets/ship.scn")!

    // create and add a camera to the scene
    let cameraNode = SCNNode()
    cameraNode.camera = SCNCamera()

    scene.rootNode.addChildNode(cameraNode)
}
override func viewDidLoad() {
    super.viewDidLoad()

    // create a new scene
    let scene = SCNScene(named: "art.scnassets/ship.scn")!

    // create and add a camera to the scene
    let cameraNode = SCNNode()
    cameraNode.camera = SCNCamera()

    scene.rootNode.addChildNode(cameraNode)
}
Hello, World!
### iPhone Xs Max

- **iOS**: 13.0 (17A5492t)
- **Model**: iPhone Xs Max
- **Identifier**: 4004E59C-C2...20EC8CE70192

#### Paired Watches

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Watch Series 4 - 44mm</td>
<td>Apple Watch Series 4 - 44mm</td>
<td>6.0</td>
<td>(17R5491t) 987D919A-46F2...</td>
</tr>
</tbody>
</table>

++
Pair an Apple Watch simulator with iPhone Xs Max:

- **Paired Simulator Name:** Apple Watch Series 2 - 38mm
- **Device Type:** Apple Watch Series 2 - 38mm
- **OS Version:** watchOS 6.0

[Button options: Cancel, Previous, Pair]
Pair an Apple Watch simulator with iPhone Xs Max:

- Paired Simulator Name: Apple Watch Series 2 - 38mm
- Device Type: Apple Watch Series 2 - 38mm
- OS Version: watchOS 6.0

[Buttons: Cancel, Previous, Pair]
# iPhone Xs Max

**iOS 13.0 (17A5492t)**  
**Model:** iPhone Xs Max  
**Identifier:** 4004E59C-C2...20EC8CE70192

### Paired Watches

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Watch Series 2 - 38mm</td>
<td>Apple Watch Series 2 - 38mm</td>
<td>6.0</td>
<td>(7R549191)8D26B652-A78...</td>
</tr>
<tr>
<td>Apple Watch Series 4 - 44mm</td>
<td>Apple Watch Series 4 - 44mm</td>
<td>6.0</td>
<td>(17R549191)987D919A-46F2...</td>
</tr>
</tbody>
</table>

+ -
# iPhone Xs Max

iOS 13.0 (17A5492t)
Model: iPhone Xs Max
Identifier: 4004E59C-C2...20EC8CE70192

## Paired Watches

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Watch Series 2 - 38mm</td>
<td>Apple Watch Series 2 - 38mm</td>
<td>6.0</td>
<td>(17R5491t) 8D26B652-A78...</td>
</tr>
<tr>
<td>Apple Watch Series 4 - 44mm</td>
<td>Apple Watch Series 4 - 44mm</td>
<td>6.0</td>
<td>(17R5491t) 987D919A-46F2...</td>
</tr>
<tr>
<td>Simulator</td>
<td>Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 12.1 Simulator</td>
<td>2.49 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 12.0 Simulator</td>
<td>2.4 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 11.4 Simulator</td>
<td>2.14 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 11.3 Simulator</td>
<td>2.14 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 11.2 Simulator</td>
<td>2.11 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 11.1 Simulator</td>
<td>2.1 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 11.0 Simulator</td>
<td>2.09 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 10.3.1 Simulator</td>
<td>1.98 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 10.2 Simulator</td>
<td>1.93 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 10.1 Simulator</td>
<td>1.9 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 10.0 Simulator</td>
<td>1.89 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 9.3 Simulator</td>
<td>1.53 GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS 9.2 Simulator</td>
<td>1.49 GB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Check for and install simulator updates automatically
<table>
<thead>
<tr>
<th>Simulator</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS 12.1 Simulator</td>
<td>2.49 GB</td>
</tr>
<tr>
<td>iOS 12.0 Simulator</td>
<td>2.4 GB</td>
</tr>
<tr>
<td>iOS 11.4 Simulator</td>
<td>2.14 GB</td>
</tr>
<tr>
<td>iOS 11.3 Simulator</td>
<td>2.14 GB</td>
</tr>
<tr>
<td>iOS 11.2 Simulator</td>
<td>2.11 GB</td>
</tr>
<tr>
<td>iOS 11.1 Simulator</td>
<td>2.1 GB</td>
</tr>
</tbody>
</table>
Wallet

Keep your boarding passes, movie tickets, retail coupons, and rewards cards in one place. And show them on your lock screen when you need them.

Find Apps for Wallet

Boarding Passes and Tickets All in One Place

Find apps and start collecting your passes.

FIND APPS
iPad Air (3rd generation)

iOS 13.0 (17A5492t)
Model: iPad Air (3rd generation)
Identifier: D89E4A13-9E...A40E45FFA8E

PAIRED WATCHES

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
</table>

This simulator does not support paired watches.
Are you sure you want to delete 'iPad Air (3rd generation)'?
All of the installed content and settings in this simulator will also be deleted.

Cancel  Delete

PAIRED WATCHES

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>watchOS</th>
<th>Identifier</th>
</tr>
</thead>
</table>

This simulator does not support paired watches.
Simulators are overtaking my screen. Can I make them smaller?
Window Menu of the Simulator

- Minimize
- Minimize All
- Zoom
- Show Device Bezels
- Physical Size
- Point Accurate
- Pixel Accurate
- Bring All to Front
- iPhone Xs Max — 12.2

Physical size of device
1 UIKit point = 1 AppKit point
1 Pixel = 1 Pixel
Can Simulator help me debug UI animations?
Is Siri available in Simulator?
How do I control my tvOS Simulator?
Keyboard for Simple Navigation

Left/right arrows

Return and esc
Can I use Xcode 10.2.1 with iOS 13.0 Simulators?
Xcode 10.x Running with New Simulator.app

If you have Xcode 10.2.1 on the same machine as Xcode 11-Beta
• Launch Xcode 11-Beta and the boot an iOS Simulator
• Leave Simulator.app open while closing Xcode 11-Beta
• Launch Xcode 10.2.1
• Build/run to Simulator
import UIKit
import SpriteKit

Xcode
Version 10.2.1 (10E1001)

Copyright © 1989-2019 Apple Inc. All rights reserved. Apple and the
Apple logo are trademarks of Apple Inc., registered in the U.S. and
other countries.
Can I copy content to Simulator?
Content Available with Drag and Drop

App bundles
Locations
Photo/image files
Video files
URLs
// ShakeMemoryViewController.swift
// ShakeMemory Tester
// Created by Jason Yao on 5/30/19.
// Copyright © 2019 Tracy. All rights reserved.

import UIKit

class ShakeMemoryViewController: UIViewController {
    @IBOutlet weak var resultImageView: UIImage!
    @IBOutlet weak var resultLabel: UILabel!

    var originalImage : UIImage?

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading view.
    }
}
import UIKit

class ShakeMemoryViewController: UIViewController {
    @IBOutlet weak var resultImageView: UIImageView!
    @IBOutlet weak var resultLabel: UILabel!
    var originalImage: UIImage?
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after
    }
}
Our vision for television.

It's TV the way you've always wanted to see it. The cable and premium channels you want. Thousands of movies to buy or rent. Popular streaming services. And coming this fall, original shows and movies from Apple TV+. It's an experience that's curated and personalized to help you discover more of what you love. And it's just the beginning.
Our vision for television.

It’s TV the way you’ve always wanted to see it. The cable and premium channels you want. Thousands of movies to buy or rent. Popular streaming services. And coming this fall, original shows and movies from Apple TV+. It’s an experience that’s curated and personalized to help you discover more of what you love. And it’s all available in one place.
Other Helpful Hints
How to use Walkie-Talkie
Share What You Learned
Simulator from the Command Line

Russ Bishop, Simulator and Devices Engineer
xcrun simctl
xcrun simctl
xcrun simctl help
$ xcrun simctl list
$ xcrun simctl list

== Device Types ==
iPhone X (com.apple.CoreSimulator.SimDeviceType.iPhone-X)
iPhone Xs (com.apple.CoreSimulator.SimDeviceType.iPhone-XS)
iPhone Xs Max (com.apple.CoreSimulator.SimDeviceType.iPhone-XS-Max)

== Runtimes ==
iOS 13.0 - com.apple.CoreSimulator.SimRuntime.iOS-13-0
watchOS 6.0 - com.apple.CoreSimulator.SimRuntime.watchOS-6-0

== Devices ==
-- iOS 13.0 --
   iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
$ xcrun simctl list

== Device Types ==
iPhone X (com.apple.CoreSimulator.SimDeviceType.iPhone-X)
iPhone Xs (com.apple.CoreSimulator.SimDeviceType.iPhone-XS)
iPhone Xs Max (com.apple.CoreSimulator.SimDeviceType.iPhone-XS-Max)

== Runtimes ==
iOS 13.0 - com.apple.CoreSimulator.SimRuntime.iOS-13-0
watchOS 6.0 - com.apple.CoreSimulator.SimRuntime.watchOS-6-0

== Devices ==
-- iOS 13.0 --
   iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
$ xcrun simctl list
== Device Types ==
iPhone X (com.apple.CoreSimulator.SimDeviceType.iPhone-X)
iPhone Xs (com.apple.CoreSimulator.SimDeviceType.iPhone-XS)
iPhone Xs Max (com.apple.CoreSimulator.SimDeviceType.iPhone-XS-Max)
== Runtimes ==
iOS 13.0 - com.apple.CoreSimulator.SimRuntime.iOS-13-0
watchOS 6.0 - com.apple.CoreSimulator.SimRuntime.watchOS-6-0
== Devices ==
-- iOS 13.0 --
  iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
$ xcrun simctl list

== Device Types ==
iPhone X (com.apple.CoreSimulator.SimDeviceType.iPhone-X)
iPhone Xs (com.apple.CoreSimulator.SimDeviceType.iPhone-XS)
iPhone Xs Max (com.apple.CoreSimulator.SimDeviceType.iPhone-XS-Max)

== Runtimes ==
iOS 13.0 - com.apple.CoreSimulator.SimRuntime.iOS-13-0
watchOS 6.0 - com.apple.CoreSimulator.SimRuntime.watchOS-6-0

== Devices ==
-- iOS 13.0 --
  iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
$ xcrun simctl list
== Device Types ==
iPhone X (com.apple.CoreSimulator.SimDeviceType.iPhone-X)
iPhone Xs (com.apple.CoreSimulator.SimDeviceType.iPhone-XS)
iPhone Xs Max (com.apple.CoreSimulator.SimDeviceType.iPhone-XS-Max)
== Runtimes ==
iOS 13.0 - com.apple.CoreSimulator.SimRuntime.iOS-13-0
watchOS 6.0 - com.apple.CoreSimulator.SimRuntime.watchOS-6-0
== Devices ==
-- iOS 13.0 --
   iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
$ xcrun simctl list devices
== Devices ==
-- iOS 13.0 --
   iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
   iPhone XS (69F48379-5DF4-42F3-BD49-2348ACD0E4E0) (Shutdown)
   iPhone XS Max (743D07F9-C418-40FC-8745-2E23E65A9A00) (Shutdown)
   iPhone XR (C6001449-5D04-4647-89FD-5052EA0D18C5) (Shutdown)
   iPhone XR too (065A5682-94E0-4537-93B8-6E5D362612E2) (Shutdown)
-- tvOS 13.0 --
   AppleTV 4K (24E9EE3F-9411-4526-ACDE-24A3F27C8CB0) (Shutdown)
-- watchOS 6.0 --
   Apple Watch Series 4 - 44mm (23F66837-5294-44B3-A787-2BF29610FB00) (Shutdown)
$ xcrun simctl list devices "iPhone X"
== Devices ==
-- iOS 13.0 --
   iPhone X (25375541-C484-47C2-BAB8-78E8CAD3F19E) (Shutdown)
   iPhone XS (69F48379-5DF4-42F3-BD49-2348ACD0E4E0) (Shutdown)
   iPhone XS Max (743D07F9-C418-40FC-8745-2E23E65A9A00) (Shutdown)
   iPhone XR (C6001449-5D04-4647-89FD-5052EA0D18C5) (Shutdown)
   iPhone XR too (065A5682-94E0-4537-93B8-6E5D362612E2) (Shutdown)
-- tvOS 13.0 --
-- watchOS 6.0 --
```bash
$ xcrun simctl list --json
{
    "com.apple.CoreSimulator.SimRuntime.iOS-13-0" : [
        { "state" : "Shutdown",
          "isAvailable" : true,
          "name" : "iPhone X",
          "udid" : "25375541-C484-47C2-BAB8-78E8CAD3F19E" },
        { "state" : "Shutdown",
          "isAvailable" : true,
          "name" : "iPhone Xs",
          "udid" : "69F48379-5DF4-42F3-BD49-2348ACD0E4E0" },
        ...
    ]
}
```
$ xcrun simctl create <name> <device type> <runtime>
$ xcrun simctl create "Test Watch" <device type> <runtime>
$ xcrun simctl create "Test Watch" "Apple Watch Series 4 - 44mm" <runtime>
$ xcrun simctl create "Test Watch" "Apple Watch Series 4 - 44mm" watchOS6.0
$ xcrun simctl create "Test Watch" "Apple Watch Series 4 - 44mm" watchOS6.0
E1CF7D9D-9961-42E7-87CB-881A14EF479C
simctl

$ xcrun simctl create "Test Watch" "Apple Watch Series 4 - 44mm" watchOS6.0
E1CF7D9D-9961-42E7-87CB-881A14EF479C

$ NEW_DEVICE=$(xcrun simctl create "Test Phone" "iPhone XR" iOS13.0)
$ xcrun simctl create "Test Watch" "Apple Watch Series 4 - 44mm" watchOS6.0
E1CF7D9D-9961-42E7-87CB-881A14EF479C

$ NEW_DEVICE=$(xcrun simctl create "Test Phone" "iPhone XR" iOS13.0)

$ echo "🤖 Created ${NEW_DEVICE}"

$ xcrun simctl create "Test Watch" "Apple Watch Series 4 - 44mm" watchOS6.0
E1CF7D9D-9961-42E7-87CB-881A14EF479C

$ NEW_DEVICE=$(xcrun simctl create "Test Phone" "iPhone XR" iOS13.0)
$ echo "🤖 Created ${NEW_DEVICE}"
🤖 Created 2772A4D0-EC31-4385-AB56-C1C6D22DA6D5
simctl

$ xcrun simctl spawn <device> <command> <arguments>
$ xcrun simctl spawn booted defaults write com.example.app ResetDatabase -bool YES
$ xcrun simctl spawn booted defaults write com.example.app ResetDatabase -bool YES
simctl

$ xcrun simctl spawn booted log stream --predicate 'senderImagePath CONTAINS "nsurlsessiond"'
$ xcrun simctl spawn booted log stream --predicate 'senderImagePath CONTAINS "nsurlsessiond"'

nsurlsessiond: [com.apple.nsurlsessiond:background] Task <0402>.<1> resuming, QOS(0x19)
nsurlsessiond: [com.apple.nsurlsessiond:background] Task <0402>.<1> resuming
nsurlsessiond: [com.apple.nsurlsessiond:background] Task <0402>.<1> has not requested a begin delay
nsurlsessiond: [com.apple.nsurlsessiond:background] Task <0402>.<1> skipping delayed request callback - delegate not implemented
nsurlsessiond: [com.apple.nsurlsessiond:background] Task <0402>.<1> will begin
nsurlsessiond: [com.apple.nsurlsessiond:background] Task <5743>.<2> dataTaskWithRequest: <NSMutableURLRequest: 0x7f955d50d2c0> { URL: <private> } [allowsCellularAccess: 1]
simctl

$xcrun simctl diagnose
$ xcrun simctl diagnose -l
$ xcrun simctl diagnose -l
Successfully wrote simctl diagnose archive to 'simctl_diagnose_2019_06_07.tar.gz'
$ xcrun simctl diagnose -l
Successfully wrote simctl diagnose archive to 'simctl_diagnose_2019_06_07.tar.gz'

Include a simctl diagnose when filing bugs
simctl

$ xcrun simctl launch <device> <bundle> <arguments>
$ xcrun simctl launch booted <bundle> <arguments>
$ xcrun simctl launch booted com.apple.example <arguments>
$ xcrun simctl launch booted com.apple.example -MyDefaultKey YES
$ xcrun simctl launch booted com.apple.example --MyDefaultKey YES


$ xcrun simctl launch --console-pty booted com.apple.example -MyDefaultKey YES
$ xcrun simctl launch --console-pty booted com.apple.example -MyDefaultKey YES
$ xcrun simctl launch --console-pty booted com.apple.example -MyDefaultKey YES
com.apple.example: 42042
2019-05-22 23:22:36.766 [Example] Starting Rendering...
$ xcrun simctl launch --console-pty booted com.apple.example -MyDefaultKey YES
com.apple.example: 42042
2019-05-22 23:22:36.766 [Example] Starting Rendering...
$ xcrun simctl launch --console-pty booted com.apple.example -MyDefaultKey YES
com.apple.example: 42042
2019-05-22 23:22:36.766 [Example] Starting Rendering...
Frame Count: 931
^C
$
Other simctl Commands
Other simctl Commands

$ xcrun simctl boot <device>
Other simctl Commands

$ xcrun simctl boot <device>
$ xcrun simctl shutdown <device>
Other simctl Commands

$ xcrun simctl boot <device>

$ xcrun simctl shutdown all
Other simctl Commands

$ xcrun simctl boot <device>
$ xcrun simctl shutdown all
$ xcrun simctl delete <device>
Other simctl Commands

$ xcrun simctl boot <device>
$ xcrun simctl shutdown all
$ xcrun simctl delete unavailable
Other simctl Commands

$ xcrun simctl boot <device>
$ xcrun simctl shutdown all
$ xcrun simctl delete unavailable
$ xcrun simctl pair <watch> <phone>
Other simctl Commands

$ xcrun simctl boot <device>
$ xcrun simctl shutdown all
$ xcrun simctl delete unavailable
$ xcrun simctl pair <watch> <phone>
$ xcrun simctl addmedia <device> <file1> <file2>
Other simctl Commands

- $ xcrun simctl boot <device>
- $ xcrun simctl shutdown all
- $ xcrun simctl delete unavailable
- $ xcrun simctl pair <watch> <phone>
- $ xcrun simctl addmedia <device> <file1> <file2>
- $ xcrun simctl get_app_container <device> <bundle id> <type>
Other simctl Commands

$ xcrun simctl boot <device>
$ xcrun simctl shutdown all
$ xcrun simctl delete unavailable
$ xcrun simctl pair <watch> <phone>
$ xcrun simctl addmedia <device> <file1> <file2>
$ xcrun simctl get_app_container <device> <bundle id> <type>
$ xcrun simctl install <device> <example.app>
Other simctl Commands

$ xcrun simctl boot <device>

$ xcrun simctl shutdow all

$ xcrun simctl delete unavailable

$ xcrun simctl pair <watch> <phone>

$ xcrun simctl addmedia <device> <file1> <file2>

$ xcrun simctl get_app_container <device> <bundle id> <type>

$ xcrun simctl install <device> <example.app>

$ xcrun simctl io <device> screenshot <output.png>
simctl

$ xcrun simctl clone <device> <clone name>
Metal in Simulator
Amazing rendering, amazing speed

Katelyn Hicks, GPU Software Engineer
Simulator Now Supports Metal

Simulator is much faster through Metal

Metal can be used in your application
Simulator Now Supports Metal
Simulator Now Supports Metal
Apple Frameworks Are Now Accelerated

UIKit  SpriteKit  SceneKit  CoreAnimation  CoreImage  MapKit
Apple Frameworks Are Now Accelerated

- UIKit
- SpriteKit
- SceneKit
- CoreAnimation
- CoreImage
- MapKit

The transition is transparent
Performance Improvements
UIKit

Previous Xcode Versions

Xcode 11 and macOS 10.15
Performance Improvements
UIKit

Previous Xcode Versions

Xcode 11 and macOS 10.15
Performance Improvements

SpriteKit

Previous Xcode Versions

Xcode 11 and macOS 10.15
Performance Improvements
SpriteKit

Previous Xcode Versions

Xcode 11 and macOS 10.15
Performance Improvements

SceneKit

Previous Xcode Versions

Xcode 11 and macOS 10.15
Performance Improvements

SceneKit

Previous Xcode Versions

Xcode 11 and macOS 10.15
Performance Improvements
Performance Improvements
Metal API Is Now Available

And fully accelerated using the Mac GPU
Enabling Metal Development in Simulator

Previous Xcode Versions

Xcode 11 and macOS 10.15
Enabling Metal Development in Simulator

Previous Xcode Versions

Xcode 11 and macOS 10.15
Enabling Metal Development in Simulator

Previous Xcode Versions

Xcode 11 and macOS 10.15
Using Metal API in Simulator
GPU Family
GPU Family

Easy to use GPU capability query
GPU Family

Easy to use GPU capability query

Introduces three families
GPU Family

Easy to use GPU capability query

Introduces three families

• Common — Supported by all platforms
Easy to use GPU capability query

Introduces three families

- Common — Supported by all platforms
- Mac — macOS GPUs
GPU Family

Easy to use GPU capability query

Introduces three families

- Common — Supported by all platforms
- Mac — macOS GPUs
- Apple — iOS and tvOS GPUs
GPU Family

Easy to use GPU capability query

Introduces three families
- Common — Supported by all platforms
- Mac — macOS GPUs
- Apple — iOS and tvOS GPUs

Simulator supports MTLGPUFamilyApple2
Achieving GPU Hardware Acceleration

iOS Application

Simulator Environment
Achieving GPU Hardware Acceleration
Achieving GPU Hardware Acceleration

iOS Application

- UIKit
- SceneKit

OpenGL ES Software Renderer
- Metal

Simulator Environment
Achieving GPU Hardware Acceleration

iOS Application

Simulator Environment

OpenGL ES Software Renderer

Metal

UIKit

SceneKit
Achieving GPU Hardware Acceleration
Achieving GPU Hardware Acceleration

iOS Application

Simulator Environment

OpenGL ES Software Renderer

UIKit
SceneKit

Metal

Native macOS Environment

macOS Metal
Mac GPU
Test Performance on Actual Devices
Test Performance on Actual Devices

Simulator performance is not representative device performance
Test Performance on Actual Devices

Simulator performance is not representative device performance

As always — profile and optimize on actual devices
Texture Storage in Metal API

Texture storage modes are different on macOS and iOS
Texture Storage in Metal API

Texture storage modes are different on macOS and iOS

<table>
<thead>
<tr>
<th></th>
<th>Private</th>
<th>Shared</th>
<th>Memoryless</th>
<th>Managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS and tvOS</td>
<td><img src="green" alt="Private" /></td>
<td><img src="green" alt="Shared" /></td>
<td><img src="green" alt="Memoryless" /></td>
<td><img src="green" alt="Managed" /></td>
</tr>
<tr>
<td>macOS</td>
<td><img src="green" alt="Private" /></td>
<td><img src="green" alt="Shared" /></td>
<td><img src="green" alt="Memoryless" /></td>
<td><img src="green" alt="Managed" /></td>
</tr>
</tbody>
</table>
# Texture Storage in Simulator

<table>
<thead>
<tr>
<th></th>
<th>Private</th>
<th>Shared</th>
<th>Memoryless</th>
<th>Managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS and tvOS</td>
<td>▫️</td>
<td>▫️</td>
<td>▫️</td>
<td>▫️</td>
</tr>
<tr>
<td>Simulator</td>
<td>▫️</td>
<td>▫️</td>
<td>▫️</td>
<td>▫️</td>
</tr>
</tbody>
</table>
Shared Memory Model

- CPU
- Shared Memory
- MTLTexture
- GPU
Shared Memory Model

CPU -- Shared Memory -- MTLTexture -- GPU
Shared Memory Model

CPU

Shared Memory

MTLTexture

GPU
Shared Memory Model

- CPU
- Shared Memory
- MTLTexture
- GPU
Shared Memory Model

- CPU
- Shared Memory
  - MTLTexture
- GPU
Shared Memory Model

CPU → Shared Memory → MTLTexture → GPU
Shared Memory Model

CPU → Shared Memory → MTLTexture → GPU
Texture Storage Differences in Simulator

Texture types that require MTLStorageMode.private in Simulator

- Multisampled
- Depth Stencil
- Linear
Private Memory Model

CPU

Private Memory

MTLTexture

GPU
Private Memory Model

CPU → Private Memory → MTLTexture → GPU
Private Memory Model
// For Simulator, multisampled, depth, stencil, and linear textures must be private
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
// For Simulator, multisampled, depth, stencil, and linear textures must be private
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
   .blitData(fromBuffer: tmpBuffer, toTexture: texture)
// For Simulator, multisampled, depth, stencil, and linear textures must be private

(textureDescriptor.pixelFormat = .depth32Float_stencil8)

#if targetEnvironment(simulator)
(textureDescriptor.storageMode = .private)
#else
(textureDescriptor.storageMode = .shared)
#endif

// Create the texture with the supported storage mode

(let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
// For Simulator, multisampled, depth, stencil, and linear textures must be private
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
// For Simulator, multisampled, depth, stencil, and linear textures must be private
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
// For Simulator, multisampled, depth, stencil, and linear textures must be private
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
} else {
    // Initialize the shared texture
// For Simulator, multisampled, depth, stencil, and linear textures must be private
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
    textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
}
#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
   .blitData(fromBuffer: tmpBuffer, toTexture: texture)
} else {
    // Initialize the shared texture
    initWithTextureData(texture: texture)
}
textureDescriptor.pixelFormat = .depth32Float_stencil8

#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
} else {
    // Initialize the shared texture
    initWithTextureData(texture: texture)
}
#if targetEnvironment(simulator)
textureDescriptor.storageMode = .private
#else
textureDescriptor.storageMode = .shared
#endif

// Create the texture with the supported storage mode
let texture = device.makeTexture(descriptor: textureDescriptor)!

if texture.storageMode == .private {
    // Create and initialize a temporary shared buffer or texture
    let tmpBuffer = device.makeBuffer(length: textureSize, options: .storageModeShared)!
    initWithTextureData(buffer: tmpBuffer)
    // Blit the contents to the private texture
    blitData(fromBuffer: tmpBuffer, toTexture: texture)
} else {
    // Initialize the shared texture
    initWithTextureData(texture: texture)
}
Supporting Simulator in a Metal App

Full documentation of the few differences in Simulator
Sample code available
Summary

Simulator is iOS, watchOS, or tvOS userspace running on macOS kernel

Simulator is fastest, easiest way to simulate devices on your Mac

`xcrun simctl` for command line and scripts

Simulator now has GPU acceleration through Metal
More Information

developer.apple.com/wwdc19/418