Introducing Core Haptics

Michael Diu, Interactive Haptics
Doug Scott, Interactive Haptics
Where to use Core Haptics

Expressing content

Our first haptics

Dynamic parameters

Apple Haptic Audio Pattern (AHAP)
Where to use Core Haptics

Expressing content

Our first haptics

Dynamic parameters

Apple Haptic Audio Pattern (AHAP)
Where to use Core Haptics
Expressing content
Our first haptics
Dynamic parameters
Apple Haptic Audio Pattern (AHAP)
Where to use Core Haptics
Expressing content
Our first haptics
Dynamic parameters
Apple Haptic Audio Pattern (AHAP)
Where to use Core Haptics
Expressing content
Our first haptics
Dynamic parameters
Apple Haptic Audio Pattern (AHAP)
Where to Use Core Haptics
What is Core Haptics?
What is Core Haptics?

CoreHaptics

**What is Core Haptics?**

CoreMIDI

App

AVFoundation

AVAudioSession

AVAudioPlayer

AVAudioEngine

AudioToolbox

SystemSound

AudioUnits

UIKit

UIFeedbackGenerator

Core Haptics

Audio Hardware

Haptics Hardware
Device Support

Same feel across products

iPhone 8
iPhone 8 Plus
iPhone X
iPhone XS
iPhone XR
iPhone XS Max
Core Haptics in Context

- CoreHaptics
- CoreMIDI
- Audio Hardware
- Haptics Hardware

App

- AVFoundation
- AVAudioSession
- AVAudioPlayer
- AVAudioEngine
- AudioToolbox
- SystemSound
- AudioUnits
- UIKit
- UIFeedbackGenerator

Core Haptics
Choose the Right Haptics API

**UIFeedbackGenerator**

Impact, Selection, Notification

- Apple designed vocabulary
- Common across apps
- Improved in iOS 13
## Choose the Right Haptics API

<table>
<thead>
<tr>
<th>UIFeedbackGenerator</th>
<th>Core Haptics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impact, Selection, Notification</td>
<td>Custom haptic and audio patterns</td>
</tr>
<tr>
<td>Apple designed vocabulary</td>
<td>Accepts timestamps for future playback</td>
</tr>
<tr>
<td>Common across apps</td>
<td>Rich playback and modulation controls</td>
</tr>
<tr>
<td>Improved in iOS 13</td>
<td></td>
</tr>
</tbody>
</table>

Haptics and Audio Duality
Haptics and Audio Duality

(Haptic) Home Button
Haptics and Audio Duality

(Haptic) Home Button

Haptic Crown
Haptics and Audio Duality

(Haptic) Home Button

Haptic Crown

UIDatePicker
Gaming Applications
Gaming Applications
Gaming Applications

Visceral feeling
Gaming Applications

Visceral feeling

Simulate physical contact
Core Haptics and Augmented Reality
Core Haptics and Augmented Reality

Increase immersion
Core Haptics and Augmented Reality

Increase immersion

Ground user gestures
Core Haptics and Augmented Reality

Increase immersion

Ground user gestures

Feedback on device or AR world events
Expressing Content
Classes for Content

CHHapticPattern
CHHapticEvent
CHHapticParameter
Classes for Content

- CHHapticPattern
- CHHapticEvent
- CHHapticParameter

Classes for Playback

- CHHapticEngine
  - owns and vends
  - CHHapticPatternPlayer(s)
  - CHHapticAdvancedPatternPlayer(s)
Events, Parameters, and Patterns

CHHapticEvent
- Time
- Type
- [EventParameters]
Events, Parameters, and Patterns

CHHapticEvent
Time
Type
[EventParameters]

Time
0.0

CHHapticEvent
Time
Type
[EventParameters]

Time
Events, Parameters, and Patterns

CHHapticPattern

CHHapticEvent
- Time
- Type
- [EventParameters]

CHHapticEvent
- Time
- Type
- [EventParameters]

CHHapticEvent
- Time
- Type
- [EventParameters]
Types of Events

**Haptic Transient**

Think “striking”; momentary; instantaneous
## Types of Events

<table>
<thead>
<tr>
<th>Haptic Transient</th>
<th>Haptic Continuous</th>
<th>Audio Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Think “striking”; momentary; instantaneous</td>
<td>Think “bowing”</td>
<td>Can be background texture Richer set of knobs</td>
</tr>
</tbody>
</table>
### Types of Events

<table>
<thead>
<tr>
<th>Haptic Transient</th>
<th>Haptic Continuous</th>
<th>Audio Continuous</th>
<th>Audio Custom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Think “striking”; momentary; instantaneous</td>
<td>Think “bowing”</td>
<td>Can be background texture</td>
<td>Developer-provided waveform</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Richer set of knobs</td>
<td></td>
</tr>
</tbody>
</table>
Our First Event Parameter

Haptic Intensity / Audio Volume
Our First Event Parameter

Haptic Intensity / Audio Volume

No output

0.0

Maximum strength

1.0
HapticSharpness
An abstraction layer for physical dimensions
HapticSharpness
An abstraction layer for physical dimensions

Round, organic

0.0 1.0
HapticSharpness
An abstraction layer for physical dimensions

Round, organic

0.0

Crisp, precise

1.0
HapticSharpness
An abstraction layer for physical dimensions

Round, organic | Crisp, precise
0.0 | 1.0
HapticSharpness
An abstraction layer for physical dimensions

Round, organic  Crisp, precise
0.0            1.0

Flashlight button

Swipe up to open
HapticSharpness
An abstraction layer for physical dimensions

Round, organic  0.0  App Switcher  1.0  Crisp, precise

Flashlight button
Palette

Sample code
Our First Haptics

Douglas Scott, Interactive Haptics
Our First Haptics

Douglas Scott, Interactive Haptics
Playing a Haptic Pattern: Recommended Flow
1. Create haptic content

CHHapticPattern → NSDictionary
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
3. Create haptic pattern player

CHHapticPattern

CHHapticPatternPlayer

CHHapticEngine
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
3. Create haptic pattern player
4. Start the engine

CHHapticPattern

CHHapticPatternPlayer

CHHapticEngine (Running)
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
3. Create haptic pattern player
4. Start the engine
5. Start the player
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
3. Create haptic pattern player
4. Start the engine
5. Start the player
6. Wait for the player to finish (optional)
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
3. Create haptic pattern player
4. Start the engine
5. Start the player
6. Wait for the player to finish (optional)...

CHHapticPattern
CHHapticPatternPlayer (Stopped)
CHHapticEngine (Running)

“Player is done”
Playing a Haptic Pattern: Recommended Flow

1. Create haptic content
2. Create haptic engine
3. Create haptic pattern player
4. Start the engine
5. Start the player
6. Wait for the player to finish (optional)...
7. Stop the engine (optional)
Demo
Using the Core Haptics API
// Using the API

import UIKit
import CoreHaptics
import CoreMotion
// Using the API

import UIKit
import CoreHaptics
import CoreMotion
// Using the API

import UIKit
import CoreHaptics
import CoreMotion

class ViewController: UIViewController, UICollisionBehaviorDelegate {
    ...
    // Haptic Engine & State:
    var engine: CHHapticEngine!
    var engineNeedsStart = true
// Using the API

import UIKit
import CoreHaptics
import CoreMotion

class ViewController: UIViewController, UICollisionBehaviorDelegate {
    ...
    // Haptic Engine & State:
    var engine: CHHapticEngine!
    var engineNeedsStart = true
override func viewDidLoad() {
    // Create and configure the engine before doing anything else
    // since the game begins immediately.
    createAndStartHapticEngine()
override func viewDidLoad() {
    // Create and configure the engine before doing anything else
    // since the game begins immediately.
    createAndStartHapticEngine()
}
private func createAndStartHapticEngine() {
    // Create and configure the haptic engine.
    do {
        engine = try CHHapticEngine()
    } catch let error {
        fatalError("Engine Creation Error: \(error)"")
    }
    ...
}
private func createAndStartHapticEngine() {
    // Create and configure the haptic engine.
    do {
        engine = try CHHapticEngine()
    } catch let error {
        fatalError("Engine Creation Error: \(error)"")
    }
    ...

// The stopped handler alerts engine stoppage.

engine.stoppedHandler = { reason in
    print("Stop Handler: The engine stopped for reason: \(reason.rawValue)")
    switch reason {
        // Handle possible reasons here.
    }
    // Indicate that the next time the app requires a haptic,
    // the app must call engine.start().
    self.engineNeedsStart = true
}
// The stopped handler alerts engine stoppage.

engine.stoppedHandler = { reason in
    print("Stop Handler: The engine stopped for reason: \(reason.rawValue)")
    switch reason {
        // Handle possible reasons here.
    }
    // Indicate that the next time the app requires a haptic,
    // the app must call engine.start().
    self.engineNeedsStart = true
}
// Start haptic engine to prepare for use.

do {
    try engine.start()
    // Indicate that the next time the app requires a haptic,
    // the app doesn't need to call engine.start().
    engineNeedsStart = false
} catch let error {
    fatalError("Engine Start Error: \(error)")
}
}
// Start haptic engine to prepare for use.
do {
    try engine.start()
    // Indicate that the next time the app requires a haptic,
    // the app doesn't need to call engine.start().
    engineNeedsStart = false
} catch let error {
    fatalError("Engine Start Error: \(error)")
}
func collisionBehavior(...) {
    do {
        ...
        // Create a haptic pattern player from this magnitude.
        let hapticPlayer = try playerForMagnitude(normalizedMagnitude)
        ...
    }
}
func collisionBehavior(...) {
    do {
        ...
        // Create a haptic pattern player from this magnitude.
        let hapticPlayer = try playerForMagnitude(normalizedMagnitude)
    ...
}
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
...
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {

...  
let hapticEvent = CHHapticEvent(eventType: .hapticTransient, parameters: [
    CHHapticEventParameter(parameterID: .hapticSharpness, value: sharpness),
    CHHapticEventParameter(parameterID: .hapticIntensity, value: intensity),
], relativeTime: 0)

...
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {

    let hapticEvent = CHHapticEvent(eventType: .hapticTransient, parameters: [
        CHHapticEventParameter(parameterID: .hapticSharpness, value: sharpness),
        CHHapticEventParameter(parameterID: .hapticIntensity, value: intensity),
    ], relativeTime: 0)

    ...
}
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...
    let hapticEvent = CHHapticEvent(eventType: .hapticTransient, parameters: [
        CHHapticEventParameter(parameterID: .hapticSharpness, value: sharpness),
        CHHapticEventParameter(parameterID: .hapticIntensity, value: intensity),
    ], relativeTime: 0)
    ...

private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...
    let hapticEvent = CHHapticEvent(eventType: .hapticTransient, parameters: [
        CHHapticEventParameter(parameterID: .hapticSharpness, value: sharpness),
        CHHapticEventParameter(parameterID: .hapticIntensity, value: intensity),
    ], relativeTime: 0)
    ...
}
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...
    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {

    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...
    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {

    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {

    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)

    let pattern = try CHHapticPattern(events: [hapticEvent, audioEvent], parameters: [])

    return pattern
}
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...

    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)

    let pattern = try CHHapticPattern(events: [hapticEvent, audioEvent], parameters: [])

private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...
    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)

    let pattern = try CHHapticPattern(events: [hapticEvent, audioEvent], parameters: []
    return try engine.makePlayer(with: pattern)
}
private func playerForMagnitude(_ magnitude: Float) throws -> CHHapticPatternPlayer? {
    ...
    let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
        CHHapticEventParameter(parameterID: .audioVolume, value: volume),
        CHHapticEventParameter(parameterID: .decayTime, value: decay),
        CHHapticEventParameter(parameterID: .sustained, value: 0),
    ], relativeTime: 0)

    let pattern = try CHHapticPattern(events: [hapticEvent, audioEvent], parameters: [])[0]
    return try engine.makePlayer(with: pattern)
}
func collisionBehavior(...) {
    do {
        ...
        // Create a haptic pattern player from this magnitude.
        let hapticPlayer = try playerForMagnitude(normalizedMagnitude)
        // Start player, “fire and forget”.
        try hapticPlayer?.start(atTime: CHHapticTimeImmediate)
    } catch let error {
        print("Haptic Playback Error: \(error)")
    }
}
func collisionBehavior(...) {
    do {
        ...
        // Create a haptic pattern player from this magnitude.
        let hapticPlayer = try playerForMagnitude(normalizedMagnitude)
        // Start player, “fire and forget”.
        try hapticPlayer?.start(atTime: CHHapticTimeImmediate)
    }
    catch let error {
        print("Haptic Playback Error: \(error)")
    }
}
func collisionBehavior(...) {
    do {
        ...
        // Create a haptic pattern player from this magnitude.
        let hapticPlayer = try playerForMagnitude(normalizedMagnitude)
        // Start player, “fire and forget”.
        try hapticPlayer?.start(atTime: CHHapticTimeImmediate)
    } catch let error {
        print("Haptic Playback Error: \(error)")
    }
}
Dynamic Parameters
Dynamic Parameters

Affects all Events

Modifies (modulates) EventParameter values

Embed in Pattern or send in real time

One pattern, infinite variations
Dynamic Parameters

Affects all Events

Modifies (modulates) EventParameter values

Embed in Pattern or send in real time

One pattern, infinite variations
Dynamic Parameters

Affects all Events

Modifies (modulates) EventParameter values

Embed in Pattern or send in real time

One pattern, infinite variations
Dynamic Parameters

Affects all Events

Modifies (modulates) EventParameter values

Embed in Pattern or send in real time

One pattern, infinite variations
Dynamic Parameter and Event Parameter Interaction
Dynamic Parameter and Event Parameter Interaction
Apple Haptic Audio Pattern (AHAP)

Playing patterns from a file
What is AHAP?
AHAP
Apple Haptic Audio Pattern

Describes a pattern as text

Schema for JSON

Can use Swift Codable

Edit, share, and integrate into workflow

Separate content from code
AHAP
Apple Haptic Audio Pattern

Describes a pattern as text

Schema for JSON

Can use Swift Codable

Edit, share, and integrate into workflow

Separate content from code
AHAP
Apple Haptic Audio Pattern

Describes a pattern as text

Schema for JSON

Can use Swift Codable

Edit, share, and integrate into workflow

Separate content from code
AHAP
Apple Haptic Audio Pattern

Describes a pattern as text
Schema for JSON
Can use Swift Codable
Edit, share, and integrate into workflow
Separate content from code
AHAP
Apple Haptic Audio Pattern

Describes a pattern as text

Schema for JSON

Can use Swift Codable

Edit, share, and integrate into workflow

Separate content from code
Building an AHAP File
// An Example of a Simple AHAP File
{
    "Version": 1.0
}

// An Example of a Simple AHAP File

{
    "Version": 1.0
}

// An Example of a Simple AHAP File

{
    "Version": 1.0,
    "Pattern": [ ]
}

// An Example of a Simple AHAP File

{
    "Version": 1.0,
    "Pattern": [
    ]
}

// Adding our first Event

{
    "Version": 1.0,
    "Pattern": [
        {
            "Event": {
                "Time": 0.0,
                "EventType": "HapticTransient"
            }
        }
    ]
}
// Adding our first Event

{
    "Version": 1.0,
    "Pattern": [
        {
            "Event": {
                "Time": 0.0,
                "EventType": "HapticTransient"
            }
        }
    ]
}
// Adding an optional EventParameter array

{
   "Version": "1.0",
   "Pattern": [
      {
         "Event": {
            "Time": 0.0,
            "EventType": "HapticTransient",
            "EventParameters": []
         }
      }
   ]
}
// Adding an optional EventParameter array

{
    "Version": 1.0,
    "Pattern": [
        {
            "Event": {
                "Time": 0.0,
                "EventType": "HapticTransient",
                "EventParameters": []
            }
        }
    ]
}
// Adding EventParameters for Intensity and Sharpness

{
  "Version": 1.0,
  "Pattern": [
    {
      "Event": {
        "Time": 0.0,
        "EventType": "HapticTransient",
        "EventParameters": [
          { "ParameterID": "HapticIntensity", "ParameterValue": 0.8 },
          { "ParameterID": "HapticSharpness", "ParameterValue": 0.4 }
        ]
      }
    }
  ]
}
// Adding EventParameters for Intensity and Sharpness

{
  "Version": 1.0,
  "Pattern": [
    {
      "Event": {
        "Time": 0.0,
        "EventType": "HapticTransient",
        "EventParameters": [
          { "ParameterID": "HapticIntensity", "ParameterValue": 0.8 },
          { "ParameterID": "HapticSharpness", "ParameterValue": 0.4 }
        ]
      }
    }
  ]
}
// A second event

"Pattern": [
    {
        "Event": { ... }
    },
    {
        "Event": {
            "Time": 0.5,
            "EventType": "HapticContinuous",
            "EventDuration": 0.25,
            "EventParameters": [
                { "ParameterID": "HapticIntensity", "ParameterValue": 0.8 },
                { "ParameterID": "HapticSharpness", "ParameterValue": 0.4 }
            ]
        }
    }
]
// A second event

"Pattern": [{
    "Event": {...
    }
},
{
    "Event": {
        "Time": 0.5,
        "Event Type": "HapticContinuous",
        "Event Duration": 0.25,
        "Event Parameters": [
            { "Parameter ID": "Haptic Intensity", "Parameter Value": 0.8 },
            { "Parameter ID": "Haptic Sharpness", "Parameter Value": 0.4 }
        ]
    }
}]}
// EventDuration key is required for AudioContinuous, HapticContinuous

"Pattern": [
    {
        "Event": { ...
    },
    {
        "Event": {
            "Time": 0.5,
            "EventType": "HapticContinuous",
            "EventDuration": 0.25,
            "EventParameters": [
                { "ParameterID": "HapticIntensity", "ParameterValue": 0.8 },
                { "ParameterID": "HapticSharpness", "ParameterValue": 0.4 }
            ]
        }
    }
]
// EventDuration key is required for AudioContinuous, HapticContinuous

"Pattern": [
  {
    "Event": { ...
    }
  },
  {
    "Event": {
      "Time": 0.5,
      "EventType": "HapticContinuous",
      "EventDuration": 0.25,
      "EventParameters": [
        { "ParameterID": "HapticIntensity", "ParameterValue": 0.8 },
        { "ParameterID": "HapticSharpness", "ParameterValue": 0.4 }
      ]
    }
  }
]
Building an AHAP File
AHAP Structure Summary

- **Version**: 1.0
- **Pattern**: 
  - **Event**
    - **Time**: 2
    - **EventType**: HapticContinuous
    - **EventDuration**: 1.0
    - **EventParameters**: 
      - **ParameterID**: HapticIntensityControl
        - **Time**: 0.0
        - **ParameterCurveControlPoints**: 0.5
      - **ParameterValue**: 0.8
  - **ParameterCurve** (Dynamic)
    - **ParameterID**: HapticSharpnessControl
      - **Time**: 3.0
      - **ParameterCurveControlPoints**: 
        - **ParameterValue**: 1.0
HapticSampler
Sample code
More to Discover

Michael Diu, Interactive Haptics
## Core Haptics: More to Discover

<table>
<thead>
<tr>
<th></th>
<th>Audio Events and Parameters</th>
<th>Modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Custom audio + synthesized audio</td>
<td>Use Parameter Curves to achieve the same effect as a sequence of DynamicParameters</td>
</tr>
<tr>
<td></td>
<td>Pitch; Pan; Brightness</td>
<td>Statically defined as part of the Pattern, or created during playback</td>
</tr>
<tr>
<td></td>
<td>Attack, Decay, Release</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Haptics-only mode</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>AudioSession Integration</th>
<th>Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Routing and interruption behavior follows AudioSession Category</td>
<td>Error Handling for Pattern Parsing</td>
</tr>
<tr>
<td></td>
<td>New API to allow haptics during recording</td>
<td>Error Handling and Recovery during Playback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Playback controls</th>
<th>Player and Engine Notifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Muting (Player; or entire Engine)</td>
<td>Playback complete callbacks</td>
</tr>
<tr>
<td></td>
<td>Looping</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Playback rate adjust</td>
<td>Engine stopped notifications</td>
</tr>
<tr>
<td></td>
<td>Pause/resume</td>
<td>autoShutdown mode</td>
</tr>
<tr>
<td></td>
<td>Seek; play or stop at timestamp</td>
<td></td>
</tr>
</tbody>
</table>
What makes a good haptic pattern?
Do the rules for sound design carry over?
Take advantage of the updated HIG for haptics.
Summary

Immersion and effortless interactions
Summary

Immersion and effortless interactions

Haptic and audio feedback — together
Summary

Immersion and effortless interactions

Haptic and audio feedback — together

Vocabulary and a file format, AHAP
Summary

Immersion and effortless interactions
Haptic and audio feedback — together
Vocabulary and a file format, AHAP
Performant API
Summary

Immersion and effortless interactions

Haptic and audio feedback — together

Vocabulary and a file format, AHAP

Performant API

Design guidelines and examples
Summary

Immersion and effortless interactions

Haptic and audio feedback — together

Vocabulary and a file format, AHAP

Performant API

Design guidelines and examples

Large base of Taptic Engines
## More Information

developer.apple.com/wwdc19/520

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Haptics Lab</td>
<td>Thursday, 11:00</td>
</tr>
<tr>
<td>Core Haptics Lab (2)</td>
<td>Friday, 9:00</td>
</tr>
</tbody>
</table>