#WWDC19

Designing Audio-Haptic Experiences

Hugo Verweij, Sound Designer
Camille Moussette, Interaction Designer

#WWDC19

Designing Audio-Haptic Experiences

Hugo Verweij, Sound Designer
Camille Moussette, Interaction Designer

#WWDC19

Designing Audio-Haptic Experiences

Hugo Verweij, Sound Designer
Camille Moussette, Interaction Designer

What is an Audio Haptic experience?

What is an Audio Haptic experience?

Three guiding principles

What is an Audio Haptic experience?

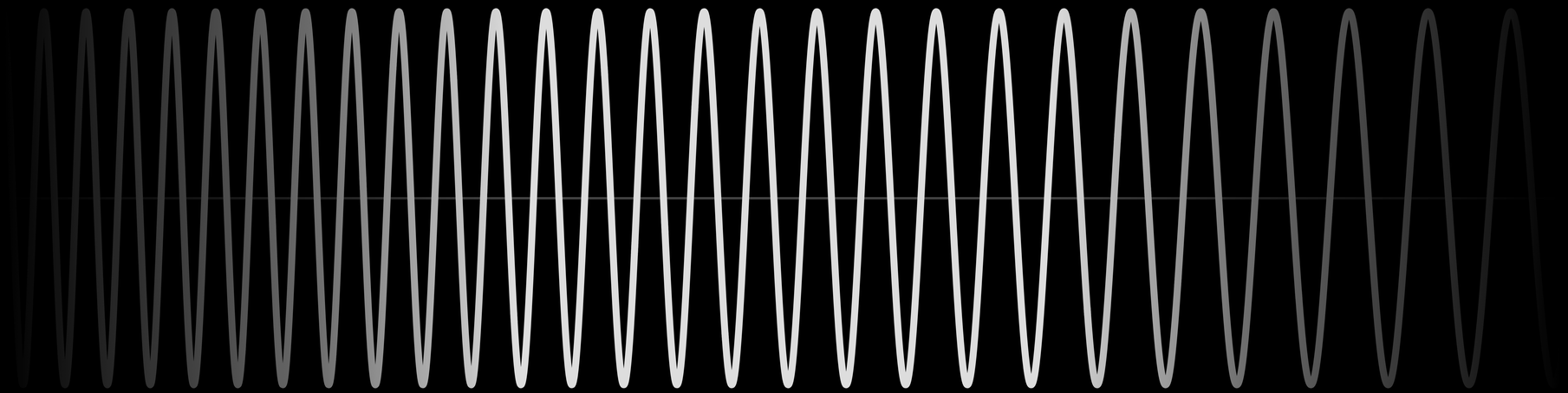
Three guiding principles

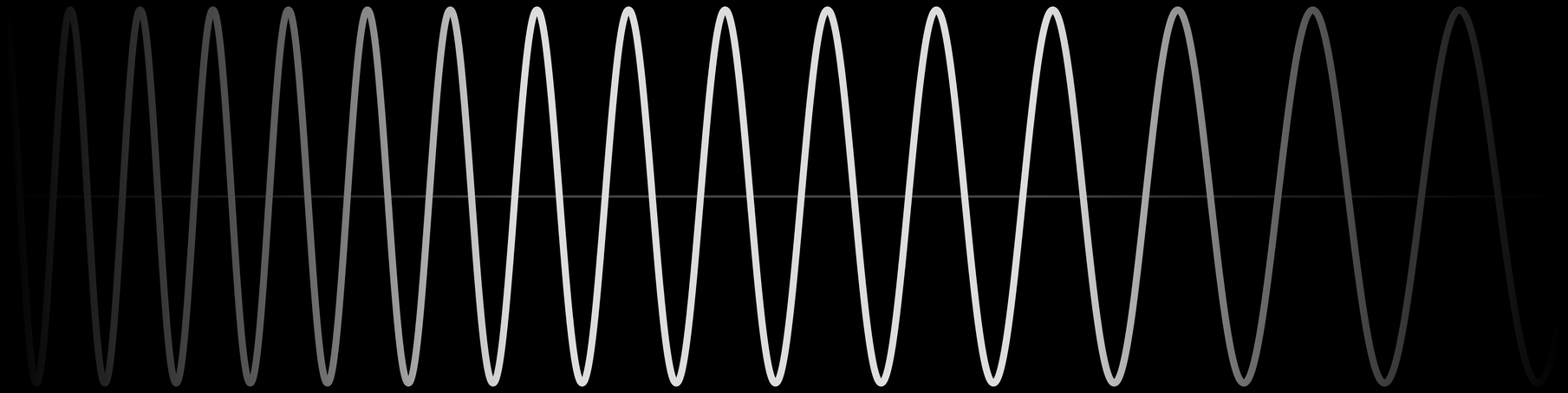
Techniques

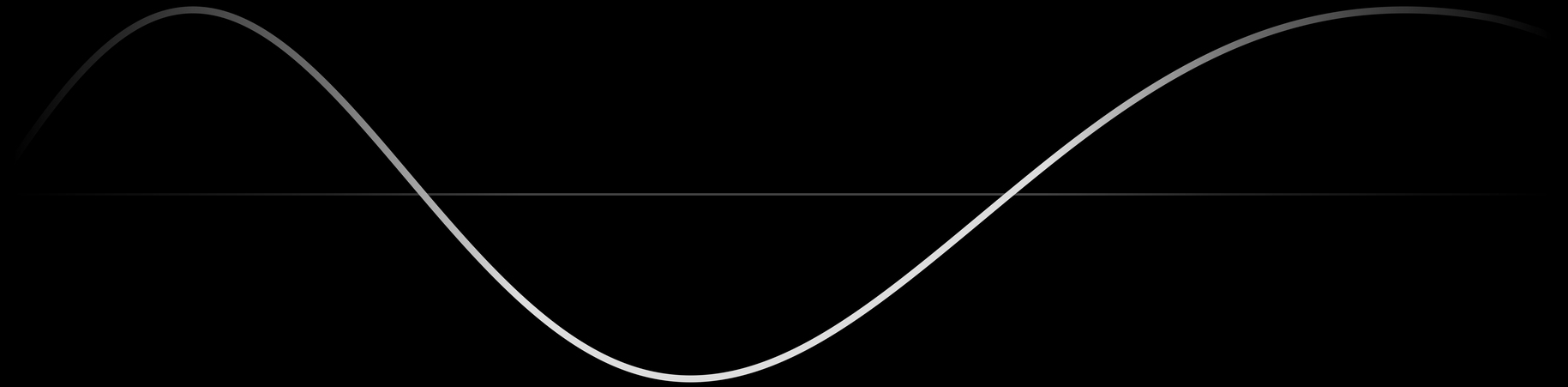
What is an Audio Haptic experience?

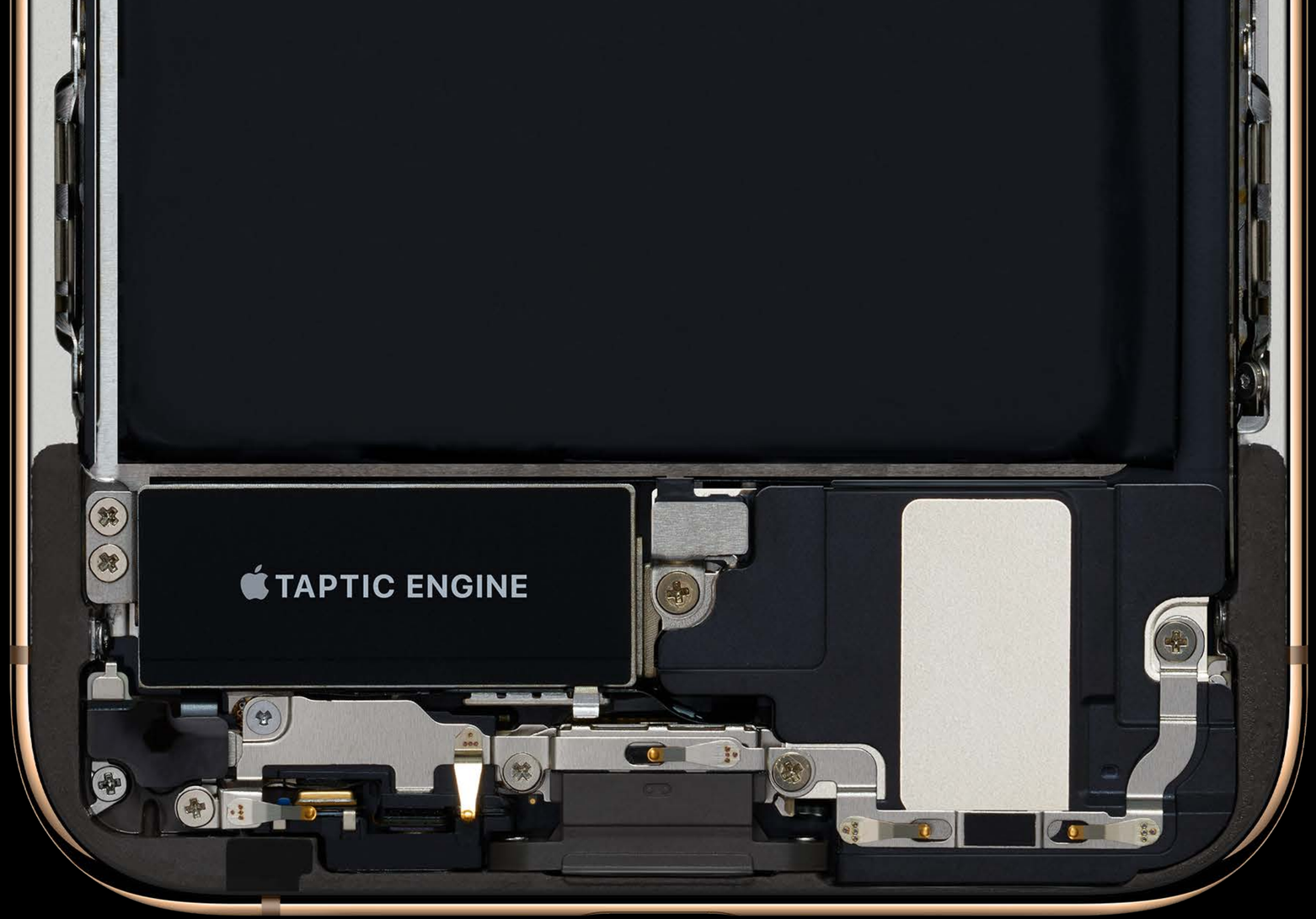
Three guiding principles

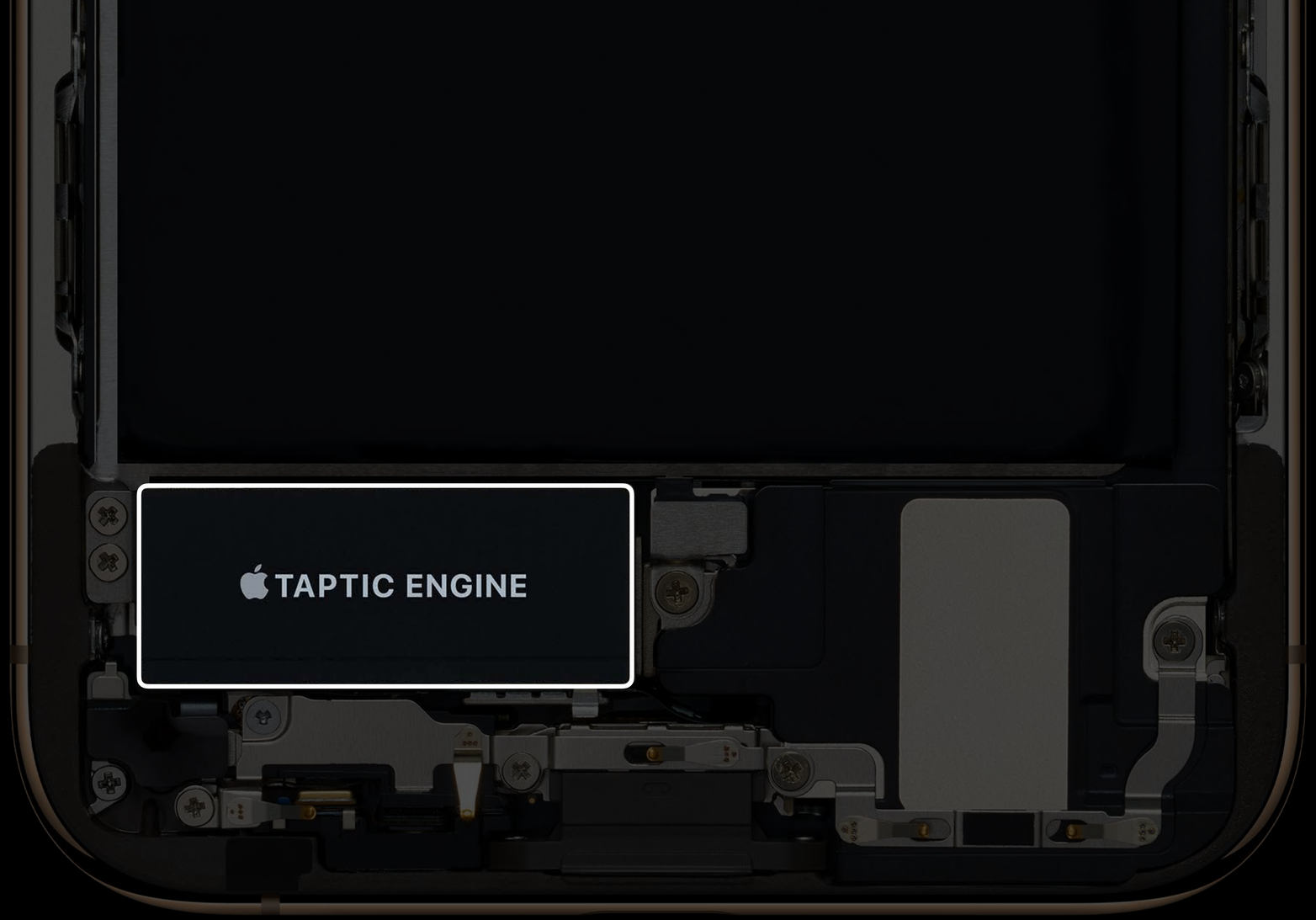
Techniques

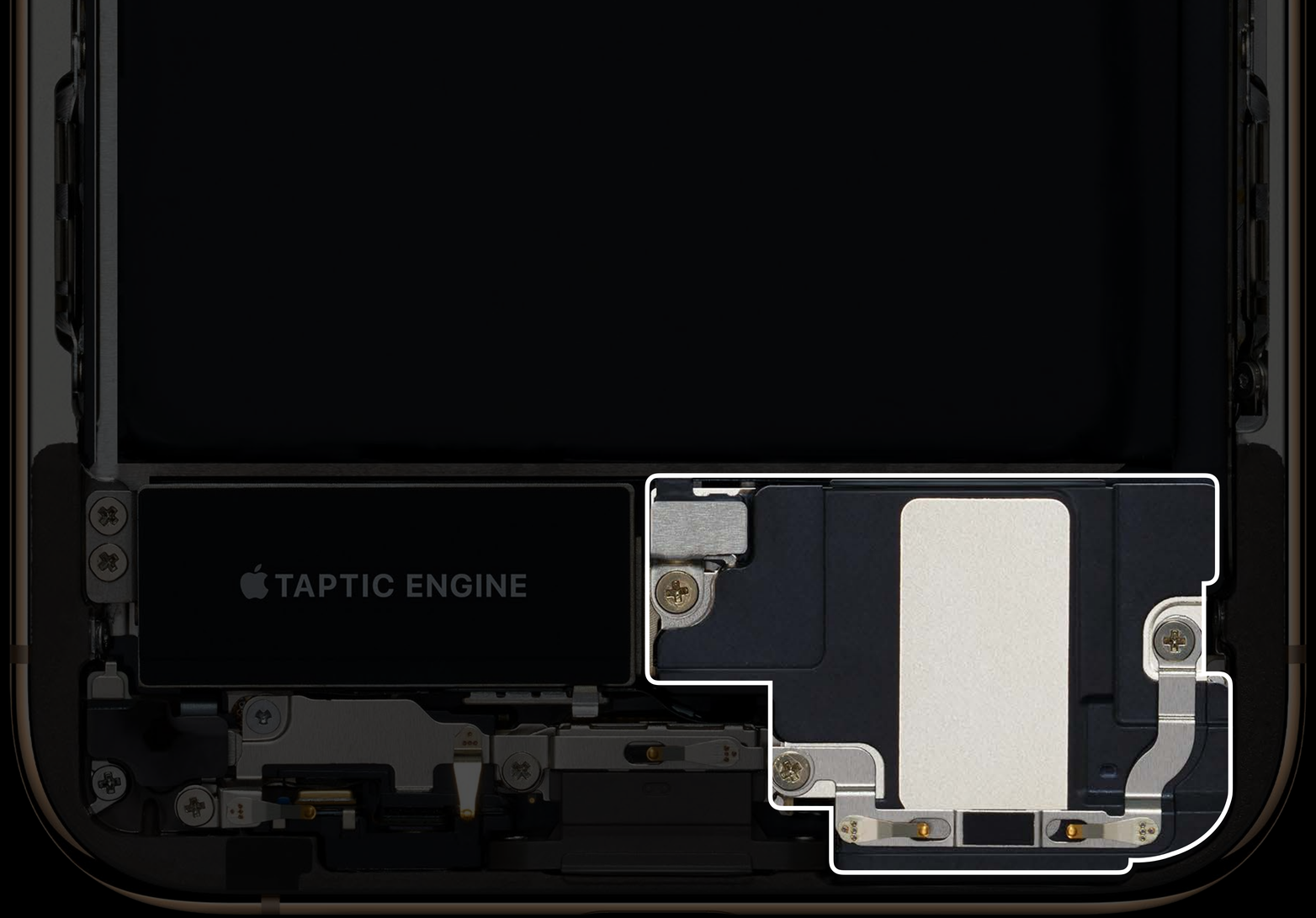


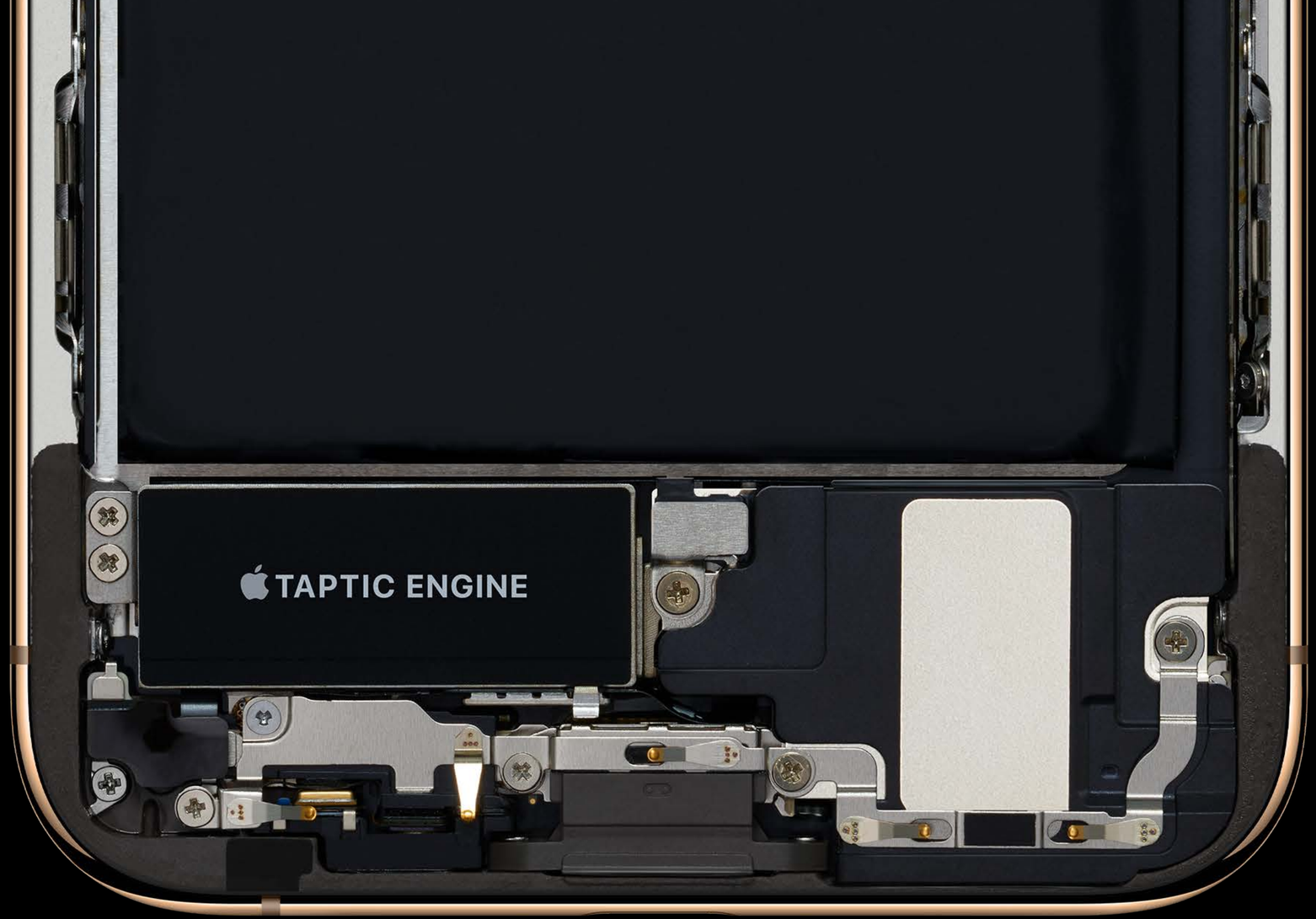




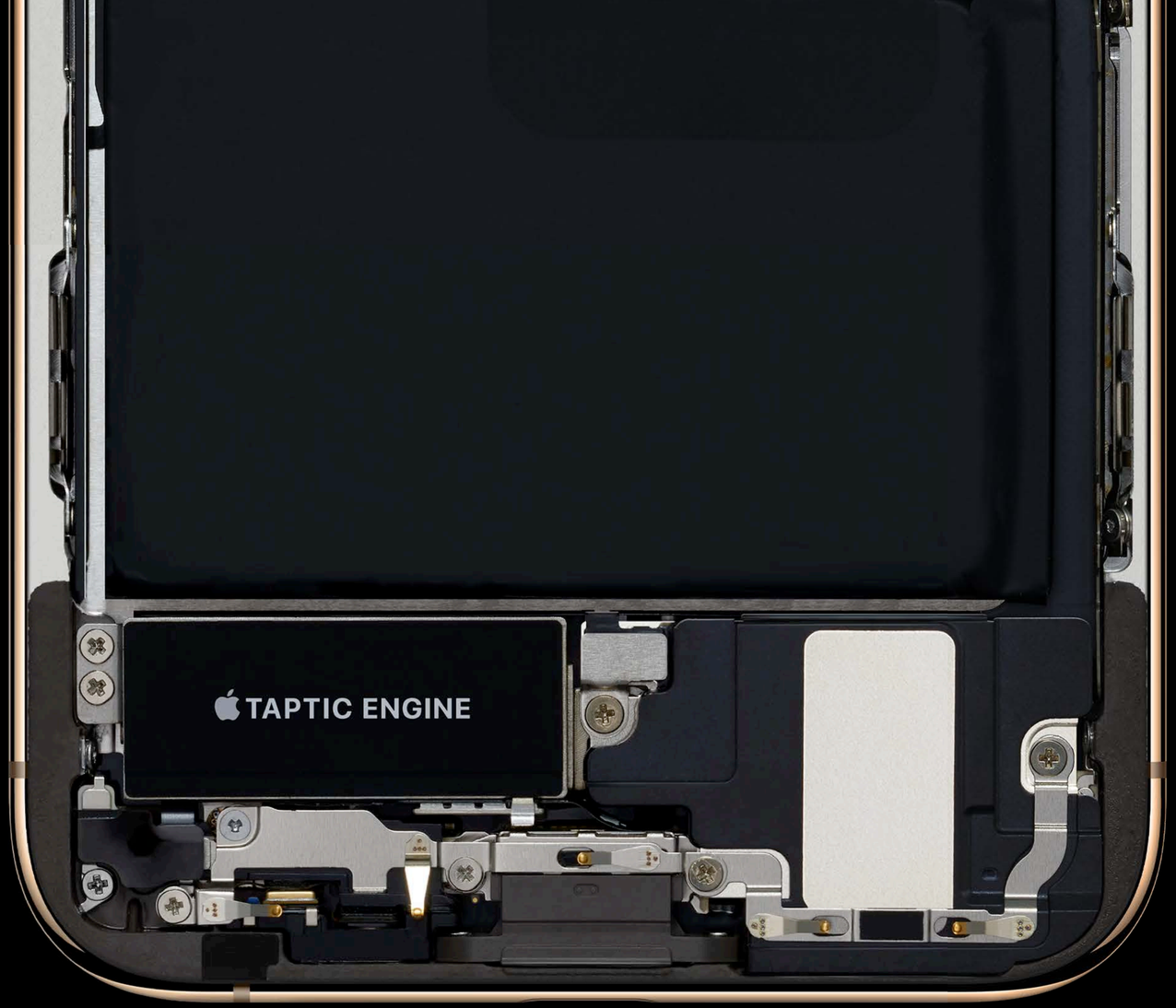

TAPTIC ENGINE

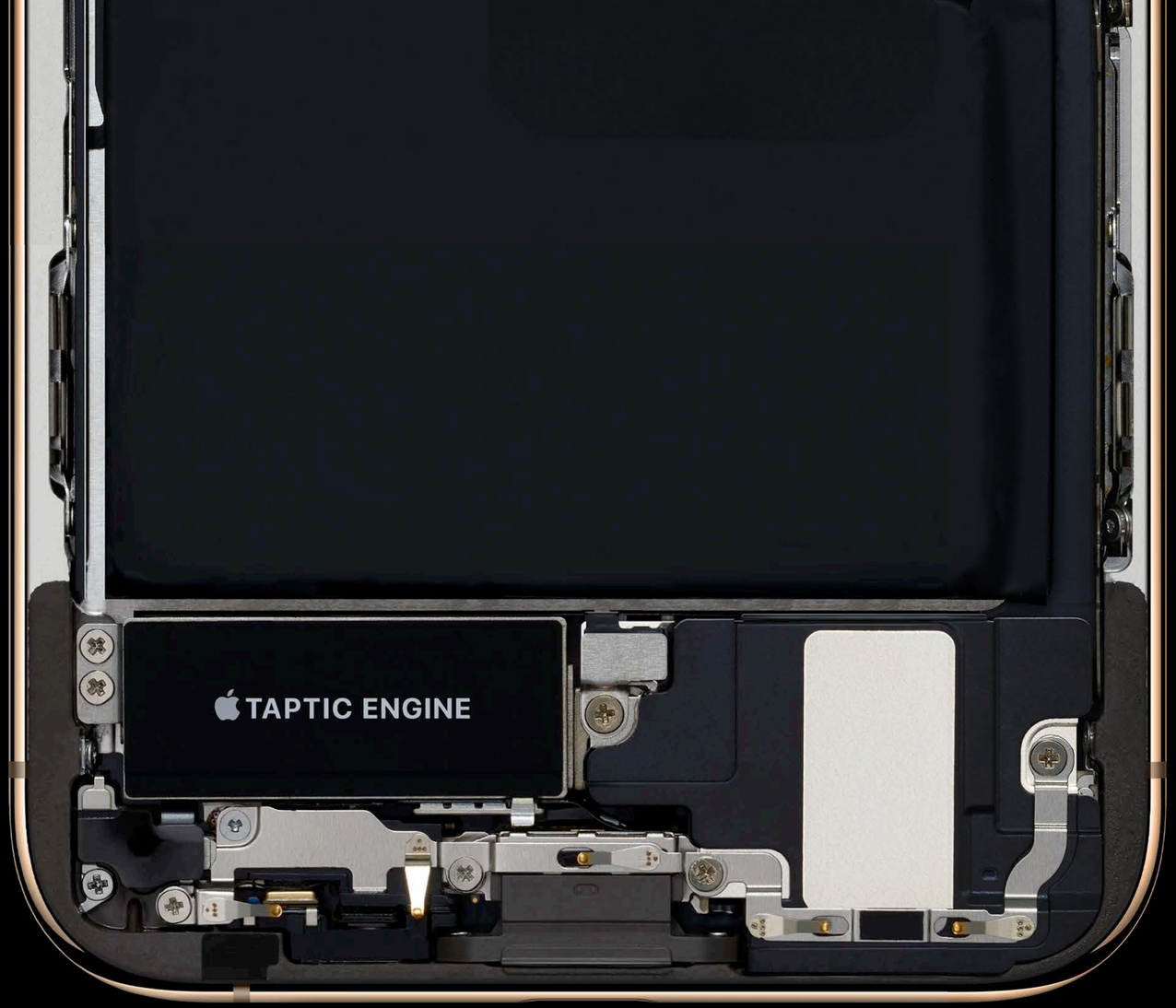

TAPTIC ENGINE

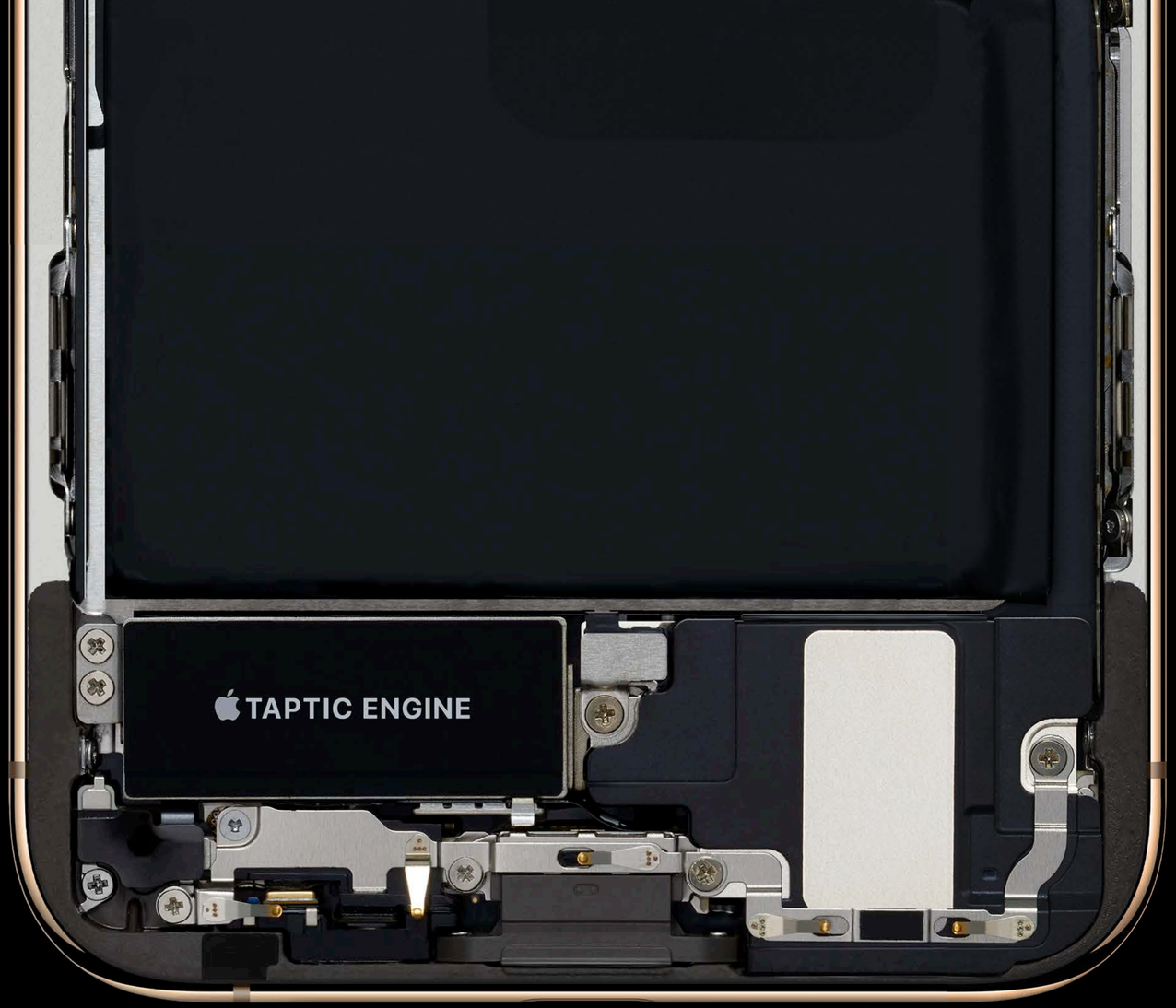

TAPTIC ENGINE


TAPTIC ENGINE

Can you feel it?


TAPTIC ENGINE

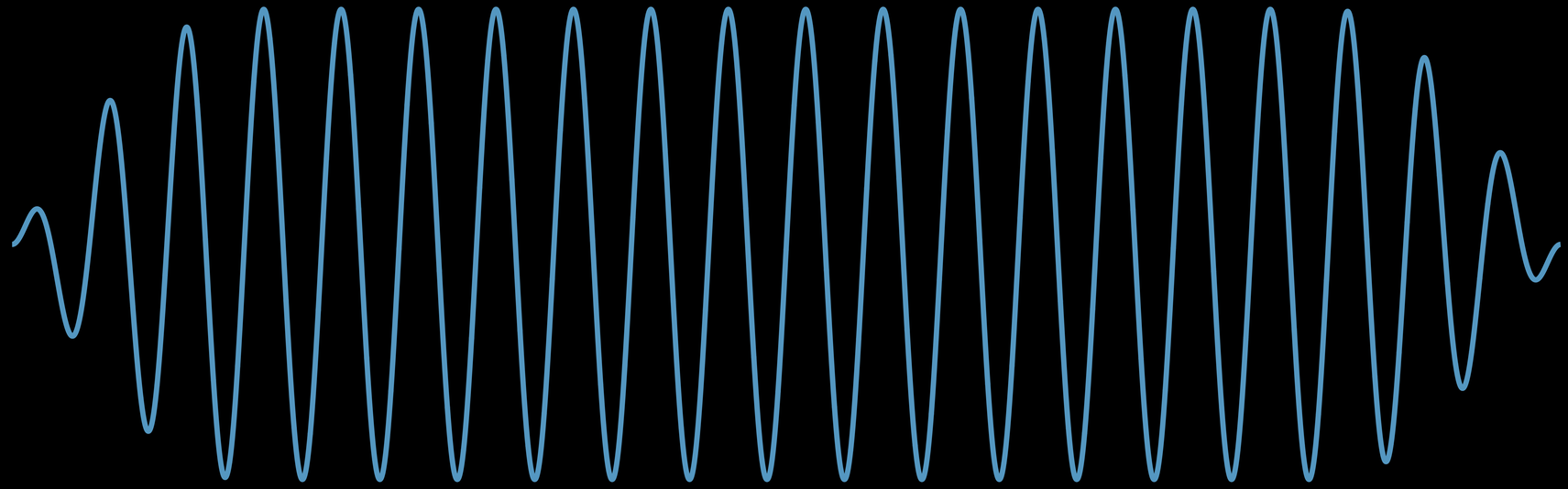

TAPTIC ENGINE


TAPTIC ENGINE

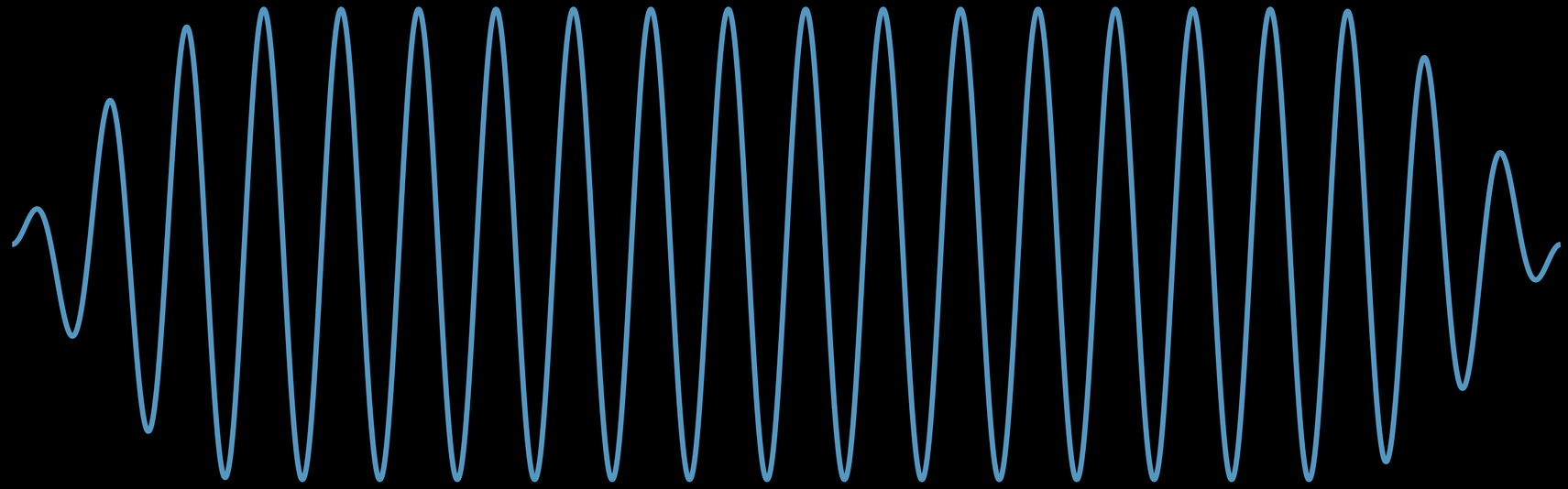
Core Haptics

Haptic Design 101

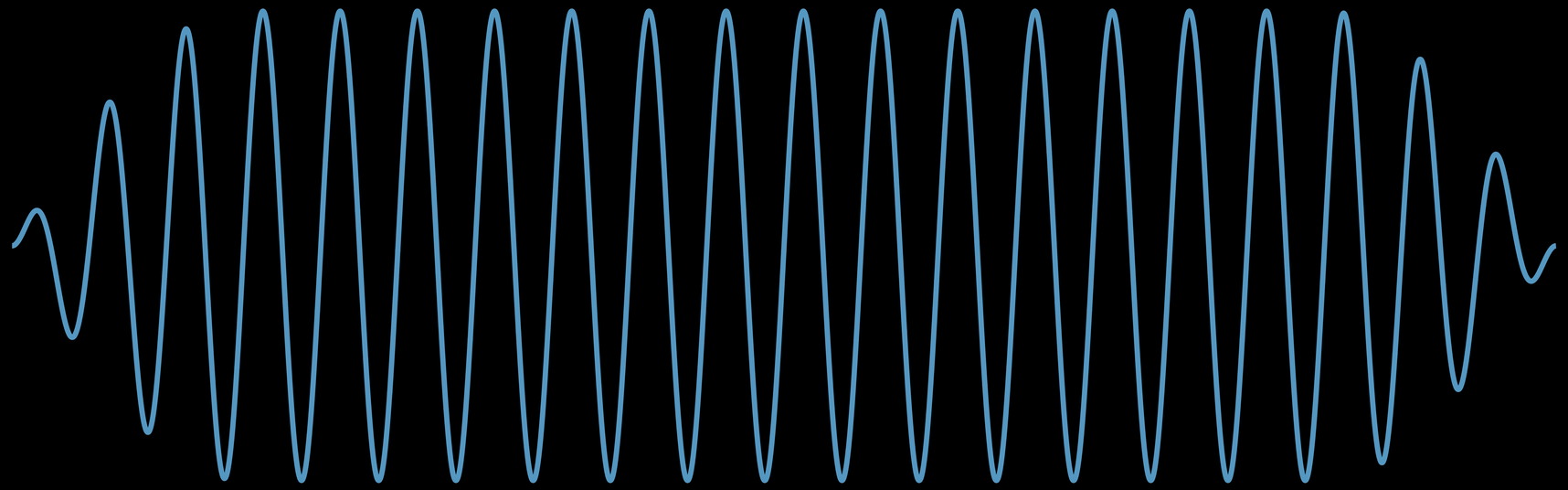
Haptic Design 101



Haptic Design 101

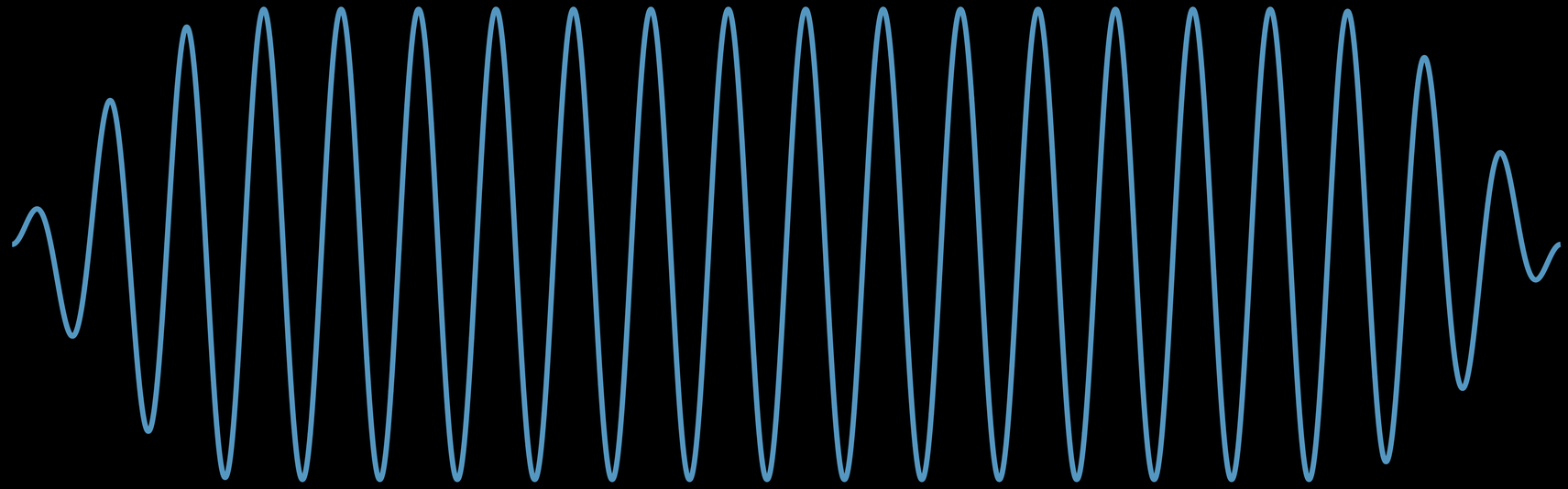


Haptic Design 101

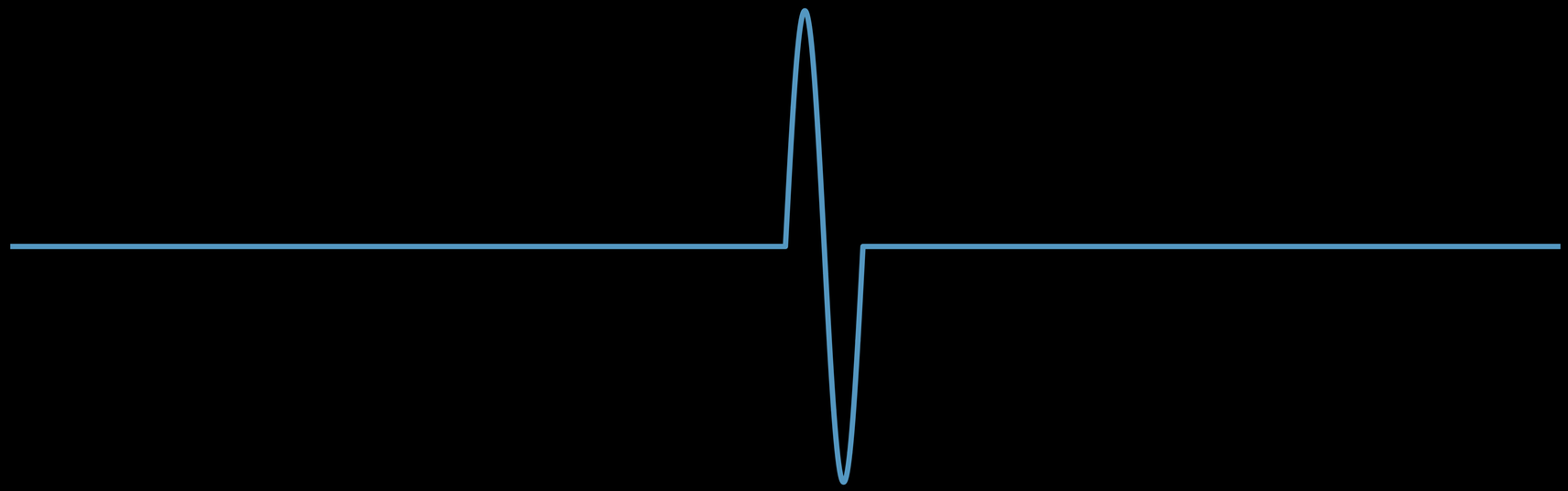


Continuous

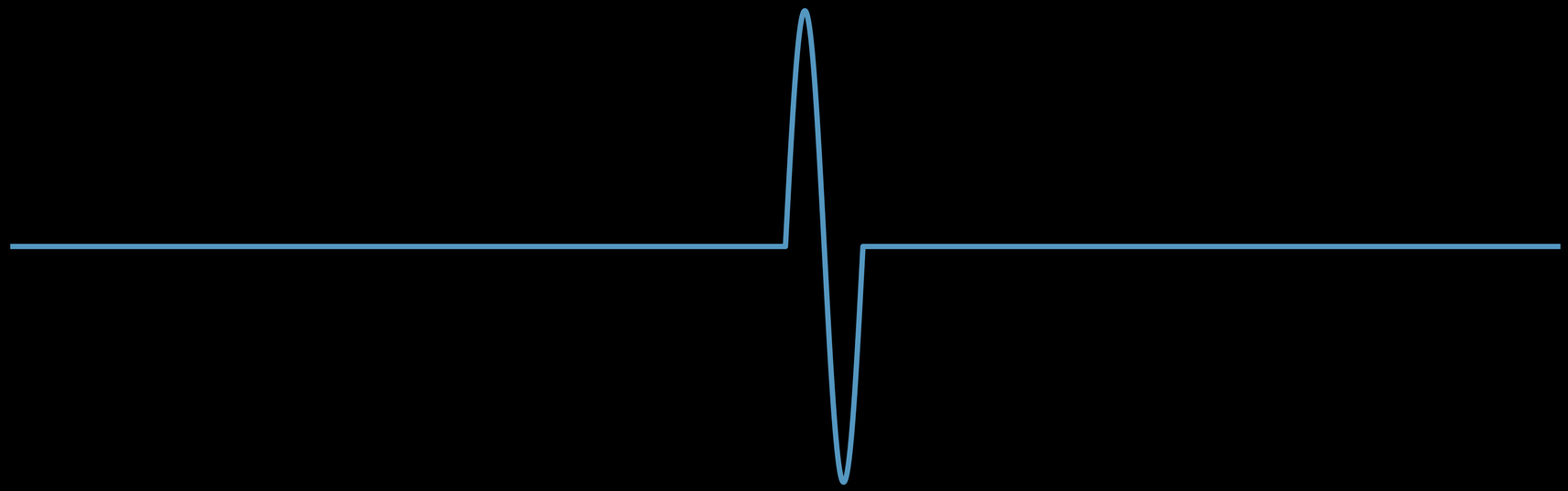
Haptic Design 101



Haptic Design 101

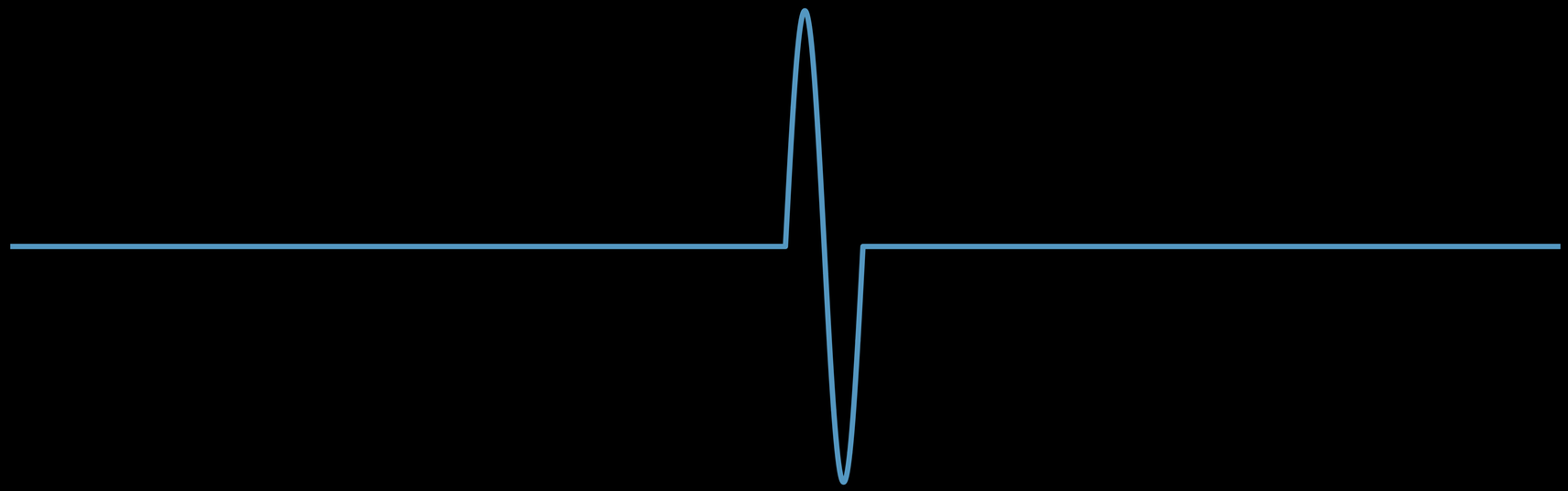


Haptic Design 101



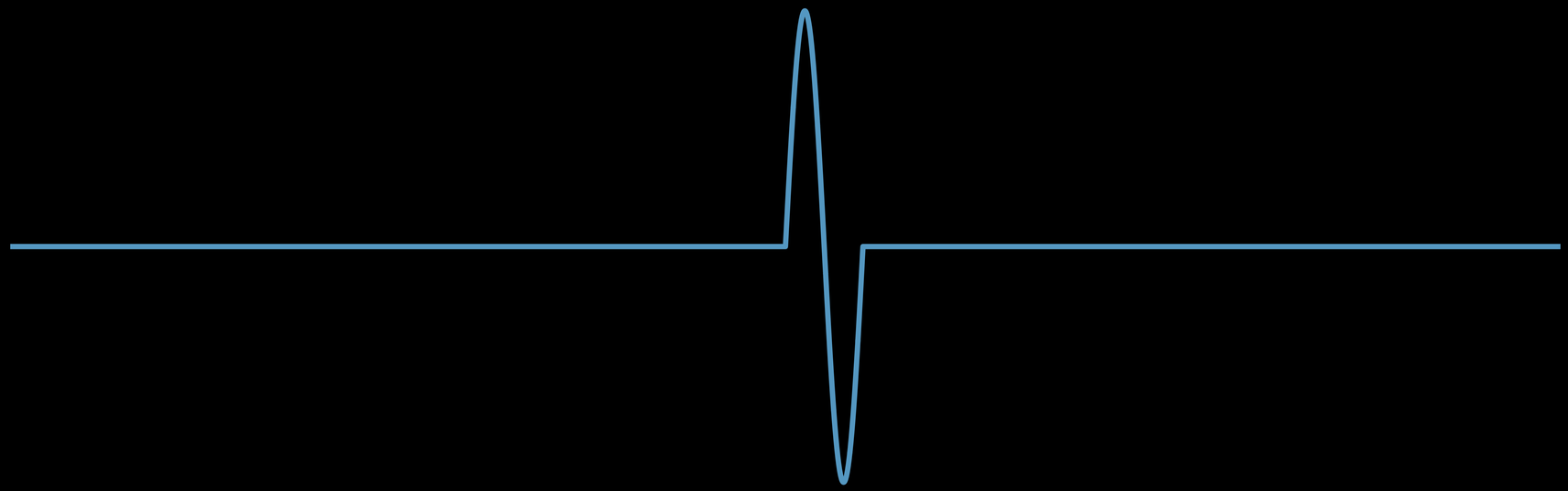
Transient

Haptic Design 101



Transient

Haptic Design 101



Transient

Haptic Design 101

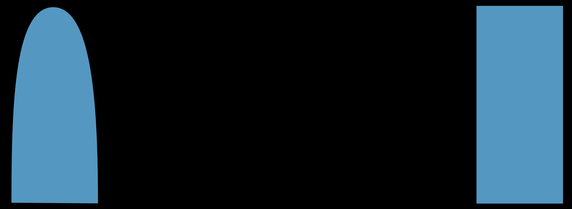
Transient

Haptic Design 101

Transient

Haptic Design 101

Round, Soft



Transient

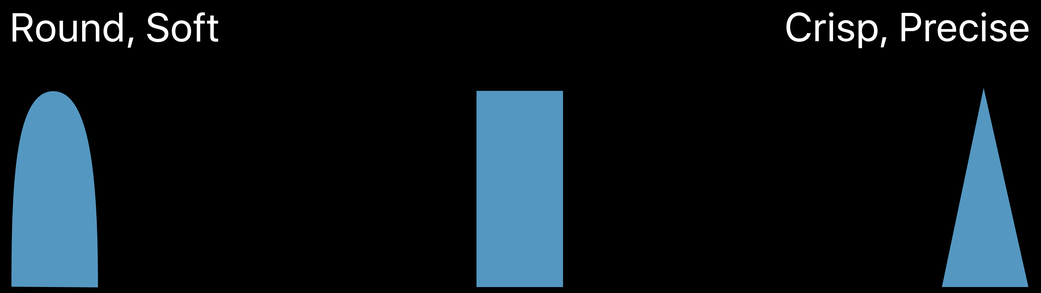
Haptic Design 101

Round, Soft

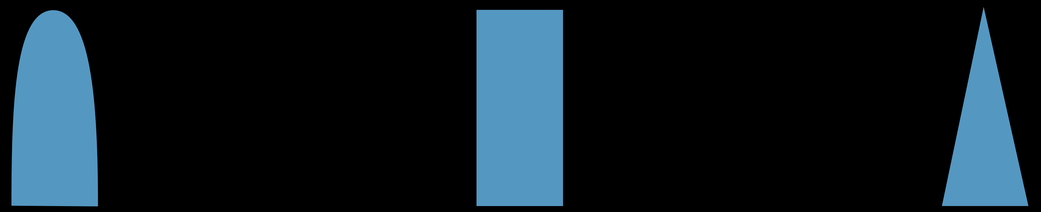
Crisp, Precise

Transient

Haptic Design 101

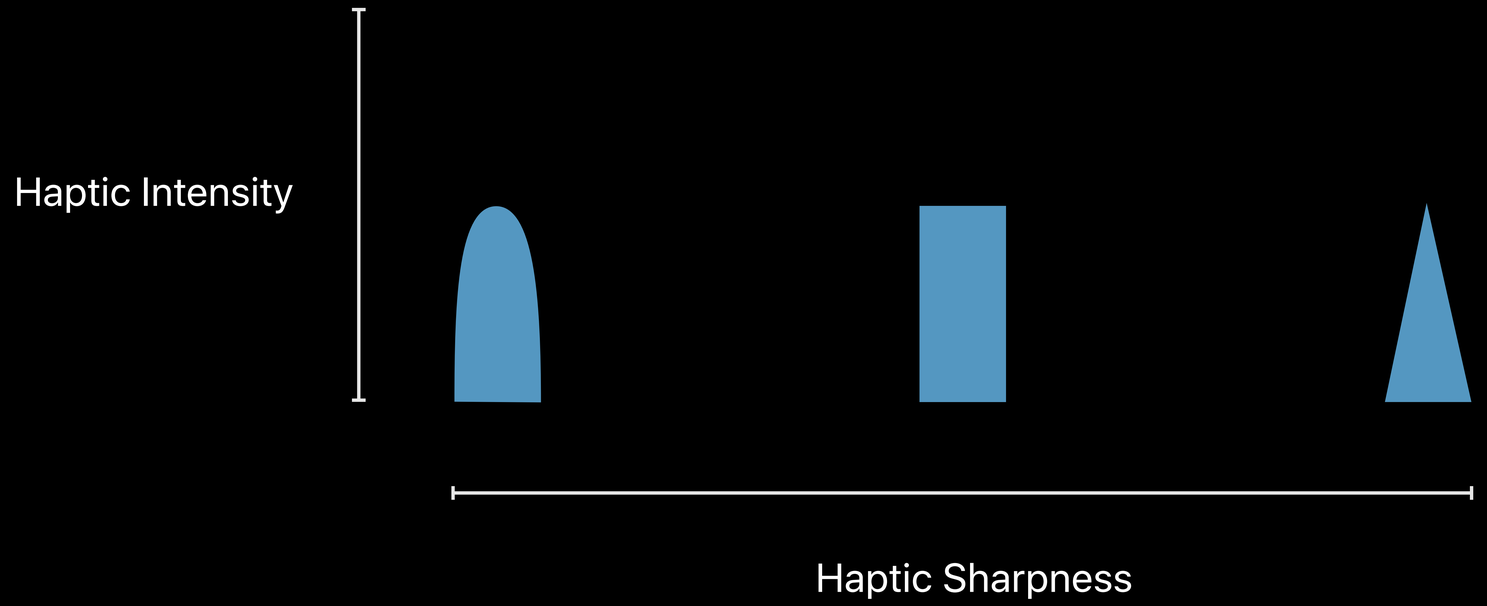


Haptic Design 101



Haptic Design 101

Haptic Design 101



Haptic Design 101

What is an Audio Haptic experience?

Three guiding principles

Techniques

Three Guiding Principles

Three Guiding Principles

Causality

Three Guiding Principles

Causality

Harmony

Three Guiding Principles

Causality

Harmony

Utility

Causality

Causality

“For feedback to be useful, it must be obvious what caused it.”

Cause

Effect

Cause Foot colliding with the ball

Effect

Cause Foot colliding with the ball

Effect Sound of impact
Feel of impact

Cause and Effect

Cause and Effect

Qualities of interacting objects

Cause and Effect

Qualities of interacting objects

Dynamics of the interaction

Cause and Effect

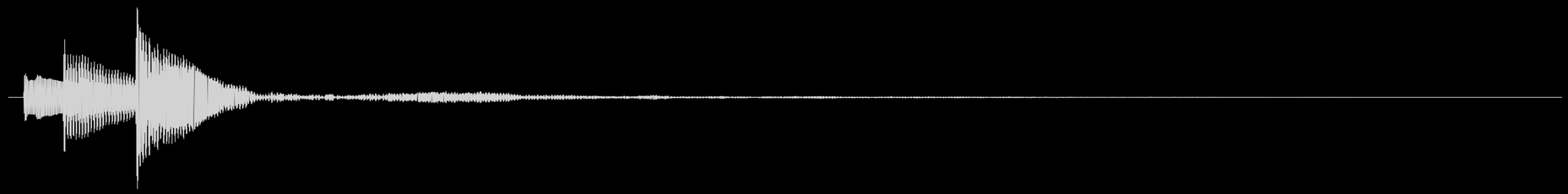
Qualities of interacting objects

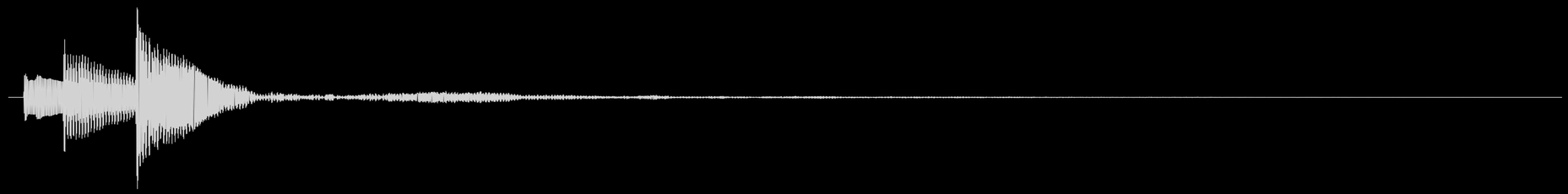
Dynamics of the interaction

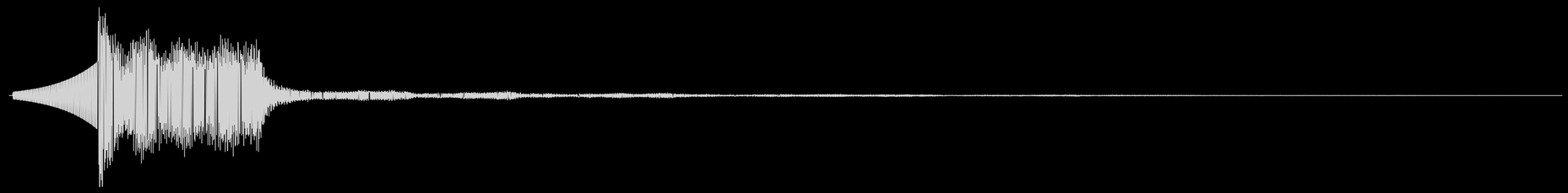
Environment

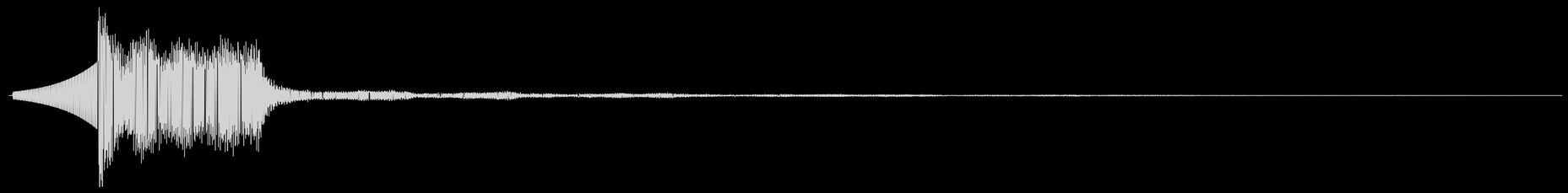

DEBIT
VISA
•••• 1234

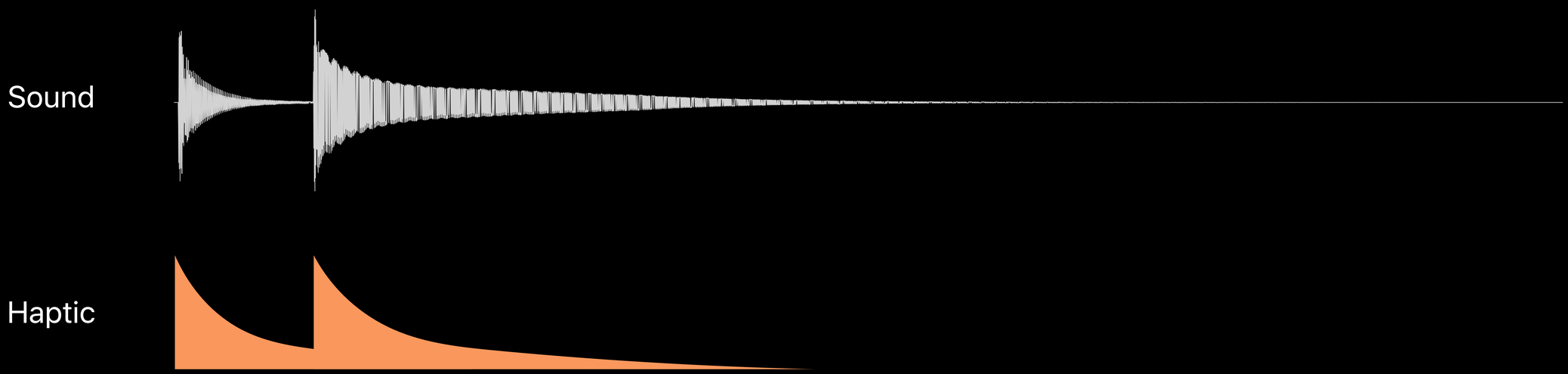

DEBIT
VISA
•••• 1234

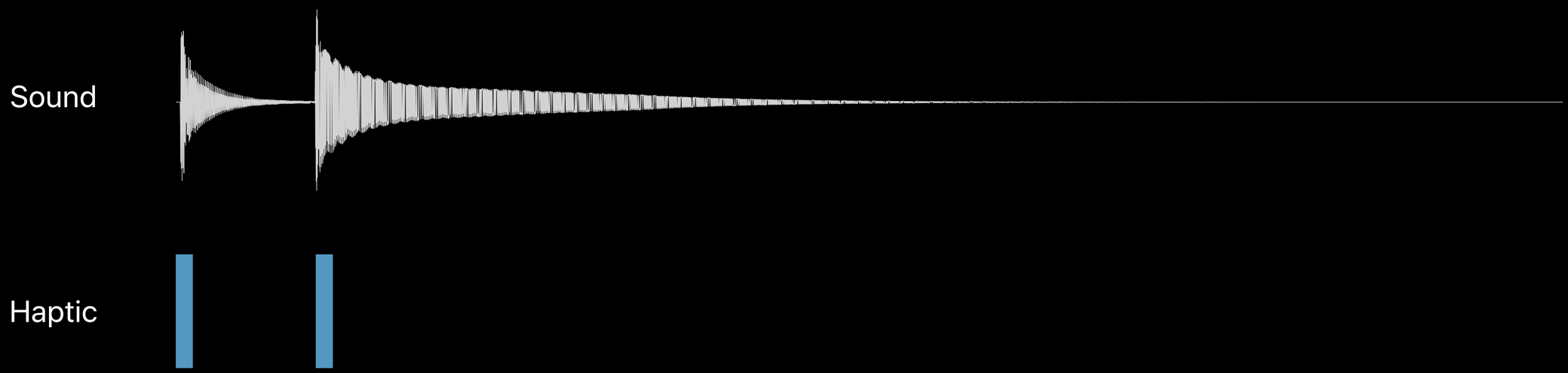


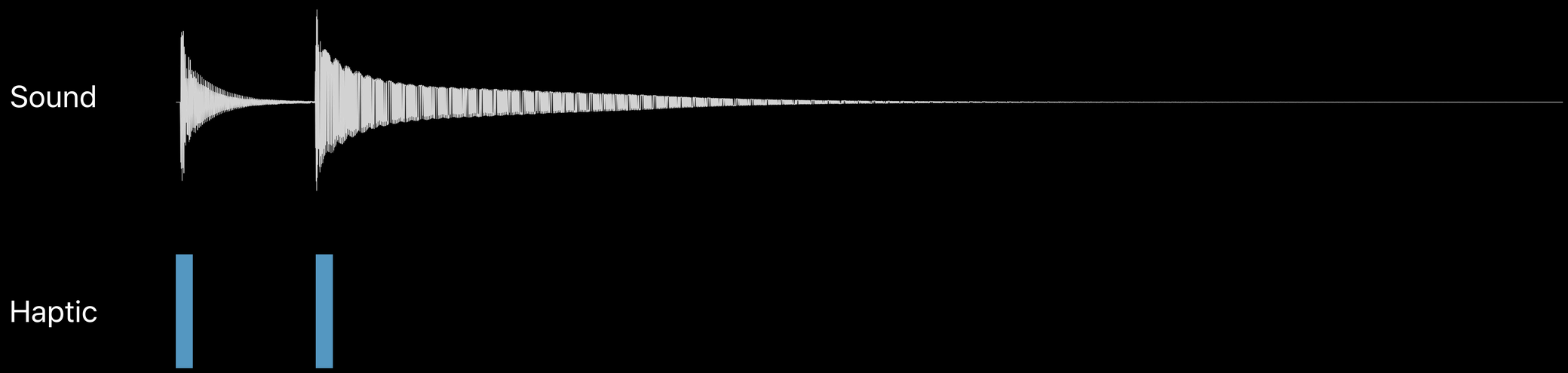







Sound
Haptic


Sound
Haptic


Sound
Haptic


DEBIT
VISA
•••• 1234


DEBIT
VISA
•••• 1234

Harmony

Harmony

“It feels the way it looks the way it sounds.”

Real World

Real World

Visual
Audio
Haptic

Real World

Visual
Audio
Haptic

Digital World

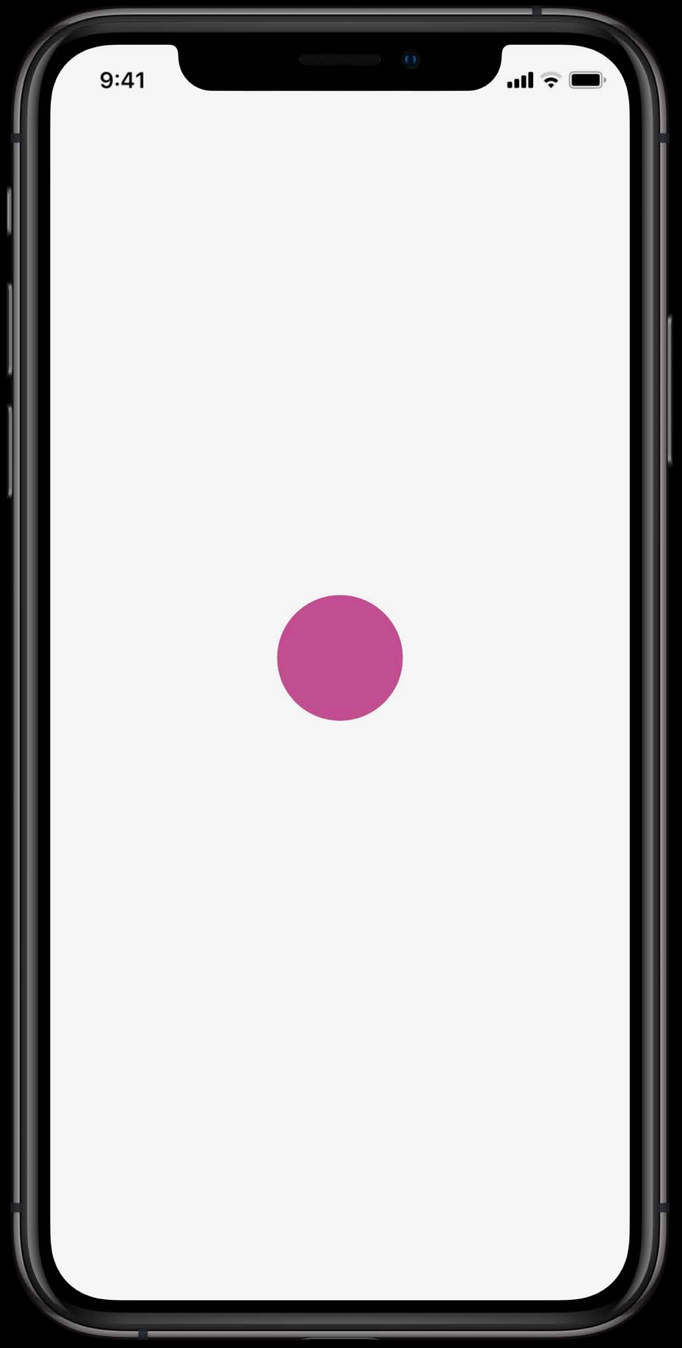
Real World

Visual
Audio
Haptic

Digital World

Real World

Visual
Audio
Haptic

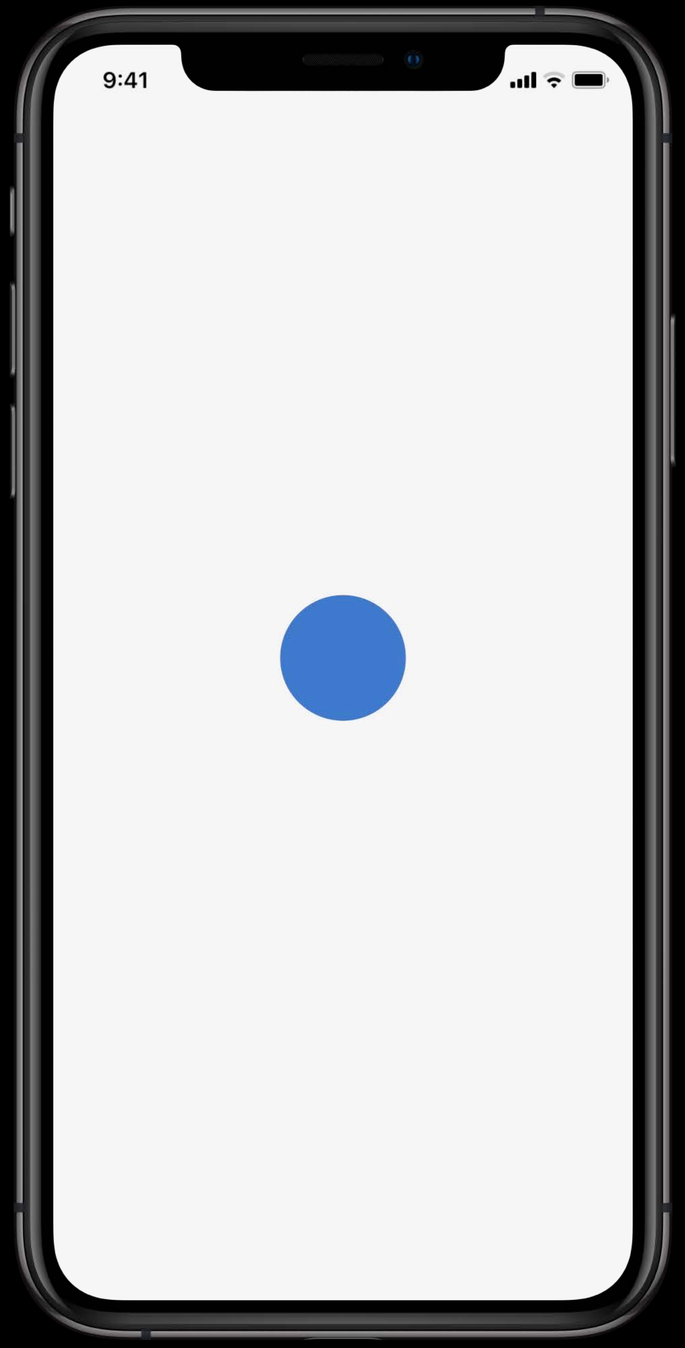


Digital World

Visual

Real World

Visual
Audio
Haptic

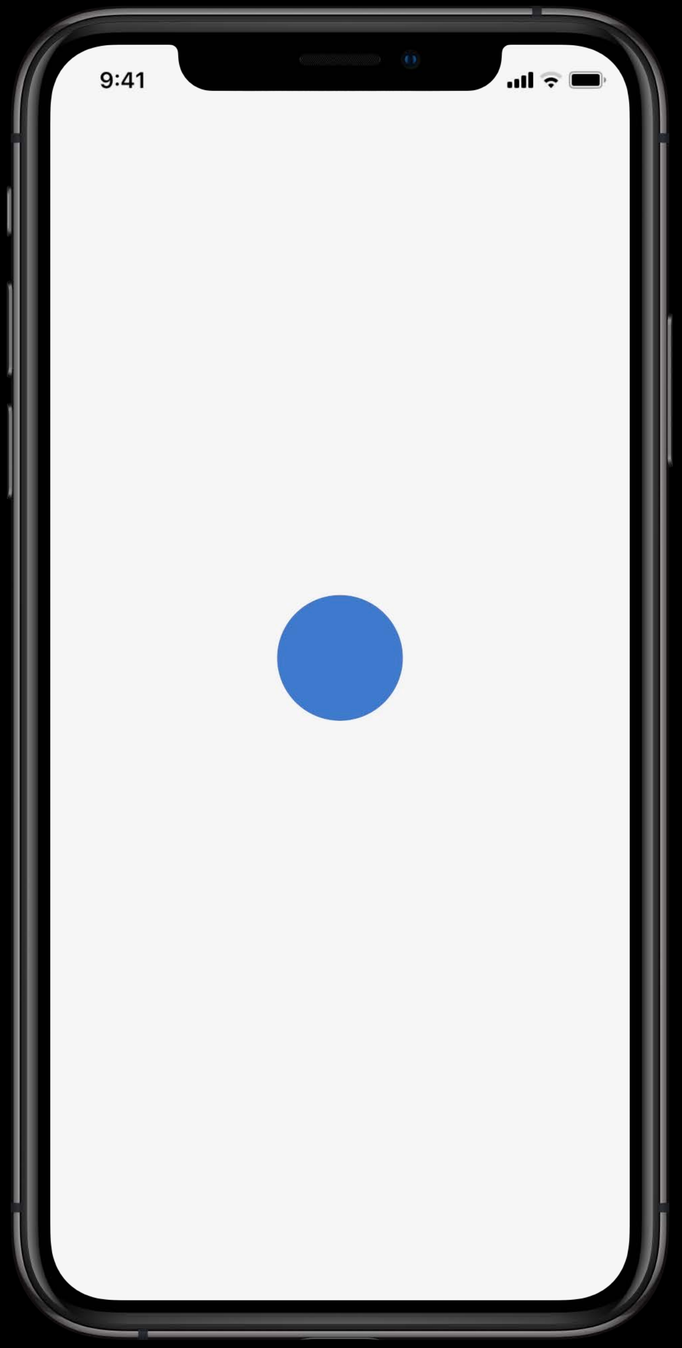


Digital World

Visual

Real World

Visual
Audio
Haptic

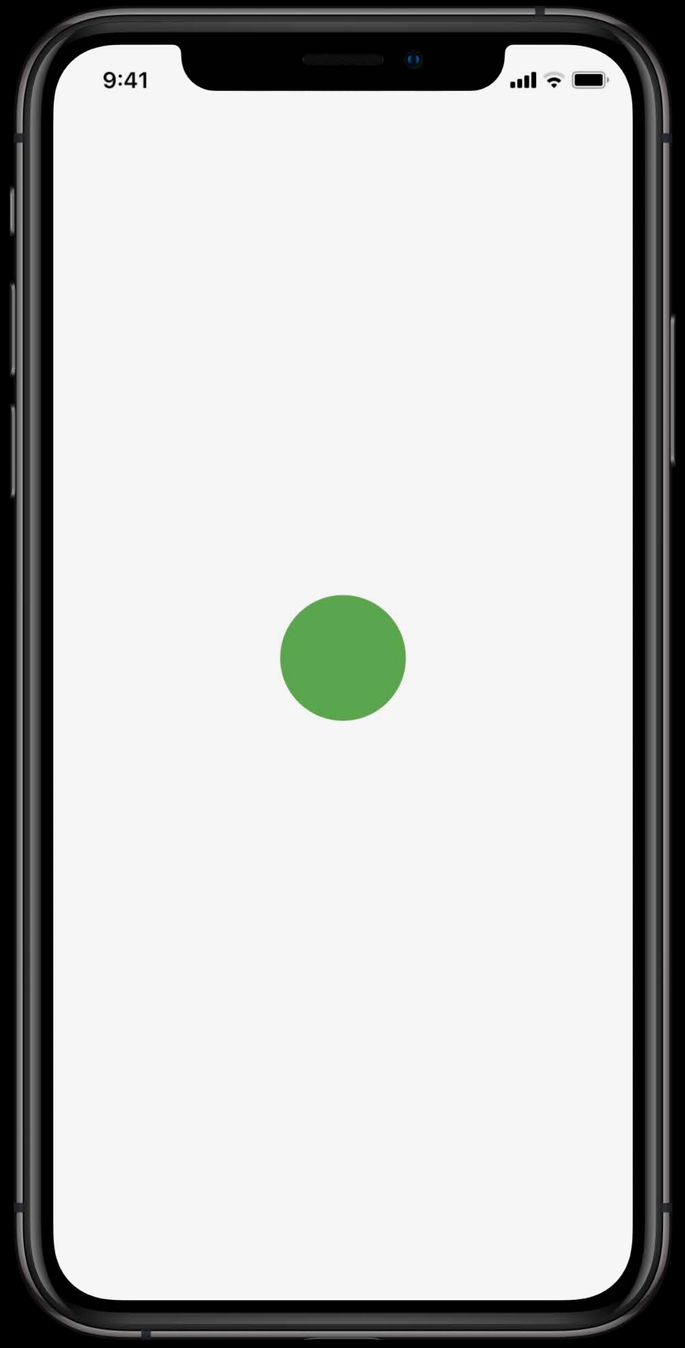


Digital World

Visual
Audio

Real World

Visual
Audio
Haptic

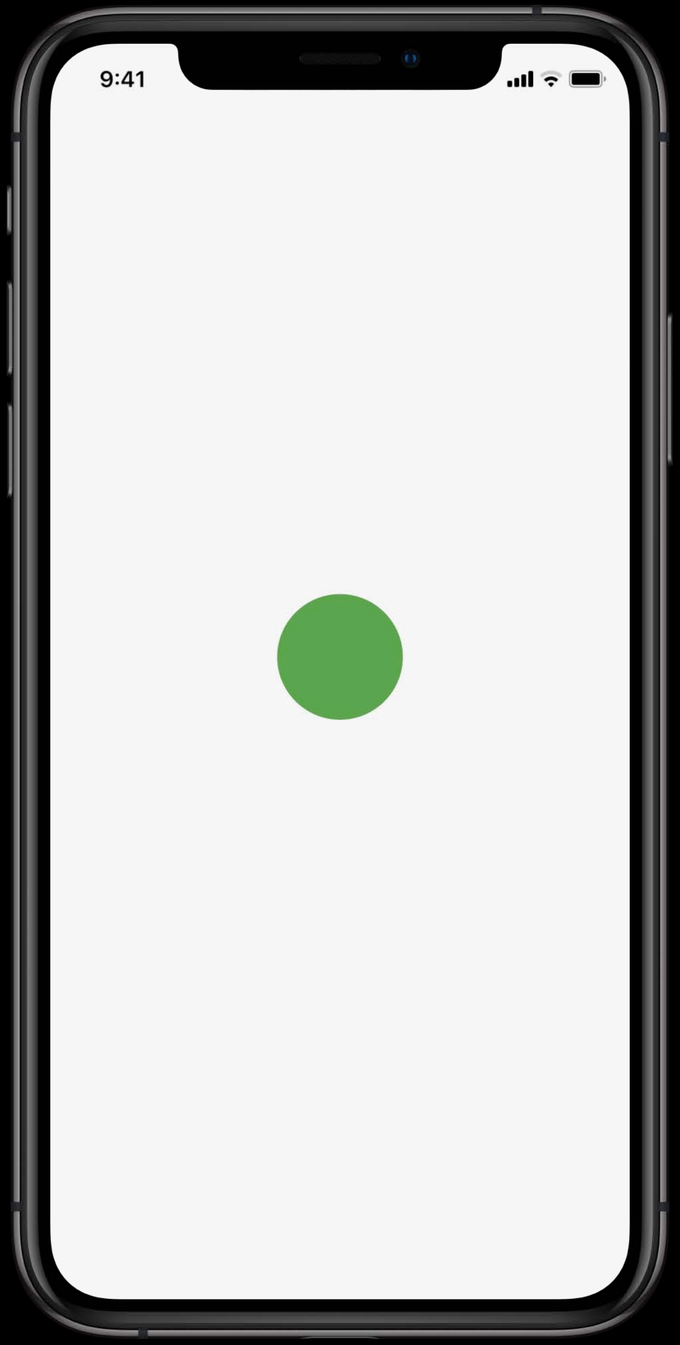


Digital World

Visual
Audio

Real World

Visual
Audio
Haptic

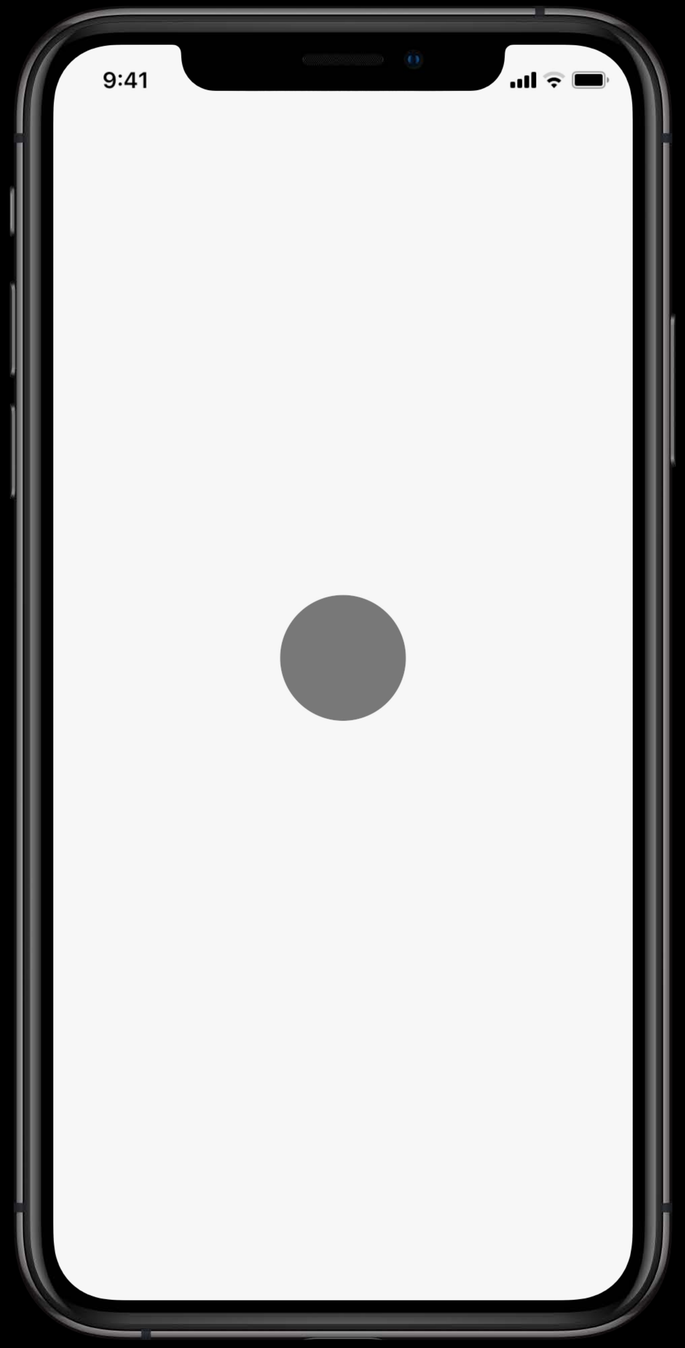


Digital World

Visual
Audio
Haptic

Real World

Visual
Audio
Haptic

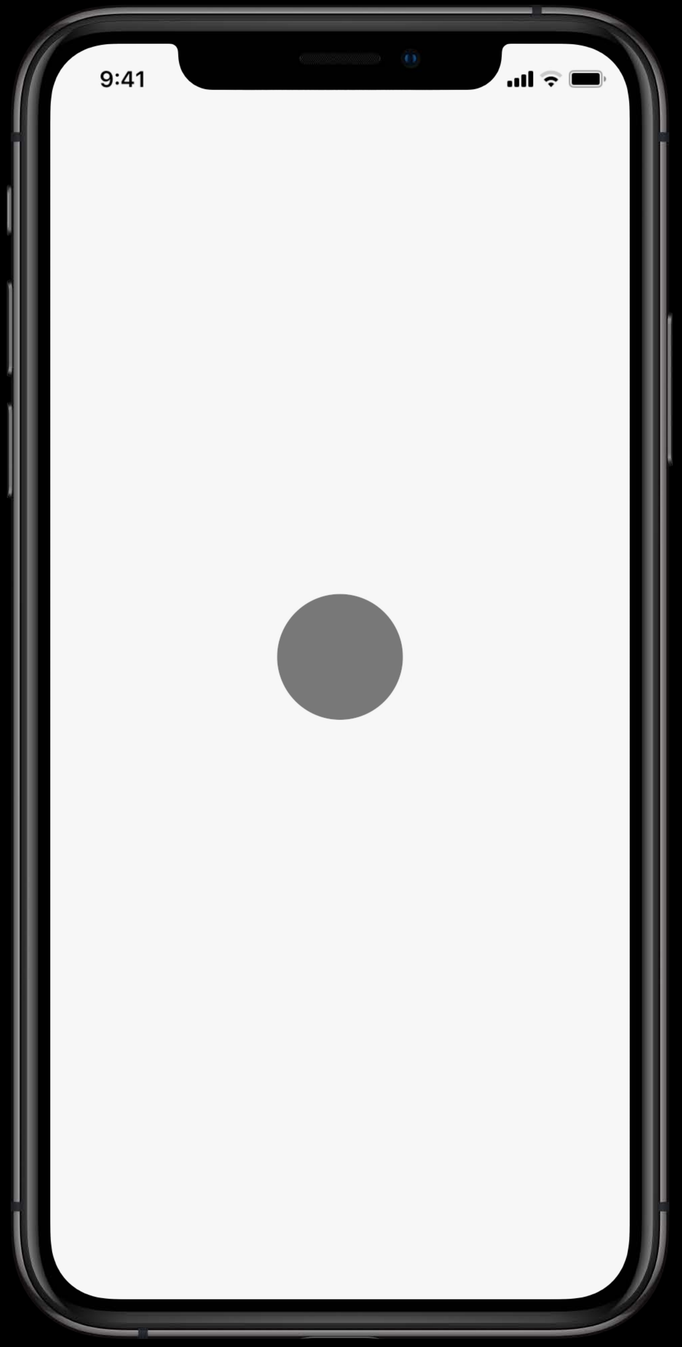


Digital World

Visual
Audio
Haptic

Real World

Visual
Audio
Haptic

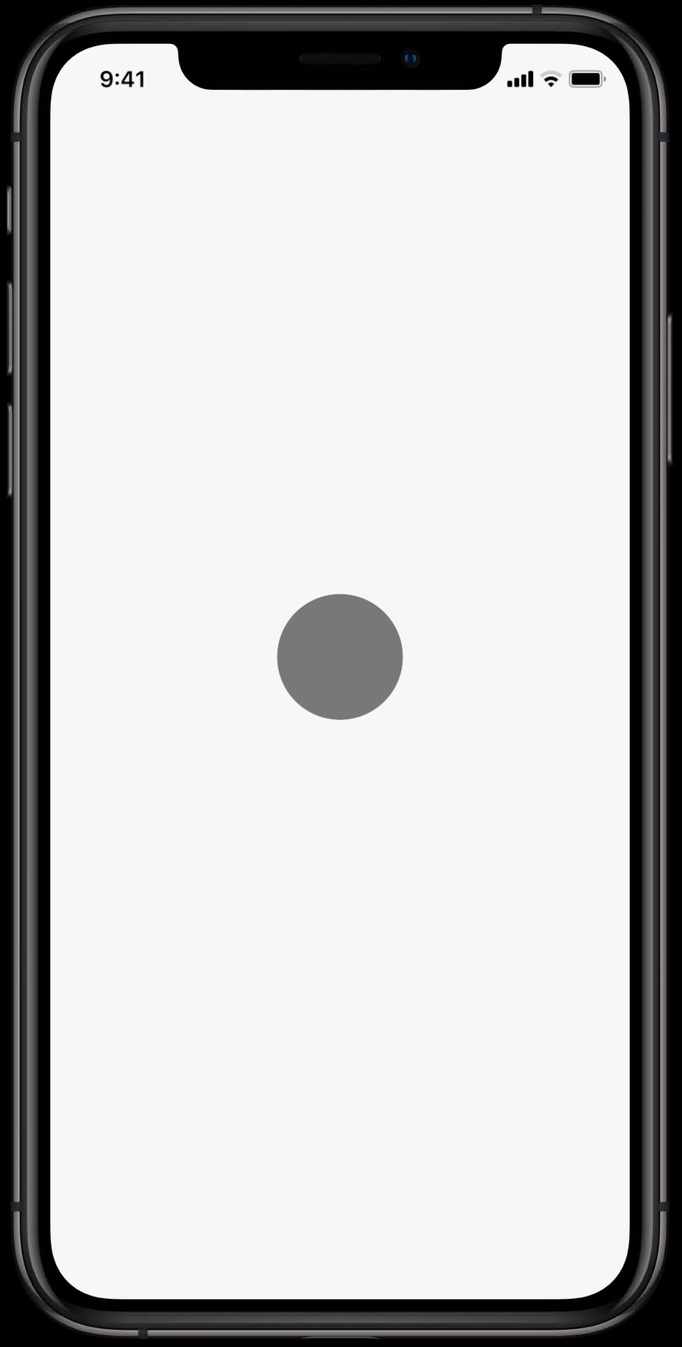


Digital World

Visual
Audio
Haptic
Synchronization

Real World

Visual
Audio
Haptic



Digital World

Visual
Audio
Haptic
Synchronization

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

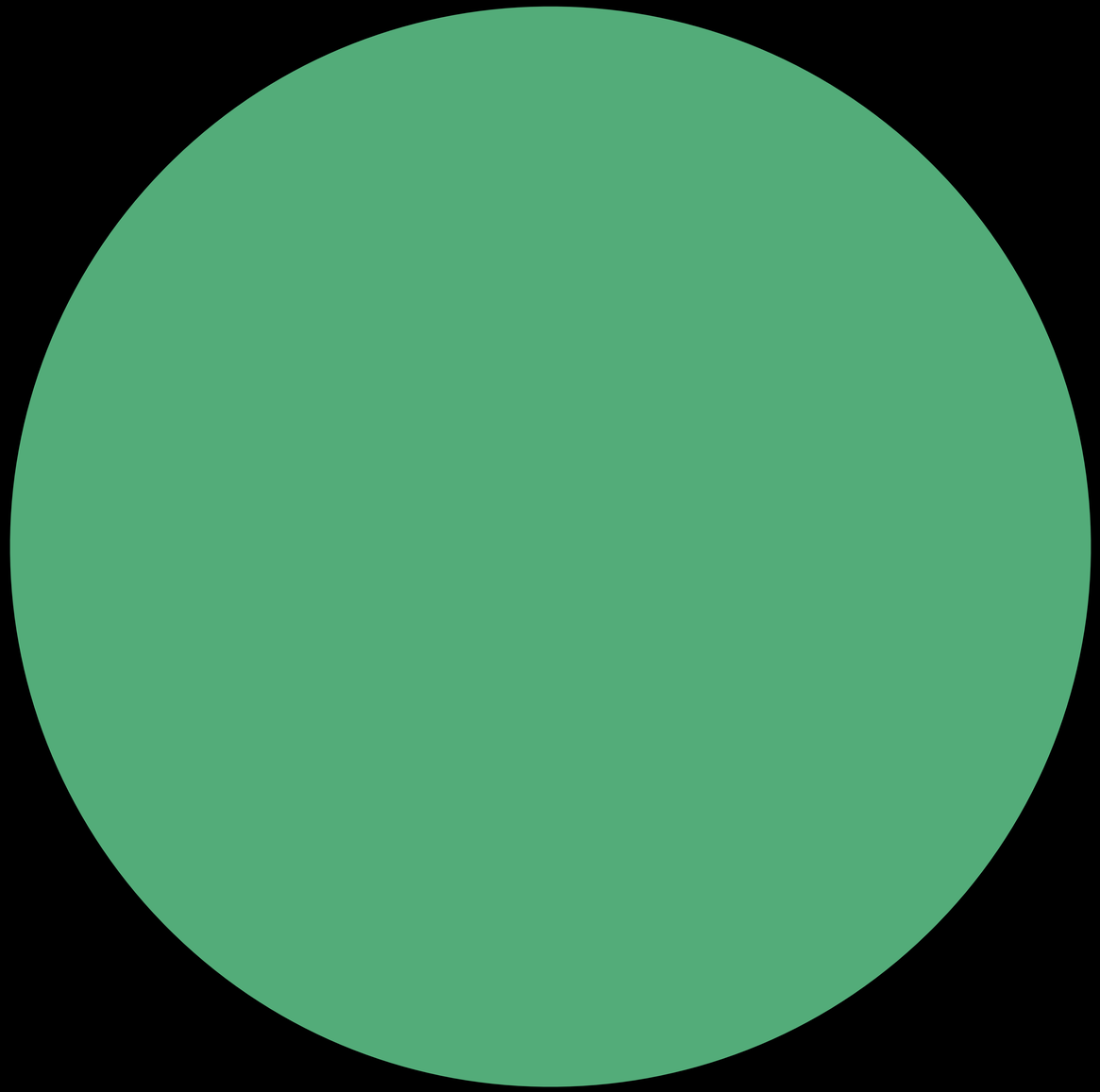
Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics



Harmony

Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics

Harmony

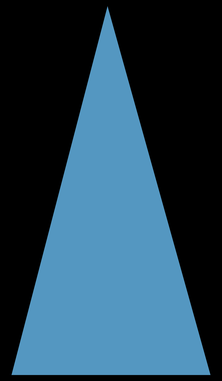
Interactions, Visuals, Audio and Haptics

Harmony

Interactions, Visuals, Audio and Haptics


14:59
11:15AM TRAINING WITH KRISTA
52
68°
89
WED
12
UVI
9.3
1:09PM, +3HRS
NYC

Digital Crown Haptic



Digital Crown Haptic


Activity
Alarms
Breathe
Tue
28
Calendar


Activity
Alarms
Breathe
Tue
28
Calendar


Activity
Alarms
Breathe
Tue
28
Calendar


Activity
Alarms
Breathe
Tue
28
Calendar

Utility

Utility

“Add audio and haptics that provide clear value to your app experience”

Utility

Moderation

Utility

Moderation

Focus

Utility

Moderation

Focus

Keep it simple

Guiding Principles

Guiding Principles

Causality

Guiding Principles

Causality

Harmony

Guiding Principles

Causality

Harmony

Utility

What is an Audio Haptic experience?

Three guiding principles

Techniques

Core Haptics Primitives

Core Haptics Primitives

Transient

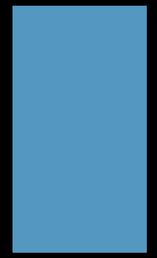
Core Haptics Primitives

Transient

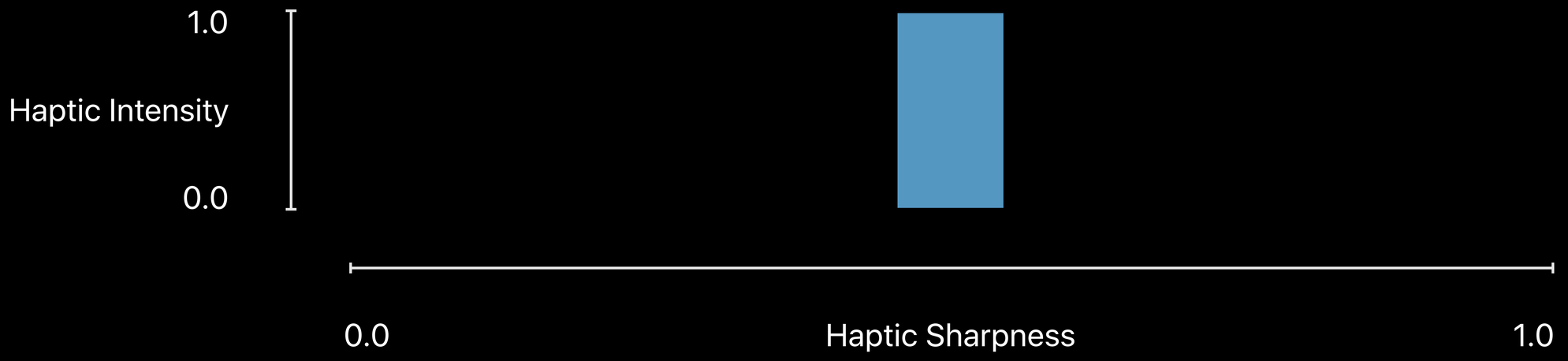
Continuous

Core Haptics Transient

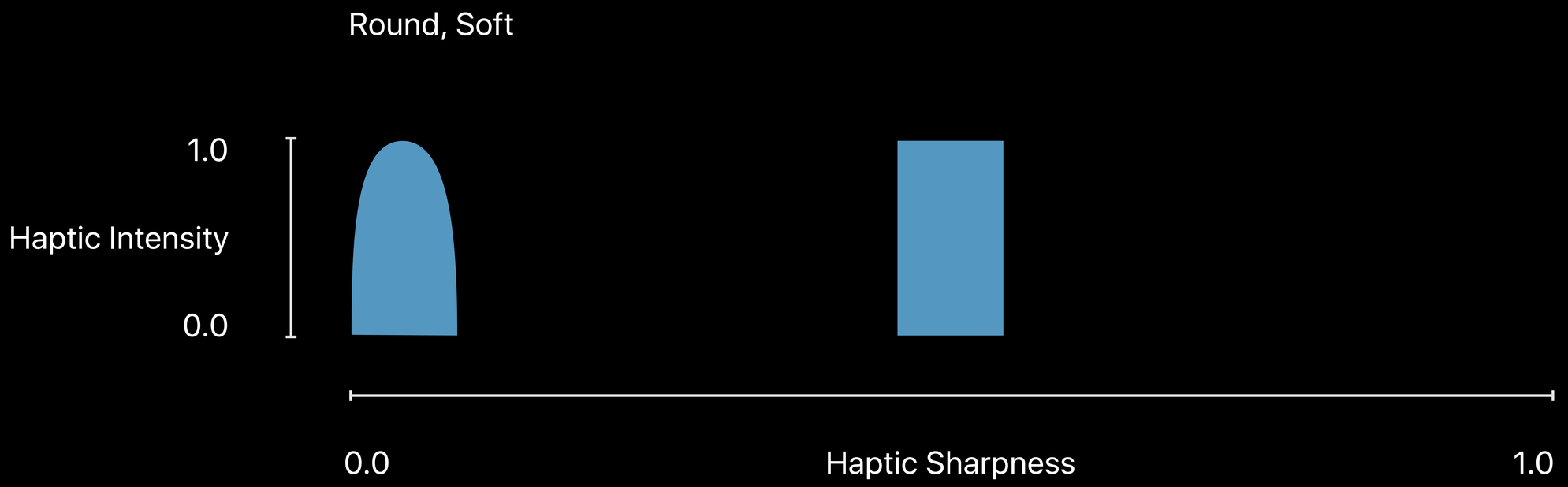
Core Haptics Transient



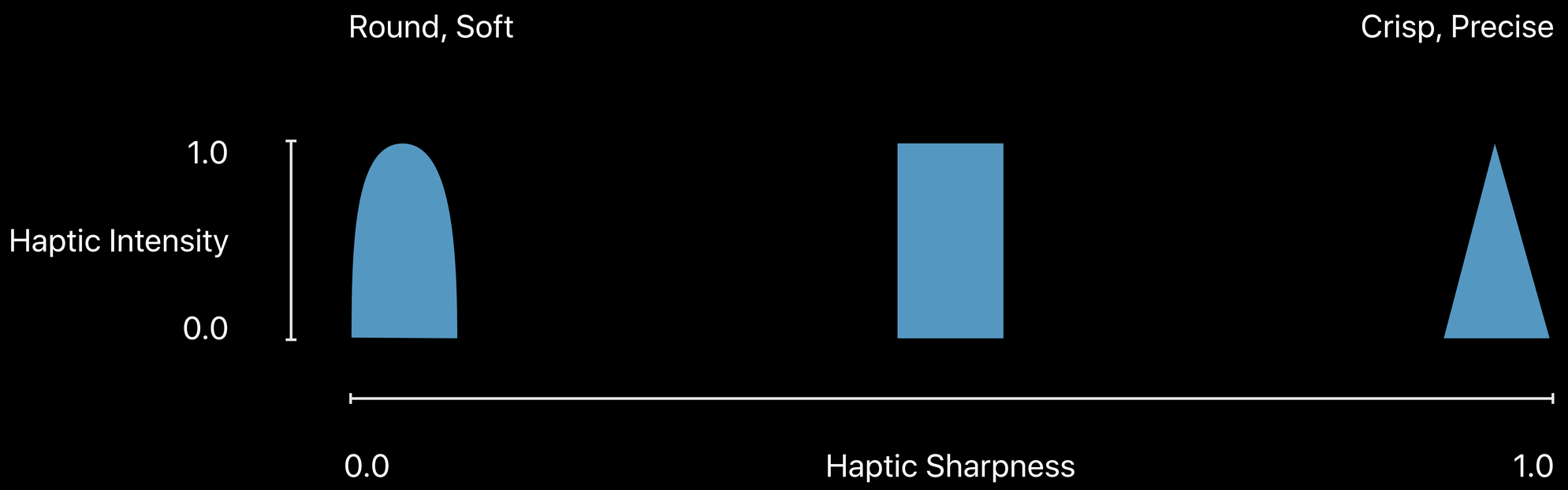
Core Haptics Transient



Core Haptics Transient



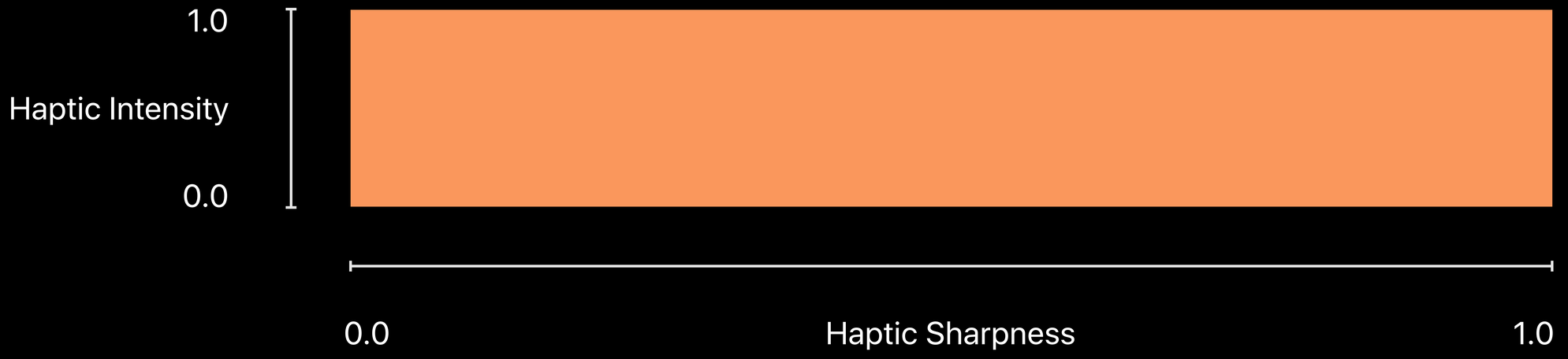
Core Haptics Transient



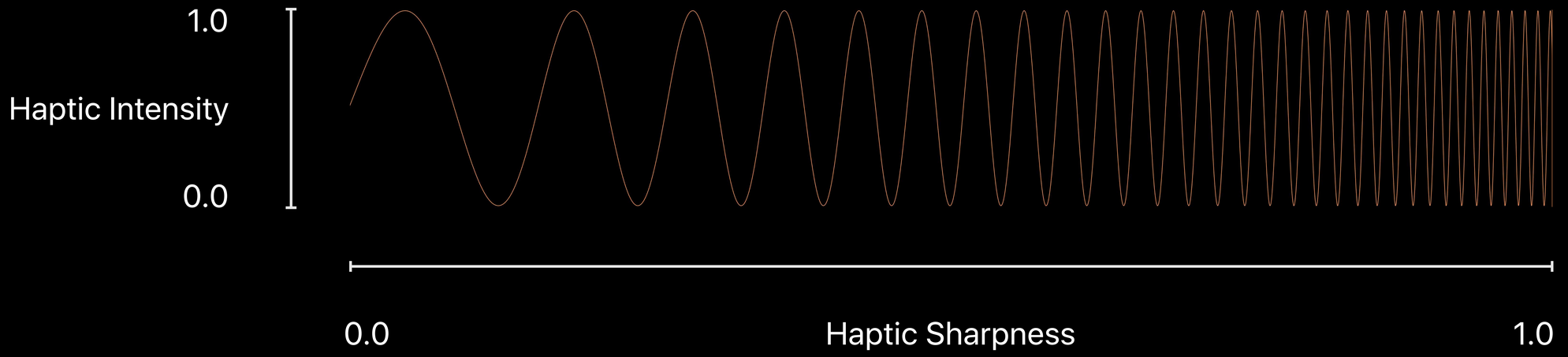
Core Haptics Continuous

Core Haptics Continuous

Core Haptics Continuous

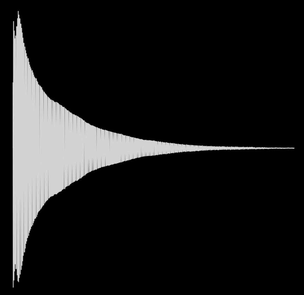


Core Haptics Continuous



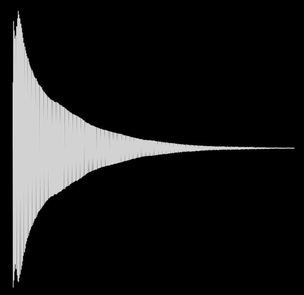
Sound Building Blocks

Sound Building Blocks

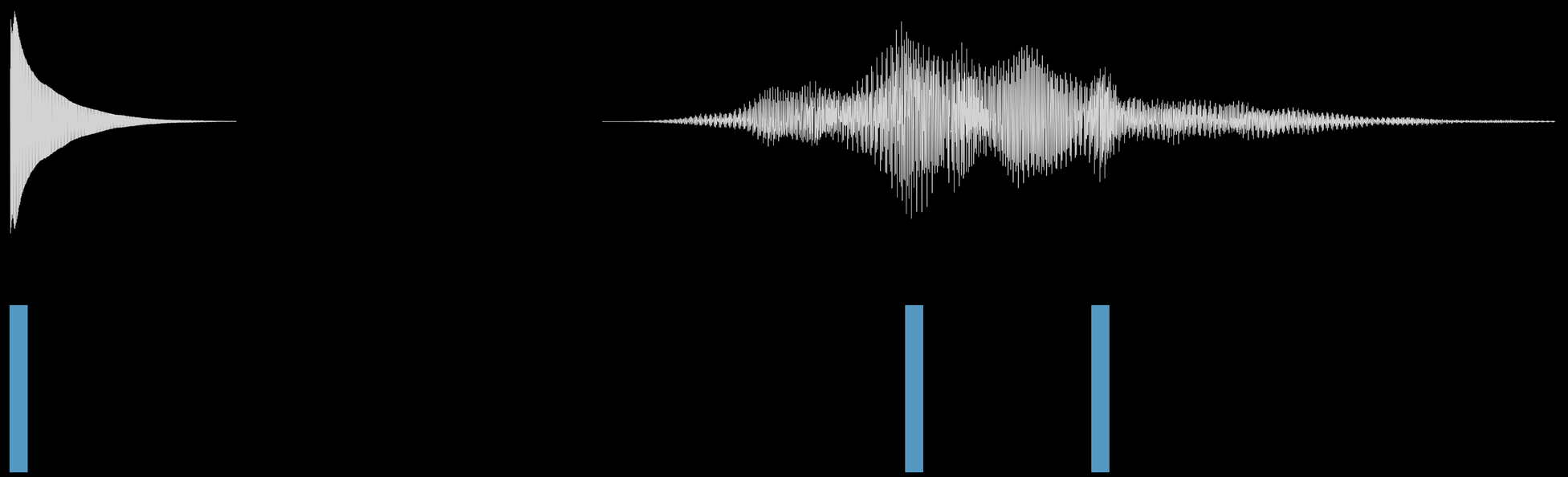


Sound Building Blocks

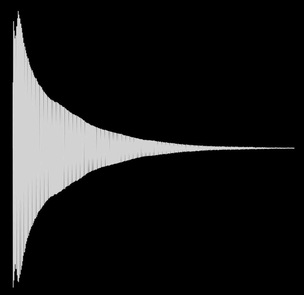
Sound Building Blocks

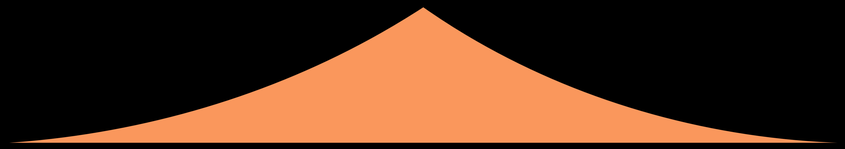


Sound Building Blocks

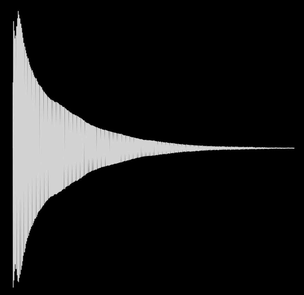


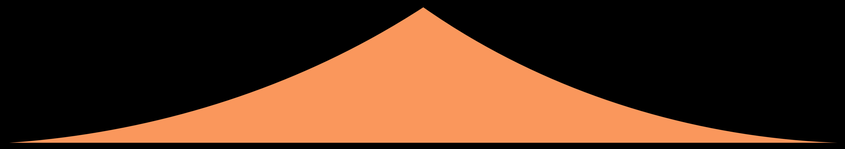
Sound Building Blocks





Sound Building Blocks



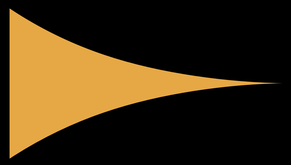


Sound

Sound

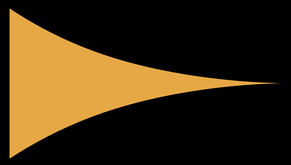
Sound

Haptic



Sound

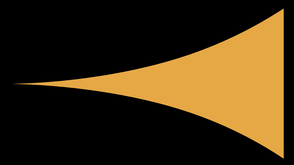
Haptic



Anticipation

Sound

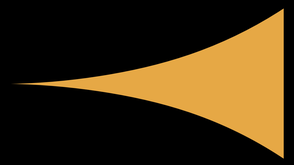
Haptic



Anticipation

Sound

Haptic



Number of Events

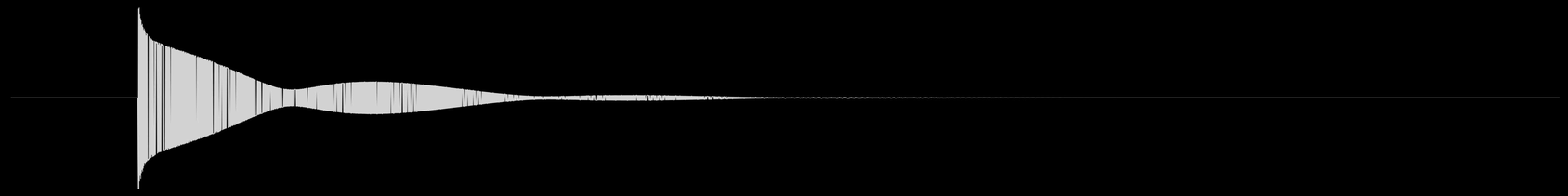
4 Events

Number of Events

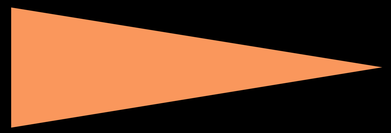
4 Events

Haptic Priming

Sound

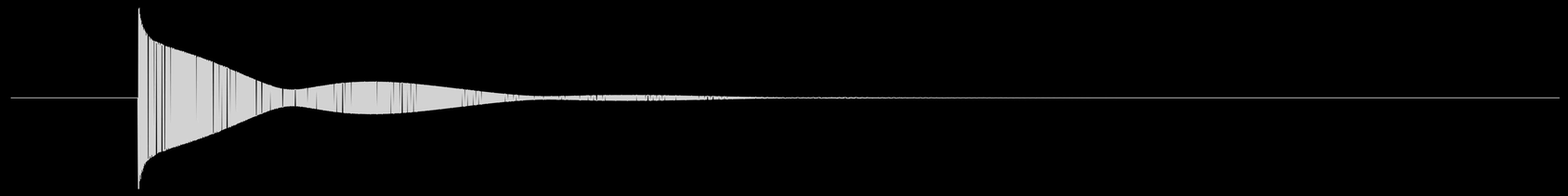


Haptic

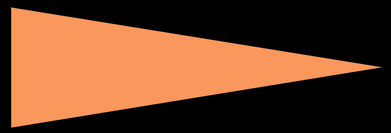


Haptic Priming

Sound

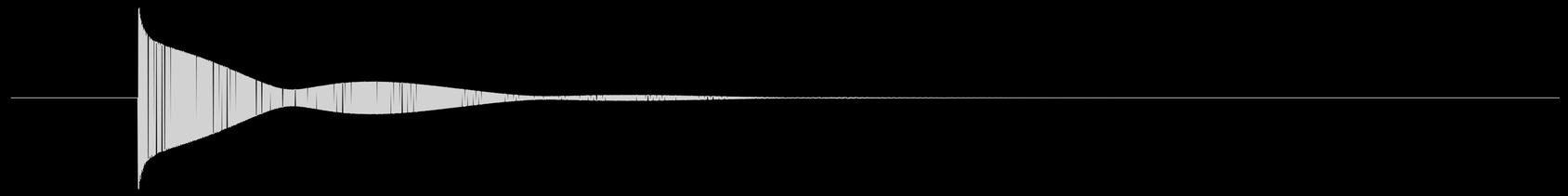


Haptic

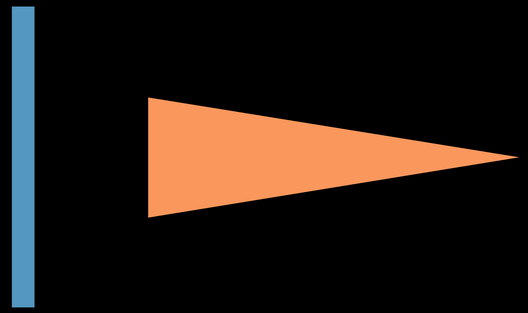


Haptic Priming

Sound

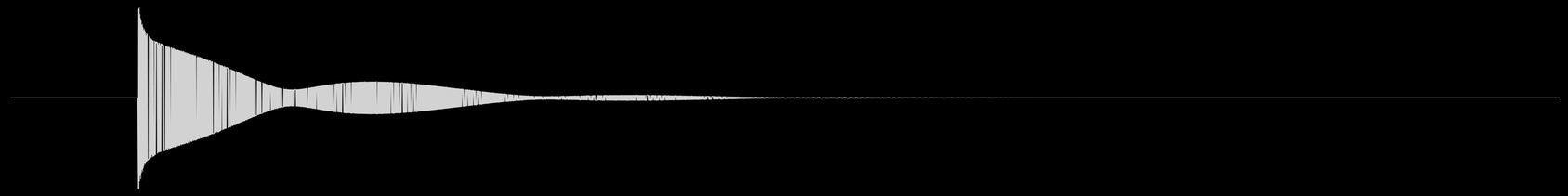


Haptic

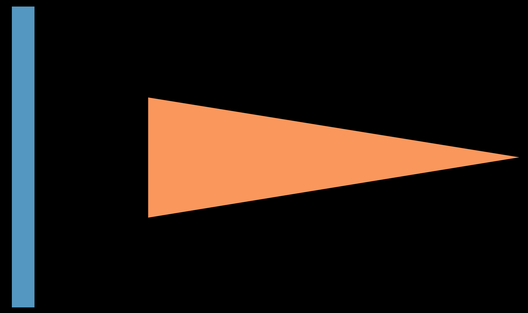


Haptic Priming

Sound

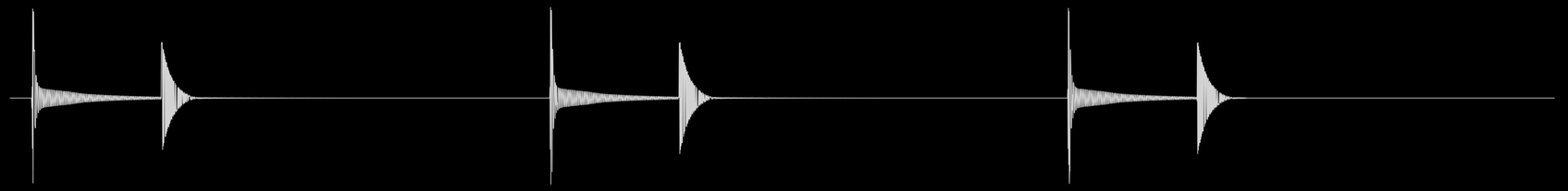


Haptic



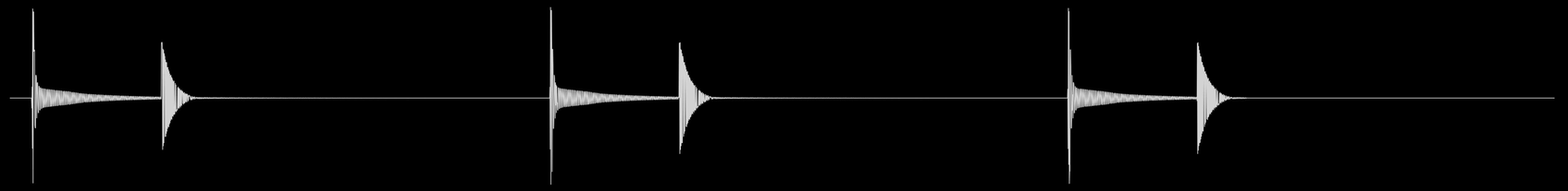
Contrast—Left

Sound



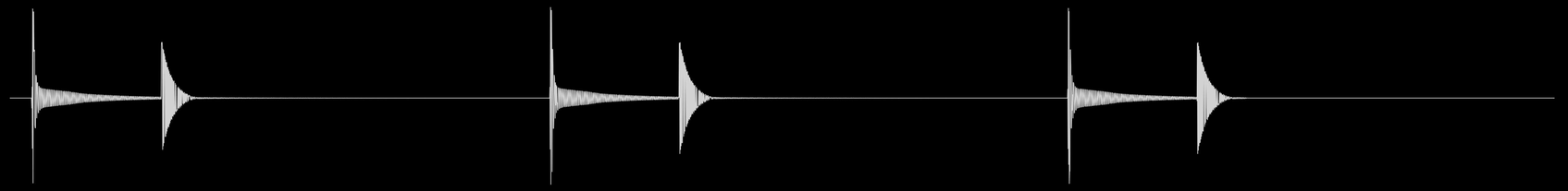
Contrast—Left

Sound

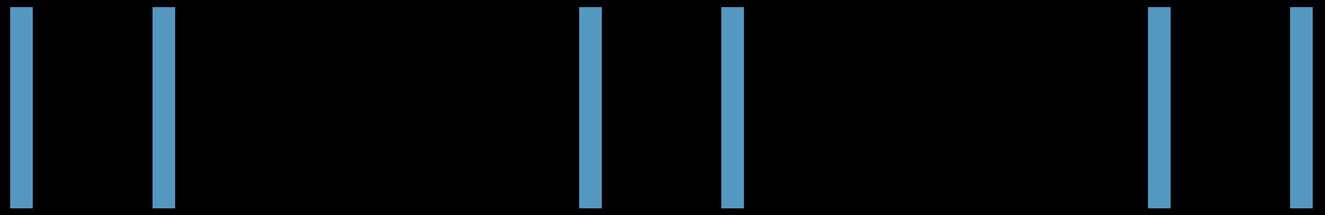


Contrast—Left

Sound

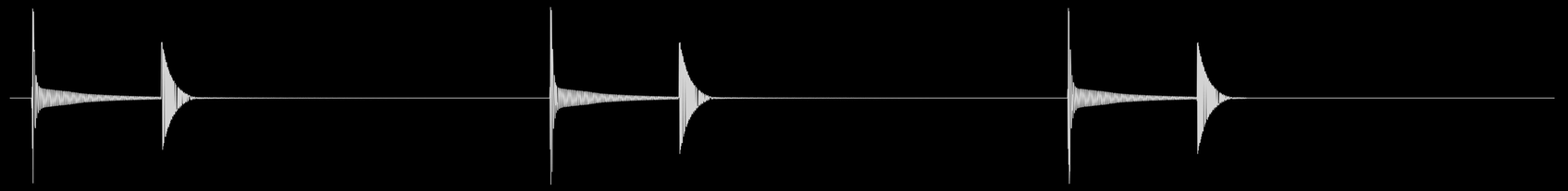


Haptic

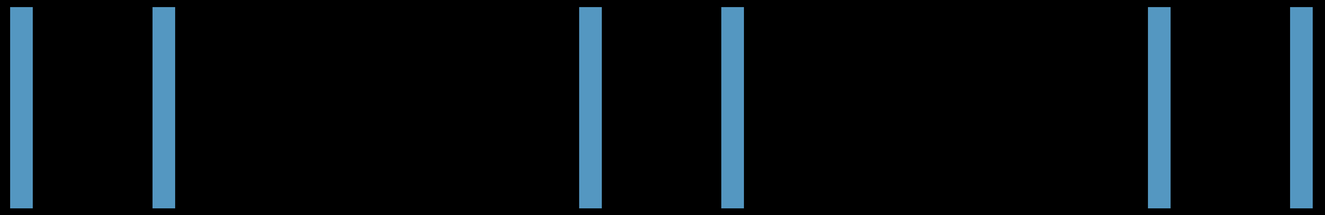


Contrast—Left

Sound

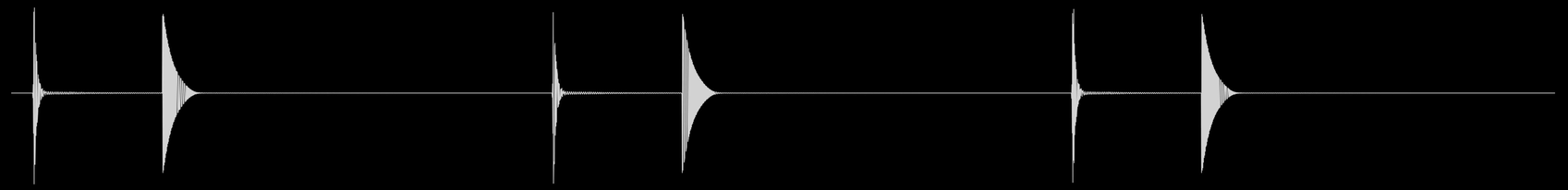


Haptic



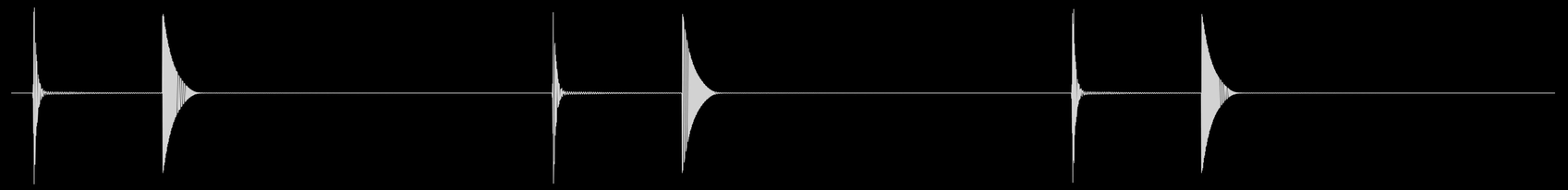
Contrast—Right

Sound



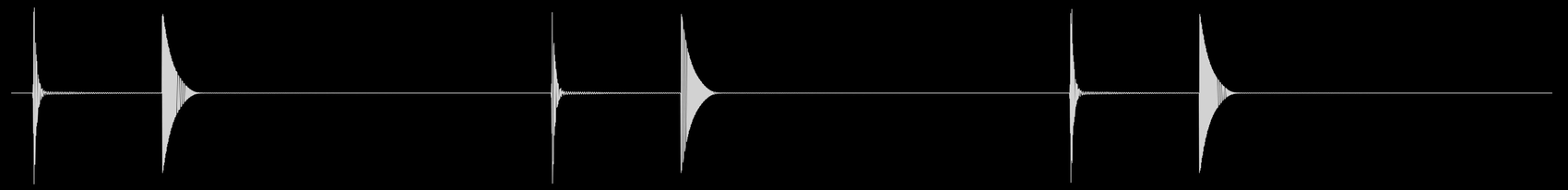
Contrast—Right

Sound



Contrast—Right

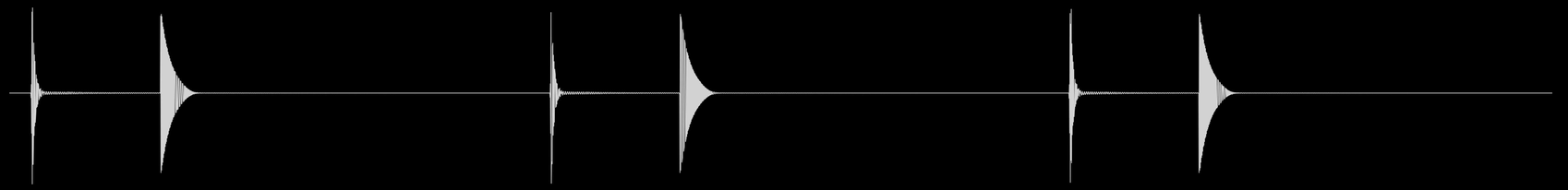
Sound



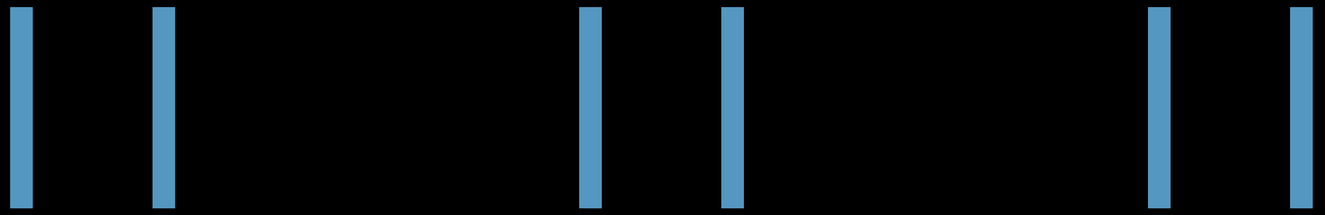
Haptic

Contrast—Right

Sound

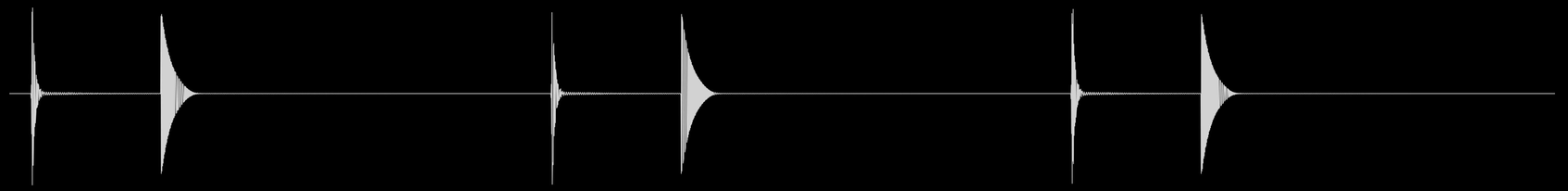


Haptic



Contrast—Right

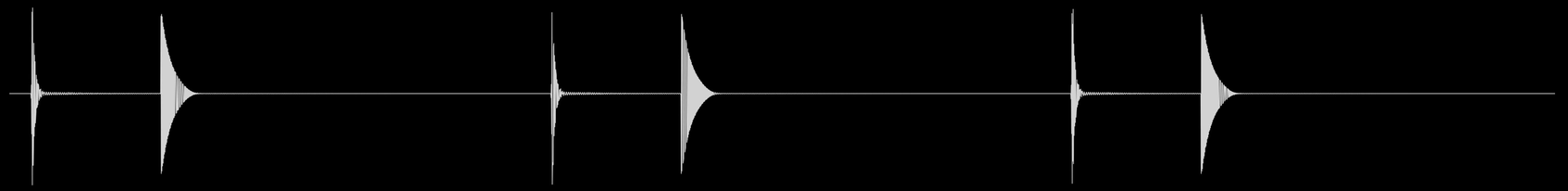
Sound



Haptic

Contrast—Right

Sound



Haptic


12:00
Camille
yesterday. Overall it has been very relaxing!!
Hi Hugo, what are you up to tonight?
Today 8:13 PM
Hey Camille, first dinner with family, then fireworks!!
Fri, May 31, 5:37 PM
Sounds nice! Where are you watching?
Today 8:14 PM
Golden Gate Bridge!! You should join us!
Fri, May 31, 5:38 PM
Sounds like fun! See you there!
Today 8:14 PM
See you then
Delivered
iMessage


12:00
Camille
yesterday. Overall it has been very relaxing!!
Hi Hugo, what are you up to tonight?
Today 8:13 PM
Hey Camille, first dinner with family, then fireworks!!
Fri, May 31, 5:37 PM
Sounds nice! Where are you watching?
Today 8:14 PM
Golden Gate Bridge!! You should join us!
Fri, May 31, 5:38 PM
Sounds like fun! See you there!
Today 8:14 PM
See you then
Delivered
iMessage


12:00
Camille
yesterday. Overall it has been very relaxing!!
Hi Hugo, what are you up to tonight?
Today 8:13 PM
Hey Camille, first dinner with family, then fireworks!!
Fri, May 31, 5:37 PM
Sounds nice! Where are you watching?
Today 8:14 PM
Golden Gate Bridge!! You should join us!
Fri, May 31, 5:38 PM
Sounds like fun! See you there!
Today 8:14 PM
See you then
Delivered
iMessage


12:00
Camille
yesterday. Overall it has been very relaxing!!
Hi Hugo, what are you up to tonight?
Today 8:13 PM
Hey Camille, first dinner with family, then fireworks!!
Fri, May 31, 5:37 PM
Sounds nice! Where are you watching?
Today 8:14 PM
Golden Gate Bridge!! You should join us!
Fri, May 31, 5:38 PM
Sounds like fun! See you there!
Today 8:14 PM
See you then
Delivered
iMessage

A Few More Thoughts

Collaborate

Experience It

Experiment

More Information

developer.apple.com/wwdc19/223

Core Haptics Lab	Thursday, 11:00
Core Haptics Lab (2)	Friday, 9:00

#WWDC19

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

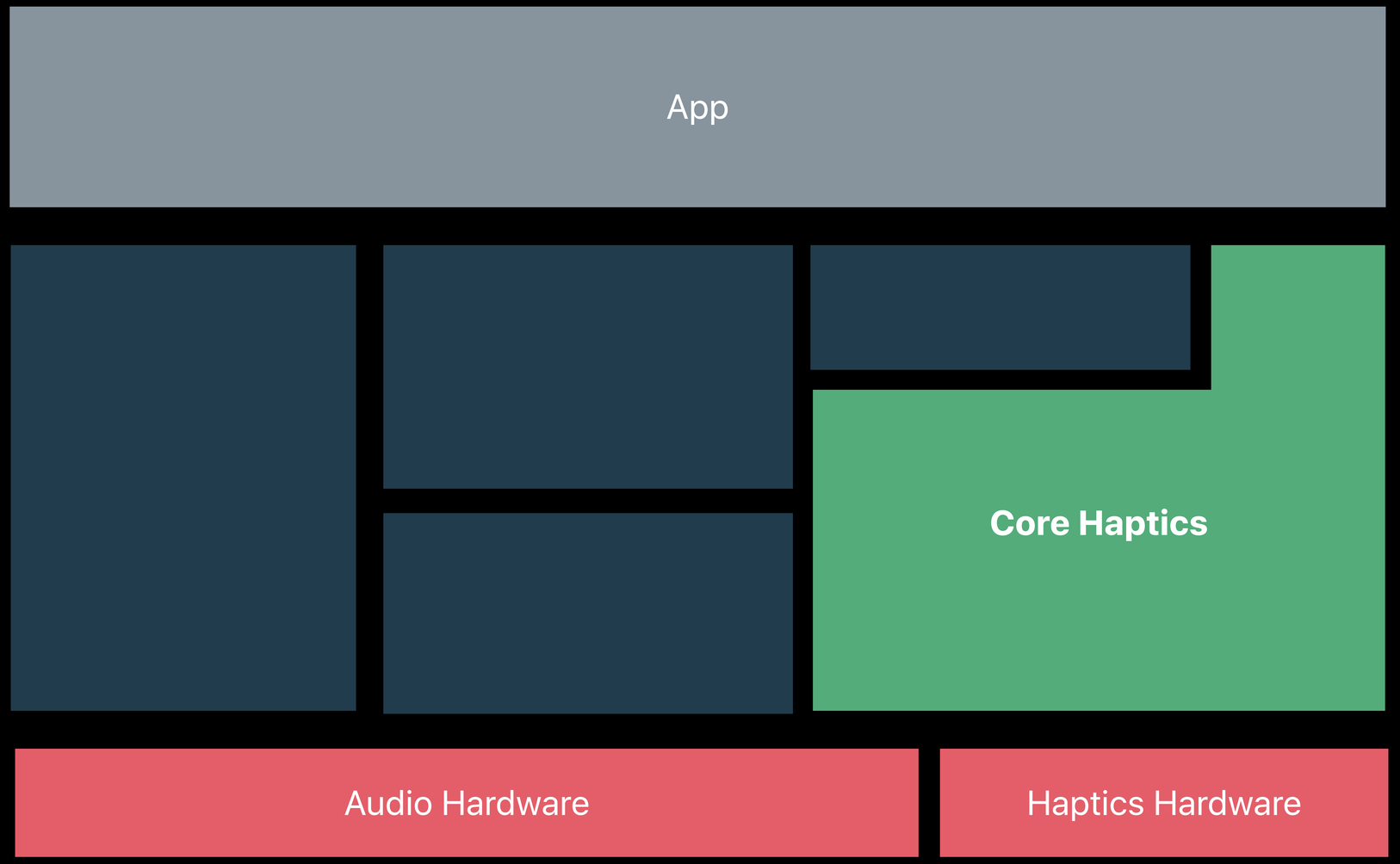
Our first haptics

Dynamic parameters

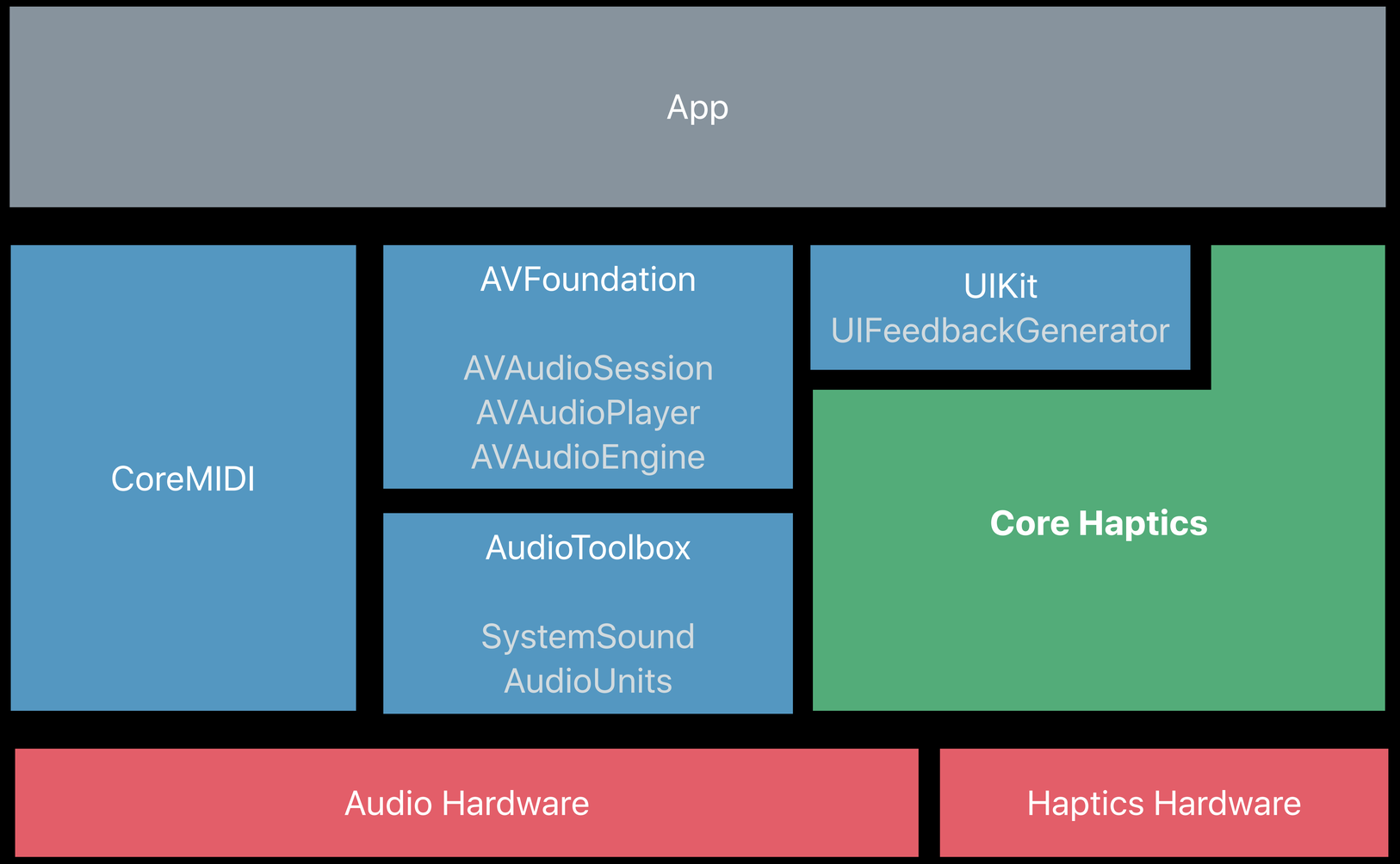
Apple Haptic Audio Pattern (AHAP)

Where to Use Core Haptics

What is Core Haptics?



What is Core Haptics?



Device Support

Same feel across products

iPhone 8

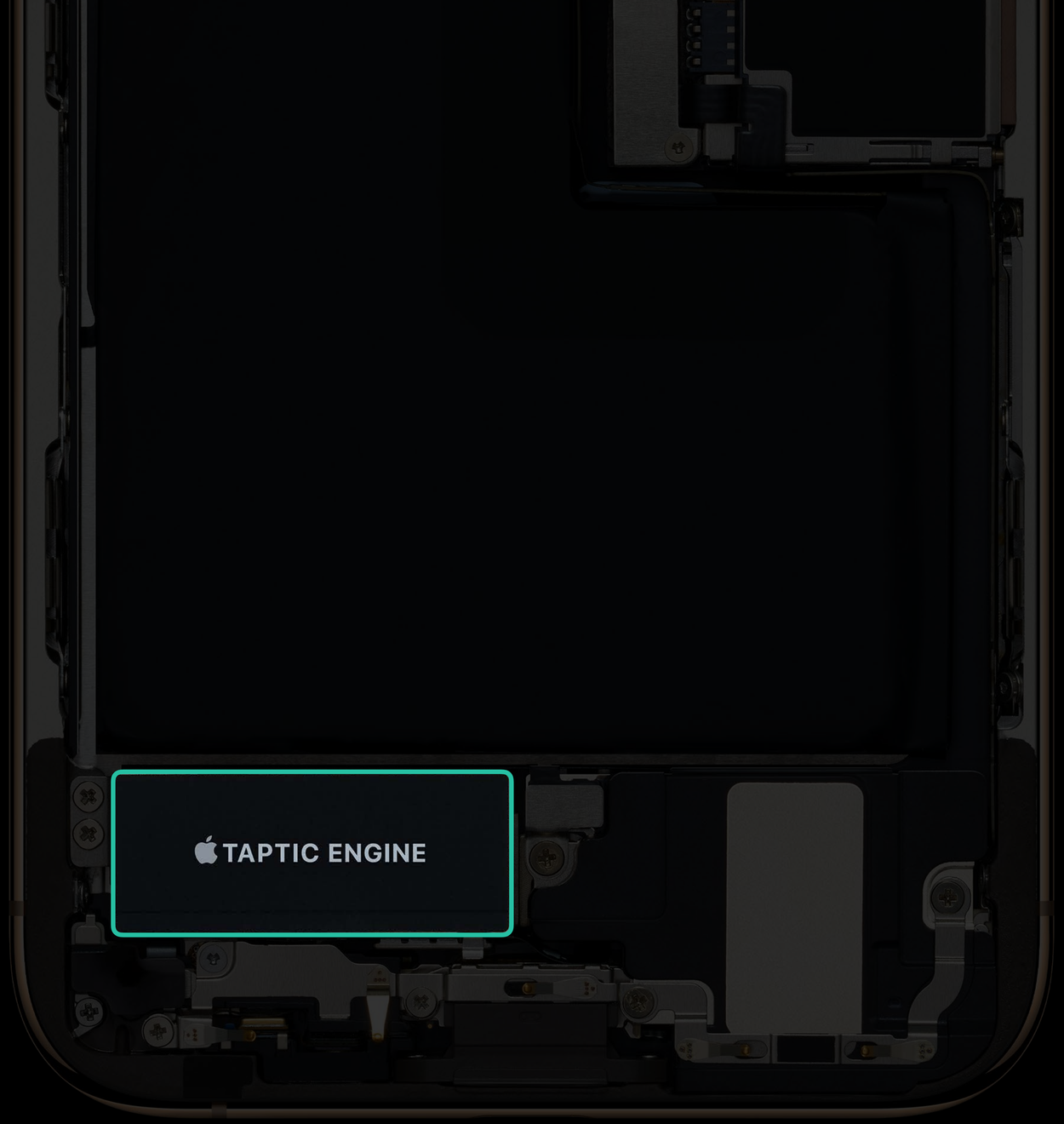
iPhone 8 Plus

iPhone X

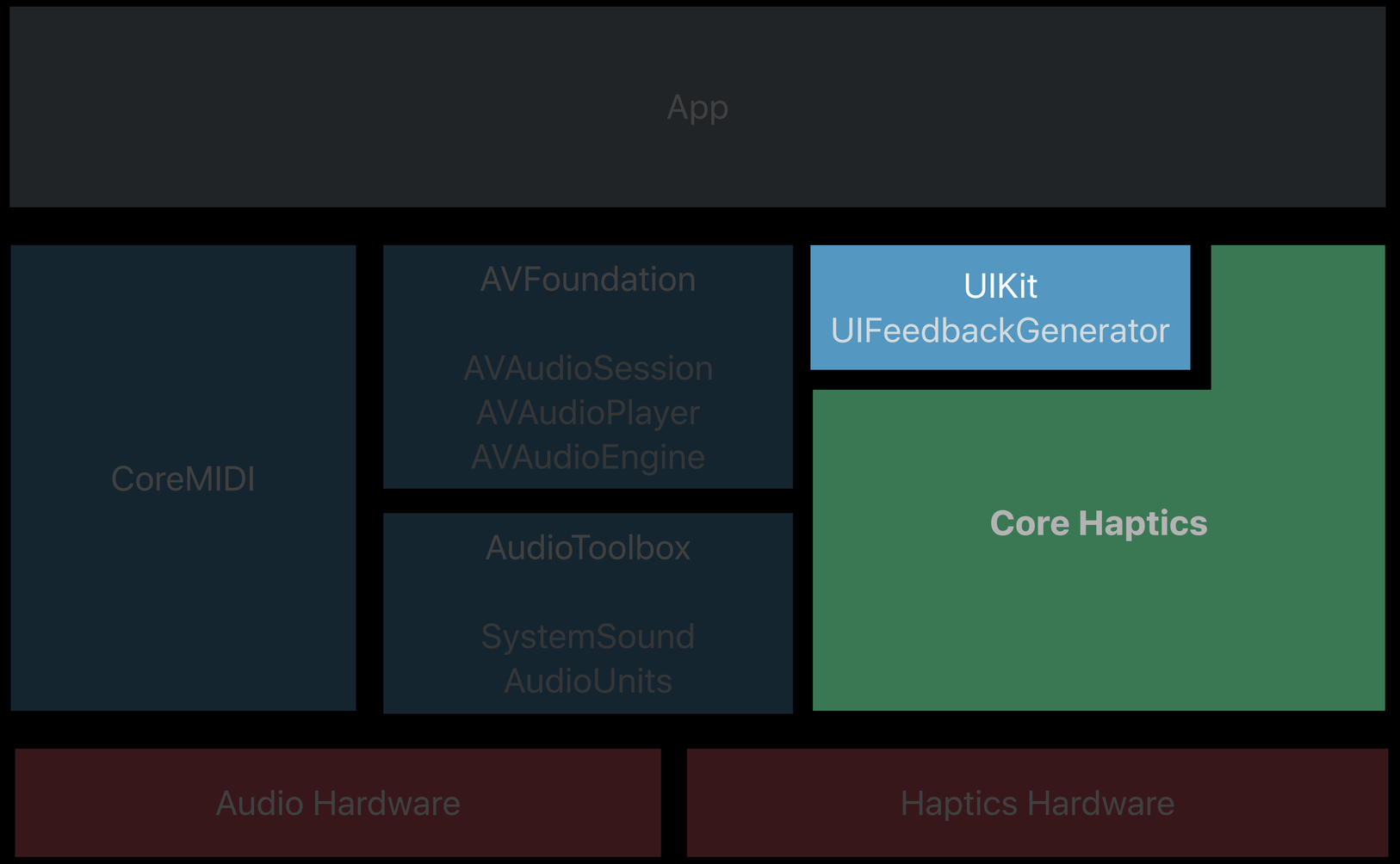
iPhone XS

iPhone XR

iPhone XS Max


TAPTIC ENGINE

Core Haptics in Context



Choose the Right Haptics API

UIFeedbackGenerator

Impact, Selection, Notification

Apple designed vocabulary

Common across apps

Improved in iOS 13

Choose the Right Haptics API

UIFeedbackGenerator

Impact, Selection, Notification

- Apple designed vocabulary
- Common across apps
- Improved in iOS 13

Core Haptics

- Custom haptic and audio patterns
- Accepts timestamps for future playback
- Rich playback and modulation controls

Haptics and Audio Duality

Haptics and Audio Duality

(Haptic) Home Button

Haptics and Audio Duality

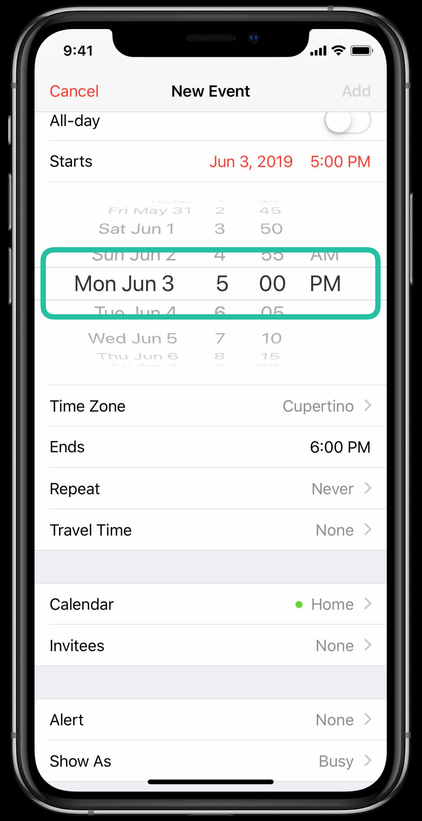
(Haptic) Home Button

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

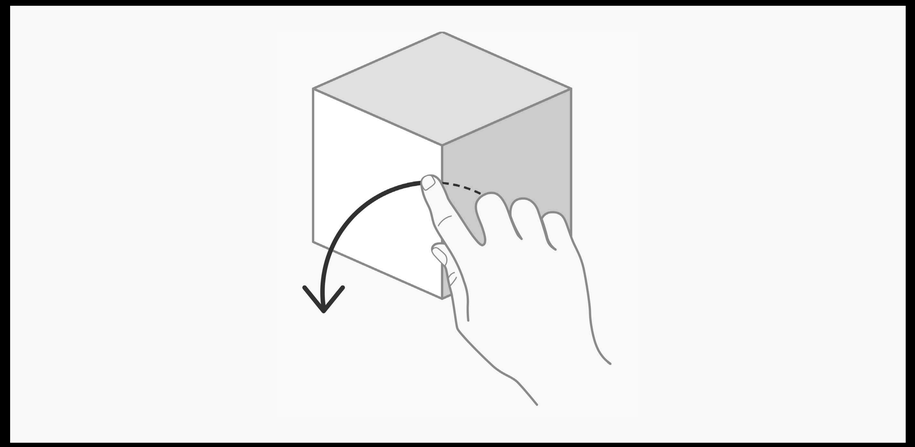
Core Haptics and Augmented Reality

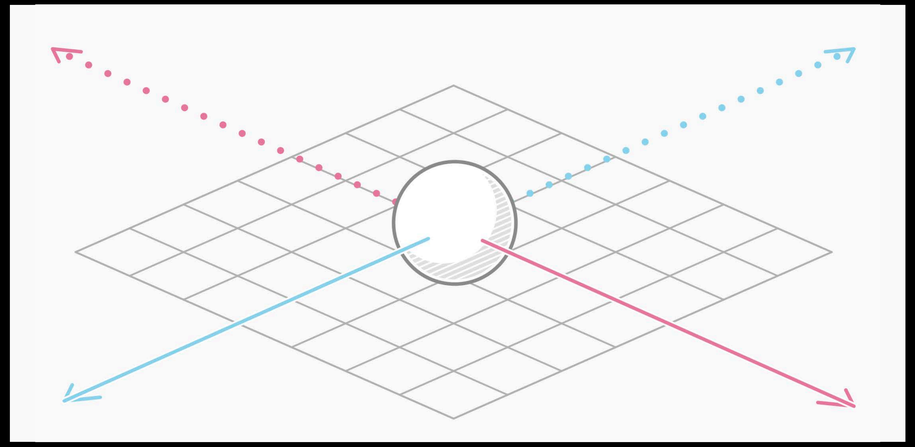
Increase immersion

Core Haptics and Augmented Reality

Increase immersion

Ground user gestures



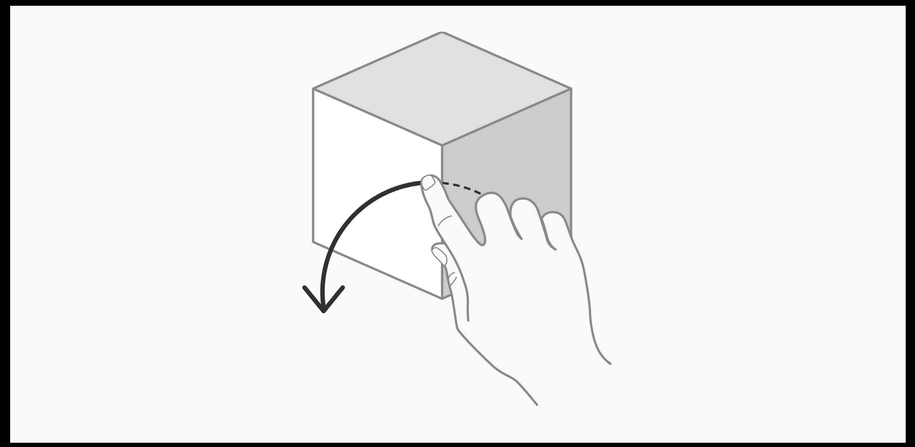


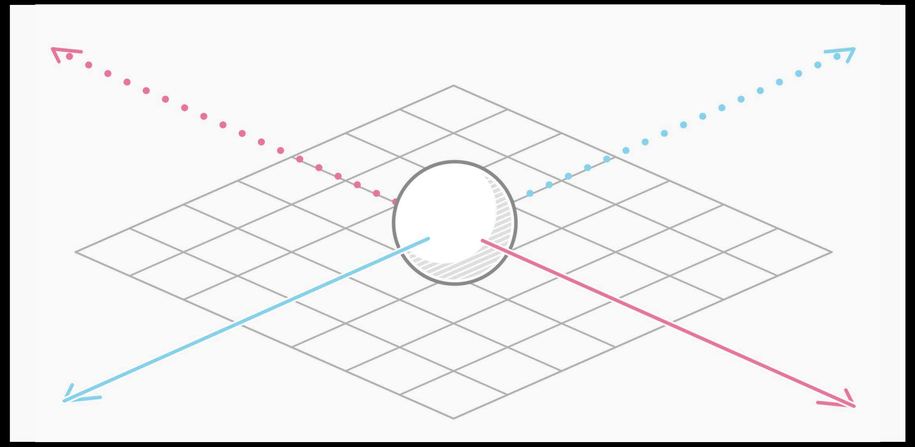
Core Haptics and Augmented Reality

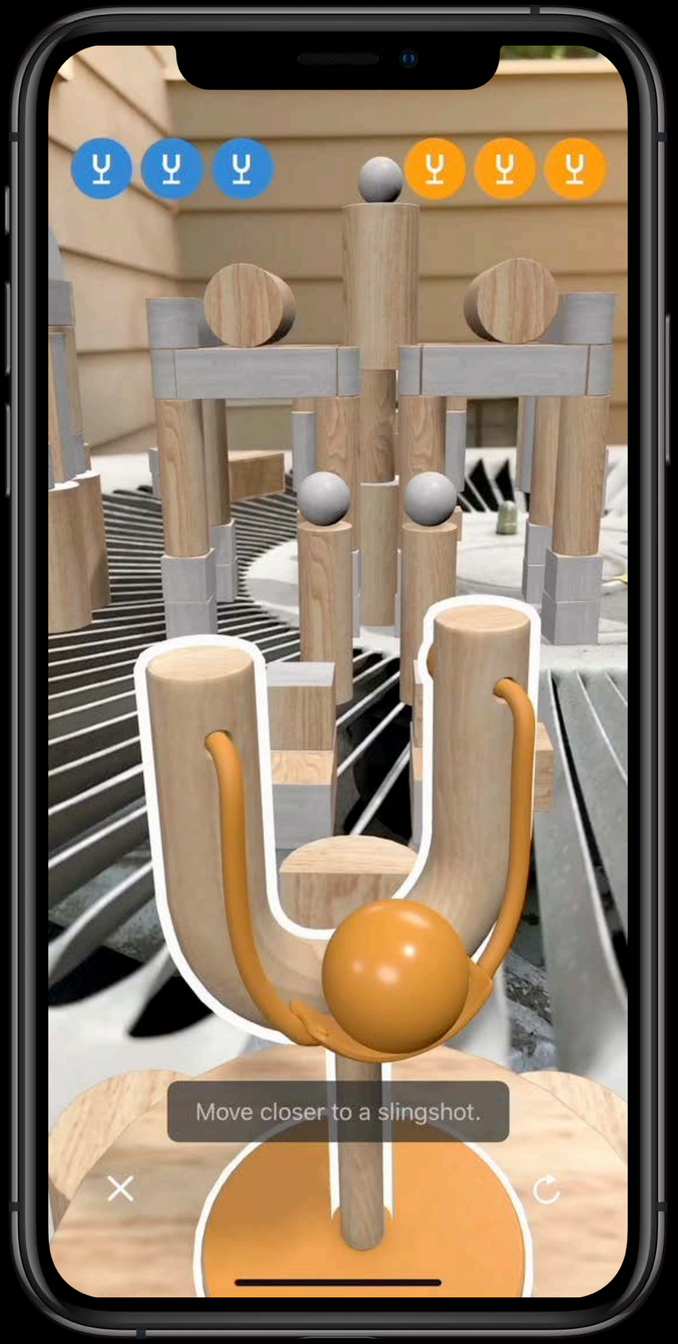
Increase immersion

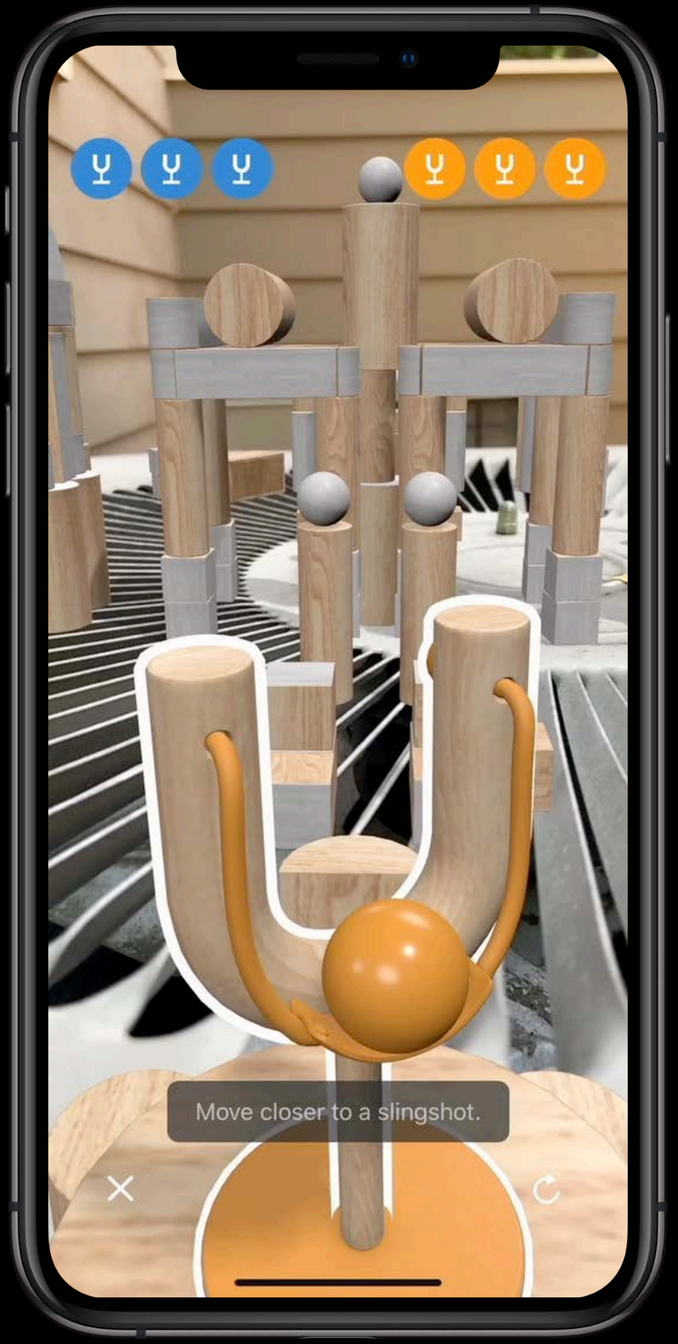
Ground user gestures

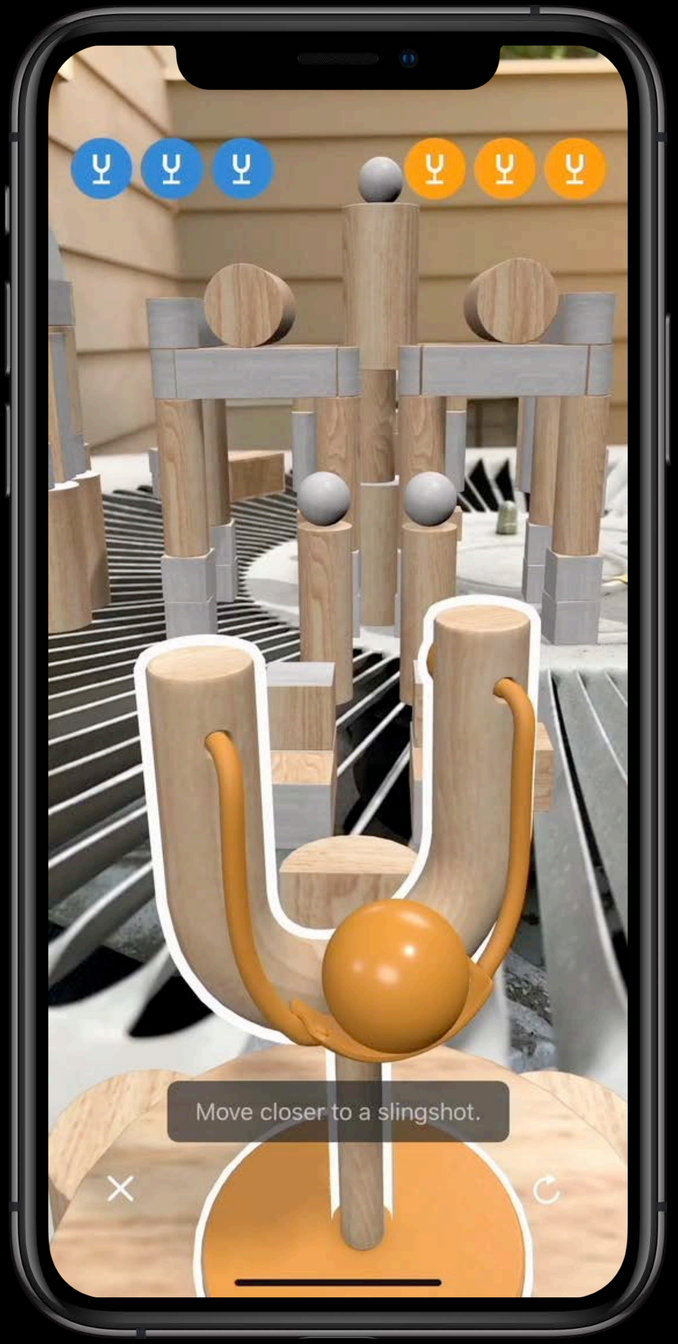
Feedback on device or AR world events





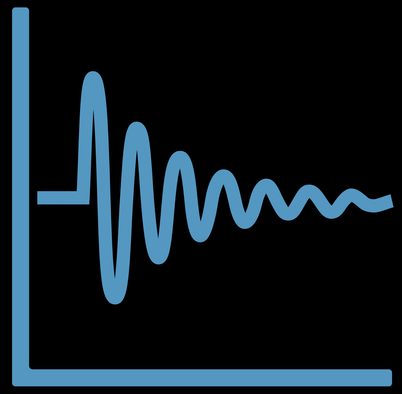

Move closer to a slingshot.


Move closer to a slingshot.


Move closer to a slingshot.

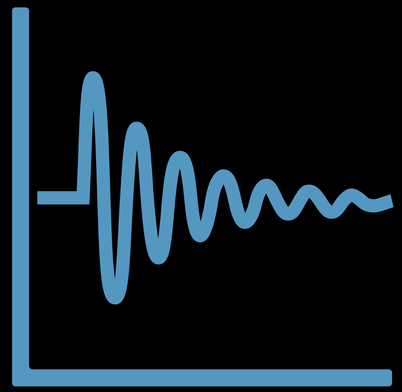
Expressing Content

Classes for Content



`CHHapticPattern`
`CHHapticEvent`
`CHHapticParameter`

Classes for Content



`CHHapticPattern`
`CHHapticEvent`
`CHHapticParameter`

Classes for Playback

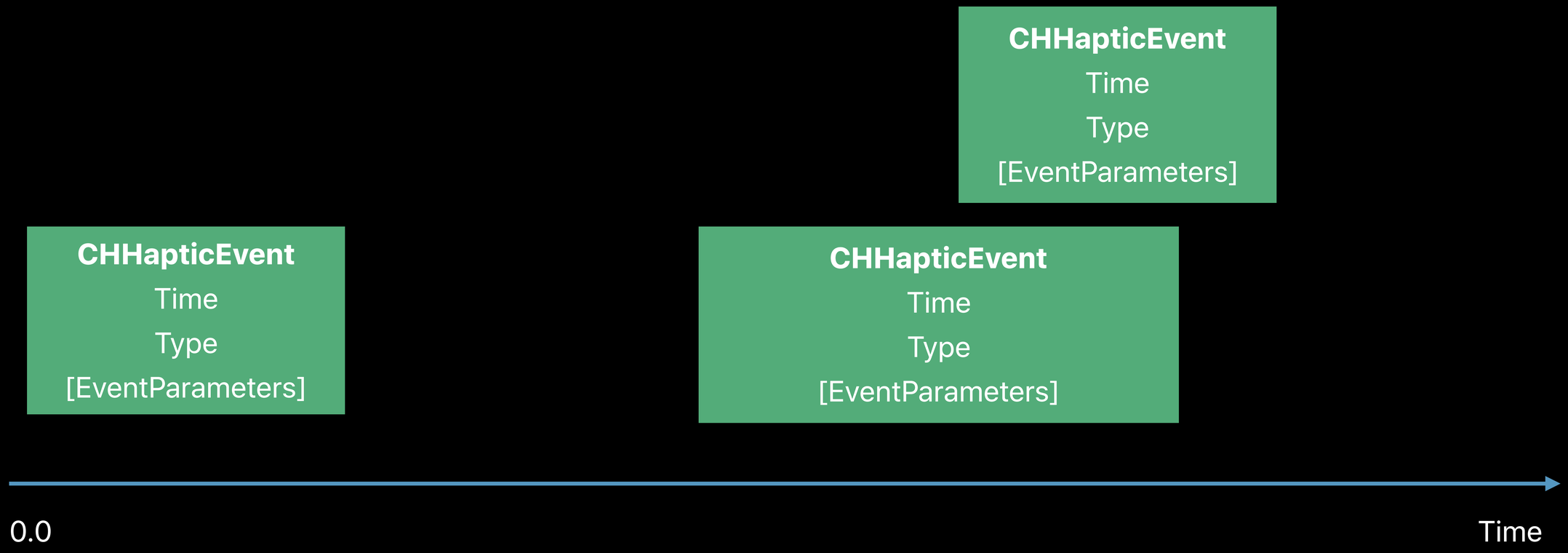
`CHHapticEngine`

owns and vends

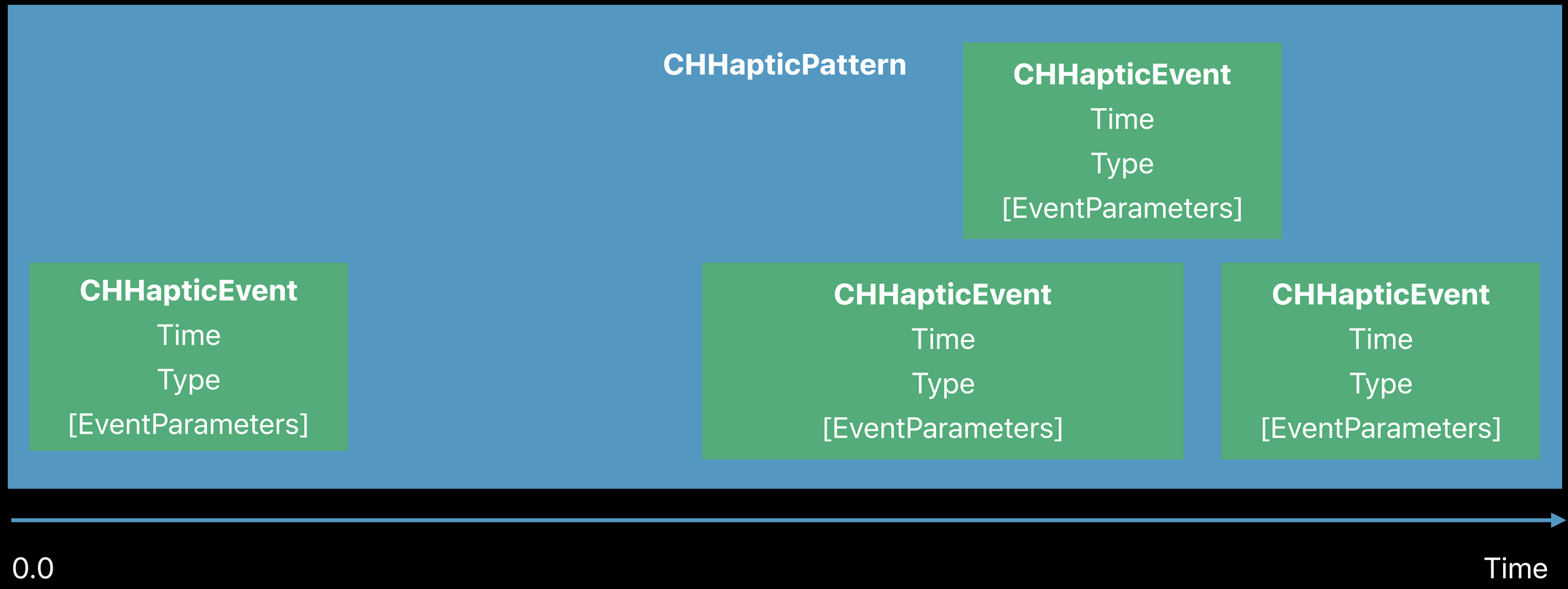
`CHHapticPatternPlayer(s)`
`CHHapticAdvancedPatternPlayer(s)`

Events, Parameters, and Patterns

Events, Parameters, and Patterns



Events, Parameters, and Patterns



Types of Events

HapticTransient

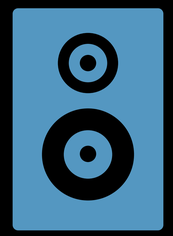
Think “striking”; momentary; instantaneous

Types of Events

HapticTransient	HapticContinuous	AudioContinuous
Think “striking”; momentary; instantaneous	Think “bowing” Can be background texture Richer set of knobs	

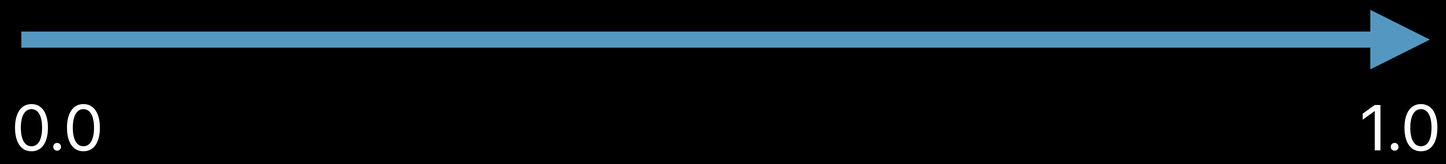
Types of Events

HapticTransient	HapticContinuous	AudioContinuous	AudioCustom
Think “striking”; momentary; instantaneous	Think “bowing” Can be background texture Richer set of knobs		Developer-provided waveform



Our First EventParameter

HapticIntensity / AudioVolume

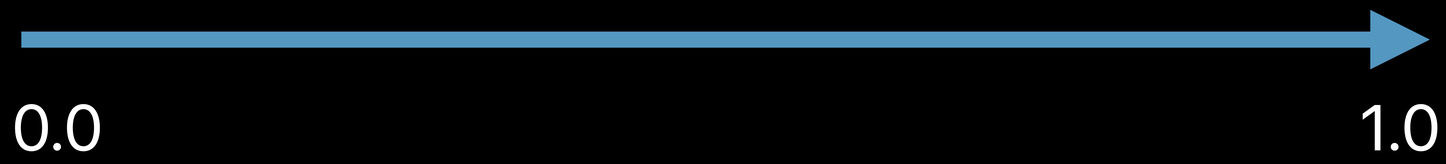


Our First EventParameter

HapticIntensity / AudioVolume

HapticSharpness

An abstraction layer for physical dimensions



HapticSharpness

An abstraction layer for physical dimensions

HapticSharpness

An abstraction layer for physical dimensions

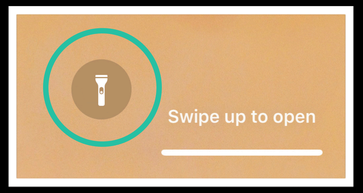
HapticSharpness

An abstraction layer for physical dimensions

HapticSharpness

An abstraction layer for physical dimensions

Flashlight button

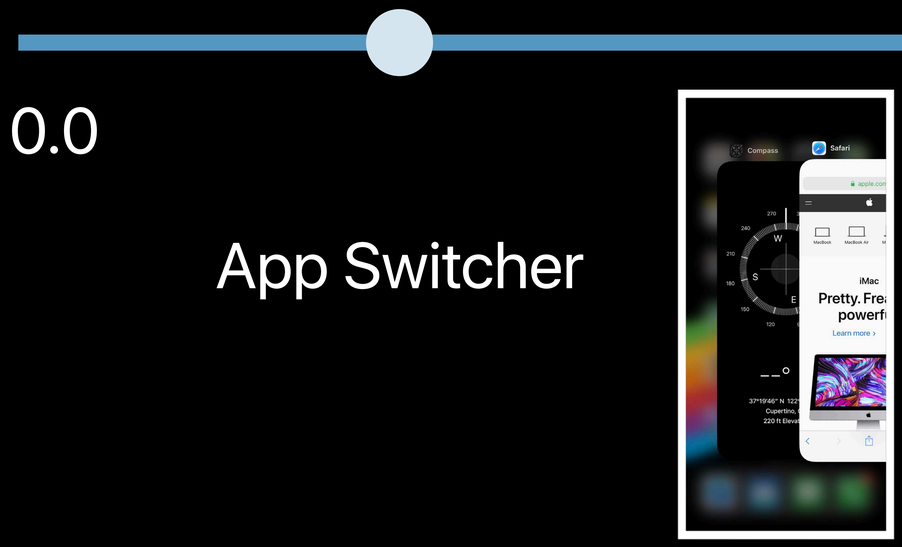


HapticSharpness

An abstraction layer for physical dimensions

Round, organic

Crisp, precise

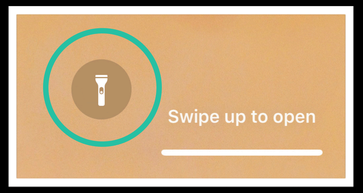


0.0

1.0

App Switcher

Flashlight button



Palette

Sample code

Our First Haptics

Douglas Scott, Interactive Haptics

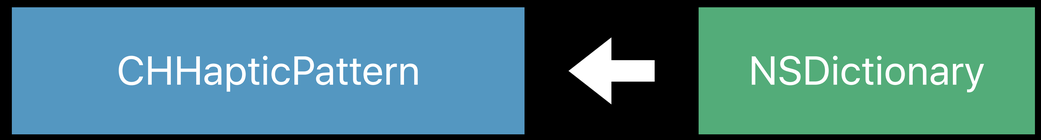
Our First Haptics

Douglas Scott, Interactive Haptics

Playing a Haptic Pattern: Recommended Flow

Playing a Haptic Pattern: Recommended Flow

1. Create haptic content



Playing a Haptic Pattern: Recommended Flow

1. Create haptic content

2. Create haptic engine

CHHapticPattern

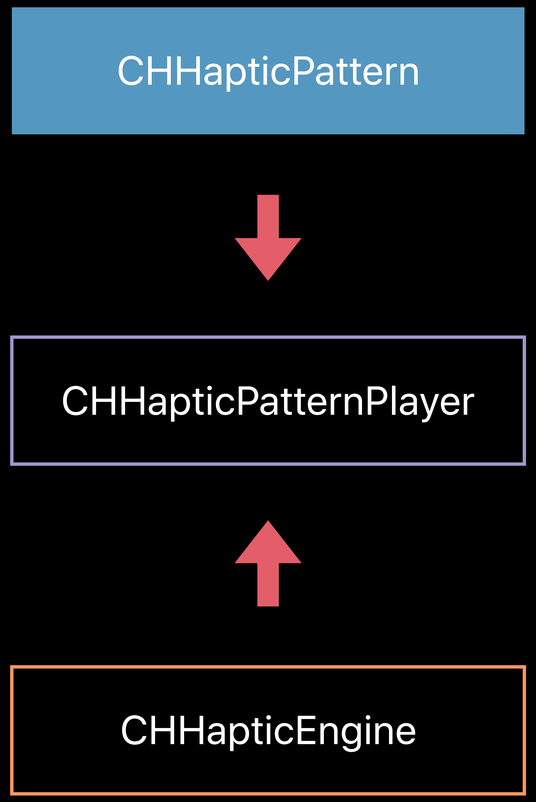
CHHapticEngine

Playing a Haptic Pattern: Recommended Flow

1. Create haptic content

2. Create haptic engine

3. Create haptic pattern player



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

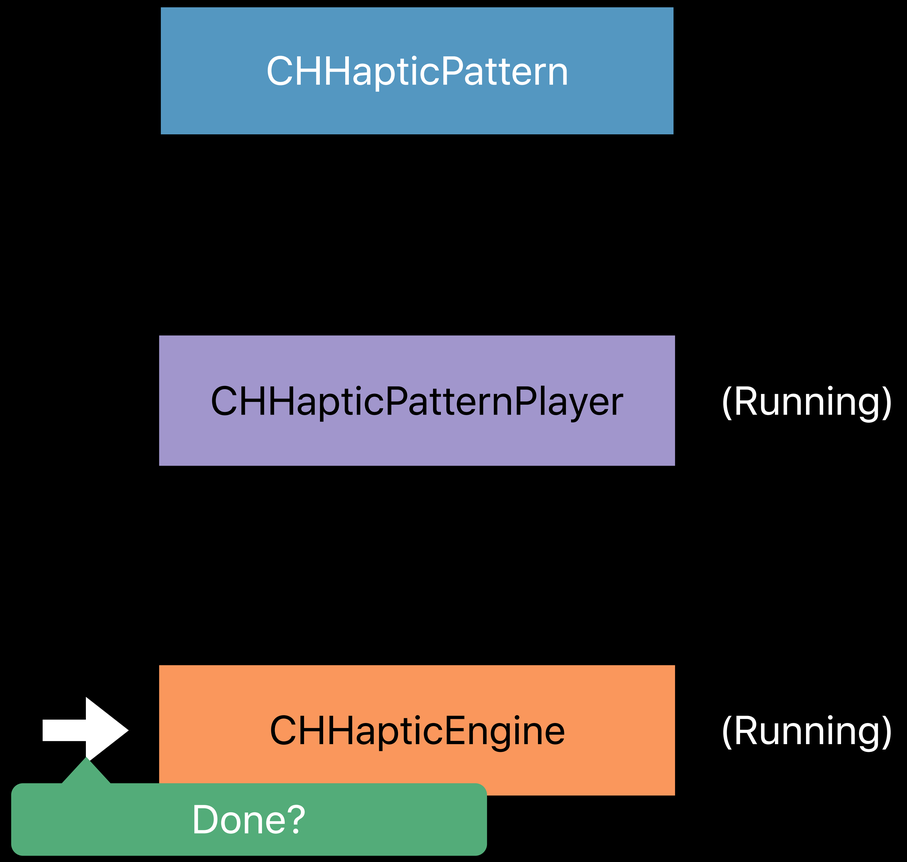
CHHapticPattern

CHHapticPatternPlayer (Running)

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
6. Wait for the player to finish (optional)…



Playing a Haptic Pattern: Recommended Flow

1. Create haptic content

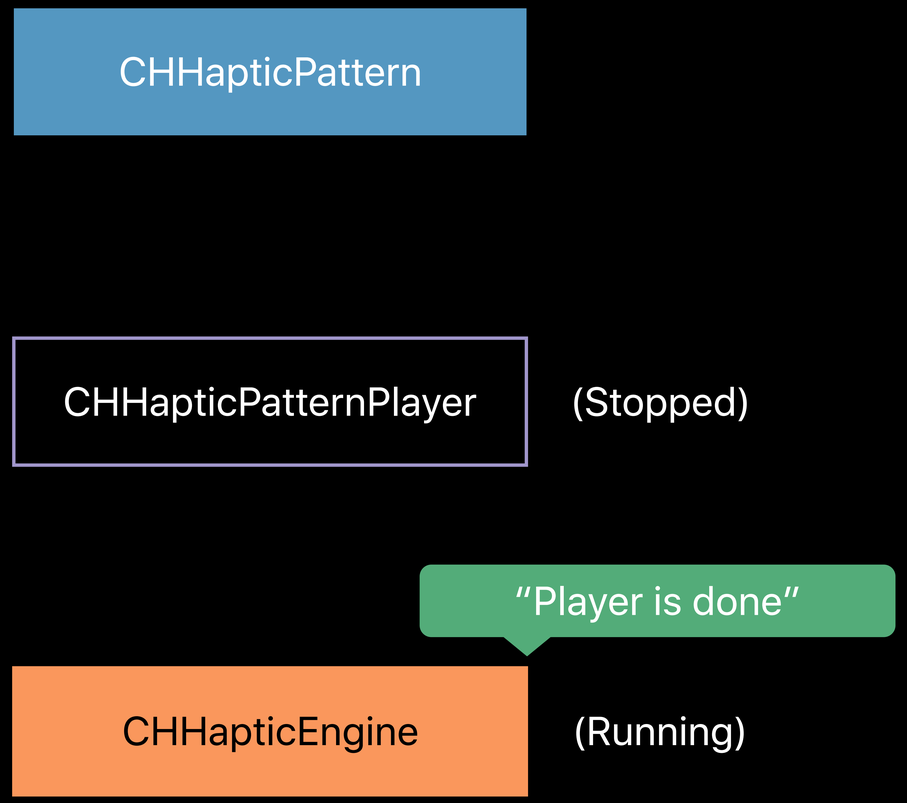
2. Create haptic engine

3. Create haptic pattern player

4. Start the engine

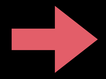
5. Start the player

6. Wait for the player to finish (optional)…



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)

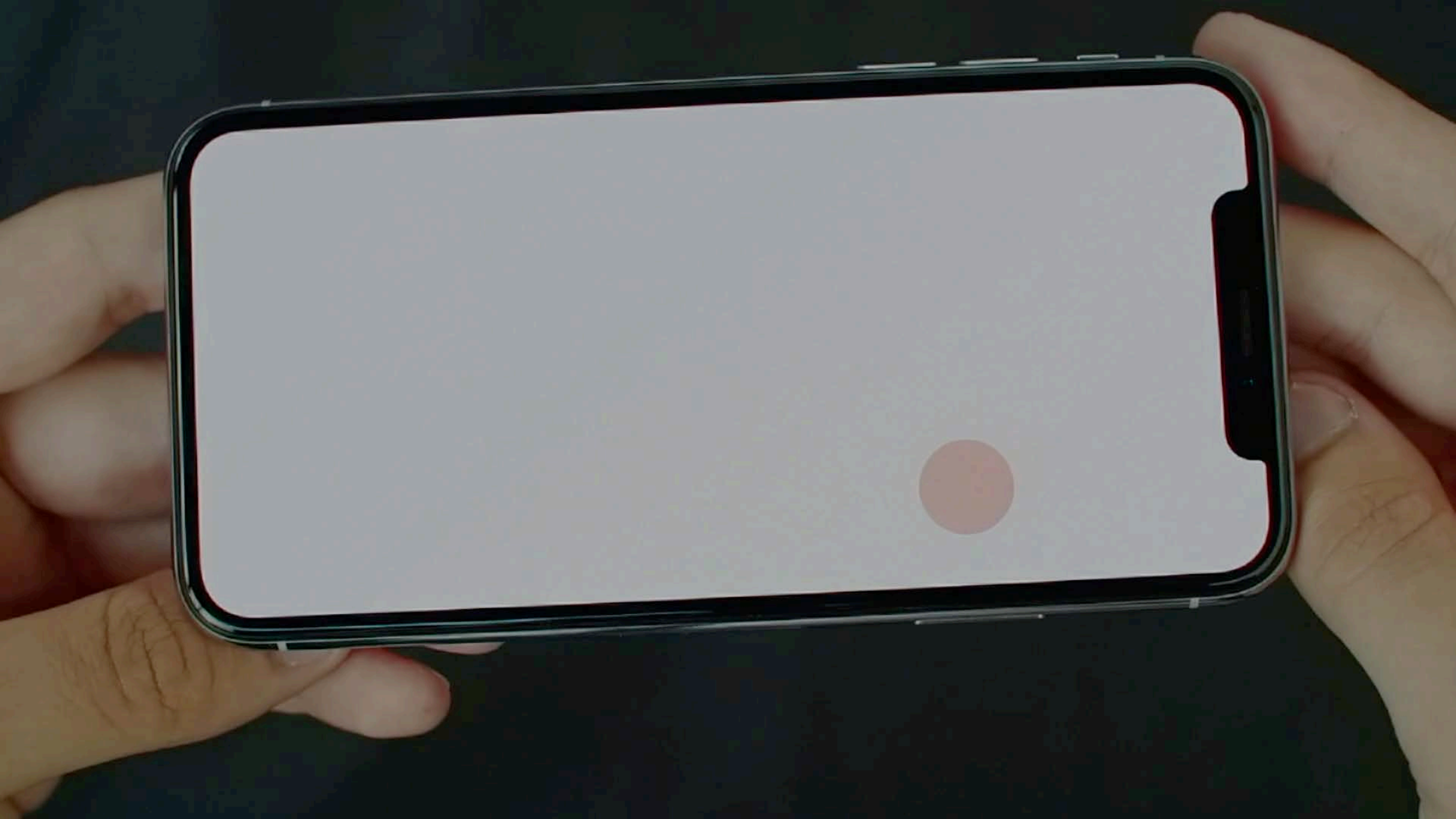


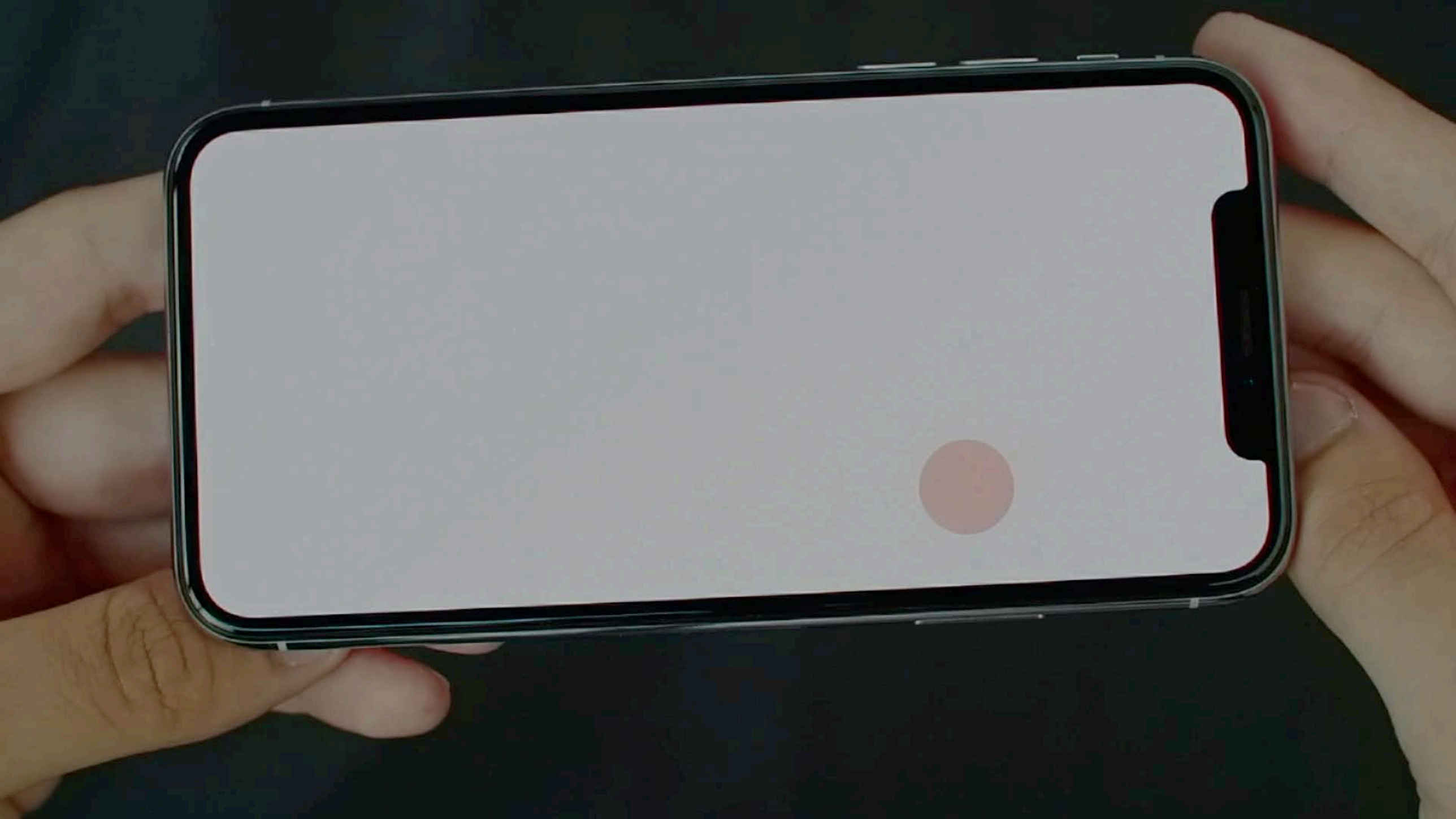
CHHapticPattern

CHHapticPatternPlayer (Stopped)

CHHapticEngine (Stopped)

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
}
```

```swift
// 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: [])
```

```
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: [])
    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

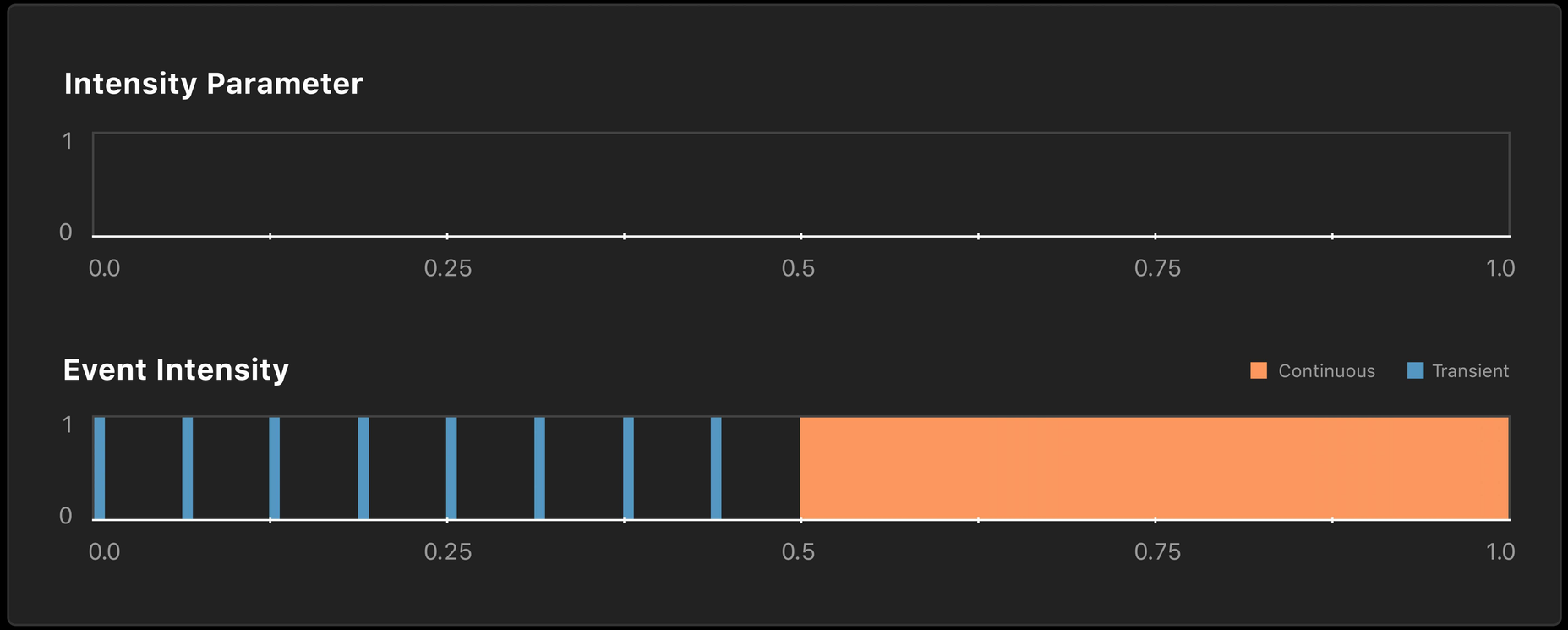
Affects all Events

Modifies (modulates) EventParameter values

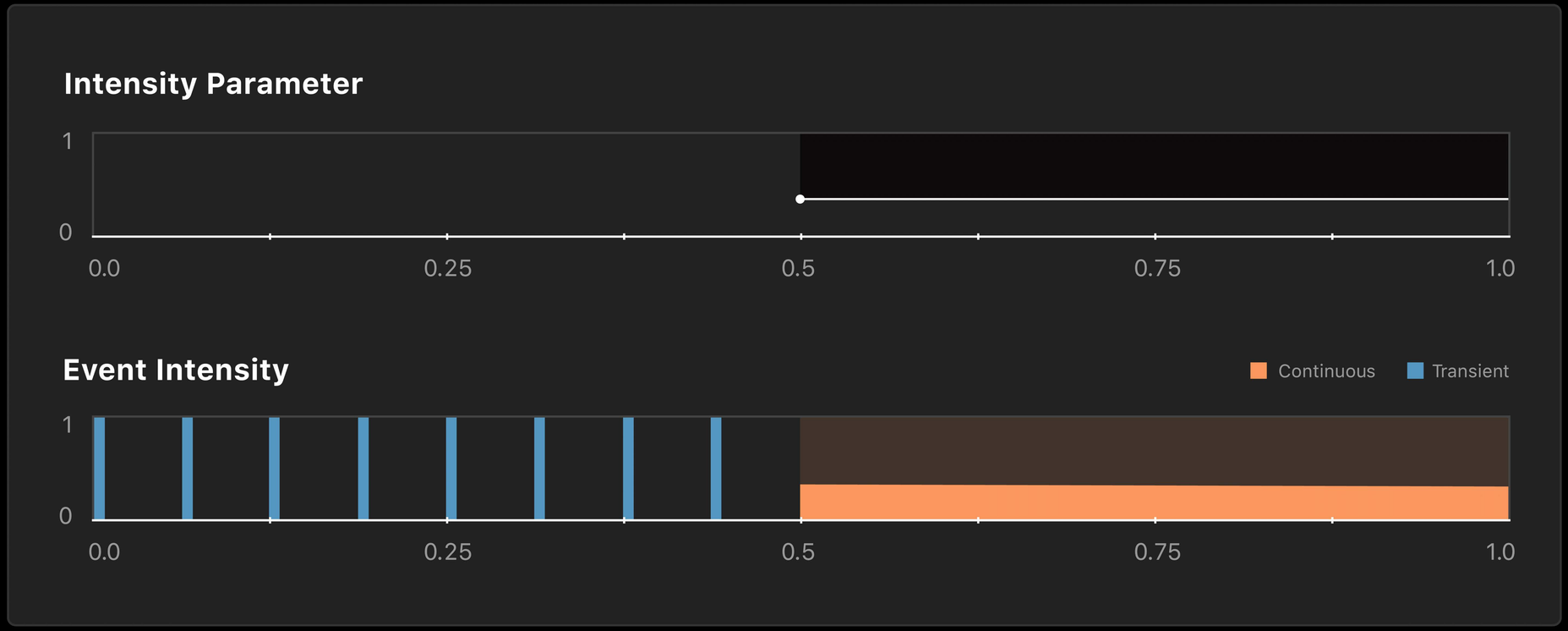
Embed in Pattern or send in real time

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 }
                ]
            }
       }
    ]
}
```

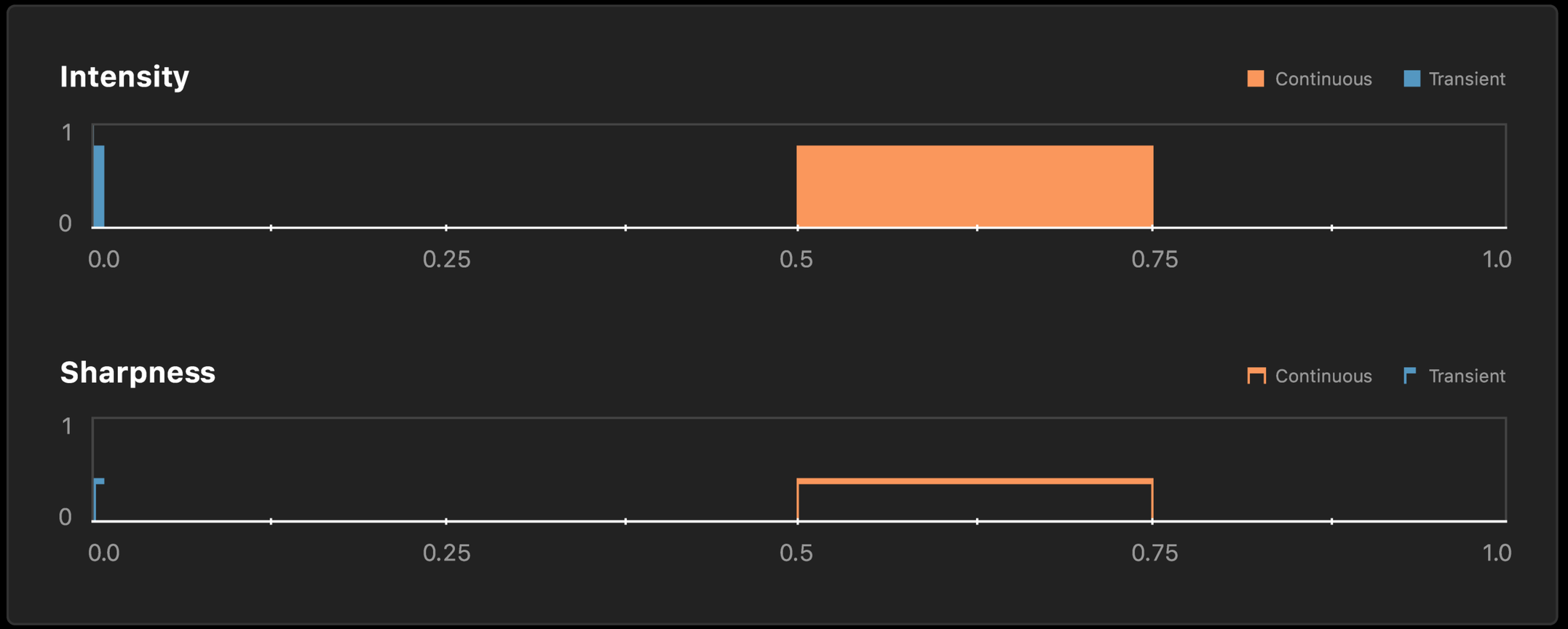
```
// A second event
   “Pattern": [
       {
            “Event": {   ...    }
       },
       {
            “Event": {
                "Time": 0.5,
                "EventType": “HapticContinuous”,
                "EventDuration": 0.25,
                “EventParameters": [
                    { "ParameterID":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

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

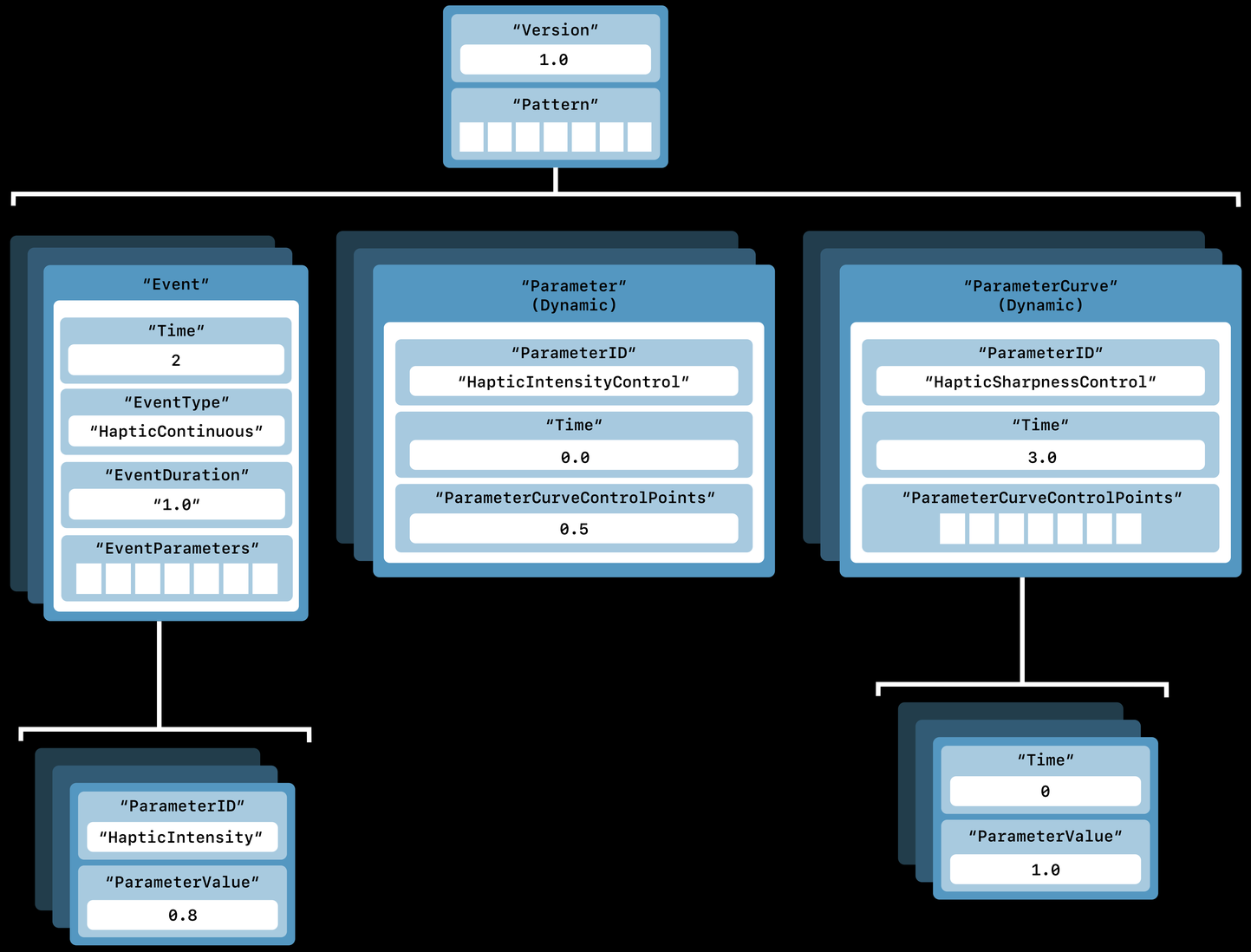
```
// EventDuration key is required for AudioContinuous, HapticContinuous
   “Pattern": [
       {
            “Event": {   ...    }
       },
       {
            “Event": {
                "Time": 0.5,
                "EventType": “HapticContinuous”,
                "EventDuration": 0.25,
       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

```
// 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

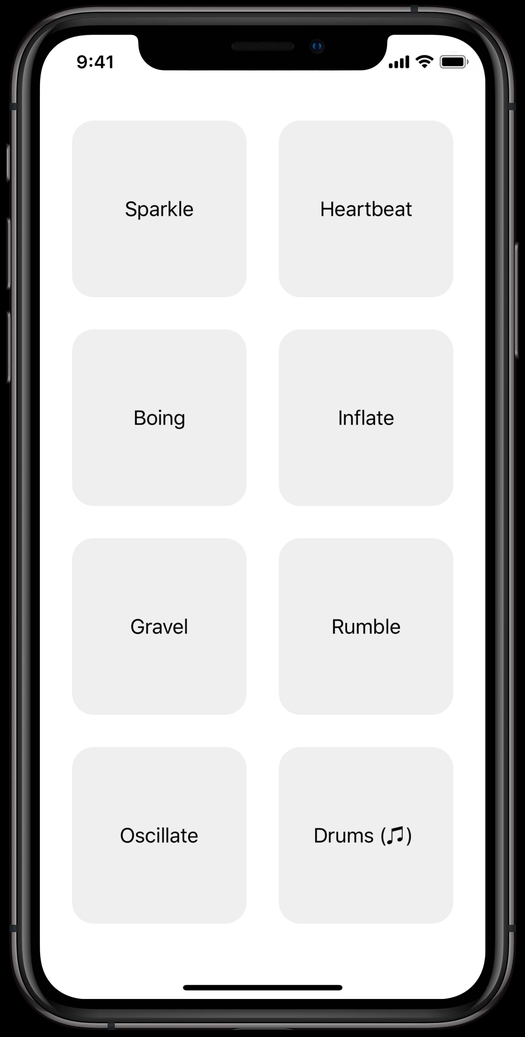


AHAP Structure Summary



HapticSampler

Sample code



More to Discover

Michael Diu, Interactive Haptics

Core Haptics : More to Discover

Audio Events and Parameters	Custom audio + synthesized audio Pitch; Pan; Brightness Attack, Decay, Release Haptics-only mode	**Modulation**	Use Parameter Curves to achieve the same effect as a sequence of DynamicParameters Statically defined as part of the Pattern, or created during playback
AudioSession Integration	Routing and interruption behavior follows AudioSession Category New API to allow haptics during recording	**Error Handling**	Error Handling for Pattern Parsing Error Handling and Recovery during Playback Query HW Capabilities
Playback controls	Muting (Player; or entire Engine) Looping Playback rate adjust Pause/resume Seek; play or stop at timestamp	**Player and Engine Notifications**	Playback complete callbacks Engine stopped notifications autoShutdown mode

What makes a good haptic pattern?

Do the rules for sound design carry over?

Take advantage of the updated HIG for haptics.

Summary

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/wwdc19/223

Core Haptics Lab	Thursday, 11:00
Core Haptics Lab (2)	Friday, 9:00


WWDC19