Sample Code

Updating Continuous and Transient Haptic Parameters in Real Time

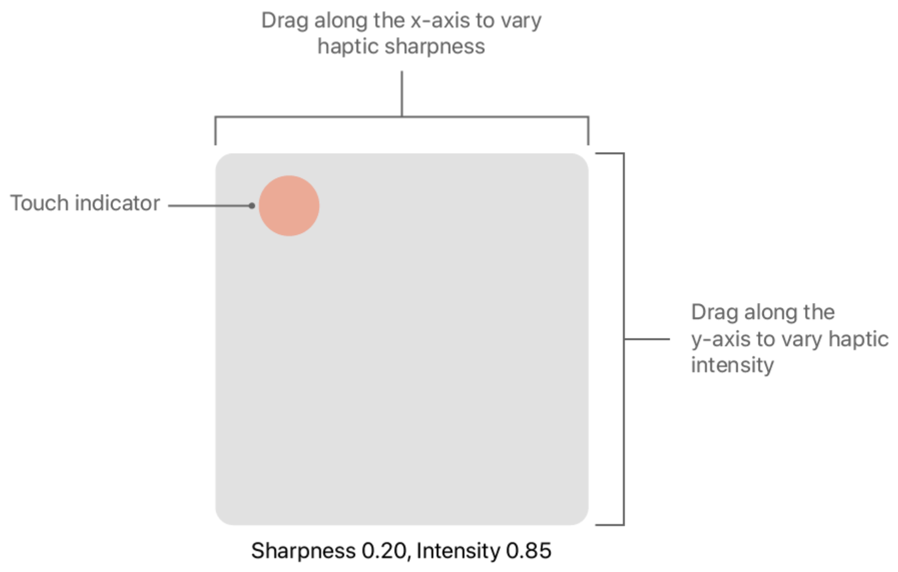
Generate continuous and transient haptic patterns in response to user touch.

Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 12.0+

Overview

This sample, HapticPalette, defines a two-dimensional touch area, where the x-axis varies the haptic pattern's sharpness, and the y-axis maps to haptic intensity. Use this interface to explore the range of haptic patterns that you can create with Core Haptics. By sliding your finger inside the bounded area, you can vary the parameters of the generated haptic pattern.


Drag along the x-axis to vary haptic sharpness
Touch indicator
Drag along the y-axis to vary haptic intensity
Sharpness 0.20, Intensity 0.85

Note

HapticPalette requires Xcode 11 or later. Simulator doesn't support a haptic interface, so build and run this sample on an iPhone 8, 8 Plus, X, XR, XS, or XS Max running iOS 13 or later.

Configure the App to Play Haptics

This sample checks for device compatibility and sets up an instance of `CHHapticEngine` on launch. Before proceeding, start the engine, so it can begin accepting haptics requests.

```
// Start the haptic engine for the first time.
do {
    try self.engine.start()
} catch {
    print("Failed to start the engine: \(error)")
}
```

See Preparing your app to play haptics for more information about setting up the engine.

Create a Continuous Haptic Pattern

As the app demonstrates, you can create a sustained vibration by generating a continuous haptic pattern of nonzero duration. When you first create the pattern, on app launch, set the intensity and sharpness through an event parameter.

```
// Create an intensity parameter:
let intensity = CHHapticEventParameter(parameterID: .hapticIntensity,
                                       value: initialIntensity)

// Create a sharpness parameter:
let sharpness = CHHapticEventParameter(parameterID: .hapticSharpness,
                                       value: initialSharpness)

// Create a continuous event with a long duration from the parameters.
let continuousEvent = CHHapticEvent(eventType: .hapticContinuous,
                                    parameters: [intensity, sharpness],
                                    relativeTime: 0,
                                    duration: 100)

do {
    // Create a pattern from the continuous haptic event.
    let pattern = try CHHapticPattern(events: [continuousEvent], parameters: [])

    // Create a player from the continuous haptic pattern.
    continuousPlayer = try engine.makeAdvancedPlayer(with: pattern)

} catch let error {
    print("Pattern Player Creation Error: \(error)")
}

continuousPlayer.completionHandler = { _ in
    DispatchQueue.main.async {
        // Restore original color.
        self.continuousPalette.backgroundColor = self.padColor
    }
}
```

The sample app allows you to change the continuous pattern's intensity and sharpness by dragging your finger inside the region labeled Continuous. This region is a view that translates the touch coordinates into normalized intensity and sharpness values. To enable this behavior, the code attaches a `UILongPressGestureRecognizer` to each palette view. When the user begins to drag a finger in the view, the gesture handler maps the touch coordinates to intensity and sharpness values.

```
// Normalize coordinates to a [0, 1] spectrum.
let normalizedLocation = normalizeCoordinates(clippedLocation, toView: view)

// The intensity should be highest at the top, opposite of the iOS y-axis direction,
let eventIntensity: Float = 1 - Float(normalizedLocation.y)

// The sharpness spectrum is 0 to 1. Map the x-coordinate to that range.
let eventSharpness: Float = Float(normalizedLocation.x)
```

The normalized mapping converts the user's finger location to the available spectrum. Using these normalized values, you can create intensity and sharpness events that determine how the haptic pattern feels to the user. In this example, touching the palette area in the upper-right corner generates a crisp haptic vibration at full strength.

The app retains the original pattern player instead of creating a new one with each change. To tell the continuous haptic pattern player to change or modulate its haptic as it's playing, send dynamic parameters to the player.

Alter the Continuous Haptic Pattern by Sending Dynamic Parameters

To vary a continuous haptic pattern, send dynamic parameters of class `CHHapticDynamicParameter` to the original pattern. This modifies the haptic pattern as it's playing. In this example, the dynamic parameters for intensity and sharpness immediately change the haptic pattern's strength and character to follow the user's finger.

```
// Create dynamic parameters for the updated intensity & sharpness.
let intensityParameter = CHHapticDynamicParameter(parameterID: .hapticIntensityContr
                                                  value: dynamicIntensity,
                                                  relativeTime: 0)

let sharpnessParameter = CHHapticDynamicParameter(parameterID: .hapticSharpnessContr
                                                  value: dynamicSharpness,
                                                  relativeTime: 0)

// Send dynamic parameters to the haptic player.
do {
    try continuousPlayer.sendParameters([intensityParameter, sharpnessParameter],
                                        atTime: 0)
} catch let error {
    print("Dynamic Parameter Error: \(error)")
}
```

Note

Sending a dynamic parameter for intensity *multiplies* the original pattern's event intensity by the dynamic parameter value. Sending a dynamic parameter for sharpness *adds* the dynamic parameter value to the original pattern's event sharpness.

Only when the gesture ends does the app stop the pattern player.

```
// Stop playing the haptic pattern.
do {
    try continuousPlayer.stop(atTime: CHHapticTimeImmediate)
} catch let error {
    print("Error stopping the continuous haptic player: \(error)")
}
```

When the app sends the continuous pattern player a dynamic parameter, the value of that parameter changes immediately, resulting in a viscerally different haptic pattern on the device. If your app needs to change the parameter value gradually, consider scheduling a `CHHapticParameterCurve`.

Create a Transient Haptic Pattern

The bottom half of the screen generates transient haptic experiences. A transient haptic event is momentary. This sample generates a transient haptic pattern when the gesture begins. It's also helpful to generate subsequent transient haptic patterns if the user continues to interact with the interface. As long as the user's finger remains inside the transient zone, the app continues to create a new transient haptic pattern based on the latest sharpness and intensity values. To accomplish this spacing, you use a dispatch timer to separate the transient haptic patterns by a time interval of 600 milliseconds, so they feel distinct.

```
timer.schedule(deadline: .now() + .milliseconds(750), repeating: .milliseconds(600))
timer.setEventHandler() { [unowned self] in

    // Recalibrate sharpness and intensity each time the timer fires.
    let newLocation = press.location(in: self.transientPalette)
    let (sharpness, intensity) = self.sharpnessAndIntensityAt(location: newLocation,

    self.playHapticTransient(time: CHHapticTimeImmediate,
                             intensity: intensity,
                             sharpness: sharpness)
}
```

Ask the engine to create a player from the transient pattern when the timer fires, and then play it.

```
// Create an event (static) parameter to represent the haptic's intensity.
let intensityParameter = CHHapticEventParameter(parameterID: .hapticIntensity,
                                                value: intensity)

// Create an event (static) parameter to represent the haptic's sharpness.
let sharpnessParameter = CHHapticEventParameter(parameterID: .hapticSharpness,
                                                value: sharpness)

// Create an event to represent the transient haptic pattern.
let event = CHHapticEvent(eventType: .hapticTransient,
                          parameters: [intensityParameter, sharpnessParameter],
                          relativeTime: 0)
```

```swift
// Create a pattern from the haptic event.
do {
    let pattern = try CHHapticPattern(events: [event], parameters: [])

    // Create a player to play the haptic pattern.
    let player = try engine.makePlayer(with: pattern)
    try player.start(atTime: CHHapticTimeImmediate) // Play now.
} catch let error {
    print("Error creating a haptic transient pattern: \(error)")
}
```

Creating haptic pattern players is inexpensive, so you don’t need to keep them around. Even if the user slides rapidly across the touch zone, the engine creates players, plays them, and discards them without lag.

> **Note**
>
> While haptic pattern players are lightweight and inexpensive to create and start, starting the haptic engine is expensive, as is stopping it, so start and stop the engine sparingly. This sample starts the engine only once, on app launch.

See Also

Programmatic haptics

{} Delivering Rich App Experiences with Haptics

Enhance your app’s experience by incorporating haptic and sound feedback into key interactive moments.

{} Playing Collision-Based Haptic Patterns

Play a custom haptic pattern whose strength depends on an object’s collision speed.

`class CHHapticEvent`

An object that describes a single haptic or audio event.

`class CHHapticEventParameter`

A static parameter value that represents a single property of the haptic pattern.

`class CHHapticDynamicParameter`

A value that you send to a haptic pattern player to alter a property value during playback.

`class CHHapticParameterCurve`

A curve that you send to a haptic pattern player to alter a property value gradually during playback.