Sample Code

# Playing Collision-Based Haptic Patterns

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

Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 12.0+

## Overview

The sample code project, HapticBounce, demonstrates how to play a haptic with audio when a bouncing sphere collides with the edges of the screen. The sample app shows how to vary the volume of the synthesized audio signal and the intensity of the transient haptic pattern based on the sphere's velocity during impact.

Although the sample app uses UIKit Dynamics and UIKit for simplicity and familiarity with UIKit classes, but your app can also use a 2D engine like SpriteKit to represent interactive object collisions.

> **Note**
>
> HapticBounce 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

Check for device compatibility and set up an instance of `CHHapticEngine`, as the sample demonstrates when you launch the app.

```
// Create and configure a haptic engine.
do {
    engine = try CHHapticEngine()
} catch let error {
    fatalError("Engine Creation Error: \(error)")
}
```

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

## Define the Objects in UIKit Dynamics

Express the walls as boundaries in a `UICollisionBehavior` with the sphere.

```
wallCollisions = UICollisionBehavior(items: [sphereView])
wallCollisions.collisionDelegate = self

// Express walls using vertices.
let upperLeft = CGPoint(x: -1, y: -1)
let upperRight = CGPoint(x: windowWidth + 1, y: -1)
let lowerRight = CGPoint(x: windowWidth + 1, y: windowHeight + 1)
let lowerLeft = CGPoint(x: -1, y: windowHeight + 1)

// Each wall is a straight line shifted one pixel offscreen, to give an impression o

// Left edge of the screen:
wallCollisions.addBoundary(withIdentifier: NSString("leftWall"),
                           from: upperLeft,
                           to: lowerLeft)

// Right edge of the screen:
wallCollisions.addBoundary(withIdentifier: NSString("rightWall"),
                           from: upperRight,
                           to: lowerRight)

// Top edge of the screen:
wallCollisions.addBoundary(withIdentifier: NSString("topWall"),
                           from: upperLeft,
                           to: upperRight)

// Bottom edge of the screen:
wallCollisions.addBoundary(withIdentifier: NSString("bottomWall"),
```

```
                                from: lowerRight,
                                to: lowerLeft)
```

Each bounce against the wall is a dynamic item behavior, which lets you tweak the elasticity to match the haptic effect. An animator of class `UIDynamicAnimator` ties together all of these dynamic objects with gravity.

```
animator = UIDynamicAnimator(referenceView: view)

// Add bounce, gravity, and collision behavior.
animator.addBehavior(bounce)
animator.addBehavior(gravity)
animator.addBehavior(wallCollisions)
```

HapticBounce uses the device's accelerometer to move the sphere. It adjusts the accelerometer by responding to motion updates in a completion handler.

```
// Manage motion events in a separate queue off the main thread.
motionQueue = OperationQueue()
motionData = CMAccelerometerData()
motionManager = CMMotionManager()

guard let manager = motionManager else { return }

manager.startDeviceMotionUpdates(to: motionQueue) { deviceMotion, error in
    guard let motion = deviceMotion else { return }

    let gravity = motion.gravity

    // Dispatch gravity updates to main queue, since they affect UI.
    DispatchQueue.main.async {
        self.gravity.gravityDirection = CGVector(dx: gravity.x * 3.5,
                                                 dy: -gravity.y * 3.5)
    }
}
```

## Play Variable Haptic Patterns on Collision

When the sphere collides with a wall, the dynamics framework sends a callback to its delegate. `ViewController` implements this delegate, `UICollisionBehaviorDelegate`, to respond to collisions.

To vary the haptic at the point of collision, check the sphere's velocity and map it to a normalized (0 to 1) intensity and sharpness value, as follows:

```
// Map the bounce velocity to intensity & sharpness.
let velocity = bounce.linearVelocity(for: item)
let xVelocity = Float(velocity.x)
let yVelocity = Float(velocity.y)

// Normalize magnitude to map one number to haptic parameters:
let magnitude = sqrtf(xVelocity * xVelocity + yVelocity * yVelocity)
let normalizedMagnitude = min(max(Float(magnitude) / kMaxVelocity, 0.0), 1.0)

// Create a haptic pattern player from normalized magnitude.
let hapticPlayer = try playerForMagnitude(normalizedMagnitude)
```

The sample sets `kMaxVelocity` based on experimentation dropping the sphere from a maximum possible height, but you can adjust the value to create a stronger or weaker intensity pattern.

On each impact, you can create a haptic player, an inexpensive operation, on the spot, using the desired haptic parameters.

```
let pattern = try CHHapticPattern(events: [audioEvent, hapticEvent], parameters: [])
return try engine.makePlayer(with: pattern)
```

Starting the player plays the haptic.

```
// Start player, fire and forget
try hapticPlayer?.start(atTime: CHHapticTimeImmediate)
```

## Synthesize Audio to Play With the Haptic

In addition to playing haptic patterns, the haptic engine also synthesizes audio. The engine in HapticBounce plays this audio by adding an audio event to the same haptic player as the haptic event, as follows:

```
let volume = linearInterpolation(alpha: magnitude, min: 0.1, max: 0.4)
let decay: Float = linearInterpolation(alpha: magnitude, min: 0.0, max: 0.1)
let audioEvent = CHHapticEvent(eventType: .audioContinuous, parameters: [
    CHHapticEventParameter(parameterID: .audioPitch, value: -0.15),
    CHHapticEventParameter(parameterID: .audioVolume, value: volume),
```

```
    CHHapticEventParameter(parameterID: .decayTime, value: decay),
    CHHapticEventParameter(parameterID: .sustained, value: 0)
], relativeTime: 0)
```

Instead of adjusting haptic intensity and sharpness based on the sphere's velocity, you vary audio volume and pitch.

# 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.

{} Updating Continuous and Transient Haptic Parameters in Real Time

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

`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.