Sample Code

Playing a Custom Haptic Pattern from a File

Sample predesigned Apple Haptic Audio Pattern files, and learn how to play your own.

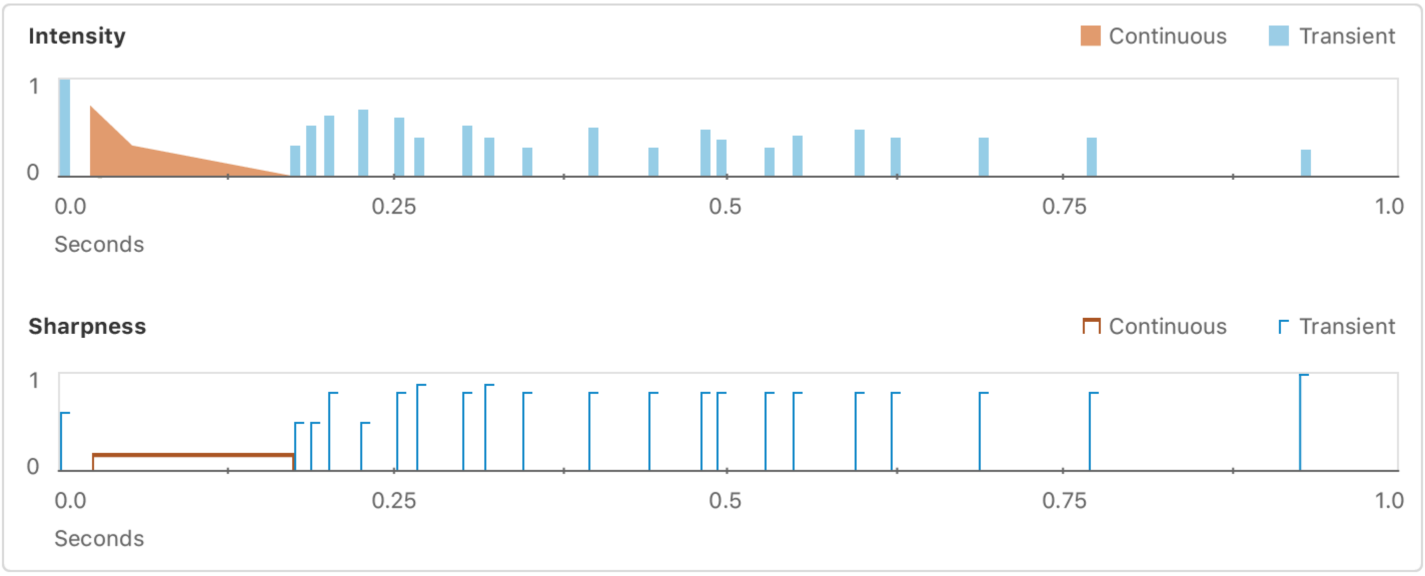
Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 13.3+

Overview

This app, HapticSampler, describes how to load files formatted as Apple Haptic Audio Pattern (AHAP). AHAP files are JSON-compliant dictionary representations of haptic and audio patterns that you can compose in a text editor, or any program that exports content in the JSON format.

The user interface, defined in `Main.storyboard`, consists of eight buttons, each of which loads and plays a different haptic pattern. AHAP files for eight custom patterns are bundled in the Xcode project. For example, the Sparkle.ahap file defines a haptic pattern with the following intensity and sharpness variations:



You can examine and modify the AHAP file parameters in Xcode or a text editor.

> **Note**
>
> HapticSampler 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

HapticSampler checks for device compatibility and sets up an instance of `CHHapticEngine` on launch.

```
// Create and configure a haptic engine.
do {
    // Associate the haptic engine with the default audio session
    // to ensure the correct behavior when playing audio-based haptics.
    let audioSession = AVAudioSession.sharedInstance()
    engine = try CHHapticEngine(audioSession: audioSession)
} catch let error {
    print("Engine Creation Error: \(error)")
}
```

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

Start the Haptic Engine

All AHAPs in the sample are included in the app bundle, so build a URL to the desired file by searching in the main bundle, as you might do for other included assets like audio files.

```swift
// Express the path to the AHAP file before attempting to load it.
guard let path = Bundle.main.path(forResource: filename, ofType: "ahap") else {
    return
}
```

Tap one of the buttons in the app to load the AHAP file associated with that button. Because the engine may not be started at that time, the app starts the engine as soon as a button is tapped.

```swift
// Start the engine in case it's idle.
try engine?.start()
```

Play the AHAP

Assuming the engine has started, call the engine's method, `playPattern(from:)`, passing it the URL to playing the file.

```swift
// Tell the engine to play a pattern.
try engine?.playPattern(from: URL(fileURLWithPath: path))
```

This method of playback follows a *fire and forget* model; each haptic pattern plays until it reaches its end, then stops automatically.

Once the haptic starts playing, you can't stop it, and pressing other buttons layers those haptics on top of any existing haptic patterns in the middle of playback. This layering allows you to combine a haptic pattern with a more continuous intent, like the rumble of thunder, with a more impulse-driven haptic pattern built from transient taps, like the strike of lightning.

Core Haptics layers simultaneously playing haptics automatically. If you don't want the layering, your app should wait out the duration of the first haptic before starting subsequent haptic players.

Note

Haptic patterns don't blend like audio waveforms, and not all combinations produce a discernible effect. Playing two haptic events of the same type at the same time makes them hard to tell apart. Experiment with various combinations to ensure that the result feels right.

Create Advanced Pattern Players for More Granular Control

To have more granular control over individual haptics, such as configuring their pausing, resuming, and seeking behavior, create and maintain advanced pattern players using `CHHapticAdvancedPatternPlayer` instead of using `playPattern(from:)`.

```
advancedPlayer = try hapticEngine.makeAdvancedPlayer(with: ahapPattern)
advancedPlayer.loopEnabled = true
```

Compose Your Own Haptic

To compose your own haptic pattern, begin by modifying one of the AHAP files included in the project. You can copy components of it for producing a transient haptic pattern or a continuous haptic pattern, then build your own haptic by combining and modifying those pieces.

Alternatively, you can write an AHAP file from scratch in a text editor, using the example AHAP files as a reference for proper nesting of event, pattern, and parameter dictionaries. For more information about the AHAP file format and the parameter keys that the format supports, see Representing Haptic Patterns in AHAP Files.

See Also

File-based haptics

Representing haptic patterns in AHAP files

Understand the Apple Haptic and Audio Pattern (AHAP) file format.