Sample Code

Capturing stereo audio from built-In microphones

Configure an iOS device's built-in microphones to add stereo recording capabilities to your app.

Download

iOS 17.0+ | iPadOS 17.0+ | Xcode 15.4+

Overview

Stereo audio uses two channels to create the illusion of multidirectional sound, adding greater depth and dimension to your audio and resulting in an immersive listening experience. iOS provides a number of ways to record audio from the built-in microphones, but until now it's been limited to mono audio only. Starting in iOS 14 and iPadOS 14, you can now capture stereo audio using the built-in microphones on supported devices.

Because a user can hold an iOS device in a variety of ways, you need to specify the orientation of the right and left channels in the stereo field. Set the built-in microphone's directionality by configuring:

- Polar pattern. The system represents the individual device microphones, and beamformers that use multiple microphones, as data sources. Select the front or back data source and set its polar pattern to stereo.

- Input orientation. When recording video, set the input orientation to match the video orientation. When recording audio only, set the input orientation to match the user interface orientation. In both cases, don't change the orientation during recording.

This sample app shows how to configure your app to record stereo audio, and helps you visualize changes to the input orientation and data-source selection.

Note

You must run the sample on a supported physical device running iOS 14, or later. To determine whether a device supports stereo recording, query the audio session's selected data source to see if its supportedPolarPatterns array contains the stereo polar pattern.

Configure the Audio Session Category

Recording stereo audio requires the app's audio session to use either the record or playAndRecord category. The sample uses the `playAndRecord` category so it can do both. It also passes the `defaultToSpeaker` and `allowBluetooth` options to route the audio to the speaker instead of the receiver, and to Bluetooth headphones.

```
func setupAudioSession() {
    do {
        let session = AVAudioSession.sharedInstance()
        try session.setCategory(.playAndRecord, options: [.defaultToSpeaker, .allowE
        try session.setActive(true)
    } catch {
        fatalError("Failed to configure and activate session.")
    }
}
```

Select and Configure a Built-In Microphone

An iOS device's built-in microphone input consists of an array of physical microphones and beamformers, each represented as an instance of `AVAudioSessionDataSourceDescription`. The sample app finds the built-in microphone input by querying the available inputs for the one where the port type equals the built-in microphone, and sets it as the preferred input.

```
private func enableBuiltInMic() {
    // Get the shared audio session.
    let session = AVAudioSession.sharedInstance()

    // Find the built-in microphone input.
    guard let availableInputs = session.availableInputs,
          let builtInMicInput = availableInputs.first(where: { $0.portType == .built
        print("The device must have a built-in microphone.")
        return
    }

    // Make the built-in microphone input the preferred input.
    do {
        try session.setPreferredInput(builtInMicInput)
    } catch {
        print("Unable to set the built-in mic as the preferred input.")
    }
}
```

Configure the Microphone Input's Directionality

To configure the microphone input's directionality, the sample sets its data source's preferred polar pattern and the session's input orientation. It performs this configuration in its `selectRecordingOption(_:orientation)` method, which it calls whenever the user rotates the device or changes the recording option selection.

```
func selectRecordingOption(_ option: RecordingOption, orientation: Orientation, comp

    // Get the shared audio session.
    let session = AVAudioSession.sharedInstance()

    // Find the built-in microphone input's data sources,
    // and select the one that matches the specified name.
    guard let preferredInput = session.preferredInput,
          let dataSources = preferredInput.dataSources,
          let newDataSource = dataSources.first(where: { $0.dataSourceName == optior
          let supportedPolarPatterns = newDataSource.supportedPolarPatterns else {
        completion(.none)
        return
    }

    do {
        isStereoSupported = supportedPolarPatterns.contains(.stereo)
        // If the data source supports stereo, set it as the preferred polar patterr
        if isStereoSupported {
            // Set the preferred polar pattern to stereo.
            try newDataSource.setPreferredPolarPattern(.stereo)
        }

        // Set the preferred data source and polar pattern.
        try preferredInput.setPreferredDataSource(newDataSource)
```

```
            // Update the input orientation to match the current user interface orientat
            try session.setPreferredInputOrientation(orientation.inputOrientation)

        } catch {
            fatalError("Unable to select the \(option.dataSourceName) data source.")
        }

        // Call the completion handler with the updated stereo layout.
        completion(StereoLayout(orientation: newDataSource.orientation!,
                                stereoOrientation: session.inputOrientation))
    }
```

This method finds the data source with the selected name, sets its preferred polar pattern to stereo, and then sets it as the input's preferred data source. Finally, it sets the preferred input orientation to match the device's user interface orientation.

See Also

System audio

Handling audio interruptions

Observe audio session notifications to ensure that your app responds appropriately to interruptions.

Responding to audio route changes

Observe audio session notifications to ensure that your app responds appropriately to route changes.

Routing audio to specific devices in multidevice sessions

Map audio channels to specific devices in multiroute sessions for recording and playback.

{} Adding synthesized speech to calls

Provide a more accessible experience by adding your app's audio to a call.

`class AVAudioSession`

An object that communicates to the system how you intend to use audio in your app.

`class AVAudioApplication`

An object that manages one or more audio sessions that belong to an app.

```
class AVAudioRoutingArbiter
```

An object for configuring macOS apps to participate in AirPods Automatic Switching.