Sample Code

Responding to changes in the flashing lights setting

Adjust your UI when a person chooses to dim flashing lights on their Apple device.

Download

iOS 16.4+ | iPadOS 16.4+ | Xcode 14.3+

Overview

Dim Flashing Lights is a setting that allows a person to indicate that they want to avoid bright, frequent flashes of light on their Apple device. When a person turns on this setting, the device automatically dims video content when the system detects flashing or strobing effects.

In your app, you can check the current value of this setting and make any necessary adjustments to your UI when the value changes. For example, this sample code project demonstrates how to draw a custom media timeline that indicates which portions of a video contain flashing lights. For other examples of changes you might make in your UI in response to this setting, read Flashing lights.

Important

This sample code project includes a sample video that contains sequences of flashing effects.

Configure the sample code project

This sample code project relies on changes to a system setting. To observe a change in the sample app's UI, change the value of the Dim Flashing Lights setting at any time, and return to the app to observe how the UI responds.

To change the setting, open Settings > Accessibility > Motion and turn on Dim Flashing Lights.

Play the video with dimmed flashing

With Dim Flashing Lights on, play the sample video in the video player. The sample app shows a custom media timeline below the video in the AVPlayer. This timeline annotates the segments of the sample video that present a high risk of flashing lights.

```
ForEach(videoFlashingTimes, id: \.self) { timeRange in
    // Draws red rectangles to indicate which segments of the video
    // contain flashing light effects. The position relies on the
    // start and end times of each segment in `videoFlashingTimes`.
    RoundedRectangle(cornerRadius: 2)
        .fill(.red)
        .position(x: (timeRange[0] * width) + ((timeRange[1] * width) - (timeRange[
        .frame(width: (timeRange[1] * width) - (timeRange[0] * width), height: 10)
}
```

View in Source

When the video playback reaches one of those segments, the system automatically dims the video to reduce the flashing effects.

Register for changes in the flashing lights setting

This sample app draws the custom media timeline only when the Dim Flashing Lights setting is on, which the app determines through a system notification. First, the app imports the Media Accessibility framework to interact with the API.

```
import MediaAccessibility
```

Then, it registers to receive the kMADimFlashingLightsChangedNotification notification, which the system sends when a person turns the setting on or off.

```
// Registers for the system notification that posts when the value of
// the Dim Flashing Lights setting changes.
NotificationCenter.default.addObserver(self,
    selector: #selector(dimFlashingLightsChanged),
    name: kMADimFlashingLightsChangedNotification as NSNotification.Name,
    object: nil)
```

View in Source

When the app receives the `kMADimFlashingLightsChangedNotification` notification, it sets the variable `dimFlashingLightsStatus` to the current value of the system setting, which the system reports through `MADimFlashingLightsEnabled()`.

```
@objc
func dimFlashingLightsChanged(_ notification: Notification) {
    // Updates `dimFlashingLightsStatus` to the new value of the
    // Dim Flashing Light setting.
    dimFlashingLightsStatus = MADimFlashingLightsEnabled()
}
```

View in Source

Draw the custom media timeline if the setting is on

When the app draws its UI, it checks the value of the current setting status, `dimFlashingLightsStatus`, to determine whether to draw the custom media timeline.

```
// Checks the status of the Dim Flashing Lights setting to determine
// whether to draw the custom media timeline.
if playerManager.dimFlashingLightsStatus {
    VStack {
        Text("Custom Media Timeline")
            .font(.title2)
            .padding(.bottom)
        GeometryReader { proxy in
            // Draws a timeline view that's the same width as the
            // video player.
            FlashingTimelineView(proxy.size.width)

            // Draws a custom playback indicator on top of the
            // timeline view.
            RoundedRectangle(cornerRadius: 2)
                .fill(.black)
                .border(.white)
                .position(x: playerManager.playbackPercentage * proxy.size.width)
                .frame(width: 9, height: 9)
        }
    }
```

```
        .padding(.top)
```

View in Source

If you turn the setting off, the sample app’s UI updates to remove the custom media timeline.

See Also

Dim flashing lights

`func MADimFlashingLightsEnabled() -> Bool`

Returns a Boolean value that indicates whether the flashing lights setting is enabled on the device.

`let kMADimFlashingLightsChangedNotification: CFString`

A notification that posts when a person changes the flashing lights setting on the device.

`class MAFlashingLightsProcessor`

A class that processes a framebuffer object to detect and dim sequences of flashing lights.