Article

Adopting smart framing in your camera app

Capture the optimal shot by providing automatic framing recommendations.

Overview

Smart framing detects when people enter or leave the scene, and frames the shot to compose a great photo. Available in iPhone 17, iPhone Air, and iPhone 17 Pro, the front Ultra Wide camera enables dynamic aspect ratio changes without interrupting the camera's preview. You can add smart framing to your app by adopting `AVCaptureSmartFramingMonitor`.

The `AVCaptureSmartFramingMonitor` class monitors a supported camera's field of view and provides aspect ratio and zoom factor suggestions for better photographic composition. When optimal framing opportunities occur, the monitor provides recommendations through its `recommendedFraming` property.

Find a smart framing-capable device

Start by verifying whether the person's device supports smart framing:

```
func findAndConfigureSmartFramingDevice() -> AVCaptureDevice? {
    // Configure the discovery session to find an Ultra Wide front camera.
    let discoverySession = AVCaptureDevice.DiscoverySession(
        deviceTypes: [.builtInUltraWideCamera],
        mediaType: .video,
        position: .front
    )

    // Find the first matching device and format that supports smart framing.
```

```
    guard let device = discoverySession.devices.first,
          let format = device.formats.first(where: \.isSmartFramingSupported) else {
        // Return nil if the device doesn't have an Ultra Wide front camera.
        return nil
    }

    if device.activeFormat.isSmartFramingSupported {
        // Return the device if already configured for smart framing.
        return device
    } else {
        // Otherwise, attempt to configure the device and return it, if successful.
        do {
            try device.lockForConfiguration()
            defer { device.unlockForConfiguration() }
            // Set the smart framing format as active.
            device.activeFormat = format
            return device
        } catch {
            return nil
        }
    }
}
```

The above code uses an `AVCaptureDevice.DiscoverySession` object to find the front Ultra Wide camera, and validates that it provides an `AVCaptureDevice.Format` that supports the feature. If the device configuration supports smart framing, it returns the capture device. Otherwise, it sets the `activeFormat` and then returns the configured device.

Configure the smart framing monitor

After finding a supported capture device, configure its monitoring behavior by accessing the device's associated `AVCaptureSmartFramingMonitor` object:

```
func configureSmartFraming() async {
    // Access the smart framing monitor, if available.
    guard let monitor = currentDevice?.smartFramingMonitor else { return }
    // Enable all supported framings.
    monitor.enabledFramings = monitor.supportedFramings
}
```

By default, the smart framing monitor isn't configured to provide recommendations. Attempting to start it in this state results in an error. To monitor the scene, specify the framings the monitor should consider by setting its `enabledFramings` property value. The example above enables all the monitor's `supportedFramings`, but you can enable a subset of these values, depending on the needs of your app. For example, it could limit the enabled framings to `ratio3x4` and `ratio4x3` for photo capture, or `ratio9x16` and `ratio16x9` for video.

Monitor framing recommendations

With the smart framing monitor configured, it's ready to start generating framing recommendations. To respond to new recommendations, key-value observe the monitor's `recommendedFraming` property like shown below:

```
func startMonitoring() async {
    // Return early if the monitor doesn't exist or is currently monitoring.
    guard let monitor = currentDevice?.smartFramingMonitor, !monitor.isMonitoring el

    // Key-value observe changes to the monitor's framing recommendations.
    framingObservation = monitor.observe(\.recommendedFraming, options: [.new]) { [w
        // Access the recommended framing.
        guard let self, let framing = monitor.recommendedFraming else { return }
        // Configure the device with the latest recommendation.
        Task { await self.applyRecommendedFraming(framing) }
    }

    do {
        // Start monitoring for framing recommendations.
        try monitor.startMonitoring()
    } catch {
        logger.error("Unable to start monitoring: \(error)")
    }
}
```

The code example configures an observer to respond to new framing recommendations. When it observes a new value, it calls a method to apply the framing to the capture device. After configuring the observer, the example calls the monitor's `startMonitoring()` method to generate framing recommendations.

To disable smart framing, and allow a person to manually frame their shot, invalidate your key-value observation and call the monitor's `stopMonitoring()` method, as shown here:

```
func stopMonitoring() async {
    // Stop key-value observing the monitor.
    framingObservation?.invalidate()
    framingObservation = nil

    guard let monitor = currentDevice?.smartFramingMonitor else { return }
    // Stop monitoring for smart framing recommendations.
    monitor.stopMonitoring()
}
```

Calling this method causes the monitor to stop generating recommendations until you restart monitoring.

> **Note**
>
> You can start and stop the smart framing monitor independently of the capture session, which lets you toggle the feature state without interrupting the video stream.

Apply framing recommendations

When the monitor generates new framing recommendations, apply them by setting the device's dynamic aspect ratio and video zoom factor as follows:

```
private func applyRecommendedFraming(_ framing: AVCaptureFraming) async {
    guard let device = currentDevice else { return }
    do {
        // Request exclusive control of the device.
        try device.lockForConfiguration()
        do {
            // Set the recommended aspect ratio and zoom factor.
            try await device.setDynamicAspectRatio(framing.aspectRatio)
            device.videoZoomFactor = CGFloat(framing.zoomFactor)
        } catch {
            logger.error("Failed to set aspect ratio: \(error)")
        }
        // Release exclusive control of the device.
        device.unlockForConfiguration()
    } catch {
        logger.error("Failed to lock device for configuration: \(error)")
    }
```

```
}
```

The order of applying aspect ratio and zoom factor affects the visual transition. Apply the aspect ratio change first, then update the zoom factor to provide a smooth transition between framings.

See Also

Capture devices

Choosing a capture device

Select the front or back camera, or use advanced features like the TrueDepth camera or dual camera.

`class AVCaptureDevice`

An object that represents a hardware or virtual capture device like a camera or microphone.

`class AVCaptureDeviceInput`

An object that provides media input from a capture device to a capture session.

`class AVContinuityDevice`

A class that represents a physical iOS device that's nearby and can provide access to its cameras and microphones.

`class AVExternalStorageDevice`

Represents a physical external storage device that stores media assets.

`class AVExternalStorageDeviceDiscoverySession`

Informs your app when the external storage devices connect to and disconnect from the system.