Article

Accessing the camera while multitasking on iPad

Operate the camera in Split View, Slide Over, Picture in Picture, and Stage Manager modes.

Overview

Camera access on iPad is normally limited to apps running in full-screen mode. If your app enters a multitasking mode, such as Split View or Stage Manager, the system disables the camera by default. In iPadOS 16 and later, your app can enable using the camera while multitasking.

Related Sessions from WWDC22

Session 110429: Discover advancements in iOS camera capture: Depth, focus, and multitasking

Multitasking modes enable people to work with multiple apps at the same time. There are four kinds:

- Split View mode accommodates two different apps, or two windows from the same app, by splitting the screen into resizable views.
- Slide Over mode lets users work on an app that slides in front of any open apps.
- Picture in Picture mode displays a draggable window over an app.
- Stage Manager lets users resize windows and see multiple overlapping windows in a single view, group apps for specific tasks or projects, and drag windows between iPad and an externally connected display.

When you enable multitasking camera access, your app can run alongside other foreground apps and it no longer receives `AVCaptureSession.InterruptionReason.videoDeviceNotAvailableWithMultipleForegroundApps` as an interruption reason.

When operating the camera capture system, using the multitasking feature introduces the possibility of performance degradation because of other apps consuming resources like memory, CPU, and GPU. Increased device temperature and power usage can lead to frame drops or poor capture quality.

> **Important**
>
> Enabling multitasking camera access isn't recommended if your app uses resource-intensive capabilities such as 4K video capture, or Apple ProRAW or Deep Fusion image capture. Even if your app doesn't use resource-intensive capabilities, you should test it alongside other resource-intensive apps and with Stage Manager to determine if there are performance issues.

For information about Split View and Slide Over modes, see Use multitasking on your iPad. For information about Stage Manager on iPad, see Move, resize, and organize windows with Stage Manager on iPad.

Enable camera access while multitasking

Set up a capture session to enable your app to capture photos or videos. See Setting up a capture session for more information. You can configure a capture session to allow use of the camera while multitasking if the current environment supports it. To determine if a capture session supports this feature, query its `isMultitaskingCameraAccessSupported`. If this value is `true`, you can enable multitasking camera access by setting `isMultitaskingCameraAccessEnabled` to `true`, as the example below shows:

```
let captureSession = AVCaptureSession()

// Configure the capture session.
captureSession.beginConfiguration()

if captureSession.isMultitaskingCameraAccessSupported {
    // Enable use of the camera in multitasking modes.
    captureSession.isMultitaskingCameraAccessEnabled = true
}
captureSession.commitConfiguration()
```

```
// Start the capture session.
```

Enabling your app to use the camera while multitasking extends to Picture in Picture mode for video calls using AVKit. See Adopting Picture in Picture for video calls to learn more.

While multitasking, after an app finishes recording a video with `AVCaptureMovieFileOutput` or `AVAssetWriter`, the system displays an alert one time only to inform the user about the potential for lower-quality videos.

> **Important**
>
> Apps that have a deployment target earlier than iOS 16 require the `com.apple.developer.avfoundation.multitasking-camera-access` entitlement to enable accessing the camera while multitasking.

Configure your app for Split View or Slide Over mode

When you enable multitasking camera access, your app doesn't need to run in full-screen mode to use the camera. When it does run in full-screen mode, the camera continues to function in Slide Over mode and when the system presents a Picture in Picture window over your app. If you want to also run your app in Split View or Slide Over mode, follow these steps:

1. Select your app target in Xcode's project editor.
2. Click the General tab.
3. Deselect the "Requires full screen" checkbox.

Respond to system pressure

Make your app resilient to increasing system pressure by monitoring the `systemPressureState` property on `AVCaptureDevice`, and take action to reduce the impact. When the pressure reaches excessive levels, the capture system shuts down and emits an `wasInterruptedNotification` notification.

Apps can reduce their footprint on the system by lowering the frame rate or requesting lower-resolution, binned, or non-HDR formats. The following code, from the sample app AVMultiCamPiP: Capturing from Multiple Cameras, shows how to reduce the capture frame rate:

```
let systemPressureStateObservation = observe(\.self.device.systemPressureState,
                                             options: .new) { [weak self] _, change
    guard let self = self else { return }
```

```
    guard let systemPressureState = change.newValue else { return }

    // The frame rates here are for demonstrative purposes only.
    // Frame rate throttling can differ depending on your app's camera configuration
    let pressureLevel = systemPressureState.level
    if pressureLevel == .serious || pressureLevel == .critical {
        do {
            try self.device.lockForConfiguration()

            print("WARNING: Reached elevated system pressure level: \(pressureLevel)

            self.device.activeVideoMinFrameDuration = CMTimeMake(value: 1, timescale
            self.device.activeVideoMaxFrameDuration = CMTimeMake(value: 1, timescale

            self.device.unlockForConfiguration()
        } catch {
            print("Could not lock device for configuration: \(error)")
        }
    } else if pressureLevel == .shutdown {
        print("Session stopped running due to system pressure level.")
    }
}
```

Handle camera use interruptions

The system only allows one app to use the device's camera at a time. Prepare your app to respond when another app starts using the camera. For example, if your app is running in Stage Manager and another app utilizes the camera, the system suspends your app's use of the camera until the other app finishes. When the system interrupts your app's use of the camera, it notifies your app so you can update your user interface.

To enable the system to notify you when your app's camera access changes, observe the notifications `wasInterruptedNotification` and `interruptionEndedNotification`, as the example below shows:

```
func addObservers() {
    let nc = NotificationCenter.default

    // Observe when the system interrupts the capture session.
    nc.addObserver(self,
                   selector: #selector(handleInterruptionStarted),
                   name: .AVCaptureSessionWasInterrupted,
                   object: captureSession)
```

```
    // Observe when the capture session interruption ends.
    nc.addObserver(self,
                   selector: #selector(handleInterruptionEnded),
                   name: .AVCaptureSessionInterruptionEnded,
                   object: captureSession)
}
```

The notification object's user information dictionary contains the reason for an interruption. Determining the reason lets you configure your user interface as your camera access changes. Use `AVCaptureSessionInterruptionReasonKey` to look up the value, as the example below shows:

```
@objc func handleInterruptionStarted(notification: Notification) {
    guard let userInfo = notification.userInfo,
          let reasonValue = userInfo[AVCaptureSessionInterruptionReasonKey] as? Int,
          let reason = AVCaptureSession.InterruptionReason(rawValue: reasonValue) el
        print("Failed to parse the interruption reason.")
        return
    }

    switch reason {
    case .videoDeviceInUseByAnotherClient:
        // Show your app's interruption user interface.
    case .videoDeviceNotAvailableDueToSystemPressure:
        // Handle an interruption from increasing system pressure.
    default:
        // Handle other interruption reasons.
    }
}
```

See Also

Picture in Picture

{} Adopting Picture in Picture Playback in tvOS

Add advanced multitasking capabilities to your video apps by using Picture in Picture playback in tvOS.

Adopting Picture in Picture in a Standard Player

Add Picture in Picture (PiP) playback to your app using a player view controller.

Adopting Picture in Picture in a Custom Player

Add controls to your custom player user interface to invoke Picture in Picture (PiP) playback.

Adopting Picture in Picture for video calls

Add multitasking capability to your video-call apps by using Picture in Picture (PiP).

`class` `AVPictureInPictureController`

A controller that responds to user-initiated Picture in Picture playback of video in a floating, resizable window.