Article

Adopting Picture in Picture for video calls

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

Overview

Use PiP in your video-call apps so users can multitask with other apps while on video calls. When a user enables PiP, your app scales down to a corner of the screen, so they can see the Home Screen and interact with other apps. In iOS 15 and later, AVKit provides PiP support for video-calling apps, which enables you to deliver a familiar video-calling experience that behaves like FaceTime.

Important

In iOS 16 and later, you can use the camera in Picture in Picture mode by enabling a capture session's `isMultitaskingCameraAccessEnabled` property. Apps that have a deployment target earlier than iOS 16 require the `com.apple.developer.avfoundation.multitasking-camera-access` entitlement to use the camera in PiP mode.

Create a source view

Providing PiP support begins by choosing a source view to display inside the video-call view controller. You need to add a `UIView` to `AVPictureInPictureVideoCallViewController`, so use `AVCaptureVideoPreviewLayer` or `AVSampleBufferDisplayLayer` depending on your need. In iOS 18 and later, you may also use `MTKView` as your source view. Video-calling apps need to display the remote view, so use `AVSampleBufferDisplayLayer` to do so.

```
class SampleBufferVideoCallView: UIView {
    override class var layerClass: AnyClass {
        AVSampleBufferDisplayLayer.self
    }

    var sampleBufferDisplayLayer: AVSampleBufferDisplayLayer {
        layer as! AVSampleBufferDisplayLayer
    }
}
```

Create a video-call controller

To display your source view, create a AVPictureInPictureVideoCallViewController and add your source as a subview.

```
let pipVideoCallViewController = AVPictureInPictureVideoCallViewController()
pipVideoCallViewController.preferredContentSize = CGSize(width: 1080, height: 1920)
pipVideoCallViewController.view.addSubview(sampleBufferVideoCallView)
```

Use isPictureInPictureSupported() to determine whether the current device supports PiP playback. If PiP isn't supported on the current device, attempting to initialize a PiP controller returns nil.

Create a PiP controller using a content source

Before you create an AVPictureInPictureController, you need to create an AVPictureInPictureController.ContentSource that represents the source of the content the system displays. A content source requires a video-call view controller, and a source view that contains the content you associate with the video call.

```
let pipContentSource = AVPictureInPictureController.ContentSource(
                            activeVideoCallSourceView: videoCallViewSourceView,
                            contentViewController: pipVideoCallViewController)
```

Important

Avoid unintentionally starting PiP by setting the content source on your PiP controller to nil or by releasing your PiP controller, when the active call ends.

After creating a content source, use it to initialize AVPictureInPictureController. By default, PiP starts when a user moves to the background if your source view is full-screen, or you set canStartPictureInPictureAutomaticallyFromInline to true. If your app is in the foreground, you can start PiP by calling startPictureInPicture().

```
let pipController = AVPictureInPictureController(contentSource: pipContentSource)
pipController.canStartPictureInPictureAutomaticallyFromInline = true
pipController.delegate = self
```

The system uses the source view to determine the source frame for the PiP animation, and the restore target for either when the user returns to the app or PiP stops.

> Note
>
> The PiP window doesn't receive touch events when you use AVPictureInPictureVideoCallViewController, so you can't customize the window's user interface by adding buttons.

Observe PiP life cycle events

When you use PiP, you respond to life-cycle events by observing AVPictureInPictureControllerDelegate. This allows you to handle your app's user interface based on the PiP state, along with observing for potential errors.

The system interrupts your capture session when the system or user stashes PiP, so observe wasInterruptedNotification for AVCaptureSession.InterruptionReason.videoDeviceNotAvailableInBackground to handle the interruption.

When your app is in PiP mode, it can't assume control of the camera. For example, Camera.app assumes control of the camera when it's opened, and the system returns camera control when Camera.app finishes with it. You observe wasInterruptedNotification for AVCaptureSession.InterruptionReason.videoDeviceInUseByAnotherClient to handle the interruption.

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.

Accessing the camera while multitasking on iPad

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

`class AVPictureInPictureController`

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