Advances in iOS Photography
Live Photo, RAW, and Wide Color Capture with AV Foundation
Session 501

Brad Ford Apple
# Past Sessions

developer.apple.com

<table>
<thead>
<tr>
<th>Session Title</th>
<th>WWDC Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New In Camera Capture (iOS 6)</td>
<td>WWDC 2012</td>
</tr>
<tr>
<td>What’s New In Camera Capture (iOS 7)</td>
<td>WWDC 2013</td>
</tr>
<tr>
<td>Camera Capture Manual Controls (iOS 8 / Yosemite)</td>
<td>WWDC 2014</td>
</tr>
</tbody>
</table>
Agenda
Agenda

New AVCaptureOutput
Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!
Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

• Live Photos
Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

• Live Photos
• RAW and DNG
Agenda

New AVCaptureOutput
New photography-driven features in iOS 10!
• Live Photos
• RAW and DNG
• Preview (Thumbnail) images
Agenda

New AVCaptureOutput

New photography-driven features in iOS 10!

• Live Photos
• RAW and DNG
• Preview (Thumbnail) images
• Wide Color
Review: AV Foundation Capture Objects
Review: AV Foundation Capture Objects
Review: AV Foundation Capture Objects

AVCaptureDeviceInput

AVCaptureDevice (Camera)

AVCaptureSession
Review: AV Foundation Capture Objects

AVCaptureDevice (Camera)

AVCaptureDeviceInput

AVCaptureSession

AVCaptureStillImageOutput  AVCaptureMovieFileOutput
Review: AV Foundation Capture Objects

AVCaptureDevice (Camera)

AVCaptureDeviceInput

AVCaptureConnection

AVCaptureConnection

AVCaptureSession

AVCaptureStillImageOutput

AVCaptureMovieFileOutput
Review: AV Foundation Capture Objects

- AVCaptureDevice (Camera)
- AVCaptureDeviceInput
- AVCaptureConnection
- AVCaptureConnection
- AVCaptureSession
  - AVCaptureStillImageOutput
  - AVCaptureMovieFileOutput
Review: AV Foundation Capture Objects

AVCaptureDevice (Camera)

AVCaptureDevice Input

AVCaptureConnection

AVCaptureConnection

AVCaptureSession

AVCaptureMovieFileOutput
Review: AV Foundation Capture Objects

AVCaptureDeviceInput

AVCaptureDevice (Camera)

AVCaptureConnection

AVCaptureSession

AVCaptureConnection

AVCaptureMovieFileOutput
Review: AV Foundation Capture Objects

- AVCaptureDevice (Camera)
- AVCaptureDeviceInput
- AVCaptureConnection
- AVCaptureConnection
- AVCaptureSession
- AVCapturePhotoOutput
- AVCaptureMovieFileOutput
AVCapturePhotoOutput Design Features
AVCapturePhotoOutput Design Features

Functional programming model
AVCapturePhotoOutput Design Features

Functional programming model
Photo settings encapsulation
AVCapturePhotoOutput Design Features

- Functional programming model
- Photo settings encapsulation
- A delegate-style interface for tracking the progress of photo capture requests
AVCapturePhotoOutput Design Features

Functional programming model
Photo settings encapsulation
A delegate-style interface for tracking the progress of photo capture requests
Resolving of photo settings to an immutable object
Using AVCapturePhotoOutput

- Read-only properties
- Feature opt-in properties
- Methods
Using AVCapturePhotoOutput

**AVCapturePhotoOutput**

- **isLivePhotoCaptureSupported**
- **availablePhotoPixelFormatTypes**
- **availablePhotoCodecTypes**

**Feature opt-in properties**

**Methods**
Using AVCapturePhotoOutput

AVCapturePhotoOutput

- isLivePhotoCaptureSupported
- availablePhotoPixelFormatTypes
- availablePhotoCodecTypes

- isHighResolutionCaptureEnabled
- isLivePhotoCaptureEnabled

Methods
Using AVCapturePhotoOutput

AVCapturePhotoOutput

- isLivePhotoCaptureSupported
- availablePhotoPixelFormatTypes
- availablePhotoCodecTypes

- isHighResolutionCaptureEnabled
- isLivePhotoCaptureEnabled

- capturePhoto(with:, delegate:)

NEW
Using AVCapturePhotoOutput
Using AVCapturePhotoOutput

AVCapturePhotoOutput

AVCapturePhotoSettings

- FeatureX
- FeatureY
- FeatureZ
Using AVCapturePhotoOutput

AVCapturePhotoOutput

-capturePhoto(with:, AVCapturePhotoSettings)

AVCapturePhotoSettings

- FeatureX
- FeatureY
- FeatureZ
Using AVCapturePhotoOutput

- capturePhoto(with:, AVCapturePhotoCaptureDelegate)

AVCapturePhotoOutput

AVCapturePhotoSettings
- FeatureX
- FeatureY
- FeatureZ

AVCapturePhotoCaptureDelegate
- photoEventAHappened
- photoEventBHappened
- photoEventCHappened
Using AVCapturePhotoOutput

AVCapturePhotoOutput
-capturePhoto(with:, delegate:)

AVCapturePhotoSettings
- FeatureX
- FeatureY
- FeatureZ

AVCapturePhotoCaptureDelegate
- photoEventAHappened
- photoEventBHappened
- photoEventCHappened
AVCapturePhotoSettings

AVCapturePhotoCaptureDelegate
- photoEventAHappened
- photoEventBHappened
- photoEventCHappened

AVCapturePhotoSettings
- FeatureX
- FeatureY
- FeatureZ
AVCapturePhotoSettings

AVCapturePhotoSettings
- FeatureX
- FeatureY
- FeatureZ

AVCapturePhotoCaptureDelegate
- photoEventAHappened
- photoEventBHappened
- photoEventCHappened
AVCapturePhotoSettings

Atomic

AVCapturePhotoSettings

<table>
<thead>
<tr>
<th>FeatureX</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureY</td>
</tr>
<tr>
<td>FeatureZ</td>
</tr>
</tbody>
</table>

AVCapturePhotoCaptureDelegate

<table>
<thead>
<tr>
<th>photoEventAHappened</th>
</tr>
</thead>
<tbody>
<tr>
<td>photoEventBHappened</td>
</tr>
<tr>
<td>photoEventCHappened</td>
</tr>
</tbody>
</table>
AVCapturePhotoSettings

Atomic
Unique

AVCapturePhotoSettings
- FeatureX
- FeatureY
- FeatureZ

AVCapturePhotoCaptureDelegate
- photoEventAHappened
- photoEventBHappened
- photoEventCHappened
Atomic
Unique
Your order form copy
AVCapturePhotoSettings

AVCapturePhotoCaptureDelegate
- photoEventAHappened
- photoEventBHappened
- photoEventCHappened

FeatureX
FeatureY
FeatureZ
AVCapturePhotoSettings

Single set of callbacks per photo settings
AVCapturePhotoSettings

Single set of callbacks per photo settings
Ordering is documented
AVCapturePhotoSettings

Single set of callbacks per photo settings
Ordering is documented
Vehicle for resolving indeterminate settings
AVCapturePhotoCaptureDelegate Usage
AVCapturePhotoCaptureDelegate Usage

capturePhoto(with: delegate: flashMode = .auto, autoSIS = true AVCapturePhotoCaptureDelegate)
AVCapturePhotoCaptureDelegate Usage

flashMode = .auto, autoSIS = true
AVCapturePhotoCaptureDelegate Usage

- `flashMode = .auto, autoSIS = true`
- `AVCaptureResolvedPhotoSettings
  flash = On, SIS = Off`

`AVCapturePhotoCaptureDelegate`

`willBeginCaptureForResolvedSettings`
AVCapturePhotoCaptureDelegate Usage

Time

`flashMode = .auto, autoSIS = true`

`willBeginCaptureForResolvedSettings`

`AVCaptureResolvedPhotoSettings`  
`flash = On, SIS = Off`
AVCapturePhotoCaptureDelegate Usage

Time!

- flashMode = .auto, autoSIS = true

willCapturePhotoForResolvedSettings

AVCapturePhotoCaptureDelegate
AVCapturePhotoCaptureDelegate Usage

Time:

- flashMode = .auto, autoSIS = true
- AVCapturePhotoCaptureDelegate
- didCapturePhotoForResolvedSettings
`AVCapturePhotoCaptureDelegate Usage`

- `flashMode = .auto, autoSIS = true`
- `CMSampleBuffer`
- `didFinishProcessingPhotoSampleBuffer`
- `AVCapturePhotoCaptureDelegate`
AVCapturePhotoCaptureDelegate Usage

flashMode = .auto, autoSIS = true

didFinishCaptureForResolvedSettings

AVCapturePhotoCaptureDelegate
AVCapturePhotoCaptureDelegate Specifics
AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request
AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

AVCapturePhotoOutput holds a weak reference to your delegate
AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

`AVCapturePhotoOutput` holds a weak reference to your delegate

All callbacks are marked `@optional`
AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

`AVCapturePhotoOutput` holds a weak reference to your delegate

All callbacks are marked `@optional`

Some callbacks are required at runtime depending on your photo settings
AVCapturePhotoCaptureDelegate Specifics

Delegate callbacks track a single photo capture request

**AVCapturePhotoOutput** holds a weak reference to your delegate

All callbacks are marked @optional

Some callbacks are required at runtime depending on your photo settings

All callbacks pass an instance of **AVCaptureResolvedPhotoSettings**
```
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto() {
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto() {
    let settings = AVCapturePhotoSettings()
    settings.flashMode = .auto

    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto() {
    let bgraFormat: [String : AnyObject] = [kCVPixelBufferPixelFormatTypeKey as String :
                                         NSNumber(value: kCVPixelFormatType_32BGRA)]

    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```
// Initiating a photo capture using AVCapturePhotoOutput

```swift
func takeHighResolutionPhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true
    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.flashMode = .auto
    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [
        kCVPixelBufferPixelFormatTypeKey as String :
        NSNumber(value: kCVPixelFormatType_32BGRA)]
    let settings = AVCapturePhotoSettings(format: bgraFormat)
    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.flashMode = .auto

    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [kCVPixelBufferPixelFormatTypeKey as String :
                                          NSNumber(value: kCVPixelFormatType_32BGRA)]

    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Initiating a photo capture using AVCapturePhotoOutput

func takeHighResolutionPhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeFlashPhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.flashMode = .auto

    photoOutput.capturePhoto(with: settings, delegate: self)
}

func takeBGRAPhoto()
{
    let bgraFormat: [String : AnyObject] = [
        kCVPixelBufferPixelFormatTypeKey as String :
        NSNumber(value: kCVPixelFormatType_32BGRA)]

    let settings = AVCapturePhotoSettings(format: bgraFormat)

    photoOutput.capturePhoto(with: settings, delegate: self)
}
AVCaptureResolvedPhotoSettings Properties
AVCaptureResolvedPhotoSettings Properties

public var uniqueID: Int64 { get }
AVCaptureResolvedPhotoSettings Properties

```swift
public var uniqueID: Int64 { get }
public var photoDimensions: CMVideoDimensions { get }
```
AVCaptureResolvedPhotoSettings Properties

public var uniqueID: Int64 { get }
public var photoDimensions: CMVideoDimensions { get }
public var isFlashEnabled: Bool { get }
AVCaptureResolvedPhotoSettings Properties

public var uniqueID: Int64 { get }
public var photoDimensions: CMVideoDimensions { get }
public var isFlashEnabled: Bool { get }
public var isStillImageStabilizationEnabled: Bool { get }
Bracketed Capture Support
Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls
Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets
Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls
Auto Exposure and Manual Exposure Brackets
AVCapturePhotoBracketSettings is a subclass of AVCapturePhotoSettings
Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a subclass of AVCapturePhotoSettings

```swift
public var bracketedSettings: [AVCaptureBracketedStillImageSettings] { get }
public var isLensStabilizationEnabled: Bool
```
Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a subclass of AVCapturePhotoSettings

```swift
public var bracketedSettings: [AVCaptureBracketedStillImageSettings] { get }
public var isLensStabilizationEnabled: Bool

func capture(_ captureOutput: AVCapturePhotoOutput,
            didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
            previewPhotoSampleBuffer: CMSampleBuffer?,
            resolvedSettings: AVCaptureResolvedPhotoSettings,
            bracketSettings: AVCaptureBracketedStillImageSettings?,
            error: NSError?)
{
    // bracketSettings tells you which item in your array of bracketedSettings
    // this image corresponds to.
}
```
Bracketed Capture Support

Review 2014 Session 508: Camera Capture Manual Controls

Auto Exposure and Manual Exposure Brackets

AVCapturePhotoBracketSettings is a subclass of AVCapturePhotoSettings

```swift
public var bracketedSettings: [AVCaptureBracketedStillImageSettings] { get }
public var isLensStabilizationEnabled: Bool

func capture(_ captureOutput: AVCapturePhotoOutput,
            didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
            previewPhotoSampleBuffer: CMSampleBuffer?,
            resolvedSettings: AVCaptureResolvedPhotoSettings?,
            bracketSettings: AVCaptureBracketedStillImageSettings?,
            error: NSError?)
{
    // bracketSettings tells you which item in your array of bracketedSettings
    // this image corresponds to.
}
```
Deprecated in iOS 10

- AVCaptureStillImageOutput
- AVCaptureDevice.flashActive
- AVCaptureDevice.isFlashModeSupported(_:)
- AVCaptureDevice.flashMode
<table>
<thead>
<tr>
<th>Function/Property</th>
<th>Deprecated in iOS 10</th>
<th>Equivalent in iOS 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVCaptureStillImageOutput</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>AVCaptureDevice.flashActive</td>
<td>×</td>
<td>AVCapturePhotoOutput.isFlashScene</td>
</tr>
<tr>
<td>AVCaptureDevice.isFlashModeSupported(::_)</td>
<td>×</td>
<td>AVCapturePhotoOutput.supportedFlashModes</td>
</tr>
<tr>
<td>AVCaptureDevice.flashMode</td>
<td>×</td>
<td>AVCapturePhotoSettings.flashMode</td>
</tr>
</tbody>
</table>
AVCapturePhotoOutput Benefits
AVCapturePhotoOutput Benefits

Easier bookkeeping
AVCapturePhotoOutput Benefits

- Easier bookkeeping
- Immediate settings resolution
AVCapturePhotoOutput Benefits

Easier bookkeeping
Immediate settings resolution
Confident request tracking
AVCapturePhotoOutput Benefits

- Easier bookkeeping
- Immediate settings resolution
- Confident request tracking
- Expandable palette of callbacks
Live Photos
Capturing memories

“A still photo captures an instant frozen in time. With Live Photos, you can turn those instants into unforgettable living memories.”
What is a Live Photo

Moment or memory?
What is a Live Photo

Moment or memory?

It’s a moment!
What is a Live Photo

Moment or memory?

It’s a moment!

- Full resolution still image (12 MP JPEG)
What is a Live Photo

Moment or memory?

It's a moment!

• Full resolution still image (12 MP JPEG)
• Same quality as non-Live Photo (SIS / OIS)
What is a Live Photo

Moment or memory?

It’s a moment!

- Full resolution still image (12 MP JPEG)
- Same quality as non-Live Photo (SIS / OIS)
- “Frictionless” capture
What is a Live Photo

Moment or memory?
What is a Live Photo

Moment or memory?

It’s a memory!
What is a Live Photo

Moment or memory?

It’s a memory!

• A short movie encompassing the still time
What is a Live Photo
Moment or memory?

It’s a memory!
- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)
What is a Live Photo

Moment or memory?

It’s a memory!

• A short movie encompassing the still time
• Screen resolution (1440x1080 or 1290x960)
• Includes audio
What is a Live Photo

Moment or memory?

It’s a memory!

• A short movie encompassing the still time
• Screen resolution (1440x1080 or 1290x960)
• Includes audio

In iOS 9.1 — “Shoe shot” auto trimming
What is a Live Photo

Moment or memory?

It’s a memory!

- A short movie encompassing the still time
- Screen resolution (1440x1080 or 1290x960)
- Includes audio

In iOS 9.1 — “Shoe shot” auto trimming

New in iOS 10 — Video stabilization
What is a Live Photo

Moment or memory?

It’s a memory!

• A short movie encompassing the still time
• Screen resolution (1440x1080 or 1290x960)
• Includes audio

In iOS 9.1 — “Shoe shot” auto trimming
New in iOS 10 — Video stabilization
New in iOS 10 — Interruption-free music during capture
What is a Live Photo

It’s a moment and a memory

JPEG

MOV
Live Photo .MOV Asset

Video
H.264, 1440x1080, 15 FPS, 7 Mbps

Audio
44.1 kHz, PCM, Mono or Stereo

Metadata

Photo Display Time

0.75
1.5
2.25
3
Capturing Live Photos
Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`
Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`
Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`.

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`.

Opt in using `AVCapturePhotoOutput.isLivePhotoCaptureEnabled = true`.
Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`

Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`

Opt in using `AVCapturePhotoOutput.isLivePhotoCaptureEnabled = true`

Add an `AVCaptureDeviceInput` for the microphone to your `AVCaptureSession`
Capturing Live Photos

Check `AVCapturePhotoOutput.isLivePhotoCaptureSupported`  
Live Photo capture is only supported with the `AVCaptureSessionPresetPhoto`  
Opt in using `AVCapturePhotoOutput.isLivePhotoCaptureEnabled = true`  
Add an `AVCaptureDeviceInput` for the microphone to your `AVCaptureSession`  
Presence of `AVCaptureMovieFileOutput` disables Live Photo capture
/** Live Photo Capture Example **/

```swift
func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
        ""/(NSTemporaryDirectory())/_uniqueAppSignature)_\(settings.uniqueID)"

    // Optional Movie-level Metadata
    var metadataItems = [AVMetadataItem]()

    let authorMetadataItem = AVMutableMetadataItem()
    authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
    authorMetadataItem.key = AVMetadataCommonKeyAuthor
    authorMetadataItem.value = "Brad Ford"
    metadataItems.append(authorMetadataItem)

    settings.livePhotoMovieMetadata = metadataItems

    photoOutput.capturePhoto(with: settings, delegate: self)
}
```
// Live Photo Capture Example

func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
        "/(NSTemporaryDirectory())/(self._uniqueAppSignature)_\(settings.uniqueID)")

    // Optional Movie-level Metadata
    var metadataItems = [AVMetadataItem]()

    let authorMetadataItem = AVMutableMetadataItem()
    authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
    authorMetadataItem.key = AVMetadataCommonKeyAuthor
    authorMetadataItem.value = "Brad Ford"
    metadataItems.append(authorMetadataItem)

    settings.livePhotoMovieMetadata = metadataItems

    photoOutput.capturePhoto(with: settings, delegate: self)
}
func takeLivePhoto()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
        "\(NSTemporaryDirectory())/\(_uniqueAppSignature)_\(settings.uniqueID)"
    )

    var metadataItems = [AVMetadataItem]()

    let authorMetadataItem = AVMutableMetadataItem()
    authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
    authorMetadataItem.key = AVMetadataCommonKeyAuthor
    authorMetadataItem.value = "Brad Ford"
    metadataItems.append(authorMetadataItem)

    settings.livePhotoMovieMetadata = metadataItems

    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Live Photo Capture Example

func takeLivePhoto() {
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true
    settings.livePhotoMovieFileURL = URL(fileURLWithPath:
        "\(NSTemporaryDirectory())/(_uniqueAppSignature)_\((settings.uniqueID)"
    )

    // Optional Movie-level Metadata
    var metadataItems = [AVMetadataItem]()
    let authorMetadataItem = AVMutableMetadataItem()
    authorMetadataItem.keySpace = AVMetadataKeySpaceCommon
    authorMetadataItem.key = AVMetadataCommonKeyAuthor
    authorMetadataItem.value = "Brad Ford"
    metadataItems.append(authorMetadataItem)
    settings.livePhotoMovieMetadata = metadataItems

    photoOutput.capturePhoto(with: settings, delegate: self)
}

Live Photo Delegate Methods

- AVCapturePhotoCaptureDelegate
- AVCaptureResolvedPhotoSettings
  - livePhotoMovieURL = someURL
  - livePhotoMovieDimensions = {1440x1080}
  - willBeginCaptureForResolvedSetting
Live Photo Delegate Methods

- `livePhotoMovieURL = someURL`
- `CMSampleBuffer` with method `didFinishProcessingPhotoSampleBuffer`
- `AVCapturePhotoCaptureDelegate`
Live Photo Delegate Methods

`livePhotoMovieURL = someURL`

`didFinishRecordingLivePhotoMovieForEventualFileAtURL`

`AVCapturePhotoCaptureDelegate`

1.5s
Live Photo Delegate Methods

- `livePhotoMovieURL = someURL`
- `didFinishProcessingLivePhotoToMovieFileAtURL`
- `AVCapturePhotoCaptureDelegate`
Live Photo Delegate Methods

- `livePhotoMovieURL = someURL`
- `AVCapturePhotoCaptureDelegate`
- `didFinishCaptureForResolvedSettings`
    if let photoSampleBuffer = photoSampleBuffer,
        data = AVCapturePhotoOutput.jpegPhotoDataRepresentation(forJPEGSampleBuffer: photoSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer) {
            do {
                try data.write(to: _uniqueJPEGFilePath, options: .atomicWrite)
            } catch {
                // Handle error
            }
    }
}
// Live Photo Capture Delegate Methods

```swift
func capture(_ captureOutput: AVCapturePhotoOutput,
    didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
    previewPhotoSampleBuffer: CMSampleBuffer?,
    resolvedSettings: AVCaptureResolvedPhotoSettings,
    bracketSettings: AVCaptureBracketedStillImageSettings?,
    error: NSError?)
{
    if let photoSampleBuffer = photoSampleBuffer,
        data = AVCapturePhotoOutput.jpegPhotoDataRepresentation(forJPEGSampleBuffer:
        photoSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        do {
        try data.write(to: _uniqueJPEGFilePath, options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
```
/ Live Photo Capture Delegate Methods

func capture(_ captureOutput: AVCapturePhotoOutput,
didFinishProcessingPhotoSampleBuffer photoSampleBuffer: CMSampleBuffer?,
previewPhotoSampleBuffer: CMSampleBuffer?,
resolvedSettings: AVCaptureResolvedPhotoSettings,
bracketSettings: AVCaptureBracketedStillImageSettings?,
error: NSError?) {

if let photoSampleBuffer = photoSampleBuffer,
data = AVCapturePhotoOutput.jpegPhotoDataRepresentation(forJPEGSampleBuffer: 
photoSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer) {

do {
    try data.write(to: _uniqueJPEGFilePath, options: .atomicWrite)
}
catch {
    // Handle error
}
}
}
AVCapturePhotoCaptureDelegate Tip
Some types of photo captures deliver multiple assets
Some types of photo captures deliver multiple assets

Instantiate a new AVCapturePhotoCaptureDelegate object for each photo request
Some types of photo captures deliver multiple assets
Instantiate a new AVCapturePhotoCaptureDelegate object for each photo request
Aggregate assets in the delegate object
AVCapturePhotoCaptureDelegate Tip

Some types of photo captures deliver multiple assets

Instantiate a new AVCapturePhotoCaptureDelegate object for each photo request

Aggregate assets in the delegate object

Dispose of the object after didFinishCaptureForResolvedSettings:
Demo
Live Photo Capture and Editing
AVCam and LivePhotoEditor
Live Photo Capture in AVCam
Live Photo Capture in AVCam

Separate video recording and photo modes
Live Photo Capture in AVCam

Separate video recording and photo modes
Shows proper “Live” badging technique
Live Photo Capture in AVCam

Separate video recording and photo modes
Shows proper “Live” badging technique
Saves assets to Photo Library
Live Photo Capture in AVCam

Separate video recording and photo modes
Shows proper “Live” badging technique
Saves assets to Photo Library
Sample code available now!
<table>
<thead>
<tr>
<th>Session 505</th>
<th>Live Photo Editing and RAW Processing with Core Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nob Hill</td>
</tr>
<tr>
<td></td>
<td>Thursday 11:00AM</td>
</tr>
</tbody>
</table>
Live Photo Capture Suspension
Live Photo Capture Suspension
Live Photo Capture Suspension

P1

Obnoxious Foghorn
Live Photo Capture Suspension

Obnoxious Foghorn
Live Photo Capture Suspension

Time

P1

Obnoxious Foghorn

P2
Live Photo Capture Suspension

output.isLivePhotoCaptureSuspended = true
Live Photo Capture Suspension

output.isLivePhotoCaptureSuspended = true
Live Photo Capture Suspension

output.isLivePhotoCaptureSuspended = true

output.isLivePhotoCaptureSuspended = false
Live Photo Capture Suspension

output.isLivePhotoCaptureSuspended = true

Obnoxious Foghorn

output.isLivePhotoCaptureSuspended = false
Live Photo Capture Support
Live Photo Capture Support

- iPhone 6s
- iPhone 6s Plus
- iPhone SE
- 9.7-inch iPad Pro
RAW Photo Capture

with a dash of DNG on the side
What is RAW?
What is RAW?
What is RAW?
What is RAW?

Light From Scene

Color Filter Array
What is RAW?

Light From Scene

Color Filter Array

Sensor Array
What is Stored in a RAW file?
What is Stored in a RAW file?

Intensity of red, green, or blue light hitting the sensor
What is Stored in a RAW file?

Intensity of red, green, or blue light hitting the sensor
Bayer pattern as metadata
What is Stored in a RAW file?

Intensity of red, green, or blue light hitting the sensor
Bayer pattern as metadata
A lot of other metadata
What Do RAW Converters Do?
What Do RAW Converters Do?

- Demosaic Bayer filter pattern
- Apply white balance
- Interpret Colorimetric Info
- Correct Gamma
- Reduce Noise
- Sharpen
Why RAW?
Why RAW?

Bake-time flexibility
Why RAW?

Bake-time flexibility
No compression
Why RAW?

Bake-time flexibility
No compression
More bits
Why RAW?

Bake-time flexibility
No compression
More bits
Lots of headroom for editing
Why RAW?

- Bake-time flexibility
- No compression
- More bits
- Lots of headroom for editing
- Greater artistic freedom to interpret the image data in post
Why JPEG?
Why JPEG?

Lovingly baked by Apple
Why JPEG?

Lovingly baked by Apple
Much faster rendering
Why JPEG?

Lovingly baked by Apple
Much faster rendering
Multiple image fusion (e.g. for stabilization)
Why JPEG?

Lovingly baked by Apple
Much faster rendering
Multiple image fusion (e.g. for stabilization)
Smaller file size
/* Bayer 14-bit Little-Endian, packed in 16-bits, ordered G R G R...
 alternation with B G B G... */
kCVPixelFormatType_14Bayer_GRBG = OSType("grb4"),

/* Bayer 14-bit Little-Endian, packed in 16-bits, ordered R G R G...
 alternation with G B G B... */
kCVPixelFormatType_14Bayer_RGGB = OSType("rgg4"),

/* Bayer 14-bit Little-Endian, packed in 16-bits, ordered B G B G...
 alternation with G R G R... */
kCVPixelFormatType_14Bayer_BGGR = OSType("bgg4"),

/* Bayer 14-bit Little-Endian, packed in 16-bits, ordered G B G B...
 alternation with R G R G... */
kCVPixelFormatType_14Bayer_GBRG = OSType("gbr4"),
Capturing RAW with AVCapturePhotoOutput
Capturing RAW with AVCapturePhotoOutput

RAW is only supported when using a photo format ( AVCaptureSessionPresetPhoto )
Capturing RAW with AVCapturePhotoOutput

RAW is only supported when using a photo format (AVCaptureSessionPresetPhoto)
Rear camera only
Capturing RAW with AVCapturePhotoOutput

RAW is only supported when using a photo format ( AVCaptureSessionPresetPhoto )
Rear camera only
RAW bracketed captures are supported
// Capturing RAW Photos

func takeRawPhoto() {
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}

// Capturing RAW Photos

func takeRawPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat)

    // RAW photo settings have autoStillImageStabilizationEnabled set to NO
    // highResolutionPhotoEnabled is also NO as it’s meaningless in RAW

    photoOutput.capturePhoto(with: settings, delegate: self)
}

// RAW related AVCapturePhotoCaptureDelegate methods

func capture(_: AVCapturePhotoOutput,
didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
previewPhotoSampleBuffer: CMSampleBuffer?,
resolvedSettings: AVCaptureResolvedPhotoSettings,
bracketSettings: AVCaptureBracketedStillImageSettings?,
error: NSError?)
{
    // Handle the RAW sample buffer
}
func capture(_ captureOutput: AVCapturePhotoOutput,
    didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
    previewPhotoSampleBuffer: CMSampleBuffer?,
    resolvedSettings: AVCaptureResolvedPhotoSettings,
    bracketSettings: AVCaptureBracketedStillImageSettings?,
    error: NSError?)
{
    // Handle the RAW sample buffer
}
func capture(_ captureOutput: AVCapturePhotoOutput, 
    didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?, 
    previewPhotoSampleBuffer: CMSampleBuffer?, 
    resolvedSettings: AVCaptureResolvedPhotoSettings, 
    bracketSettings: AVCaptureBracketedStillImageSettings?, 
    error: NSError?)
{
    // Handle the RAW sample buffer
}
RAW + Processed Image Support
RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format
RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

The processed image is delivered to didFinishProcessingPhotoSampleBuffer:
RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format

The processed image is delivered to didFinishProcessingPhotoSampleBuffer:

The RAW image is delivered to didFinishProcessingRAWPhotoSampleBuffer:
RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format.
The processed image is delivered to `didFinishProcessingPhotoSampleBuffer`.
The RAW image is delivered to `didFinishProcessingRAWPhotoSampleBuffer`.
RAW + processed brackets are supported.
RAW + Processed Image Support

Processed image may be JPEG, or an uncompressed format.

The processed image is delivered to `didFinishProcessingPhotoSampleBuffer`:

The RAW image is delivered to `didFinishProcessingRAWPhotoSampleBuffer`:

RAW + processed brackets are supported.

RAW + still image stabilization processed image is NOT supported.
func takeRAWPlusJPEGPhoto() {
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let processedFormat = photoOutput.availablePhotoCodecTypes.first!
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat, processedFormat:
        [AVVideoCodecKey : processedFormat])

    // highResolutionPhotoEnabled YES or NO applies only to the processed photo

    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Capturing RAW Photos

func takeRAWPlusJPEGPhoto()
{

    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let processedFormat = photoOutput.availablePhotoCodecTypes.first!
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat, processedFormat:
                                            [AVVideoCodecKey : processedFormat])

    // highResolutionPhotoEnabled YES or NO applies only to the processed photo

    photoOutput.capturePhoto(with: settings, delegate: self)
}
func takeRAWPlusJPEGPhoto()
{
    let rawFormat = photoOutput.availableRawPhotoPixelFormatTypes.first!.uint32Value
    let processedFormat = photoOutput.availablePhotoCodecTypes.first!
    let settings = AVCapturePhotoSettings(rawPixelFormatType: rawFormat, processedFormat: [AVVideoCodecKey : processedFormat])

    photoOutput.capturePhoto(with: settings, delegate: self)
}

// highResolutionPhotoEnabled YES or NO applies only to the processed photo
Storing RAW Buffers
// Writing RAW to DNG

func capture(_ captureOutput: AVCapturePhotoOutput,
        didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
        previewPhotoSampleBuffer: CMSampleBuffer?,
        resolvedSettings: AVCaptureResolvedPhotoSettings,
        bracketSettings: AVCaptureBracketedStillImageSettings?,
        error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer: rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
/ Writing RAW to DNG

func capture(_ captureOutput: AVCapturePhotoOutput,
didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
previewPhotoSampleBuffer: CMSampleBuffer?,
resolvedSettings: AVCaptureResolvedPhotoSettings,
bracketSettings: AVCaptureBracketedStillImageSettings?,
error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
            rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        }
        catch {
            // Handle error
        }
    }
}
Demo

RAW Capture and Editing

AVCamManual and RawExpose
More on RAW Processing in Session 505

Live Photo Editing and RAW Processing with Core Image

Nob Hill

Thursday 11:00AM
RAW Photo Capture Support
RAW Photo Capture Support

- iPhone 6s
- iPhone 6s Plus
- iPhone SE
- 9.7-inch iPad Pro
Capturing Preview Images

a.k.a. Thumbnails
Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image
Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image
Preview image is uncompressed
Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image
Preview image is uncompressed
You can specify the preview image dimensions
Preview Images

didFinishProcessing{Raw}PhotoSampleBuffer: callback can deliver a preview image
Preview image is uncompressed
You can specify the preview image dimensions
Photo output can select a size for you
// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
    let previewFormat = [
        kCVPixelBufferPixelFormatTypeKey as String : previewPixelType as! AnyObject,
        kCVPixelBufferWidthKey as String : 160,
        kCVPixelBufferHeightKey as String : 160
    ]
    settings.previewPhotoFormat = previewFormat

    photoOutput.capturePhoto(with: settings, delegate: self)
}

// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
    let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String : previewPixelType as! AnyObject,
                         kCVPixelBufferWidthKey as String : 160,
                         kCVPixelBufferHeightKey as String : 160]

    settings.previewPhotoFormat = previewFormat

    photoOutput.capturePhoto(with: settings, delegate: self)
}
// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
    let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String : previewPixelType as! AnyObject,
                         kCVPixelBufferWidthKey as String : 160,
                         kCVPixelBufferHeightKey as String : 160]
    settings.previewPhotoFormat = previewFormat

    photoOutput.capturePhoto(with: settings, delegate: self)
}

// Capturing a processed image with a preview (thumbnail)

func takePhotoWithPreviewImage()
{
    let settings = AVCapturePhotoSettings()
    settings.isHighResolutionPhotoEnabled = true

    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!.uint32Value
    let previewFormat = [
        kCVPixelBufferPixelFormatTypeKey as String : previewPixelType as! AnyObject,
        kCVPixelBufferWidthKey as String : 160,
        kCVPixelBufferHeightKey as String : 160
    ]
    settings.previewPhotoFormat = previewFormat

    photoOutput.capturePhoto(with: settings, delegate: self)
}
Preview Image Retrieval

JPEG + previewFormat = 420f, 160x160

willBeginCaptureForResolvedSettings

AVCapturePhotoCaptureDelegate
Preview Image Retrieval

- JPEG + previewFormat = 420f, 160x160
- AVCapturePhotoCaptureDelegate
  - willBeginCaptureForResolvedSettings
  - AVCaptureResolvedPhotoSettings
    - photoDimensions = 4032x3024
    - previewPhotoDimensions = 160x120
Preview Image Retrieval

JPEG + previewFormat = 420f, 160x160

AVCapturePhotoCaptureDelegate

willBeginCaptureForResolvedSettings

didFinishProcessingPhotoSampleBuffer
previewPhotoSampleBuffer

CMSampleBuffer

CMSampleBuffer
// Use preview image as a thumbnail

func capture(_ captureOutput: AVCapturePhotoOutput,
didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
previewPhotoSampleBuffer: CMSampleBuffer?,
resolvedSettings: AVCaptureResolvedPhotoSettings,
bracketSettings: AVCaptureBracketedStillImageSettings?,
error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer: rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        } catch {
            // Handle error
        }
    }
}
// Use preview image as a thumbnail

func capture(_ captureOutput: AVCapturePhotoOutput,
    didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
    previewPhotoSampleBuffer: CMSampleBuffer?,
    resolvedSettings: AVCaptureResolvedPhotoSettings,
    bracketSettings: AVCaptureBracketedStillImageSettings?,
    error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer: rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        } catch {
            // Handle error
        }
    }
}
// Use preview image as a thumbnail

func capture(_ captureOutput: AVCapturePhotoOutput,
            didFinishProcessingRawPhotoSampleBuffer rawSampleBuffer: CMSampleBuffer?,
            previewPhotoSampleBuffer: CMSampleBuffer?,
            resolvedSettings: AVCaptureResolvedPhotoSettings,
            bracketSettings: AVCaptureBracketedStillImageSettings?,
            error: NSError?)
{
    if let rawSampleBuffer = rawSampleBuffer,
        data = AVCapturePhotoOutput.dngPhotoDataRepresentation(forRawSampleBuffer:
                        rawSampleBuffer, previewPhotoSampleBuffer: previewPhotoSampleBuffer)
    {
        let filePath = (_uniqueFilePath as NSString).appendingPathExtension(".dng")!
        do {
            try data.write(to: URL(fileURLWithPath: filePath), options: .atomicWrite)
        } catch {
            // Handle error
        }
    }
}

Everywhere!
Wide Color Capture
Brief Overview
Brief Overview

iPad Pro 9.7 True Tone Display
Brief Overview

iPad Pro 9.7 True Tone Display
Color management in iOS 9.3!
Intro to Wide Color

sRGB color space
Intro to Wide Color

sRGB color space

Based on the ITU-R BT.709 standard
Intro to Wide Color

sRGB color space

Based on the ITU-R BT.709 standard

Gamma $\approx 2.2$
Intro to Wide Color
sRGB color space

Based on the ITU-R BT.709 standard

Gamma $\approx 2.2$

White point of 6500 degrees K (D65)
Intro to Wide Color
Display P3 color space
Intro to Wide Color

Display P3 color space

Based on the DCI-P3 standard
Intro to Wide Color

Display P3 color space

Based on the DCI-P3 standard

Gamma ≈ 2.2
Intro to Wide Color

Display P3 color space

Based on the DCI-P3 standard

Gamma $\approx 2.2$

White point of 6500 degrees K (D65)
Profile Information:

- Name: Display P3
- Path: /System/Library/ColorSync/Profiles/Display P3.icc
- Class: Display
- Space: RGB
- PCS: XYZ
- Version: 4.0.0
- Created: Wednesday, October 14, 2015 at 1:08:57 PM Pacific Daylight Time
- Size: 648 bytes

Lab Plot:
Capturing Display P3 Color on iPad Pro 9.7

Diagram:

- AVCaptureDevice (Camera)
  - AVCaptureDeviceInput
    - AVCaptureConnection
      - AVCaptureSession
        - AVCapturePhotoOutput
        - AVCaptureMovieFileOutput
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

AVCaptureDeviceInput

AVCaptureConnection

AVCaptureConnection

AVCaptureSession

AVCapturePhotoOutput

AVCaptureMovieFileOutput
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

**formats**

- 1920x1080 (1080p) @ 30 FPS, 420v
- 1920x1080 (1080p) @ 30 FPS, 420f
- 4032x3024 (12 MP) @ 30 FPS, 420v
- 4032x3024 (12 MP) @ 30 FPS, 420f
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

<table>
<thead>
<tr>
<th>formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420f</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420f</td>
</tr>
</tbody>
</table>

```swift
public var supportedColorSpaces: [NSNumber]! { get }

public enum AVCaptureColorSpace : Int {
    case sRGB
    case P3_D65
}
```
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

<table>
<thead>
<tr>
<th>formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420f</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420f</td>
</tr>
</tbody>
</table>
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

<table>
<thead>
<tr>
<th>formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420f</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420f</td>
</tr>
</tbody>
</table>

420v supports sRGB only
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

<table>
<thead>
<tr>
<th>formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420f</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420f</td>
</tr>
</tbody>
</table>

420f supports sRGB and Display P3
Capturing Display P3 Color on iPad Pro 9.7

**AVCaptureDevice (Camera)**

<table>
<thead>
<tr>
<th>Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420f</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420v</td>
</tr>
<tr>
<td><strong>4032x3024 (12 MP) @ 30 FPS, 420f</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>activeFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4032x3024 (12 MP) @ 30 FPS, 420f</strong></td>
</tr>
</tbody>
</table>
Capturing Display P3 Color on iPad Pro 9.7

AVCaptureDevice (Camera)

<table>
<thead>
<tr>
<th>formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>1920x1080 (1080p) @ 30 FPS, 420f</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420v</td>
</tr>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>activeFormat</th>
</tr>
</thead>
<tbody>
<tr>
<td>4032x3024 (12 MP) @ 30 FPS, 420f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>activeColorSpace</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3 D65</td>
</tr>
</tbody>
</table>
Automatic Color Space Selection in AVCaptureSession
Automatic Color Space Selection in AVCaptureSession

```objective-c
public var automaticallyConfiguresCaptureDeviceForWideColor: Bool
```
Automatic Color Space Selection in AVCaptureSession

```swift
public var automaticallyConfiguresCaptureDeviceForWideColor: Bool
```

Session sets your device’s `activeColorSpace` to `P3_D65` depending on your config.
Automatic Color Space Selection in AVCaptureSession

`public var automaticallyConfiguresCaptureDeviceForWideColor: Bool`

Session sets your device’s `activeColorSpace` to `P3_D65` depending on your config.

`AVCapturePhotoOutput` must be present in the session.
Automatic Color Space Selection in AVCaptureSession

Session contains photo output and...  |  Session configures device for...
---|---
AVCaptureVideoPreviewLayer  |  Display P3

AVCaptureMovieFileOutput  |  sRGB

AVCaptureVideoDataOutput  |  Display P3
*only if sessionPreset is Photo*
Forcing Display P3 Color
Forcing Display P3 Color

Set `session.automaticalyConfiguresDeviceForWideColor = false`
Forcing Display P3 Color

Set `session.automaticallyConfiguresDeviceForWideColor = false`

Set `device.activeFormat` to a format that supports wide color
Forcing Display P3 Color

- Set `session.automaticalyConfiguresDeviceForWideColor = false`
- Set `device.activeFormat` to a format that supports wide color
- Set `device.activeColorSpace = P3_D65`
The Danger of Forcing Display P3 Color
The Danger of Forcing Display P3 Color

Display P3 is not well-supported in video
The Danger of Forcing Display P3 Color

Display P3 is not well-supported in video. VideoDataOutput callback should be color-aware and propagate color tags.
The Danger of Forcing Display P3 Color

Display P3 is not well-supported in video
VideoDataOutput callback should be color-aware and propagate color tags
Display P3 movies from MovieFileOutput may render incorrectly on other platforms
Sharing Wide Color Photos
Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile
Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile
iCloud Photo Library is color-aware
Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile

iCloud Photo Library is color-aware

Display P3 JPEGs via Messages and Mail are converted to Apple Wide Color Sharing Profile
Sharing Wide Color Photos

Wide color JPEG files use a Display P3 Color Profile

iCloud Photo Library is color-aware

Display P3 JPEGs via Messages and Mail are converted to Apple Wide Color Sharing Profile

Live Photo Editing and RAW Processing with Core Image

Nob Hill
Thursday 11:00AM

Working With Wide Color

Mission
Thursday 1:40PM
AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7
AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

‘420f’, ‘BGRA’, and ‘jpeg’
AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

‘420f’, ‘BGRA’, and ‘jpeg’

Live Photos (still image and movie)
AVCapturePhotoOutput Wide Color Support on iPad Pro 9.7

‘420f’, ‘BGRA’, and ‘jpeg’

Live Photos (still image and movie)

Bracketed capture
Wide Color and RAW
Wide Color and RAW

Apple camera RAW images are inherently wide color
Wide Color and RAW

Apple camera RAW images are inherently wide color
Camera sensor primaries are rich enough to extract Display P3 or sRGB
More on Wide Color

<table>
<thead>
<tr>
<th>Live Photo Editing and RAW Processing with Core Image</th>
<th>Nob Hill</th>
<th>Thursday 11:00AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working With Wide Color</td>
<td>Mission</td>
<td>Thursday 1:40PM</td>
</tr>
</tbody>
</table>
Summary

Use `AVCapturePhotoOutput` for improved usability
Summary

Use **AVCapturePhotoOutput** for improved usability

- Live Photos
Summary

Use **AVCapturePhotoOutput** for improved usability

- Live Photos
- RAW, RAW + JPEG and DNG
Use `AVCapturePhotoOutput` for improved usability

- Live Photos
- RAW, RAW + JPEG and DNG
- Preview Images
Use `AVCapturePhotoOutput` for improved usability

- Live Photos
- RAW, RAW + JPEG and DNG
- Preview Images
- Wide Color Photos
AVCapturePhotoOutput—Beyond the Basics

Session 511: A chalk talk addendum

Scene monitoring in **AVCapturePhotoOutput**
Scene monitoring in \texttt{AVCapturePhotoOutput}

Resource preparation and reclamation in \texttt{AVCapturePhotoOutput}
AVCapturePhotoOutput—Beyond the Basics

Session 511: A chalk talk addendum

Scene monitoring in AVCapturePhotoOutput

Resource preparation and reclamation in AVCapturePhotoOutput

Changes to camera privacy policy in iOS 10
More Information

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVCapturePhotoOutput—Beyond the Basics</td>
<td>Video</td>
<td>At your leisure</td>
</tr>
<tr>
<td>Live Photo Editing and RAW Processing with Core Image</td>
<td>Nob Hill</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Working With Wide Color</td>
<td>Mission</td>
<td>Thursday 1:40PM</td>
</tr>
<tr>
<td>Labs</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Photo Capture Lab</td>
<td>Graphics, Games, and Media Lab D</td>
<td>Tuesday 1:00PM</td>
</tr>
<tr>
<td>PhotoKit Lab</td>
<td>Graphics, Games, and Media Lab D</td>
<td>Tuesday 4:00PM</td>
</tr>
<tr>
<td>Color Lab</td>
<td>Frameworks Lab A</td>
<td>Wednesday 1:00PM</td>
</tr>
<tr>
<td>Photo Capture Lab</td>
<td>Graphics, Games, and Media Lab C</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Live Photo &amp; Core Image Lab</td>
<td>Graphics, Games, and Media Lab C</td>
<td>Thursday 1:30PM</td>
</tr>
<tr>
<td>Live Photo &amp; Core Image Lab</td>
<td>Graphics, Games, and Media Lab D</td>
<td>Friday 9:00AM</td>
</tr>
<tr>
<td>Color Lab</td>
<td>Graphics, Games, and Media Lab C</td>
<td>Friday 4:00PM</td>
</tr>
</tbody>
</table>