What’s New in Camera Capture

A plethora of API enhancements

Session 610

Brad Ford
Core Media Engineering

What You Will Learn

- Greater transparency for users
- New AV Foundation capture features in iOS 7
- Sample code update

What You Will Not Learn

- CoreMedia basics
- AV Foundation basics
- Review prior WWDC camera sessions in the WWDC app

What's New in Camera Capture

- Greater transparency for users
- New AV Foundation capture features in iOS 7
- Sample code update

Greater Transparency for Users

iOS 6 Camera Ecosystem (Review)

- Photos and videos are personal, sensitive data
- iOS 6 introduced a prompt for access to AssetsLibrary

iOS 6 Camera Ecosystem (Review)

- Photos and videos are personal, sensitive data
- iOS 6 introduced a prompt for access to AssetsLibrary

iOS 6 Camera Ecosystem (Review)

- Photos and videos are personal, sensitive data
- iOS 6 introduced a prompt for access to AssetsLibrary

- Reminder: Handle errors from ALAssetsLibrary methods

Greater Transparency for Users

Greater Transparency for Users

- iOS devices have no “recording in progress” lights

Greater Transparency for Users

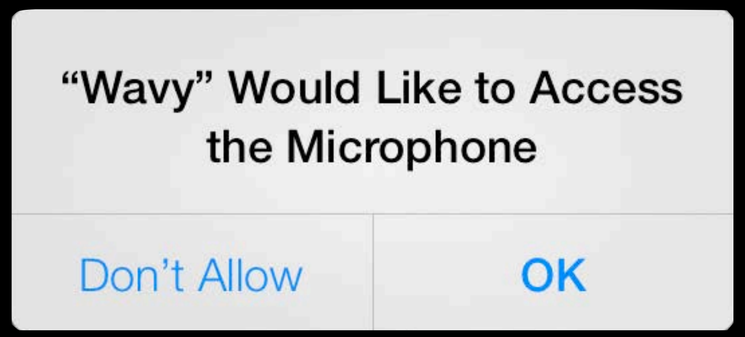
- iOS devices have no “recording in progress” lights
- AV Foundation APIs present no “recording in progress” UI

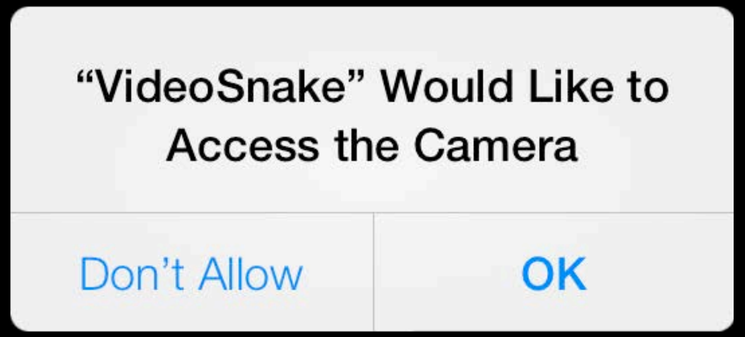
Greater Transparency for Users

- iOS devices have no “recording in progress” lights
- AV Foundation APIs present no “recording in progress” UI
- In some regions, law requires user consent to use the camera or mic

Greater Transparency for Users

- iOS devices have no “recording in progress” lights
- AV Foundation APIs present no “recording in progress” UI
- In some regions, law requires user consent to use the camera or mic





Greater Transparency for Users

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

```
AVCaptureDeviceInput *di =
      [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
```

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

```
AVCaptureDeviceInput *di =
      [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
```

- AVCaptureDeviceInput creation succeeds

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

```
AVCaptureDeviceInput *di =
      [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
```

- AVCaptureDeviceInput creation succeeds
  - Mic produces silence until user grants access

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

```
AVCaptureDeviceInput *di =
      [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
```

- AVCaptureDeviceInput creation succeeds
  - Mic produces silence until user grants access
  - Camera produces black frames until user grants access

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

```
AVCaptureDeviceInput *di =
      [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
```

- AVCaptureDeviceInput creation succeeds
  - Mic produces silence until user grants access
  - Camera produces black frames until user grants access
- Subsequent DeviceInput creation fails if user denied access

Greater Transparency for Users

- First-time AVCaptureDeviceInput creation invokes dialog

```
AVCaptureDeviceInput *di =
      [AVCaptureDeviceInput deviceInputWithDevice:device error:&error];
```

- AVCaptureDeviceInput creation succeeds
  - Mic produces silence until user grants access
  - Camera produces black frames until user grants access
- Subsequent DeviceInput creation fails if user denied access
- Check for `AVErrorApplicationIsNotAuthorizedToUseDevice`

What's New in Camera Capture

- Greater transparency for users
- New AV Foundation capture features in iOS 7
- Sample code update

iOS 7 AV Foundation Capture APIs

A veritable smörgåsbord of enhancements

New in iOS 7

- 60 FPS support
- Video zoom
- Machine Readable Code detection
- Focus enhancements
- Integration with application AudioSession

New in iOS 7

- 60 FPS support
- Video zoom
- Machine Readable Code detection
- Focus enhancements
- Integration with application AudioSession

60 FPS Support

NEW

60 FPS Support

- Full iOS ecosystem support for high frame-rate content

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback
  - Enhanced audio processing support for slow and fast playback

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback
  - Enhanced audio processing support for slow and fast playback
- Editing

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback
  - Enhanced audio processing support for slow and fast playback
- Editing
  - Full support for scaled edits in mutable compositions

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback
  - Enhanced audio processing support for slow and fast playback
- Editing
  - Full support for scaled edits in mutable compositions
- Export

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback
  - Enhanced audio processing support for slow and fast playback
- Editing
  - Full support for scaled edits in mutable compositions
- Export
  - Variable frame rate preservation (slow and fast motion)

60 FPS Support

- Full iOS ecosystem support for high frame-rate content
- Capture
  - 720p60 support with video stabilization and droppable P-Frames
- Playback
  - Enhanced audio processing support for slow and fast playback
- Editing
  - Full support for scaled edits in mutable compositions
- Export
  - Variable frame rate preservation (slow and fast motion)
  - Frame rate conversion to 30 FPS output

Demo

SloPoke

60 FPS Support

Playback

60 FPS Support

Playback

- AVPlayer slows down content automatically using `[player setRate:]`

60 FPS Support

Playback

- AVPlayer slows down content automatically using `[player setRate:]`
- New property `[playerItem setAudioTimePitchAlgorithm:]`

Time Pitch Algorithm	Quality	Snaps to Specific Rates	Rate Range
`LowQualityZeroLatency`	Low, suitable for FF / Rew or low quality voice	YES	0.5 - 2 x
`TimeDomain`	Modest, less expensive, suitable for voice	NO	0.5 - 2 x
`Spectral`	Highest, most expensive, preserves pitch	NO	1/32 - 32 x
`Varispeed`	High, no pitch correction	NO	1/32 - 32 x

60 FPS Support

Editing

60 FPS Support

Editing

- Use AVMutableComposition to build up temporal edits

60 FPS Support

Editing

- Use AVMutableComposition to build up temporal edits

```
mutableComposition = [AVMutableComposition composition];
```

60 FPS Support

Editing

- Use AVMutableComposition to build up temporal edits

```
mutableComposition = [AVMutableComposition composition];
[mutableComposition insertTimeRange:ofAsset:atTime:error:];
```

60 FPS Support

Editing

- Use AVMutableComposition to build up temporal edits

```
mutableComposition = [AVMutableComposition composition];
[mutableComposition insertTimeRange:ofAsset:atTime:error:];
[mutableComposition scaleTimeRange:toDuration:];
```

60 FPS Support

Editing

- Use AVMutableComposition to build up temporal edits

```
mutableComposition = [AVMutableComposition composition];
[mutableComposition insertTimeRange:ofAsset:atTime:error:];
[mutableComposition scaleTimeRange:toDuration:];
```

- See SloPoke Sample code for reference

Advanced Editing with AV Foundation	Marina Thursday 9:00 AM

60 FPS Support

Export

60 FPS Support

Export

- Use AVAssetExportSession

60 FPS Support

Export

- Use AVAssetExportSession
  - “Flatten” a mutable composition into a new movie

60 FPS Support

Export

- Use AVAssetExportSession
  - “Flatten” a mutable composition into a new movie
  - Use `AVAssetExportPresetPassthrough` to avoid re-encoding

60 FPS Support

Export

- Use AVAssetExportSession
  - "Flatten" a mutable composition into a new movie
  - Use `AVAssetExportPresetPassthrough` to avoid re-encoding
  - Constant frame rate export

60 FPS Support

Export

- Use AVAssetExportSession
  - "Flatten" a mutable composition into a new movie
  - Use `AVAssetExportPresetPassthrough` to avoid re-encoding
  - Constant frame rate export
    - Set `videoComposition.frameDuration`

60 FPS Support

Export

- Use AVAssetExportSession
  - "Flatten" a mutable composition into a new movie
  - Use `AVAssetExportPresetPassthrough` to avoid re-encoding
  - Constant frame rate export
    - Set `videoComposition.frameDuration`
    - Maximum playback compatibility

60 FPS Support

Export

- Use AVAssetExportSession
  - "Flatten" a mutable composition into a new movie
  - Use `AVAssetExportPresetPassthrough` to avoid re-encoding
  - Constant frame rate export
    - Set `videoComposition.frameDuration`
    - Maximum playback compatibility
  - Use new `[exportSession setAudioTimePitchAlgorithm:]`

60 FPS Support

Export

- Use AVAssetExportSession
  - "Flatten" a mutable composition into a new movie
  - Use `AVAssetExportPresetPassthrough` to avoid re-encoding
  - Constant frame rate export
    - Set `videoComposition.frameDuration`
    - Maximum playback compatibility
  - Use new `[exportSession setAudioTimePitchAlgorithm:]`
- See SloPoke Sample code for reference

60 FPS Support

Recording

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically
- AVAssetWriter capture requires some set-up

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically
- AVAssetWriter capture requires some set-up

```
[assetWriterInput setExpectsMediaDataInRealTime:YES]
```

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically
- AVAssetWriter capture requires some set-up

```
[assetWriterInput setExpectsMediaDataInRealTime:YES]
```

  - Use new AVOutputSettingsAssistant

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically
- AVAssetWriter capture requires some set-up

```
[assetWriterInput setExpectsMediaDataInRealTime:YES]
```

  - Use new AVOutputSettingsAssistant

```
setSourceVideoFormat:(CMFormatDescription *)format;
```

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically
- AVAssetWriter capture requires some set-up

```
[assetWriterInput setExpectsMediaDataInRealTime:YES]
```

  - Use new AVOutputSettingsAssistant

```
setSourceVideoFormat:(CMFormatDescription *)format;
setSourceVideoAverageFrameDuration:(CMTime)duration;
```

60 FPS Support

Recording

- AVCaptureMovieFileOutput capture works automatically
- AVAssetWriter capture requires some set-up

```
[assetWriterInput setExpectsMediaDataInRealTime:YES]
```

  - Use new AVOutputSettingsAssistant

```
setSourceVideoFormat:(CMFormatDescription *)format;
setSourceVideoAverageFrameDuration:(CMTime)duration;
- (NSDictionary *)videoSettings;
```

60 FPS Support

Configuring Capture for 60 FPS

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

```
[avCaptureSession setSessionPreset:aPreset];
```

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

```
[avCaptureSession setSessionPreset:aPreset];
```

  - Session configures inputs and outputs for the given preset

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

```
[avCaptureSession setSessionPreset:aPreset];
```

  - Session configures inputs and outputs for the given preset
- iOS 7—A parallel configuration mechanism

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

```
[avCaptureSession setSessionPreset:aPreset];
```

  - Session configures inputs and outputs for the given preset
- iOS 7—A parallel configuration mechanism
  - AVCaptureDevice format inspection

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

```
[avCaptureSession setSessionPreset:aPreset];
```

  - Session configures inputs and outputs for the given preset
- iOS 7—A parallel configuration mechanism
  - AVCaptureDevice format inspection
  - AVCaptureDevice active format selection

60 FPS Support

Configuring Capture for 60 FPS

- iOS 6 and earlier

```
[avCaptureSession setSessionPreset:aPreset];
```

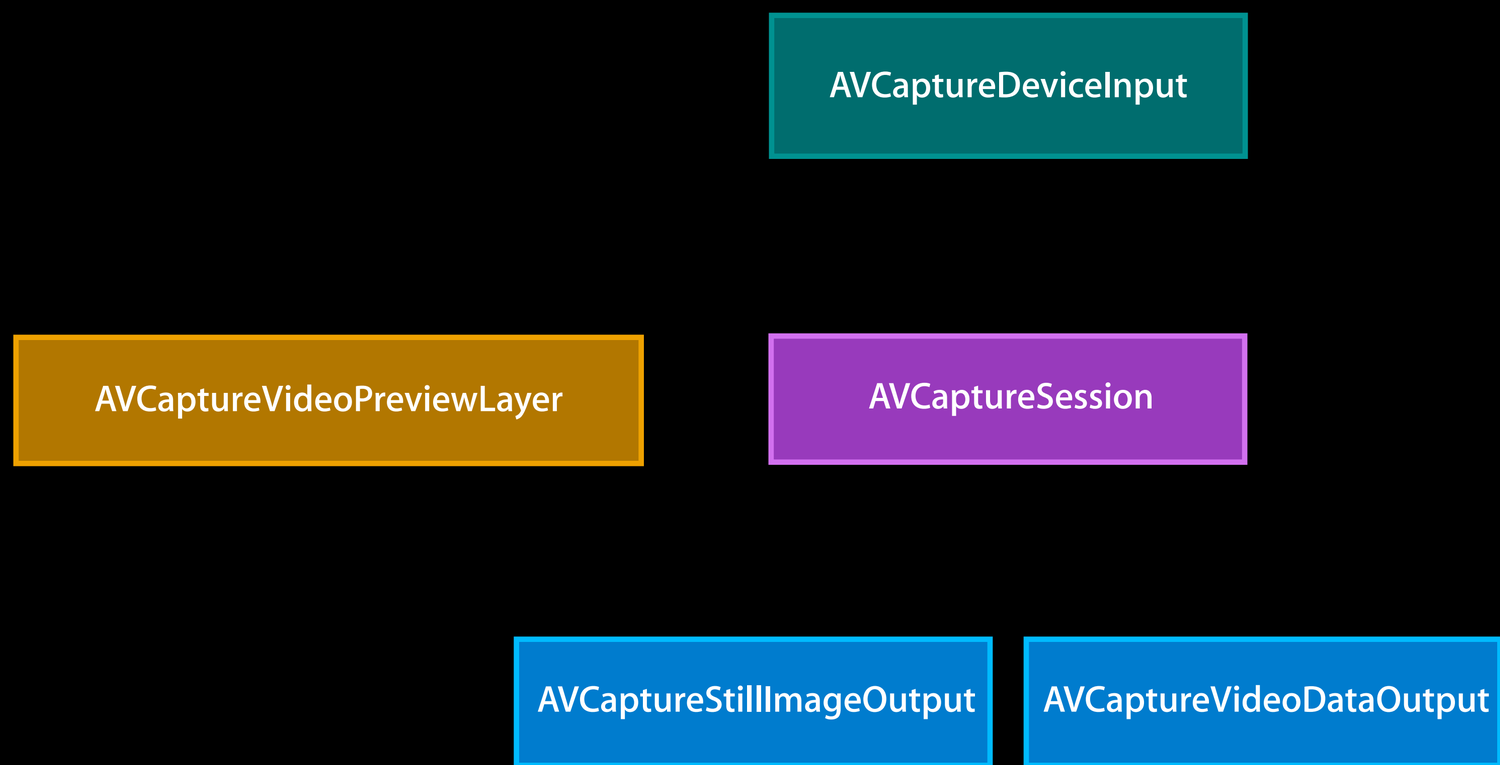
  - Session configures inputs and outputs for the given preset
- iOS 7—A parallel configuration mechanism
  - AVCaptureDevice format inspection
  - AVCaptureDevice active format selection
  - Session no longer configures inputs and outputs automatically

1280x720 @ 60 FPS Capture

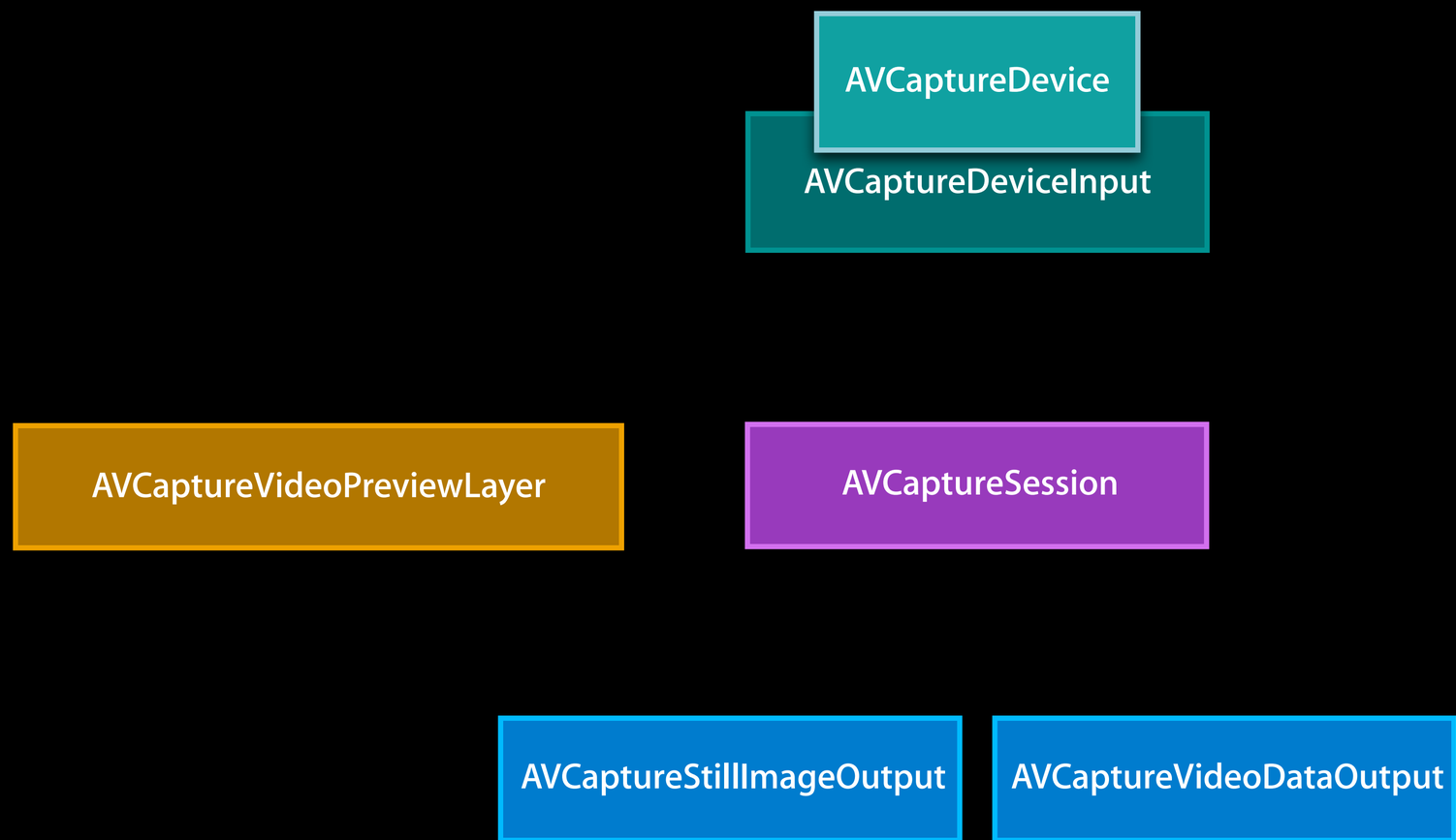
Supported platforms

iPhone 5	✓
iPod touch (5th Generation)	✓
iPad mini	✓

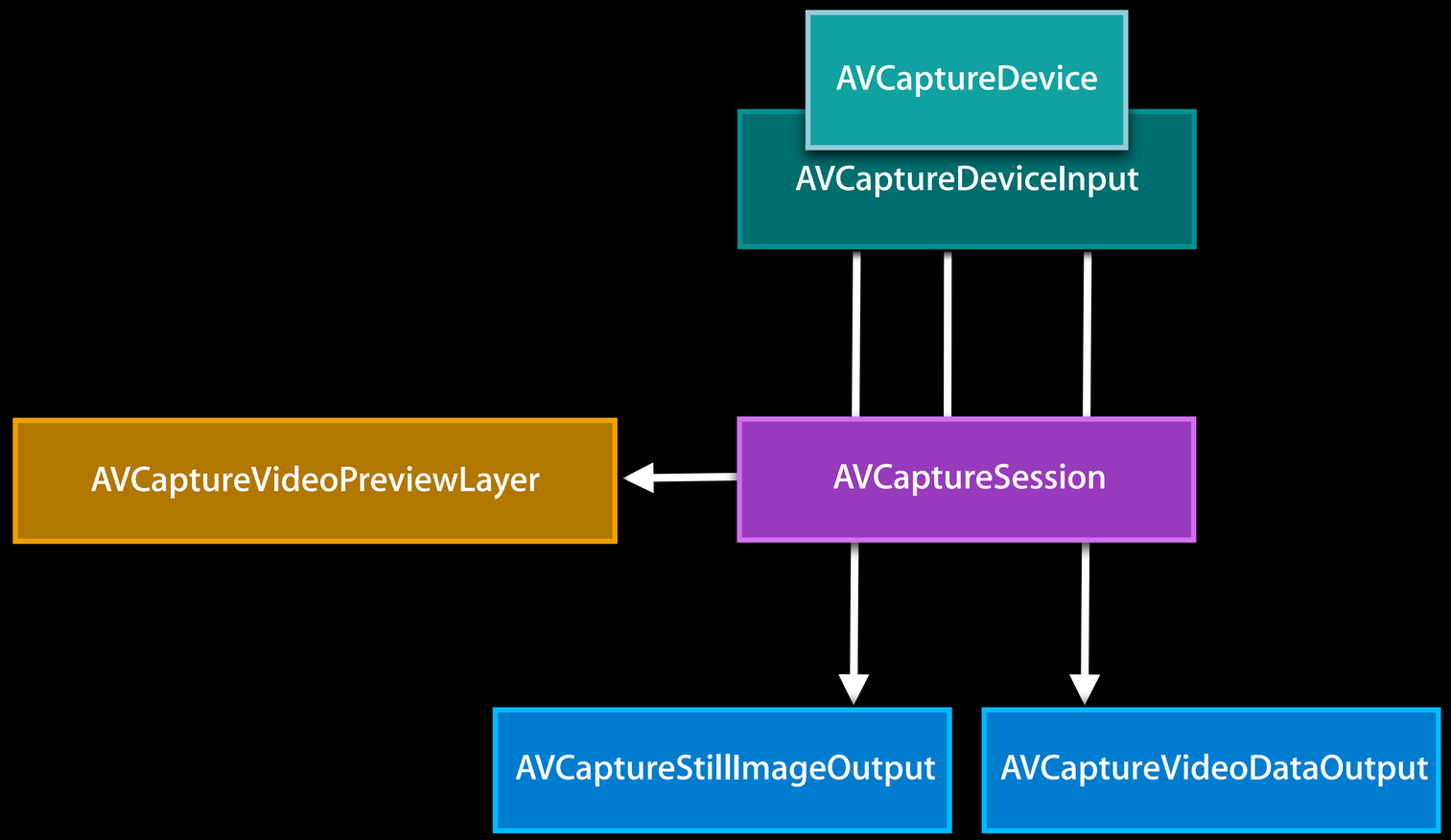
Configuration Using `-setSessionPreset:`



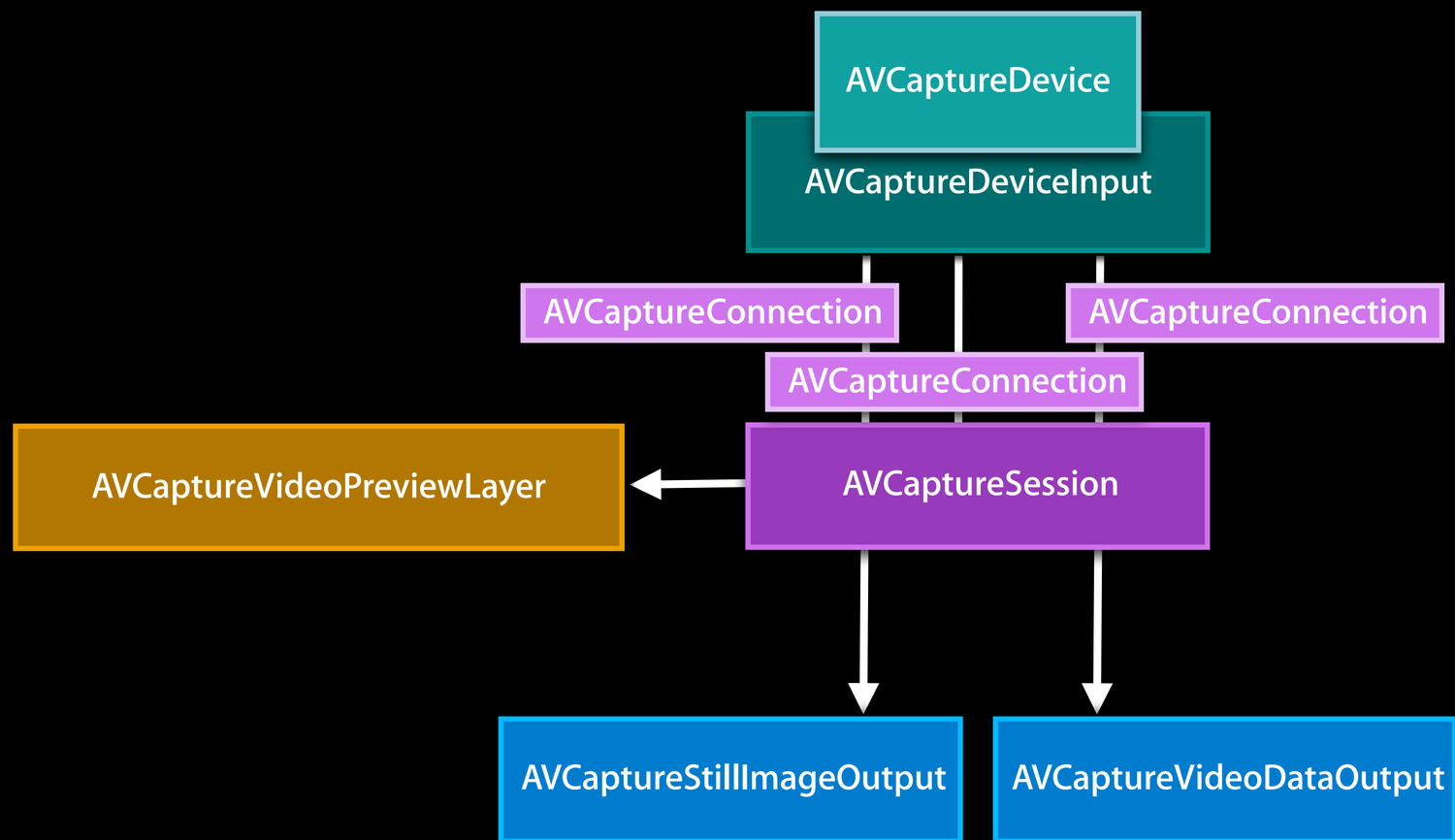
Configuration Using `-setSessionPreset:`



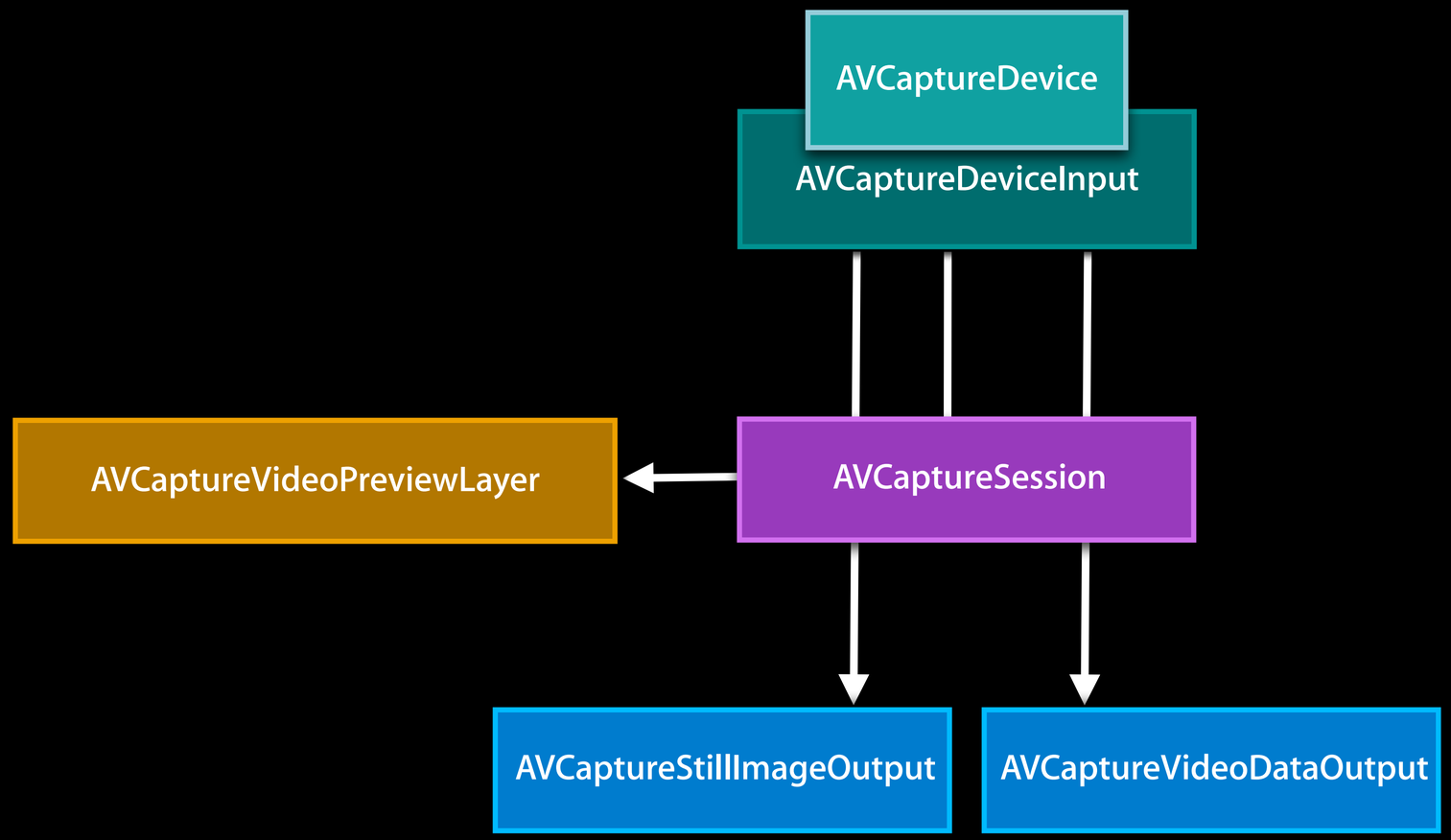
Configuration Using `-setSessionPreset:`



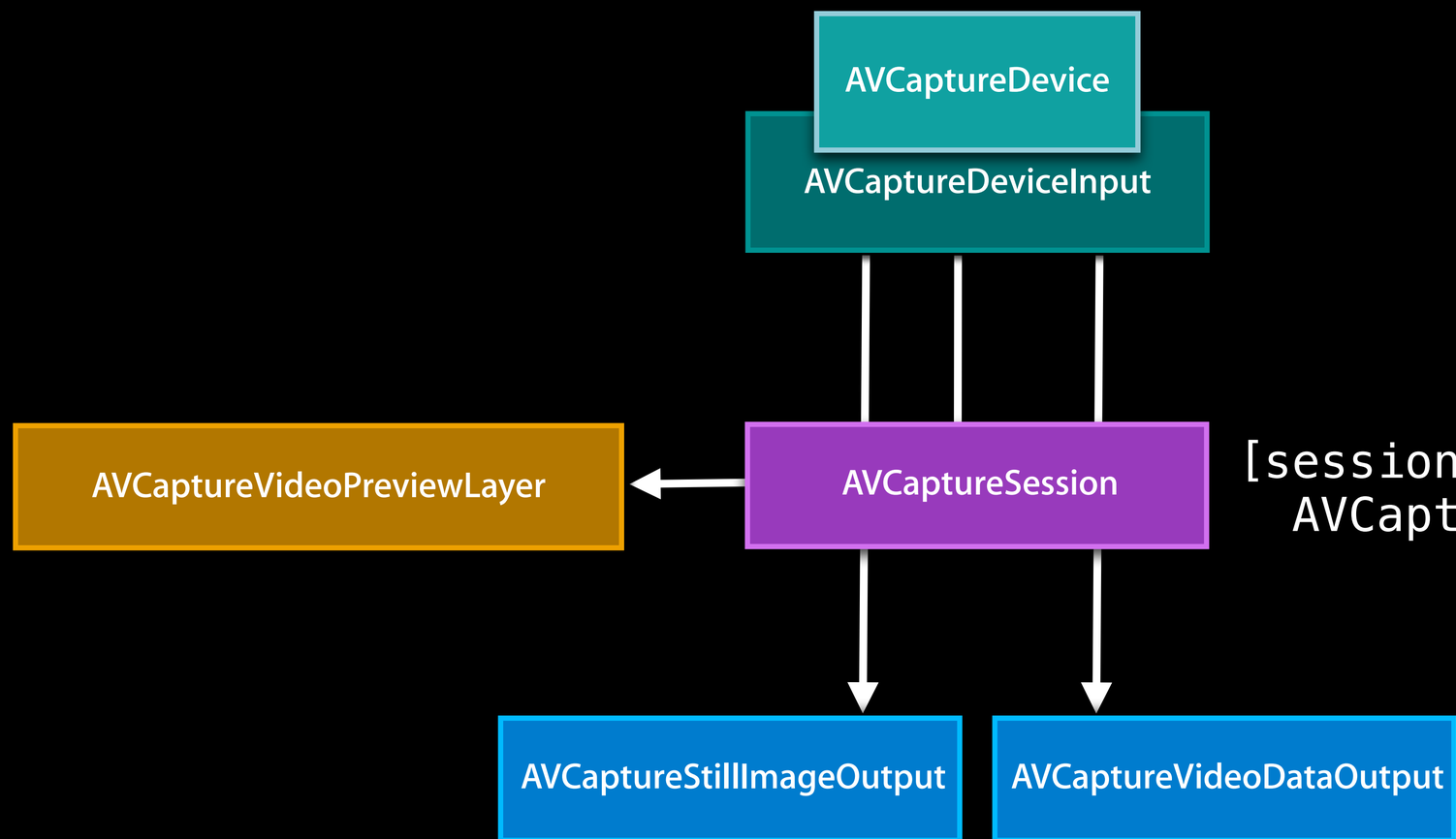
Configuration Using `-setSessionPreset:`



Configuration Using `-setSessionPreset:`

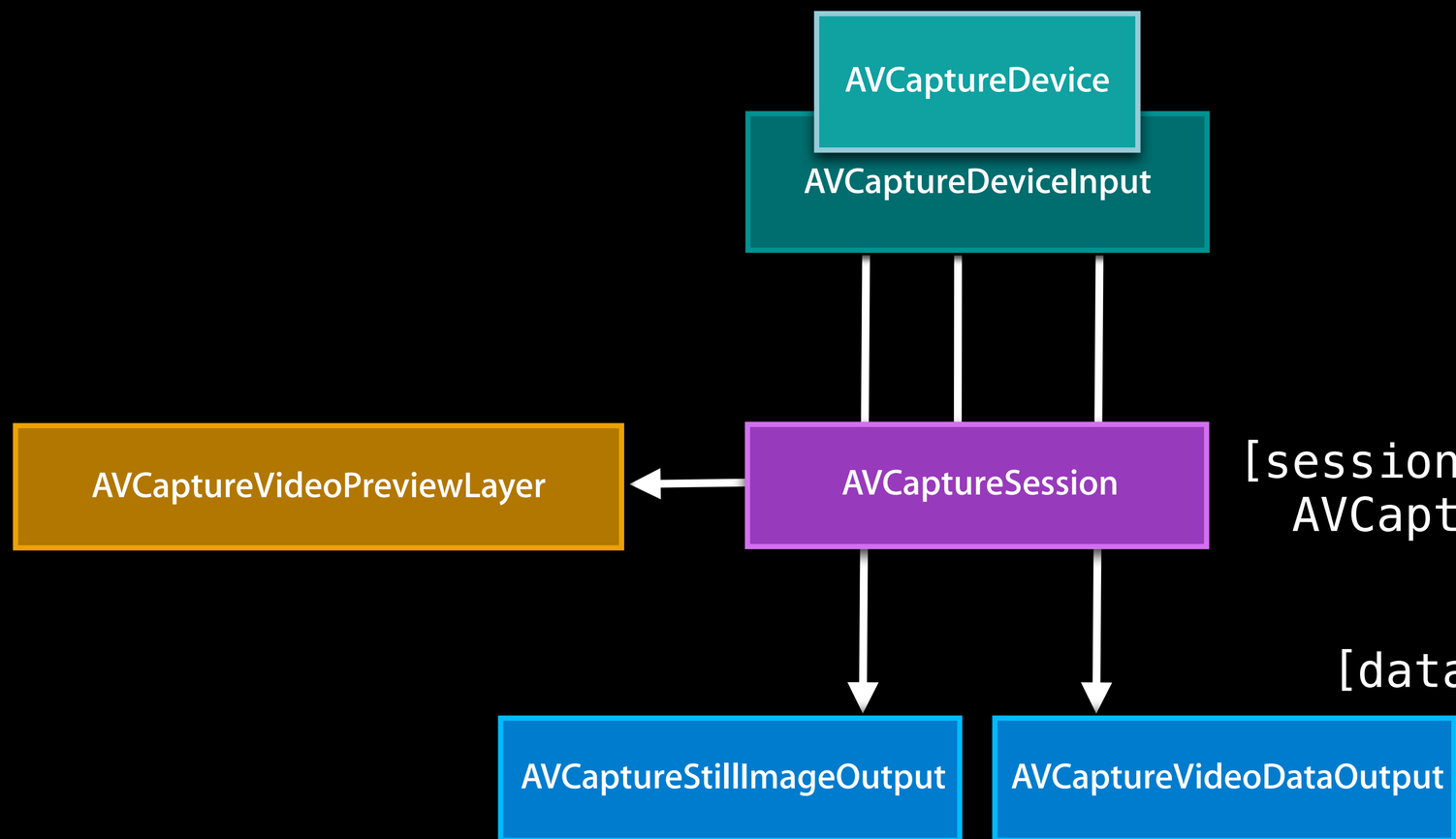


Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

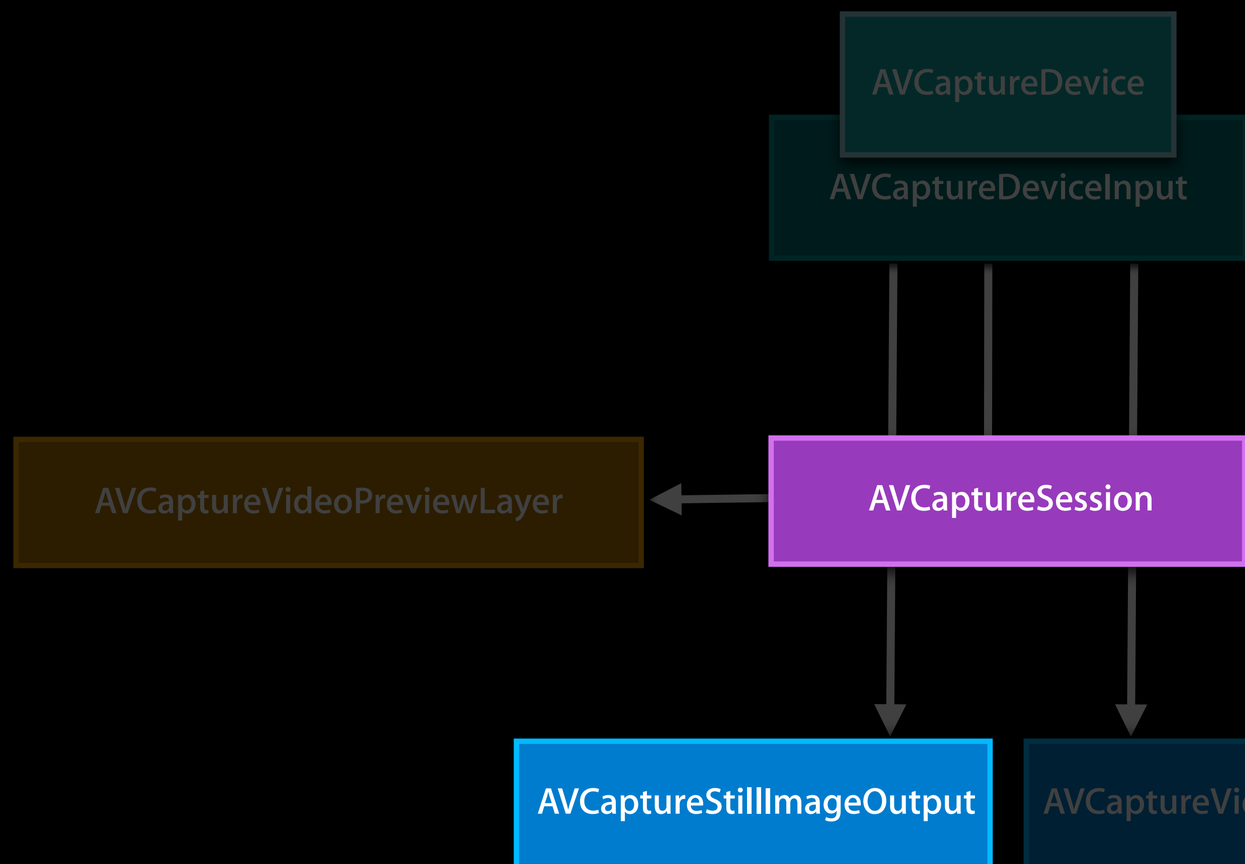
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

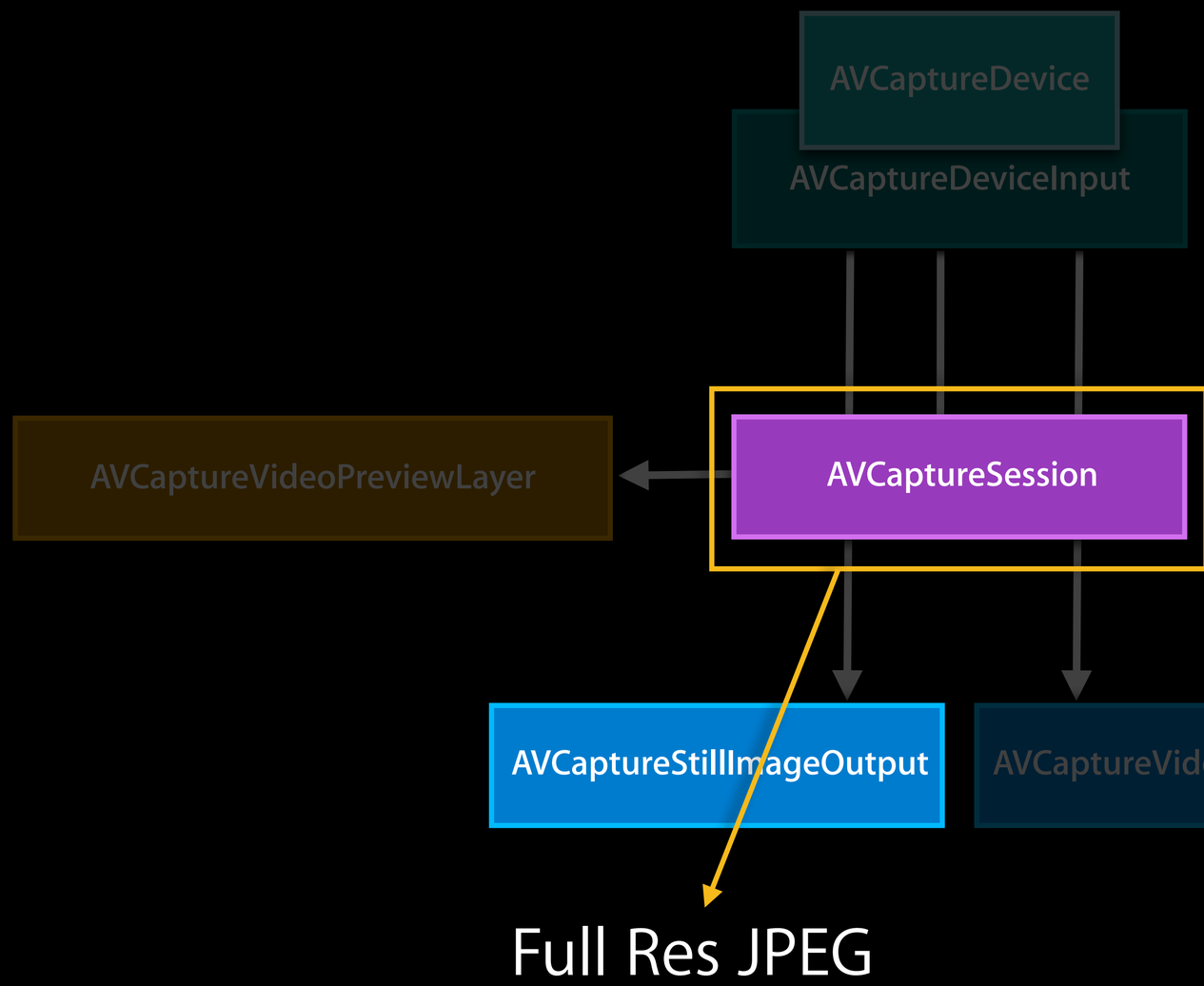
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

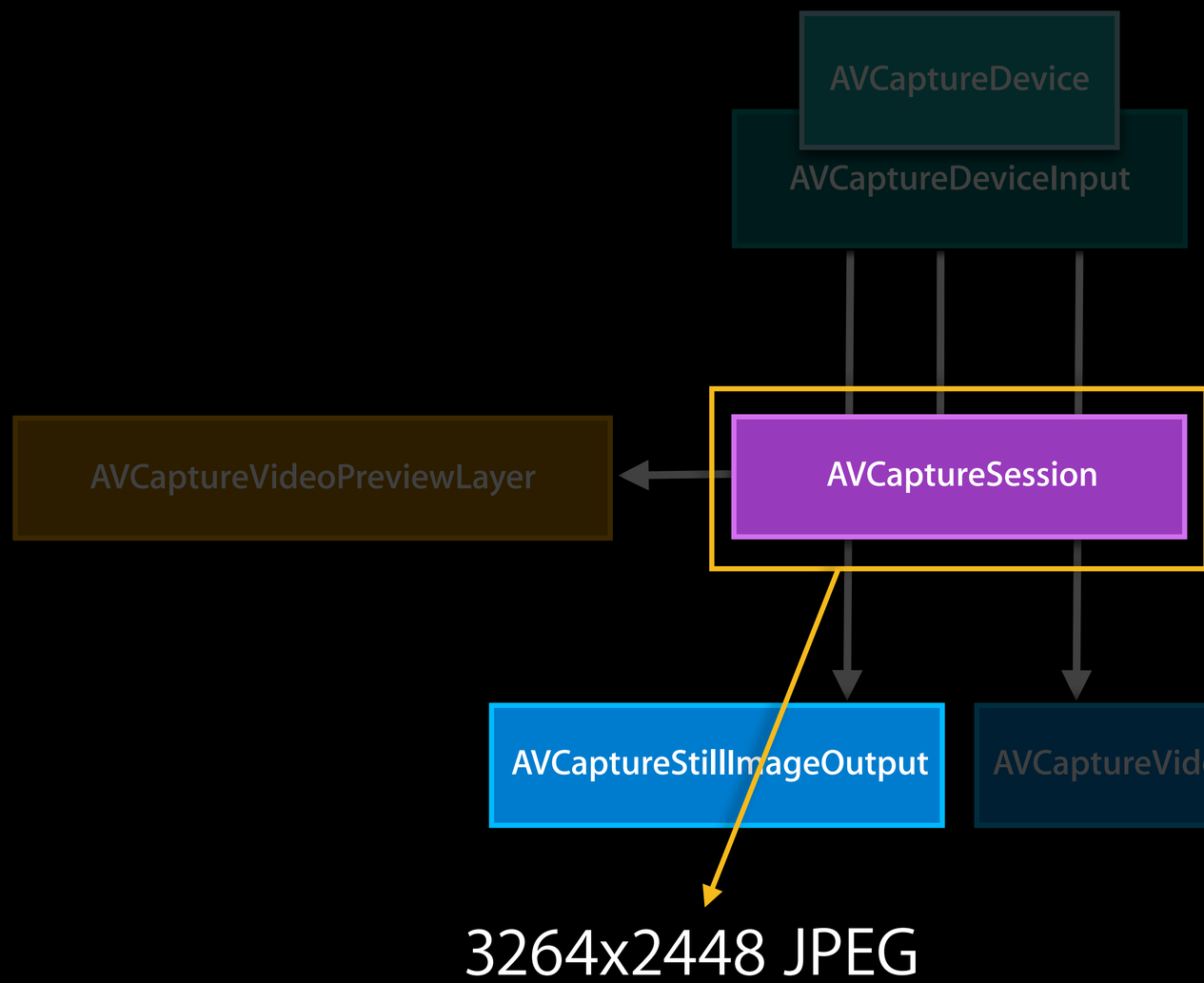
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
 AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
 @{ @"PixelFormatType" :
 @('BGRA') }];
```

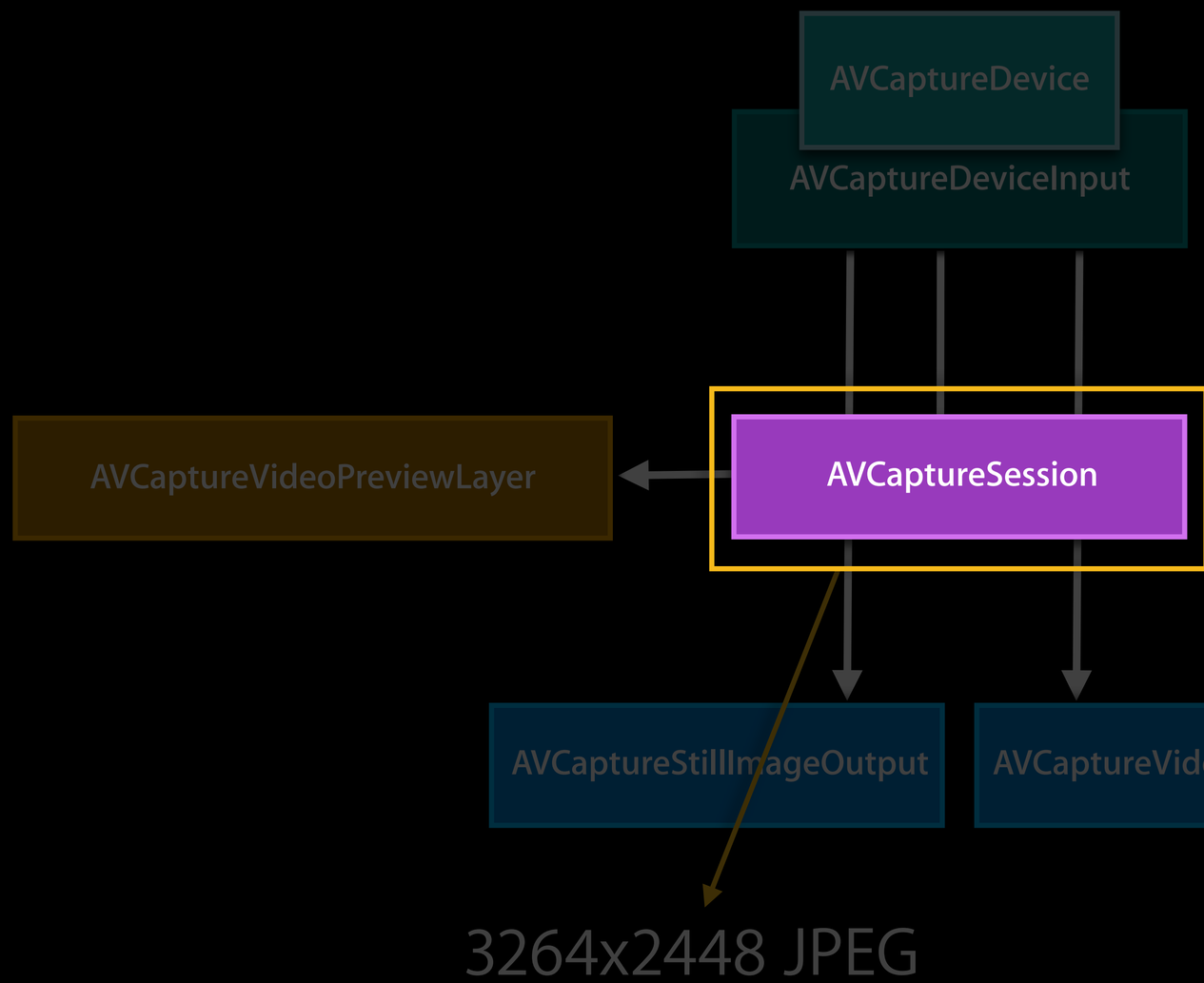
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
 AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
 @{ @"PixelFormatType" :
 @('BGRA') }];
```

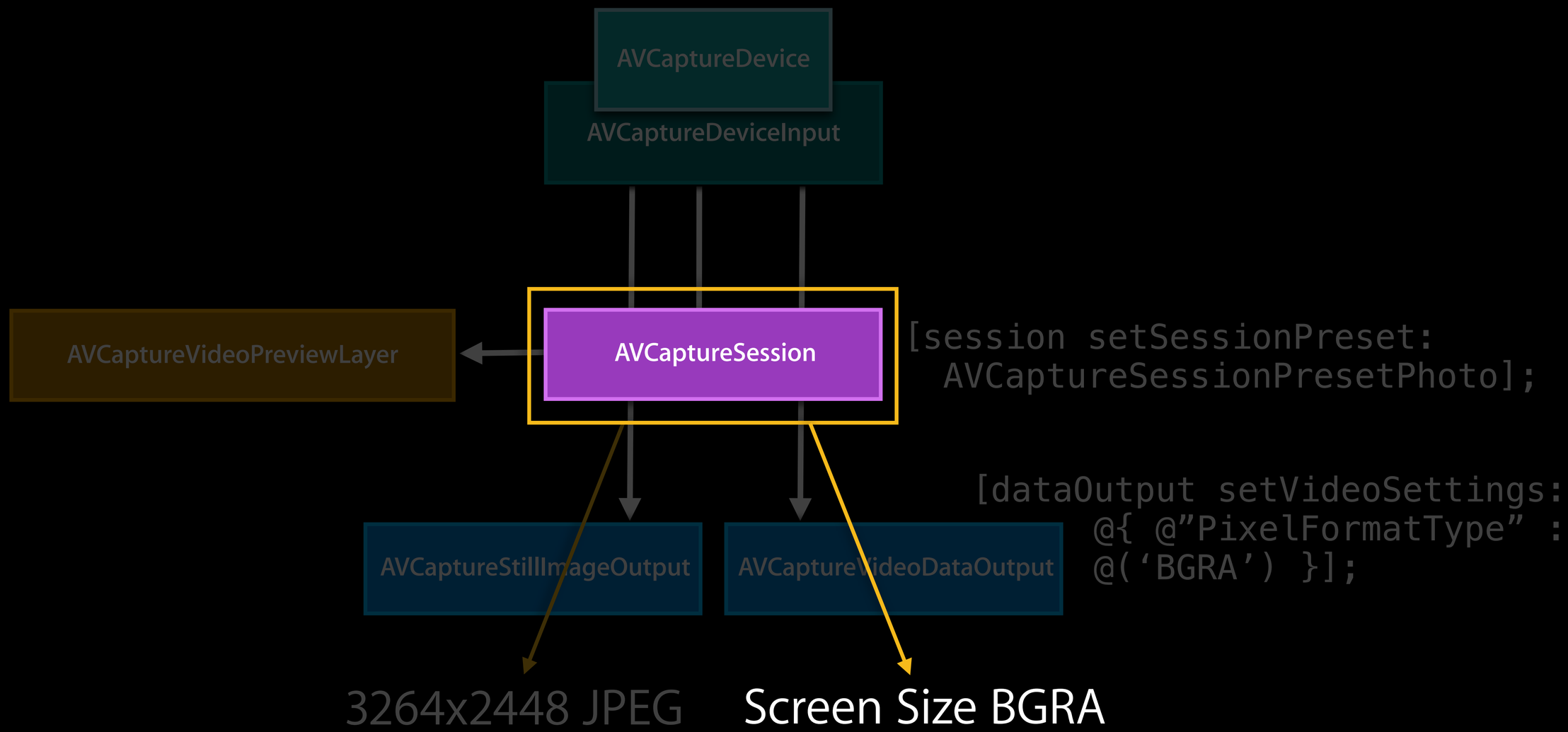
Configuration Using `-setSessionPreset:`



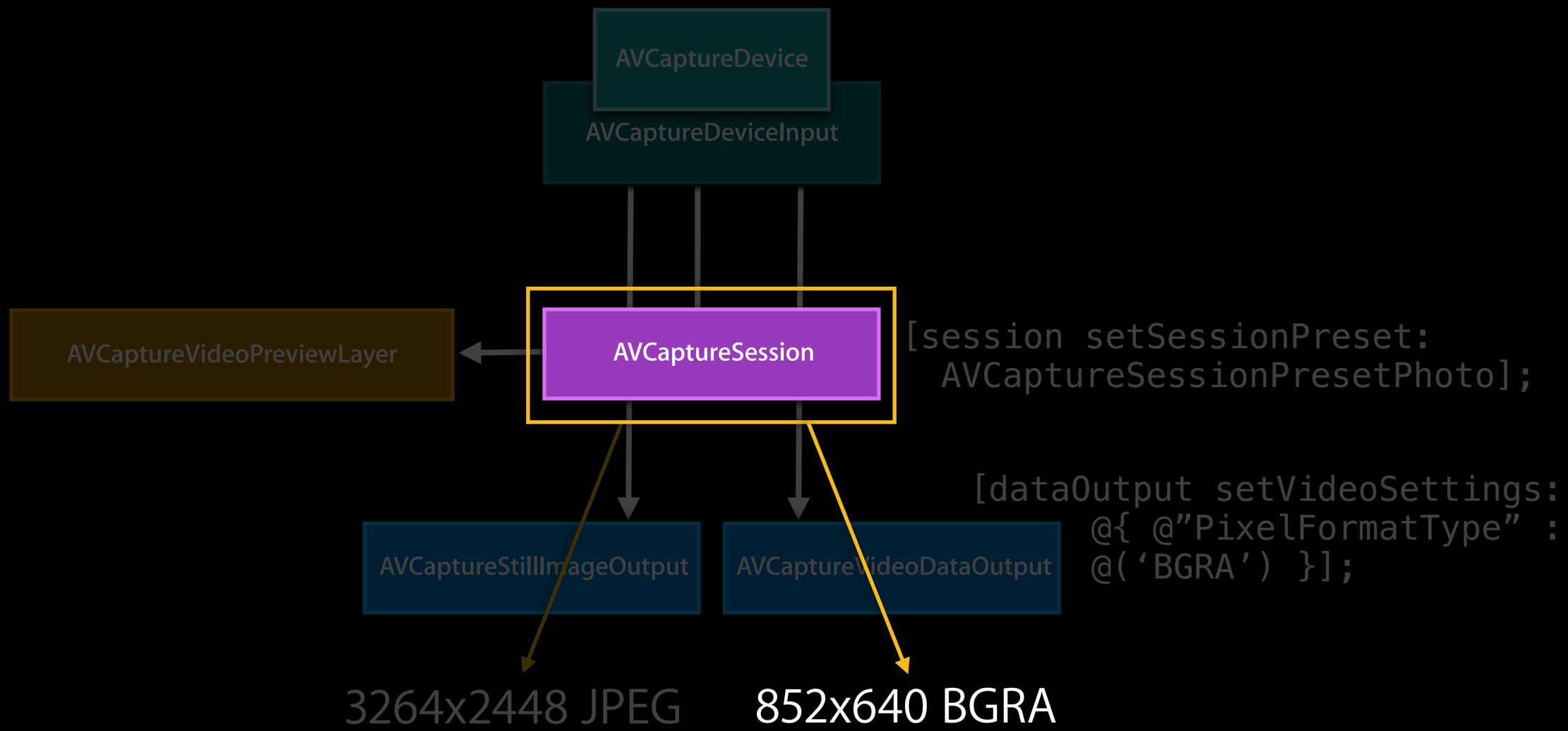

```
[session setSessionPreset:
 AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
 @{ @"PixelFormatType" :
 @('BGRA') }];
```

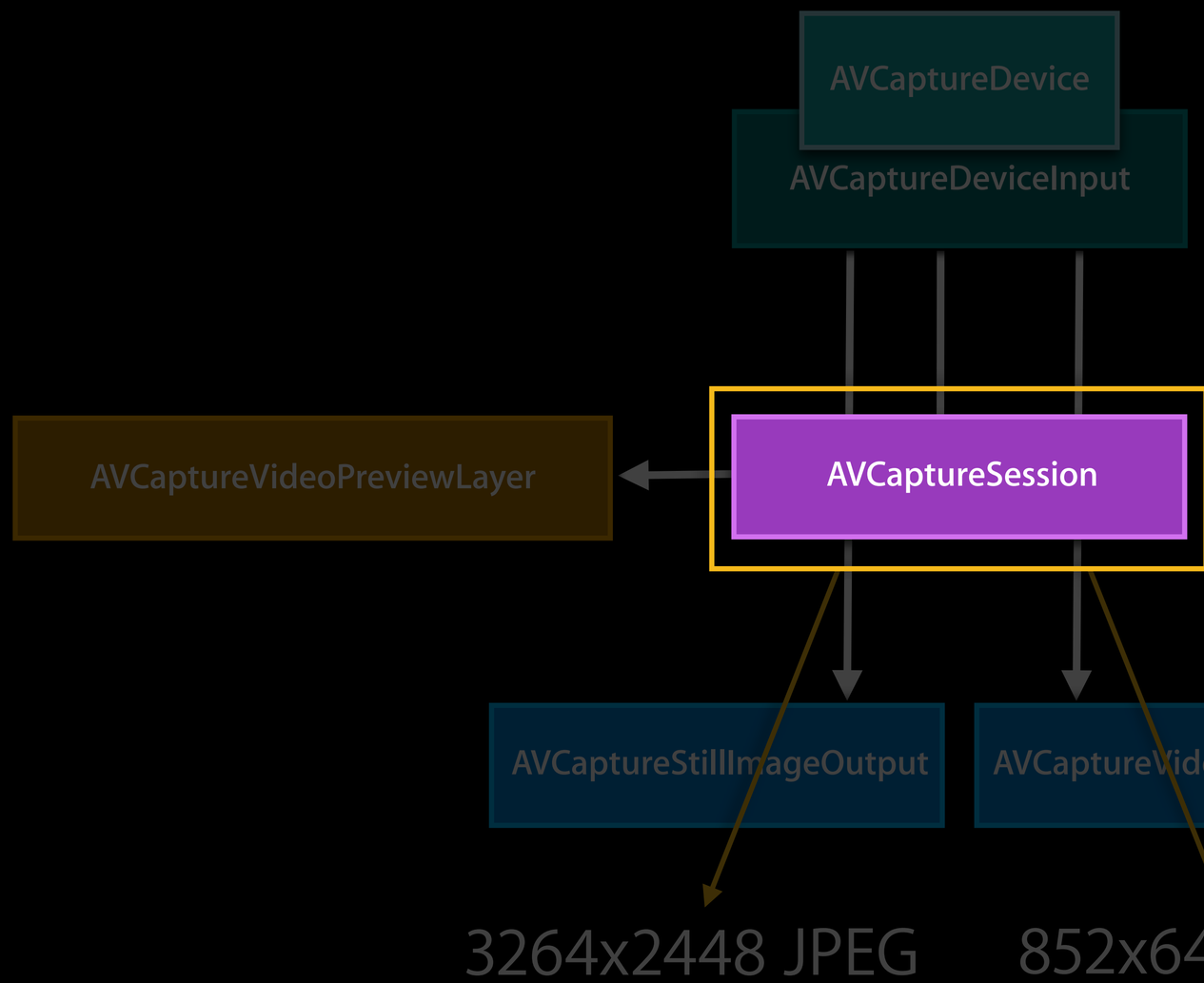
Configuration Using `-setSessionPreset:`



Configuration Using `-setSessionPreset:`



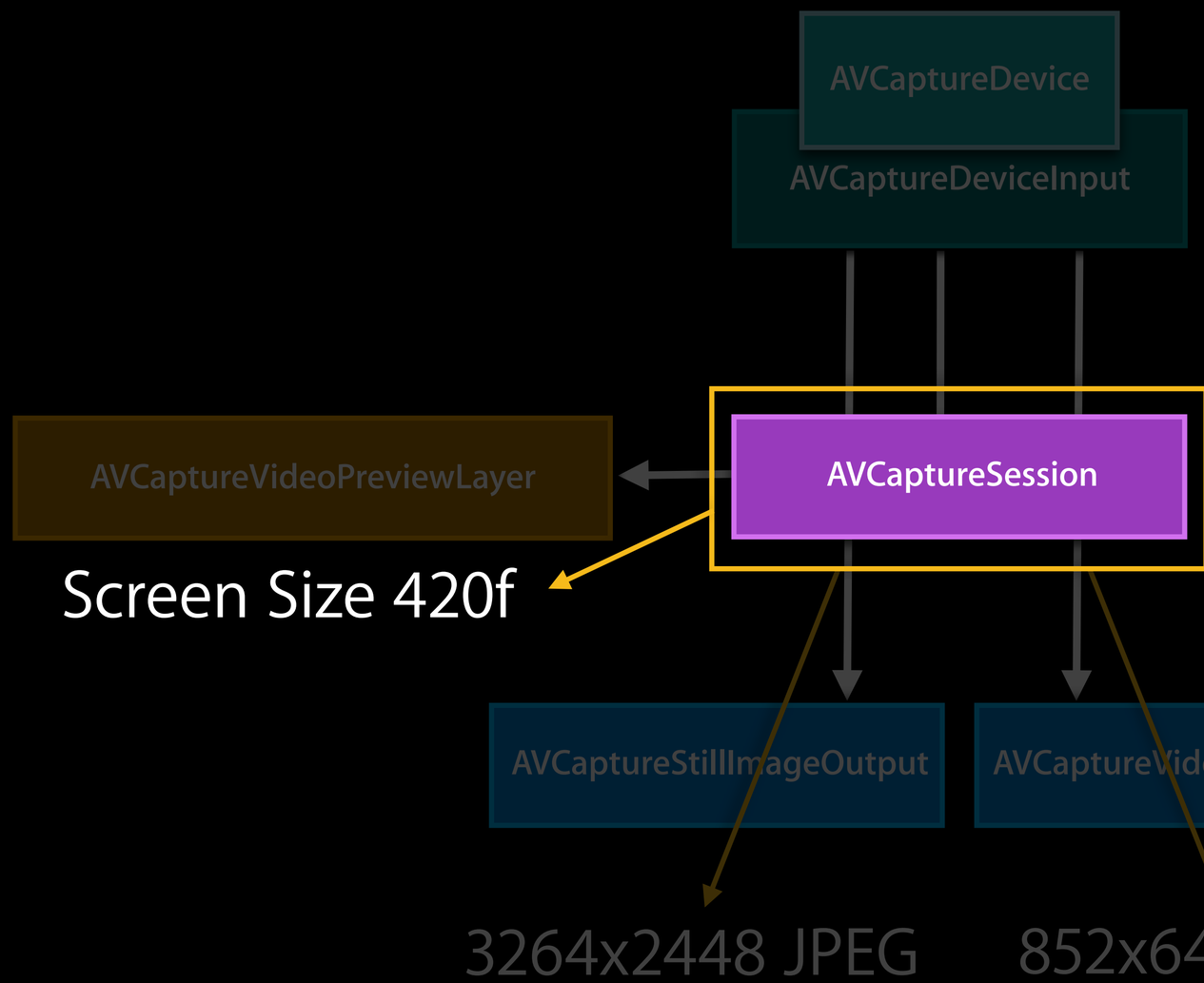
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
 AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
 @{ @"PixelFormatType" :
 @('BGRA') }];
```

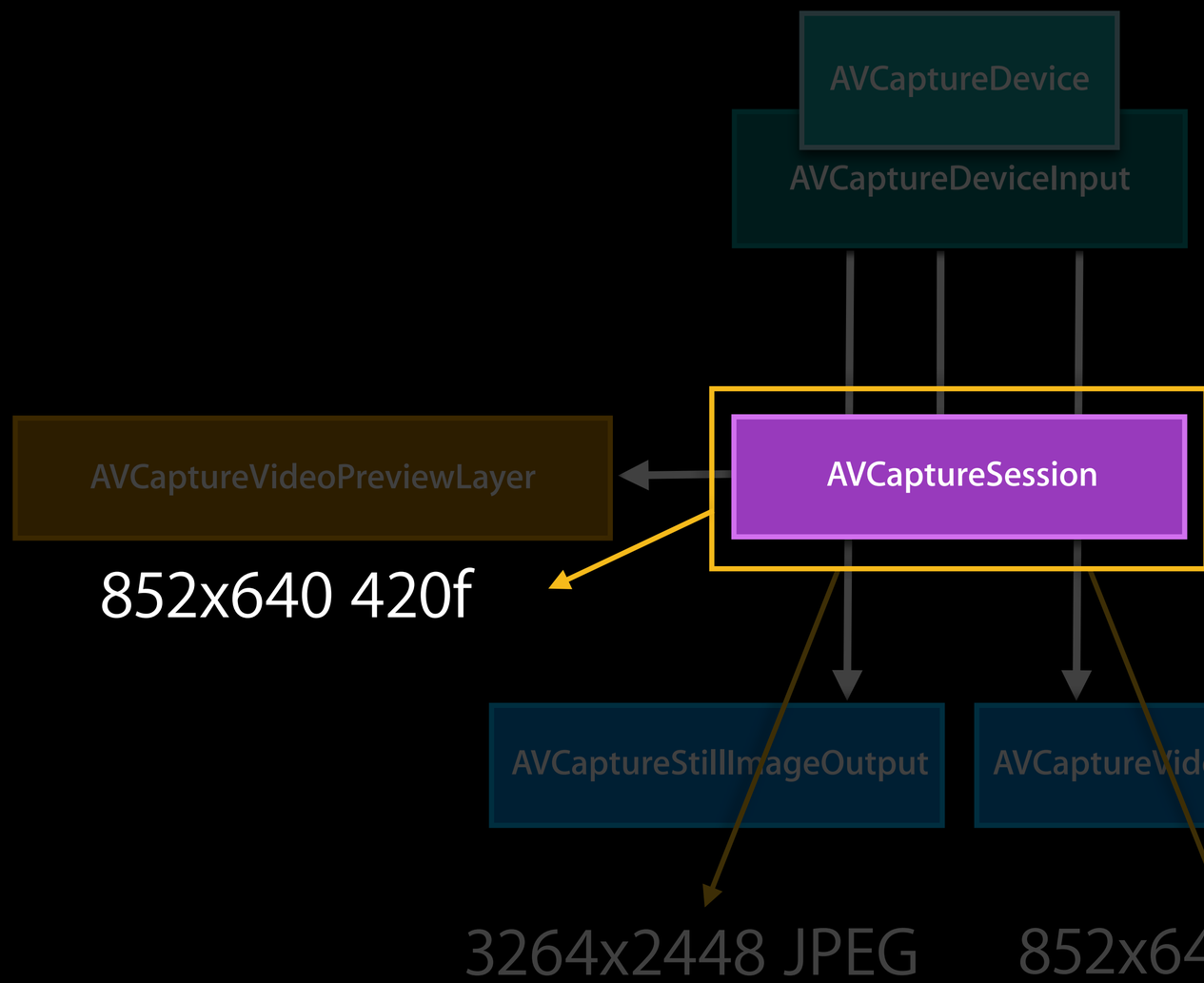
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
 AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
 @{ @"PixelFormatType" :
 @('BGRA') }];
```

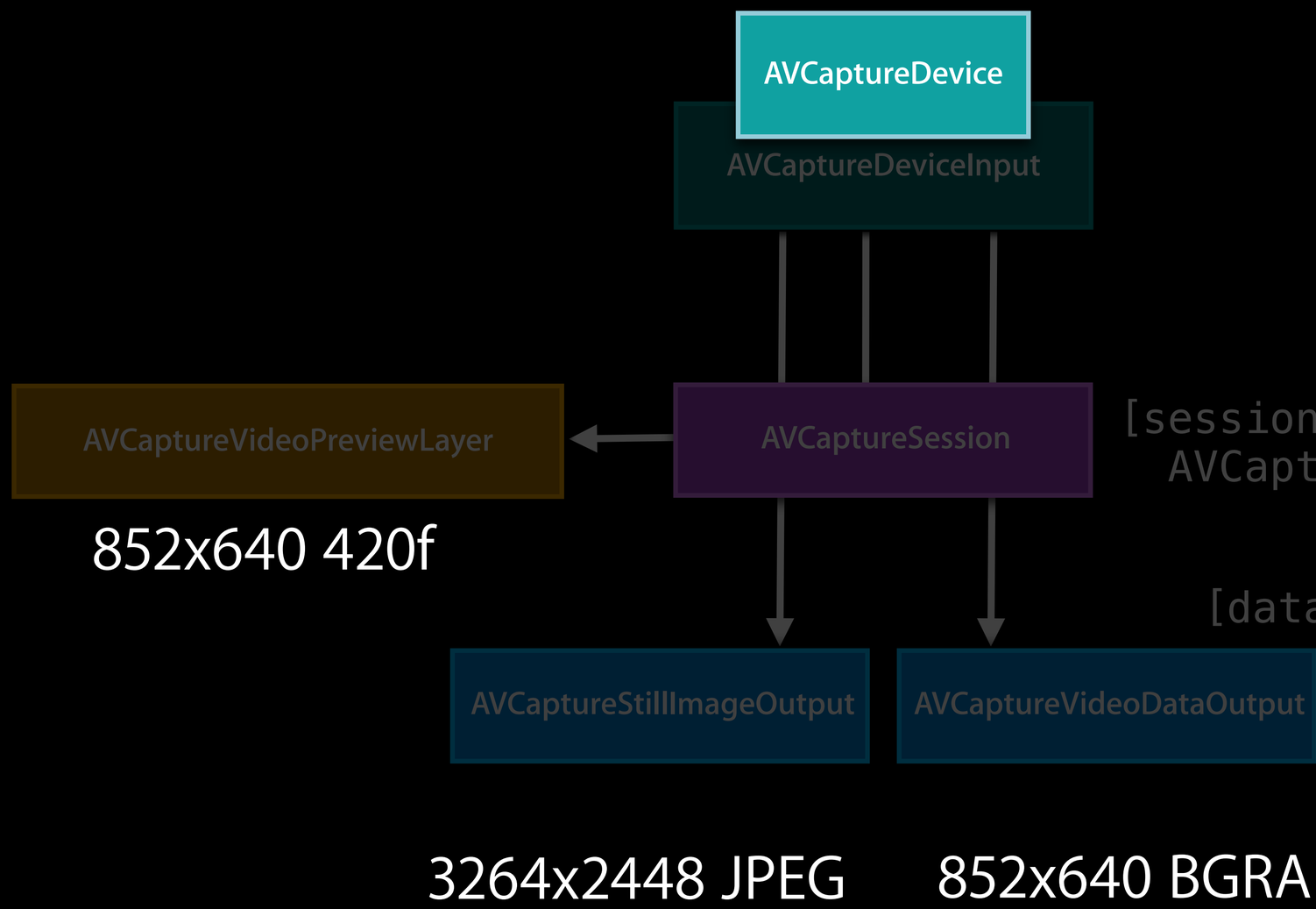
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

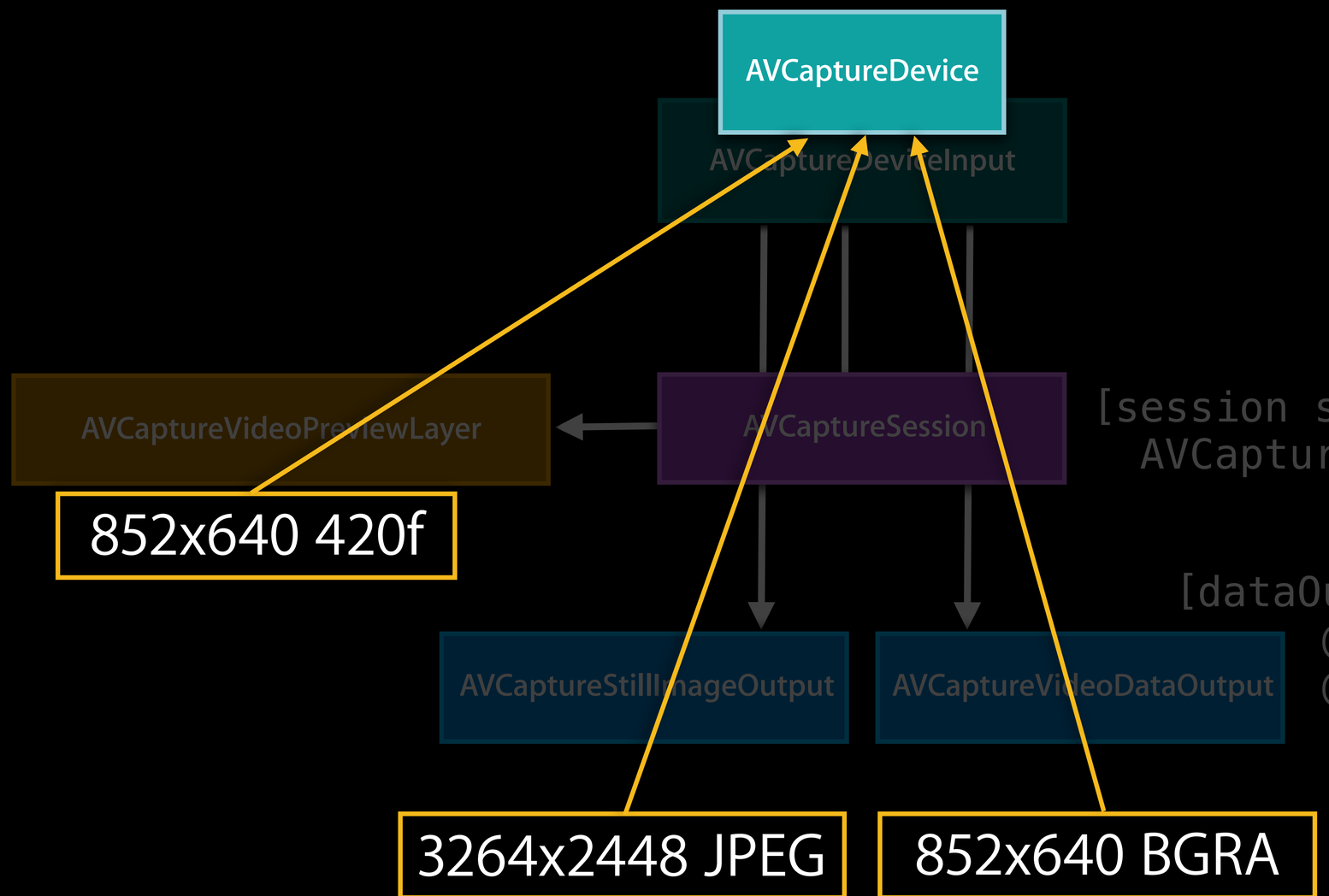
Configuration Using `-setSessionPreset:`




```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

Configuration Using –setSessionPreset:

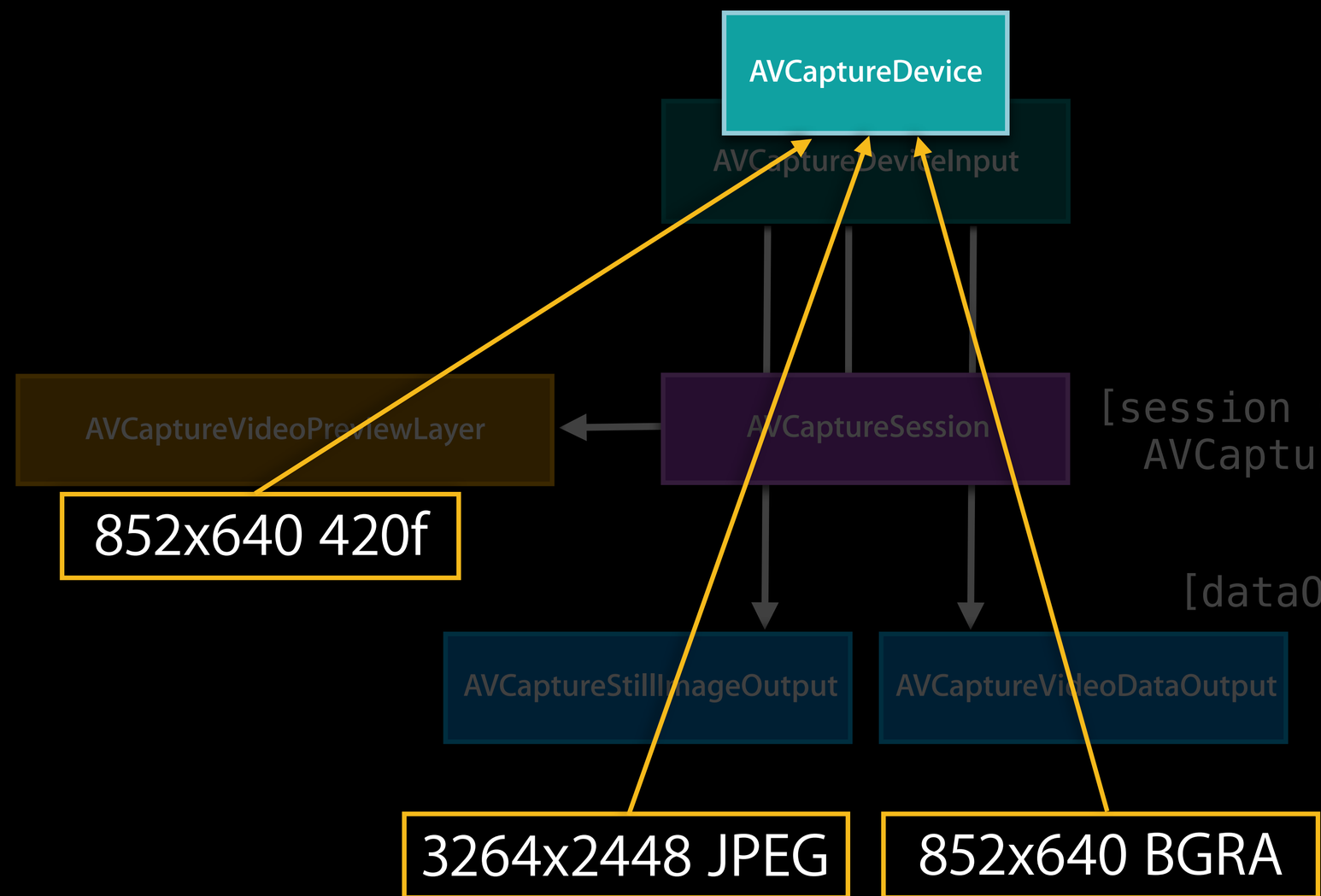



```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

Configuration Using –setSessionPreset:

```
NSArray *formats = [device formats];
```

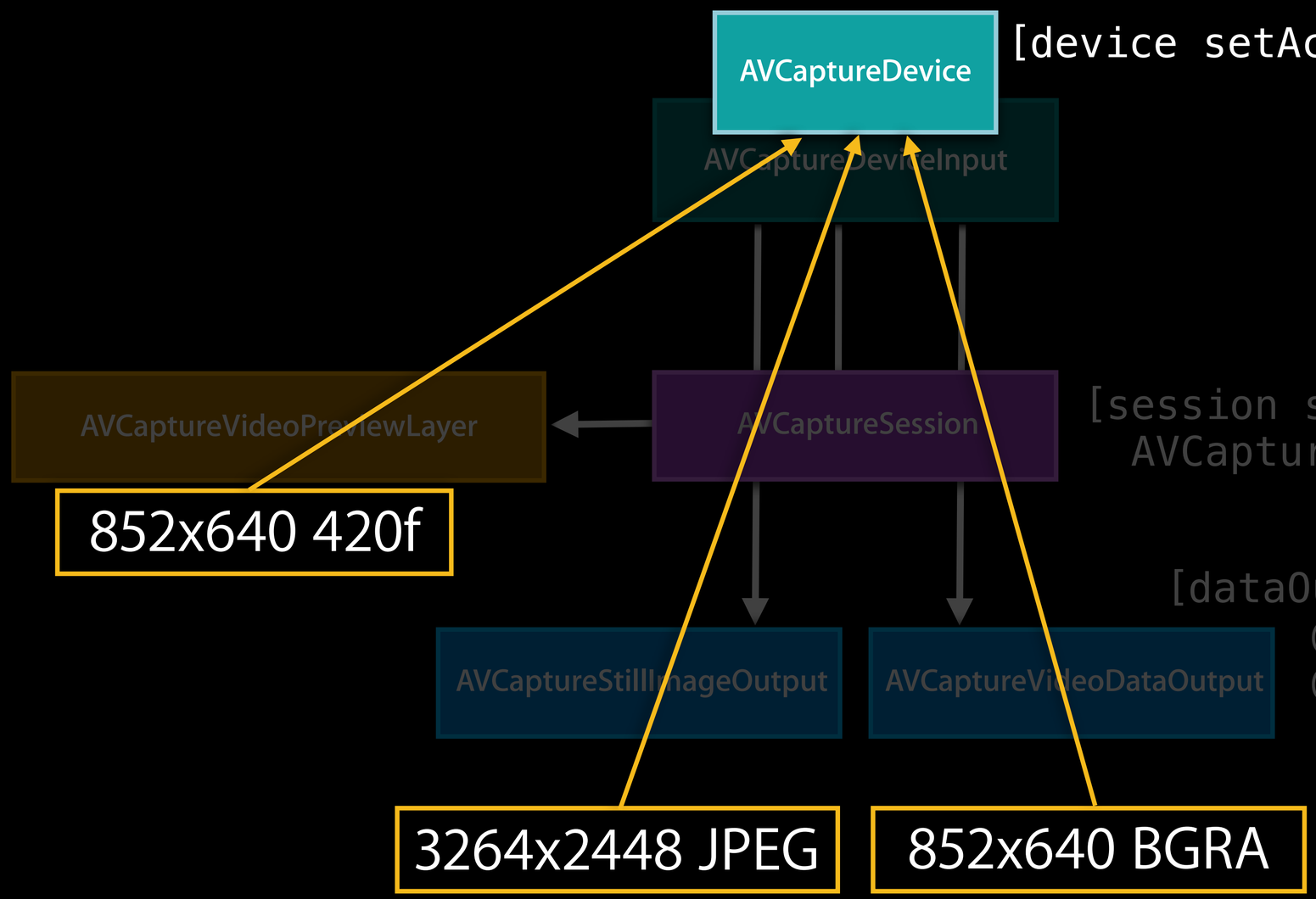



```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

Configuration Using –setSessionPreset:

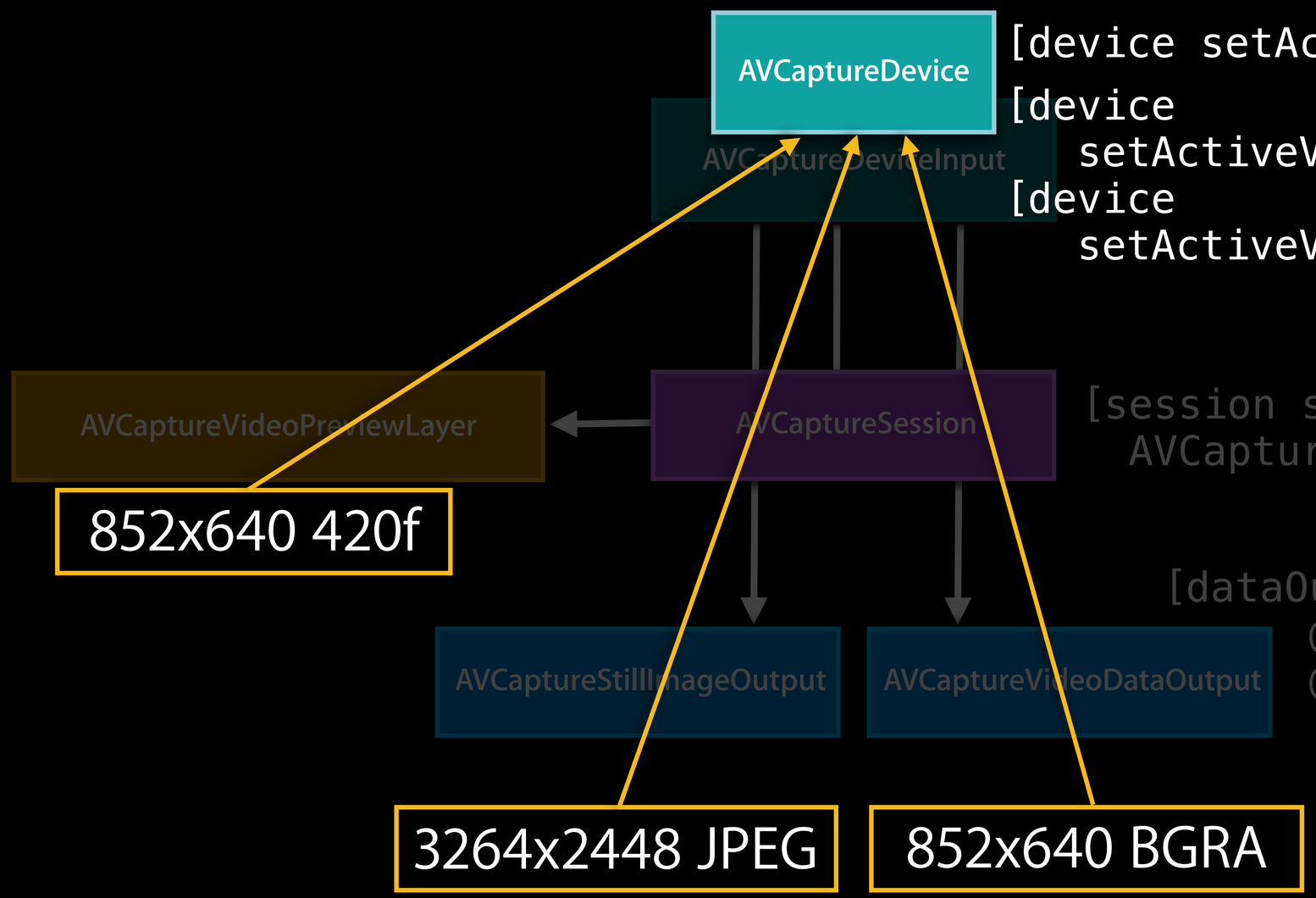
```
NSArray *formats = [device formats];
[device setActiveFormat:bestFormat];
```




```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
  @{ @"PixelFormatType" :
  @('BGRA') }];
```

Configuration Using –setSessionPreset:




```
NSArray *formats = [device formats];
[device setActiveFormat:bestFormat];
[device
    setActiveVideoMinFrameDuration:];
[device
    setActiveVideoMaxFrameDuration:];
```

```
[session setSessionPreset:
  AVCaptureSessionPresetPhoto];
```

```
[dataOutput setVideoSettings:
    @{ @"PixelFormatType" :
    @('BGRA') }];
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

Configuration Using Device Format Selection


```
- (void)configureCameraForHighestFrameRate:(AVCaptureDevice *)device
{
AVCaptureDeviceFormat *bestFormat = nil;
AVFrameRateRange *bestFrameRateRange = nil;

for ( AVCaptureDeviceFormat *format in [device formats] ) {
  for ( AVFrameRateRange *range in format.videoSupportedFrameRateRanges ) {
    if ( range.maxFrameRate > bestFrameRateRange.maxFrameRate ) {
      bestFormat = format;
      bestFrameRateRange = range;
    }
  }
}
if ( bestFormat ) {
  if ( YES == [device lockForConfiguration:NULL] ) {
    device.activeFormat = bestFormat;
    device.activeVideoMinFrameDuration = bestFrameRateRange.minFrameDuration;
    device.activeVideoMaxFrameDuration = bestFrameRateRange.minFrameDuration;
    [device unlockForConfiguration];
  }
}
}
```

AVCaptureDevice Frame Rate Selection

AVCaptureDevice Frame Rate Selection

- Use AVCaptureDevice for frame rate selection

```
[device setActiveVideoMinFrameDuration:];
[device setActiveVideoMaxFrameDuration:];
```

AVCaptureDevice Frame Rate Selection

- Use AVCaptureDevice for frame rate selection

```
[device setActiveVideoMinFrameDuration:];
[device setActiveVideoMaxFrameDuration:];
```

- AVCaptureDevice frame rates can be set any time

AVCaptureDevice Frame Rate Selection

- Use AVCaptureDevice for frame rate selection

```
[device setActiveVideoMinFrameDuration:];
[device setActiveVideoMaxFrameDuration:];
```

- AVCaptureDevice frame rates can be set any time
- Use kCMTimeInvalid to restore default frame rates

```
[device setActiveVideoMinFrameDuration:kCMTimeInvalid];
[device setActiveVideoMaxFrameDuration:kCMTimeInvalid];
```

AVCaptureDevice Frame Rate Selection

- Don’t use AVCaptureConnection for frame rate selection

```
[connection setVideoMinFrameDuration:];
[connection setVideoMaxFrameDuration:];
```

iPhone 5 Back-Facing Camera Formats

Dimensions	Frame Rates	Field of View	Binned	Stabilization	Used by Preset
192x144	1-30	56.70	Yes	No	Low
352x288	1-30	51.98	Yes	No	352x288
480x360	1-30	56.70	Yes	No	Medium
640x480	1-30	56.70	Yes	No	640x480
960x540	1-30	53.89	No	Yes	iFrame
1280x720	1-30	53.89	No	Yes	1280x720
1280x720	1-60	51.94	Yes	Yes	
1920x1080	1-30	53.89	No	Yes	High
2592x1936	1-20	56.70	No	No	
3264x2448	1-20	56.70	No	No	Photo

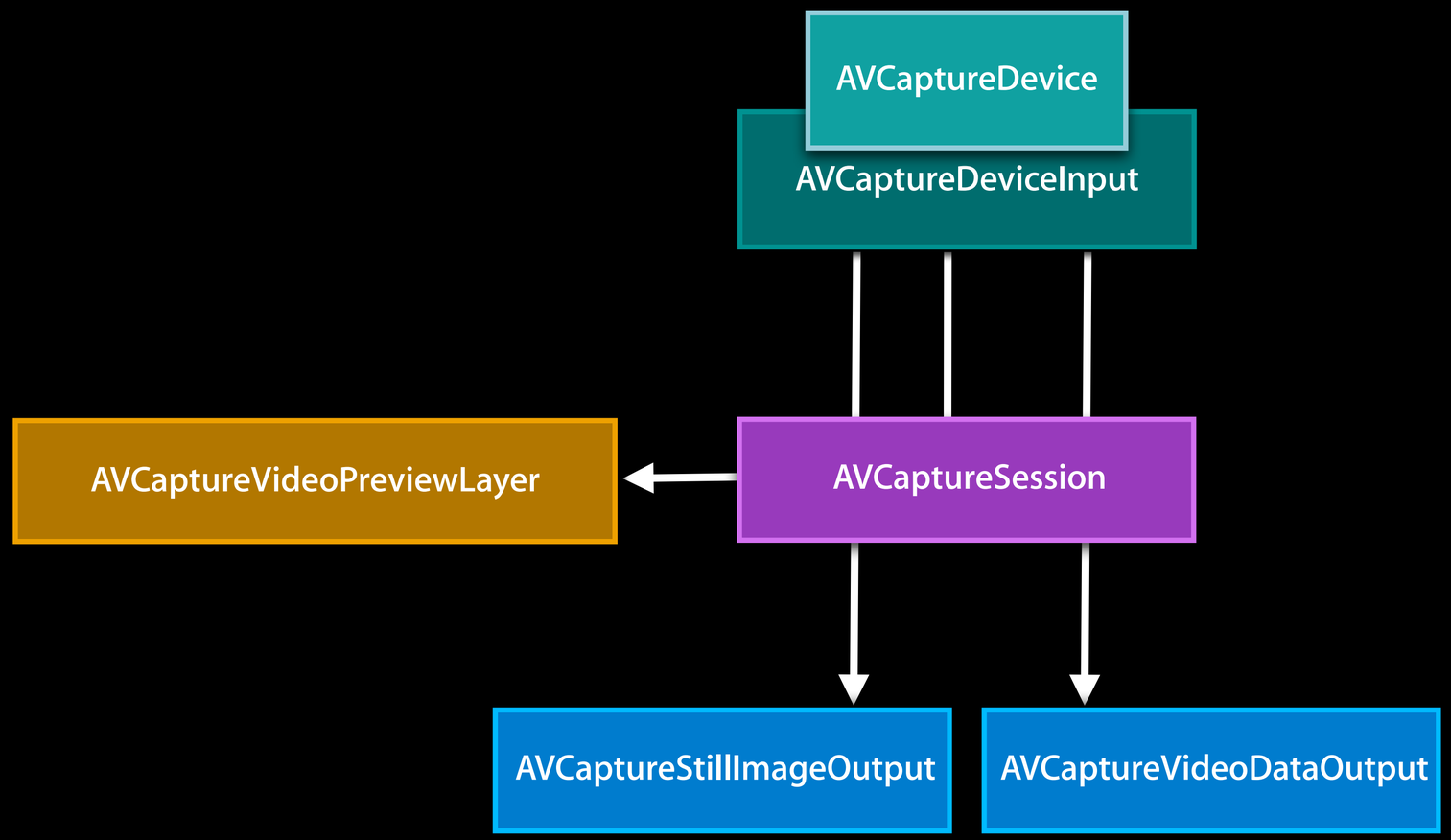
iPhone 5 Back-Facing Camera Formats

Dimensions	Frame Rates	Field of View	Binned	Stabilization	Used by Preset
192x144	1-30	56.70	Yes	No	Low
352x288	1-30	51.98	Yes	No	352x288
480x360	1-30	56.70	Yes	No	Medium
640x480	1-30	56.70	Yes	No	640x480
960x540	1-30	53.89	No	Yes	iFrame
1280x720	1-30	53.89	No	Yes	1280x720
1280x720	1-60	51.94	Yes	Yes	
1920x1080	1-30	53.89	No	Yes	High
2592x1936	1-20	56.70	No	No	
3264x2448	1-20	56.70	No	No	Photo

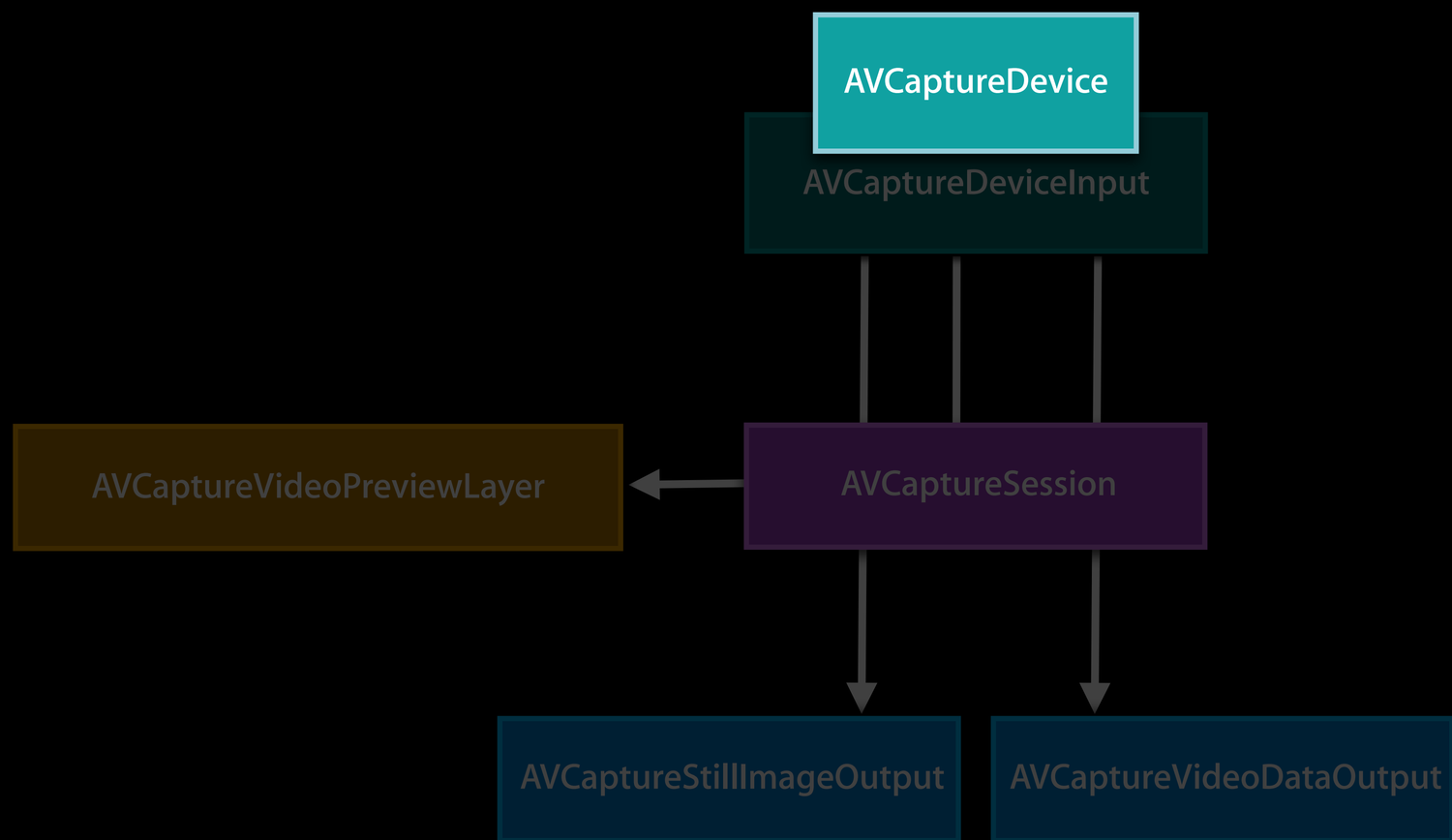
iPhone 5 Back-Facing Camera Formats

Dimensions	Frame Rates	Field of View	Binned	Stabilization	Used by Preset
192x144	1-30	56.70	Yes	No	Low
352x288	1-30	51.98	Yes	No	352x288
480x360	1-30	56.70	Yes	No	Medium
640x480	1-30	56.70	Yes	No	640x480
960x540	1-30	53.89	No	Yes	iFrame
1280x720	1-30	53.89	No	Yes	1280x720
1280x720	1-60	51.94	Yes	Yes	
1920x1080	1-30	53.89	No	Yes	High
2592x1936	1-20	56.70	No	No	
3264x2448	1-20	56.70	No	No	Photo

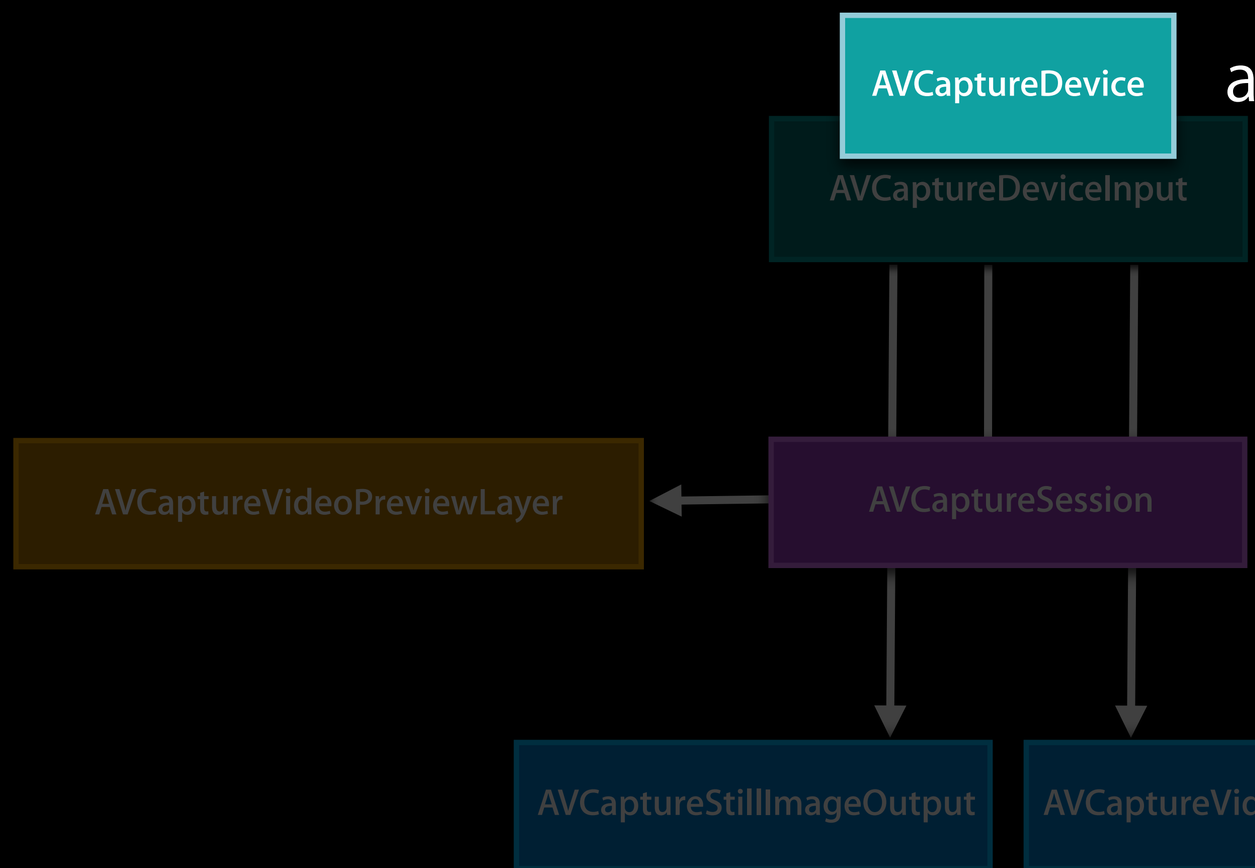
Configuration Using Device Format Selection



Configuration Using Device Format Selection



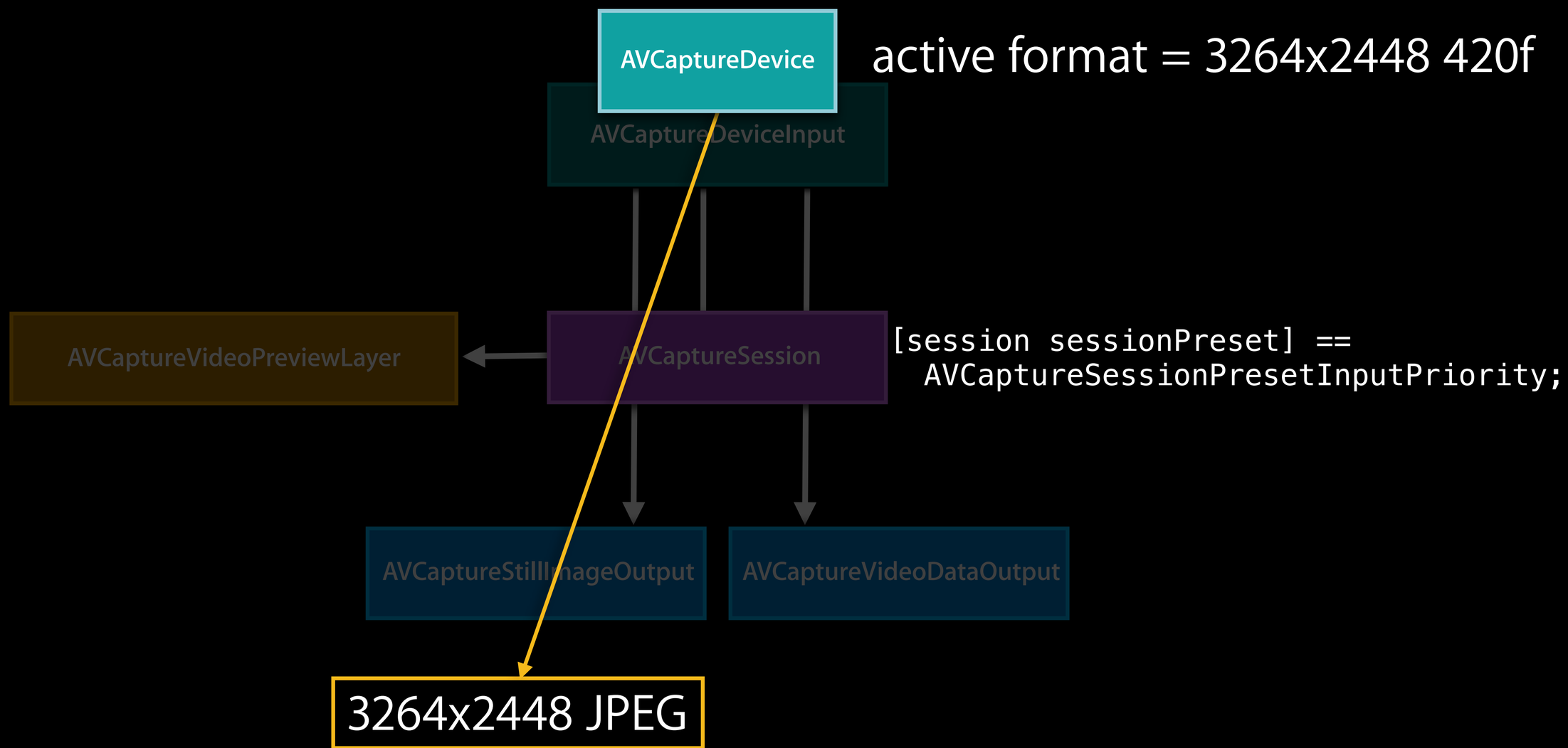
Configuration Using Device Format Selection



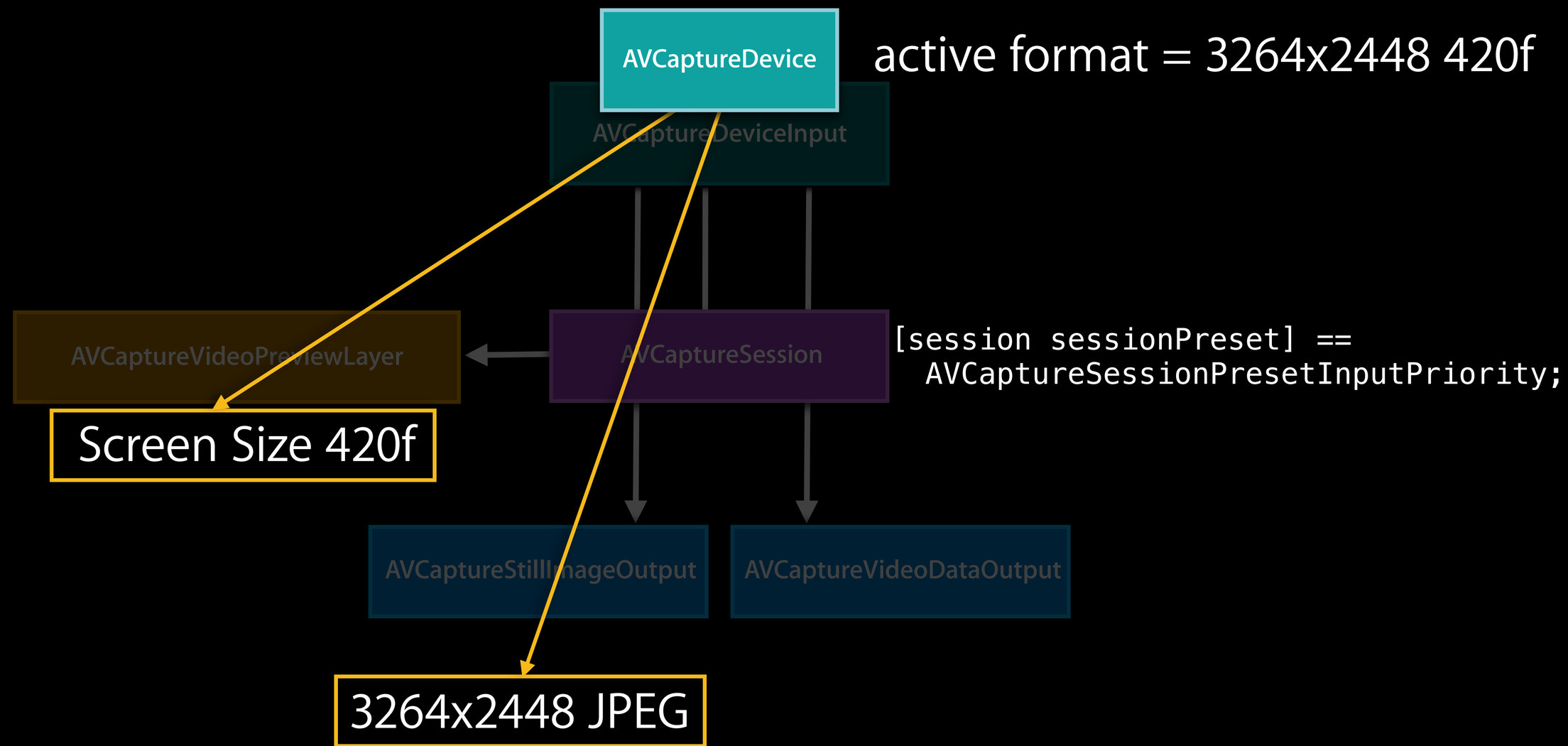
active format = 3264x2448 420f

```
[session sessionPreset] ==
  AVCaptureSessionPresetInputPriority;
```

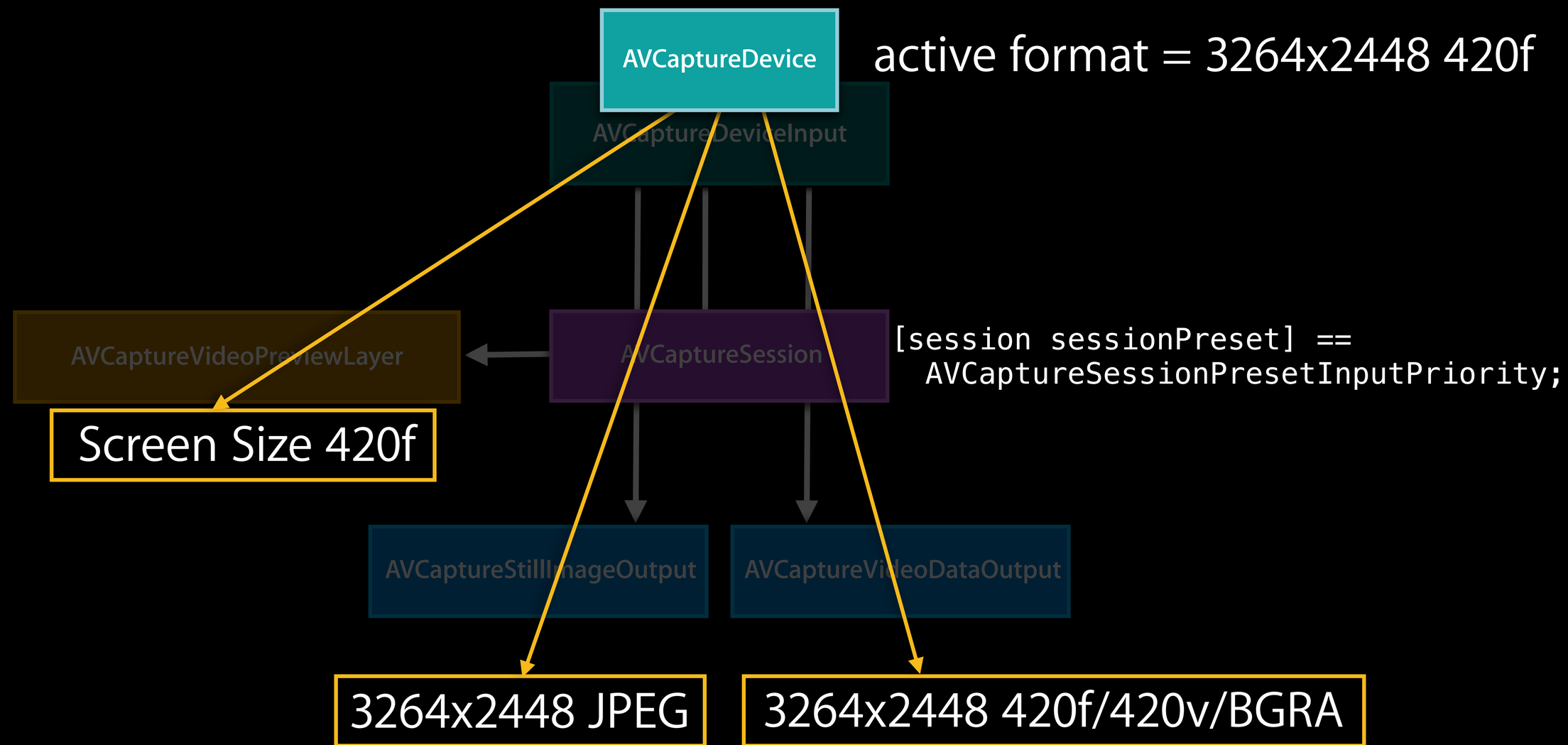
Configuration Using Device Format Selection



Configuration Using Device Format Selection



Configuration Using Device Format Selection



What's in a Format?

AVCaptureDeviceFormat

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`
  - Pixel format—`CMFormatDescriptionGetMediaSubType(fdesc);`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`
  - Pixel format—`CMFormatDescriptionGetMediaSubType(fdesc);`
  - Dimensions—`CMVideoFormatDescriptionGetDimensions(fdesc);`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`
  - Pixel format—`CMFormatDescriptionGetMediaSubType(fdesc);`
  - Dimensions—`CMVideoFormatDescriptionGetDimensions(fdesc);`
- `@property(readonly) float videoFieldOfView;`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`
  - Pixel format—`CMFormatDescriptionGetMediaSubType(fdesc);`
  - Dimensions—`CMVideoFormatDescriptionGetDimensions(fdesc);`
- `@property(readonly) float videoFieldOfView;`
- `@property(readonly) BOOL videoStabilizationSupported;`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`
  - Pixel format—`CMFormatDescriptionGetMediaSubType(fdesc);`
  - Dimensions—`CMVideoFormatDescriptionGetDimensions(fdesc);`
- `@property(readonly) float videoFieldOfView;`
- `@property(readonly) BOOL videoStabilizationSupported;`
- `@property(readonly) NSArray *videoSupportedFrameRateRanges;`

What's in a Format?

AVCaptureDeviceFormat

- `@property(readonly) NSString *mediaType;`
- `@property(readonly) CMFormatDescriptionRef formatDescription;`
  - Pixel format—`CMFormatDescriptionGetMediaSubType(fdesc);`
  - Dimensions—`CMVideoFormatDescriptionGetDimensions(fdesc);`
- `@property(readonly) float videoFieldOfView;`
- `@property(readonly) BOOL videoStabilizationSupported;`
- `@property(readonly) NSArray *videoSupportedFrameRateRanges;`
- `@property(readonly, getter=isVideoBinned) BOOL videoBinned;`

iPhone 5 Back-Facing Camera Formats

Dimensions	Frame Rates	Field of View	Binned	Stabilization	Used by Preset
1280x720	1-30	53.89	No	Yes	1280x720
1280x720	1-60	51.94	Yes	Yes	

iPhone 5 Back-Facing Camera Formats

Dimensions	Frame Rates	Field of View	Binned	Stabilization	Used by Preset
1280x720	1-30	53.89	No	Yes	1280x720
1280x720	1-60	51.94	Yes	Yes	

When to Use `[session setSessionPreset:]`

- Presets guarantee quality of service
- Presets optimally configure outputs and inputs
- Presets provide the best "bang for the buck" camera configuration
- Presets are the preferred mechanism for simple camera apps

When to Use `[device setActiveFormat:]`

- When you need precise control, such as:
  - A sensor format that supports 60 FPS
  - A specific resolution with a specific field of view
  - Full-resolution video data output buffers

New in iOS 7

- 60 FPS Support
- Video zoom
- Machine Readable Code detection
- Focus enhancements
- Integration with application AudioSession

Video Zoom

Ethan Tira-Thompson
Core Media Engineering

Zooooooom!

Zooooooom!

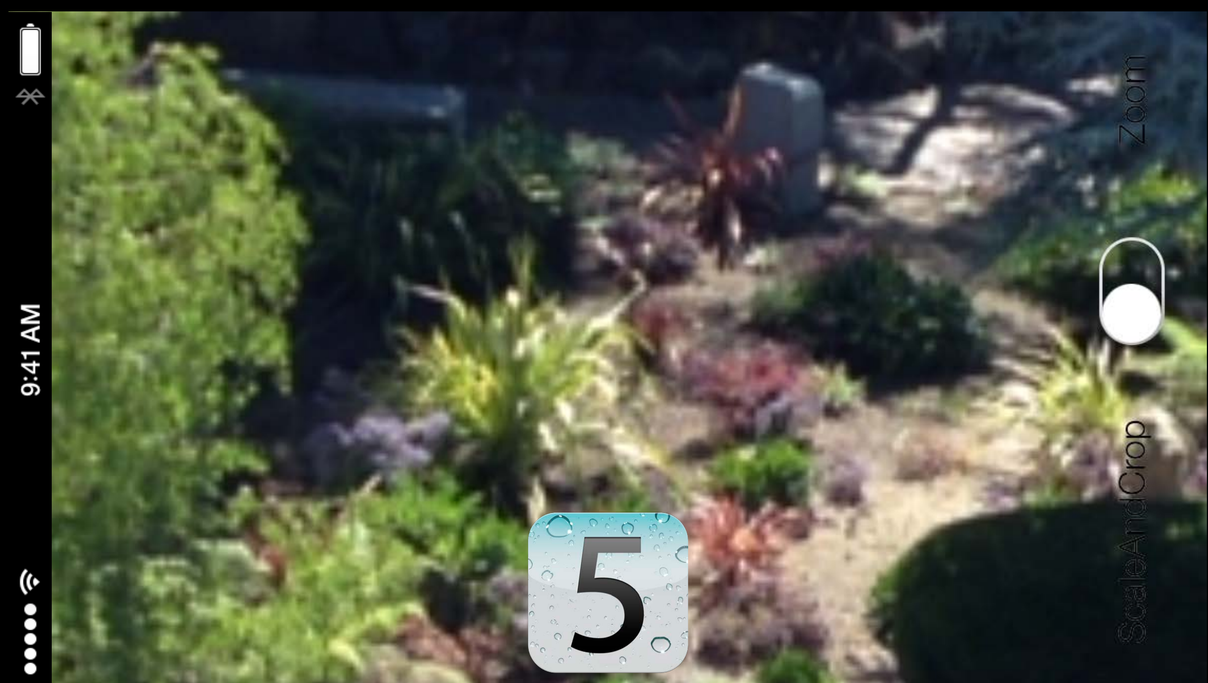
AVCaptureConnection
videoScaleAndCropFactor
(still images only)

Zooooooom!

Preview Layer Transformation

Zooooooom!

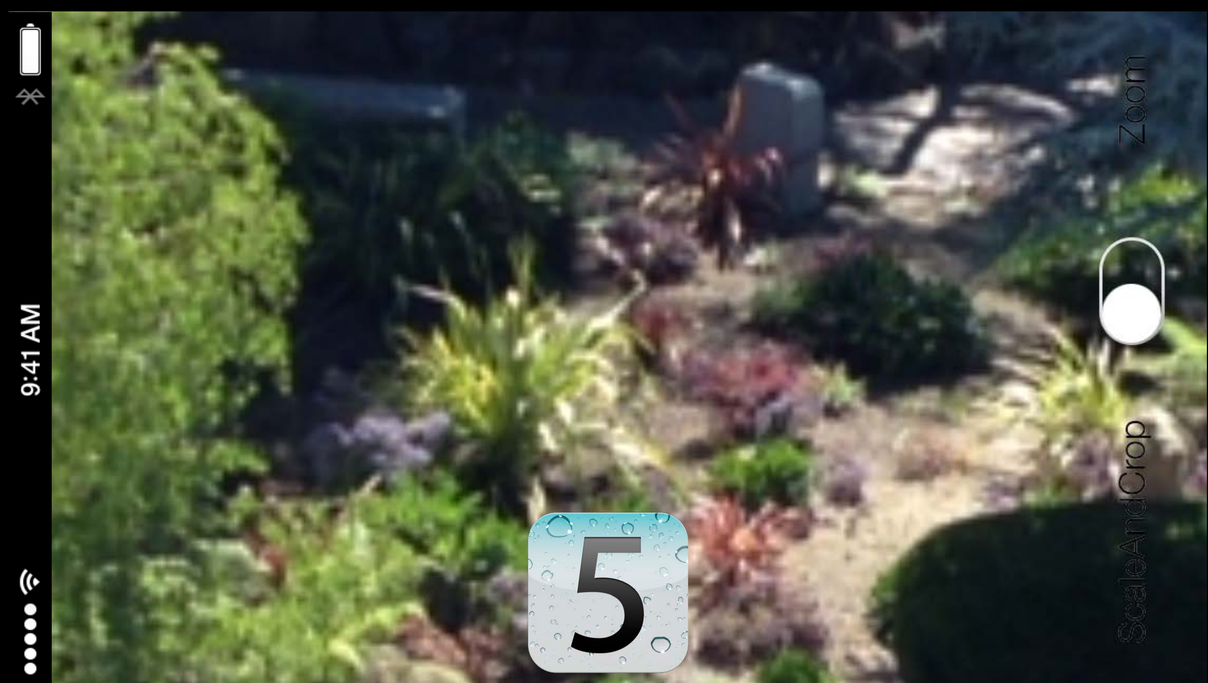
iPhone 5 Preview @ 5x Zoom



Preview Layer Transformation

Zooooooom!

iPhone 5 Preview @ 5x Zoom

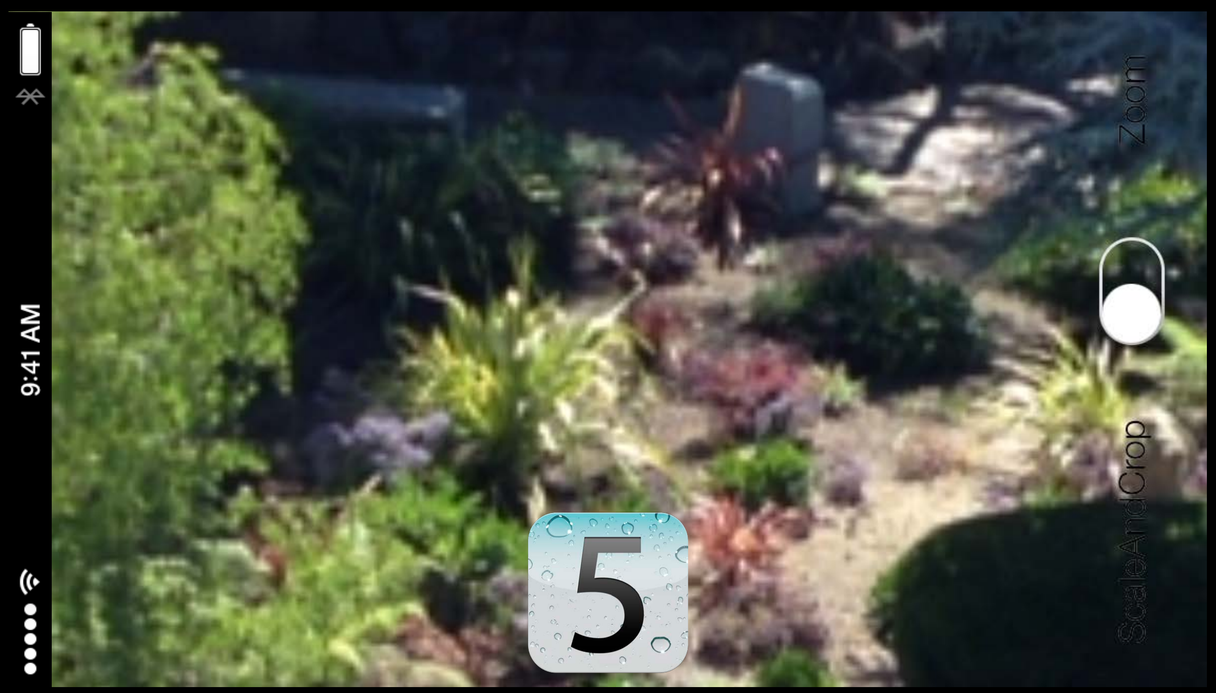


Preview Layer Transformation

`AVCaptureDevice videoZoomFactor`

Zooooooom!

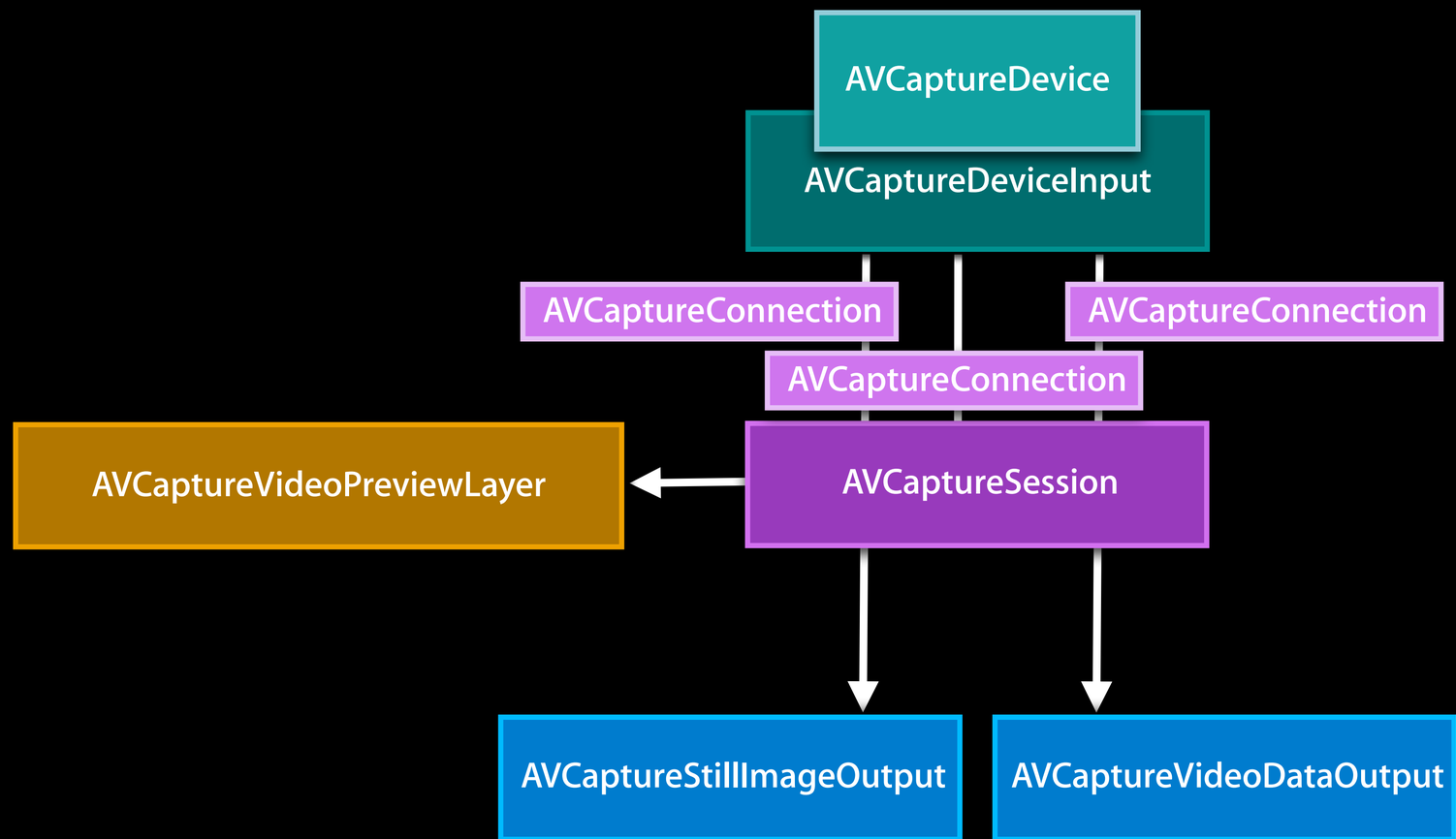
iPhone 5 Preview @ 5x Zoom



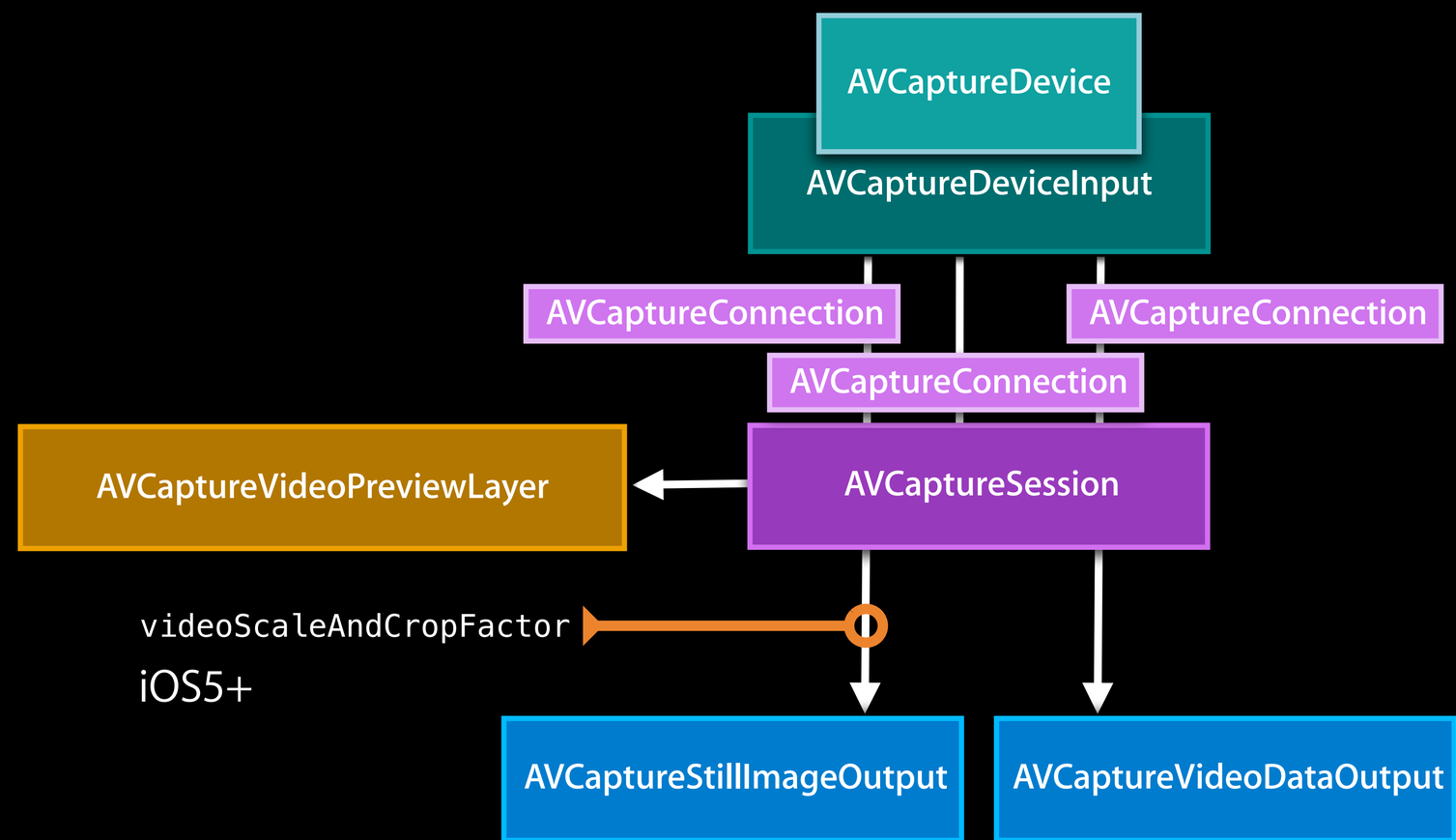
Preview Layer Transformation

`AVCaptureDevice videoZoomFactor`

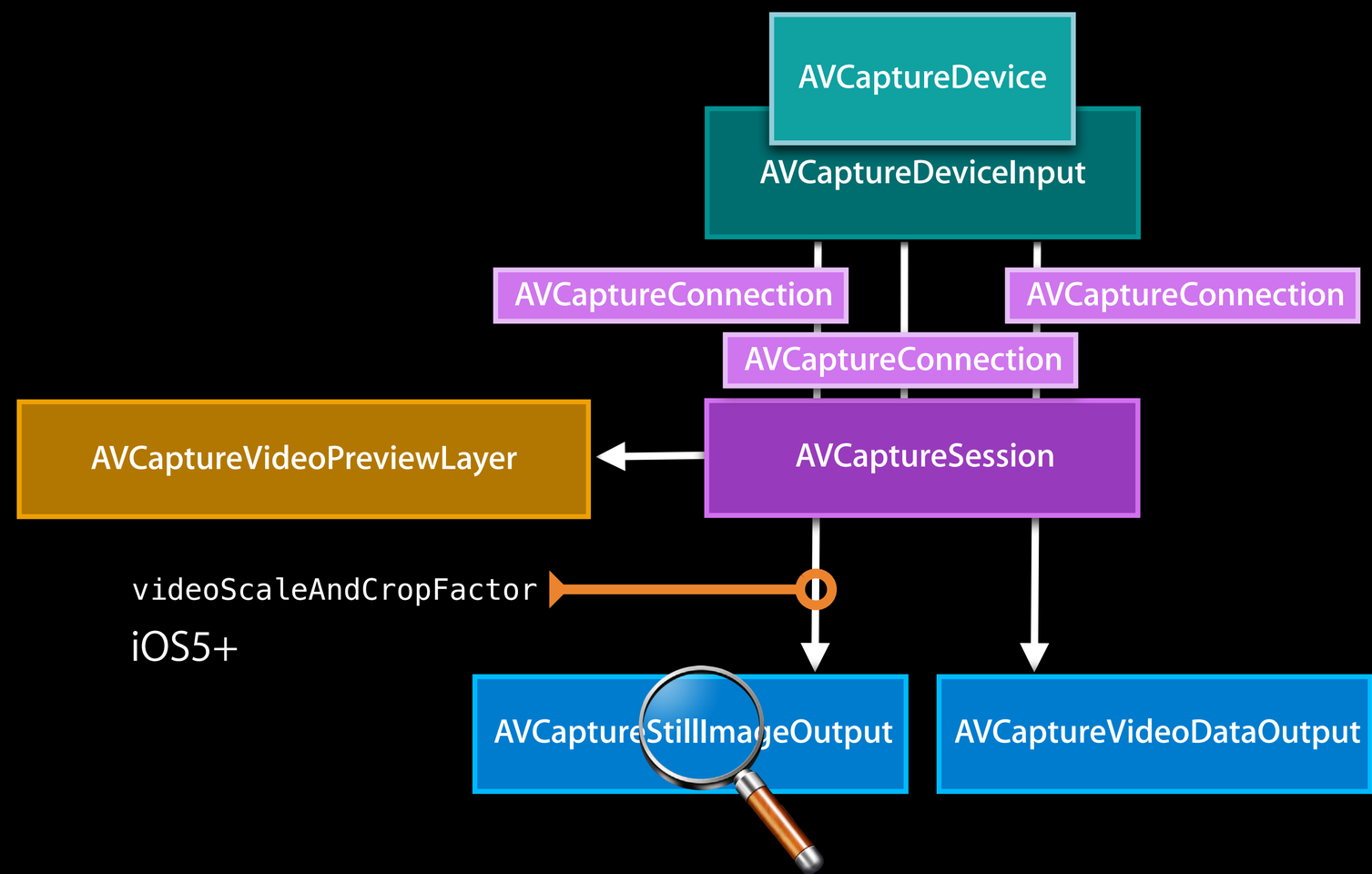
Zoom


AVCaptureDevice
AVCaptureDeviceInput
AVCaptureConnection
AVCaptureConnection
AVCaptureConnection
AVCaptureVideoPreviewLayer
AVCaptureSession
AVCaptureStillImageOutput
AVCaptureVideoDataOutput

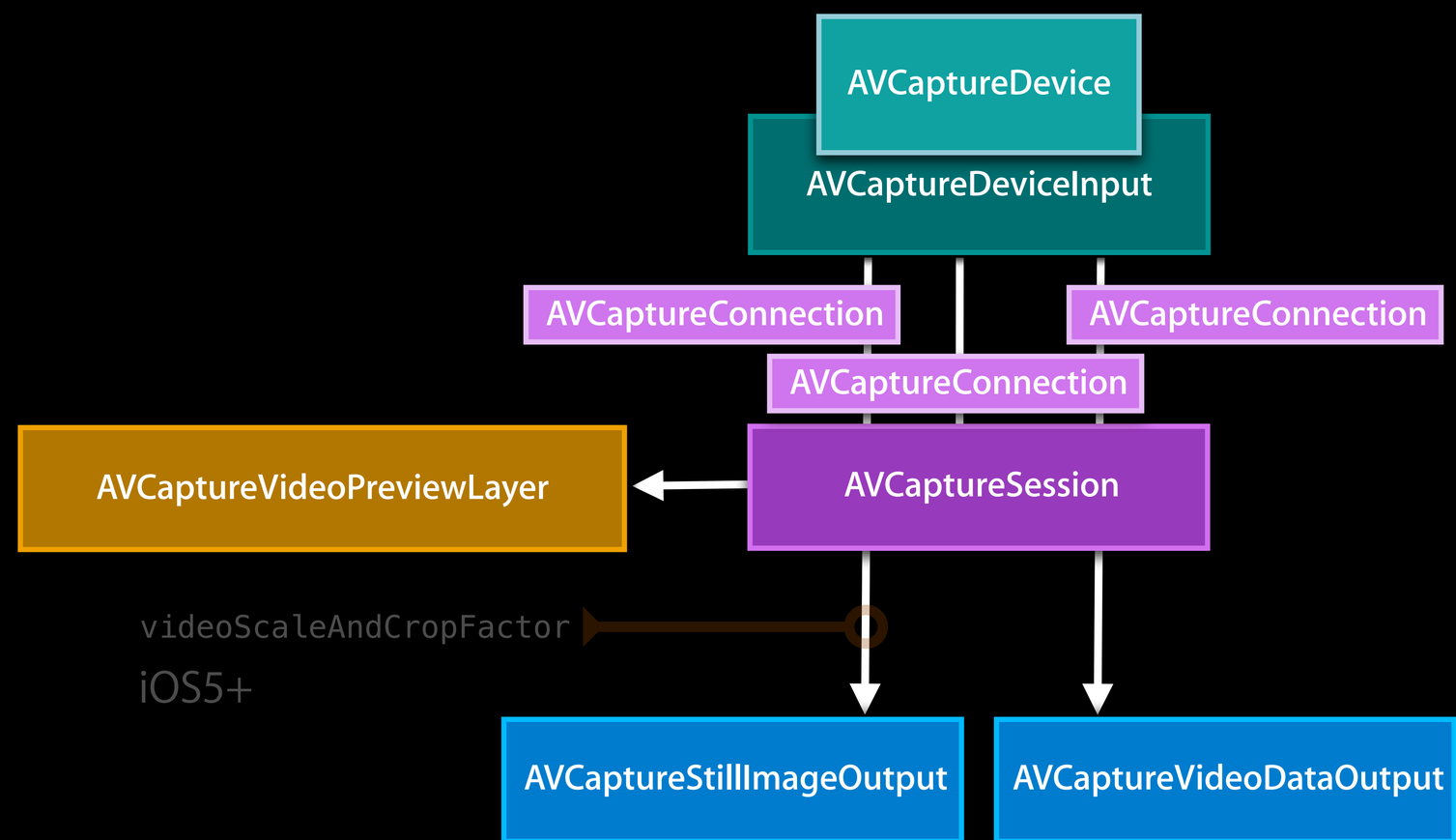
Zoom


AVCaptureDevice
AVCaptureDeviceInput
AVCaptureConnection
AVCaptureConnection
AVCaptureConnection
AVCaptureVideoPreviewLayer
AVCaptureSession
videoScaleAndCropFactor
iOS5+
AVCaptureStillImageOutput
AVCaptureVideoDataOutput

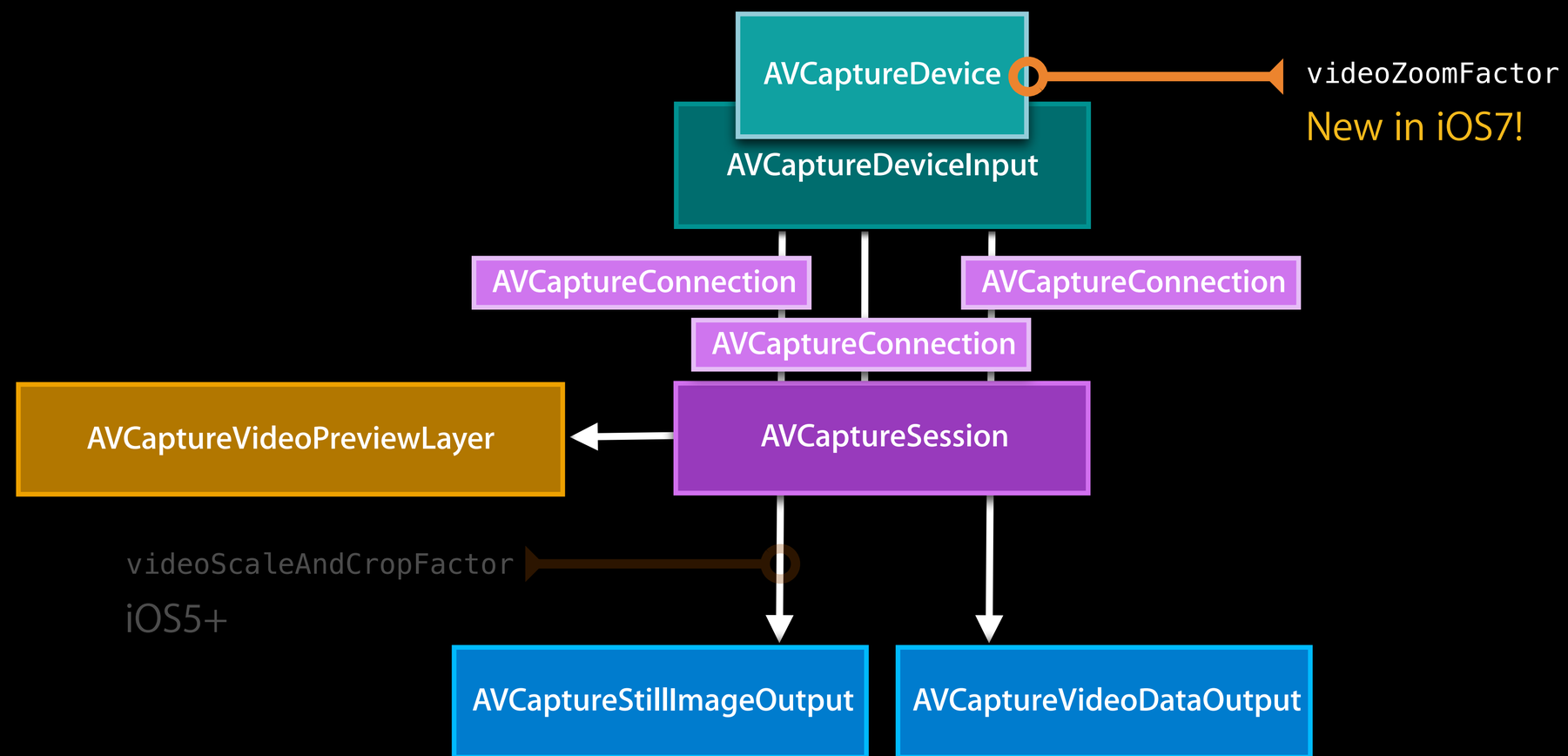
Zoom


AVCaptureDevice
AVCaptureDeviceInput
AVCaptureConnection
AVCaptureConnection
AVCaptureConnection
AVCaptureVideoPreviewLayer
AVCaptureSession
videoScaleAndCropFactor
iOS5+
AVCaptureStillImageOutput
AVCaptureVideoDataOutput

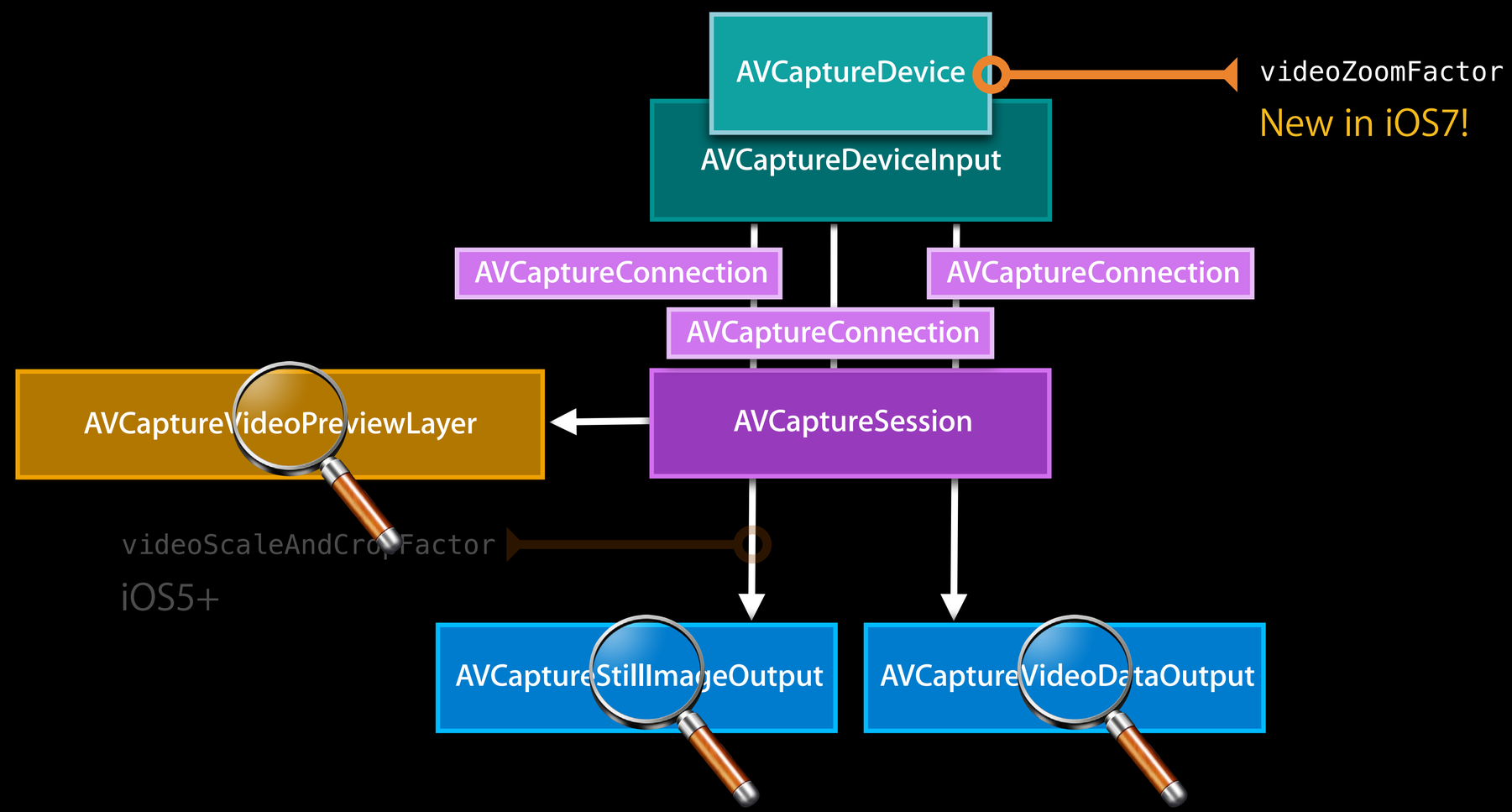
Zoom


AVCaptureDevice
AVCaptureDeviceInput
AVCaptureConnection
AVCaptureConnection
AVCaptureConnection
AVCaptureVideoPreviewLayer
AVCaptureSession
videoScaleAndCropFactor
iOS5+
AVCaptureStillImageOutput
AVCaptureVideoDataOutput

Zoom


AVCaptureDevice
videoZoomFactor
New in iOS7!
AVCaptureDeviceInput
AVCaptureConnection
AVCaptureConnection
AVCaptureConnection
AVCaptureVideoPreviewLayer
AVCaptureSession
videoScaleAndCropFactor
iOS5+
AVCaptureStillImageOutput
AVCaptureVideoDataOutput

Zoom


AVCaptureDevice
videoZoomFactor
New in iOS7!
AVCaptureDeviceInput
AVCaptureConnection
AVCaptureConnection
AVCaptureConnection
AVCaptureVideoPreviewLayer
AVCaptureSession
videoScaleAndCropFactor
iOS5+
AVCaptureStillImageOutput
AVCaptureVideoDataOutput

Zoom

Normal processing

Sensor Pixels

Zoom

Normal processing

Sensor Pixels

Scaled Output Pixels
(Video Resolution)

- Video resolution is smaller than sensor resolution, we must scale down

Zoom

Cropping, not upscaling

Cropped Sensor Pixels

Scaled Output Pixels

- We can retain detail by cropping before the downscale

Zoom

Cropping, not upscaling

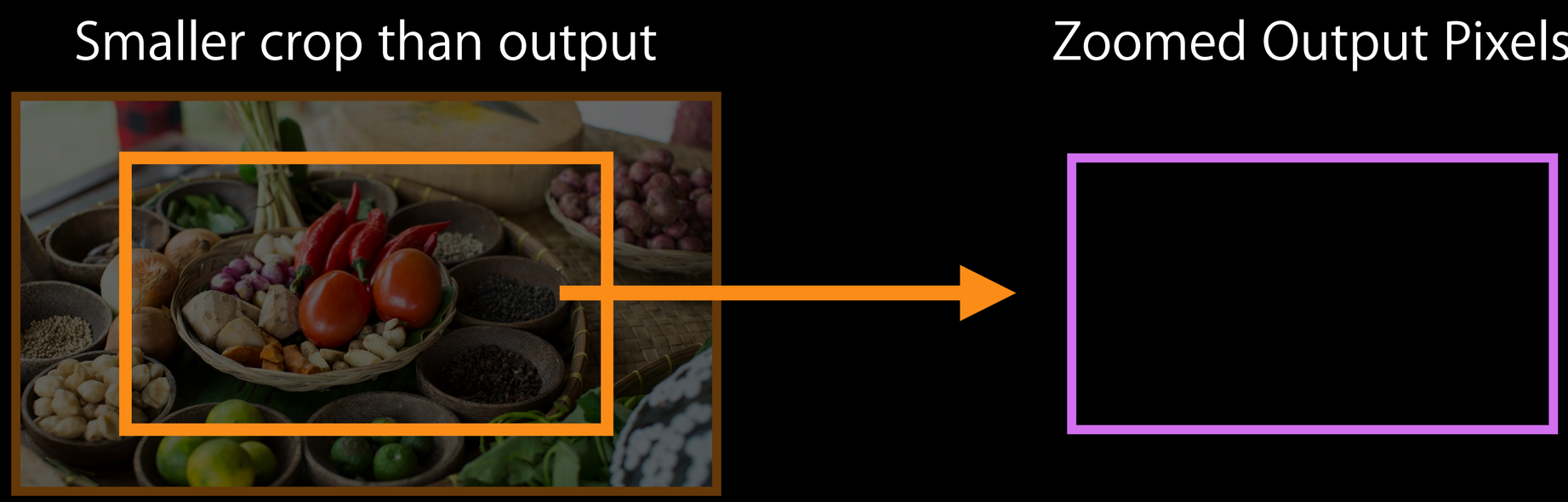
Cropped Sensor Pixels

Zoomed Output Pixels

- We can retain detail by cropping before the downscale

Zoom

It’s not pixels all the way down

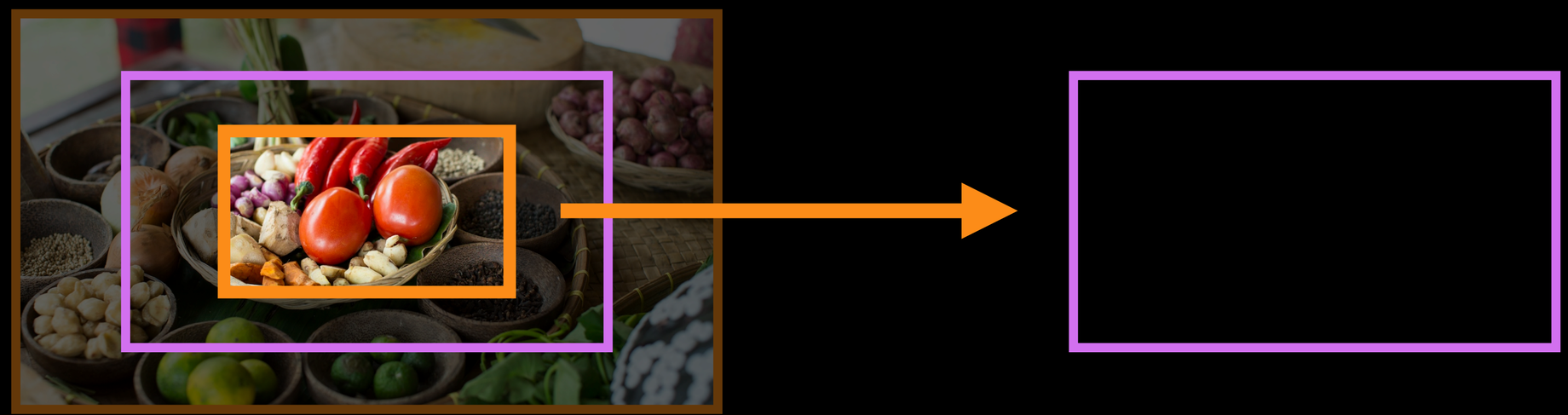


- We can retain detail by cropping before the downscale
- But at some point we do run out of sensor pixels…

Zoom

It’s not pixels all the way down

Smaller crop than output



- We can retain detail by cropping before the downscale
- But at some point we do run out of sensor pixels…

Zoom

It's not pixels all the way down

Smaller crop than output

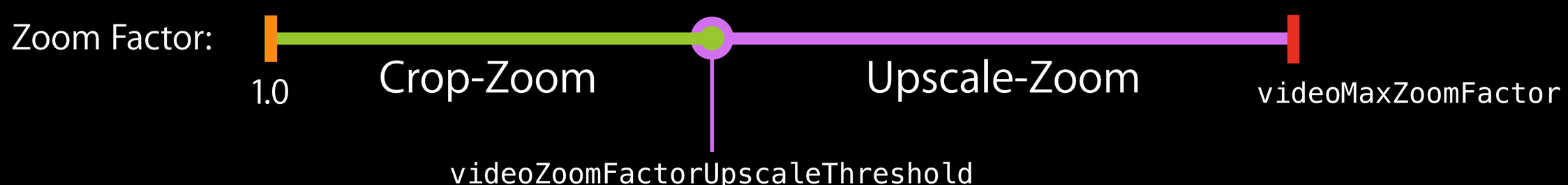
Zoomed Output Pixels
(Upscaled)

- The point at which upscaling begins is:

  `AVCaptureDeviceFormat videoZoomFactorUpscaleThreshold`

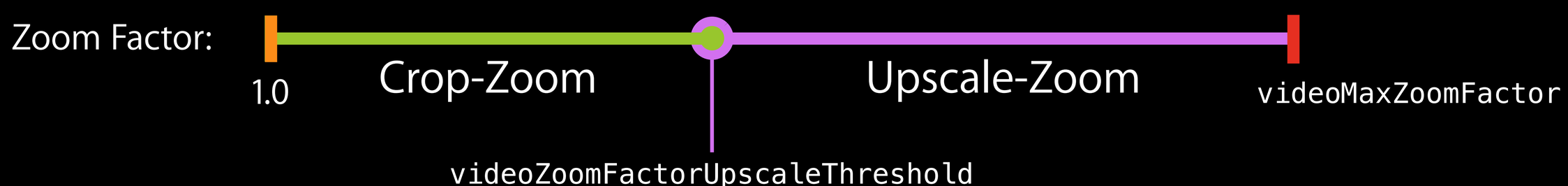
Zoom

Scaling ranges



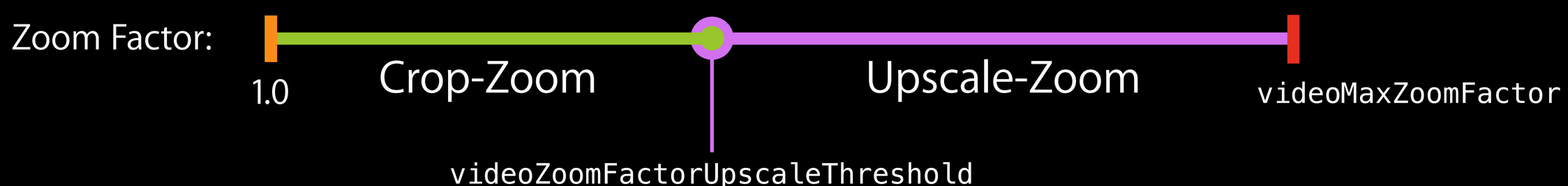
Zoom

Scaling ranges



Zoom

Scaling ranges



Zoom

API

NEW

Zoom

API

- AVCaptureDevice property:

```
@property(nonatomic) CGFloat videoZoomFactor;
```

Zoom

API

- AVCaptureDevice property:

```
@property(nonatomic) CGFloat videoZoomFactor;
```

- Affects all image outputs: still, video, movie file, metadata, and preview

Zoom

API

- AVCaptureDevice property:

```
@property(nonatomic) CGFloat videoZoomFactor;
```

- Affects all image outputs: still, video, movie file, metadata, and preview
- Up to limit defined by the AVCaptureDeviceFormat property

```
@property(nonatomic, readonly) CGFloat videoMaxZoomFactor;
```

  - e.g., `device.activeFormat.videoMaxZoomFactor`

Zoom

API

- AVCaptureDevice property:

```
@property(nonatomic) CGFloat videoZoomFactor;
```

- Affects all image outputs: still, video, movie file, metadata, and preview
- Up to limit defined by the AVCaptureDeviceFormat property

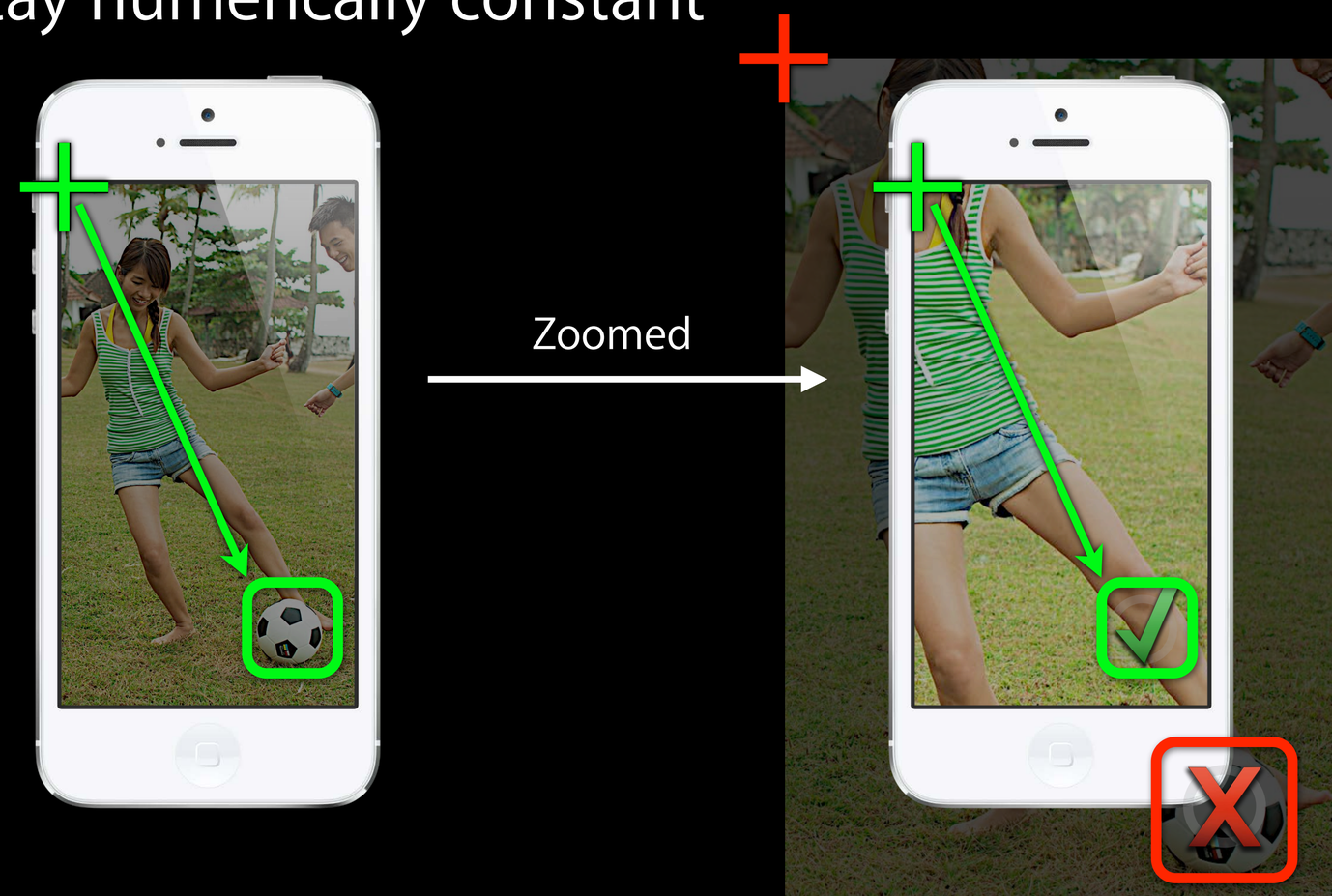
```
@property(nonatomic, readonly) CGFloat videoMaxZoomFactor;
```

  - e.g., `device.activeFormat.videoMaxZoomFactor`
- Device coordinates (e.g., focus points of interest) stay relative to the visible image frame

Zoom

Coordinates relative to visible frame

- Faces and other metadata overlay relative to current frame
  - `AVCaptureVideoPreviewLayer -transformedMetadataObjectForMetadataObject:`
- Focus points stay numerically constant



Zoom

NEW

Video Zoom

NEW

- Zoom is a dramatic cinematic effect

Video Zoom

- Zoom is a dramatic cinematic effect
- Ensure smooth zooms by using ramp rather than incrementing factor

```
– (void)rampToVideoZoomFactor:(CGFloat)factor withRate:(float)rate;
```

Video Zoom

- Zoom is a dramatic cinematic effect
- Ensure smooth zooms by using ramp rather than incrementing factor

```
– (void)rampToVideoZoomFactor:(CGFloat)factor withRate:(float)rate;
```

- AVFoundation will consistently increment zoom factor on each frame

Video Zoom

- Zoom is a dramatic cinematic effect
- Ensure smooth zooms by using ramp rather than incrementing factor

```
– (void)rampToVideoZoomFactor:(CGFloat)factor withRate:(float)rate;
```

- AVFoundation will consistently increment zoom factor on each frame
- Call again to change the current ramp, or:

```
– (void)cancelVideoZoomRamp
```

Video Zoom

- Zoom is a dramatic cinematic effect
- Ensure smooth zooms by using ramp rather than incrementing factor

```
– (void)rampToVideoZoomFactor:(CGFloat)factor withRate:(float)rate;
```

- AVFoundation will consistently increment zoom factor on each frame
- Call again to change the current ramp, or:

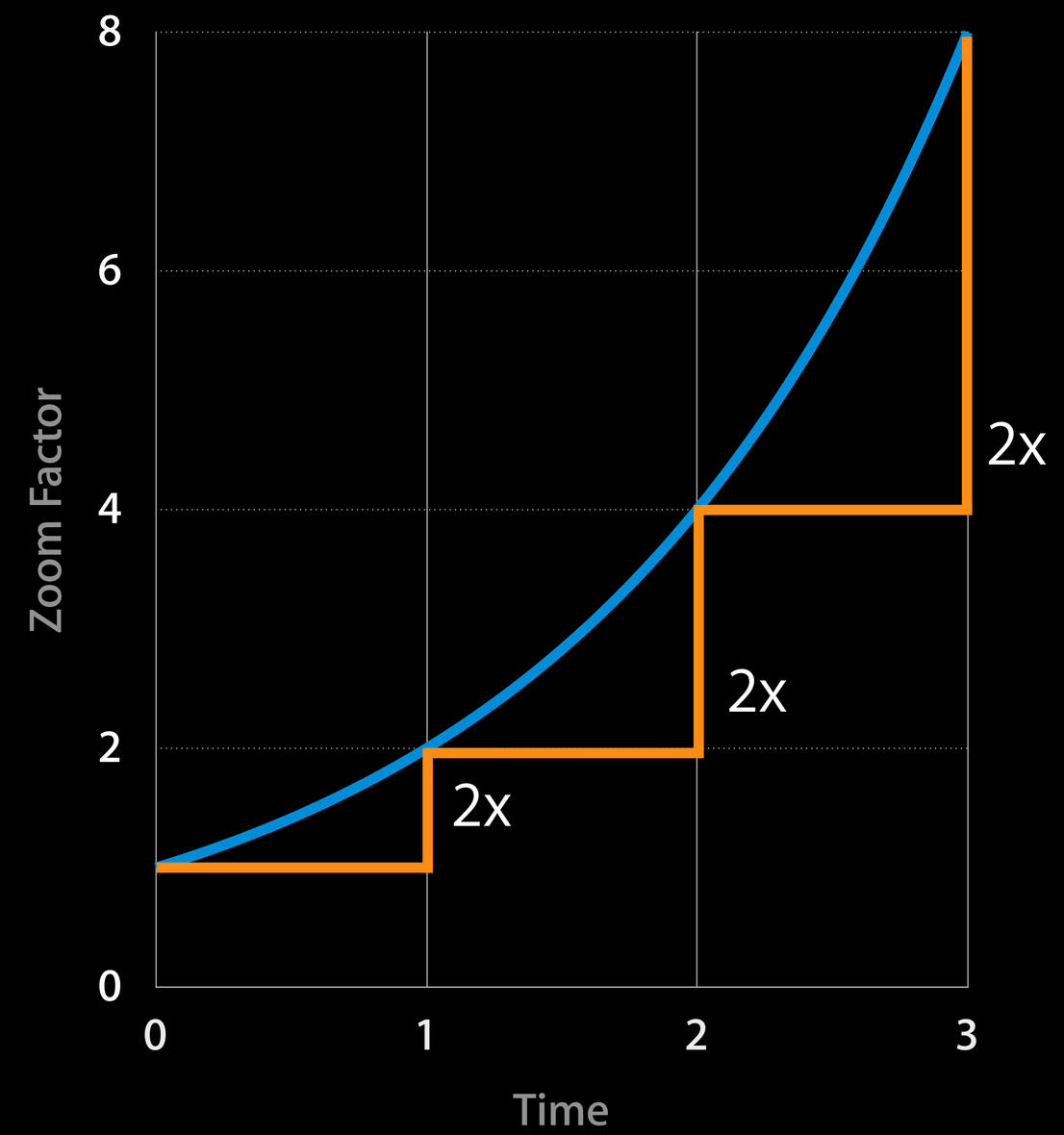
```
– (void)cancelVideoZoomRamp
```

- Changes in rate are smoothed by an internal acceleration factor

Video Zoom

Setting the rate

- Rate is specified in powers of two per second
  - Moving from 1x to 2x is visually equivalent to moving from 4x to 8x
  - "1" means double or halve the magnification every second (direction is inferred from target factor)
  - Comfortable values are around 1–3

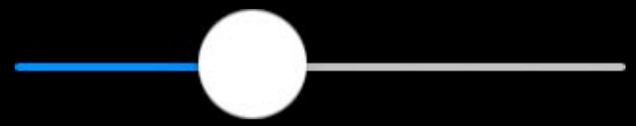


Demo

SoZoomy

Zoom

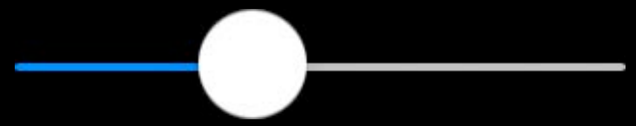
Example: Direct control for zoom



```
[device lockForConfiguration: &err];
// Convert a 0-1 linear slider value to 1-max exponential
device.videoZoomFactor = pow( maxZoom, normalizedTarget );
[device unlockForConfiguration];
```

Zoom

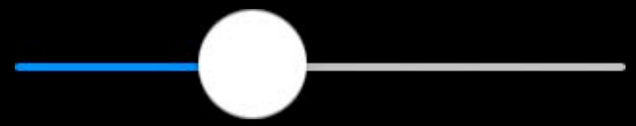
Example: Direct control for zoom



```
[device lockForConfiguration: &err];
// Convert a 0-1 linear slider value to 1-max exponential
device.videoZoomFactor = pow( maxZoom, normalizedTarget );
[device unlockForConfiguration];
```

Zoom

Example: Direct control for zoom



```
[device lockForConfiguration: &err];
// Convert a 0-1 linear slider value to 1-max exponential
device.videoZoomFactor = pow( maxZoom, normalizedTarget );
[device unlockForConfiguration];
```

Zoom

Example: Speed control for zoom

```
[device lockForConfiguration: &err];
CGFloat targetLimit = ( isZoomIn ) ? maxZoom : 1.0;
[device rampToVideoZoomFactor: targetLimit withRate: 2.0 ];
[device unlockForConfiguration];

[device lockForConfiguration: &err];
[device cancelVideoZoomRamp];
[device unlockForConfiguration];
```

Zoom

Example: Speed control for zoom

```
[device lockForConfiguration: &err];
CGFloat targetLimit = ( isZoomIn ) ? maxZoom : 1.0;
[device rampToVideoZoomFactor: targetLimit withRate: 2.0 ];
[device unlockForConfiguration];

[device lockForConfiguration: &err];
[device cancelVideoZoomRamp];
[device unlockForConfiguration];
```

Zoom

Example: Speed control for zoom

```
[device lockForConfiguration: &err];
CGFloat targetLimit = ( isZoomIn ) ? maxZoom : 1.0;
[device rampToVideoZoomFactor: targetLimit withRate: 2.0 ];
[device unlockForConfiguration];

[device lockForConfiguration: &err];
[device cancelVideoZoomRamp];
[device unlockForConfiguration];
```

Zoom

Example: Speed control for zoom

```
[device lockForConfiguration: &err];
CGFloat targetLimit = ( isZoomIn ) ? maxZoom : 1.0;
[device rampToVideoZoomFactor: targetLimit withRate: 2.0 ];
[device unlockForConfiguration];

[device lockForConfiguration: &err];
[device cancelVideoZoomRamp];
[device unlockForConfiguration];
```

Zoom

Feature summary

	AVCaptureConnection videoScaleAndCropFactor	AVCaptureDevice videoZoomFactor
Still Image Output	✓	✓
Video Preview Layer		✓
Video Data Output		✓
Movie File Output		✓
Metadata Output		✓
Set Zoom Factor	✓	✓
Set Zoom Rate		✓

Zoom

Platform support

	`AVCaptureDevice videoZoomFactor`	Back Camera 1080p Upscaling Threshold
iPhone 5	✓	1.45
iPod touch (5th Generation)	✓	1.23

New in iOS 7

- 60 FPS Support
- Video zoom
- Machine Readable Code detection
- Focus enhancements
- Integration with application AudioSession

Machine Readable Code Detection

Rob Simutis
Core Media Engineering

Machine Readable Codes

Machine Readable Codes

Machine Readable Codes

Machine Readable Codes

- Real-time detection of 1-D and 2-D codes

Machine Readable Codes

- Real-time detection of 1-D and 2-D codes
- Up to four codes

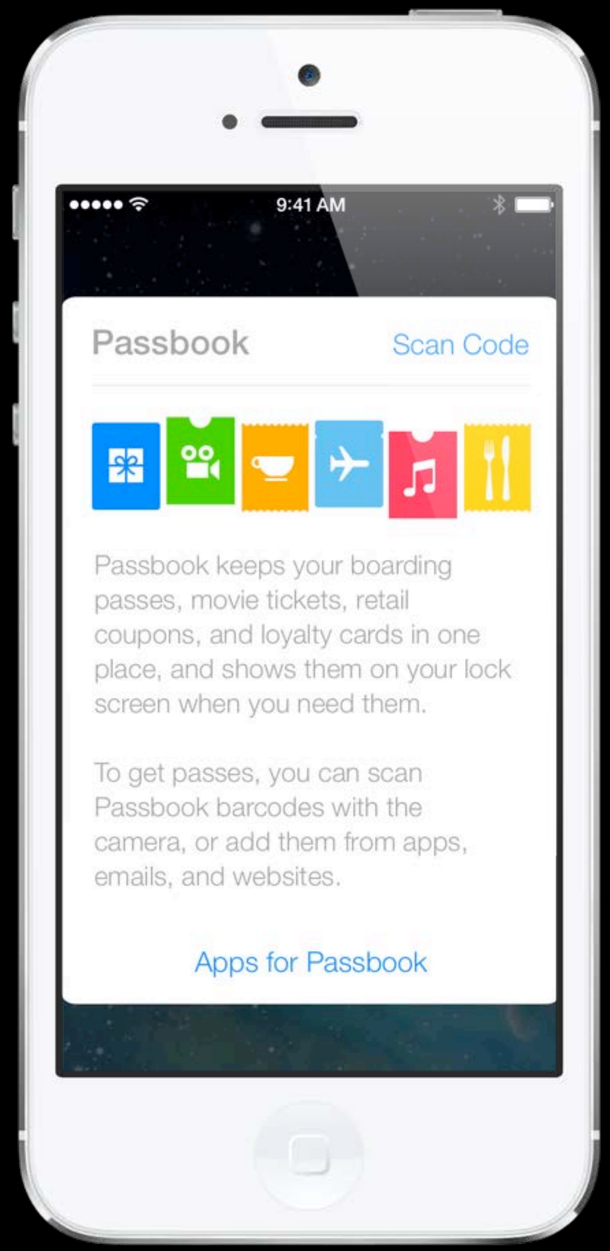
Machine Readable Codes

- Real-time detection of 1-D and 2-D codes
- Up to four codes
- Front and back cameras

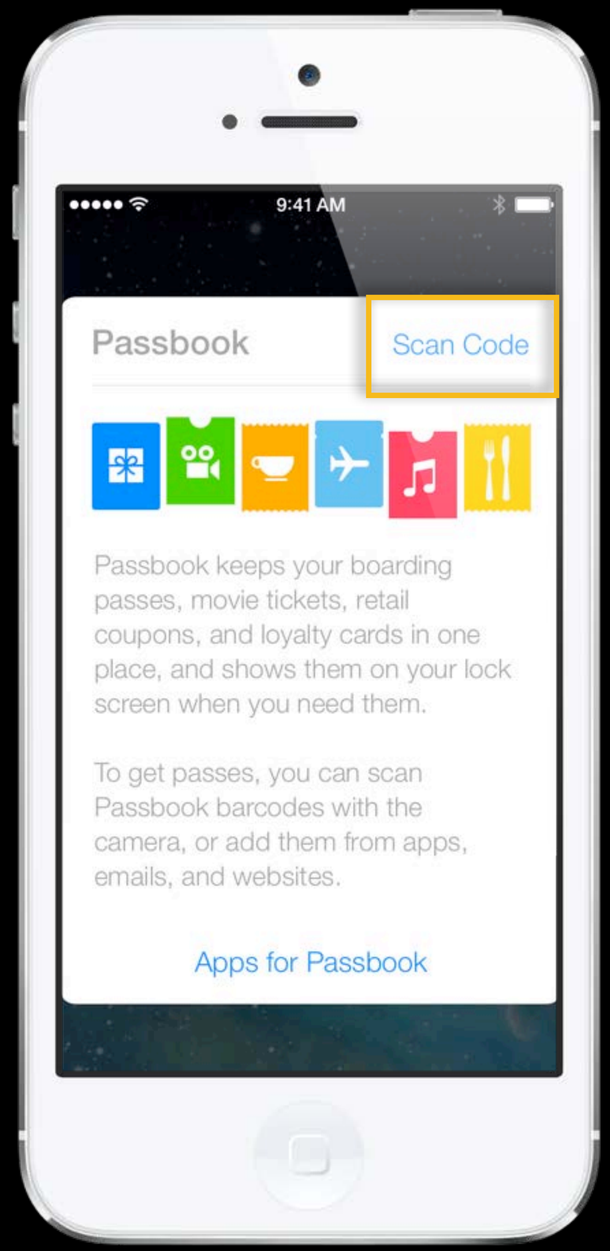
Machine Readable Codes

- Real-time detection of 1-D and 2-D codes
- Up to four codes
- Front and back cameras
- All iOS 7 hardware

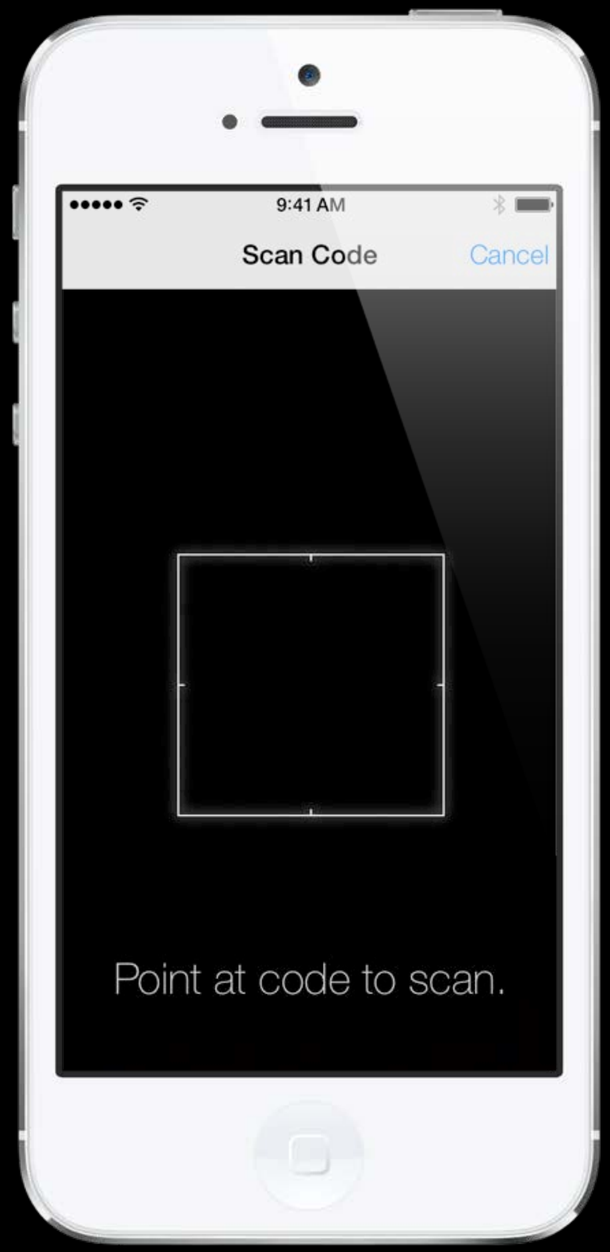
Passbook


9:41 AM
Passbook
Scan Code
Passbook keeps your boarding passes, movie tickets, retail coupons, and loyalty cards in one place, and shows them on your lock screen when you need them.
To get passes, you can scan Passbook barcodes with the camera, or add them from apps, emails, and websites.
Apps for Passbook

Passbook


9:41 AM
Passbook
Scan Code
Passbook keeps your boarding passes, movie tickets, retail coupons, and loyalty cards in one place, and shows them on your lock screen when you need them.
To get passes, you can scan Passbook barcodes with the camera, or add them from apps, emails, and websites.
Apps for Passbook

Passbook



Supported Types (Symbologies)

1-D

Supported Types (Symbologies)

1-D

UPC-E	8 85909 47529 2
EAN-8, EAN-13	1111 1115 1 111111 111116
Code 39 (with and without checksum)	1 2 3 4 5 6 7 8
Code 93	111 South Fakington Avenue
Code 128	111 South Fakington Avenue

Supported Types (Symbologies)

2-D

Supported Types (Symbologies)

2-D

PDF417	
QR	
Aztec	

Demo

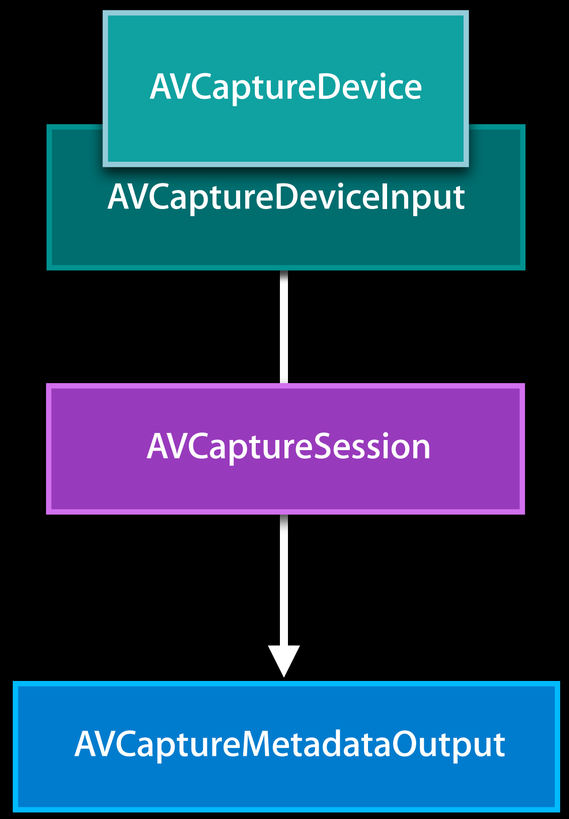
QRchestra

Programming Model

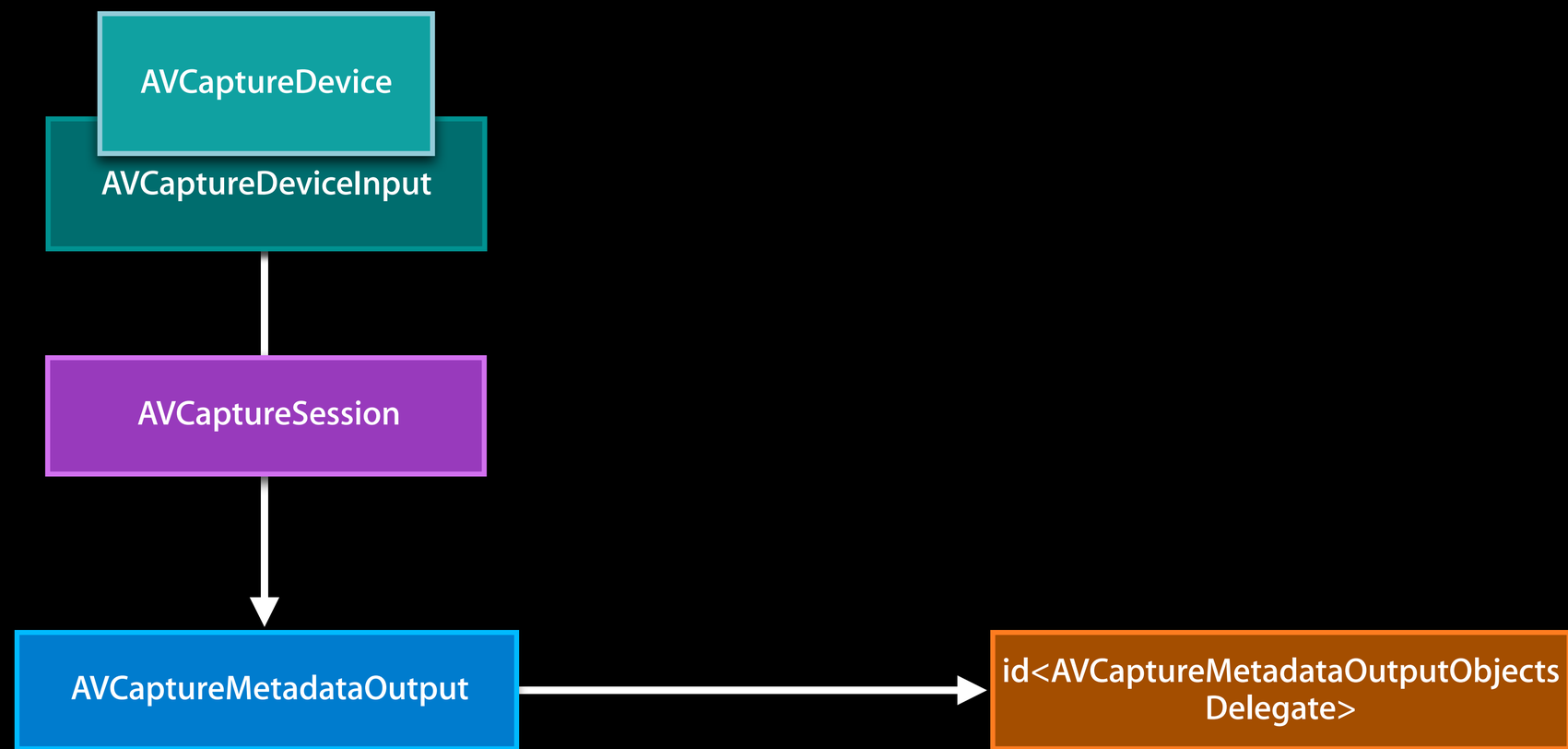
Programming Model

AVCaptureMetadataOutput

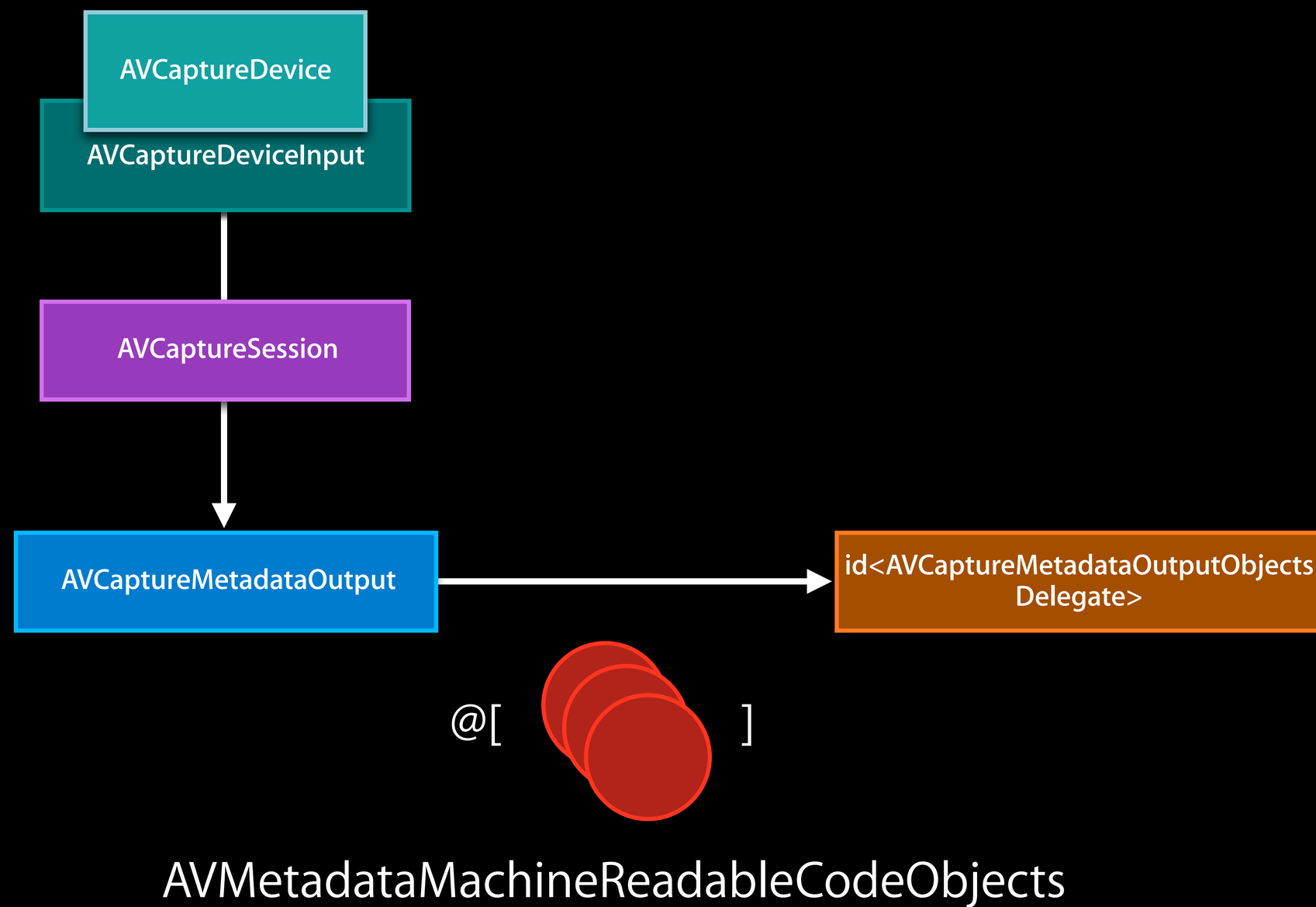
Programming Model



Programming Model



Programming Model



Metadata Output

Setup

```
-(void)setup:(AVCaptureSession *)session {

    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];

    [session addOutput:metadataOutput];
    [metadataOutput release];

    id<AVCaptureMetadataOutputObjectsDelegate> metadataDelegate = ...
    dispatch_queue queue = ...

    [metadataOutput setMetadataObjectsDelegate:metadataDelegate queue:queue];

    NSArray *metadataTypes = @[ AVMetadataObjectTypeAztecCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];

}
```

Metadata Output

Setup

```
-(void)setup:(AVCaptureSession *)session {

    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];

    [session addOutput:metadataOutput];
    [metadataOutput release];

    id<AVCaptureMetadataOutputObjectsDelegate> metadataDelegate = ...
    dispatch_queue queue = ...

    [metadataOutput setMetadataObjectsDelegate:metadataDelegate queue:queue];

    NSArray *metadataTypes = @[ AVMetadataObjectTypeAztecCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];

}
```

Metadata Output

Setup

```
-(void)setup:(AVCaptureSession *)session {

    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];

    [session addOutput:metadataOutput];
    [metadataOutput release];

    id<AVCaptureMetadataOutputObjectsDelegate> metadataDelegate = ...
    dispatch_queue queue = ...

    [metadataOutput setMetadataObjectsDelegate:metadataDelegate queue:queue];

    NSArray *metadataTypes = @[ AVMetadataObjectTypeAztecCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];

}
```

Metadata Output

Setup

```
-(void)setup:(AVCaptureSession *)session {

    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];

    [session addOutput:metadataOutput];
    [metadataOutput release];

    id<AVCaptureMetadataOutputObjectsDelegate> metadataDelegate = ...
    dispatch_queue queue = ...

    [metadataOutput setMetadataObjectsDelegate:metadataDelegate queue:queue];

    NSArray *metadataTypes = @[ AVMetadataObjectTypeAztecCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];

}
```

Metadata Output

Setup

```
-(void)setup:(AVCaptureSession *)session {

    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];

    [session addOutput:metadataOutput];
    [metadataOutput release];

    id<AVCaptureMetadataOutputObjectsDelegate> metadataDelegate = ...
    dispatch_queue queue = ...

    [metadataOutput setMetadataObjectsDelegate:metadataDelegate queue:queue];

    NSArray *metadataTypes = @[ AVMetadataObjectTypeAztecCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];

}
```

Metadata Output

Setup

```
-(void)setup:(AVCaptureSession *)session {

    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];

    [session addOutput:metadataOutput];
    [metadataOutput release];

    id<AVCaptureMetadataOutputObjectsDelegate> metadataDelegate = ...
    dispatch_queue queue = ...

    [metadataOutput setMetadataObjectsDelegate:metadataDelegate queue:queue];

    NSArray *metadataTypes = @[ AVMetadataObjectTypeAztecCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];

}
```

Metadata Output Refresher

Metadata output delegate

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput
        didOutputMetadataObjects:(NSArray *)metadataObjects
        fromConnection:(AVCaptureConnection *)connection
{
   for ( AVMetadataObject *object in metadataObjects ) {

      if ( [[object class] isKindOfClass:
                  [AVMetadataMachineReadableCodeObject class]] ) {

            // Do interesting things with this barcode
      }
   }
}
```

Metadata Output Refresher

Metadata output delegate

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput
        didOutputMetadataObjects:(NSArray *)metadataObjects
        fromConnection:(AVCaptureConnection *)connection
{
   for ( AVMetadataObject *object in metadataObjects ) {

      if ( [[object class] isKindOfClass:
                  [AVMetadataMachineReadableCodeObject class]] ) {

         // Do interesting things with this barcode
      }
   }
}
```

Machine Readable Codes

`AVMetadataMachineReadableCodeObject`

Machine Readable Codes

AVMetadataMachineReadableCodeObject

```
@property(readonly) CGRect bounds;
```

Machine Readable Codes

AVMetadataMachineReadableCodeObject

```
@property(readonly) CGRect bounds;
@property(readonly) NSArray *corners;
```

Machine Readable Codes

AVMetadataMachineReadableCodeObject

```
@property(readonly) CGRect bounds;
@property(readonly) NSArray *corners;
@property(readonly) NSString *type;
```

Machine Readable Codes

AVMetadataMachineReadableCodeObject

```
@property(readonly) CGRect bounds;
@property(readonly) NSArray *corners;
@property(readonly) NSString *type;
@property(readonly) NSString *stringValue;
```

- best-effort attempt to decode payload
- may return nil if decoding fails

bounds vs. corners

bounds vs. corners

bounds vs. corners

bounds vs. corners

bounds vs. corners

bounds
rectangle, axis-aligned

bounds vs. corners

bounds
rectangle, axis-aligned

corners
points, may be non-rectangular

bounds vs. corners

bounds
rectangle, axis-aligned

corners
points, may be non-rectangular

Performance Considerations

Performance Considerations

- Enable just the codes you are interested in finding

Performance Considerations

- Enable just the codes you are interested in finding
- Use new `AVCaptureMetadataOutput.rectOfInterest` property

Performance Considerations

- Enable just the codes you are interested in finding
- Use new `AVCaptureMetadataOutput.rectOfInterest` property
- Pick the right session preset for your use case
  - Start with `AVCaptureSessionPreset640x480`

Performance Considerations

- Enable just the codes you are interested in finding
- Use new `AVCaptureMetadataOutput.rectOfInterest` property
- Pick the right session preset for your use case
  - Start with `AVCaptureSessionPreset640x480`
- Consider new auto-focus range restriction API

Performance Considerations

- Enable just the codes you are interested in finding
- Use new `AVCaptureMetadataOutput.rectOfInterest` property
- Pick the right session preset for your use case
  - Start with `AVCaptureSessionPreset640x480`
- Consider new auto-focus range restriction API
- Use new zoom APIs

Performance Considerations

Request the codes you want

Performance Considerations

Request the codes you want

- `AVCaptureMetadataOutput.metadataObjectTypes`

Performance Considerations

Request the codes you want

- `AVCaptureMetadataOutput.metadataObjectTypes`
- iOS 6 behavior
  - Defaults to all metadata types being enabled

Performance Considerations

Request the codes you want

- `AVCaptureMetadataOutput.metadataObjectTypes`
- iOS 6 behavior
  - Defaults to all metadata types being enabled
- iOS 7 behavior
  - Apps linked on or after iOS 7 must opt-in to desired metadata types
  - Apps linked before iOS 7 get the old behavior (faces only)

Performance Considerations

AVCaptureMetadataOutput.metadataObjectTypes

```
-(void)setup:(AVCaptureSession *)session {
    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];
   //...

    NSArray *metadataTypes = [metadataOutput availableMetadataObjectTypes];
    [metadataOutput setMetadataObjectTypes:metadataTypes];
}
```

Performance Considerations

AVCaptureMetadataOutput.metadataObjectTypes

```
-(void)setup:(AVCaptureSession *)session {
    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];
    //...

    NSArray *metadataTypes = [metadataOutput availableMetadataObjectTypes];
    [metadataOutput setMetadataObjectTypes:metadataTypes];
}
```

Performance Considerations

AVCaptureMetadataOutput.metadataObjectTypes

```
-(void)setup:(AVCaptureSession *)session {
    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];
    //...

    NSArray *metadataTypes = [metadataOutput availableMetadataObjectTypes];
    [metadataOutput setMetadataObjectTypes:metadataTypes];
}
```

Performance Considerations

AVCaptureMetadataOutput.metadataObjectTypes

```
-(void)setup:(AVCaptureSession *)session {
    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];
   //...

    NSArray *metadataTypes = @[ AVMetadataObjectTypeFace,
                                AVMetadataObjectTypeQRCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];
}
```

Performance Considerations

AVCaptureMetadataOutput.metadataObjectTypes

```
-(void)setup:(AVCaptureSession *)session {
    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];
   //...

    NSArray *metadataTypes = @[ AVMetadataObjectTypeFace,
                                AVMetadataObjectTypeQRCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];
}
```

Performance Considerations

AVCaptureMetadataOutput.metadataObjectTypes

```
-(void)setup:(AVCaptureSession *)session {
    AVCaptureMetadataOutput *metadataOutput =
                                    [[AVCaptureMetadataOutput alloc] init];
   //...

    NSArray *metadataTypes = @[ AVMetadataObjectTypeFace,
                                AVMetadataObjectTypeQRCode ];
    [metadataOutput setMetadataObjectTypes:metadataTypes];
}
```

Performance Considerations

Limit your search area

Performance Considerations

Limit your search area

- `AVCaptureMetadataOutput.rectOfInterest`

Performance Considerations

Limit your search area

- `AVCaptureMetadataOutput.rectOfInterest`
- Decreases time to detect metadata

Performance Considerations

Limit your search area

- `AVCaptureMetadataOutput.rectOfInterest`
- Decreases time to detect metadata
- Faces as well as barcodes

Performance Considerations

Limit your search area

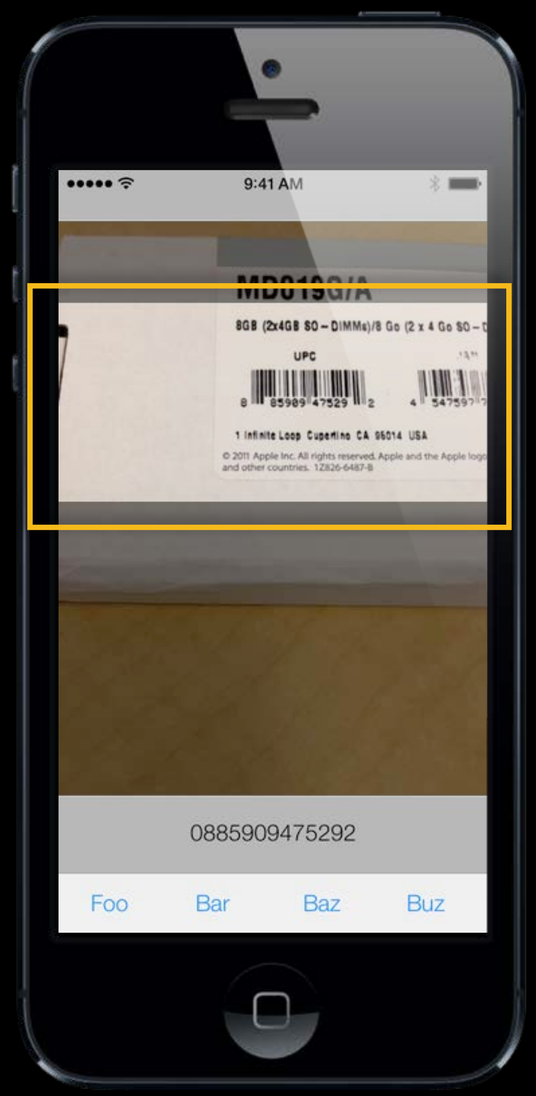
- `AVCaptureMetadataOutput.rectOfInterest`
- Decreases time to detect metadata
- Faces as well as barcodes
- Conversion methods translate between coordinate spaces

Coordinate Spaces

AVCaptureVideoPreviewLayer

Coordinate Spaces

AVCaptureVideoPreviewLayer



Coordinate Spaces

AVCaptureVideoPreviewLayer

Coordinate Spaces

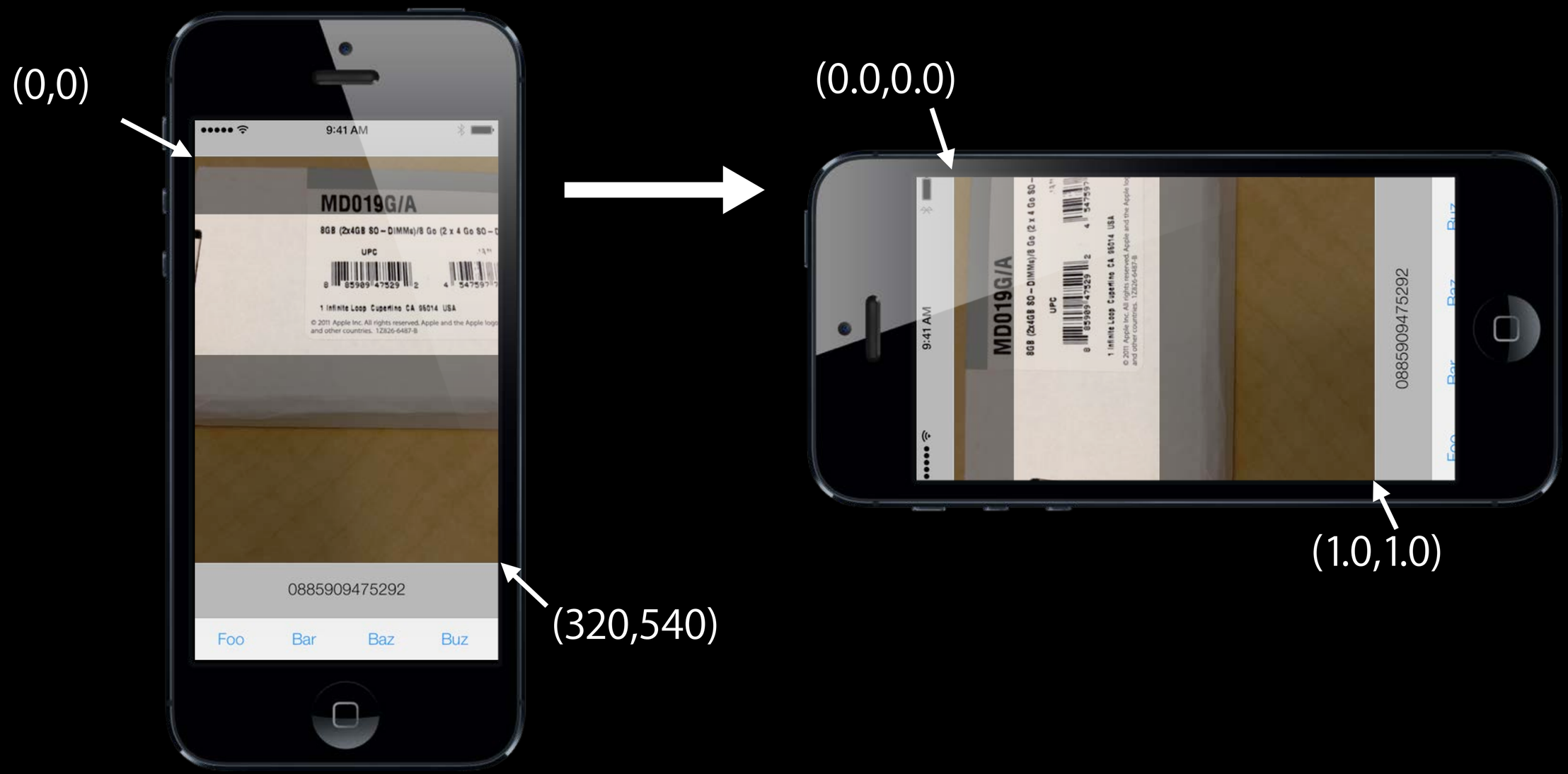
AVCaptureVideoPreviewLayer

AVCaptureMetadataOutput

Coordinate Spaces

AVCaptureVideoPreviewLayer

AVCaptureMetadataOutput



Coordinate Spaces

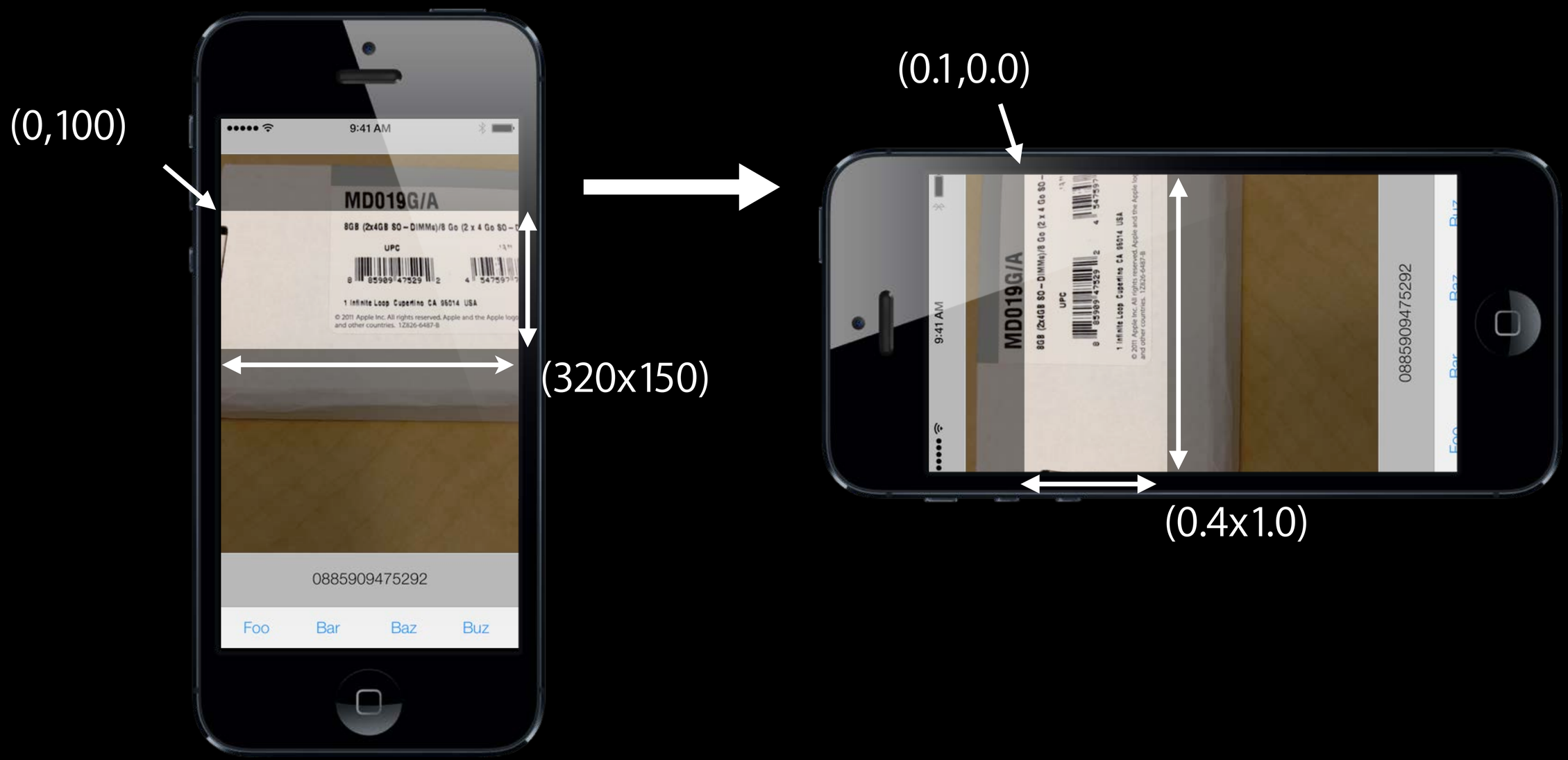
AVCaptureVideoPreviewLayer

AVCaptureMetadataOutput

Coordinate Spaces

AVCaptureVideoPreviewLayer

AVCaptureMetadataOutput



Coordinate Spaces

`AVCaptureMetadataOutput.rectOfInterest`

Coordinate Spaces

AVCaptureMetadataOutput.rectOfInterest

AVCaptureVideoPreviewLayer → AVCaptureMetadataOutput

```
-[AVCaptureVideoPreviewLayer metadataOutputRectOfInterestForRect:]
```

Coordinate Spaces

`AVCaptureMetadataOutput.rectOfInterest`

`AVCaptureVideoPreviewLayer` → `AVCaptureMetadataOutput`

`-[AVCaptureVideoPreviewLayer metadataOutputRectOfInterestForRect:]`

`AVCaptureVideoPreviewLayer` ← `AVCaptureMetadataOutput`

`-[AVCaptureVideoPreviewLayer rectForMetadataOutputRectOfInterest:]`

Coordinate Spaces

AVCaptureMetadataOutput.rectOfInterest

```
CGRect bounds = [previewLayer bounds];

CGRect previewRect = CGRectMake(0, bounds.origin.y+100,
                                bounds.size.width, 150);

CGRect metadataRect = [previewLayer
                  metadataOutputRectOfInterestForRect:rectOfInterest];

[metadataOutput setRectOfInterest:metadataRect];
```

Coordinate Spaces

AVCaptureMetadataOutput.rectOfInterest

```
CGRect bounds = [previewLayer bounds];

CGRect previewRect = CGRectMake(0, bounds.origin.y+100,
                                bounds.size.width, 150);

CGRect metadataRect = [previewLayer
                  metadataOutputRectOfInterestForRect:rectOfInterest];

[metadataOutput setRectOfInterest:metadataRect];
```

Coordinate Spaces

AVCaptureMetadataOutput.rectOfInterest

```
CGRect bounds = [previewLayer bounds];

CGRect previewRect = CGRectMake(0, bounds.origin.y+100,
                                bounds.size.width, 150);

CGRect metadataRect = [previewLayer
                  metadataOutputRectOfInterestForRect:rectOfInterest];

[metadataOutput setRectOfInterest:metadataRect];
```

Coordinate Spaces

AVCaptureMetadataOutput.rectOfInterest

```
CGRect bounds = [previewLayer bounds];

CGRect previewRect = CGRectMake(0, bounds.origin.y+100,
                                bounds.size.width, 150);

CGRect metadataRect = [previewLayer
                  metadataOutputRectOfInterestForRect:rectOfInterest];

[metadataOutput setRectOfInterest:metadataRect];
```

Coordinate Spaces

AVCaptureMetadataOutput.rectOfInterest

```
CGRect bounds = [previewLayer bounds];

CGRect previewRect = CGRectMake(0, bounds.origin.y+100,
                                bounds.size.width, 150);

CGRect metadataRect = [previewLayer
                  metadataOutputRectOfInterestForRect:rectOfInterest];

[metadataOutput setRectOfInterest:metadataRect];
```

Machine Readable Codes

Supported platforms

iPhone 4	✓
iPhone 4S	✓
iPhone 5	✓
iPod touch (fifth generation)	✓
iPad mini	✓
iPad	✓

Machine Readable Codes

Supported platforms

iPhone 4	✓
iPhone 4S	✓
iPhone 5	✓
iPod touch (fifth generation)	✓
iPad mini	✓
iPad	✓

ALL iOS 7 DEVICES

New in iOS 7

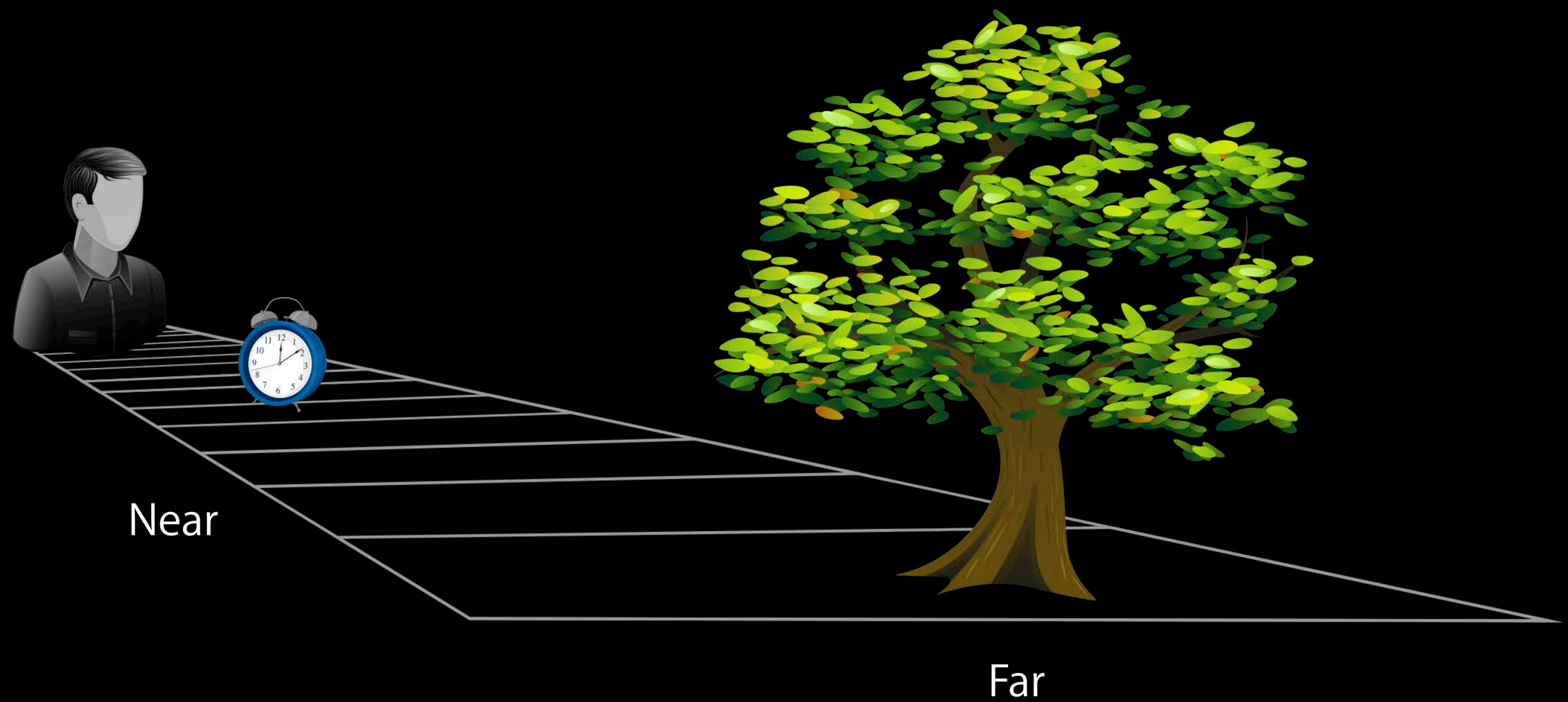
- 60 FPS Support
- Video zoom
- Machine Readable Code detection
- Focus enhancements
- Integration with application AudioSession

Focus Enhancements

Brad Ford
Core Media Engineering

Focus Enhancements

Auto-focus range restrictions



Focus Enhancements

Auto-focus range restrictions

NEW

Focus Enhancements

Auto-focus range restrictions

- `[avCaptureDevice setAutoFocusRangeRestriction:]`

```
AVCaptureAutoFocusRangeRestrictionNone
AVCaptureAutoFocusRangeRestrictionNear
AVCaptureAutoFocusRangeRestrictionFar
```

Focus Enhancements

Auto-focus range restrictions

- `[avCaptureDevice setAutoFocusRangeRestriction:]`

```
AVCaptureAutoFocusRangeRestrictionNone
AVCaptureAutoFocusRangeRestrictionNear
AVCaptureAutoFocusRangeRestrictionFar
```

- For Machine-Readable Code detection, use “Near”

Focus Enhancements

Auto-focus range restrictions

- `[avCaptureDevice setAutoFocusRangeRestriction:]`

```
AVCaptureAutoFocusRangeRestrictionNone
AVCaptureAutoFocusRangeRestrictionNear
AVCaptureAutoFocusRangeRestrictionFar
```

- For Machine-Readable Code detection, use “Near”
- Supported on all cameras that support focus!

Focus Enhancements

Smooth auto-focus

Fast Focus

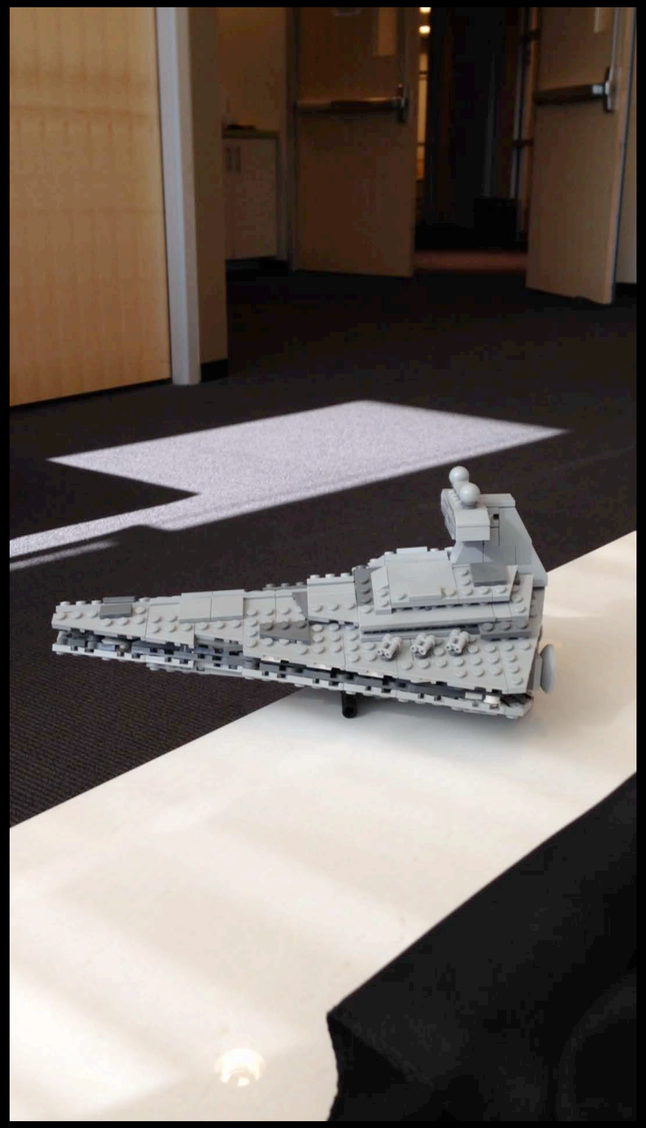
Smooth Focus

Focus Enhancements

Smooth auto-focus

Fast Focus

Smooth Focus



Focus Enhancements

Smooth auto-focus

NEW

Focus Enhancements

Smooth auto-focus

- `[avCaptureDevice setSmoothAutoFocusEnabled:YES];`

Focus Enhancements

Smooth auto-focus

- `[avCaptureDevice setSmoothAutoFocusEnabled:YES];`
- Smooth auto-focus slows the focus scan and is less visually intrusive

Focus Enhancements

Smooth auto-focus

- `[avCaptureDevice setSmoothAutoFocusEnabled:YES];`
- Smooth auto-focus slows the focus scan and is less visually intrusive
- Use smooth auto-focus for movie recording

Focus Enhancements

Smooth auto-focus

- `[avCaptureDevice setSmoothAutoFocusEnabled:YES];`
- Smooth auto-focus slows the focus scan and is less visually intrusive
- Use smooth auto-focus for movie recording
- Use fast focus for still image taking

Focus Enhancements

Smooth auto-focus

- `[avCaptureDevice setSmoothAutoFocusEnabled:YES];`
- Smooth auto-focus slows the focus scan and is less visually intrusive
- Use smooth auto-focus for movie recording
- Use fast focus for still image taking
- Supported on iPhone 5

Focus Enhancements

Smooth auto-focus

- `[avCaptureDevice setSmoothAutoFocusEnabled:YES];`
- Smooth auto-focus slows the focus scan and is less visually intrusive
- Use smooth auto-focus for movie recording
- Use fast focus for still image taking
- Supported on iPhone 5
- See SloPoke Sample code for reference

Focus Enhancements

Programming notes

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

Focus Enhancements

Programming notes

```
NSError *error = nil;
BOOL success = [videoDevice lockForConfiguration:&error];
if ( success ) {
  if ( videoDevice.isAutoFocusRangeRestrictionSupported ) {
    [videoDevice
      setAutoFocusRangeRestriction:AVCaptureAutoFocusRangeRestrictionFar];
  }
  if ([videoDevice isSmoothAutoFocusSupported]) {
    [videoDevice setSmoothAutoFocusEnabled:YES];
  }
  if ([videoDevice isFocusPointOfInterestSupported]) {
    [videoDevice setFocusPointOfInterest:CGPointMake(.5, .5)];
  }

  if ([videoDevice isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
    [videoDevice setFocusMode:AVCaptureFocusModeAutoFocus];
  }

  [videoDevice unlockForConfiguration];
}
```

New in iOS 7

- 60 FPS Support
- Video zoom
- Machine Readable Code detection
- Focus enhancements
- Integration with application AudioSession

What Is an AVAudioSession?

What Is an AVAudioSession?

- Every application has a singleton AVAudioSession

What Is an AVAudioSession?

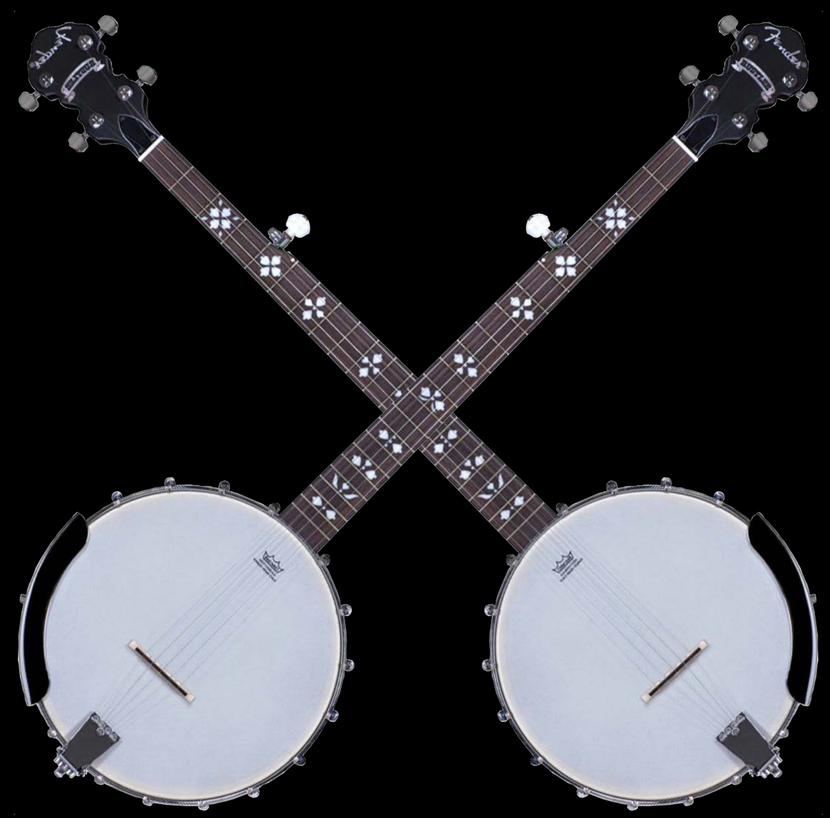
- Every application has a singleton AVAudioSession
- AVAudioSession interfaces let you configure:
  - Audio category
  - Category options
  - Mode
  - Preferred sample rate
  - Preferred IOBufferDuration
  - Input gain

What Is an AVAudioSession?

What Is an AVAudioSession?

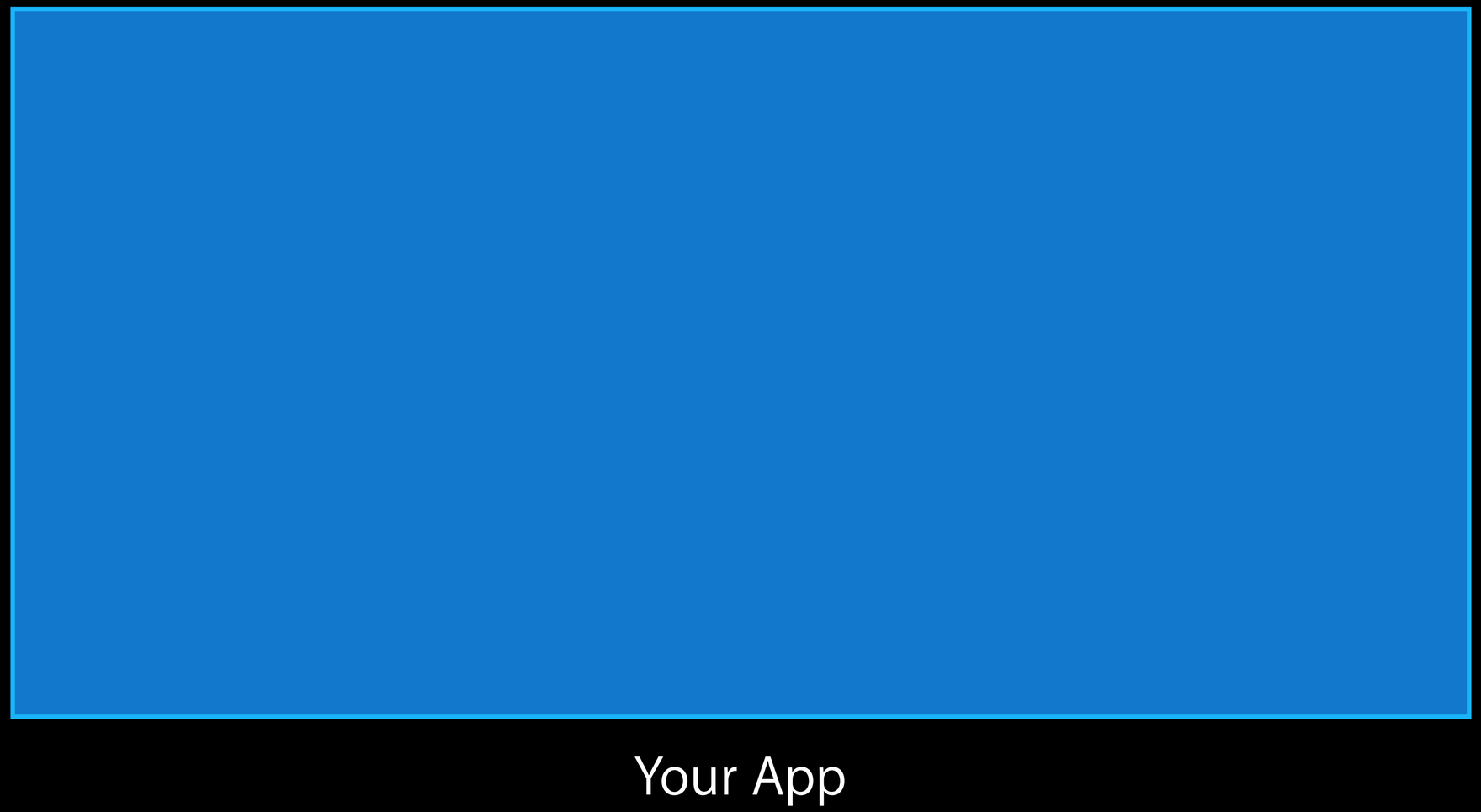
- (NEW in iOS 7) AVAudioSession Microphone selection
  - Physical location (upper or lower)
  - Physical orientation (front, back, top, bottom)
  - Polar pattern (omnidirectional, cardioid, etc.)

Dueling Audio Sessions



Dueling Audio Sessions

Dueling Audio Sessions



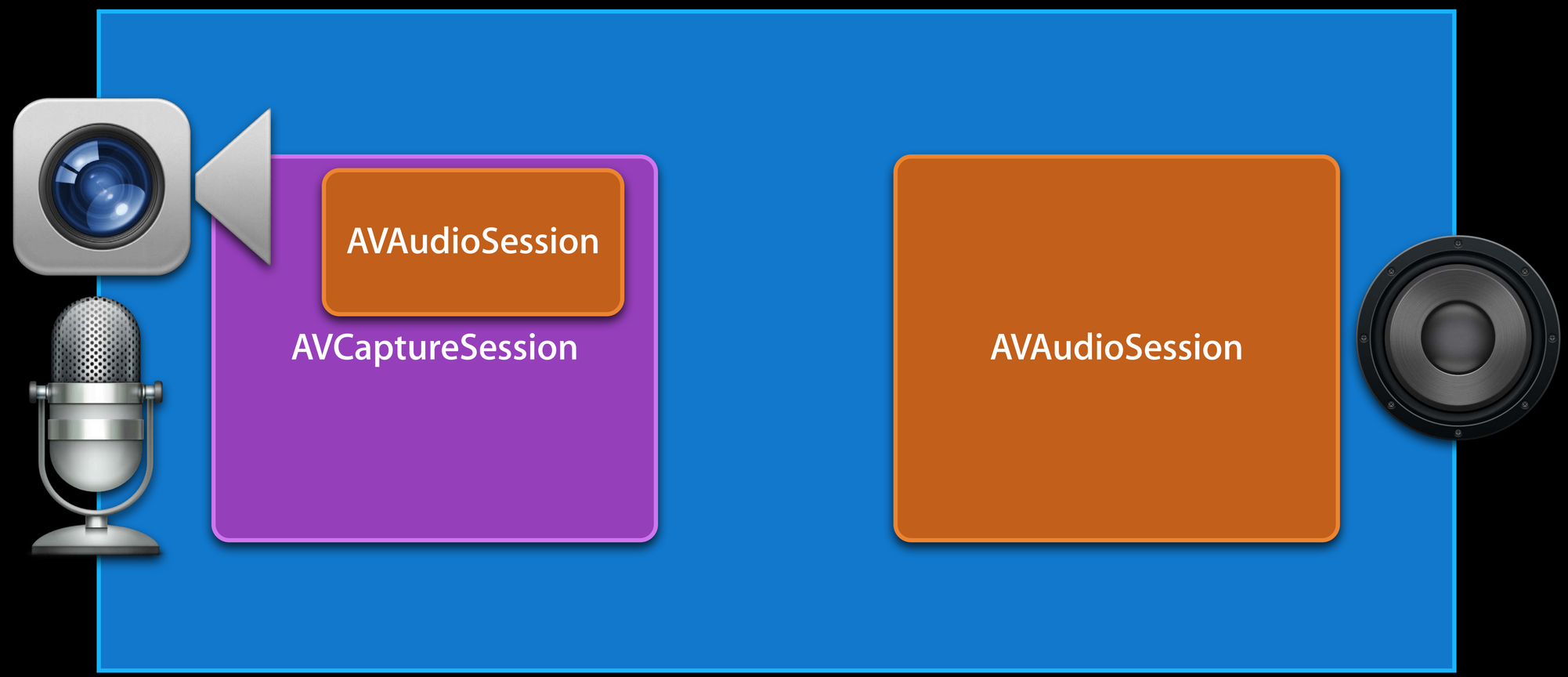
Dueling Audio Sessions

Dueling Audio Sessions

Dueling Audio Sessions

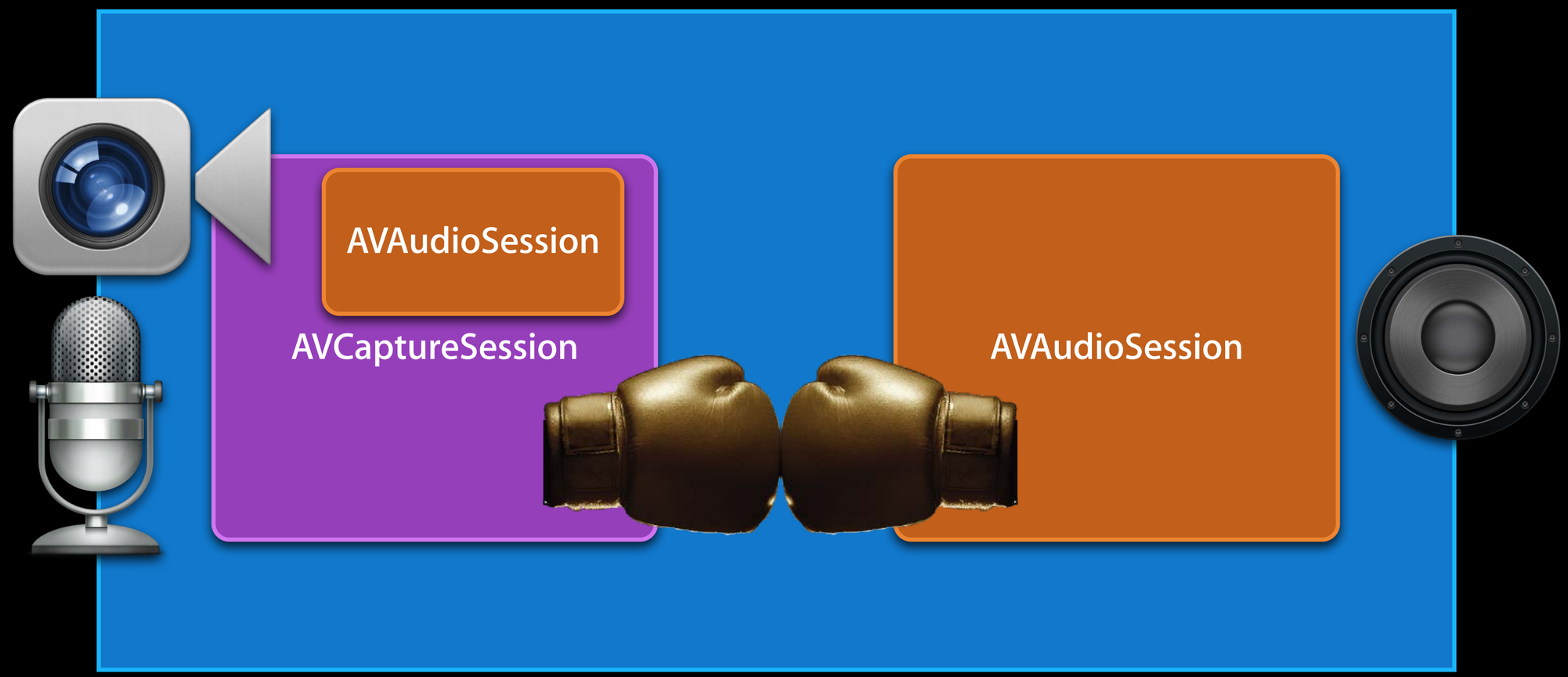
Your App

Dueling Audio Sessions



Your App

Dueling Audio Sessions



AVCapture App AudioSession Adoption

AVCapture App AudioSession Adoption

- AVCaptureSession uses your app audio session by default

```
[session usesApplicationAudioSession] == YES
```

  - Apps linked before iOS 7 get the old behavior

AVCapture App AudioSession Adoption

- AVCaptureSession uses your app audio session by default

```
[session usesApplicationAudioSession] == YES
```

  - Apps linked before iOS 7 get the old behavior
- AVCaptureSession still configures the audio session by default

```
[session automaticallyConfiguresApplicationAudioSession] == YES
```

  - After capture is finished, AVCaptureSession does not restore any AVAudioSession state

AVCapture App AudioSession Adoption

- AVCaptureSession uses your app audio session by default

```
[session usesApplicationAudioSession] == YES
```

  - Apps linked before iOS 7 get the old behavior
- AVCaptureSession still configures the audio session by default

```
[session automaticallyConfiguresApplicationAudioSession] == YES
```

  - After capture is finished, AVCaptureSession does not restore any AVAudioSession state

Be careful, recording may fail

Audio AVCaptureDevice Formats

Audio AVCaptureDevice Formats

- Audio AVCaptureDevices expose no formats array

Audio AVCaptureDevice Formats

- Audio AVCaptureDevices expose no formats array
- Use AVAudioSession to configure audio input

AVCapture App AudioSession Best Practices

- DO let AVCaptureSession share your app audio session
- DO let AVCaptureSession auto-configure your app audio session
  - Exceptions
    - Custom microphone selection
    - Custom category
    - Custom sample rate, etc.

What's New in Camera Capture

- Greater transparency for users
- New AV Foundation capture features in iOS 7
- Sample code update

Sample Code Update

- VideoSnake (WWDC 2013 edition)
  - Updated to iOS 7 APIs
  - Illustrates best practices
  - Integrates with OpenGL
  - Illustrates synchronized a/v movie writing with AssetWriter
  - Download it today!

Sample Code Update

Sample Code Update

Apple Sample Code Is Your Friend

Summary

- iOS user consent for microphone and camera use
- Powerful new AV Foundation capture features
  - 60 FPS capture, playback, editing, and export
  - Full-resolution video data output support
  - Video zoom
  - Machine Readable Code detection
  - Focus enhancements
  - App audio session integration
- Best practices sample code update

Sample Code for This Session

- SloPoke for iOS
- SoZoomy for iOS
- QRchestra for iOS
- VideoSnake for iOS (updated!)

Materials available at:
https://developer.apple.com/library/prerelease/ios/navigation/index.html#section=Resource%20Types&topic=Sample%20Code

More Information

John Geleynse
Director, Technology Evangelist
geleynse@apple.com

Documentation
AV Foundation Programming Guide
http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG/

Apple Developer Forums
http://devforums.apple.com

Related Sessions

What's New in Core Audio for iOS	Nob Hill Tuesday 11:30 AM
Moving to AV Kit and AV Foundation	Pacific Heights Tuesday 4:30 PM
Preparing and Presenting Media for Accessibility	Nob Hill Wednesday 10:15 AM
Advanced Editing with AV Foundation	Marina Thursday 9:00 AM

Labs

Audio Lab	Media Lab B Wednesday 9:00 AM
iOS Camera Capture Lab	Media Lab B Wednesday 2:00 PM
OS X and iOS Capture Lab	Media Lab B Thursday 9:00 AM
HTTP Live Streaming Lab	Media Lab B Thursday 11:30 AM
AV Foundation Lab	Media Lab B Thursday 2:00 PM
AV Foundation Lab	Media Lab B Friday 9:00 AM

WWDC2013