Using the Camera with AV Foundation

Overview and best practices

Brad Ford
iPhone Engineering

What You'll Learn

- Why and when you should use AV Foundation capture
- The AV Foundation capture programming model
- Performance considerations when working with the camera

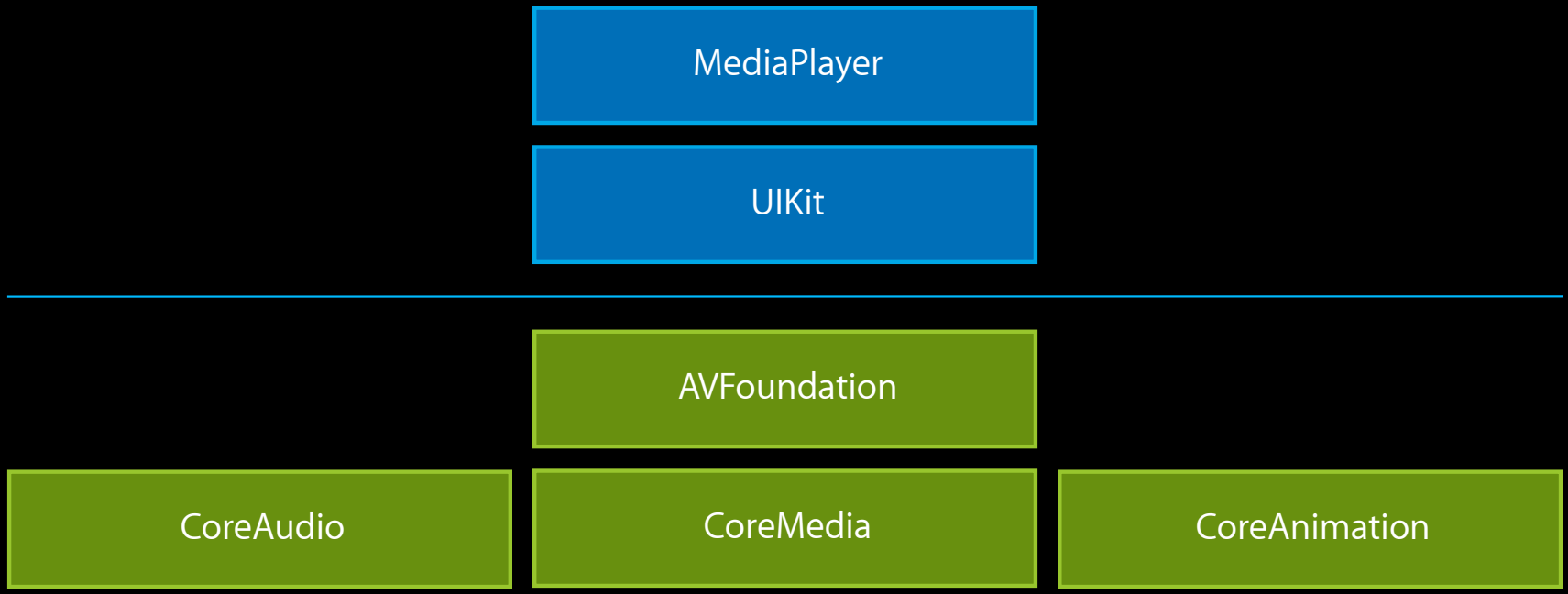
Sample Code for This Session

Only on iPhone

- Find My iCone
- AVCam Demo
- PinchyPreview
- Wavy

Materials available at:
http://developer.apple.com/wwdc/attendee/

Technology Framework



Using the Camera in iPhone OS 3.x

Simple programmatic access

UIImagePickerController

API for high, medium, or low quality recording

Hideable camera controls UI

takePhoto API

Control overlay for start/stop recording

User touch to focus, like Camera app

Using the Camera in iOS 4

UIImagePickerController offers more control

- Movie recording at high, medium, and low quality
- High resolution still image capture
- Access to the flash
- Switching between front and back cameras

Incorporating the Camera and Photo Library in your App	Presidio Thursday 9:00AM

Why Use AV Foundation Capture?

Full access to the camera

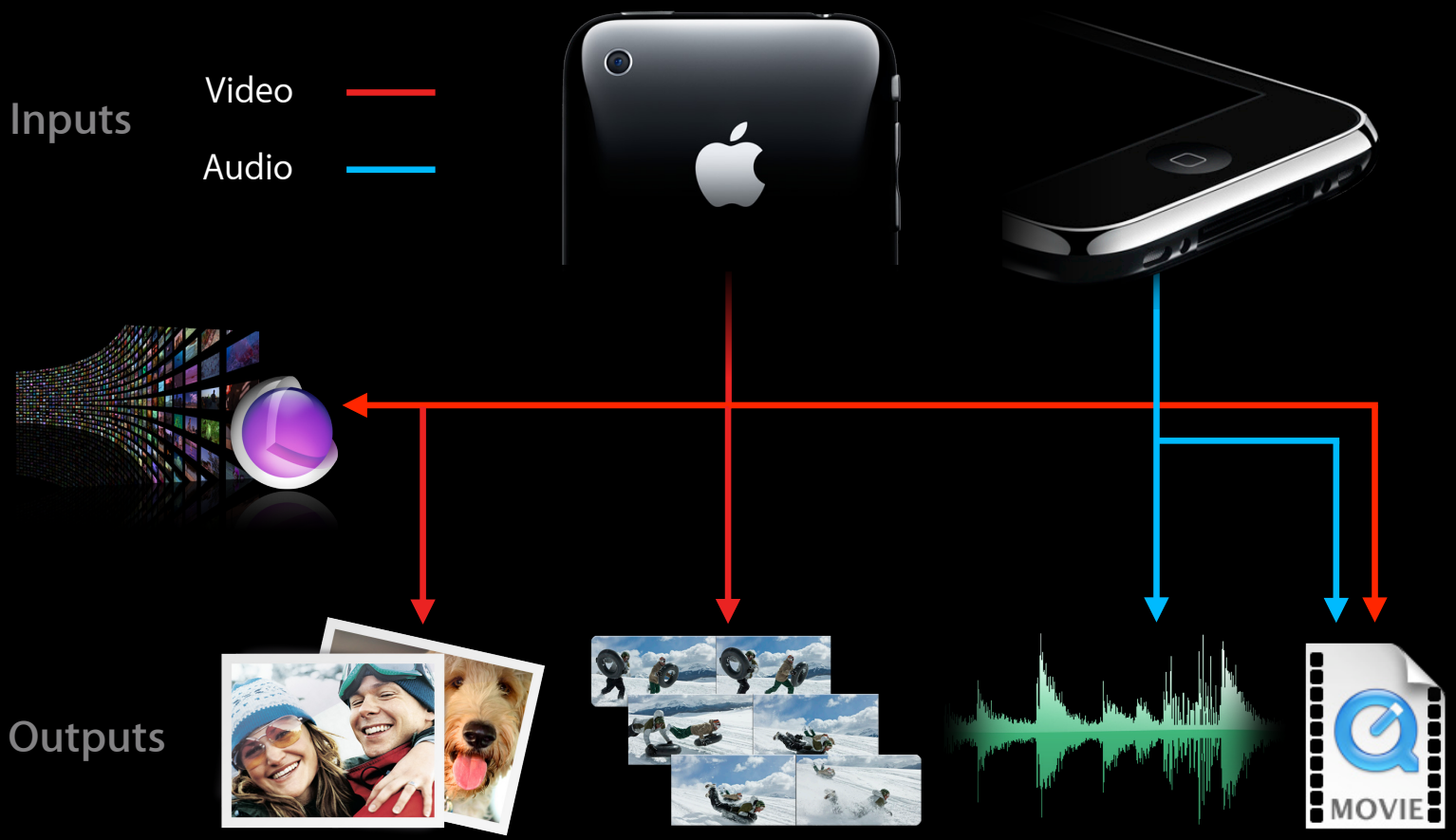
- Independent focus, exposure, and white balance controls
  - Independent locking
  - Independent points of interest for focus and exposure
- Access to video frame data
  - Accurate timestamps
  - Per-frame metadata
  - Configurable output format (e.g. 420v, BGRA)
  - Configurable max frame rate
  - Configurable resolution

Why Use AV Foundation Capture?

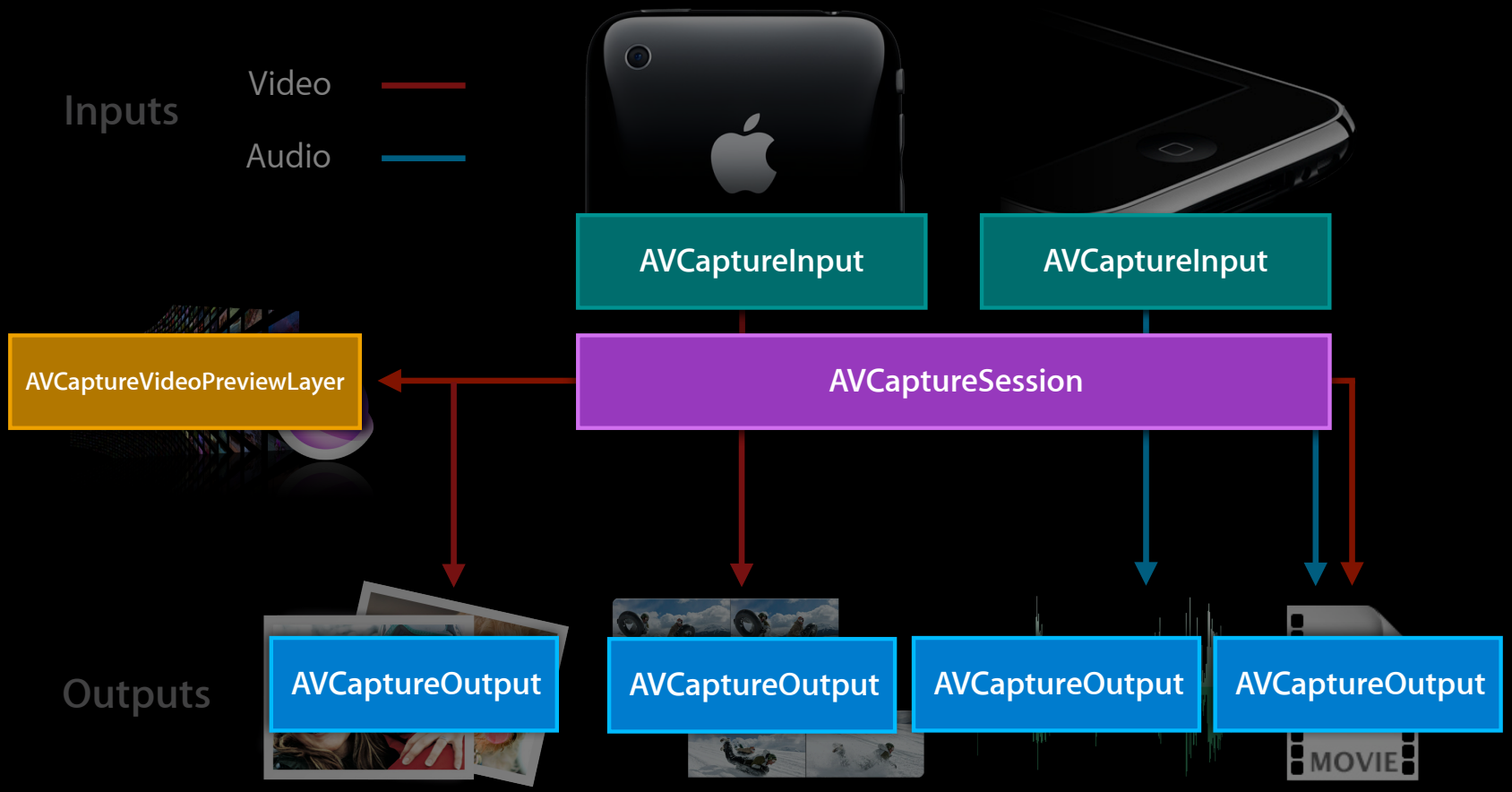
Flexible output

- Still Image
  - YUV and RGB output
  - Exif metadata insertion
- QuickTime Movie
  - Movie metadata insertion
  - Orientation lock
- Video Preview in a CALayer
  - Aspect fit, aspect fill, and stretch
- Audio level metering
- Access to audio sample data

Capture Basics: Inputs and Outputs



Capture Basics: Inputs and Outputs



Common Capture Use Cases

- Process YUV video frames from the camera
- Control the camera, take photos, and record movies
- Preview video from the camera to a Core Animation layer
- Process PCM audio data from the microphone to draw a waveform

Common Capture Use Cases

- Process YUV video frames from the camera
- Control the camera, take photos, and record movies
- Preview video from the camera to a Core Animation layer
- Process PCM audio data from the microphone to draw a waveform

Demo

Find My iCone

Processing frames using AVCaptureVideoDataOutput

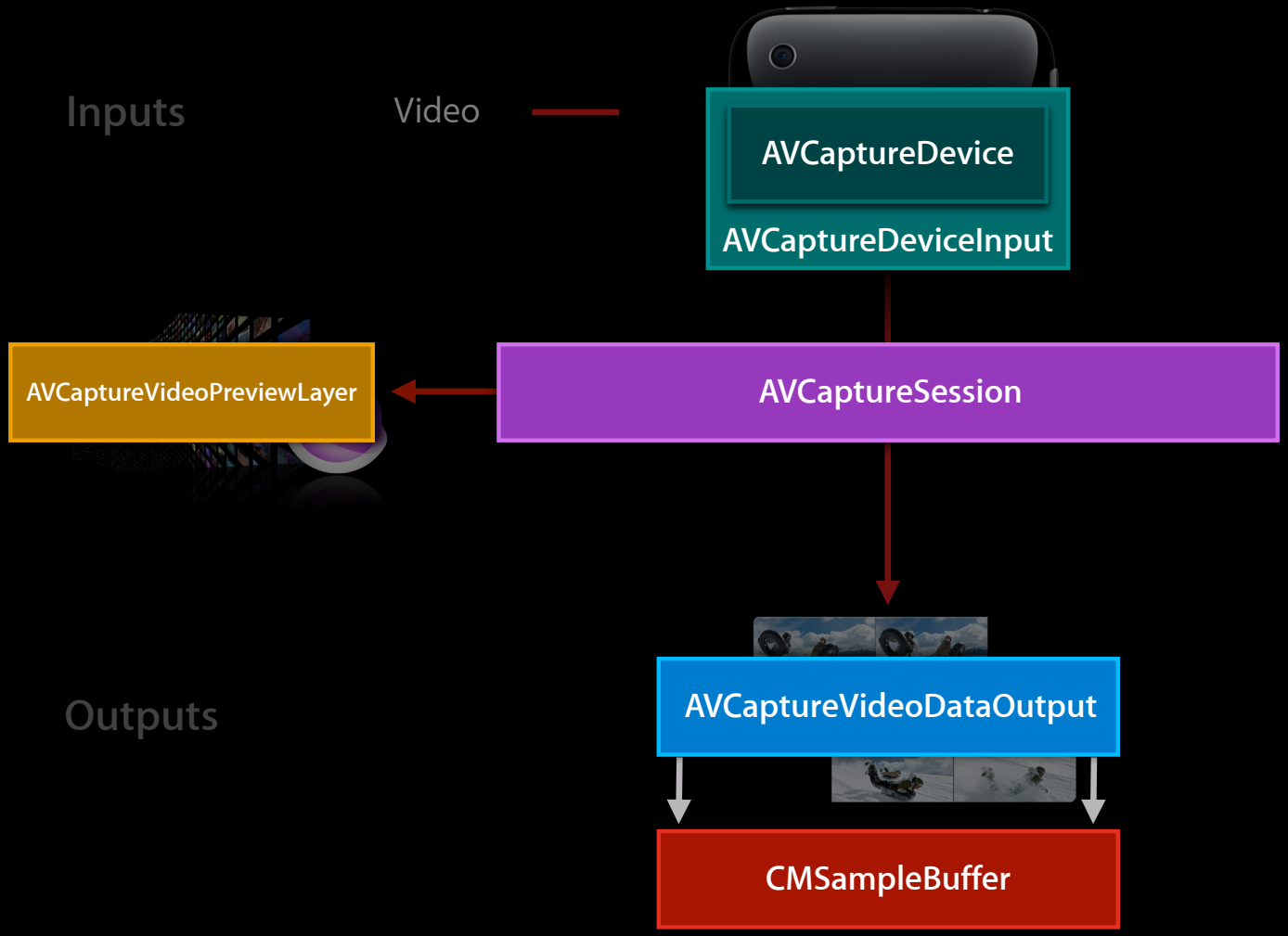
Find My iCone

Inputs

Video

Outputs

Find My iCone



Find My iCone *(Code Snippet SP16)*

- Create an AVCaptureSession

```
AVCaptureSession *session = [[AVCaptureSession alloc] init];

session.sessionPreset = AVCaptureSessionPresetHigh;
```

- Find a suitable AVCaptureDevice

```
AVCaptureDevice *device = [AVCaptureDevice
                           defaultDeviceWithMediaType:AVMediaTypeVideo];
```

- Create and add an AVCaptureDeviceInput

```
AVCaptureDeviceInput *input = [AVCaptureDeviceInput deviceInputWithDevice:device
                                error:&error];
[session addInput:input];
```

Find My iCone *(Code Snippet SP16)*

- Create and add an AVCaptureVideoDataOutput

```
AVCaptureVideoDataOutput *output = [[AVCaptureVideoDataOutput alloc] init];
[session addOutput:output];
```

- Configure your output, and start the session

```
dispatch_queue_t queue = dispatch_queue_create("myQueue", NULL);
[output setSampleBufferDelegate:self queue:queue];
[session startRunning];
```

- Implement your delegate callback

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput
        didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
        fromConnection:(AVCaptureConnection *)connection
{
  // CFShow(sampleBuffer);
}
```

AVCaptureVideoDataOutput

What is a CMSampleBuffer?

- Defined in <CoreMedia/CMSampleBuffer.h>
- Reference-counted Core Foundation object containing:
  - Sample data

```
CVPixelBufferRef pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer);

// With the pixelBuffer, you can access the buffer's base address,
// row bytes, etc.

// See <CoreVideo/CVPixelBuffer.h>
```

AVCaptureVideoDataOutput

What is a CMSampleBuffer?

- Defined in <CoreMedia/CMSampleBuffer.h>
- Reference-counted Core Foundation object containing:
  - Timing Information

```
CMTime presentationTime = CMSampleBufferGetPresentationTimeStamp(sampleBuffer);

CMTime decodeTime = CMSampleBufferGetDecodeTimeStamp(sampleBuffer);
```

AVCaptureVideoDataOutput

What is a CMSampleBuffer?

- Defined in <CoreMedia/CMSampleBuffer.h>
- Reference-counted Core Foundation object containing:
  - Format Information

```
CMFormatDescriptionRef desc = CMSampleBufferGetFormatDescription(sampleBuffer);

int32_t pixelType = CMVideoFormatDescriptionGetCodecType(desc);
CMVideoDimensions dimensions = CMVideoFormatDescriptionGetDimensions(desc);
```

AVCaptureVideoDataOutput

What is a CMSampleBuffer?

- Defined in <CoreMedia/CMSampleBuffer.h>
- Reference-counted Core Foundation object containing:
  - Metadata about the sample data

```
// Metadata are carried as attachments
CFDictionaryRef metadataDictionary =
  CMGetAttachment(sampleBuffer, CFSTR("MetadataDictionary"), NULL);

if (metadataDictionary)
  CFShow(metadataDictionary);
```

AVCaptureVideoDataOutput

What is a CMSampleBuffer?

- Defined in <CoreMedia/CMSampleBuffer.h>
- Reference-counted Core Foundation object containing:
  - Sample Data
  - Timing Information
  - Format Information
  - Metadata about the sample data

- See *Code Snippet SP20*.
- Refer to the CoreMedia framework documentation

AVCaptureVideoDataOutput

Performance considerations

- `setSampleBufferDelegate:queue:` requires a serial dispatch queue to ensure properly ordered buffer callbacks
- Note: Don't pass `dispatch_get_current_queue()`.
- By default, buffers are emitted in the camera's most efficient format.
- Set the `videoSettings` property to specify a custom output format, such as 'BGRA' *(see Code Snippet SP16)*
- Hint: Both CoreGraphics and OpenGL work well with 'BGRA'

AVCaptureVideoDataOutput

Performance considerations

- Set the `minFrameDuration` property to cap the max frame rate
- Configure the session to output the lowest practical resolution
- Set the `alwaysDiscardsLateVideoFrames` property to YES (the default) for early, efficient dropping of late video frames
- Your sample buffer delegate's callback must be FAST!
- The camera may stop delivering frames if you hold onto buffers for too long

AVCaptureSession

- The central hub of the AV Foundation capture APIs
- The place for adding inputs and outputs
- Starts the flow of data when you call `-startRunning`
- Allows quality customization using `-sessionPreset`

High	Highest recording quality
Medium	Suitable for WiFi sharing
Low	Suitable for 3G sharing
640x480	VGA
1280x720	720p HD
Photo	Full photo resolution

AVCaptureVideoDataOutput

Supported resolutions and formats

Preset	iPhone 3G (2vuy, BGRA)	iPhone 3GS (420v, BGRA)	iPhone 4 (Back) (420v, BGRA)	iPhone 4 (Front) (420v, BGRA)
High	400x304	640x480	1280x720	640x480
Medium	400x304	480x360	480x360	480x360
Low	400x304	192x144	192x144	192x144
640x480	N/A	640x480	640x480	640x480
1280x720	N/A	N/A	1280x720	N/A
Photo	N/A	N/A	N/A	N/A

Common Capture Use Cases

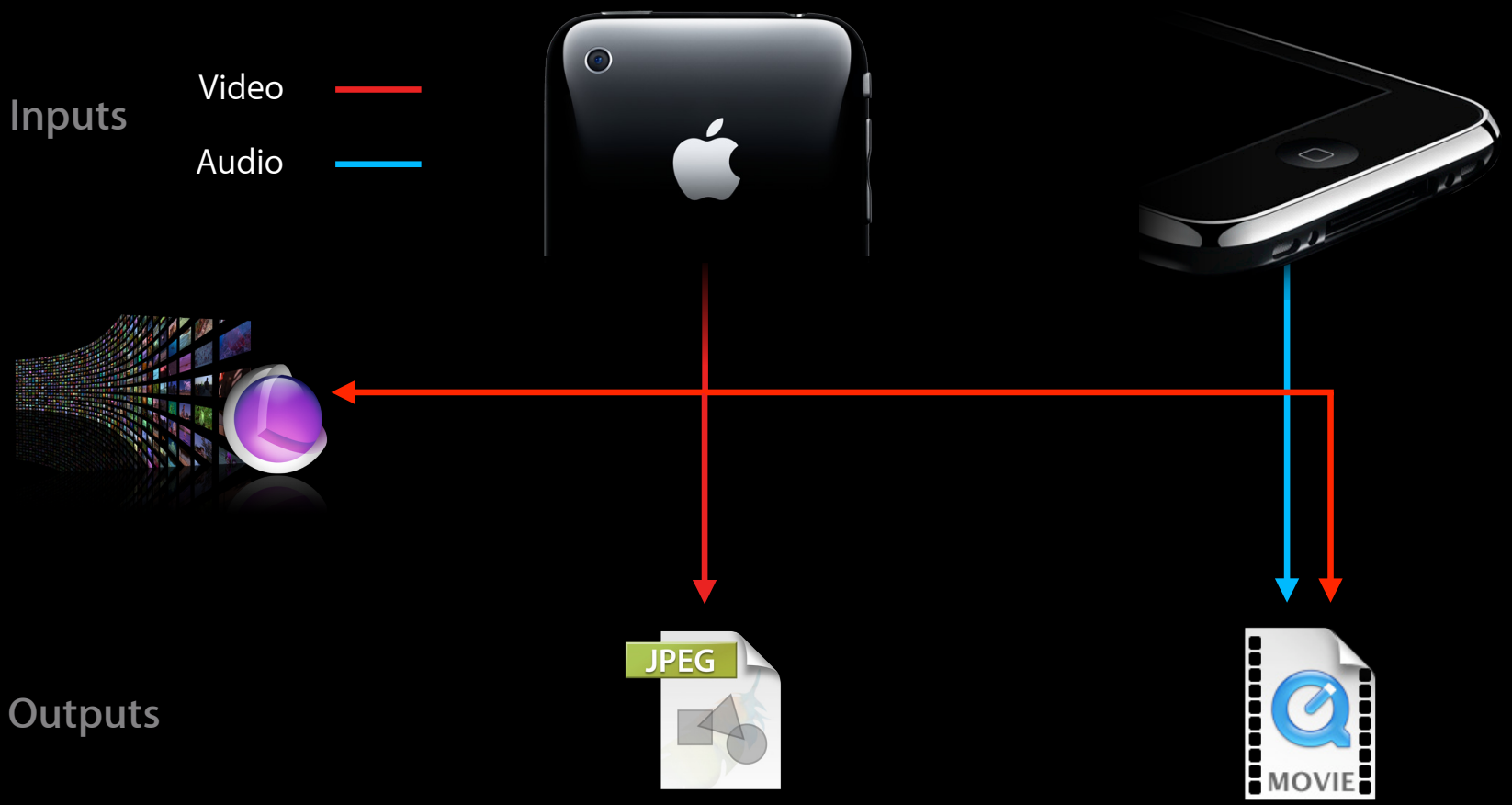
- Process YUV video frames from the camera
- Control the camera, take photos, and record movies
- Preview video from the camera to a Core Animation layer
- Process PCM audio data from the microphone to draw a waveform

Demo

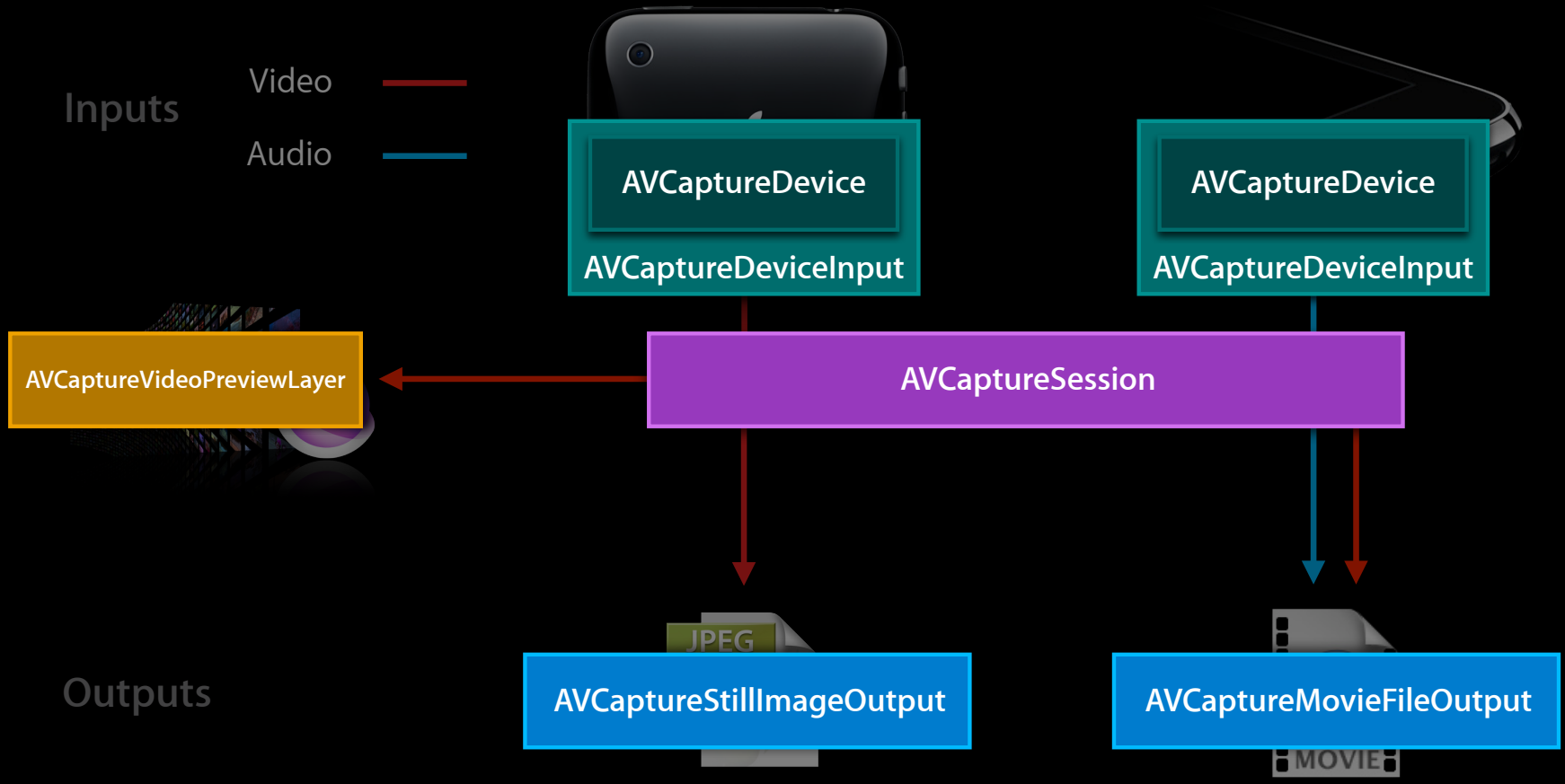
AVCam

Controlling the camera, taking still images, and recording video

AVCam Demo



AVCam Demo


Inputs
Video
Audio
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureVideoPreviewLayer
AVCaptureSession
Outputs
JPEG
AVCaptureStillImageOutput
AVCaptureMovieFileOutput
MOVIE

AVCam Demo: Camera Controls

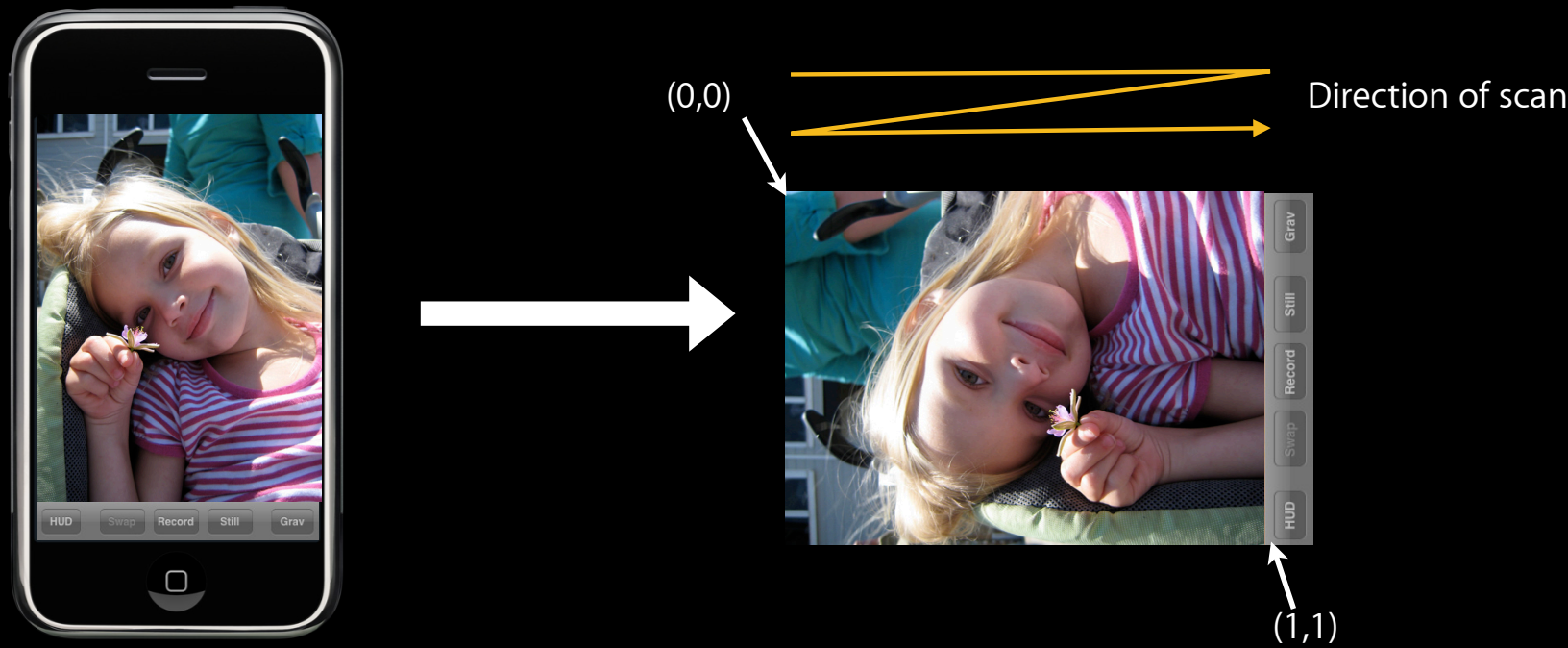
Focus support

- Focus modes:
  - AVCaptureFocusModeLocked
  - AVCaptureFocusModeAutoFocus
  - AVCaptureFocusModeContinuousAutoFocus
- Use isFocusModeSupported: To determine if the AVCaptureDevice supports a given mode
- The adjustingFocus property is key-value observable

AVCam Demo: Camera Controls

Focus support

- focusPointOfInterest is a CGPoint from (0, 0) to (1, 1)
- Top-left is (0,0), bottom-right is (1,1)
- iPhone sensor is mounted portrait, but scans landscape



AVCam Demo: Camera Controls

Focus support

- Setting the focus mode or focus point of interest requires calling lockForConfiguration: On the AVCaptureDevice

```
if ( [device isFocusModeSupported:AVCaptureFocusModeLocked] ) {
  if ( YES == [device lockForConfiguration:&error] ) {
    device.focusMode = AVCaptureFocusModeLocked;
    [device unlockForConfiguration];
  }
}
```

Avoid holding lockForConfiguration: Unnecessarily

AVCam Demo: Camera Controls

Exposure support

- Exposure modes:
  - `AVCaptureExposureModeLocked`
  - `AVCaptureExposureModeContinuousAutoExposure`
- Use `isExposureModeSupported:` To determine if the AVCaptureDevice supports a given mode
- The `adjustingExposure` property is key-value observable
- Exposure point of interest and focus point of interest are mutually exclusive, as are focus mode and exposure mode
- You must call lockForConfiguration: To set exposure mode or point of interest

AVCam Demo: Camera Controls

White balance support

- White balance modes:
  - `AVCaptureWhiteBalanceModeLocked`
  - `AVCaptureWhiteBalanceModeContinuousAutoWhiteBalance`
- Use `isWhiteBalanceModeSupported:` To determine if the AVCaptureDevice supports a given mode
- The `adjustingWhiteBalance` property is key-value observable
- You must call lockForConfiguration: To set white balance mode

AVCam Demo: Camera Controls

Flash support

- Flash modes:
  - `AVCaptureFlashModeOff`
  - `AVCaptureFlashModeOn`
  - `AVCaptureFlashModeAuto`
- Use `hasFlash` to determine if the AVCaptureDevice has this feature
- You must call lockForConfiguration: To set the flash mode

AVCam Demo: Camera Controls

Torch support

- Torch modes:
  - `AVCaptureTorchModeOff`
  - `AVCaptureTorchModeOn`
  - `AVCaptureTorchModeAuto`
- Use `hasTorch` to determine if the AVCaptureDevice has this feature
- You must call lockForConfiguration: To set the torch mode
- *Note* The torch only turns on if the device is associated with an AVCaptureSession instance that is running

AVCaptureDevice

Supported camera controls

Feature	iPhone 3G	iPhone 3GS	iPhone 4 (Back)	iPhone 4 (Front)
focusMode		√	√	
focusPointOfInterest		√	√	
exposureMode	√	√	√	√
exposurePointOfInterest		√	√	√
whiteBalanceMode	√	√	√	√
flashMode			√	
torchMode			√	

AVCam Demo: Camera switching

Performance consideration

- `startRunning` and `stopRunning` are synchronous
- An AVCaptureSession may be reconfigured while running
- Use `beginConfiguration` and `commitConfiguration`
- *Follow along in Code Snippet SP21*

```
// Don’t -stopRunning before reconfiguring.
[session beginConfiguration];

[session removeInput:frontFacingCameraDeviceInput];
[session addInput:backFacingCameraDeviceInput];

[session commitConfiguration];
// Changes are only committed when the outermost -commitConfiguration
// is invoked.
```

AVCam Demo: Movie Recording

Performance consideration

- Initiate a QuickTime movie recording by supplying a file URL and delegate

```
[movieFileOutput startRecordingToOutputFileURL:myURL
                                recordingDelegate:self];
```

- One AVCaptureFileOutputRecordingDelegate method is mandatory

```
- (void)captureOutput:(AVCaptureFileOutput *)captureOutput
        didFinishRecordingToOutputFileAtURL:(NSURL *)outputFileURL
        fromConnections:(NSArray *)connections error:(NSError *)error
{
  // Write resulting movie to the camera roll.
  // See Code Snippet SP24
}
```

AVCam Demo: Movie Recording

AVCaptureMovieFileOutput writing policy

- You must pass a file-based NSURL
- You may not pass a URL to an existing file; The movie file output does not overwrite existing resources
- You must have permission to write to the URL specified

AVCam Demo: Movie Recording

Setting limits

- -maxRecordedDuration
- -maxRecordedFileSize
- -minFreeDiskSpaceLimit

```
- (void)captureOutput:(AVCaptureFileOutput *)captureOutput
        didFinishRecordingToOutputFileAtURL:(NSURL *)outputFileURL
        fromConnections:(NSArray *)connections error:(NSError *)error
{
  // Check the error AND its userInfo dictionary!
  BOOL recordingSuccess = YES; // assume success.
  if ( [error code] != noErr ) {
    id val = [[error userInfo] objectForKey:AVErrorRecordingSuccessfullyFinishedKey];
    if ( val )
      recordingSuccess = [val boolValue];
  }
}
```

AVCam Demo: Movie Recording

Early recording termination conditions

- AVErrorDiskFull
- AVErrorDeviceWasDisconnected
- AVErrorMaximumDurationReached
- AVErrorMaximumFileSizeReached
- AVErrorSessionWasInterrupted

- See *Code Snippet SP24* for more detail

AVCam Demo: Movie Recording

Metadata

- You may set movie level metadata at any time while recording
- Allows for "slow" metadata, such as GPS location, to be acquired after the recording has started

```
NSMutableArray *metadata = [[NSMutableArray alloc] initWithCapacity:1];
AVMutableMetadataItem *item = [[AVMutableMetadataItem alloc] init];
item.keySpace = AVMetadataKeySpaceCommon;
item.key = AVMetadataCommonKeyLocation;
item.value = [NSString stringWithFormat:@"%+08.4lf%+09.4lf/",
    location.coordinate.latitute, location.coordinate.longitude];
[metadata addObject:item];
[item release];
movieFileOutput.metadata = metadata;
```

See *Code Snippet SP25* for more metadata examples

AVCaptureMovieFileOutput

Supported resolutions and bitrates (H.264 + AAC except where noted by *)

Preset	iPhone 3G	iPhone 3GS	iPhone 4 (Back)	iPhone 4 (Front)
High	No video *Apple Lossless	640x480 3.5 mbps	1280x720 10.5 mbps	640x480 3.5 mbps
Medium	No video *Apple Lossless	480x360 700 kbps	480x360 700 kbps	480x360 700 kbps
Low	No video *Apple Lossless	192x144 128 kbps	192x144 128 kbps	192x144 128 kbps
640x480	No video *Apple Lossless	640x480 3.5 mbps	640x480 3.5 mbps	640x480 3.5 mbps
1280x720	No video *Apple Lossless	No video 64 kbps AAC	No video 64 kbps AAC	No video 64 kbps AAC
Photo	N/A	N/A	N/A	N/A

AVCam Demo: Photos

AVCaptureStillImageOutput considerations

- Uses a block-style completion handler to deliver image data as a CMSampleBuffer
- The sample buffer contains useful still image metadata

```
[stillImageOutput captureStillImageAsynchronouslyFromConnection:connection
   completionHandler:
     ^(CMSampleBufferRef imageSampleBuffer, NSError *error) {
       CFDictionaryRef exifAttachments =
         CMGetAttachment(imageSampleBuffer,
                         kCGImagePropertyExifDictionary, NULL);
       if (exifAttachments) {
         // Attachments may be read or additional ones written
       }
     }
];
```

AVCam Demo: Photos

Performance considerations

- Use `-availableImageDataCVPixelFormatTypes` and `-availableImageDataCodecTypes` to discover supported output formats
- Set your desired output format using `outputSettings`
- If your final destination still image format is ‘jpeg’, let AVCaptureStillImageOutput do the compression for you
- Use `+jpegStillImageNSDataRepresentation:` To write a jpeg sample buffer to an NSData without recompression, merging in all metadata buffer attachments

AVCaptureStillImageOutput

Supported resolutions and formats

Preset	iPhone 3G (yuvs, 2vuy, BGRA, jpeg)	iPhone 3GS (420f, 420v, BGRA, jpeg)	iPhone 4 (Back) (420f, 420v, BGRA, jpeg)	iPhone 4 (Front) (420f, 420v, BGRA, jpeg)
High	400x304	640x480	1280x720	640x480
Medium	400x304	480x360	480x360	480x360
Low	400x304	192x144	192x144	192x144
640x480	N/A	640x480	640x480	640x480
1280x720	N/A	N/A	1280x720	N/A
Photo	1600x1200	2048x1536	2592x1936	640x480

Common Capture Use Cases

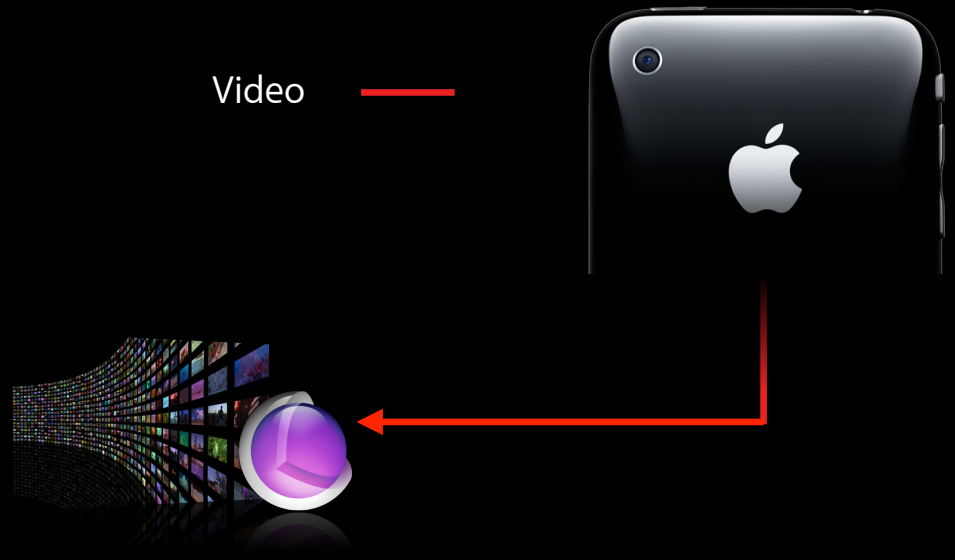
- Process YUV video frames from the camera
- Control the camera, take photos, and record movies
- Preview video from the camera to a Core Animation layer
- Process PCM audio data from the microphone to draw a waveform

Demo

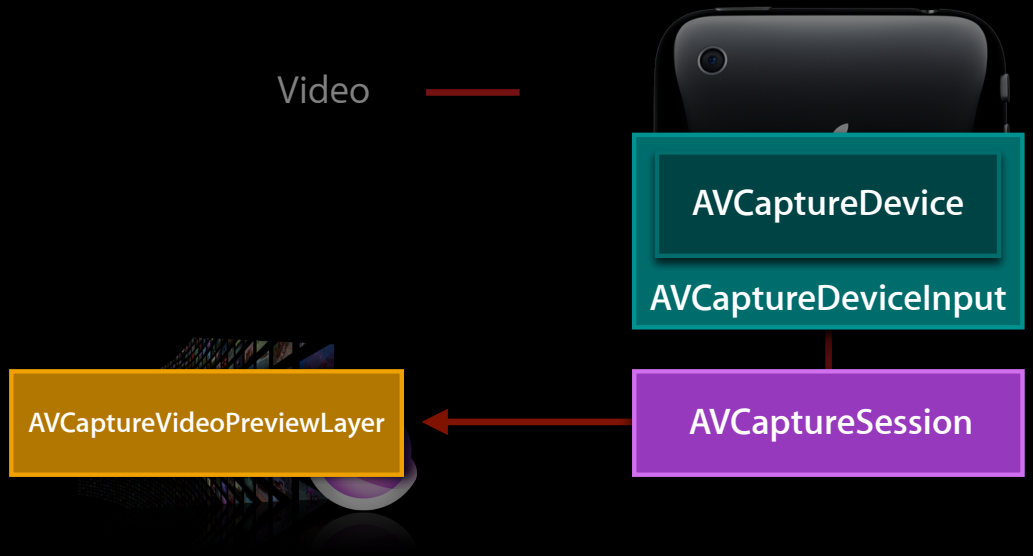
PinchyPreview

Previewing images from the camera using AVCaptureVideoPreviewLayer

PinchyPreview



PinchyPreview



PinchyPreview

AVCaptureVideoPreviewLayer considerations

- Unlike AVCaptureOutput, it retains and owns its AVCaptureSession
- Behaves like any other CALayer in a rendering tree
- You may set the `orientation` property to tell the preview layer how it should rotate images coming from the camera
- On iPhone 4, the preview layer supports mirroring, which is the default when previewing from the front-facing camera

PinchyPreview

AVCaptureVideoPreviewLayer considerations

- Supports three video gravity modes:
  - `AVLayerVideoGravityResizeAspect`
  - `AVLayerVideoGravityResizeAspectFill`
  - `AVLayerVideoGravityResize`

PinchyPreview

AVCaptureVideoPreviewLayer considerations

- Supports three video gravity modes:
  - `AVLayerVideoGravityResizeAspect`
  - `AVLayerVideoGravityResizeAspectFill`
  - `AVLayerVideoGravityResize`

PinchyPreview

AVCaptureVideoPreviewLayer considerations

- Supports three video gravity modes:
  - `AVLayerVideoGravityResizeAspect`
  - `AVLayerVideoGravityResizeAspectFill`
  - `AVLayerVideoGravityResize`

PinchyPreview

AVCaptureVideoPreviewLayer considerations

- Supports three video gravity modes:
  - `AVLayerVideoGravityResizeAspect`
  - `AVLayerVideoGravityResizeAspectFill`
  - `AVLayerVideoGravityResize`

PinchyPreview

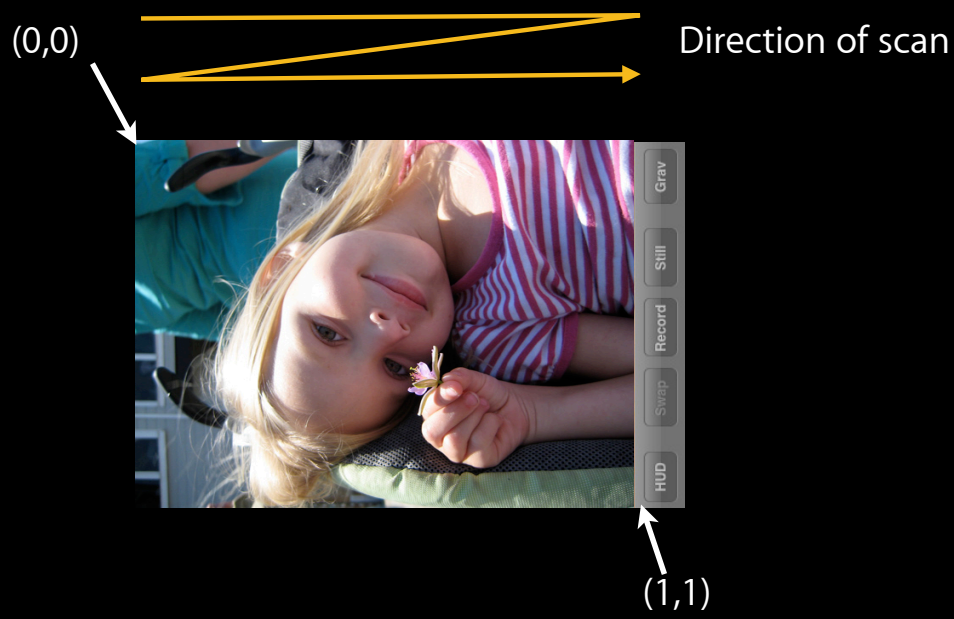
AVCaptureVideoPreviewLayer considerations

- Supports three video gravity modes:
  - `AVLayerVideoGravityResizeAspect`
  - `AVLayerVideoGravityResizeAspectFill`
  - `AVLayerVideoGravityResize`

AVCaptureVideoPreviewLayer

Considerations when driving "tap-to-focus" using a preview

- When mapping a focus touch from a video preview, you must account for:
  - Preview orientation
  - Preview video gravity (aspect fit/fill/stretch)
  - Mirroring
- *Refer to AVCam Demo*



Common Capture Use Cases

- Process YUV video frames from the camera
- Control the camera, take photos, and record movies
- Preview video from the camera to a Core Animation layer
- Process PCM audio data from the microphone to draw a waveform

Demo

Wavy

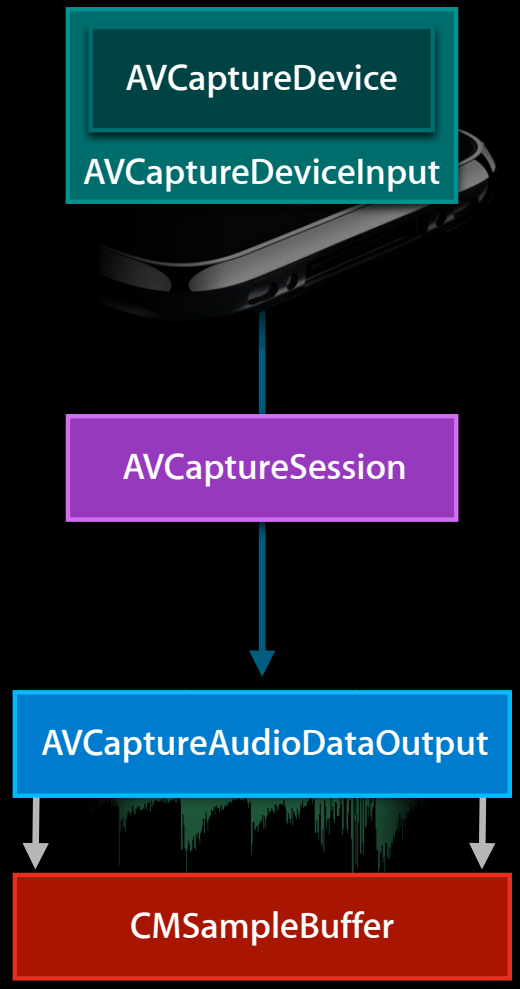
Audio sample processing using AVCaptureAudioDataOutput

Wavy

Audio

Wavy

Audio


AVCaptureDevice
AVCaptureDeviceInput
AVCaptureSession
AVCaptureAudioDataOutput
CMSampleBuffer

AVCaptureAudioDataOutput

Performance considerations

- `setSampleBufferDelegate:queue:` Requires a serial dispatch queue to ensure properly ordered buffer callbacks
- Note: Don't pass `dispatch_get_current_queue()`.
- Emitted sample buffers contain interleaved 16-bit signed integer PCM samples, mono or stereo, at the audio hardware's sample rate
- Unlike AVCaptureVideoDataOutput, there is no affordance for dropping samples
- Be FAST in your delegate callback!

AVCaptureAudioDataOutput

CMSampleBuffer usage

- Contains no image buffer, unlike AVCaptureVideoDataOutput
- Uses CMBlockBuffer instead

```
CMBlockBufferRef blockBuffer = CMSampleBufferGetDataBuffer(sampleBuffer);

char *dataPointer = NULL;
size_t lengthAtOffset, totalLength;
OSStatus err = CMBlockBufferGetDataPointer(
               blockBuffer,
               0, // offset
               &lengthAtOffset,
               &totalLength,
               &dataPointer);
```

Multitasking Considerations

- You may record/process audio in the background if you set the appropriate UIBackgroundModes in your plist

Key	Value
Information Property List	(13 items)
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	☑
Main nib file base name	MainWindow
Required background modes	(1 item)
Item 0	App plays audio

Multitasking Considerations

- Users receive the double-height red status bar when your app runs the audio device in the background

Multitasking Considerations

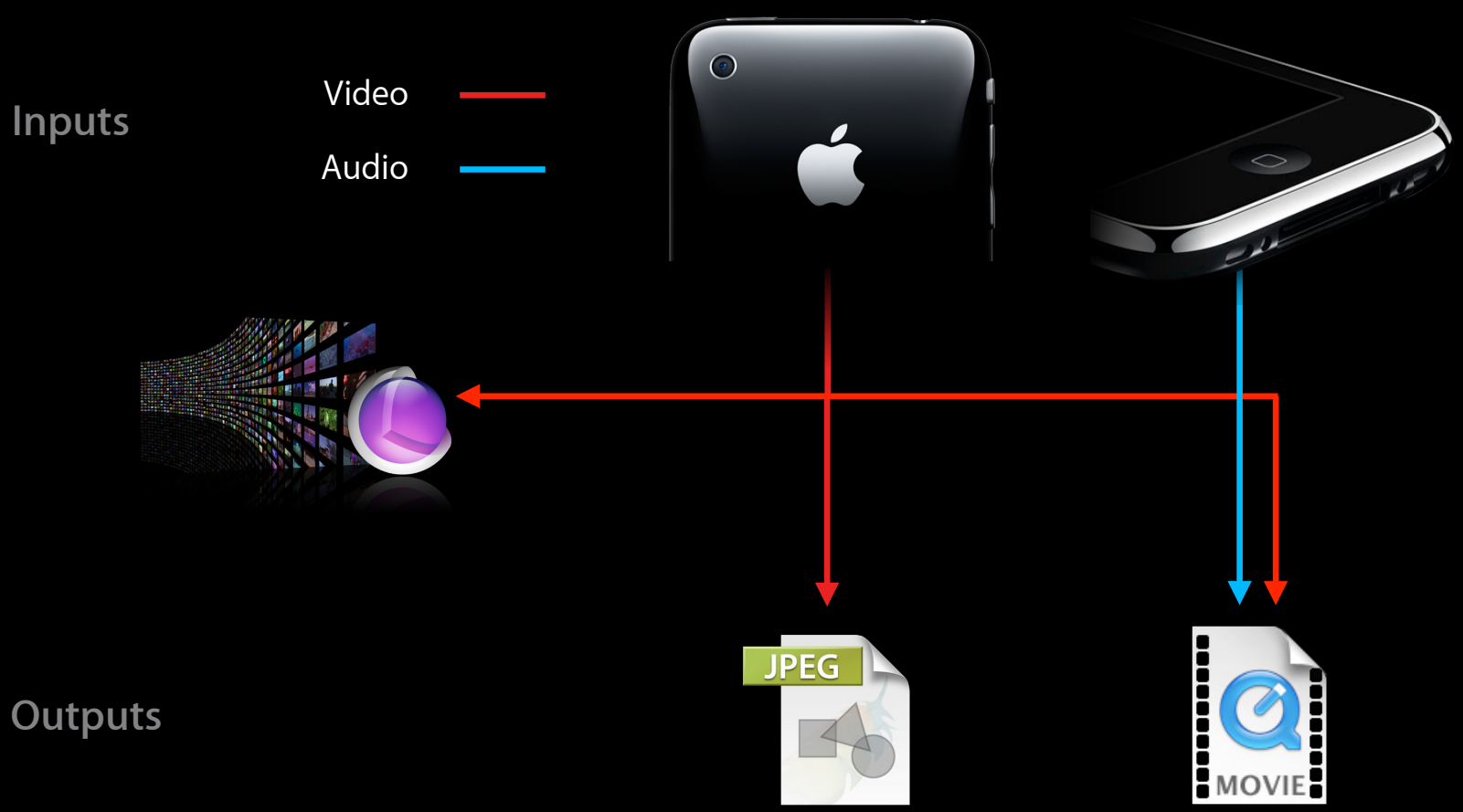
- Background apps may not record/process video from cameras
- When backgrounded, apps running the camera:
  - Receive `AVCaptureSessionWasInterruptedNotification`
  - Receive `AVCaptureSessionDidStopRunningNotification`
  - Movie recordings in progress are stopped
- When re-entering foreground, apps running the camera:
  - Receive `AVCaptureSessionInterruptionEndedNotification`
  - Receive `AVCaptureSessionDidStartRunningNotification`
- `running` and `interrupted` are also key-value observable properties

AVCaptureConnections

The glue that holds the session, inputs, and outputs together

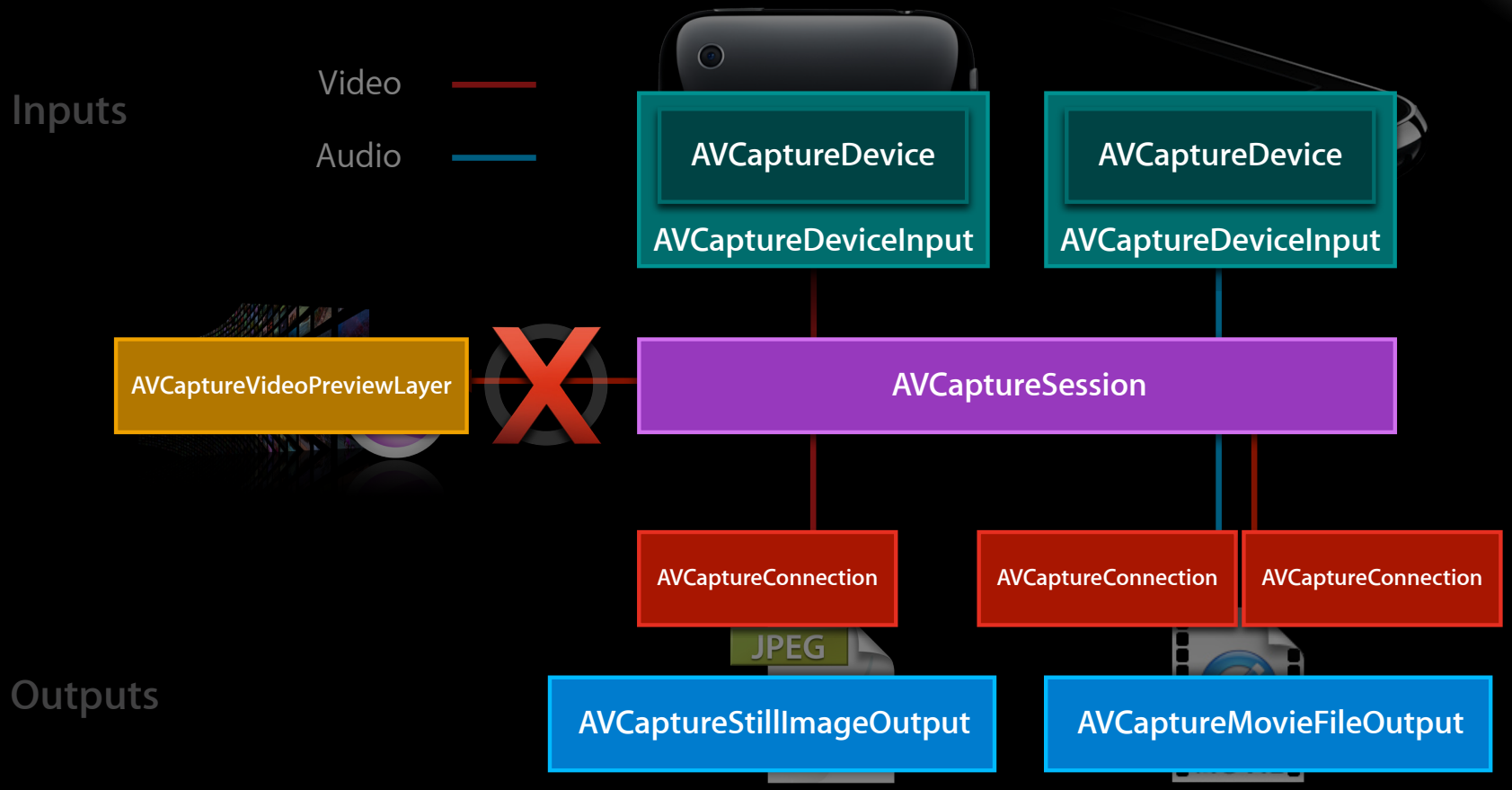
Using AVCaptureConnections

Advanced



Using AVCaptureConnections

Advanced



Purpose of AVCaptureConnection

- Identifies a specific stream of captured media
- Allows you to control exactly what is captured by enabling or disabling the connection
- An audio connection allows you to monitor audio levels
- A video connection allows you to affect the video delivered to the output
  - video orientation
  - video mirroring
- *Note: AVCaptureVideoDataOutput does not support `setVideoOrientation` or `setVideoMirrored` on its connection

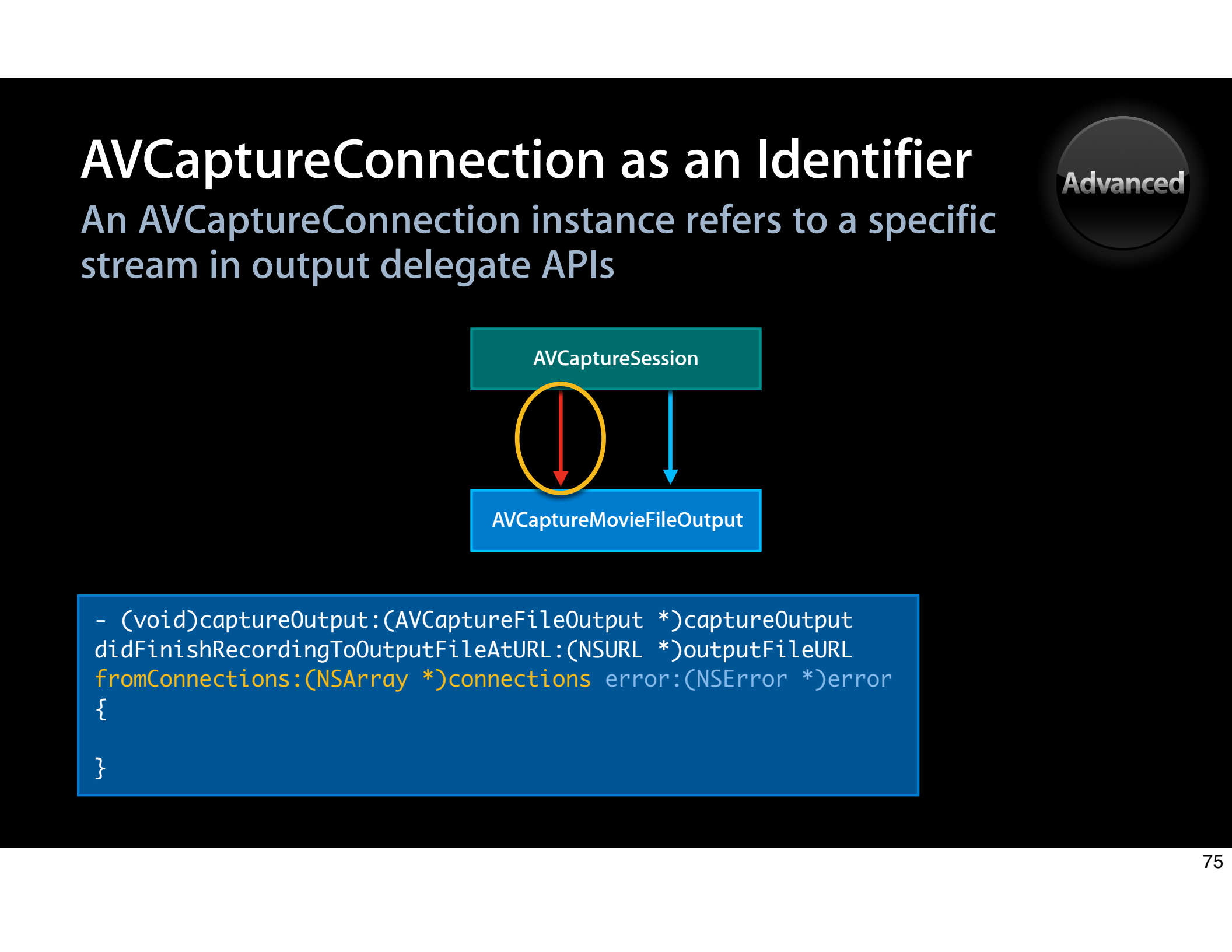
AVCaptureConnection as Status Monitor

- AVCaptureConnection exposes the current state of the media stream while the session is running
- *Refer to Code Snippet SP27*

```
AVCaptureConnection *connection;

// ... find an audio connection ...

NSArray *audioChans = connection.audioChannels;

for (AVCaptureAudioChannel *channel in audioChans) {
   float avg = channel.averagePowerLevel;
    float peak= channel.peakHoldLevel;

   // update level meter UI
}
```

AVCaptureConnection as an Identifier

An AVCaptureConnection instance refers to a specific stream in output delegate APIs




```
- (void)captureOutput:(AVCaptureFileOutput *)captureOutput
didFinishRecordingToOutputFileAtURL:(NSURL *)outputFileURL
fromConnections:(NSArray *)connections error:(NSError *)error
{

}
```

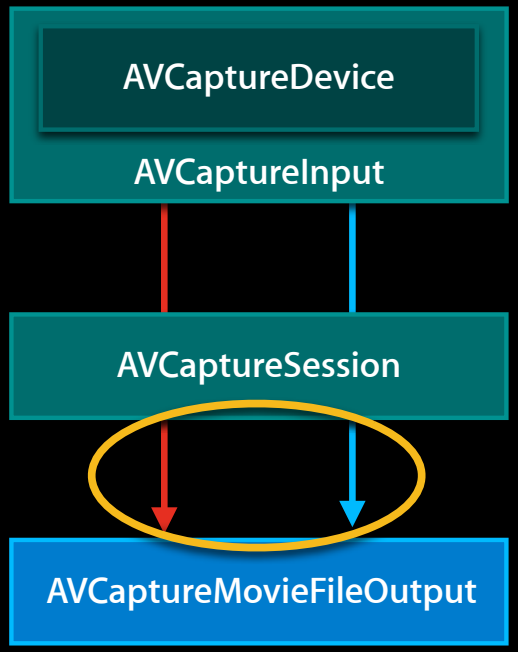
Finding AVCaptureConnections

- AVCaptureOutput implements

```
- (NSArray *)connections;
```

- Examples:

```
NSArray *movieFileOutputConnections
  = [movieFileOutput connections];
```



Disabling Connections

- Follow along in *Code Snippet SP26*

```
- (void)disableOutputConnection:(AVCaptureOutput *)output
        connectionWithMediaType:(NSString *)mediaType
{
    NSArray *connections = output.connections;
    for ( AVCaptureConnection *conn in connections ) {
        // Ask the current connection's input for its media type
        NSArray *inputPorts = conn.inputPorts;
        for ( AVCaptureInputPort *port in inputPorts ) {
            if ( [port.mediaType isEqual:mediaType] ) {
                // Found a match. Disable the connection.
                connection.enabled = NO;
                return;
            }
        }
    }
}
```

Summary

- Process YUV video frames from the camera
- Control the camera, take photos, and record movies
- Preview video from the camera to a Core Animation layer
- Process PCM audio data from the microphone to draw a waveform

- AVCaptureVideoDataOutput
- AVCaptureDevice, AVCaptureStillImageOutput, AVCaptureMovieFileOutput
- AVCaptureVideoPreviewLayer
- AVCaptureAudioDataOutput

More Information

Eryk Vershen
Media Technologies Evangelist
evershen@apple.com

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

Related Sessions

Discovering AV Foundation	Presidio Tuesday 2:00PM
Editing Media with AV Foundation	Presidio Tuesday 3:15PM
Incorporating the Camera and Photo Library in your App	Presidio Thursday 9:00AM
Discovering AV Foundation (Repeat)	Nob Hill Thursday 4:30PM

Labs

AV Foundation Lab	Graphics and Media Lab C Wednesday 9:00AM
AV Foundation Lab	Graphics and Media Lab B Friday 11:30AM
Core Animation Lab	Graphics and Media Lab D Thursday 3:15PM

WWDC10