Introducing AV Foundation Capture for Lion

Overview and best practices

Session 417

Brad Ford
Core Media Engineering

What You Will Learn

Only on Mac OS

New

- Why and when you should use AV Foundation capture
- The AV Foundation capture programming model
- AV Foundation capture differences on Lion and iOS

Sample Code for This Session

Only on Mac OS

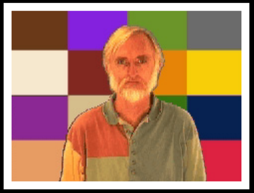
- AVRecorder
- AVScreenShack
- StopNGo (OSX edition)
- avvideowall

Materials available at:
https://developer.apple.com/wwdc/schedule/details.php?id=417

Capture on Mac OS

A brief history

- Video digitizer components introduced in QuickTime 1.0
- December, 1991 (20 years ago)
- Sequence Grabber came along in QuickTime 1.5
- These APIs still work today



Capture on Mac OS X

A brief history

- QTKit introduced modern Objective-C capture APIs in 2005
- A simpler programming model
- Sits atop CoreMedia
- Provides a legacy bridge to 32-bit ‘vdig’ capture devices
- These APIs also still work today

Capture on Mac OS X

AV Foundation

- Introduced in iOS 4.0 for iPhone, iPad, and iPod touch
- Interfaces are inspired by QTKit capture APIs
- Sits atop CoreMedia
- Encompasses all of QTKit capture API features
- Provides new features not available in QTKit
- Supports third-party CoreMedia IO video device drivers
- Available in Lion and forward

QTKit or AV Foundation?

Which one should I use for capture?

AV Foundation.

Really.

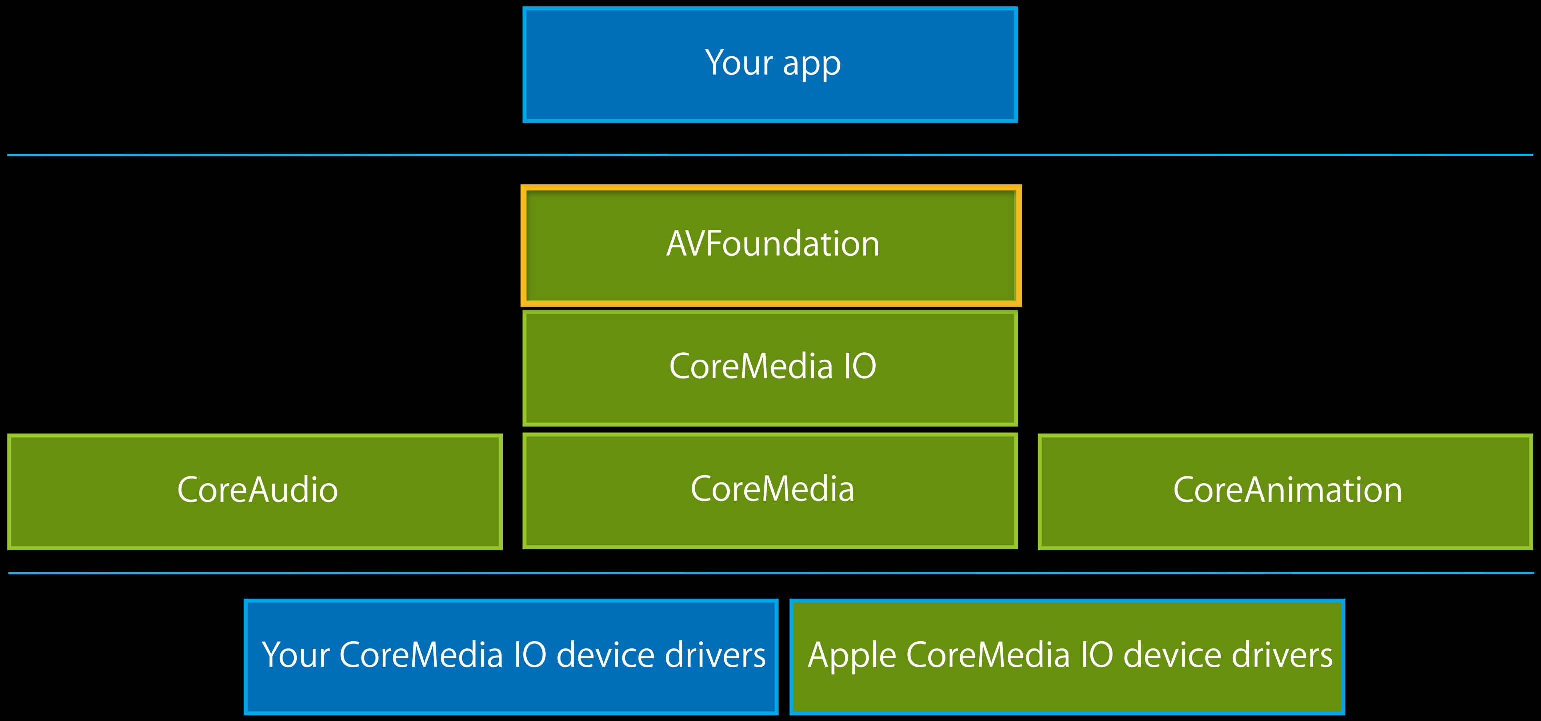
Unless…

Capture on Mac OS X

QTKit or AV Foundation?

- Continue to use QTKit capture APIs if
  - You need legacy ‘vdig’ support
  - You need legacy video encoder support
  - You need to run on Snow Leopard or earlier

Technology Framework


Your app
AVFoundation
CoreMedia IO
CoreAudio
CoreMedia
CoreAnimation
Your CoreMedia IO device drivers
Apple CoreMedia IO device drivers

New in Lion

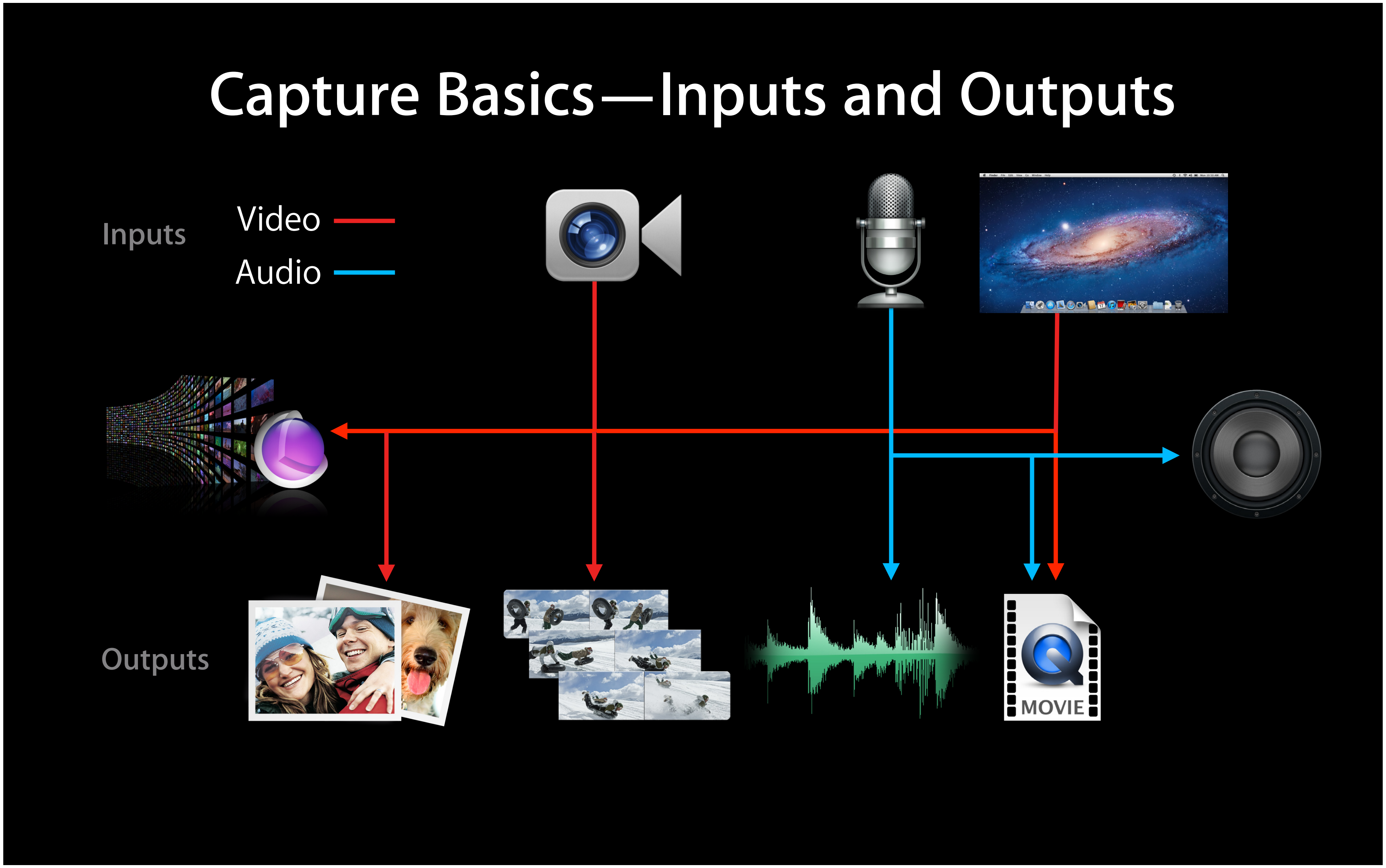
More features, more flexibility

New

- AVCaptureDevice enhancements
  - Discovery and selection of supported formats and frame rates
  - System-wide device sharing
  - Locking of shared devices for configuration
  - Support for closed captions
- Support for compressed AVCaptureVideoDataOutput
- Support for arbitrarily complex AVCaptureSessions
- AVCaptureScreenInput
- AVCaptureAudioPreviewOutput
- AVCaptureAudioFileOutput

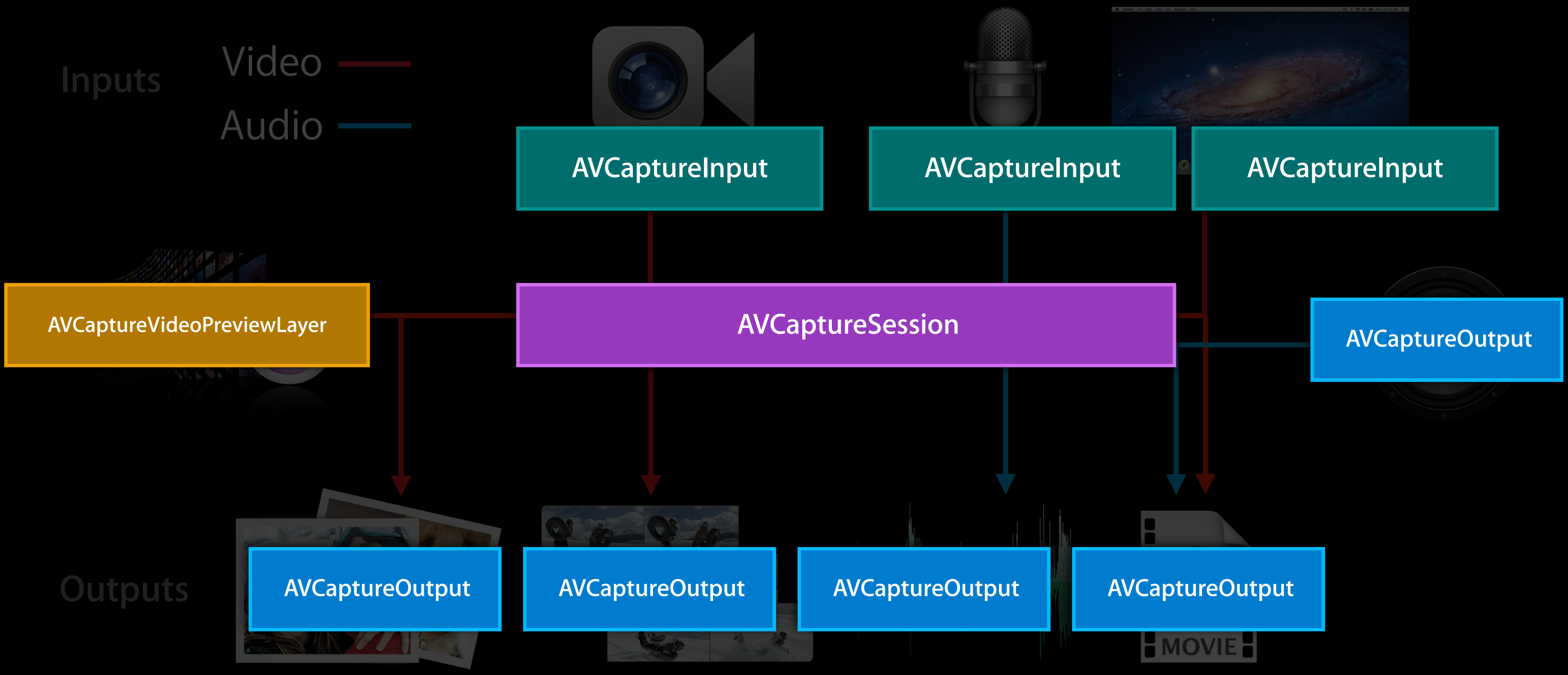
AV Foundation Capture

Programming model



Capture Basics—Inputs and Outputs

Capture Basics—Inputs and Outputs



Common Capture Use Cases

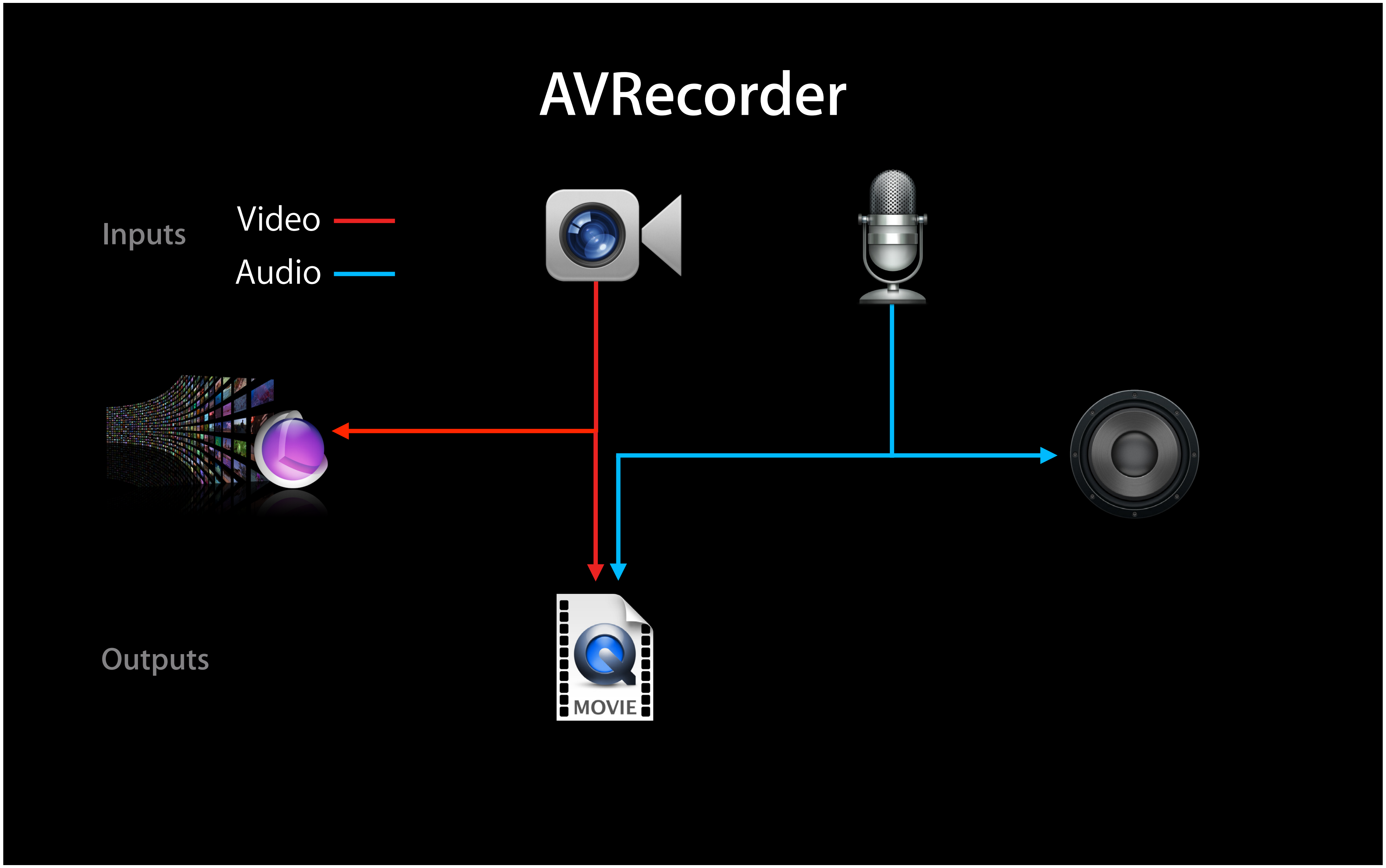
- Control the camera and record movies
- Capture the screen to a QuickTime movie
- Process frames from the camera
- Capture from multiple video devices simultaneously

Common Capture Use Cases

- Control the camera and record movies
- Capture the screen to a QuickTime movie
- Process frames from the camera
- Capture from multiple video devices simultaneously

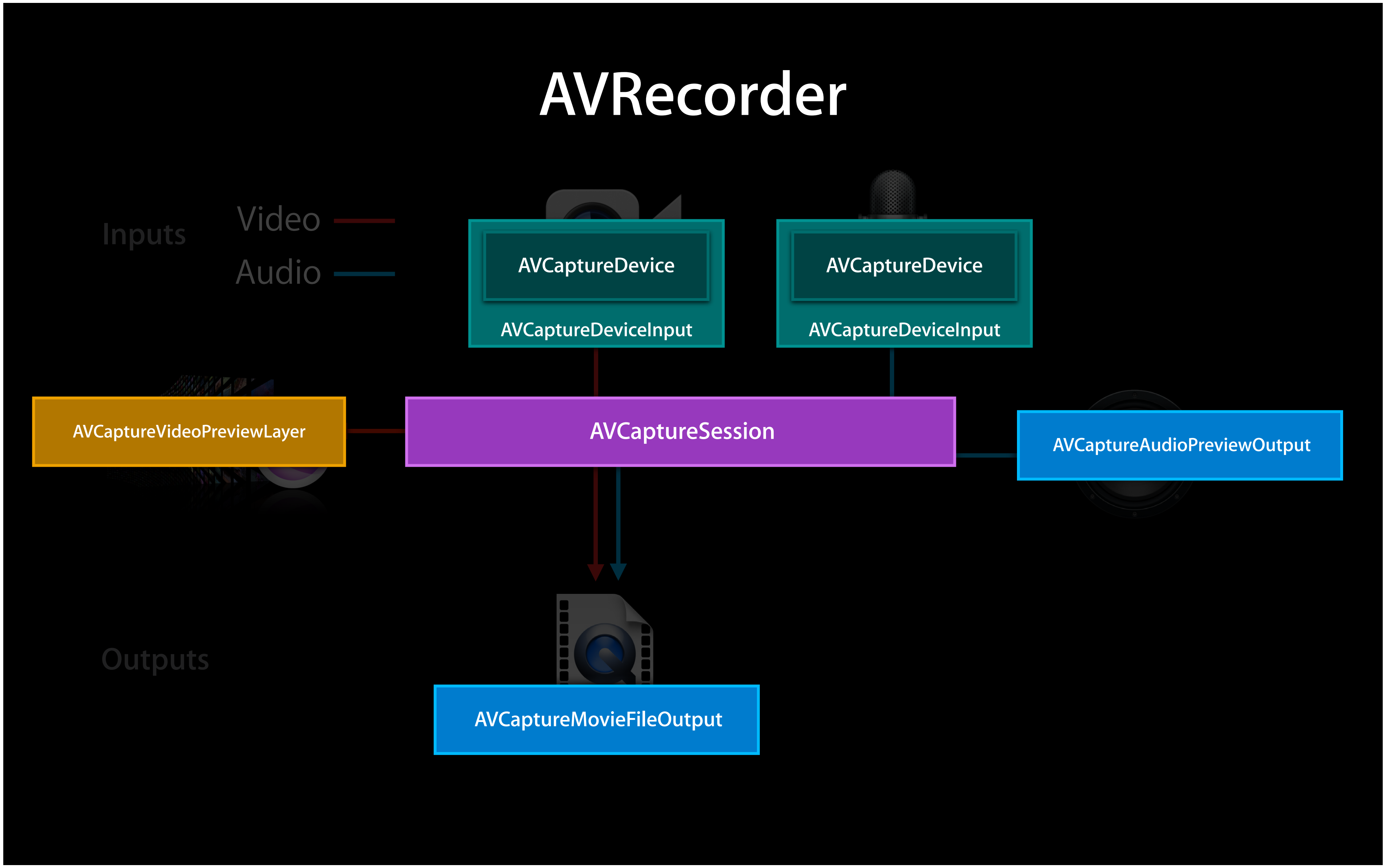
Demo—AVRecorder

Controlling the camera with AV Foundation



AVRecorder

AVRecorder


Inputs
Video
Audio
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureVideoPreviewLayer
AVCaptureSession
AVCaptureAudioPreviewOutput
Outputs
AVCaptureMovieFileOutput

AVRecorder

- 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];
```

AVRecorder

- Create and add outputs

```
AVCaptureMovieFileOutput *movieOutput = [AVCaptureMovieFileOutput new];
[session addOutput:movieOutput];

AVCaptureAudioPreviewOutput *audioOut = [AVCaptureAudioPreviewOutput new];
[session addOutput:audioOut];
```

- Create video preview layer and add it to a view

```
AVCaptureVideoPreviewLayer *layer = [AVCaptureVideoPreviewLayer
                                     layerWithSession:session];
[layer setFrame:[view bounds]];
[parentLayer addSublayer:layer];
```

AVRecorder

- Start the session

```
[session startRunning];
```

- You're done!

AVRecorder

- Enumerate AVCaptureDeviceFormats

```
for (AVCaptureDeviceFormat *format in [device formats]) {
   NSString *mediaType = [format mediaType];

   CMFormatDescriptionRef description = [format formatDescription];

   for (AVFrameRateRange *range in [format videoSupportedFrameRateRanges) {
      float minRate = [range minFrameRate];
      float maxRate = [range maxFrameRate];
    }
}
```

AVRecorder

- Select a device format

```
if ( YES == [device lockForConfiguration:&error] ) {
    [device setActiveFormat:theChosenFormat];
}
```

AVRecorder

Important AVCaptureDevice concepts

- AVCaptureDevice allows you to set the format and frame rate
- Not all devices expose formats and frame rates
- AVCaptureSession will try to configure devices automatically
- AVCaptureDevices are shared across the system
- Last one in wins
- Use `-lockForConfiguration:` to gain exclusive control
- `-unlockForConfiguration` to be a good OS X citizen
- Locked devices may still be used in other processes
- Code defensively!

AVRecorder

Switching cameras

- `-startRunning` and `-stopRunning` are synchronous
- An AVCaptureSession may be reconfigured while running
- Use `-beginConfiguration` and `-commitConfiguration`

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

[session removeInput:faceTimeCameraDeviceInput];
[session addInput:externalUSBCameraDeviceInput];

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

AVRecorder

Movie recording

- 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
{
     // Handle success or failure
}
```

AVCaptureMovieFileOutput

New

Enhanced movie file writing support

- AVCaptureMovieFileOutput supports frame accurate start/stop using the AVCaptureFileOutputDelegate

```
- (void)captureOutput:(AVCaptureFileOutput *)captureOutput
     didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
     fromConnection:(AVCaptureConnection *)connection
{
  // Inspect the sampleBuffer’s data, timestamps, metadata, etc.
  // Determine an exact record time
  [captureOutput startRecordingToOutputFileURL:url recordingDelegate:self];
}
```

- Use of AVCaptureFileOutputDelegate is optional

AVCaptureMovieFileOutput

New

Enhanced movie file writing support

- AVCaptureMovieFileOutput supports frame accurate pause and resume using the AVCaptureFileOutputDelegate as well

```
- (void)captureOutput:(AVCaptureFileOutput *)captureOutput
     didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
     fromConnection:(AVCaptureConnection *)connection
{
   [captureOutput pauseRecording];
   // or
   [captureOutput resumeRecording];
}
```

AVRecorder—Movie Recording

Setting limits

- `-setMaxRecordedDuration:`
- `-setMaxRecordedFileSize:`
- `-setMinFreeDiskSpaceLimit:`

AVRecorder—Movie Recording

Setting limits

- When a recording limit is reached, your delegate is called with an appropriate error

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

AVRecorder—Movie Recording

Early recording termination conditions

- `AVErrorDiskFull`
- `AVErrorDeviceWasDisconnected`
- `AVErrorMaximumDurationReached`
- `AVErrorMaximumFileSizeReached`

AVRecorder—Movie Recording

Metadata

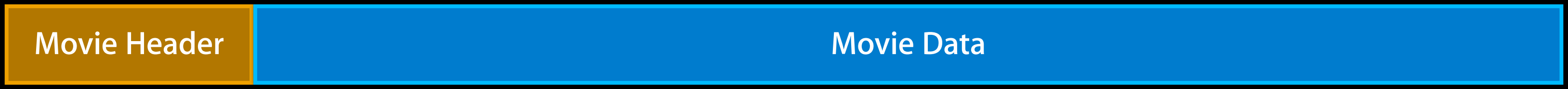
- You may set movie level metadata at any time while recording
- Allows for “slow” metadata, such as GPS location, to be acquired after the recording has started

```
NSMutableArray *metadata = [[NSMutableArray alloc] init];
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;
```

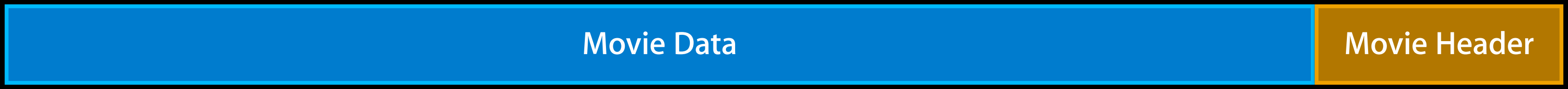
AVRecorder—Movie Recording

Movie fragments

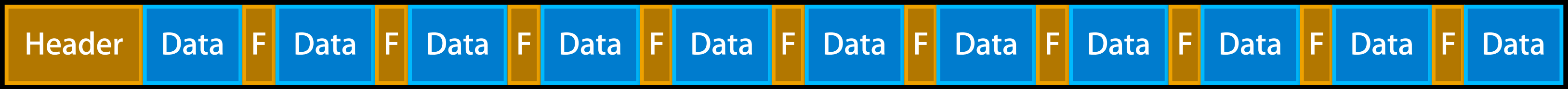
- Fast start QuickTime movie



- Non fast start (captured) QuickTime movie



- QuickTime movie with movie fragments



- Movie fragments = crash protection

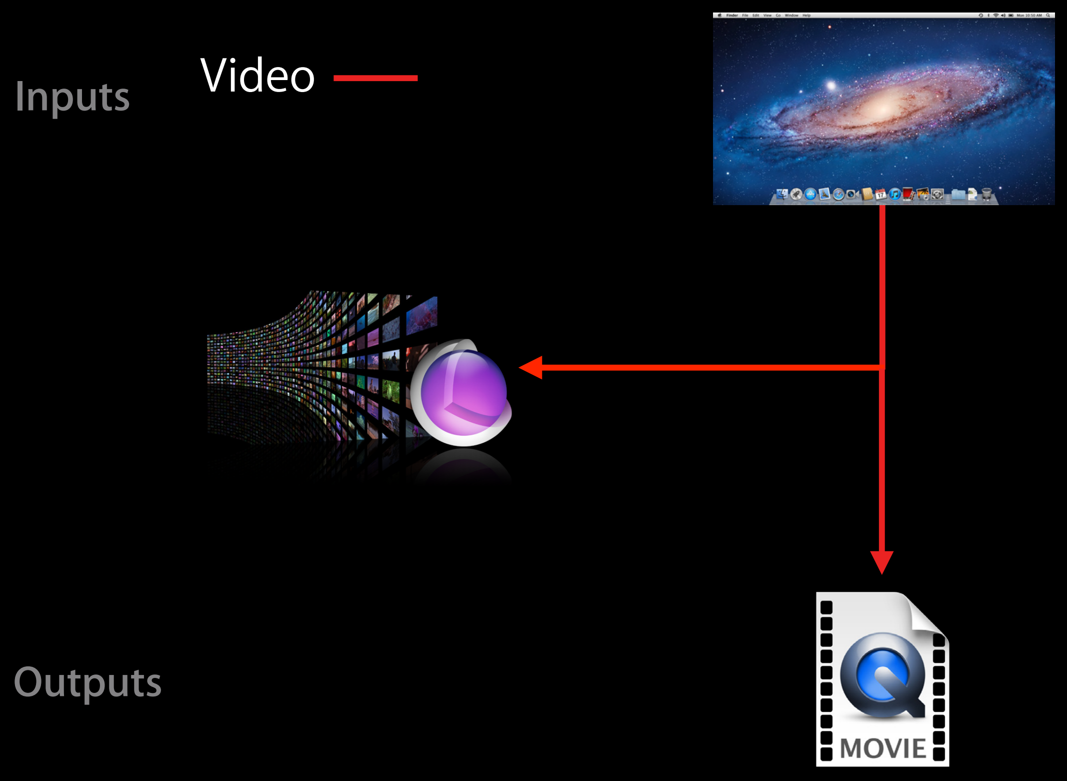
Common Capture Use Cases

- Control the camera and record movies
- Capture the screen to a QuickTime movie
- Process frames from the camera
- Capture from multiple video devices simultaneously

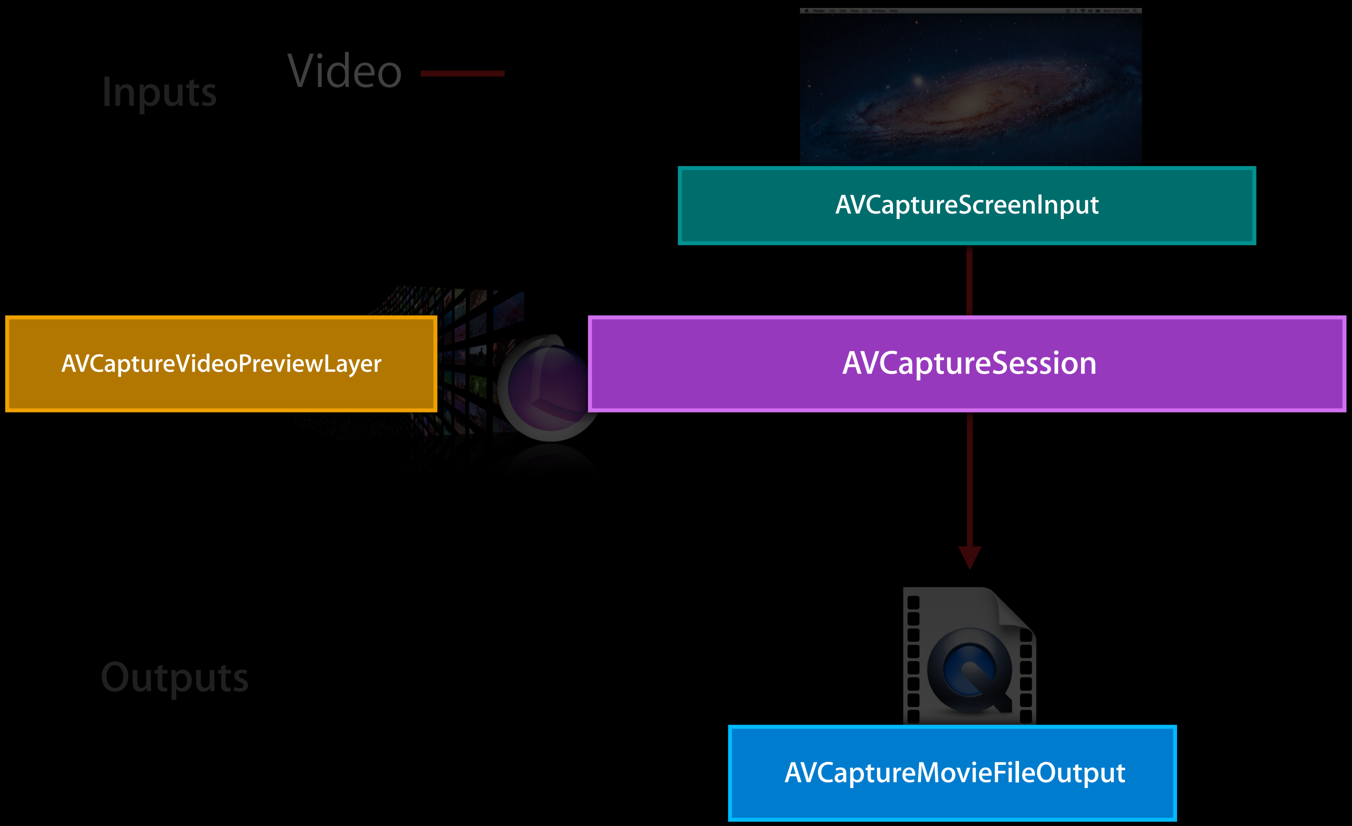
Demo—AVScreenShack

Capturing the screen to a QuickTime movie

AVScreenShack



AVScreenShack



AVCaptureScreenInput

Only on Mac OS

New

Features

- Fast frame grabbing (~60 fps on recent hardware)
- Efficient colorspace conversion to ‘2vuy’ for video applications
- Respects protected content

AVCaptureScreenInput

New

Usage

- Grabs frames from a specified CGDirectDisplayID
- Use `-setCropRect:` to capture a subsection of a display
- Use `-setScaleFactor:` to capture and scale (aspect is preserved)
- Use `-setMinFrameDuration:` to adjust max frame rate
- Use `-setCapturesMouseClicks:` to draw a ring around the mouse

Common Capture Use Cases

- Control the camera and record movies
- Capture the screen to a QuickTime movie
- Process frames from the camera
- Capture from multiple video devices simultaneously

Demo—StopNGo

Processing video frames and writing using AVAssetWriter

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

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

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

Output Settings

Understanding where and when format conversions occur

Output Settings

New

- All file and data outputs support customized output settings
- By default, AVCaptureSession's current `-sessionPreset` determines the baseline output settings for each AVCaptureOutput
- Set custom output settings to override the session's `-sessionPreset`
- Once set, output settings "stick"
- Set an empty dictionary of output settings for source passthrough
- Set nil output settings to restore the session preset's default settings

Output Settings

Sample video settings

New

```
 Settings are defined in <AVFoundation/AVVideoSettings.h>
 NSDictionary *outputSettings = [NSDictionary
        dictionaryWithObjectsAndKeys:
             AVVideoCodecH264, AVVideoCodecKey,
             [NSNumber numberWithInt:1280], AVVideoWidthKey,
             [NSNumber numberWithInt:720], AVVideoHeightKey,
             [NSDictionary dictionaryWithObjectsAndKeys:
               [NSNumber numberWithInt:10500000], AVVideoAverageBitRateKey,
               [NSNumber numberWithInt:1], AVVideoMaxKeyFrameIntervalKey,
               nil], AVVideoCompressionPropertiesKey,
             AVVideoScalingModeFit, AVVideoScalingModeKey,
                nil];
 [movieFileOutput setOutputSettings:outputSettings forConnection:
     [movieFileOutput connectionWithMediaType:AVMediaTypeVideo]];
```

Output Settings

New

Supported video formats

- `AVVideoCodecH264`
- `AVVideoCodecJPEG`
- `AVVideoCodecAppleProRes4444`
  - Preserves high bit depth source, up to 12 bits/ch
  - Mathematically lossless alpha channel
  - No subsampling
- `AVVideoCodecAppleProRes422`
  - Smaller files
  - Chroma subsampling
- Lots of CVPixelFormats, such as `kCVPixelFormatType_422YpCbCr8`

Video Scaling Modes

New

AVVideoScalingModeFit

- Crops source processing region
- Scales down if necessary, preserving aspect ratio
- Never ever upscales
- The default scaling mode for most capture session presets

Video Scaling Modes

AVVideoScalingModeFit

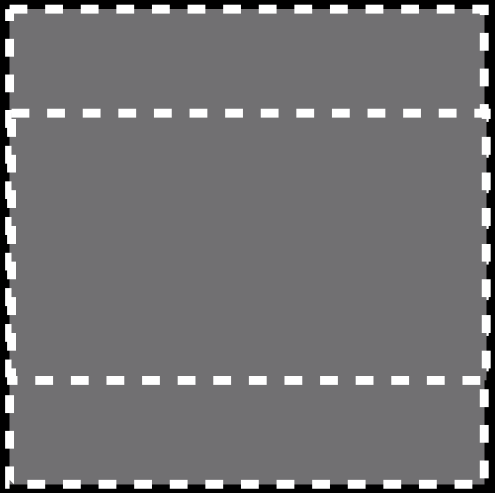
New

1280 x 720 Source Image

+

640 x 640 Settings

=



640 x 360

Video Scaling Modes

AVVideoScalingModeFit

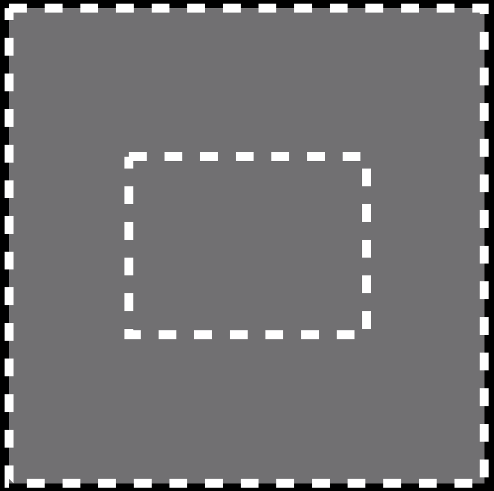
New

320 x 240 Source Image

+

=

640 x 640 Settings



Video Scaling Modes

New

AVVideoScalingModeResize

- “FunHouse Mode”
- Crops source to remove edge processing region
- Scales remainder to destination box
- Does not preserve aspect ratio

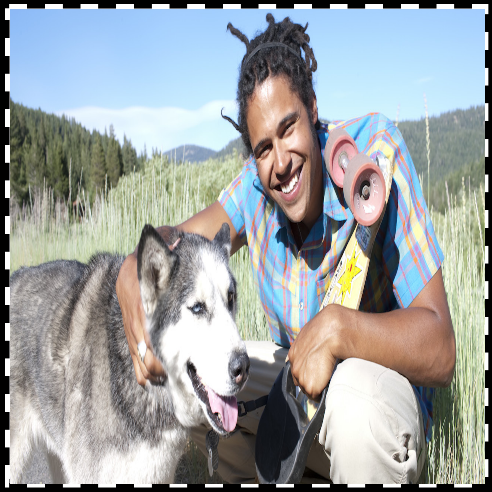
Video Scaling Modes

AVVideoScalingModeResize

New

1280 x 720 Source Image + 640 x 640 Settings

=



640 x 640

Video Scaling Modes

AVVideoScalingModeResize

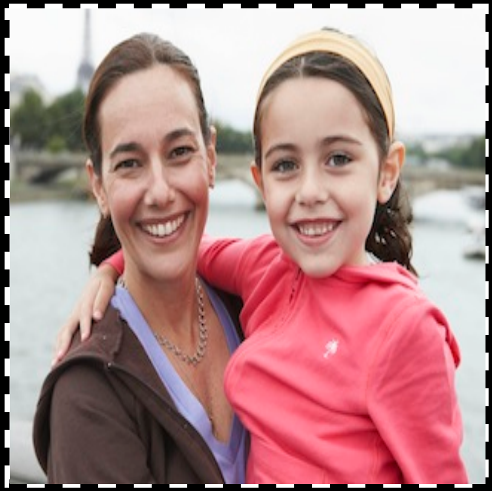
New

320 x 240 Source Image

+

640 x 640 Settings

=



640 x 640

Video Scaling Modes

New

AVVideoScalingModeResizeAspect

- “Letterbox Mode”
- Preserves source aspect ratio
- Fills remainder of destination box with black

Video Scaling Modes

AVVideoScalingModeResizeAspect

New

1280 x 720 Source Image

+

640 x 640 Settings

=

640 x 640 (with black bars)

Video Scaling Modes

AVVideoScalingModeResizeAspect

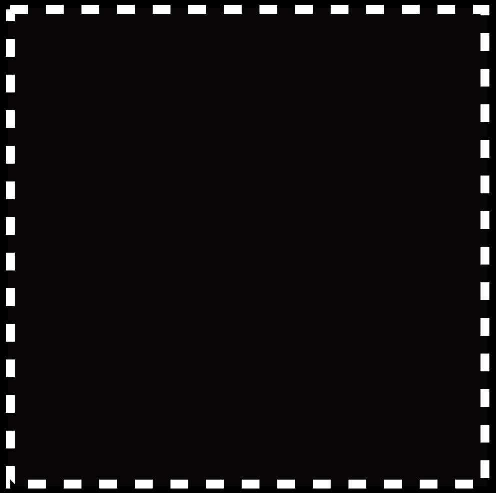
New

320 x 240 Source Image

+

640 x 640 Settings

=



640 x 640 (with black bars)

Video Scaling Modes

New

AVVideoScalingModeResizeAspectFill

- “Zoom Mode”
- Preserves source aspect ratio while scaling
- Crops picture to fit the destination box
- If source and destination aspect are not equal, some source pixels will be cropped out

Video Scaling Modes

New

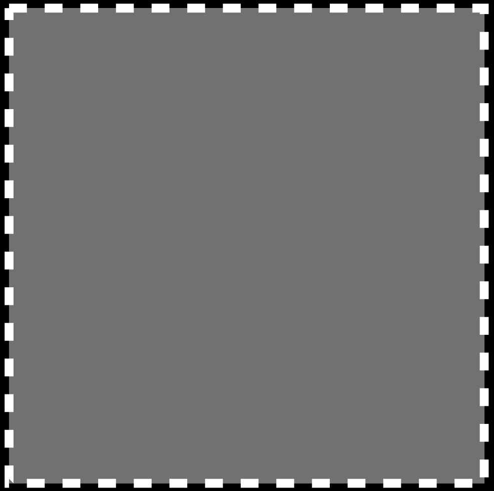
AVVideoScalingModeResizeAspectFill

1280 x 720 Source Image

+

640 x 640 Settings

=



640 x 640 (cropped)

Video Scaling Modes

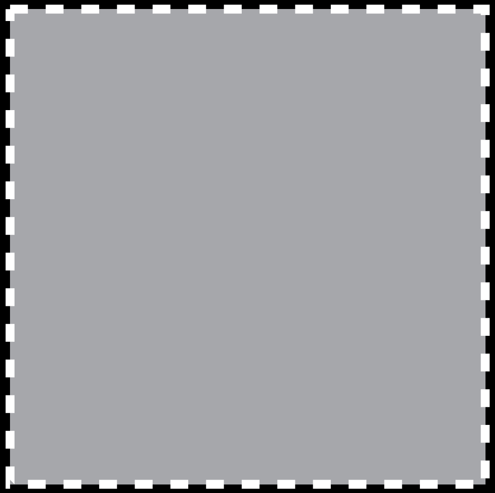
AVVideoScalingModeResizeAspectFill

New

320 x 240 Source Image

+

640 x 640 Settings



Video Scaling Modes

New

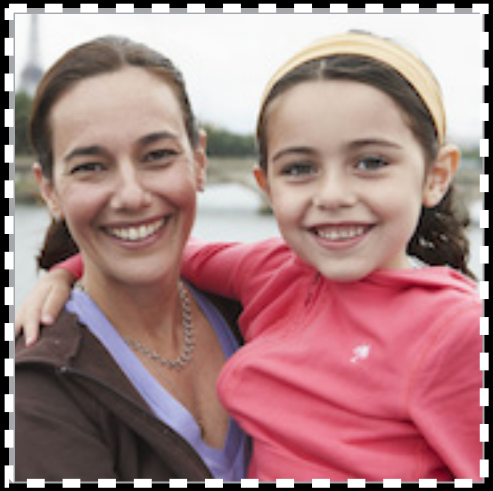
AVVideoScalingModeResizeAspectFill

320 x 240 Source Image

+

640 x 640 Settings

=



640 x 640 (cropped)

Common Capture Use Cases

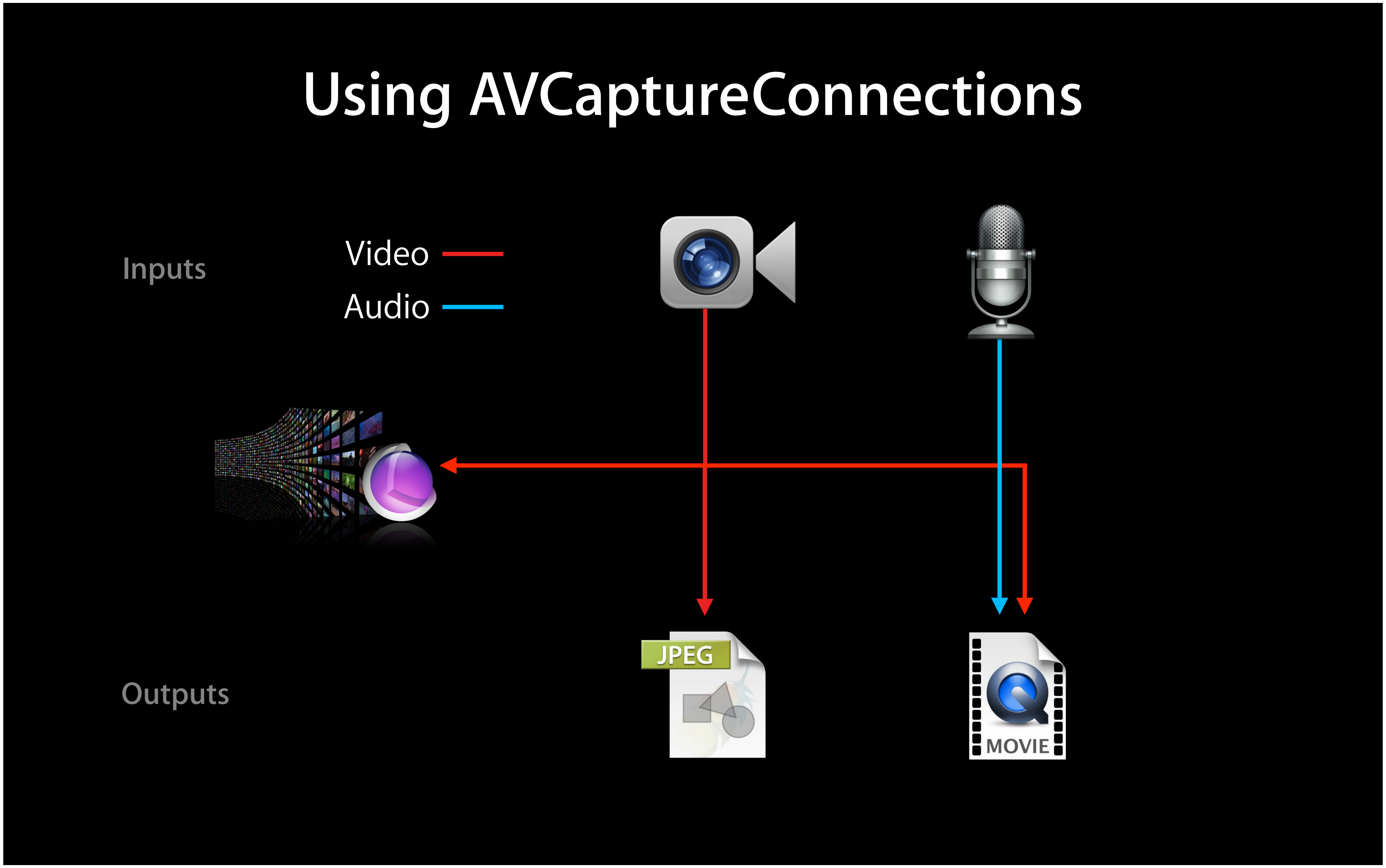
- Control the camera and record movies
- Capture the screen to a QuickTime movie
- Process frames from the camera
- Capture from multiple video devices simultaneously

Demo—avvideowall

Simultaneous capture from multiple devices

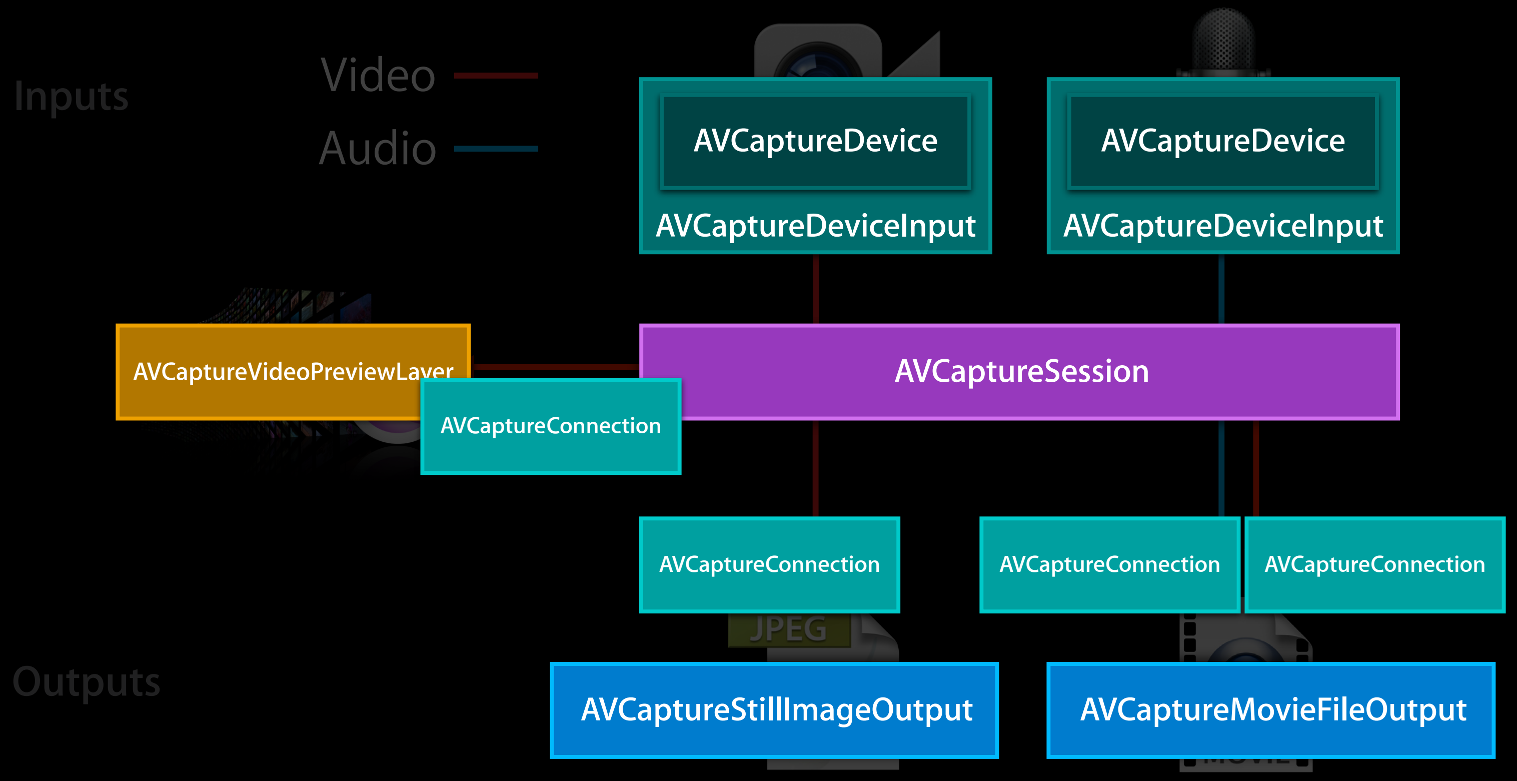
AVCaptureConnections

The glue that holds the inputs and outputs together



Using AVCaptureConnections

Using AVCaptureConnections



Purpose of AVCaptureConnection

- An AVCaptureInput has an array of AVCaptureInputPorts
- An AVCaptureConnection ties a specific AVCaptureInputPort to a specific AVCaptureOutput or AVCaptureVideoPreviewLayer

Purpose of AVCaptureConnection

- A video AVCaptureConnection allows you to manipulate the video delivered to the output
    - Orientation (rotation)
    - Mirroring
    - Deinterlacing
    - Frame rate limiting

Purpose of AVCaptureConnection

- An audio AVCaptureConnection lets you manipulate or monitor the audio data delivered to the output
  - Monitors audio levels
  - Enable or disable individual source audio channels
  - Adjust individual audio channel volume levels

AVCaptureConnection as Status Monitor

- AVCaptureConnection exposes the current state of the media stream while the session is running

```
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
}
```

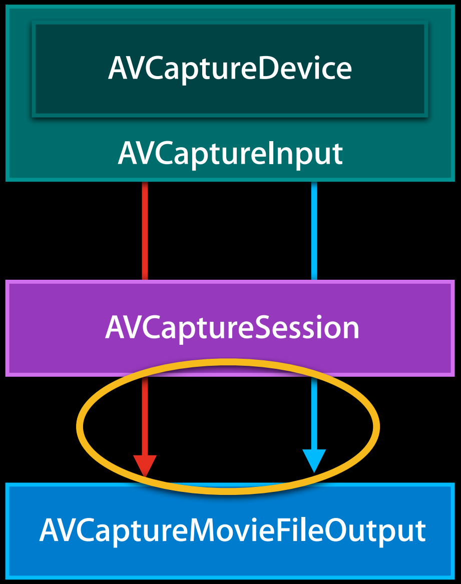
Finding AVCaptureConnections

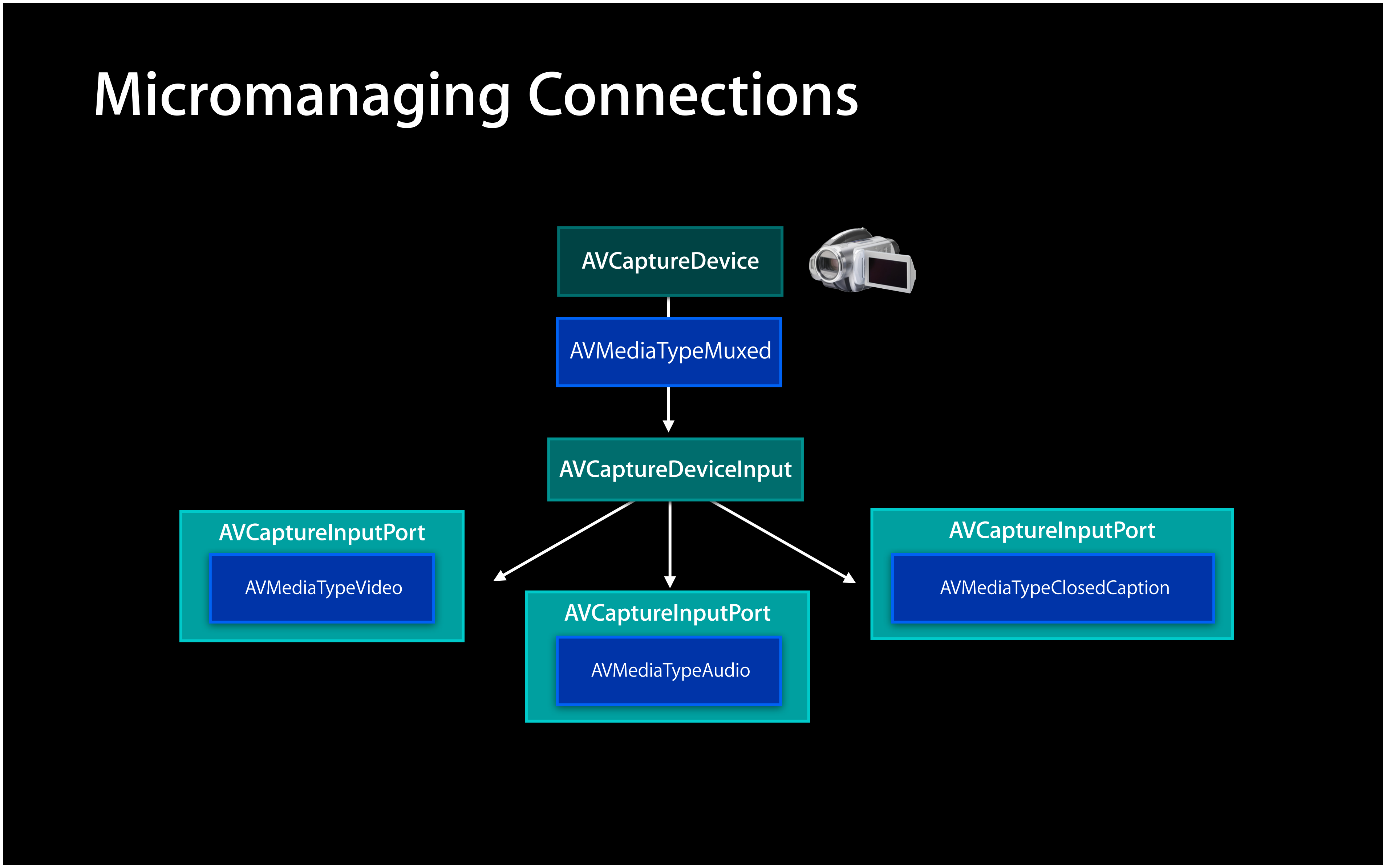
- AVCaptureOutput implements

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

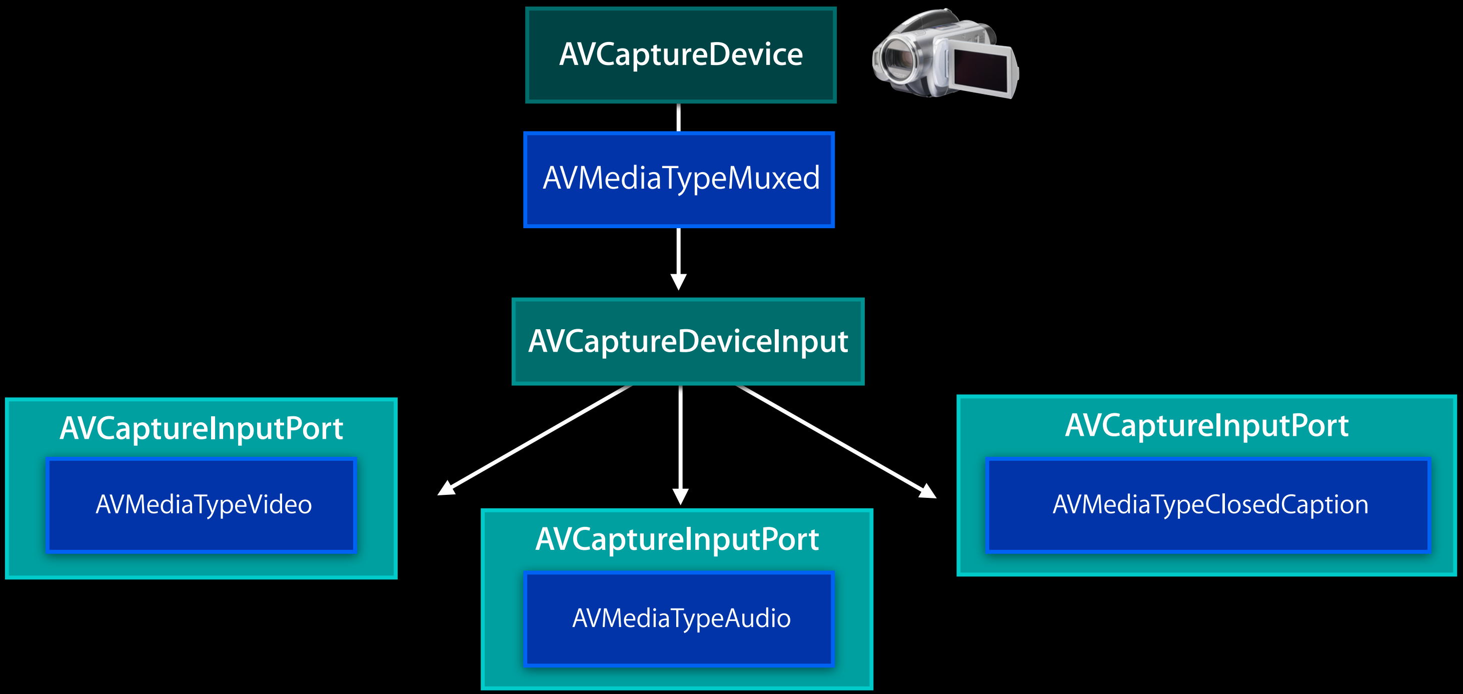
- Example

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





Micromanaging Connections



Micromanaging Connections

New

Power users can avoid implicit connections

- `[session addInputWithNoConnections:input]`
- `[session addOutputWithNoConnections:output]`
- `[avCapturePreviewLayer setSessionWithNoConnections:session]`
- `[AVCaptureConnection connectionWithInputPorts:ports output:output]`

Summary

- Use AV Foundation capture for all new development
- AV Foundation capture in Lion provides
  - More features, more functionality
  - Screen grabbing
  - Enhanced device control
  - Flexible output
  - Complex session support

More Information

Eryk Vershen
Media Technologies Evangelist
evershen@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

Exploring AV Foundation	Presidio Tuesday 2:00PM
AirPlay and External Displays in iOS apps	Presidio Tuesday 3:15PM
HTTP Live Streaming Update	Nob Hill Tuesday 4:30PM
Working with Media in AV Foundation	Pacific Heights Wednesday 2:00PM
Capturing from the Camera using AV Foundation on iOS 5	Pacific Heights Wednesday 4:30PM

Labs

AirPlay Lab	Graphics, Media & Games Lab B Wednesday 9:00AM-1:30PM
AV Foundation Lab	Graphics, Media & Games Lab C Wednesday 9:00AM-1:30PM
HTTP Live Streaming Lab	Graphics, Media & Games Lab D Wednesday 9:00AM-1:30PM
QT Kit Lab	Graphics, Media & Games Lab A Wednesday 9:00AM-1:30PM

AV Foundation Lab	Graphics, Media & Games Lab B Thursday 9:00AM-1:30PM
QuickTime Lab	Graphics, Media & Games Lab D Thursday 9:00AM-1:30PM
DAL Lab	Graphics, Media & Games Lab C Thursday 9:00AM-1:30PM

WWDC2011

One more thing...