What's New In Camera Capture

iOS 6 API enhancements and performance improvements

Session 520

Brad Ford
Core Media Engineering

What You Will Learn

What You Will Learn

- Performance improvements in Mac OS X 10.8

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6
- Solutions for performance problems in your capture app

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6
- Solutions for performance problems in your capture app
- Synchronizing motion data with video

What You Will Not Learn

What You Will Not Learn

- AV Foundation and CoreMedia basics

What You Will Not Learn

- AV Foundation and CoreMedia basics
- AV Foundation class hierarchy

What You Will Not Learn

- AV Foundation and CoreMedia basics
- AV Foundation class hierarchy
- Review last year's WWDC capture sessions at developer.apple.com

Sample Code for This Session

- AVRecorder (OS X)
- AVScreenShack (OS X)
- ‘StacheCam 2 (iOS)
- VideoSnake (iOS)
- AVCam (iOS)

Materials available at:
https://developer.apple.com/library/wwdc/mac/
https://developer.apple.com/library/wwdc/ios/

New AV Foundation Capture APIs

Mac OS X 10.8 (Mountain Lion) enhancements

Mac OS X 10.8 Capture Enhancements

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients
  - Better frame rates for AVCaptureVideoDataOutput clients

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients
  - Better frame rates for AVCaptureVideoDataOutput clients
  - ‘BGRA’ output with no intermediate ‘2vuy’ conversion

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients
  - Better frame rates for AVCaptureVideoDataOutput clients
  - 'BGRA' output with no intermediate '2vuy' conversion
  - Opt out for drawing the mouse cursor

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients
  - Better frame rates for AVCaptureVideoDataOutput clients
  - 'BGRA' output with no intermediate '2vuy' conversion
  - Opt out for drawing the mouse cursor
  - Mouse position metadata attached to video sample buffers

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients
  - Better frame rates for AVCaptureVideoDataOutput clients
  - 'BGRA' output with no intermediate '2vuy' conversion
  - Opt out for drawing the mouse cursor
  - Mouse position metadata attached to video sample buffers
  - Opt out for duplicate frame removal

Mac OS X 10.8 Capture Enhancements

- Major improvements to AVCaptureScreenInput performance
  - Lower latency for AVCaptureVideoDataOutput clients
  - Better frame rates for AVCaptureVideoDataOutput clients
  - 'BGRA' output with no intermediate '2vuy' conversion
  - Opt out for drawing the mouse cursor
  - Mouse position metadata attached to video sample buffers
  - Opt out for duplicate frame removal

See (updated) AVScreenShack sample code!

Mac OS X 10.8 Capture Enhancements

Mac OS X 10.8 Capture Enhancements

- Support for hardware accelerated H.264 encoding

Mac OS X 10.8 Capture Enhancements

- Support for hardware accelerated H.264 encoding
  - 2011 and newer Macs with SandyBridge / IvyBridge chipset

Mac OS X 10.8 Capture Enhancements

- Support for hardware accelerated H.264 encoding
  - 2011 and newer Macs with SandyBridge / IvyBridge chipset
  - Up to 1920x1088

Mac OS X 10.8 Capture Enhancements

- Support for hardware accelerated H.264 encoding
  - 2011 and newer Macs with SandyBridge / IvyBridge chipset
  - Up to 1920x1088
  - AVCaptureMovieFileOutput and AVAssetWriter (in real-time mode)

Mac OS X 10.8 Capture Enhancements

- Support for hardware accelerated H.264 encoding
  - 2011 and newer Macs with SandyBridge / IvyBridge chipset
  - Up to 1920x1088
  - AVCaptureMovieFileOutput and AVAssetWriter (in real-time mode)
  - No code changes required!

Mac OS X 10.8 Capture Enhancements

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression
  - AVCaptureMovieFileOutput supports frame accurate start and stop

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression
  - AVCaptureMovieFileOutput supports frame accurate start and stop
  - In Mac OS X 10.7, output compresses all the time

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression
  - AVCaptureMovieFileOutput supports frame accurate start and stop
  - In Mac OS X 10.7, output compresses all the time
  - In Mac OS X 10.8, you must opt in for frame accurate start

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression
  - AVCaptureMovieFileOutput supports frame accurate start and stop
  - In Mac OS X 10.7, output compresses all the time
  - In Mac OS X 10.8, you must opt in for frame accurate start

```
-(BOOL)captureOutputShouldProvideSampleAccurateRecordingStart:
```

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression
  - AVCaptureMovieFileOutput supports frame accurate start and stop
  - In Mac OS X 10.7, output compresses all the time
  - In Mac OS X 10.8, you must opt in for frame accurate start

    `-(BOOL)captureOutputShouldProvideSampleAccurateRecordingStart:`
  - Lowers power consumption when previewing

Mac OS X 10.8 Capture Enhancements

- Support for "just-in-time" compression
  - AVCaptureMovieFileOutput supports frame accurate start and stop
  - In Mac OS X 10.7, output compresses all the time
  - In Mac OS X 10.8, you must opt in for frame accurate start

    `-(BOOL)captureOutputShouldProvideSampleAccurateRecordingStart:`
  - Lowers power consumption when previewing

See (updated) AVRecorder sample code

Mac OS X 10.8 Capture Enhancements

Mac OS X 10.8 Capture Enhancements

- Newly published CoreMediaIO "DAL" SDK

Mac OS X 10.8 Capture Enhancements

- Newly published CoreMediaIO "DAL" SDK
- Includes sample device

Mac OS X 10.8 Capture Enhancements

- Newly published CoreMediaIO "DAL" SDK
- Includes sample device
- Makes life easier for video driver writers

Mac OS X 10.8 Capture Enhancements

- Newly published CoreMediaIO "DAL" SDK
- Includes sample device
- Makes life easier for video driver writers
- See us in the labs for more details!

Mac OS X 10.8 Capture Enhancements

- Newly published CoreMediaIO "DAL" SDK
- Includes sample device
- Makes life easier for video driver writers
- See us in the labs for more details!

SDK available at:
http://developer.apple.com/library/mac/samplecode/CoreMediaIO/index.html

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6
- Solutions for performance problems in your capture app
- Synchronizing motion data with video

The iOS Camera Ecosystem

How your app fits into the big picture

iOS Camera Ecosystem

iOS Camera Ecosystem

- Apple's Camera app saves photos and videos to a central library

iOS Camera Ecosystem

- Apple's Camera app saves photos and videos to a central library
- AssetsLibrary APIs allow your app to access this library

iOS Camera Ecosystem

- Apple's Camera app saves photos and videos to a central library
- AssetsLibrary APIs allow your app to access this library
  - Camera roll

iOS Camera Ecosystem

- Apple's Camera app saves photos and videos to a central library
- AssetsLibrary APIs allow your app to access this library
  - Camera roll
  - Synced assets from iTunes

iOS Camera Ecosystem

- Apple's Camera app saves photos and videos to a central library
- AssetsLibrary APIs allow your app to access this library
  - Camera roll
  - Synced assets from iTunes
  - Saved assets from Mail, your app, etc.

iOS Camera Ecosystem

- Apple's Camera app saves photos and videos to a central library
- AssetsLibrary APIs allow your app to access this library
  - Camera roll
  - Synced assets from iTunes
  - Saved assets from Mail, your app, etc.
  - Photo streams

iOS Camera Ecosystem

iOS Camera Ecosystem

- Photos and videos are personal, sensitive data

iOS Camera Ecosystem

- Photos and videos are personal, sensitive data
- iOS 6 devices now prompt user to grant access to the library

iOS Camera Ecosystem

- Photos and videos are personal, sensitive data
- iOS 6 devices now prompt user to grant access to the library

- Handle errors!

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6
- Solutions for performance problems in your capture app
- Synchronizing motion data with video

New AV Foundation Capture APIs

iOS 6 enhancements

New in iOS 6

New in iOS 6

- Video stabilization

New in iOS 6

- Video stabilization
- Real-time face detection

New in iOS 6

- Video stabilization
- Real-time face detection
- AVCaptureVideoPreviewLayer enhancements

New in iOS 6

- Video stabilization
- Real-time face detection
- AVCaptureVideoPreviewLayer enhancements

Video Stabilization

Video Stabilization

- Video stabilization steadies shaky shots

Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts

Video Stabilization

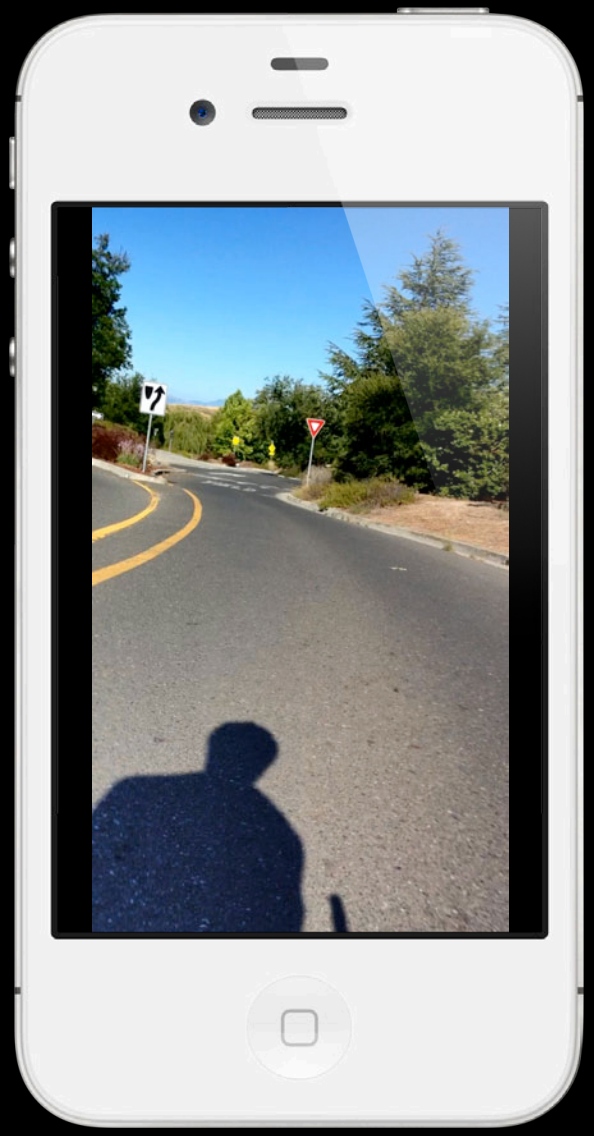
- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts

Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts

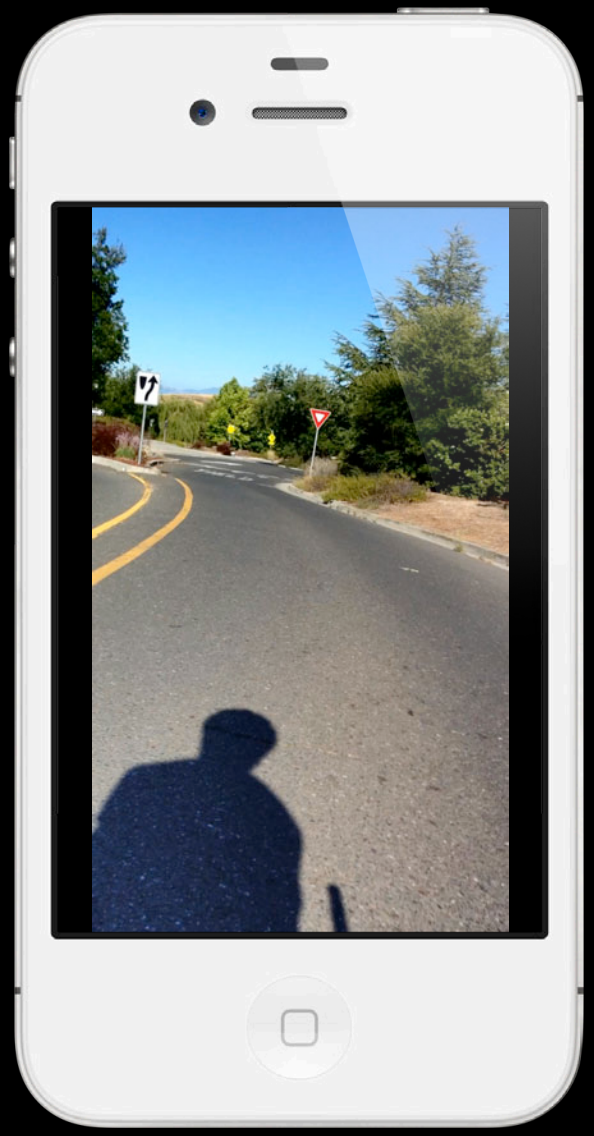
Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts



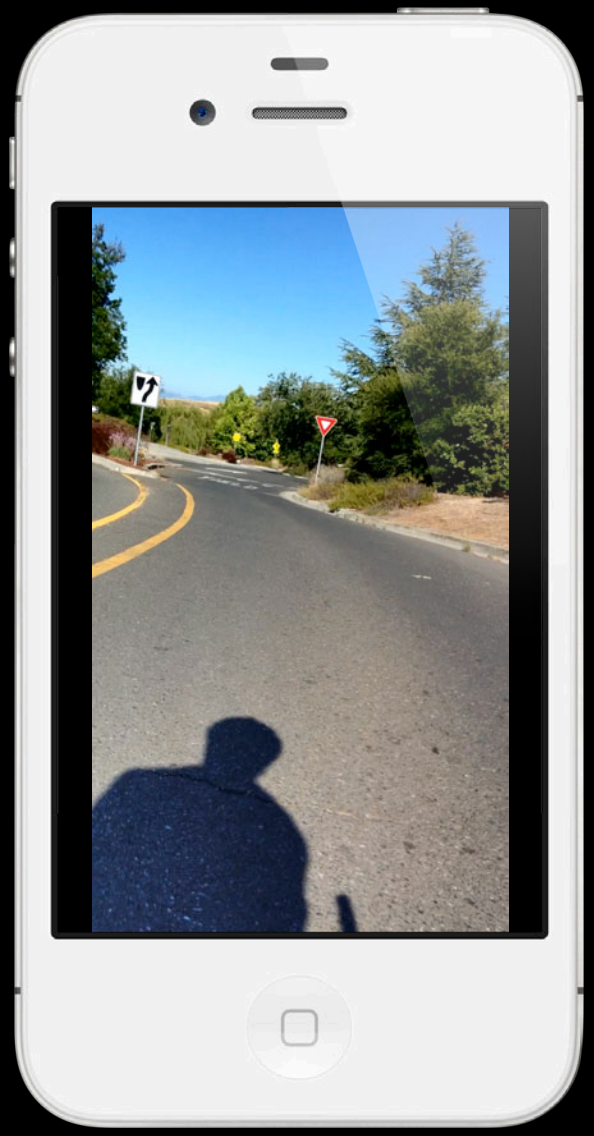
Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts



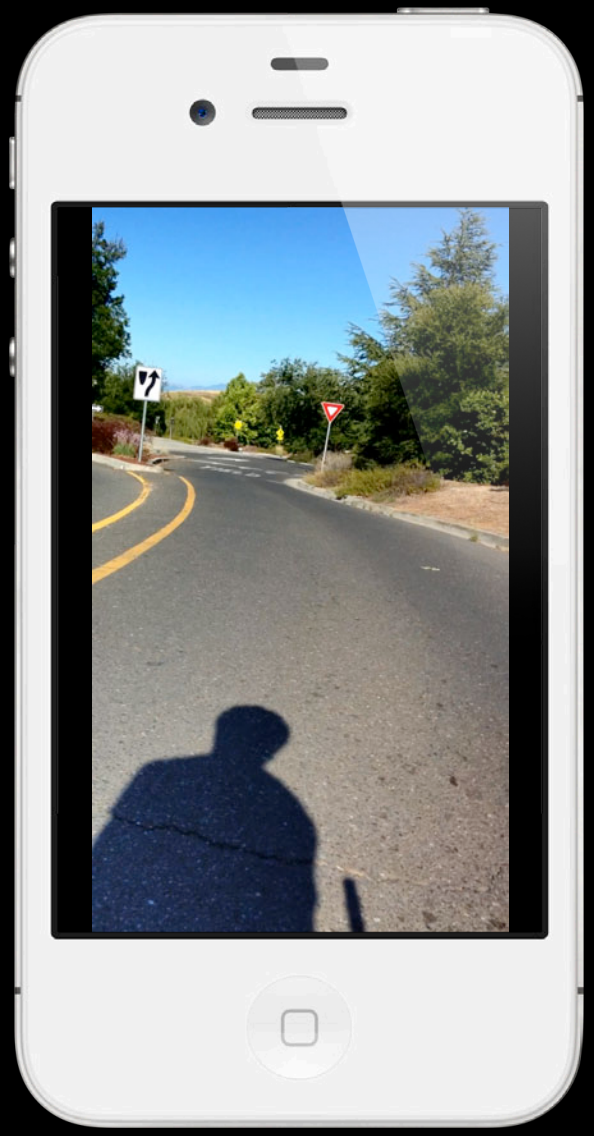
Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts



Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts

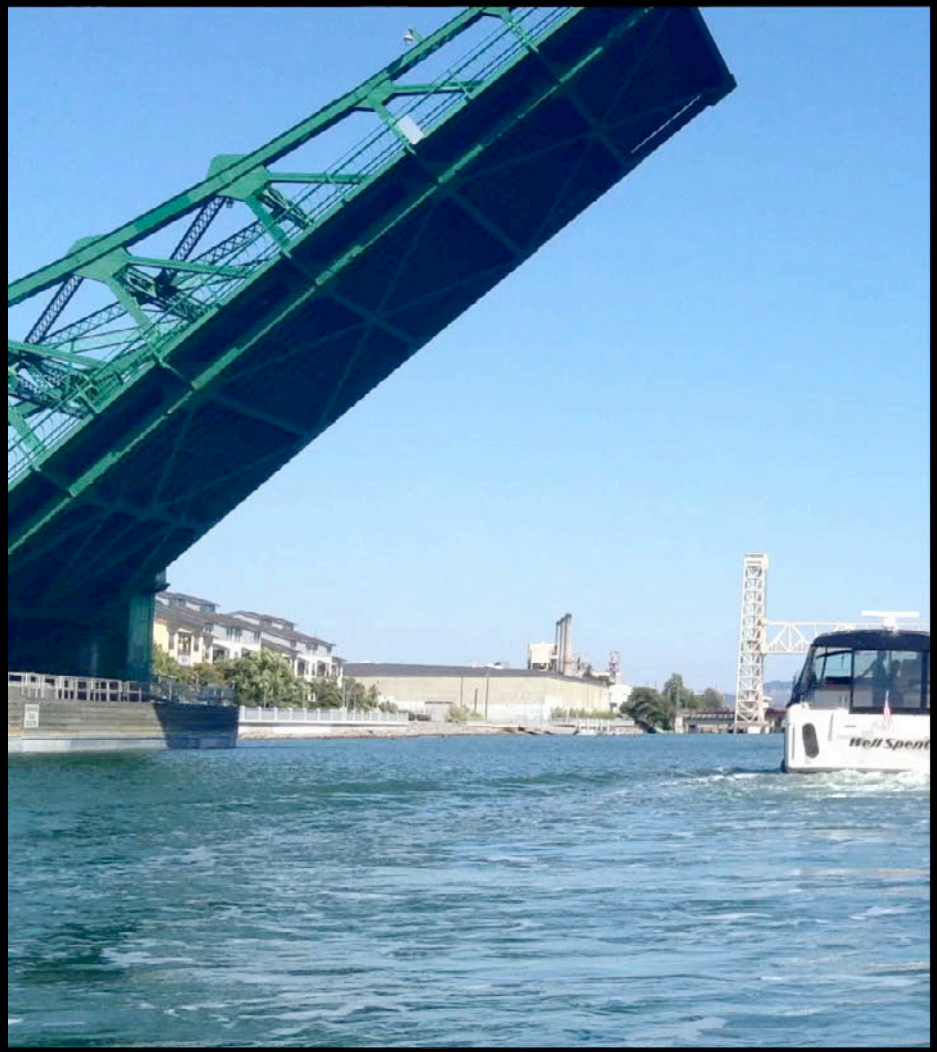


Video Stabilization

- Video stabilization steadies shaky shots
- Compensates for rolling shutter artifacts

Demo

Video stabilization



Before

After

Video Stabilization

Why use it?

Video Stabilization

Why use it?

- Camera phones are susceptible to shake

Video Stabilization

Why use it?

- Camera phones are susceptible to shake
- HD resolution recordings are especially susceptible to rolling shutter

Video Stabilization

Why use it?

- Camera phones are susceptible to shake
- HD resolution recordings are especially susceptible to rolling shutter
- Stabilization saves otherwise unusable footage

Video Stabilization

Why use it?

- Camera phones are susceptible to shake
- HD resolution recordings are especially susceptible to rolling shutter
- Stabilization saves otherwise unusable footage
- It works in real-time

Video Stabilization

Why not use stabilization?

Video Stabilization

Why not use stabilization?

- Stabilization alters the pixels

Video Stabilization

Why not use stabilization?

- Stabilization alters the pixels
- Output no longer matches preview layer

Video Stabilization

Why not use stabilization?

- Stabilization alters the pixels
- Output no longer matches preview layer
- It may not interoperate well with other pixel processing algorithms

Video Stabilization

Why not use stabilization?

- Stabilization alters the pixels
- Output no longer matches preview layer
- It may not interoperate well with other pixel processing algorithms
- Stabilization adds latency to video data output

Video Stabilization

Supported platforms

iPhone 4S	✓
The new iPad	✓

Video Stabilization

Compatibility

Video Stabilization

Compatibility

- All HD video resolutions are compatible

```
AVCaptureSessionPresetHigh
AVCaptureSessionPreset1920x1080
AVCaptureSessionPreset1280x720
AVCaptureSessionPresetiFrame1280x720
AVCaptureSessionPresetiFrame960x540
```

Video Stabilization

Compatibility

Video Stabilization

Compatibility

- Does NOT work with front camera

Video Stabilization

Compatibility

- Does NOT work with front camera
- Does NOT work with AVCaptureStillImageOutput

Video Stabilization

Compatibility

- Does NOT work with front camera
- Does NOT work with AVCaptureStillImageOutput
- Does NOT work with AVCaptureVideoPreviewLayer

Video Stabilization

iOS 5 behavior

Video Stabilization

iOS 5 behavior

- AVCaptureMovieFileOutput always stabilizes 1080p video

Video Stabilization

iOS 5 behavior

- AVCaptureMovieFileOutput always stabilizes 1080p video
- AVCaptureMovieFileOutput never stabilizes any other resolution

Video Stabilization

iOS 5 behavior

- AVCaptureMovieFileOutput always stabilizes 1080p video
- AVCaptureMovieFileOutput never stabilizes any other resolution
- AVCaptureVideoDataOutput never stabilizes video

Video Stabilization

iOS 5 behavior

- AVCaptureMovieFileOutput always stabilizes 1080p video
- AVCaptureMovieFileOutput never stabilizes any other resolution
- AVCaptureVideoDataOutput never stabilizes video
- No API to opt in or out

Video Stabilization

iOS 6 behavior

Video Stabilization

iOS 6 behavior

- Apps linked before iOS 6 continue to get the iOS 5 behavior

Video Stabilization

iOS 6 behavior

- Apps linked before iOS 6 continue to get the iOS 5 behavior
- Apps linked on or after iOS 6 must opt in for stabilization

Video Stabilization

iOS 6 behavior

- Apps linked before iOS 6 continue to get the iOS 5 behavior
- Apps linked on or after iOS 6 must opt in for stabilization
- Both movie file output and video data output support stabilization

Video Stabilization

Opting in

Video Stabilization

Opting in

- Create an AVCaptureSession

Video Stabilization

Opting in

- Create an AVCaptureSession
- Add an AVCaptureDeviceInput

Video Stabilization

Opting in

- Create an AVCaptureSession
- Add an AVCaptureDeviceInput
- Add an AVCaptureMovieFileOutput or AVCaptureVideoDataOutput

Video Stabilization

Opting in

- Create an AVCaptureSession
- Add an AVCaptureDeviceInput
- Add an AVCaptureMovieFileOutput or AVCaptureVideoDataOutput
- Get the output's video connection

```
AVCaptureConnection *c = [output connectionWithMediaType:AVMediaTypeVideo];
```

Video Stabilization

Opting in

- Create an AVCaptureSession
- Add an AVCaptureDeviceInput
- Add an AVCaptureMovieFileOutput or AVCaptureVideoDataOutput
- Get the output's video connection

```
AVCaptureConnection *c = [output connectionWithMediaType:AVMediaTypeVideo];
```

- Opt in for video stabilization when available

```
if ( [c isVideoStabilizationSupported] )
  [c setEnablesVideoStabilizationWhenAvailable:YES];
```

Video Stabilization

Opting in

- Create an AVCaptureSession
- Add an AVCaptureDeviceInput
- Add an AVCaptureMovieFileOutput or AVCaptureVideoDataOutput
- Get the output's video connection

```
AVCaptureConnection *c = [output connectionWithMediaType:AVMediaTypeVideo];
```

- Opt in for video stabilization when available

```
if ( [c isVideoStabilizationSupported] )
  [c setEnablesVideoStabilizationWhenAvailable:YES];
```

- Key-value observe the connection's `@"videoStabilizationEnabled"` property

Video Stabilization

Gotchas

- When inputs or outputs are added, connections are implicitly formed between compatible input ports and outputs

Video Stabilization

Gotchas

- When inputs or outputs are added, connections are implicitly formed between compatible input ports and outputs

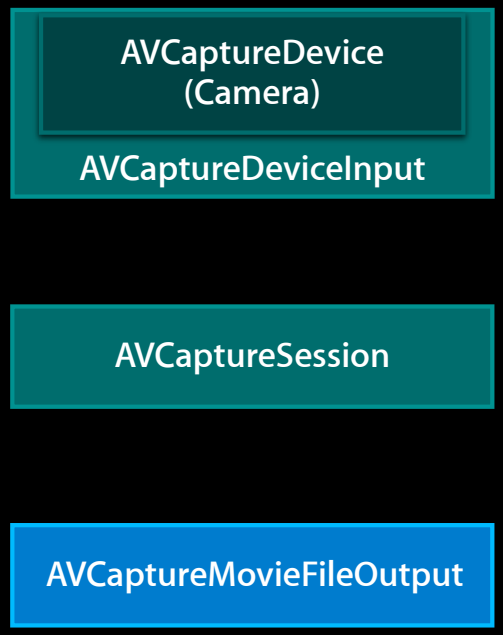
AVCaptureSession

AVCaptureMovieFileOutput

Video Stabilization

Gotchas

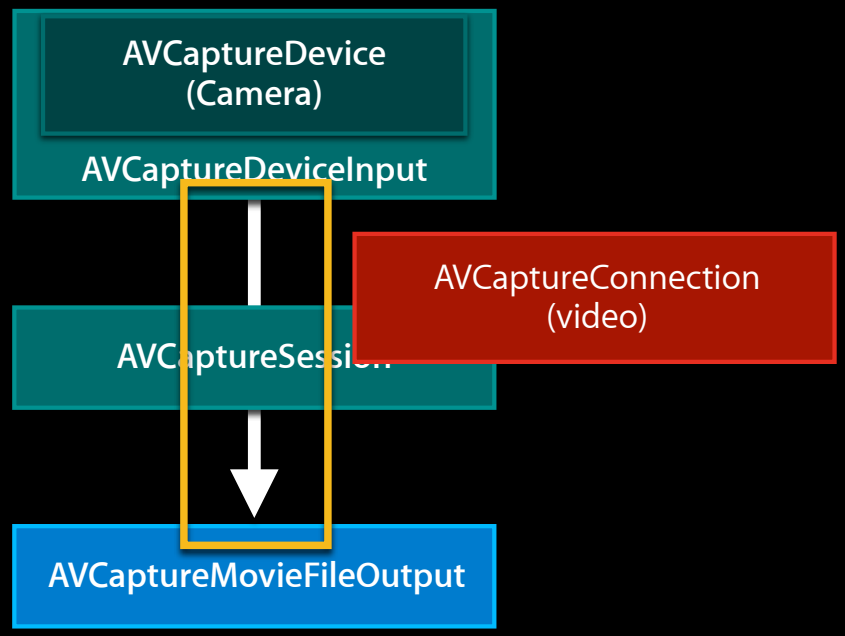
- When inputs or outputs are added, connections are implicitly formed between compatible input ports and outputs



Video Stabilization

Gotchas

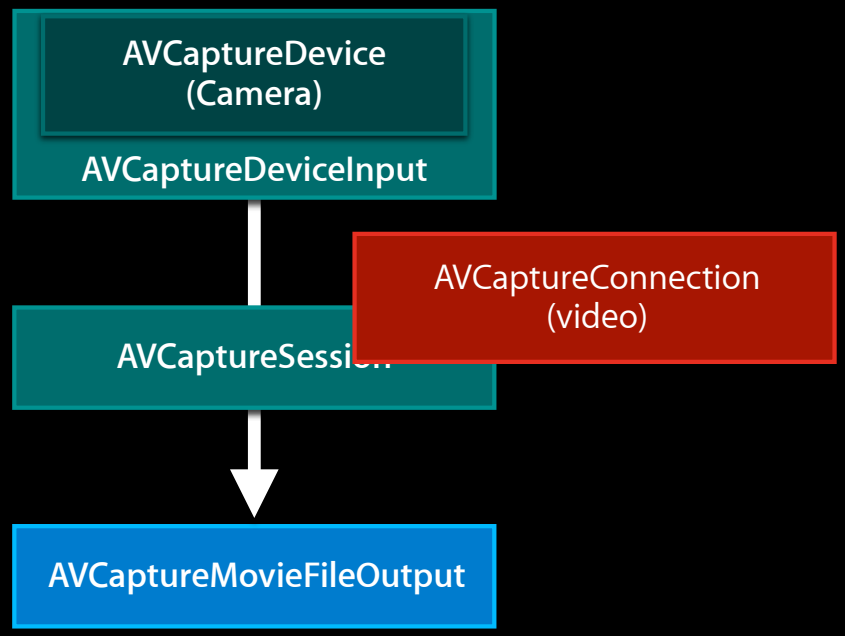
- When inputs or outputs are added, connections are implicitly formed between compatible input ports and outputs



Video Stabilization

Gotchas

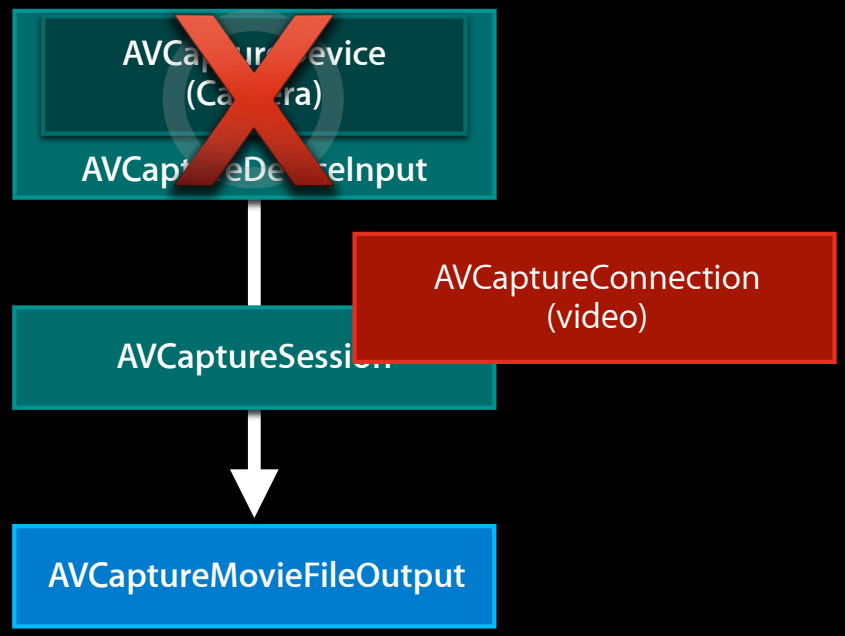
- Connections are implicitly severed when inputs or outputs are removed



Video Stabilization

Gotchas

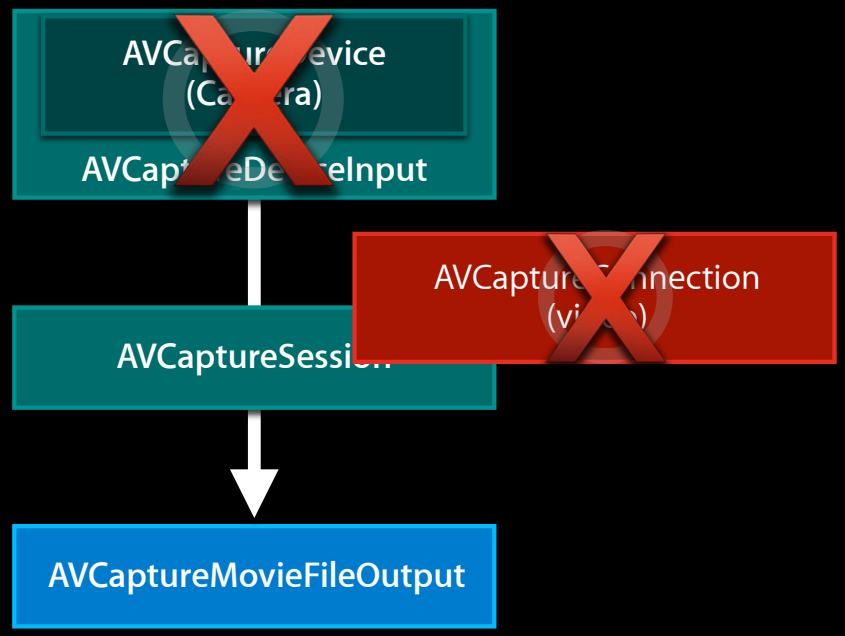
- Connections are implicitly severed when inputs or outputs are removed



Video Stabilization

Gotchas

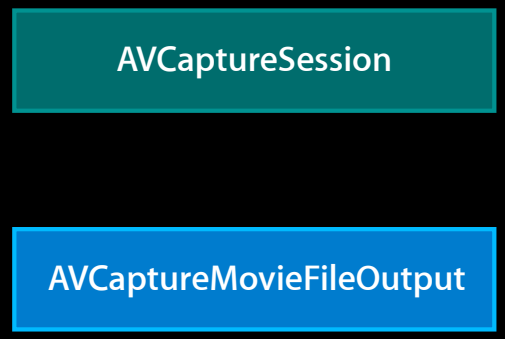
- Connections are implicitly severed when inputs or outputs are removed



Video Stabilization

Gotchas

- Connections are implicitly severed when inputs or outputs are removed



Video Stabilization

Gotchas

- Connections are implicitly severed when inputs or outputs are removed

Video Stabilization

Gotchas

- Connections are implicitly severed when inputs or outputs are removed
- When you switch cameras, all your connection settings are lost

Video Stabilization

Gotchas

- Connections are implicitly severed when inputs or outputs are removed
- When you switch cameras, all your connection settings are lost
- After adding your new input, you must configure its new connection

Video Stabilization

Gotchas

- Connections are implicitly severed when inputs or outputs are removed
- When you switch cameras, all your connection settings are lost
- After adding your new input, you must configure its new connection
- Use AVCaptureSession's `-beginConfiguration` / `-commitConfiguration` when reconfiguring inputs or outputs to a session

Video Stabilization

Gotchas

- Connections are implicitly severed when inputs or outputs are removed
- When you switch cameras, all your connection settings are lost
- After adding your new input, you must configure its new connection
- Use AVCaptureSession's `-beginConfiguration` / `-commitConfiguration` when reconfiguring inputs or outputs to a session

See updated AVCam sample code

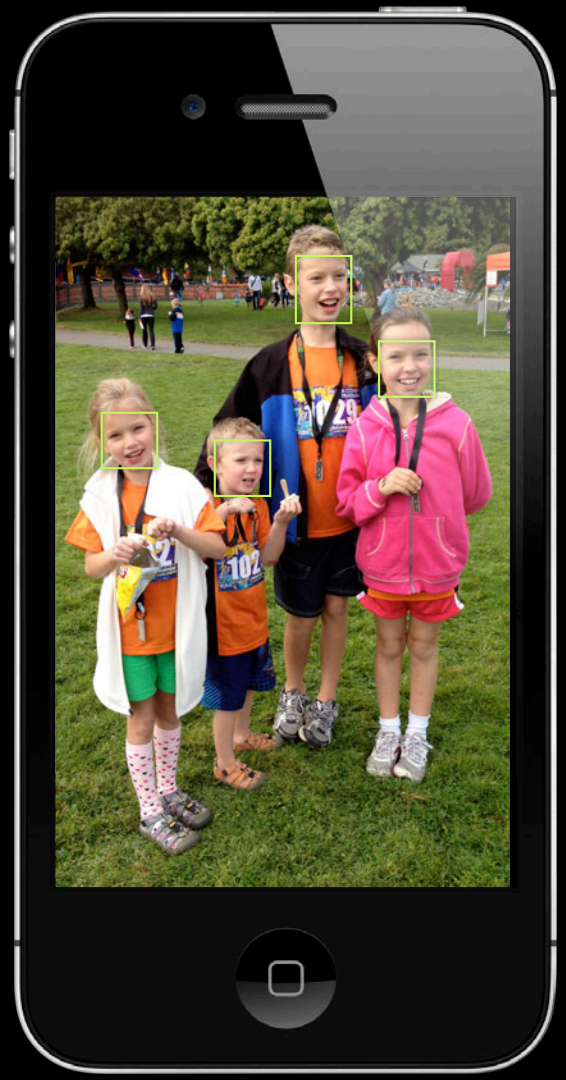
New in iOS 6

- Video stabilization
- Real-time face detection
- AVCaptureVideoPreviewLayer enhancements

Face Detection

Face Detection

- Scans for faces in real-time



Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face

Face Detection

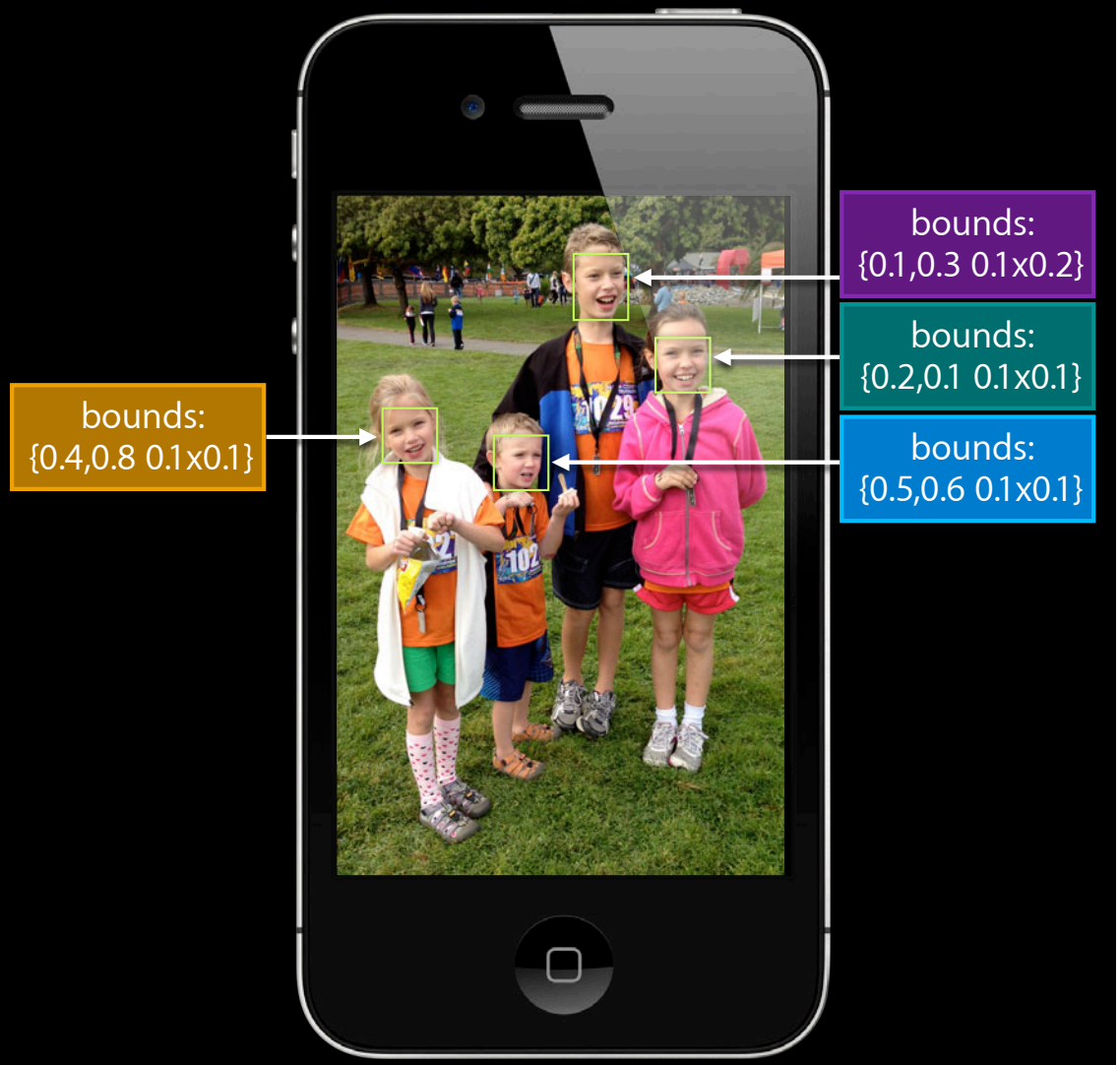
- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face



Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle
- Determines the yaw angle

Face Detection

- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle
- Determines the yaw angle

Face Detection

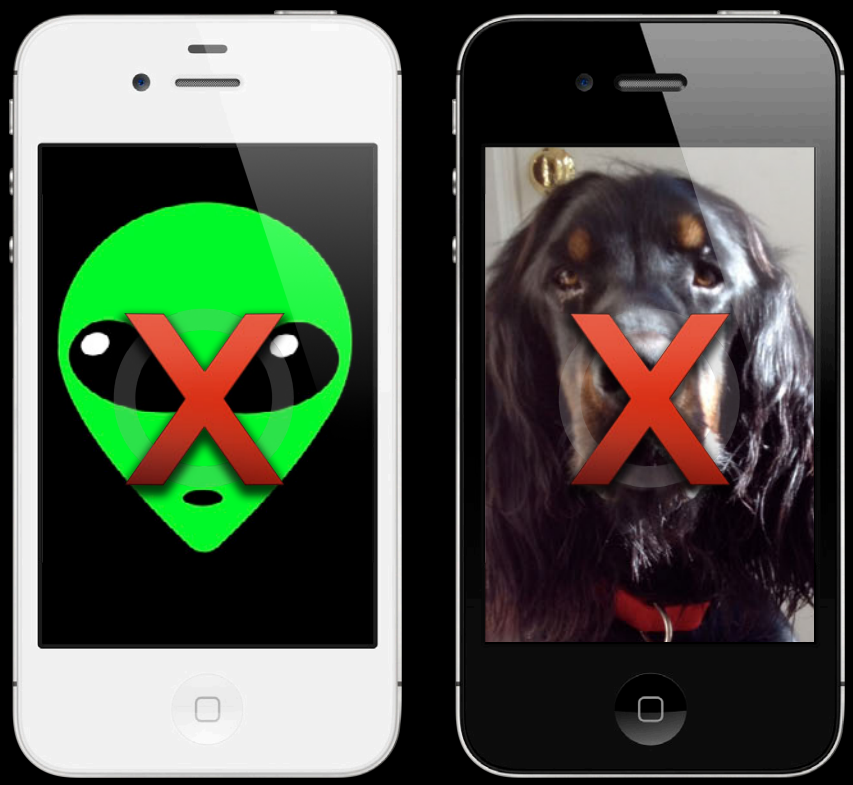
- Scans for faces in real-time
- Tracks up to 10 faces
- Assigns a unique ID to each face
- Provides a timestamp for each face
- Finds the rectangle bounding each face
- Determines the roll angle
- Determines the yaw angle
- Works with front and back camera (all presets!)

Face Detection

- Does NOT find alien or pet faces

Face Detection

- Does NOT find alien or pet faces

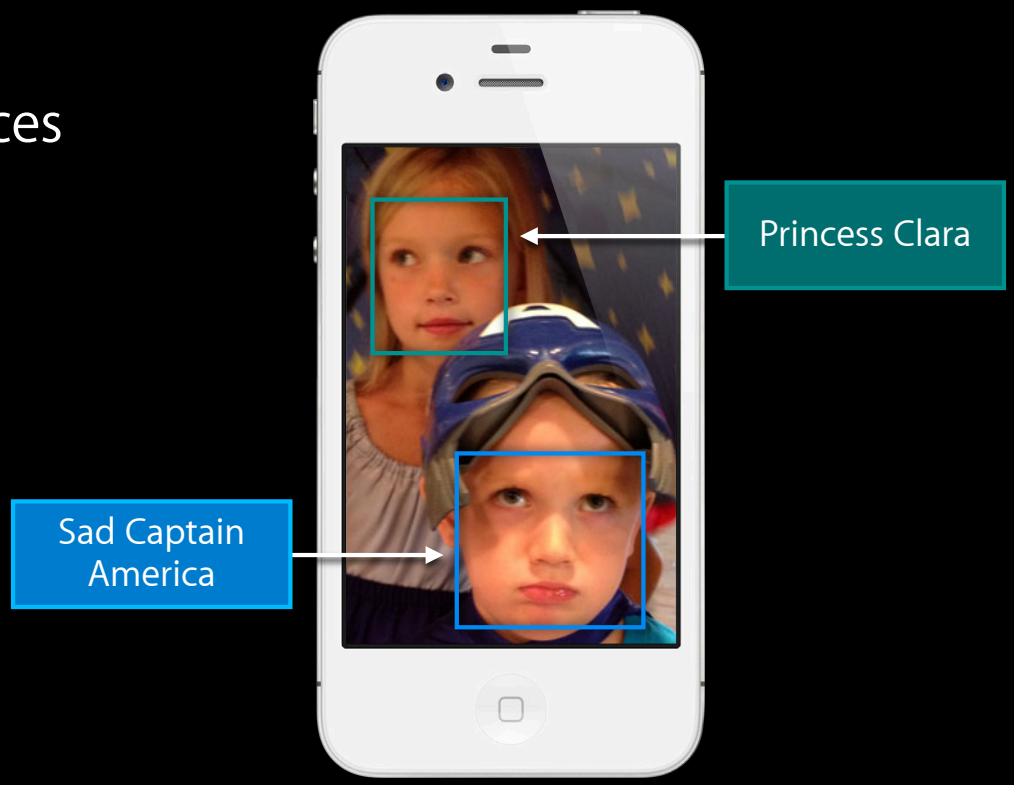


Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces

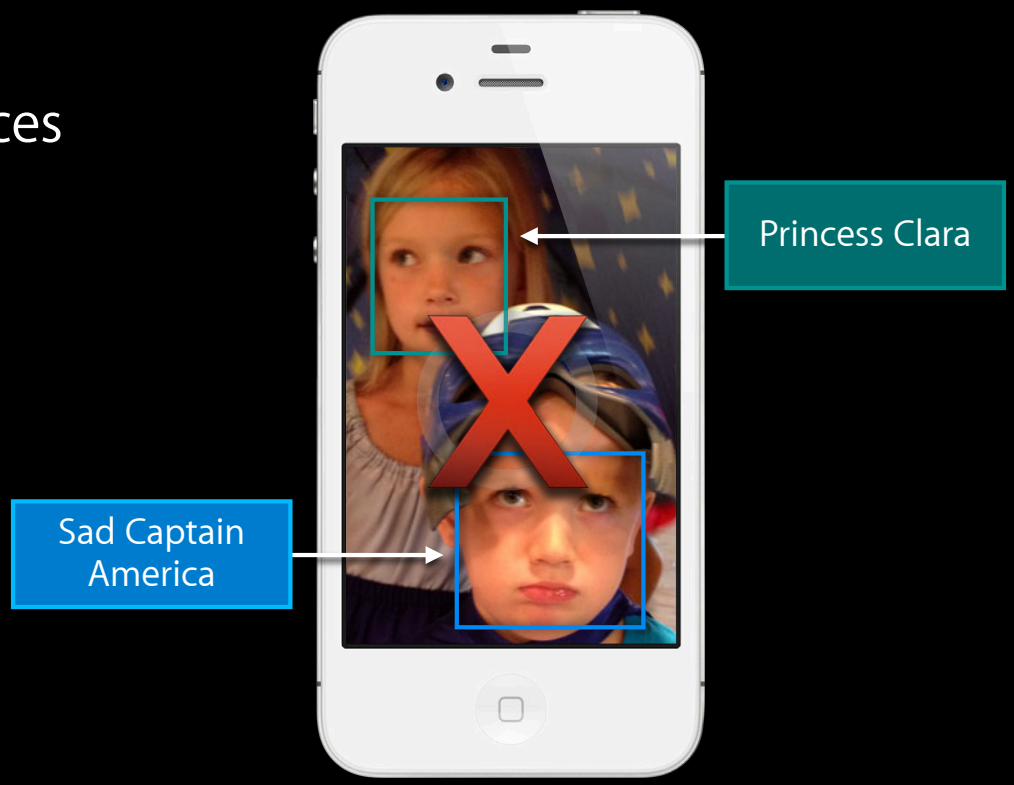
Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces



Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces

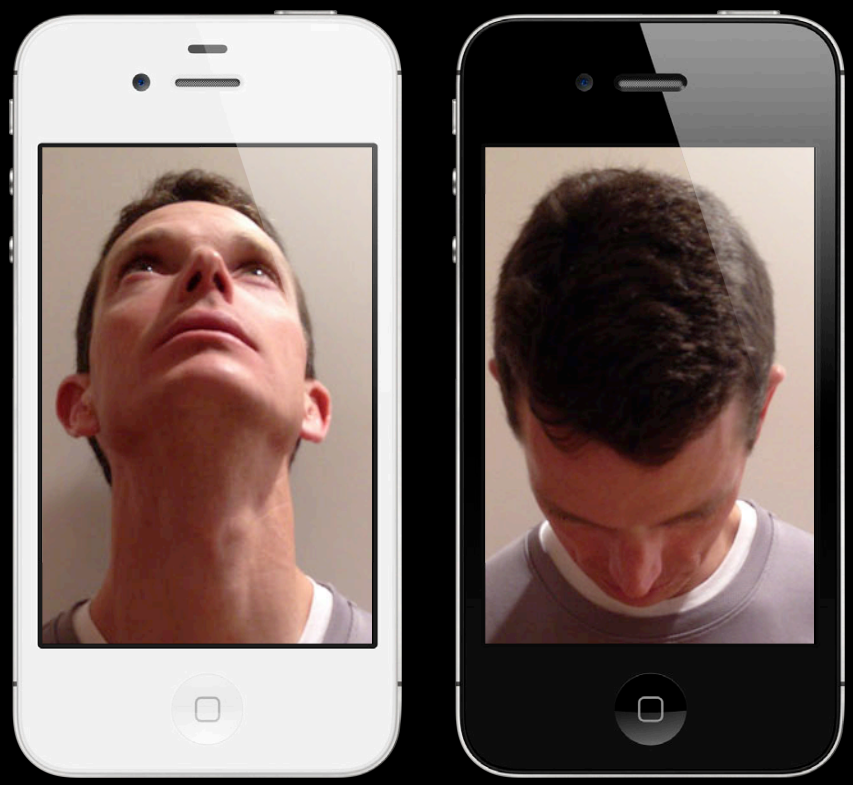


Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces
- Does NOT remember faces

Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces
- Does NOT remember faces
- Does NOT determine pitch



Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces
- Does NOT remember faces
- Does NOT determine pitch

Face Detection

- Does NOT find alien or pet faces
- Does NOT recognize particular faces
- Does NOT remember faces
- Does NOT determine pitch
- Does NOT find faces with a yaw angle between 91 and 269 degrees

Face Detection

Why use AV Foundation Face Detection?

Face Detection

Why use AV Foundation Face Detection?

- Optimized for real-time capture

Face Detection

Why use AV Foundation Face Detection?

- Optimized for real-time capture
- Incurs very little CPU

Face Detection

Why use AV Foundation Face Detection?

- Optimized for real-time capture
- Incurs very little CPU
- Capture resolution independent

Face Detection

Why use AV Foundation Face Detection?

- Optimized for real-time capture
- Incurs very little CPU
- Capture resolution independent
- Supports tracking faces over time

Face Detection

Why use Core Image's CIFaceDetector

Face Detection

Why use Core Image's CIFaceDetector

- Available on all supported iOS devices

Face Detection

Why use Core Image's CIFaceDetector

- Available on all supported iOS devices
- "Push" interface suitable for arbitrary source images

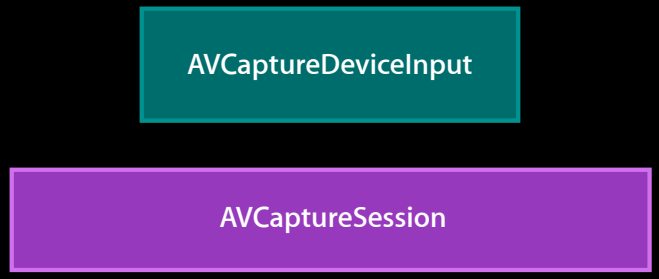
Demo

‘StacheCam 2

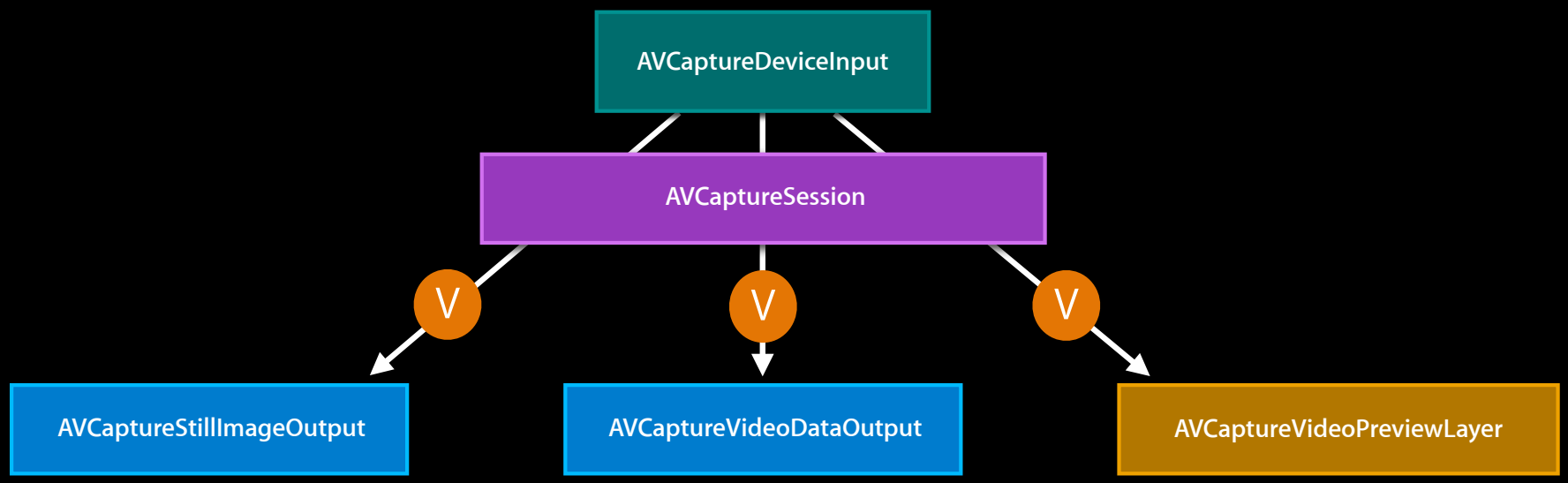
Ethan Tira-Thompson
Core Media Engineering

‘StacheCam 2 (CIFaceDetector path)

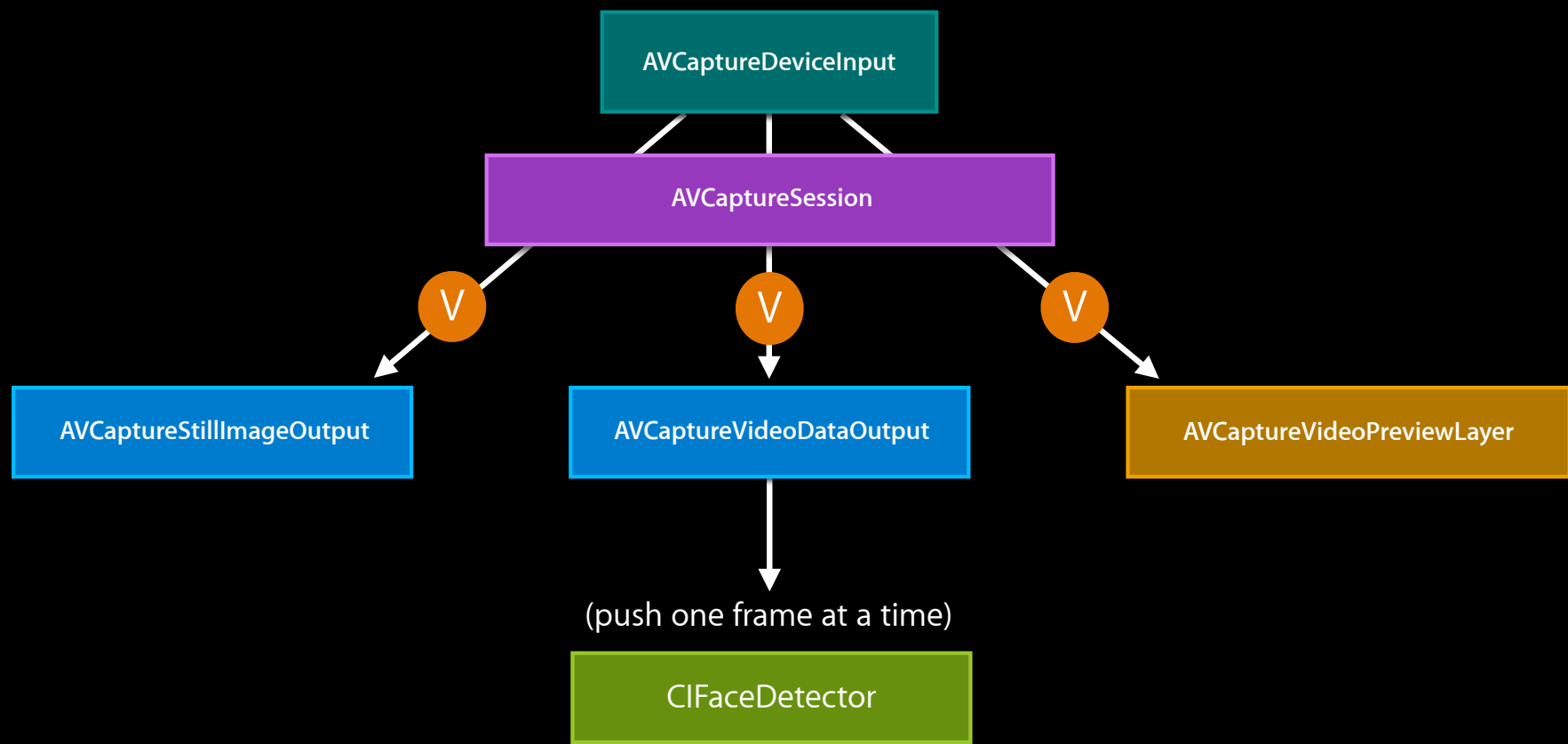
‘StacheCam 2 (CIFaceDetector path)



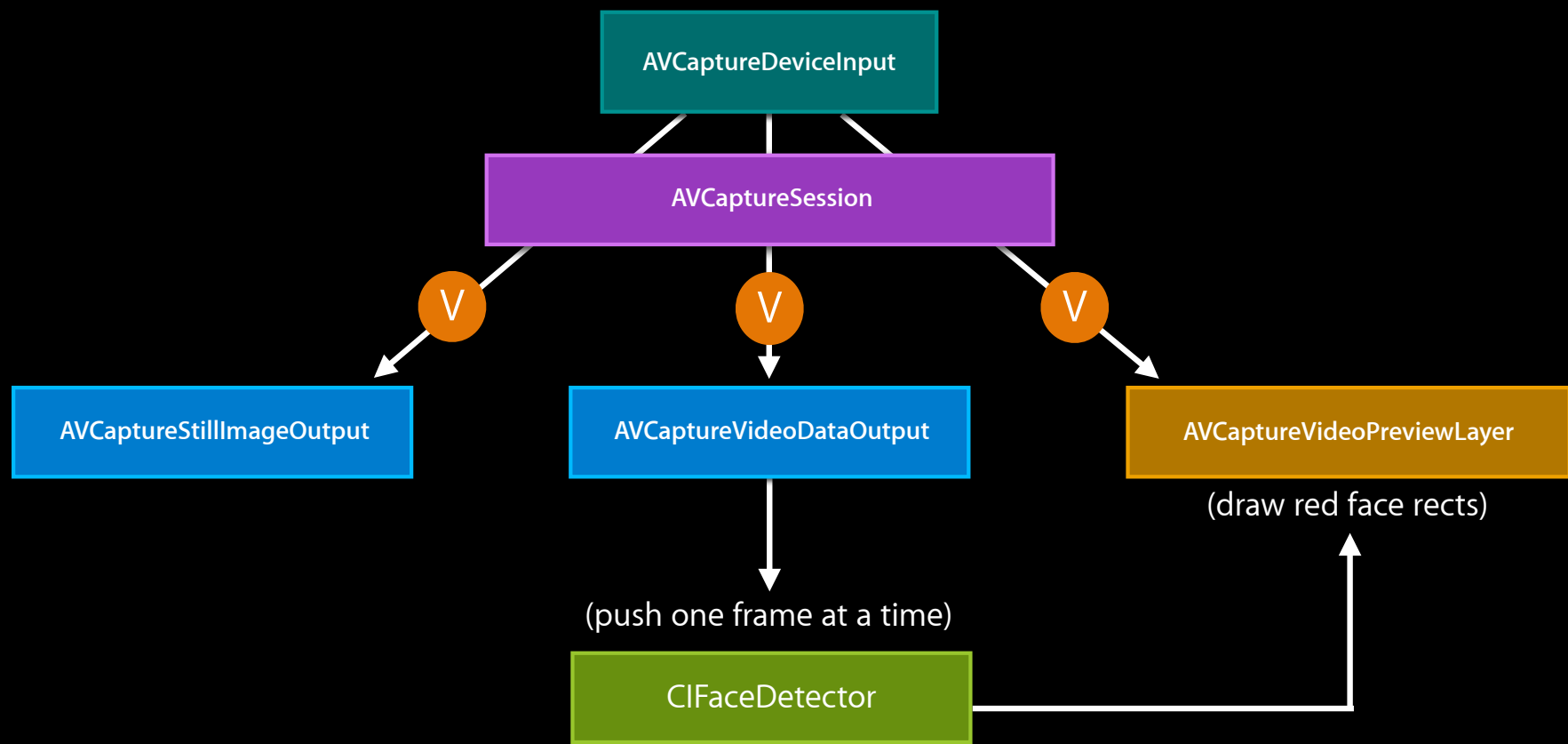
‘StacheCam 2 (CIFaceDetector path)



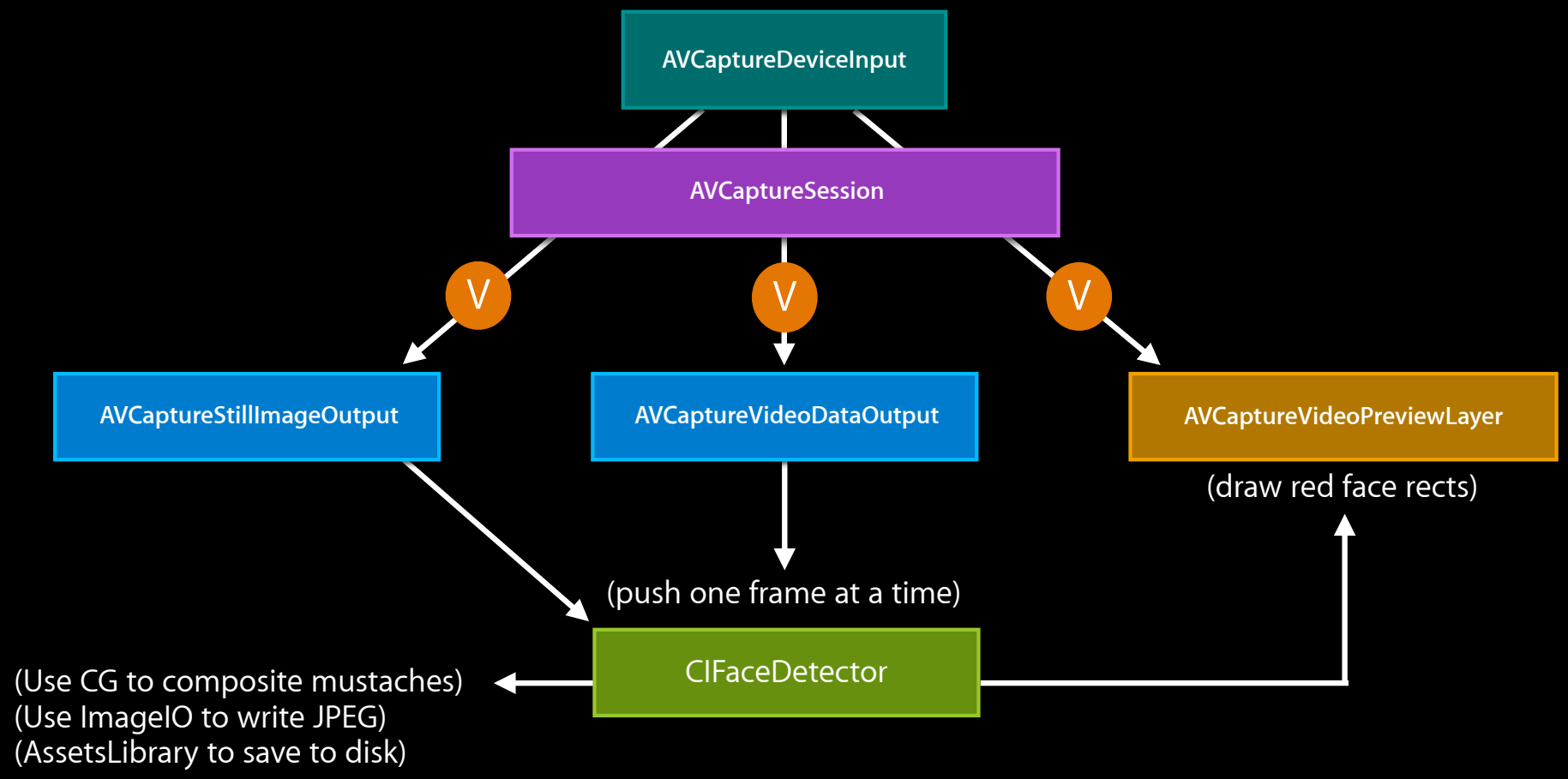
‘StacheCam 2 (CIFaceDetector path)



‘StacheCam 2 (CIFaceDetector path)

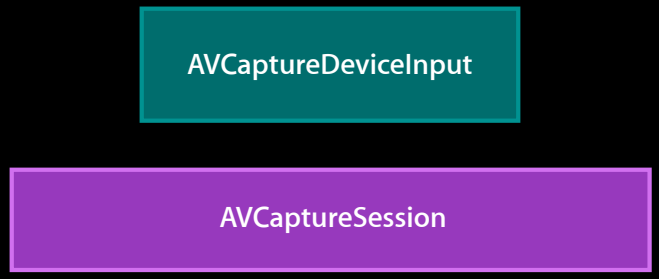


‘StacheCam 2 (CIFaceDetector path)

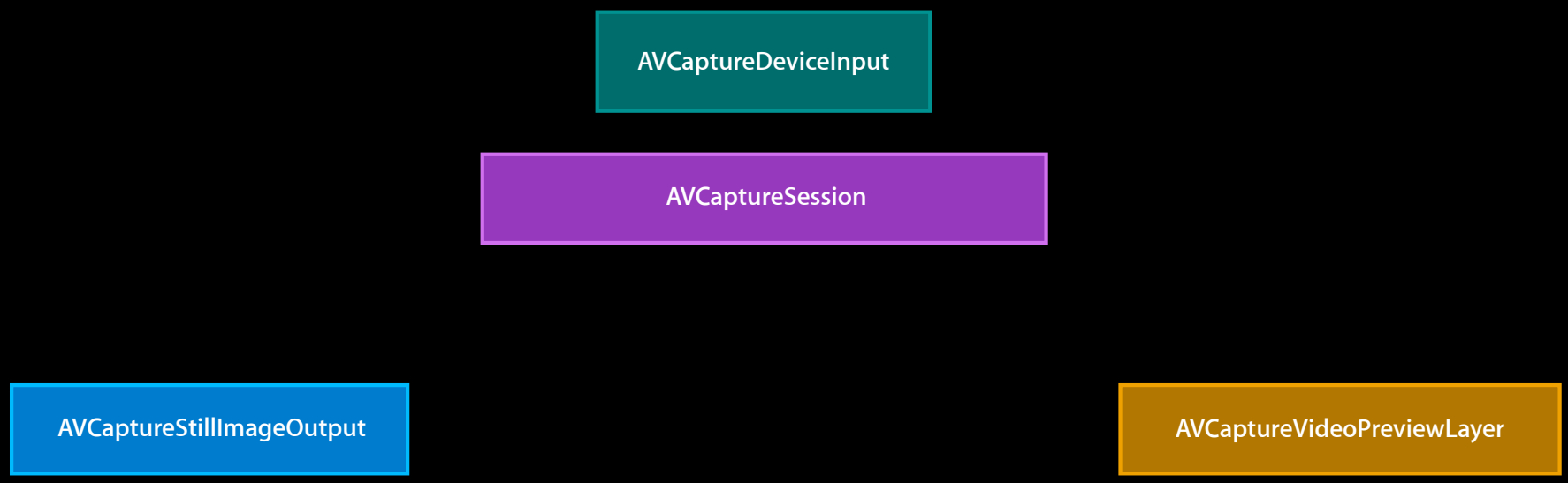


‘StacheCam 2 (real-time path)

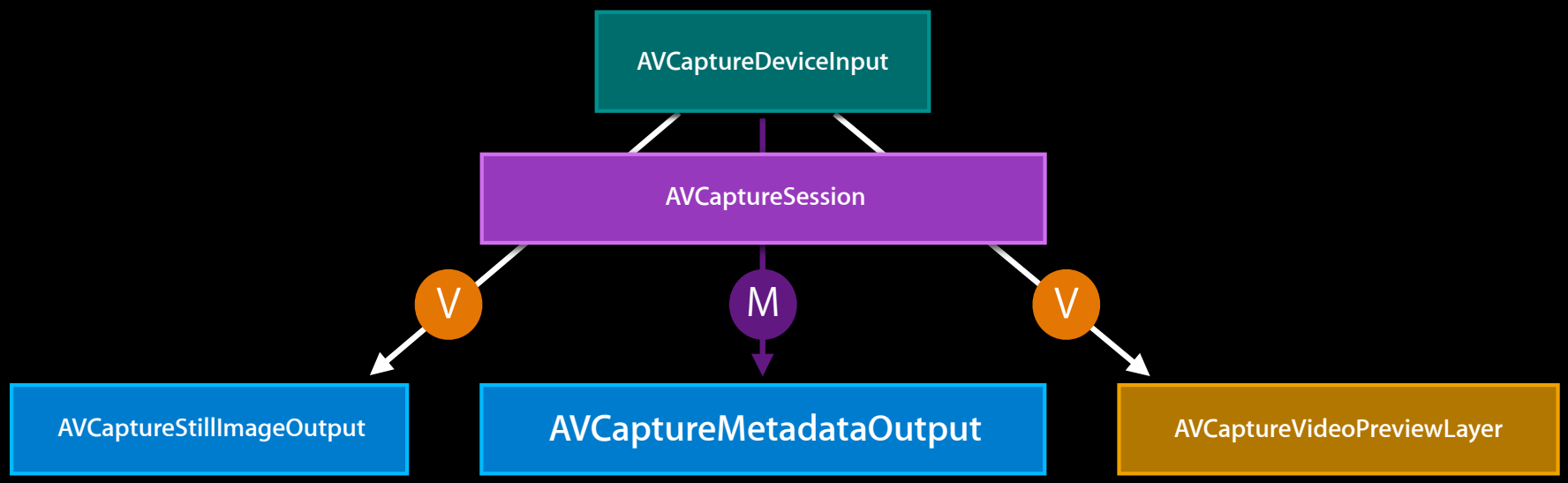
‘StacheCam 2 (real-time path)



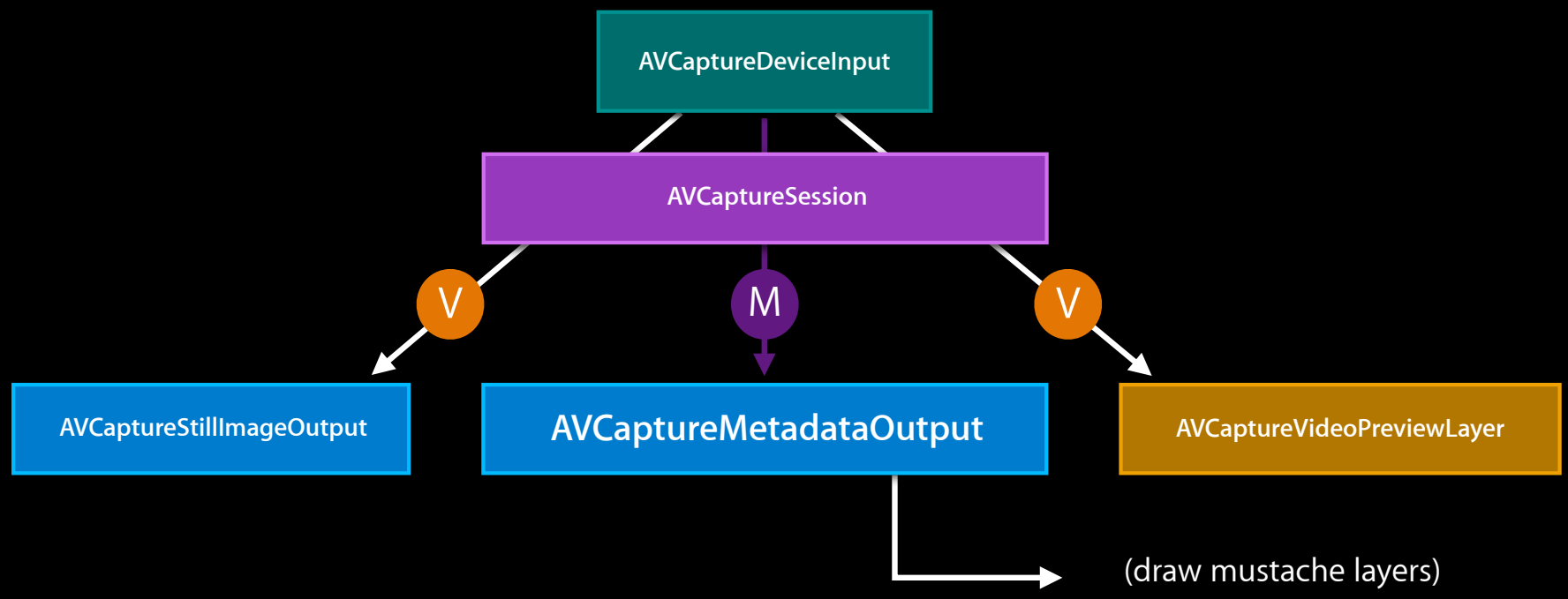
‘StacheCam 2 (real-time path)



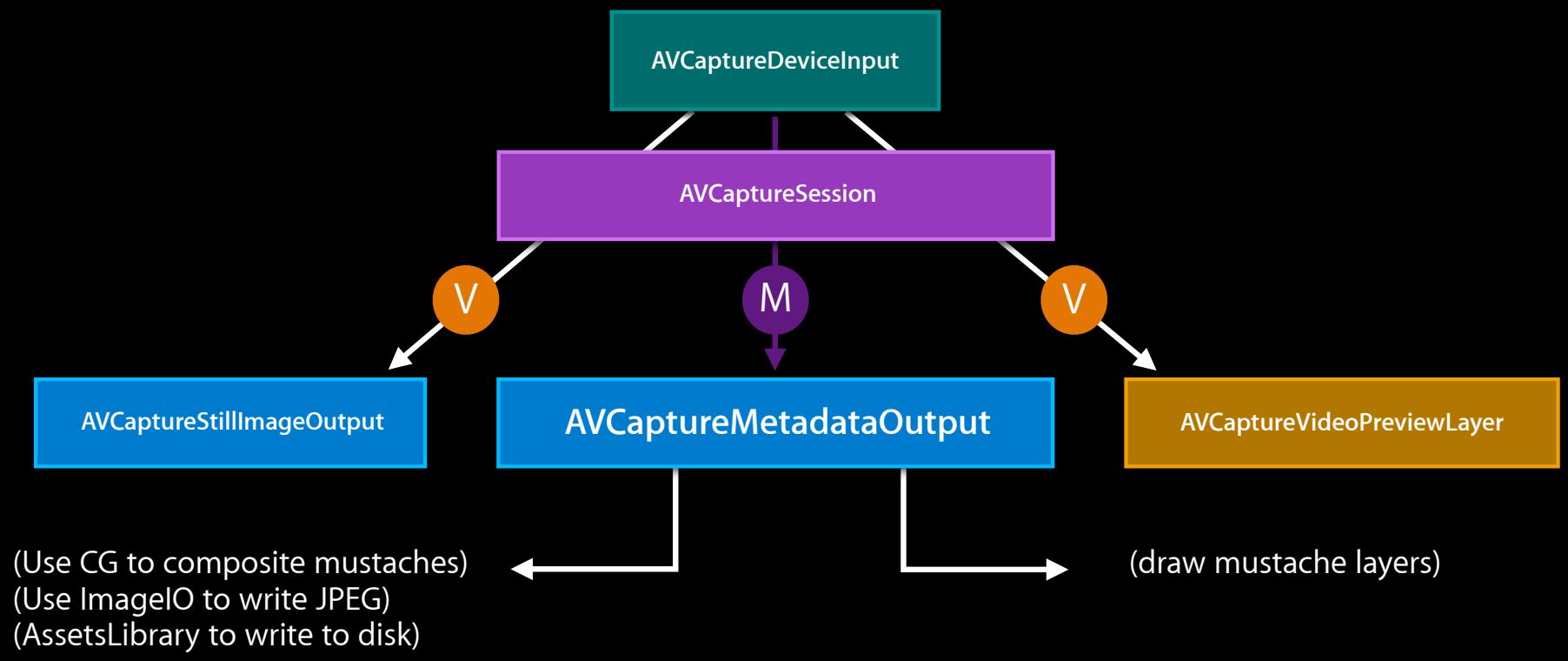
‘StacheCam 2 (real-time path)



‘StacheCam 2 (real-time path)



‘StacheCam 2 (real-time path)



Face Detection

Programming model

Face Detection

Programming model

- AVCaptureDeviceInput exposes an input port of AVMediaTypeMetadata

Face Detection

Programming model

- AVCaptureDeviceInput exposes an input port of AVMediaTypeMetadata
- New AVCaptureOutput subclass AVCaptureMetadataOutput

Face Detection

Programming model

- AVCaptureDeviceInput exposes an input port of AVMediaTypeMetadata
- New AVCaptureOutput subclass AVCaptureMetadataOutput
  - Patterned after AVCaptureVideoDataOutput

Face Detection

Programming model

- AVCaptureDeviceInput exposes an input port of AVMediaTypeMetadata
- New AVCaptureOutput subclass AVCaptureMetadataOutput
  - Patterned after AVCaptureVideoDataOutput
  - Outputs an NSArray of AVMetadataObjects to a delegate

Face Detection

Programming model

- AVCaptureDeviceInput exposes an input port of AVMediaTypeMetadata
- New AVCaptureOutput subclass AVCaptureMetadataOutput
  - Patterned after AVCaptureVideoDataOutput
  - Outputs an NSArray of AVMetadataObjects to a delegate
  - Allows discovery of `-availableMetadataObjectTypes`

Face Detection

Programming model

- AVCaptureDeviceInput exposes an input port of AVMediaTypeMetadata
- New AVCaptureOutput subclass AVCaptureMetadataOutput
  - Patterned after AVCaptureVideoDataOutput
  - Outputs an NSArray of AVMetadataObjects to a delegate
  - Allows discovery of `-availableMetadataObjectTypes`
  - Lets you request a subset of available metadata

```
NSArray *faceMetadata = [NSArray arrayWithObject:AVMetadataObjectTypeFace];
[metadataOutput setMetadataObjectTypes:faceMetadata];
```

Face Detection

What's in a face?

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

     if ( [[object type] isEqual:AVMetadataObjectTypeFace] ) {

      CMTime timestamp = [face time];
      CGRect faceRectangle = [face bounds];
      NSInteger faceID = [face faceID];
      CGFloat rollAngle = [face rollAngle];
      CGFloat yawAngle = [face yawAngle];

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

Face Detection

What's in a face?

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

     if ( [[object type] isEqual:AVMetadataObjectTypeFace] ) {

       CMTime timestamp = [face time];
       CGRect faceRectangle = [face bounds];
       NSInteger faceID = [face faceID];
       CGFloat rollAngle = [face rollAngle];
       CGFloat yawAngle = [face yawAngle];

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

Face Detection

AVFaceMetadataObject

Face Detection

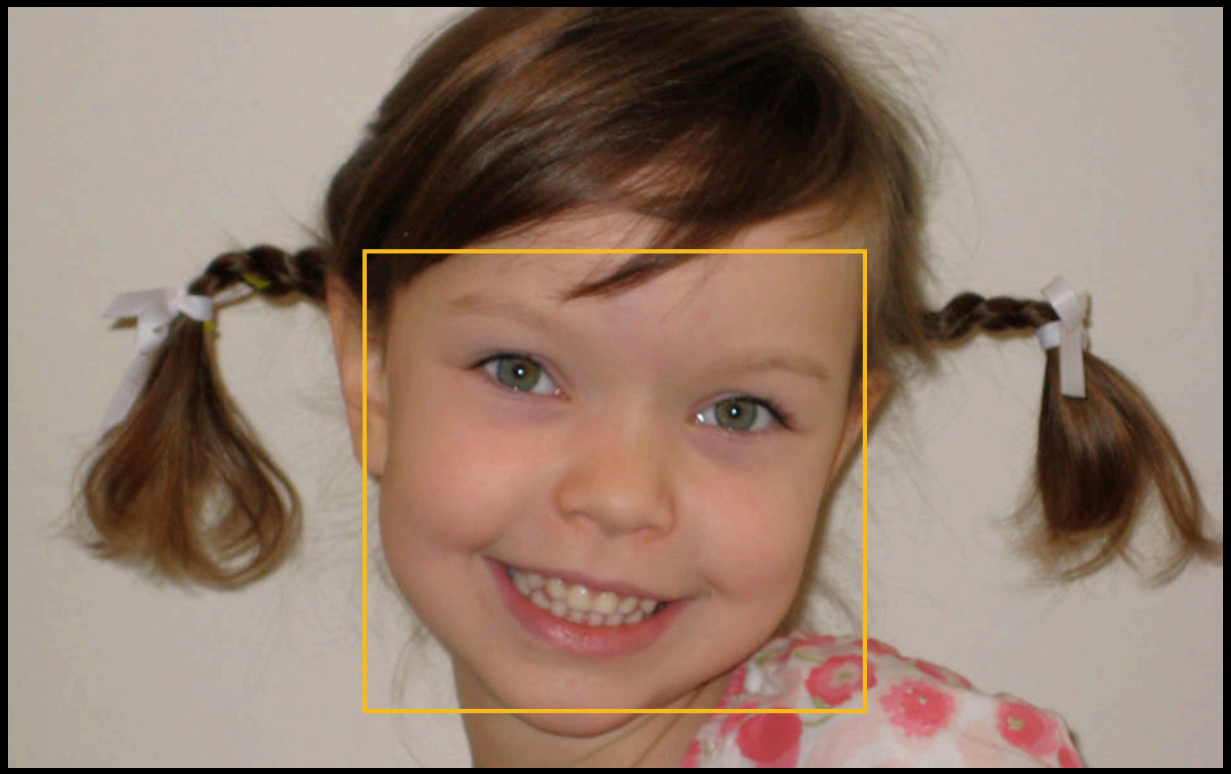
AVFaceMetadataObject

- Face bounds extend from above the eye brows to below the lips

Face Detection

AVFaceMetadataObject

- Face bounds extend from above the eye brows to below the lips



Face Detection

AVFaceMetadataObject

- Face bounds extend from above the eye brows to below the lips
- CGRect coordinates are scalar values from 0 to 1

Face Detection

AVFaceMetadataObject

- Face bounds extend from above the eye brows to below the lips
- CGRect coordinates are scalar values from 0 to 1
- CGRect origin is top-left

Face Detection

AVFaceMetadataObject

- Face bounds extend from above the eye brows to below the lips
- CGRect coordinates are scalar values from 0 to 1
- CGRect origin is top-left
- CGRect coordinates refer to an untransformed source picture

Face Detection

AVFaceMetadataObject

- Face bounds extend from above the eye brows to below the lips
- CGRect coordinates are scalar values from 0 to 1
- CGRect origin is top-left
- CGRect coordinates refer to an untransformed source picture
- CIFaceDetector and AVCaptureMetadataOutput rectangles are comparable in size and origin

Face Detection

Still image support

Face Detection

Still image support

- When using AVCaptureMetadataOutput + AVCaptureStillImageOutput, face rectangles are included with the still image Exif metadata

Face Detection

Still image support

- When using AVCaptureMetadataOutput + AVCaptureStillImageOutput, face rectangles are included with the still image Exif metadata
- Still image output's `-jpegStillImageNSDataRepresentation:` preserves face metadata in XMP Regions

```
[stillImageOutput captureStillImageAsynchronouslyFromConnection:connection
                                                completionHandler:
     ^(CMSampleBufferRef imageSampleBuffer, NSError *error) {
        if ( ! error ) {
           NSData *jpegData = [AVCaptureStillImageOutput
                 jpegStillImageNSDataRepresentation:imageSampleBuffer];

           // Write to disk or AssetsLibrary
     }
```

Face Detection

Still image support

- When using AVCaptureMetadataOutput + AVCaptureStillImageOutput, face rectangles are included with the still image Exif metadata
- Still image output's `-jpegStillImageNSDataRepresentation:` preserves face metadata in XMP Regions

```
[stillImageOutput captureStillImageAsynchronouslyFromConnection:connection
                                                completionHandler:
     ^(CMSampleBufferRef imageSampleBuffer, NSError *error) {
        if ( ! error ) {
           NSData *jpegData = [AVCaptureStillImageOutput
                 jpegStillImageNSDataRepresentation:imageSampleBuffer];

           // Write to disk or AssetsLibrary
     }
```

Face Detection

Supported platforms

iPhone 4S	✓
iPad 2	✓
The new iPad	✓

New in iOS 6

- Video stabilization
- Real-time face detection
- AVCaptureVideoPreviewLayer enhancements

AVCaptureVideoPreviewLayer enhancements

- Conversion methods for focus and exposure points of interest

AVCaptureVideoPreviewLayer enhancements

- Conversion methods for focus and exposure points of interest

ID: 10118665

Title: [API] AVFoundation: Setting focus and exposure points of interest is ridiculously hard

AVCaptureVideoPreviewLayer enhancements

- Conversion methods for focus and exposure points of interest

ID: 10118665

Title: [API] AVFoundation: Setting focus and exposure points of interest is ridiculously hard

"Setting an AVCaptureDevice's focusPointOfInterest and exposurePointOfInterest requires a CGPoint between {0,0} and {1,1}, in a totally arbitrary space, regardless of device orientation. This makes using said API extremely difficult."

AVCaptureVideoPreviewLayer Enhancements

AVCaptureDevice pointOfInterest review

AVCaptureVideoPreviewLayer Enhancements

AVCaptureDevice pointOfInterest review

- `focusPointOfInterest` is a CGPoint from {0, 0} to {1, 1}

AVCaptureVideoPreviewLayer Enhancements

AVCaptureDevice pointOfInterest review

- `focusPointOfInterest` is a CGPoint from {0, 0} to {1, 1}
- Top-left is {0,0}, bottom-right is {1,1}

AVCaptureVideoPreviewLayer Enhancements

AVCaptureDevice pointOfInterest review

- `focusPointOfInterest` is a CGPoint from {0, 0} to {1, 1}
- Top-left is {0,0}, bottom-right is {1,1}
- Camera sensor native (unrotated) orientation is landscape

AVCaptureVideoPreviewLayer Enhancements

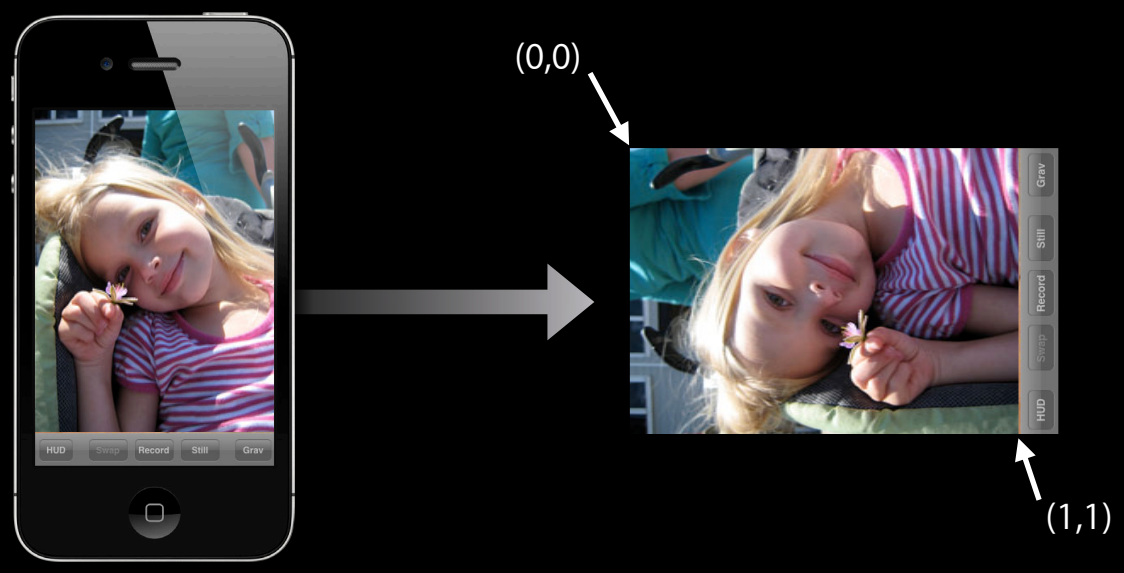
AVCaptureDevice pointOfInterest review

- `focusPointOfInterest` is a CGPoint from {0, 0} to {1, 1}
- Top-left is {0,0}, bottom-right is {1,1}
- Camera sensor native (unrotated) orientation is landscape

AVCaptureVideoPreviewLayer Enhancements

AVCaptureDevice pointOfInterest review

- `focusPointOfInterest` is a CGPoint from {0, 0} to {1, 1}
- Top-left is {0,0}, bottom-right is {1,1}
- Camera sensor native (unrotated) orientation is landscape



AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- Preview may be rotated (videoOrientation)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- Preview may be rotated (videoOrientation)
- Preview may be mirrored (videoMirrored)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- Preview may be rotated (videoOrientation)
- Preview may be mirrored (videoMirrored)
- Preview bounds rect may not have the same aspect ratio as the sensor video buffers (bounds)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- Preview may be rotated (videoOrientation)
- Preview may be mirrored (videoMirrored)
- Preview bounds rect may not have the same aspect ratio as the sensor video buffers (bounds)
- Preview may stretch, shrink, crop, or letterbox the source content (videoGravity)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- AVLayerVideoGravityResizeAspect
- “Letterbox mode"

AVCaptureVideoPreviewLayer Enhancements

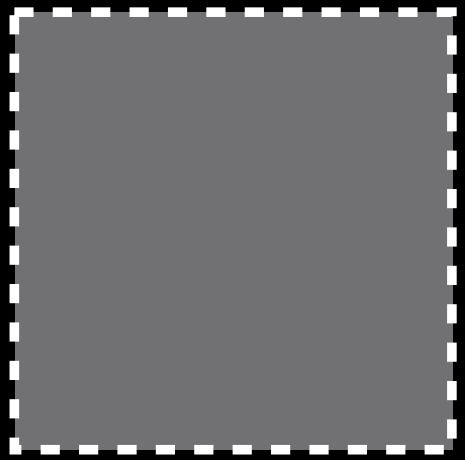
What makes coordinate conversion so hard?

- AVLayerVideoGravityResizeAspect
- “Letterbox mode"

1280 x 720 Source Image

+

640 x 640 Settings



AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- AVLayerVideoGravityResizeAspect
- “Letterbox mode"

1280 x 720 Source Image

+

640 x 640 Settings

=

640 x 640 (with black bars)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- AVLayerVideoGravityResizeAspectFill
- “Crop mode"

1280 x 720 Source Image

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

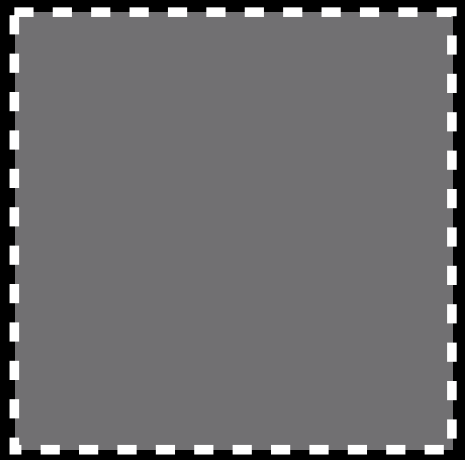
- AVLayerVideoGravityResizeAspectFill
- “Crop mode"

1280 x 720 Source Image

+

640 x 640 Settings

=



640 x 640 (cropped)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- AVLayerVideoGravityResizeAspectFill
- “Crop mode"

1280 x 720 Source Image

+

640 x 640 Settings

=

640 x 640 (cropped)

AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- AVLayerVideoGravityResize
- “Funhouse mode"

1280 x 720 Source Image

AVCaptureVideoPreviewLayer Enhancements

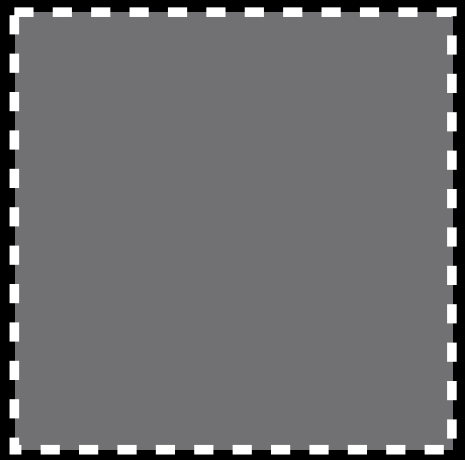
What makes coordinate conversion so hard?

- AVLayerVideoGravityResize
- “Funhouse mode"

1280 x 720 Source Image

+

640 x 640 Settings



AVCaptureVideoPreviewLayer Enhancements

What makes coordinate conversion so hard?

- AVLayerVideoGravityResize
- “Funhouse mode"

1280 x 720 Source Image

+

640 x 640 Settings

=

640 x 640

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

- Convert from a touch point to an AVCaptureDevice point of interest

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

- Convert from a touch point to an AVCaptureDevice point of interest

```
// Set point of interest
CGPoint tapPoint = [gestureRecognizer locationInView:previewView];
CGPoint convertedPoint =
  [videoPreviewLayer captureDevicePointOfInterestForPoint:tapPoint]
[captureDevice setFocusPointOfInterest:convertedPoint];
```

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

- Convert from a touch point to an AVCaptureDevice point of interest

```
// Set point of interest
CGPoint tapPoint = [gestureRecognizer locationInView:previewView];
CGPoint convertedPoint =
  [videoPreviewLayer captureDevicePointOfInterestForPoint:tapPoint]
[captureDevice setFocusPointOfInterest:convertedPoint];
```

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

- Convert from an AVCaptureDevice point of interest to a touch point

```
// Get the current point of interest to draw on preview layer
CGPoint poi = [device focusPointOfInterest];
CGPoint layerPoint =
  [videoPreviewLayer pointForCaptureDevicePointOfInterest:poi];

// Draw something at layerPoint
```

AVCaptureVideoPreviewLayer Enhancements

Conversion methods to the rescue

- Convert from an AVCaptureDevice point of interest to a touch point

```
// Get the current point of interest to draw on preview layer
CGPoint poi = [device focusPointOfInterest];
CGPoint layerPoint =
  [videoPreviewLayer pointForCaptureDevicePointOfInterest:poi];

// Draw something at layerPoint
```

AVMetadataObject Conversion For Preview

AVMetadataObject Conversion For Preview

- Convert face metadata for video preview layer drawing

```
for ( AVMetadataFaceObject *face in metadataObjects ) {
  AVMetadataFaceObject *transformedFace =
    [previewLayer transformedMetadataObjectForMetadataObject:face];
  CGRect transformedFaceRect = [transformedFace bounds];

  // Draw a funny mustache on the face
}
```

AVMetadataObject Conversion For Preview

- Convert face metadata for video preview layer drawing

```
for ( AVMetadataFaceObject *face in metadataObjects ) {
  AVMetadataFaceObject *transformedFace =
    [previewLayer transformedMetadataObjectForMetadataObject:face];
  CGRect transformedFaceRect = [transformedFace bounds];

  // Draw a funny mustache on the face
}
```

AVMetadataObject Conversion for Output

AVMetadataObject Conversion for Output

- Convert face metadata for AVCaptureOutput drawing

AVMetadataObject Conversion for Output

- Convert face metadata for AVCaptureOutput drawing
- Align faces with physically rotated video data output

```
for ( AVMetadataFaceObject *face in metadataObjects ) {
  AVCaptureConnection *c = [vdo connectionWithMediaType:AVMediaTypeVideo];
  AVMetadataFaceObject *transformedFace =
    [vdo transformedMetadataObjectForMetadataObject:face connection:c];
  CGRect transformedFaceRect = [transformedFace bounds];

  // Draw a funny mustache on the face
}
```

AVMetadataObject Conversion for Output

- Convert face metadata for AVCaptureOutput drawing
- Align faces with physically rotated video data output

```
for ( AVMetadataFaceObject *face in metadataObjects ) {
  AVCaptureConnection *c = [vdo connectionWithMediaType:AVMediaTypeVideo];
  AVMetadataFaceObject *transformedFace =
    [vdo transformedMetadataObjectForMetadataObject:face connection:c];
  CGRect transformedFaceRect = [transformedFace bounds];

  // Draw a funny mustache on the face
}
```

AVCaptureVideoPreviewLayer Enhancements

Pause and resume video preview

- AVCaptureVideoPreviewLayer exposes an AVCaptureConnection

AVCaptureVideoPreviewLayer Enhancements

Pause and resume video preview

- AVCaptureVideoPreviewLayer exposes an AVCaptureConnection
- All connection properties are available to the layer

AVCaptureVideoPreviewLayer Enhancements

Pause and resume video preview

- AVCaptureVideoPreviewLayer exposes an AVCaptureConnection
- All connection properties are available to the layer
- To pause video preview, disable the connection

AVCaptureVideoPreviewLayer Enhancements

Pause and resume video preview

- AVCaptureVideoPreviewLayer exposes an AVCaptureConnection
- All connection properties are available to the layer
- To pause video preview, disable the connection
- Causes no glitch in any of the outputs

```
AVCaptureConnection *previewConnection = [videoPreviewLayer connection];

// pause preview
[previewConnection setEnabled:NO];
```

AVCaptureVideoPreviewLayer Enhancements

Pause and resume video preview

- AVCaptureVideoPreviewLayer exposes an AVCaptureConnection
- All connection properties are available to the layer
- To pause video preview, disable the connection
- Causes no glitch in any of the outputs

```
AVCaptureConnection *previewConnection = [videoPreviewLayer connection];

// pause preview
[previewConnection setEnabled:NO];
```

AVCaptureVideoPreviewLayer

Deprecations

AVCaptureVideoPreviewLayer

Deprecations

- Preview layer's `-connection` property makes some methods redundant

AVCaptureVideoPreviewLayer

Deprecations

- Preview layer's `-connection` property makes some methods redundant
- See AVCaptureVideoPreviewLayer.h

AVCaptureVideoPreviewLayer

Deprecations

- Preview layer's `-connection` property makes some methods redundant
- See AVCaptureVideoPreviewLayer.h

Deprecated	Instead Use
	`conn = [layer connection];`
`layer.isOrientationSupported`	`conn.isVideoOrientationSupported`
`layer.orientation`	`conn.videoOrientation`
`layer.isMirroringSupported`	`conn.isVideoMirroringSupported`
`layer.automaticallyAdjustsMirroring`	`conn.automaticallyAdjustsVideoMirroring`
`layer.isMirrored`	`conn.isVideoMirrored`

Miscellaneous API Enhancements

- AVCaptureDevice's `-torchActive` property
- AVCaptureDevice's `-setTorchModeOnWithLevel:error:` method
- AVCaptureStillImageOutput's support for AVVideoQualityKey

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6
- Solutions for performance problems in your capture app
- Synchronizing motion data with video

Solutions for Performance Problems

Solving Performance Problems

Common performance problems

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture
  - Is it my fault?

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture
  - Is it my fault?
  - What can I do to recover?

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture
  - Is it my fault?
  - What can I do to recover?
- My AVAssetWriter recorded movies have frame drops at the beginning

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture
  - Is it my fault?
  - What can I do to recover?
- My AVAssetWriter recorded movies have frame drops at the beginning
- My AVAssetWriter recorded movies have garbage (I use OpenGL)

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture
  - Is it my fault?
  - What can I do to recover?
- My AVAssetWriter recorded movies have frame drops at the beginning
- My AVAssetWriter recorded movies have garbage (I use OpenGL)
- My DIY preview is slow

Solving Performance Problems

Common performance problems

- My app is dropping frames during video capture
  - Is it my fault?
  - What can I do to recover?
- My AVAssetWriter recorded movies have frame drops at the beginning
- My AVAssetWriter recorded movies have garbage (I use OpenGL)
- My DIY preview is slow
  - How do I speed it up?

Solving Performance Problems

Handling frame drops

Solving Performance Problems

Handling frame drops

- Set AVCaptureVideoDataOutput's `-alwaysDiscardsLateVideoFrames` to YES

Solving Performance Problems

Handling frame drops

- Set AVCaptureVideoDataOutput's `-alwaysDiscardsLateVideoFrames` to YES
  - Unless you are recording

Solving Performance Problems

Handling frame drops

- Set AVCaptureVideoDataOutput's `-alwaysDiscardsLateVideoFrames` to YES
  - Unless you are recording
  - Enforces a buffer queue size of 1 at the end of video data output's processing pipeline

Solving Performance Problems

Handling frame drops

- Set AVCaptureVideoDataOutput's `-alwaysDiscardsLateVideoFrames` to YES
  - Unless you are recording
  - Enforces a buffer queue size of 1 at the end of video data output's processing pipeline
  - Saves you from periodically slow processing

Solving Performance Problems

Handling frame drops

- Set AVCaptureVideoDataOutput's `-alwaysDiscardsLateVideoFrames` to YES
  - Unless you are recording
  - Enforces a buffer queue size of 1 at the end of video data output's processing pipeline
  - Saves you from periodically slow processing
  - Does not save you from chronically slow processing

Solving Performance Problems

Handling frame drops

Solving Performance Problems

Handling frame drops

- New in iOS 6, AVCaptureVideoDataOutput can report frame drops

```
// New optional AVCaptureVideoDataOutputDelegate method
- (void)captureOutput:(AVCaptureOutput *)captureOutput
    didDropSampleBuffer:(CMSampleBufferRef)sampleBuffer
         fromConnection:(AVCaptureConnection *)connection
{
    // We just dropped a frame!

}
```

Solving Performance Problems

Handling frame drops

- New in iOS 6, AVCaptureVideoDataOutput can report frame drops

```
// New optional AVCaptureVideoDataOutputDelegate method
- (void)captureOutput:(AVCaptureOutput *)captureOutput
    didDropSampleBuffer:(CMSampleBufferRef)sampleBuffer
         fromConnection:(AVCaptureConnection *)connection
{
    // We just dropped a frame!

}
```

Solving Performance Problems

Handling frame drops

Solving Performance Problems

Handling frame drops

- The didDropSampleBuffer contains no image data

Solving Performance Problems

Handling frame drops

- The didDropSampleBuffer contains no image data
- Does contain timing information and format description

Solving Performance Problems

Handling frame drops

- The didDropSampleBuffer contains no image data
- Does contain timing information and format description
- Does contain `kCMSampleBufferAttachmentKey_DroppedFrameReason`

Solving Performance Problems

Handling frame drops

- The didDropSampleBuffer contains no image data
- Does contain timing information and format description
- Does contain `kCMSampleBufferAttachmentKey_DroppedFrameReason`
  - `kCMSampleBufferDroppedFrameReason_FrameWasLate`

Solving Performance Problems

Handling frame drops

- The didDropSampleBuffer contains no image data
- Does contain timing information and format description
- Does contain `kCMSampleBufferAttachmentKey_DroppedFrameReason`
  - `kCMSampleBufferDroppedFrameReason_FrameWasLate`
  - `kCMSampleBufferDroppedFrameReason_OutOfBuffers`

Solving Performance Problems

Handling frame drops

- The didDropSampleBuffer contains no image data
- Does contain timing information and format description
- Does contain `kCMSampleBufferAttachmentKey_DroppedFrameReason`
  - `kCMSampleBufferDroppedFrameReason_FrameWasLate`
  - `kCMSampleBufferDroppedFrameReason_OutOfBuffers`
  - `kCMSampleBufferDroppedFrameReason_Discontinuity`

Solving Performance Problems

Handling frame drops

Solving Performance Problems

Handling frame drops

- Frame drops can be mitigated by lowering the frame rate

Solving Performance Problems

Handling frame drops

- Frame drops can be mitigated by lowering the frame rate
- As of iOS 5, the video data output frame rate can be altered dynamically

Solving Performance Problems

Handling frame drops

- Frame drops can be mitigated by lowering the frame rate
- As of iOS 5, the video data output frame rate can be altered dynamically
- No glitch in preview or output

```
// Lower the min and max frame rate to recover from slow processing
AVCaptureConnection *c = [dataOutput connectionWithMediaType:AVMediaTypeVideo];

// min duration is 1 / max frame rate
int32_t newFrameRate = currentRate - 1;
[c setVideoMinFrameDuration:CMTimeMake( 1, newFrameRate )];
[c setVideoMaxFrameDuration:CMTimeMake( 1, newFrameRate )];
```

Solving Performance Problems

Handling frame drops

- Frame drops can be mitigated by lowering the frame rate
- As of iOS 5, the video data output frame rate can be altered dynamically
- No glitch in preview or output

```
// Lower the min and max frame rate to recover from slow processing
AVCaptureConnection *c = [dataOutput connectionWithMediaType:AVMediaTypeVideo];

// min duration is 1 / max frame rate
int32_t newFrameRate = currentRate - 1;
[c setVideoMinFrameDuration:CMTimeMake( 1, newFrameRate )];
[c setVideoMaxFrameDuration:CMTimeMake( 1, newFrameRate )];
```

Solving Performance Problems

Solving AVAssetWriter frame drops

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing
  - Preallocates buffers for glitch free movie writing

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing
  - Preallocates buffers for glitch free movie writing
- AVAssetWriter

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing
  - Preallocates buffers for glitch free movie writing
- AVAssetWriter
  - Does not know the source format

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing
  - Preallocates buffers for glitch free movie writing
- AVAssetWriter
  - Does not know the source format
  - Cannot prime the render pipeline

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing
  - Preallocates buffers for glitch free movie writing
- AVAssetWriter
  - Does not know the source format
  - Cannot prime the render pipeline
  - Sets things up on the first `-appendSampleBuffer:`

Solving Performance Problems

Solving AVAssetWriter frame drops

- AVCaptureMovieFileOutput
  - Optimized for real-time file writing
  - Preallocates buffers for glitch free movie writing
- AVAssetWriter
  - Does not know the source format
  - Cannot prime the render pipeline
  - Sets things up on the first `-appendSampleBuffer:`
  - Result: dropped frames at the very beginning

Solving Performance Problems

Solving AVAssetWriter frame drops

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

```
+ (AVAssetWriterInput *)assetWriterInputWithMediaType:(NSString *)mediaType
```

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

```
+ (AVAssetWriterInput *)assetWriterInputWithMediaType:(NSString *)mediaType
     outputSettings:(NSDictionary *)outputSettings
```

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

```
+ (AVAssetWriterInput *)assetWriterInputWithMediaType:(NSString *)mediaType
     outputSettings:(NSDictionary *)outputSettings
     sourceFormatHint:(CMFormatDescriptionRef)sourceFormatHint;
```

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

```
+ (AVAssetWriterInput *)assetWriterInputWithMediaType:(NSString *)mediaType
      outputSettings:(NSDictionary *)outputSettings
      sourceFormatHint:(CMFormatDescriptionRef)sourceFormatHint;
```

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

```
+ (AVAssetWriterInput *)assetWriterInputWithMediaType:(NSString *)mediaType
      outputSettings:(NSDictionary *)outputSettings
      sourceFormatHint:(CMFormatDescriptionRef)sourceFormatHint;
```

- Start up costs move to [AVAssetWriter startWriting]

Solving Performance Problems

Solving AVAssetWriter frame drops

- Set AVAssetWriterInput's `-expectsMediaDataInRealTime` flag to YES
- New in iOS 6, AVAssetWriterInput allows you to hint the source format up front

```
+ (AVAssetWriterInput *)assetWriterInputWithMediaType:(NSString *)mediaType
      outputSettings:(NSDictionary *)outputSettings
      sourceFormatHint:(CMFormatDescriptionRef)sourceFormatHint;
```

- Start up costs move to [AVAssetWriter startWriting]
- Set up your AVAssetWriter outside of `-captureOutput:didOutputSampleBuffer:fromConnection:`

Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

- When rendering to a texture using CVOpenGLESTextureCache, ensure GL has finished rendering before passing to AVAssetWriter

Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

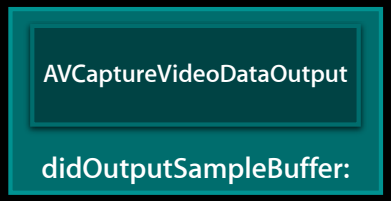
- When rendering to a texture using CVOpenGLESTextureCache, ensure GL has finished rendering before passing to AVAssetWriter
- `glFinish()` is safe but may block

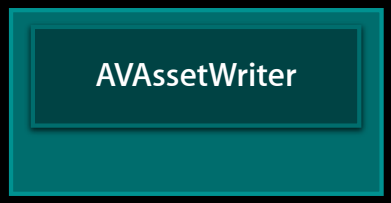
Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

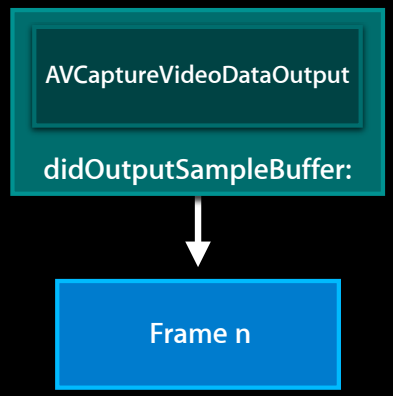
- When rendering to a texture using CVOpenGLESTextureCache, ensure GL has finished rendering before passing to AVAssetWriter
- `glFinish()` is safe but may block
- `glFlush()` + delayed `glFinish()` keeps both GPU and CPU busy

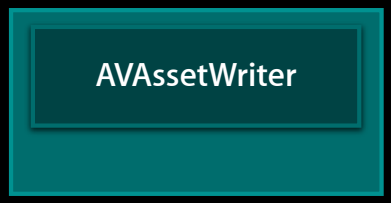
Solving Performance Problems



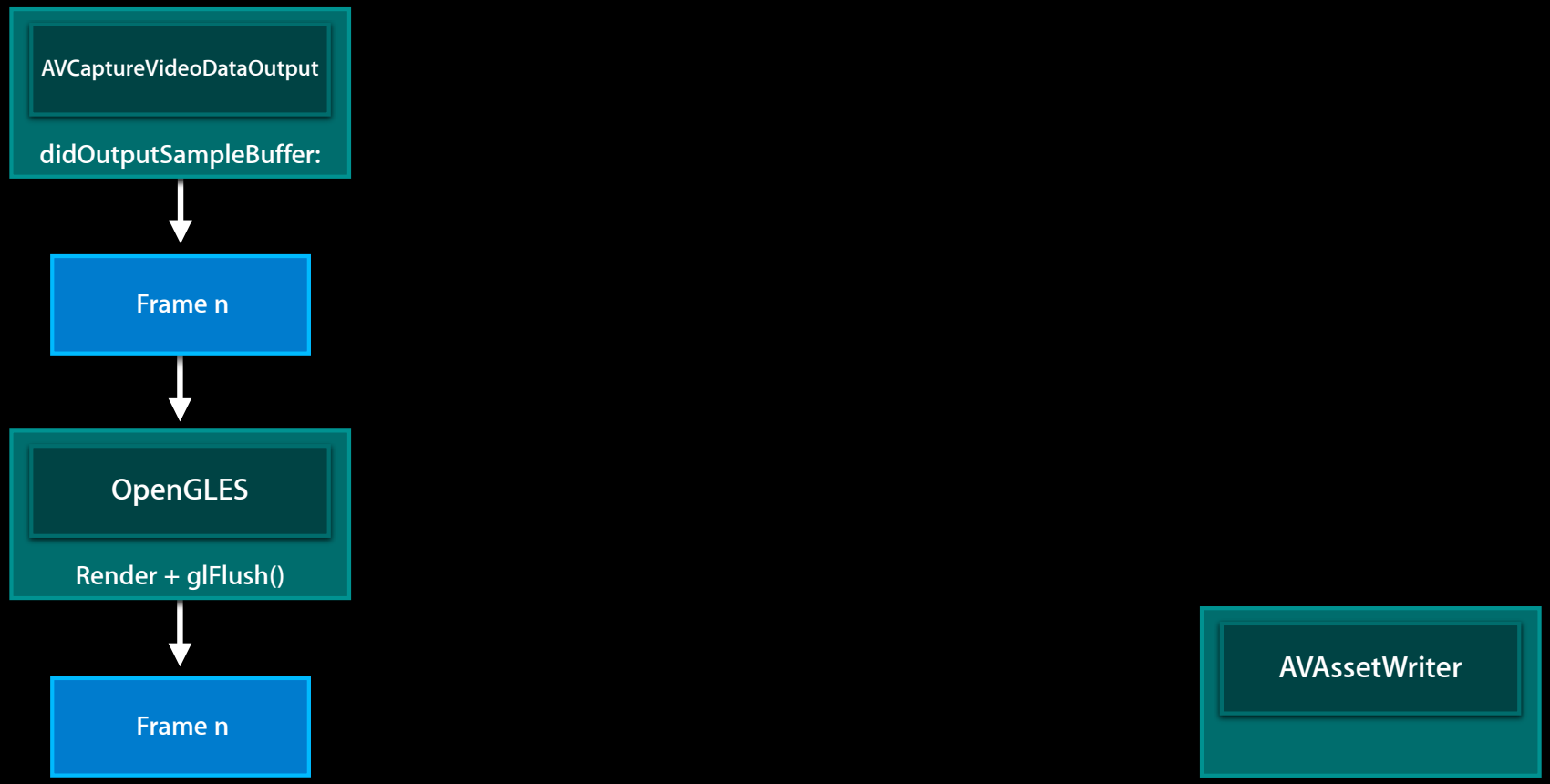


Solving Performance Problems

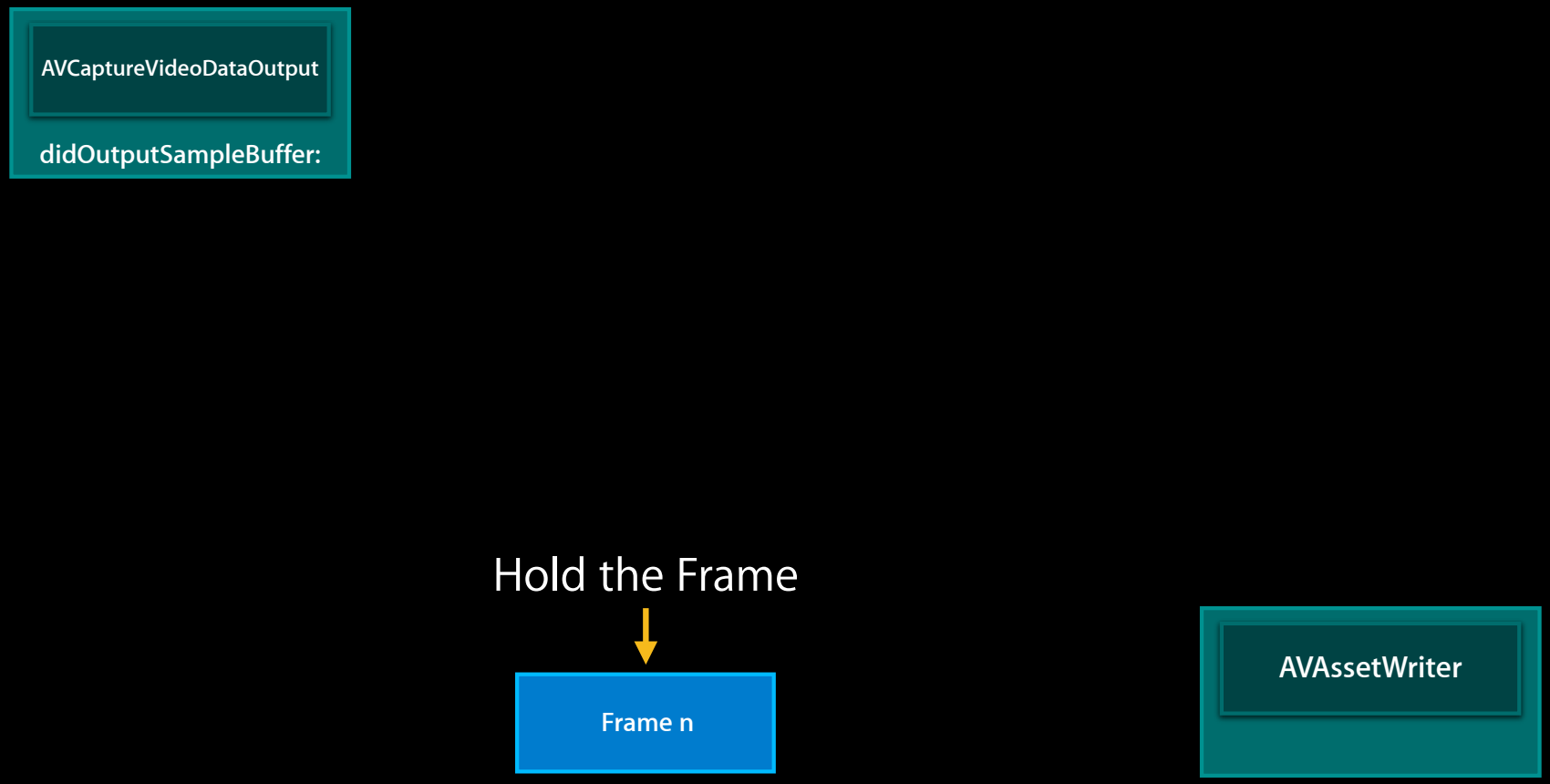




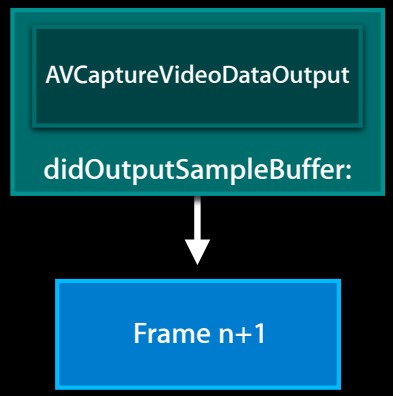
Solving Performance Problems

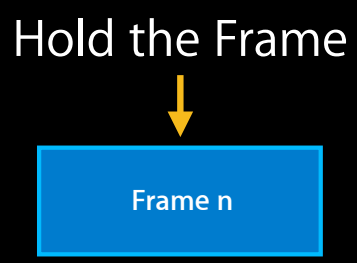


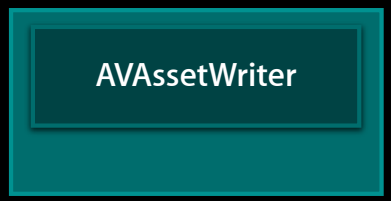
Solving Performance Problems



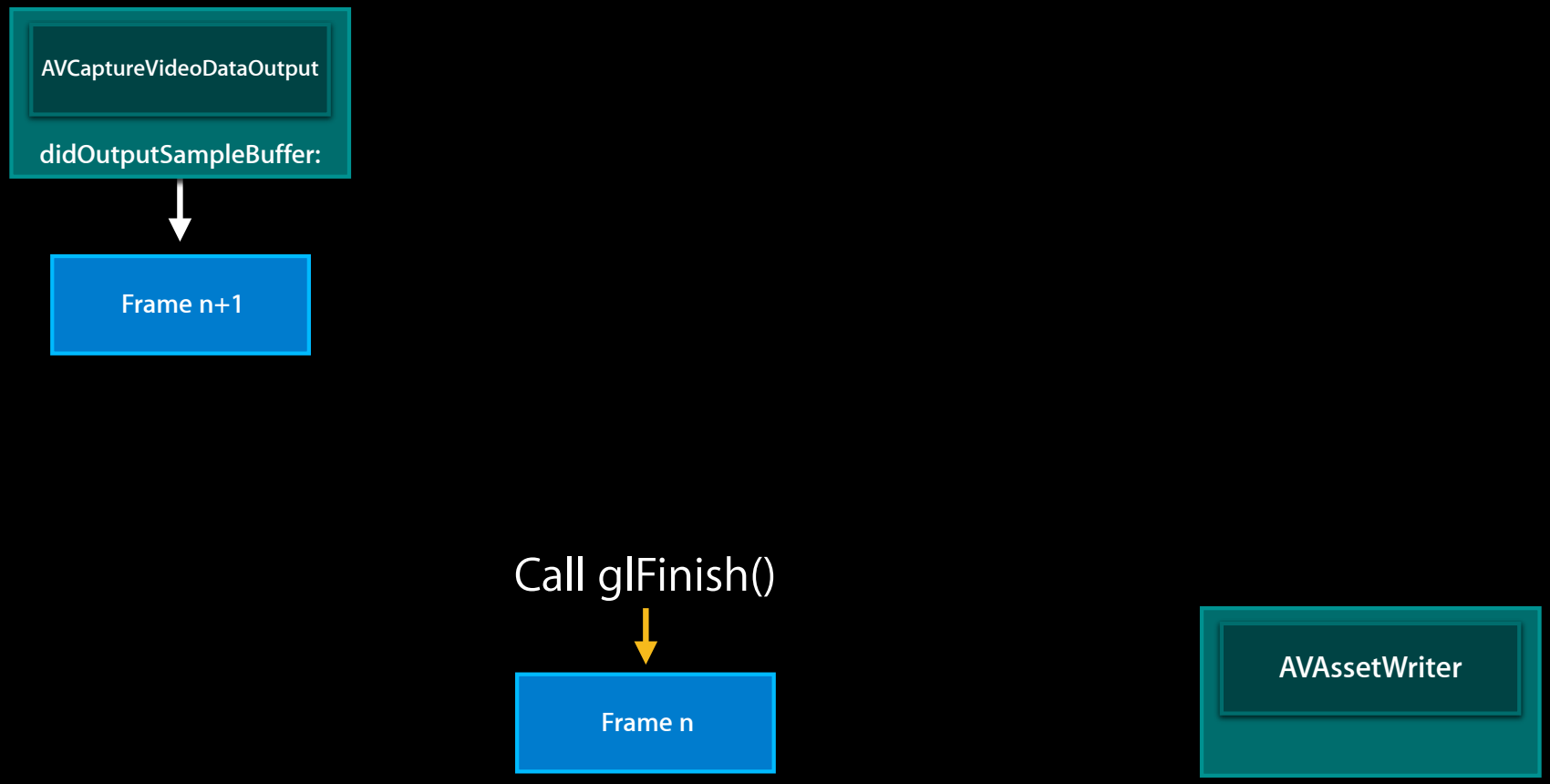
Solving Performance Problems



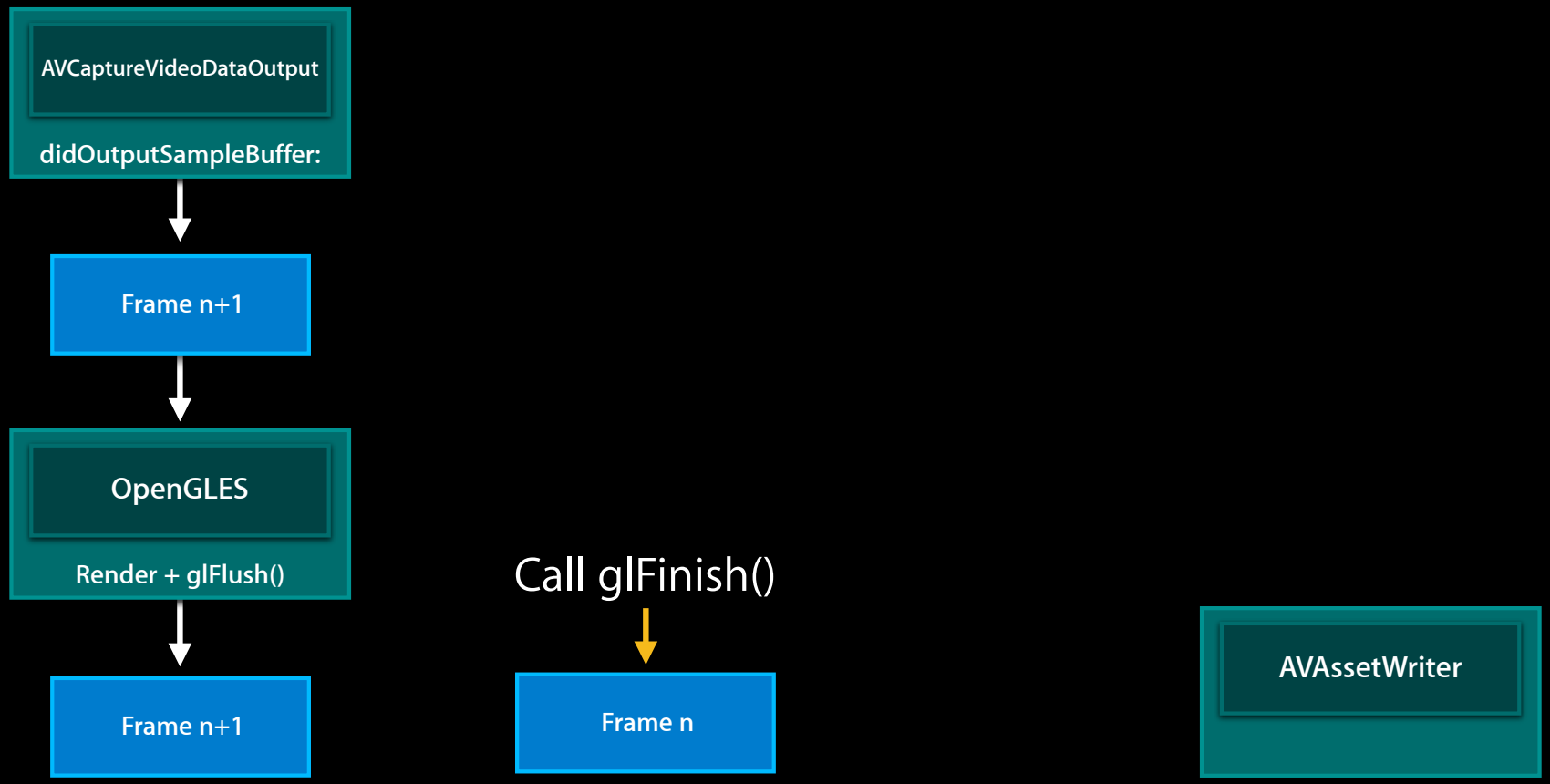




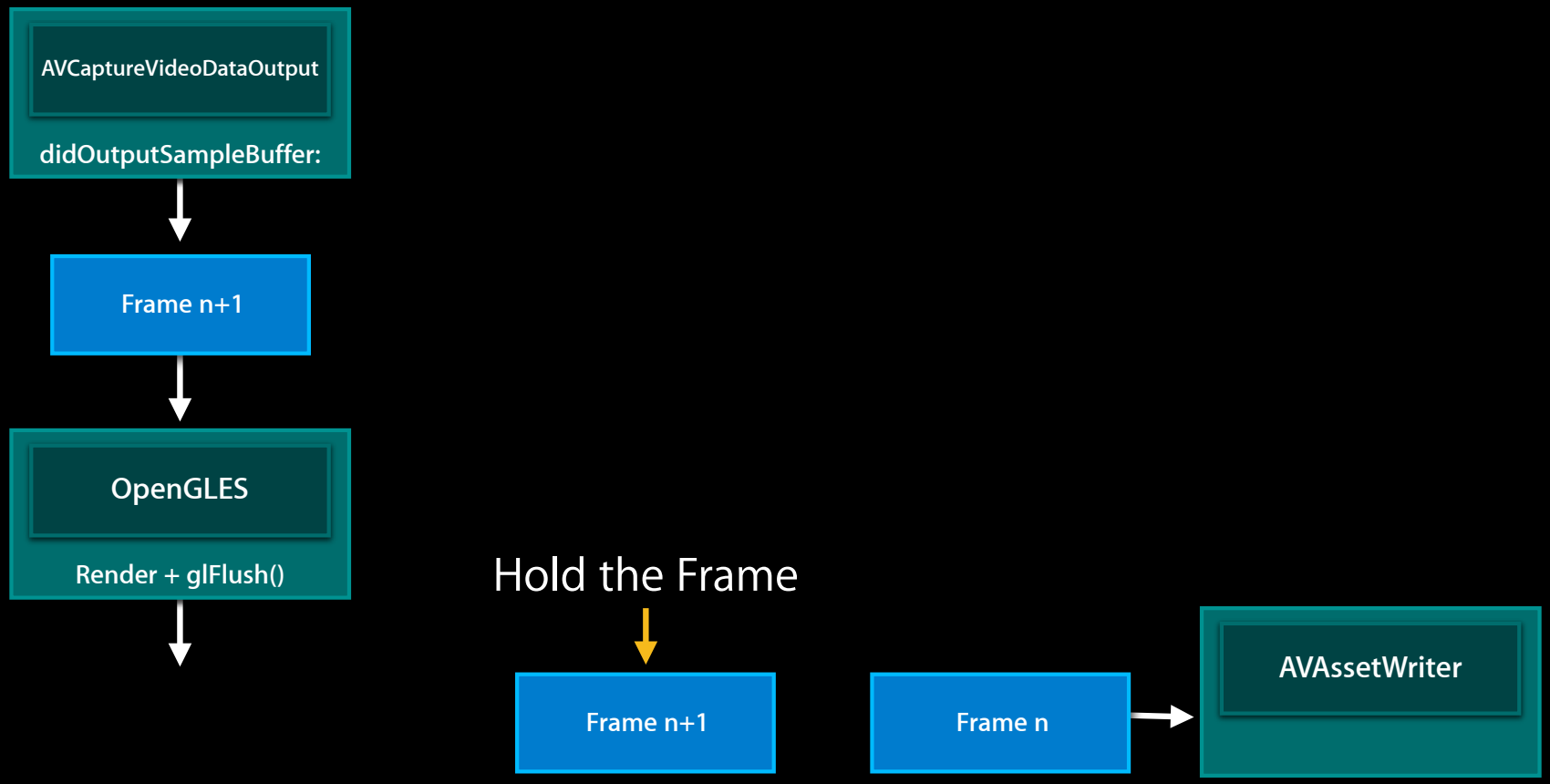
Solving Performance Problems



Solving Performance Problems



Solving Performance Problems



Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

- `glFlush()` is not necessary if you present the render buffer for preview

Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

- `glFlush()` is not necessary if you present the render buffer for preview
- In iOS 6, `glFinish()` is not necessary

Solving Performance Problems

Rendering with OpenGL, writing to AVAssetWriter

- `glFlush()` is not necessary if you present the render buffer for preview
- In iOS 6, `glFinish()` is not necessary
- AVAssetWriter ensures the GPU rendering is complete before writing

Solving Performance Problems

How to draw my own preview (fast!)

Solving Performance Problems

How to draw my own preview (fast!)

- Use AVCaptureVideoPreviewLayer + your own CALayers for simple overlays

Solving Performance Problems

How to draw my own preview (fast!)

- Use AVCaptureVideoPreviewLayer + your own CALayers for simple overlays
- Use OpenGL for preview if you are manipulating pixels

Solving Performance Problems

How to draw my own preview (fast!)

- Use AVCaptureVideoPreviewLayer + your own CALayers for simple overlays
- Use OpenGL for preview if you are manipulating pixels
- Review GLCameraRipple sample code

Solving Performance Problems

How to draw my own preview (fast!)

- Use AVCaptureVideoPreviewLayer + your own CALayers for simple overlays
- Use OpenGL for preview if you are manipulating pixels
- Review GLCameraRipple sample code
  - Operates in '420v'

Solving Performance Problems

How to draw my own preview (fast!)

- Use AVCaptureVideoPreviewLayer + your own CALayers for simple overlays
- Use OpenGL for preview if you are manipulating pixels
- Review GLCameraRipple sample code
  - Operates in '420v'
- Review RosyWriter sample code

Solving Performance Problems

How to draw my own preview (fast!)

- Use AVCaptureVideoPreviewLayer + your own CALayers for simple overlays
- Use OpenGL for preview if you are manipulating pixels
- Review GLCameraRipple sample code
  - Operates in '420v'
- Review RosyWriter sample code
  - Operates in 'BGRA'

What You Will Learn

- Performance improvements in Mac OS X 10.8
- Camera ecosystem
- New AV Foundation capture features in iOS 6
- Solutions for performance problems in your capture app
- Synchronizing motion data with video

Synchronizing Motion Data with Video

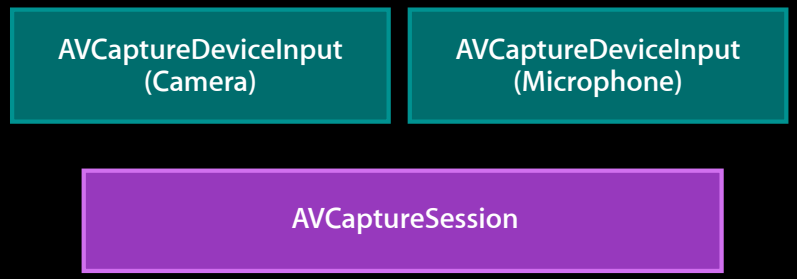
Demo

VideoSnake

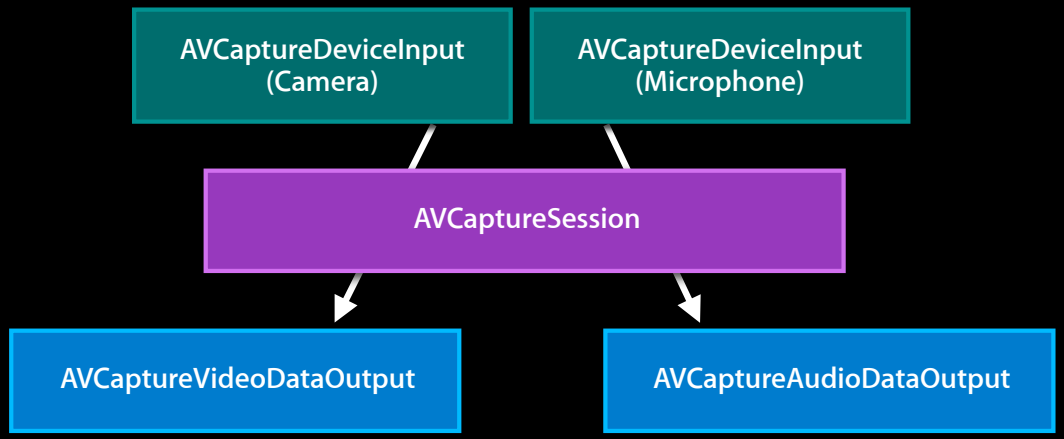
Walker Eagleston
Core Media Engineering

VideoSnake

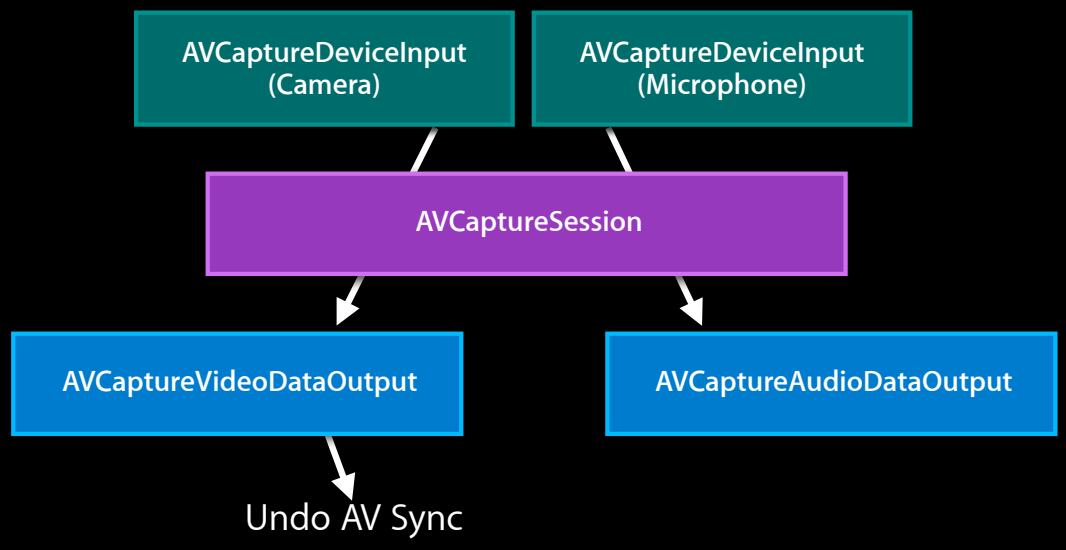
VideoSnake


AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession

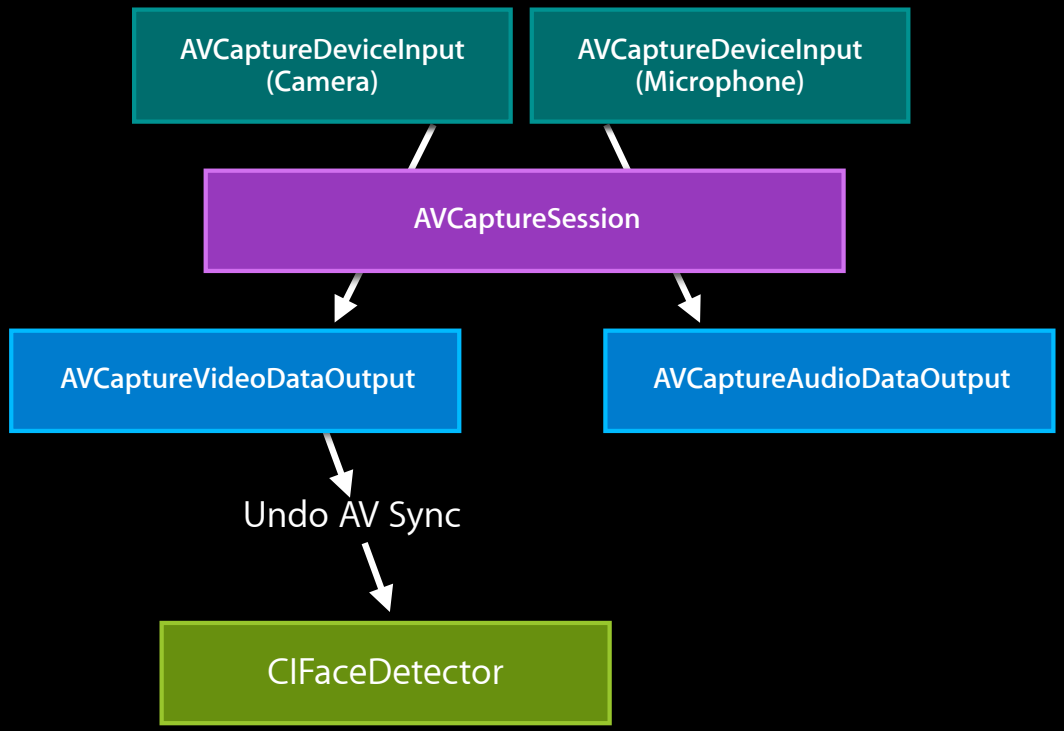
VideoSnake


AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput

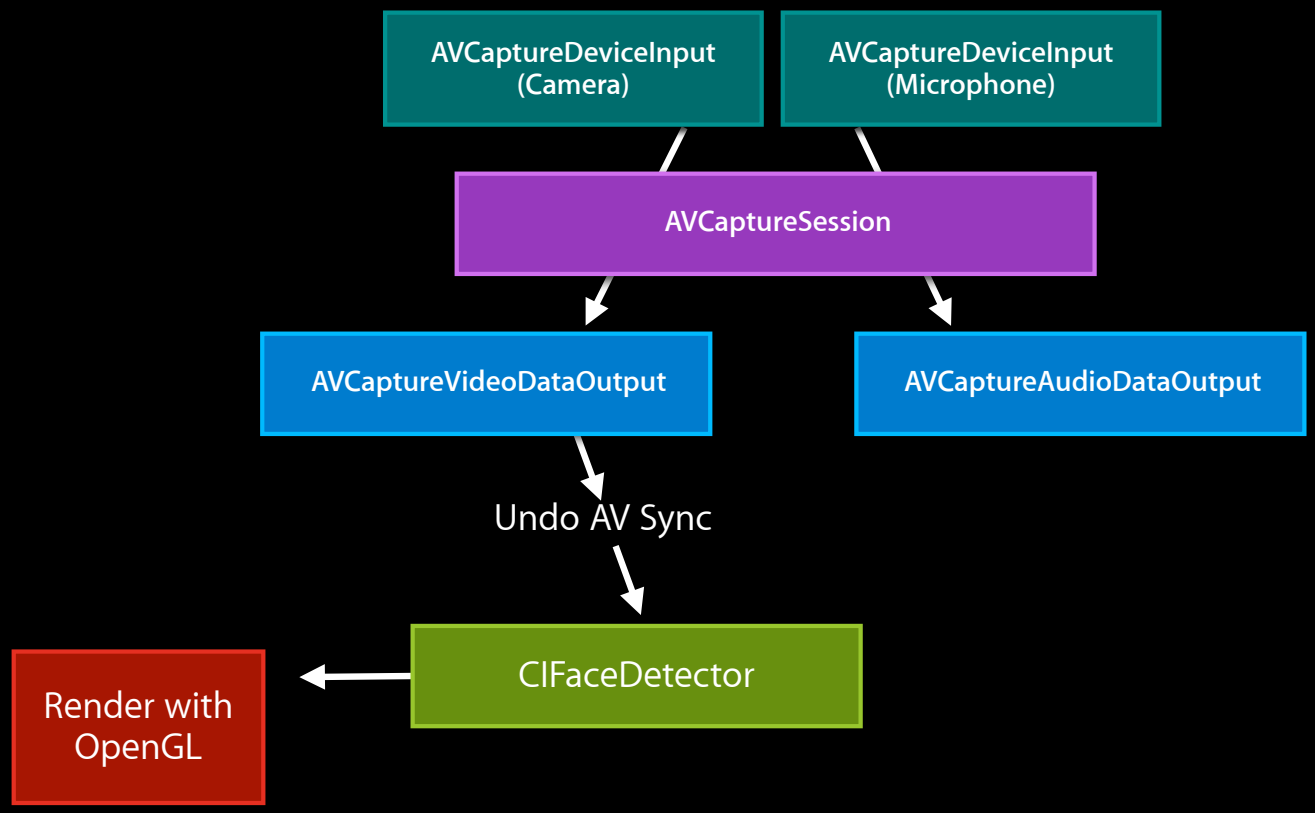
VideoSnake


AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput
Undo AV Sync

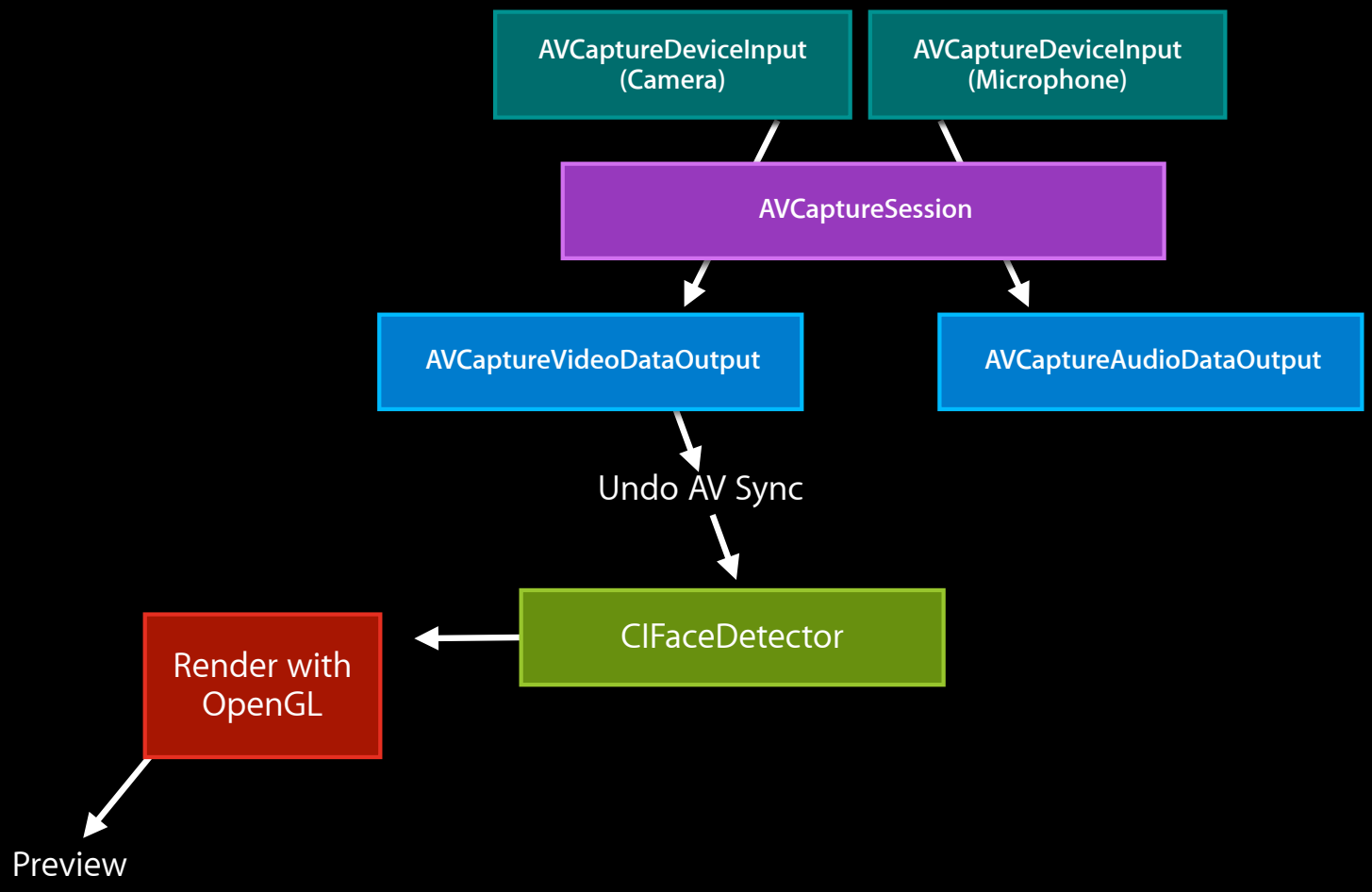
VideoSnake


AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput
Undo AV Sync
CIFaceDetector

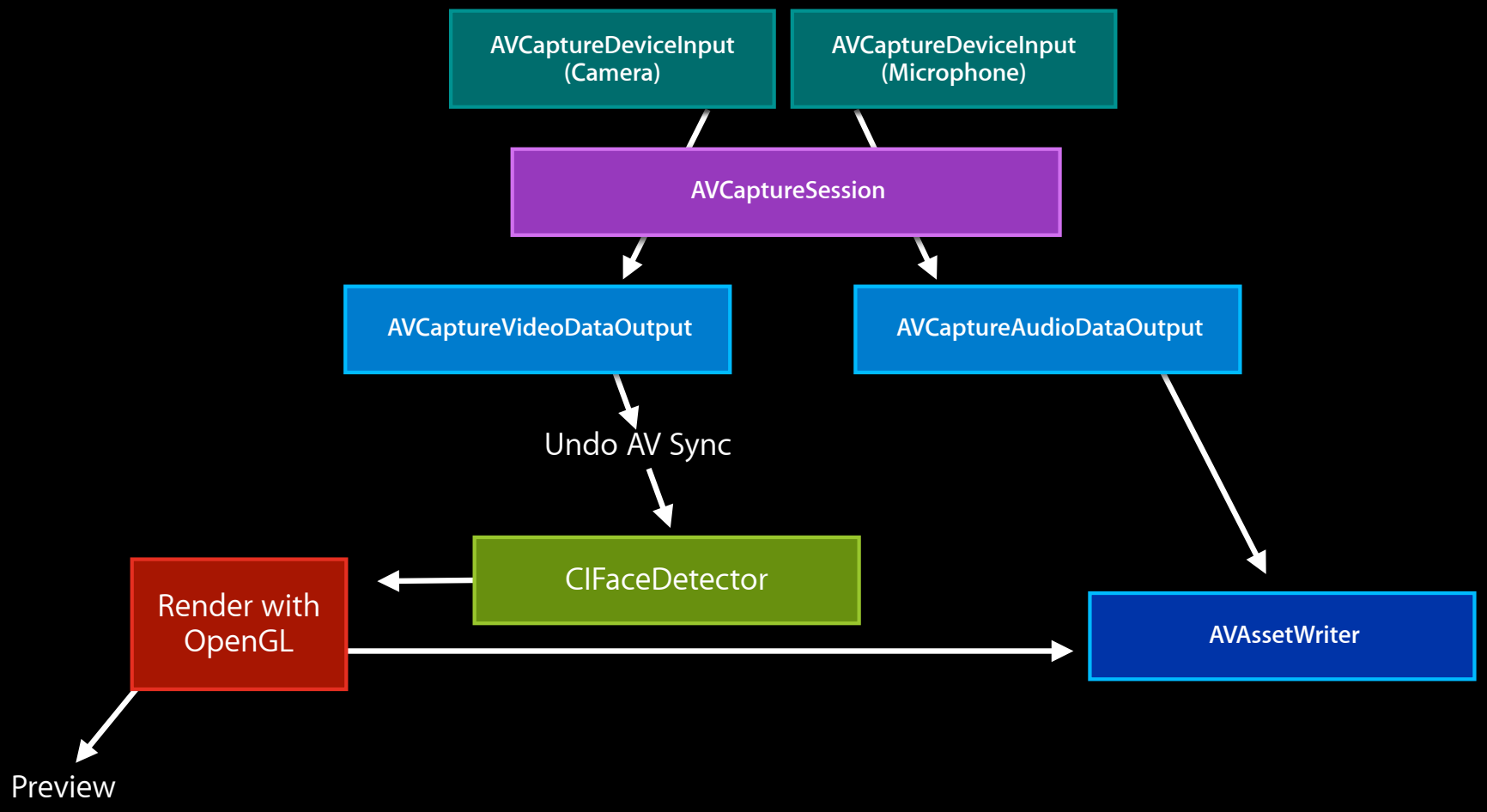
VideoSnake


AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput
Undo AV Sync
CIFaceDetector
Render with
OpenGL

VideoSnake

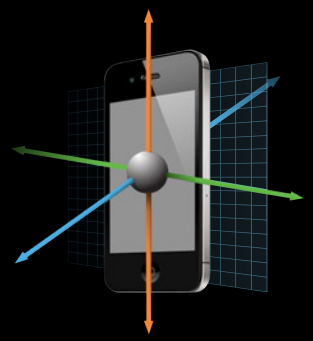

AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput
Undo AV Sync
CIFaceDetector
Render with
OpenGL
Preview

VideoSnake

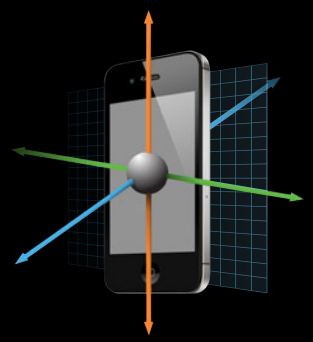

AVCaptureDeviceInput
(Camera)
AVCaptureDeviceInput
(Microphone)
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput
Undo AV Sync
CIFaceDetector
Render with
OpenGL
AVAssetWriter
Preview

Synchronizing Motion Data with Video

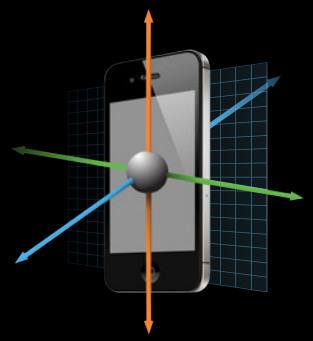
Synchronizing Motion Data with Video



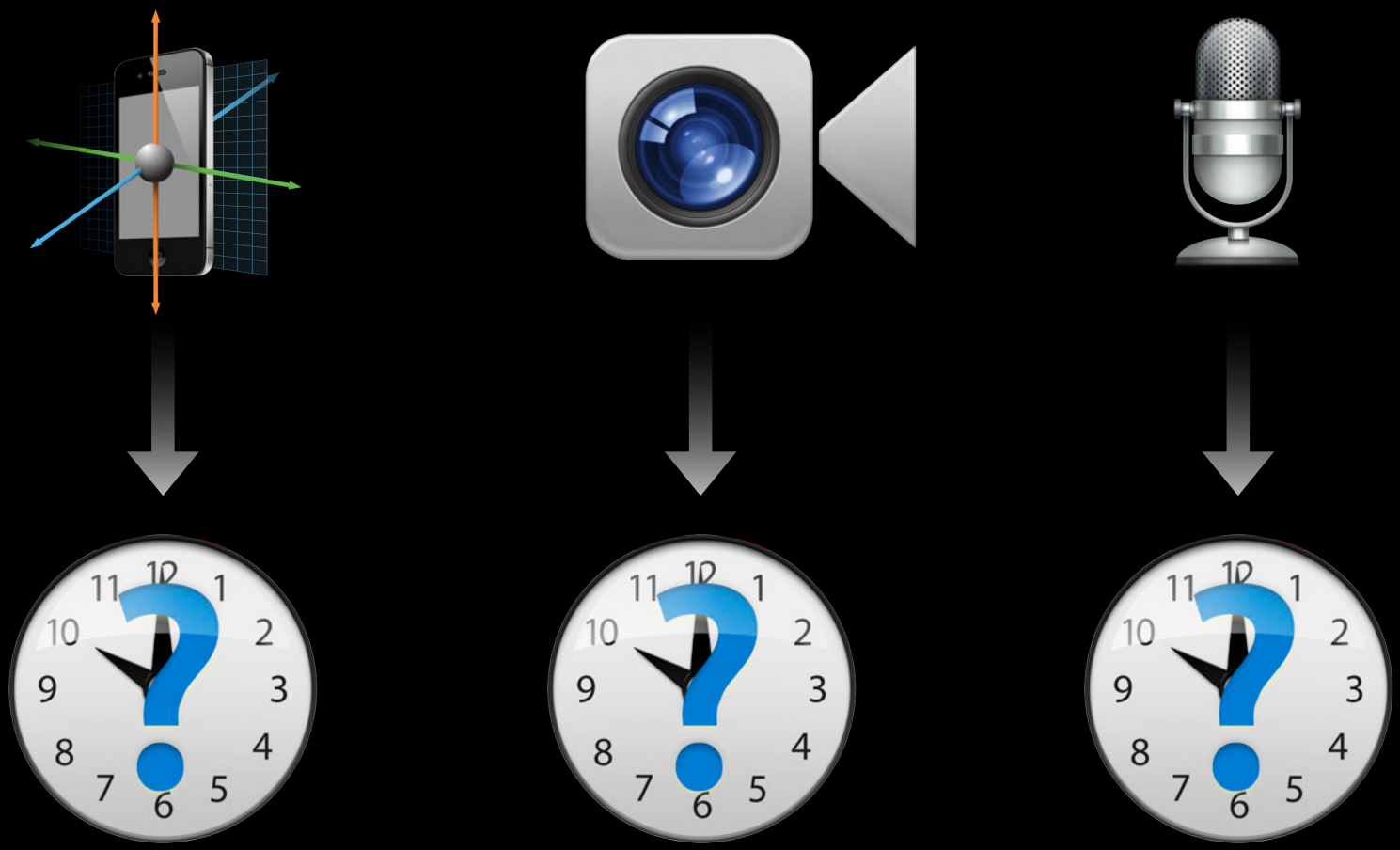
Synchronizing Motion Data with Video



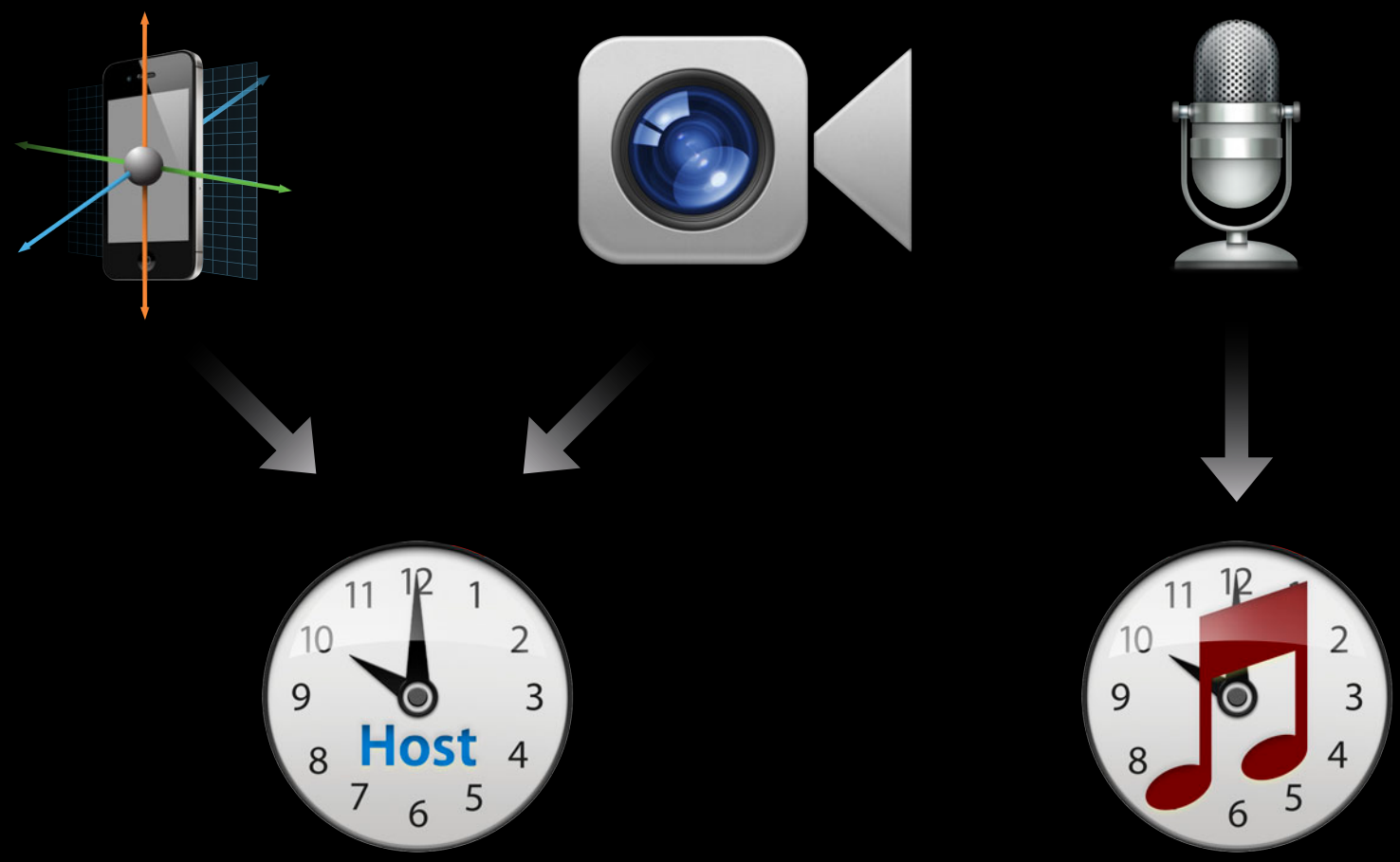
Synchronizing Motion Data with Video



Synchronizing Motion Data with Video



Synchronizing Motion Data with Video



Synchronizing Motion Data with Video

Synchronizing Motion Data with Video

- CoreMotion samples contain a timestamp

Synchronizing Motion Data with Video

- CoreMotion samples contain a timestamp

```
NSTimeInterval motionTimestamp = [(CMDeviceMotion *)motion timestamp];
```

Synchronizing Motion Data with Video

- CoreMotion samples contain a timestamp

```
NSTimeInterval motionTimestamp = [(CMDeviceMotion *)motion timestamp];
```

- Timestamp is the mach_absolute_time() of the motion

Synchronizing Motion Data with Video

- CoreMotion samples contain a timestamp

```
NSTimeInterval motionTimestamp = [(CMDeviceMotion *)motion timestamp];
```

- Timestamp is the mach_absolute_time() of the motion
- CoreMotion uses the host time clock

Synchronizing Motion Data with Video

- CoreMotion samples contain a timestamp

```
NSTimeInterval motionTimestamp = [(CMDeviceMotion *)motion timestamp];
```

- Timestamp is the mach_absolute_time() of the motion
- CoreMotion uses the host time clock
- CoreMotion sampling rate should be at least 2x your video frame rate

Synchronizing Motion Data with Video

Video timestamps

Synchronizing Motion Data with Video

Video timestamps

- Sample buffers contain a timestamp

Synchronizing Motion Data with Video

Video timestamps

- Sample buffers contain a timestamp

```
CMTime pts = CMSampleBufferGetPresentationTime(sampleBuffer);
```

Synchronizing Motion Data with Video

Video timestamps

- Sample buffers contain a timestamp

```
CMTime pts = CMSampleBufferGetPresentationTime(sampleBuffer);
```

- Presentation time is the mach_absolute_time() of the frame

Synchronizing Motion Data with Video

Video timestamps

- Sample buffers contain a timestamp

```
CMTime pts = CMSampleBufferGetPresentationTime(sampleBuffer);
```

- Presentation time is the mach_absolute_time() of the frame
- Front and Back Camera AVCaptureDevices use the host time clock

Synchronizing Motion Data with Video

Audio timestamps

Synchronizing Motion Data with Video

Audio timestamps

- Audio sample buffers contain n samples (frames) of audio

Synchronizing Motion Data with Video

Audio timestamps

- Audio sample buffers contain n samples (frames) of audio
- Presentation time is the time at which the first sample in the buffer was picked up by the microphone

Synchronizing Motion Data with Video

Audio timestamps

- Audio sample buffers contain n samples (frames) of audio
- Presentation time is the time at which the first sample in the buffer was picked up by the microphone
- The audio AVCaptureDevice uses the audio clock

Synchronizing Motion Data with Video

A/V Sync

Synchronizing Motion Data with Video

A/V Sync

- Audio clock != video clock

Synchronizing Motion Data with Video

A/V Sync

- Audio clock != video clock
- Audio and video might drift

Synchronizing Motion Data with Video

A/V Sync

- Audio clock != video clock
- Audio and video might drift
- When recording audio, the video sample buffers are synced to the audio (master) clock

Synchronizing Motion Data with Video

A/V Sync

- Audio clock != video clock
- Audio and video might drift
- When recording audio, the video sample buffers are synced to the audio (master) clock
- Re-clocking alters the video timestamps

Synchronizing Motion Data with Video

“Undoing” A/V Sync

Synchronizing Motion Data with Video

“Undoing” A/V Sync

```
CMClockRef audioClock = NULL, videoClock = NULL;
OSStatus err = CMAudioClockCreate( NULL, &audioClock );
videoClock = CMClockGetHostTimeClock();
CMTime pts = CMSampleBufferGetPresentationTime(videoBuffer);
CMTime convertedPTS = CMSyncConvertTime(pts, audioClock, videoClock);
// now match convertedPTS with CoreMotion timestamps
```

Synchronizing Motion Data with Video

“Undoing” A/V Sync

```
CMClockRef audioClock = NULL, videoClock = NULL;
OSStatus err = CMAudioClockCreate( NULL, &audioClock );
videoClock = CMClockGetHostTimeClock();
CMTime pts = CMSampleBufferGetPresentationTime(videoBuffer);
CMTime convertedPTS = CMSyncConvertTime(pts, audioClock, videoClock);
// now match convertedPTS with CoreMotion timestamps
```

Synchronizing Motion Data with Video

“Undoing” A/V Sync

```
CMClockRef audioClock = NULL, videoClock = NULL;
OSStatus err = CMAudioClockCreate( NULL, &audioClock );
videoClock = CMClockGetHostTimeClock();
CMTime pts = CMSampleBufferGetPresentationTime(videoBuffer);
CMTime convertedPTS = CMSyncConvertTime(pts, audioClock, videoClock);
// now match convertedPTS with CoreMotion timestamps
```

Summary

Summary

- What's new in camera capture

Summary

- What's new in camera capture
  - Mac OS X 10.8 performance improvements

Summary

- What's new in camera capture
  - Mac OS X 10.8 performance improvements
  - iOS camera ecosystem

Summary

- What's new in camera capture
  - Mac OS X 10.8 performance improvements
  - iOS camera ecosystem
  - New iOS 6 AV Foundation capture features

Summary

- What's new in camera capture
  - Mac OS X 10.8 performance improvements
  - iOS camera ecosystem
  - New iOS 6 AV Foundation capture features
  - Solving performance problems in your capture app

Summary

- What's new in camera capture
  - Mac OS X 10.8 performance improvements
  - iOS camera ecosystem
  - New iOS 6 AV Foundation capture features
  - Solving performance problems in your capture app
  - Synchronizing motion data with video

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

Audio Session and Multiroute Audio in iOS	Pacific Heights Tuesday 2:00PM
Audio and Video for Media and Games	Presidio Thursday 9:00AM
Understanding Core Motion	Pacific Heights Friday 10:15AM

Labs

OS X Capture Lab	GMG Lab A Tuesday 9:00AM - 1:30PM
AV Foundation Lab	GMG Lab A Tuesday 2:00PM - 6:00 PM
AVAudioSession Lab	GMG Lab D Wednesday 4:30PM - 6:00 PM
iOS Camera Capture Lab	GMG Lab D Thursday 2:00PM - 6:00 PM
AV Foundation Lab	GMG Lab C Thursday 2:00PM - 6:00 PM
iOS Camera Capture Lab	GMG Lab D Friday 9:00AM - 11:15 AM

WWDC2012