Capturing From the Camera Using AV Foundation on iOS 5

API enhancements and performance improvements

Session 419

Brad Ford
iOS Engineering

What You Will Learn

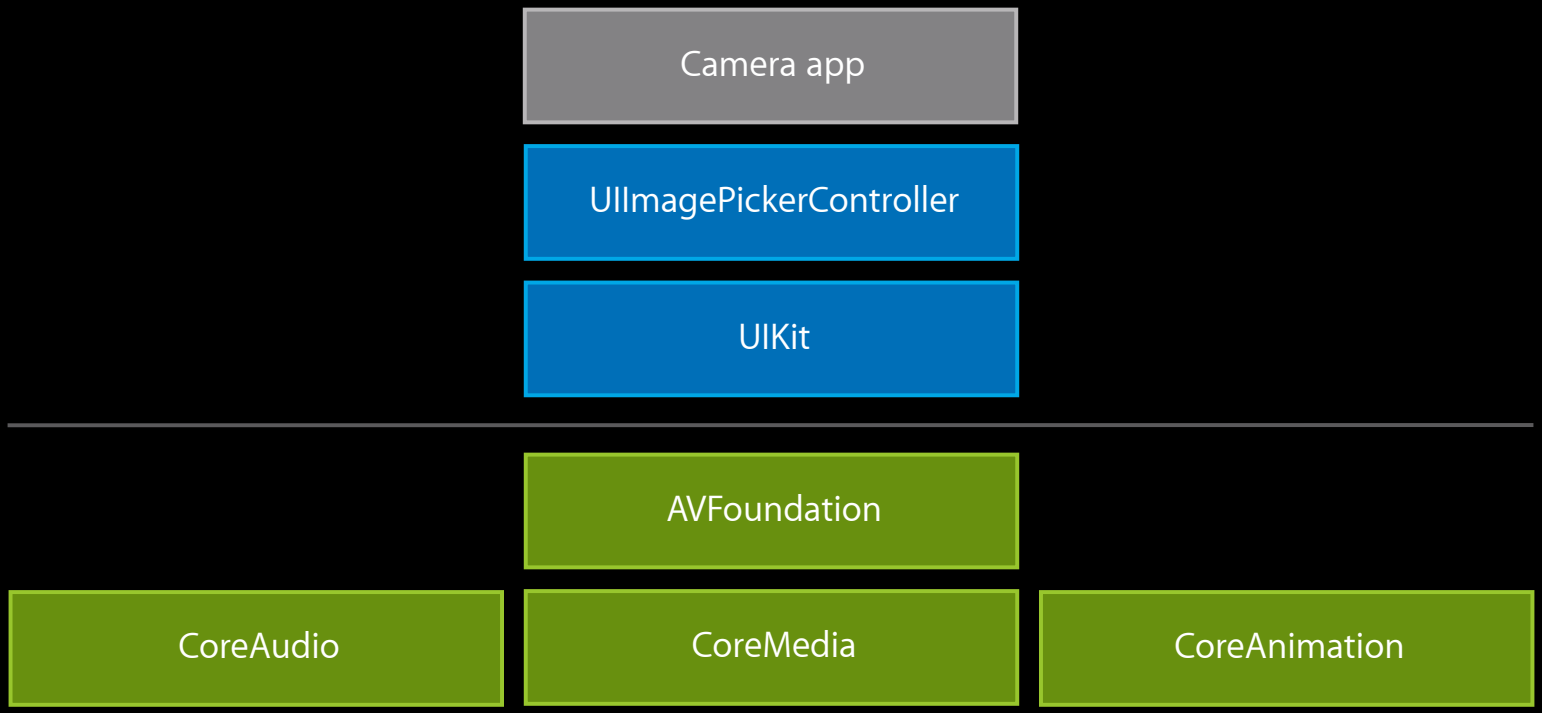
- Which iOS 5 captures APIs to use in your app
- The AV Foundation capture programming model
- iOS 5 Performance improvements in AV Foundation
- iOS 5 AV Foundation API enhancements

Sample Code for This Session

- ChromaKey
- RosyWriter
- ‘StacheCam
- StopNGo (iOS Edition)

Materials available at:
https://developer.apple.com/wwdc/schedule/details.php?id=419

Technology Framework



Using the Camera in iOS 5

Simple programmatic access

API for high, medium, or low quality recording

Hideable camera controls UI

```
-(void)takePicture
-(BOOL)startVideoCapture
```

```
-(void)setCameraDevice:
-(void)setCameraFlashMode:
```

User touch to focus, like Camera app

User touch and hold to lock AE and AF

Why Use AV Foundation Capture?

Full access to the camera

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

Why Use AV Foundation Capture?

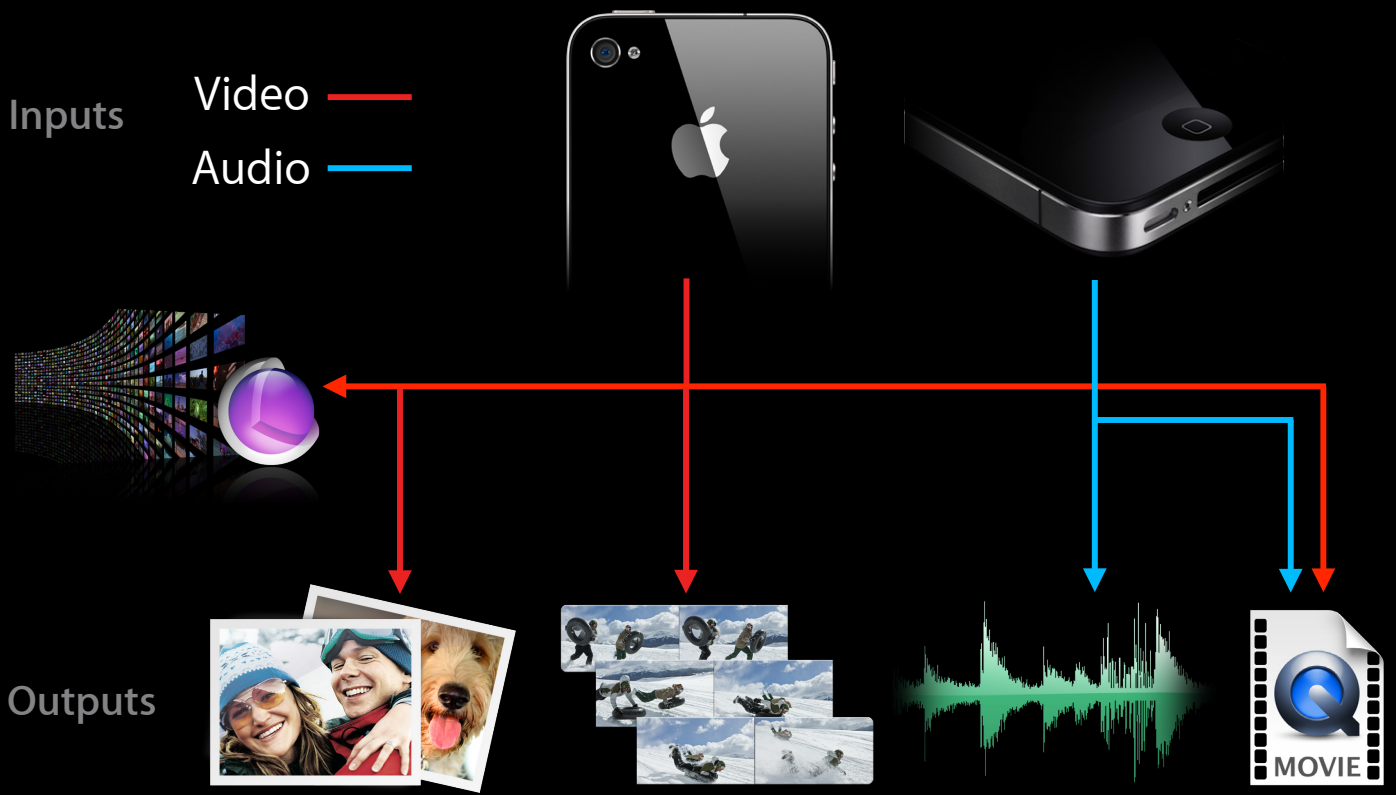
Flexible output

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

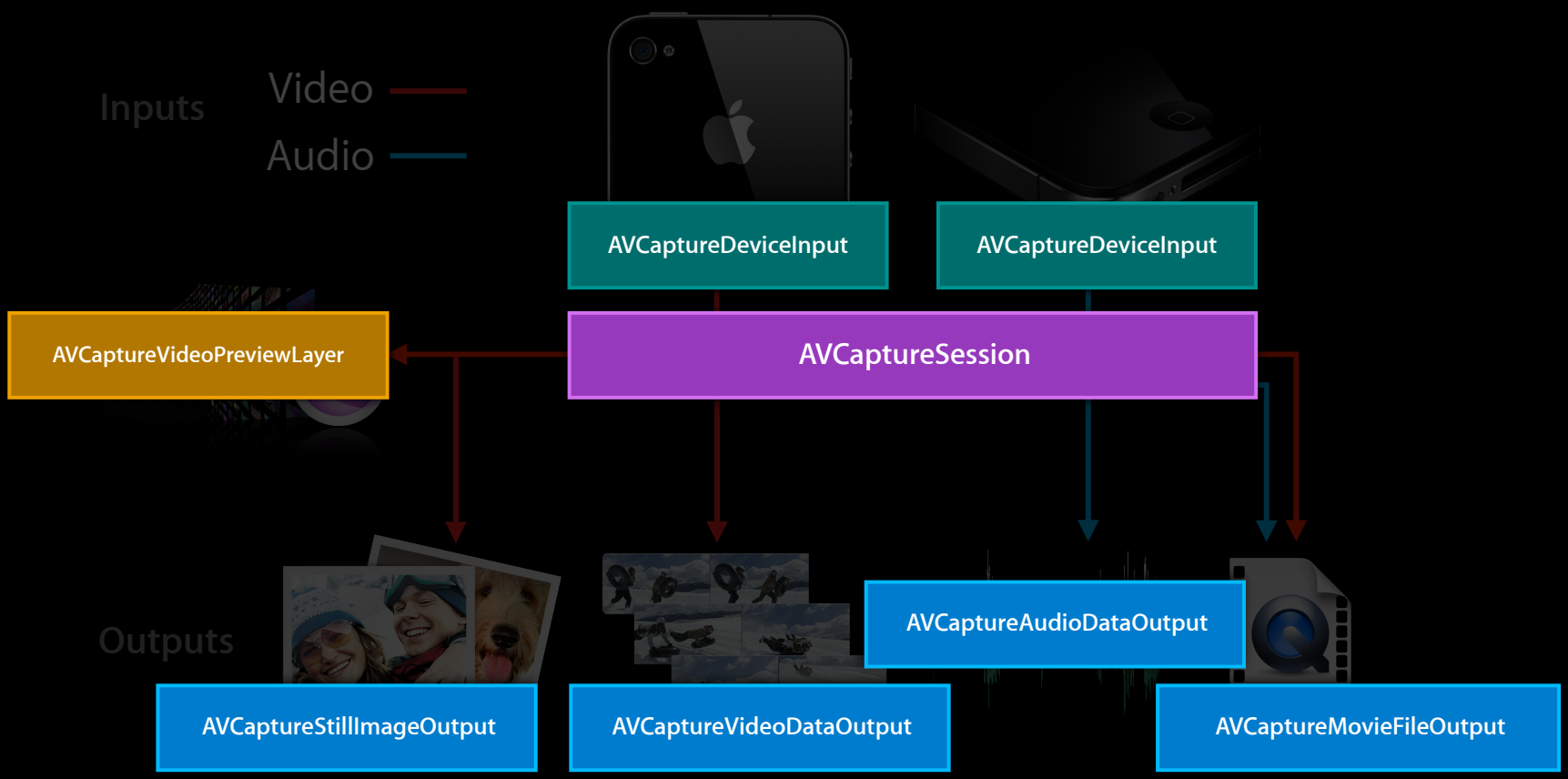
AV Foundation Capture

Programming model

Capture Basics—Inputs and Outputs



Capture Basics—Inputs and Outputs



Capture Use Cases We Will Cover

- Process video frames from the camera and render with OpenGL
- Process live video and audio, and write them to a QuickTime movie
- Scan video frames for patterns using the flash and VideoDataOutput
- Process captured still images with CoreImage

Capture Use Cases We Will Cover

- Process video frames from the camera and render with OpenGL
- Process live video and audio, and write them to a QuickTime movie
- Scan video frames for patterns using the flash and VideoDataOutput
- Process captured still images with CoreImage

Demo

ChromaKey

Sylvain Nuz
Core Media Engineering

ChromaKey

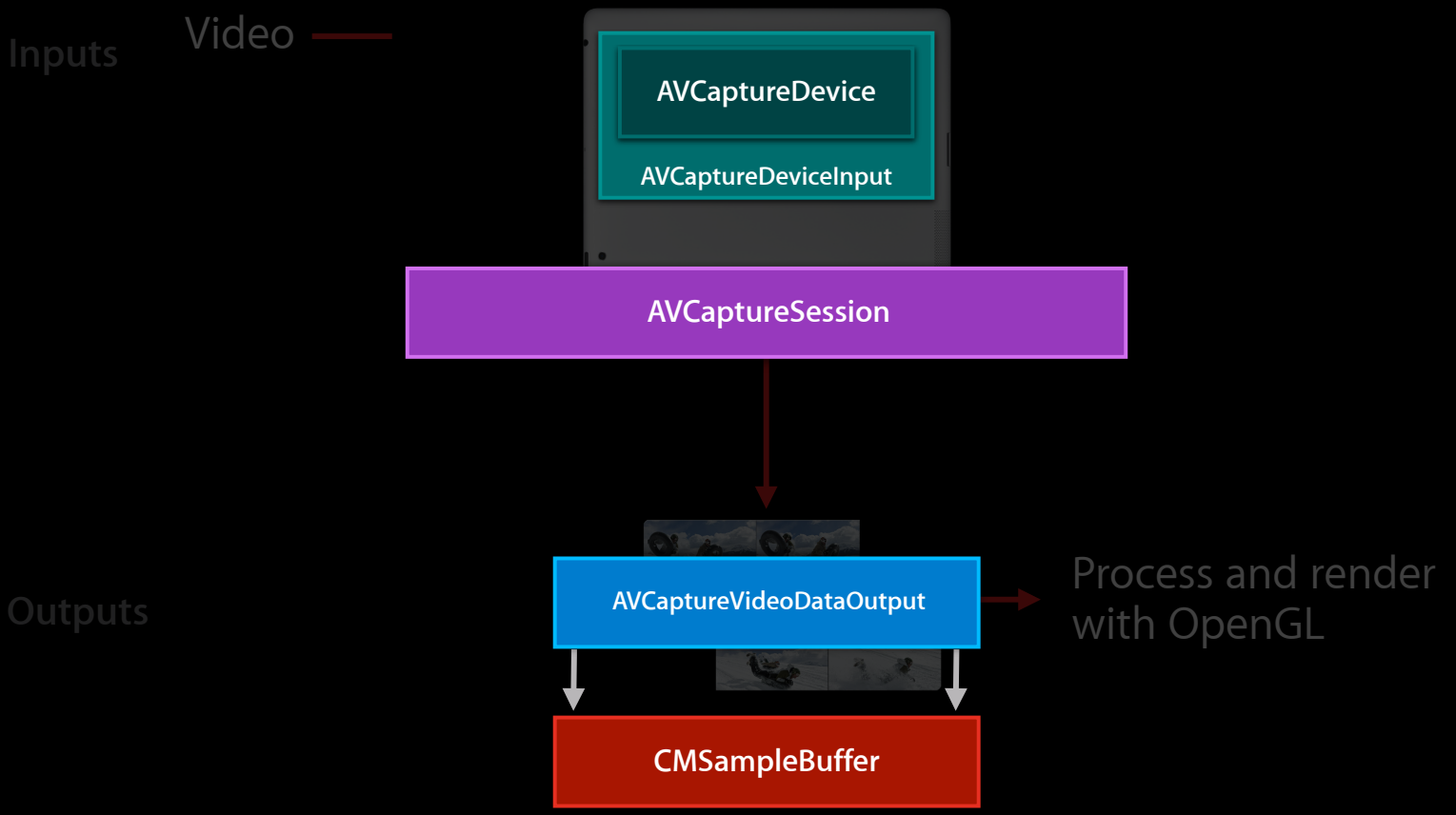
Inputs

Video

Outputs

Process and render with OpenGL

ChromaKey


Inputs
Video
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureSession
Outputs
AVCaptureVideoDataOutput
Process and render with OpenGL
CMSampleBuffer

ChromaKey

- Create an AVCaptureSession

```
AVCaptureSession *session = [[AVCaptureSession alloc] init];
session.sessionPreset = AVCaptureSessionPreset1280x720;
```

- Find a suitable AVCaptureDevice

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

- Create and add an AVCaptureDeviceInput

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

ChromaKey

- Create and add an AVCaptureVideoDataOutput

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

- Configure your output, and start the session

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

- Implement your delegate callback

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

ChromaKey

- Lock autoexposure, focus, and white balance

```
AVCaptureDevice *device = self.device;
[device lockForConfiguration:&error];

[device setFocusMode:AVCaptureFocusModeLocked];
[device setExposureMode:AVCaptureExposureModeLocked];
[device setWhiteBalanceMode:AVCaptureWhiteBalanceModeLocked];

[device unlockForConfiguration];
```

ChromaKey

Bridging CoreVideo and OpenGLES

Brandon Corey
iOS Engineering

ChromaKey

CVOpenGLESTextureCache

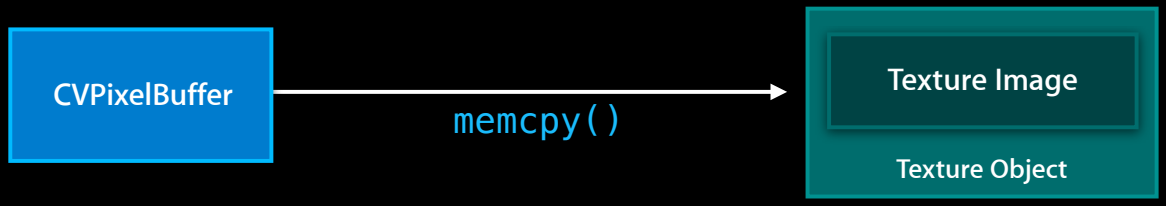
- Bridges CoreVideo PixelBuffers to OpenGLES textures
  - Avoids copies to and from the GPU
  - Allows recycling of textures
- Supported in OpenGL ES 2.0 and later
- Defined in <CoreVideo/CVOpenGLESTextureCache.h>

ChromaKey

- Efficiently maps a BGRA buffer as a source texture (OpenGL ES 2.0)

```
CVOpenGLESTextureCacheCreateTextureFromImage(
    kCFAllocatorDefault,
    openGLESTextureCacheRef,
    pixelBuffer, // a CVPixelBuffer received from the delegate callback
    NULL, // optional texture attributes
    GL_TEXTURE_2D, // the target
    GL_RGBA, // internal format
    width,
    height,
    GL_BGRA, // pixel format
    GL_UNSIGNED_BYTE, // data type of the pixels
    0, // plane index to map (BGRA has only one plane)
    &outTexture );
```

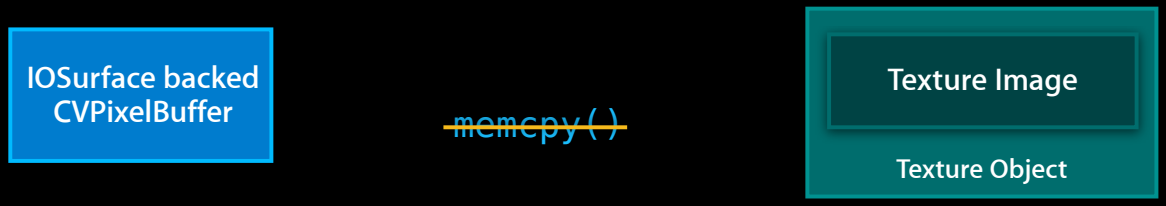
Standard Texture Binding



- Uploading data to a Texture Image

```
GLuint texture;
glGenTexture(1, &texture);
void *pixelData = CVPixelBufferGetBaseAddress(pixelBuffer);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 1280, 720,
             0, GL_BGRA, GL_UNSIGNED_BYTE, pixelData);
```

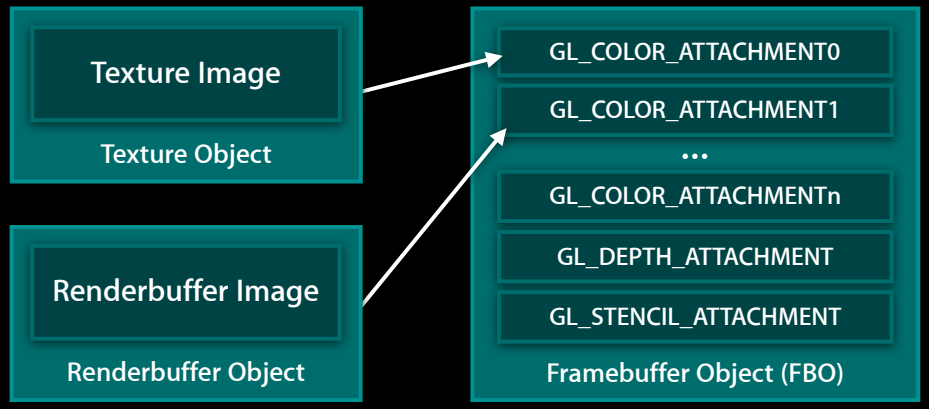
CVOpenGLES Texture Binding



- Binding data to a Texture Object

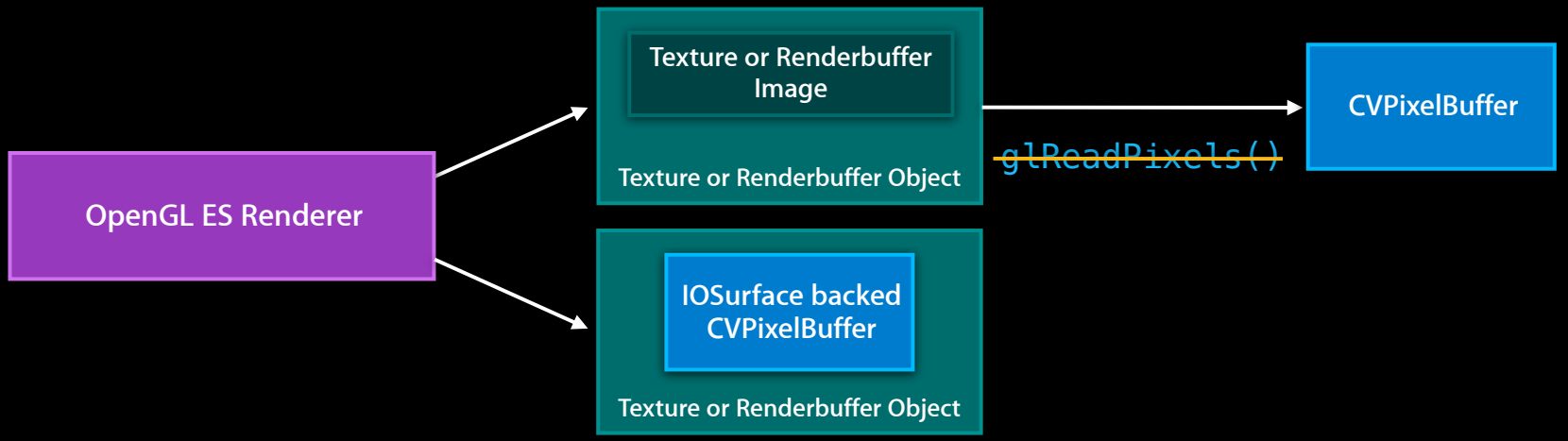
```
GLuint texture;
CVPixelBufferRef pixelBuffer;

CVOpenGLESTextureCacheCreateTextureFromImage(kCFAllocatorDefault,
    textureCacheRef, pixelBuffer, NULL,
    GL_RGBA, 1280, 720, GL_BGRA, GL_UNSIGNED_BYTE, 0, &texture);
```

Getting Data Back Out



- Have OpenGL ES render to a Framebuffer Object (FBO)
- Done one of two ways
  - Create and attach a Renderbuffer to an FBO
  - Create and attach a Texture to an FBO
- glReadPixels or glTexImage2D to move data into a buffer

CVOpenGLES Framebuffer Binding



- Binding to a Texture and attaching to a Framebuffer Object

```
GLuint texture, fbo;
glGenFrameBuffer(1, &fbo);
CVOpenGLESTextureCacheCreateTextureFromImage(... &texture);
glBindFrameBuffer(GL_FRAMEBUFFER, fbo);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
                       GL_TEXTURE_2D, texture, 0);
```

CVOpenGLESTextureCache

Usage notes

- Buffers from AVCapture and AVAssetReader are created with the appropriate formatting
- If you create your own CVPixelBuffers, you must pass `kCVPixelBufferIOSurfacePropertiesKey` as a pixelBufferAttribute to `CVPixelBufferCreate()` or `CVPixelBufferPoolCreate()`
- If you use AVAssetWriter, be sure to use the pixel buffer pool it provides

CVOpenGLESTextureCache

Usage notes (continued)

- BGRA, 420v, and 420f are supported
- OpenGLES now supports `GL_RED` and `GL_RG` single channel render targets (iPad 2 only)

Performance Enhancements in iOS 5

AVCaptureVideoDataOutput

Brad Ford
iOS Engineering

AVCaptureVideoDataOutput

Usage

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

AVCaptureVideoDataOutput

Performance tips

- Set the `minFrameDuration` property to cap the max frame rate
- Configure the session to output the lowest practical resolution
- Set the `alwaysDiscardsLateVideoFrames` property to YES (the default) for early, efficient dropping of late video frames
- Your sample buffer delegate's callback must be FAST!

AVCaptureVideoDataOutput

Supported pixel formats

- '420v'
  - Planar yuv 4:2:0 video format (luma and chroma are in separate planes)
  - Chroma is subsampled in horizontal and vertical direction
  - "Video Range" samples (16-235)
  - This is the default format on all iOS 5 supported cameras
- '420f'
  - Full color range planar yuv (0-255)
  - This is the default format when using AVCaptureSessionPresetPhoto
- 'BGRA'
  - Blue / Green / Red / Alpha, 8 bits per pixel (more than 2x bandwidth)

AVCaptureVideoDataOutput

Enhancements in iOS 5

- Support for rotation of CVPixelBuffers
  - Rotation is hardware accelerated
  - Use AVCaptureConnection's `setVideoOrientation`: property
  - All 4 AVCaptureVideoOrientations are supported
  - Default is non-rotated buffers
    - Front camera = `AVCaptureVideoOrientationLandscapeLeft`
    - Back camera = `AVCaptureVideoOrientationLandscapeRight`

AVCaptureVideoDataOutput

Enhancements in iOS 5

- Support for pinning of minimum frame rate
  - Enables fixed frame rate captures
  - Use AVCaptureConnection's `-setMaxFrameDuration:` property
  - WARNING: Fixed frame rate captures may result in reduced image quality in low light

AVCaptureSession + AVCaptureVideoDataOutput

AVCaptureSession's `setSessionPreset:` affects video data output resolution

High	Highest recording quality
Medium	Suitable for WiFi sharing
Low	Suitable for 3G sharing
640x480	VGA
1280x720	720p HD
Photo	Full photo resolution
352x288	CIF for streaming apps
iFrame 960x540	540p HD for editing
iFrame 1280x720	720p HD for editing

AVCaptureVideoDataOutput

Supported resolutions

	Back Camera			Front Camera
Preset	iPhone 3GS	iPhone 4	iPad 2 / iPod Touch	All models
High	640x480	1280x720	1280x720	640x480
Medium	480x360			
Low	192x144			
Photo	512x384	852x640	960x720	640x480

AVCaptureVideoDataOutput

Supported resolutions (continued)

- Special consideration for Photo mode as of iOS 4.3
- AVCaptureSessionPresetPhoto delivers
  - Full resolution buffers to AVCaptureStillImageOutput
  - Preview sized buffers to AVCaptureVideoDataOutput
- Aspect ratio is unchanged

AVCaptureVideoDataOutput

Supported resolutions (continued)

	Back Camera			Front Camera
Preset	iPhone 3GS	iPhone 4	iPad 2 / iPod touch	All models
352x288	✓	✓	✓	✓
640x480	✓	✓	✓	✓
1280x720		✓	✓	
iFrame960x540		✓	✓	
iFrame1280x720		✓	✓	

What is iFrame?

What Are I Frames?

- I—Intra

- P—Predictive

- B—Bidirectionally predictive

What Is iFrame?

- Apple "ecosystem friendly" video
- Supported by 30+ camera/camcorder models
  - H.264 I frame only video + AAC or PCM audio
  - Constant frame rate (25 or 29.97 fps)
  - ~30 megabits/s @ 960x540 or ~40 megabits/s @ 1280x720
  - MOV or MP4 container
- Great for editing
- Produces larger files

Supported on all iOS devices with HD cameras

Capture Use Cases We Will Cover

- Process video frames from the camera and render with OpenGL
- Process live video and audio, and write them to a QuickTime movie
- Scan video frames for patterns using the flash and VideoDataOutput
- Process captured still images with CoreImage

Demo

RosyWriter

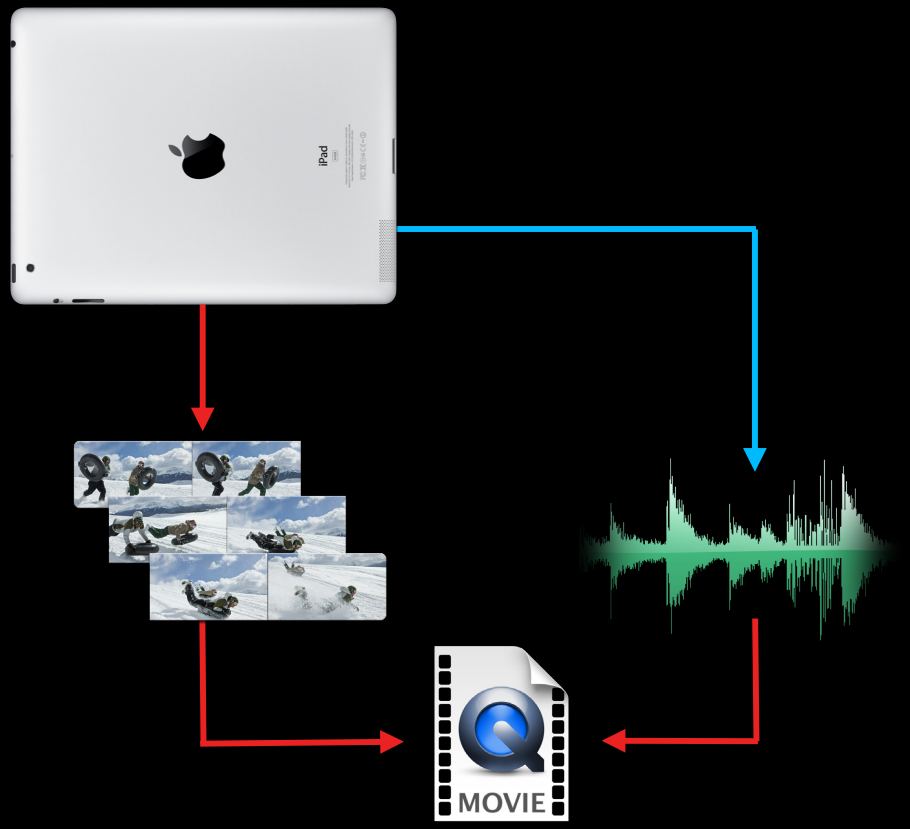
Matthew Calhoun
iOS Engineering

RosyWriter

Inputs

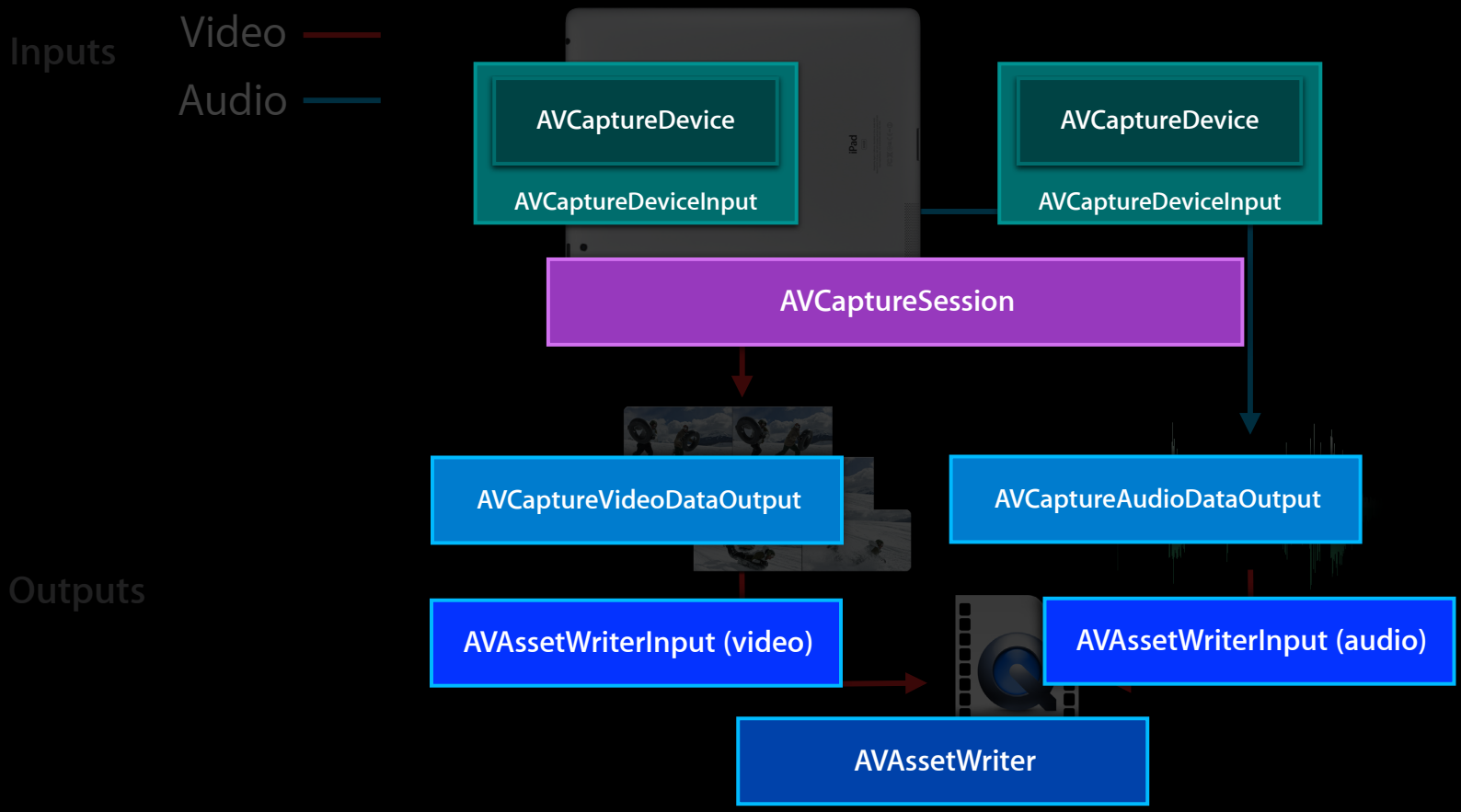
Video

Audio



Outputs

RosyWriter


Inputs
Video
Audio
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureSession
AVCaptureVideoDataOutput
AVCaptureAudioDataOutput
Outputs
AVAssetWriterInput (video)
AVAssetWriterInput (audio)
AVAssetWriter

What Is an AVAsset?

- Defined in `<AVFoundation/AVAsset.h>`
- Abstraction for a media asset
  - Can be URL-based
  - Can be stream-based
  - Can be inspected for properties
  - Can be played with an AVPlayer
  - Can be read with an AVAssetReader
  - Can be exported with an AVAssetExportSession

Can be written with an AVAssetWriter

RosyWriter

- Create an AVAssetWriter

```
AVAssetWriter *writer = [[AVAssetWriter alloc] initWithURL:url
    fileType:AVFileTypeQuickTimeMovie error:&error];
```

- Create and add AVAssetWriterInputs

```
AVAssetWriterInput *videoInput = [[AVAssetWriterInput alloc]
      initWithMediaType:AVMediaTypeVideo outputSettings:outputSettings];
[videoInput setExpectsMediaDataInRealTime:YES];
[writer addInput:videoInput];
dispatch_queue_t serial_queue = dispatch_queue_create("myQueue", NULL);
[output setSampleBufferDelegate:self queue:serial_queue];
[session startRunning];
```

- Append samples in AVCaptureDataOutputs' delegate callbacks

```
[writer startWriting];
if ( [videoInput isReadyForMoreMediaData] )
  [videoInput appendSampleBuffer:sampleBuffer];
```

AVAssetWriter or AVCaptureMovieFileOutput

Which one to use?

- AVCaptureMovieFileOutput
  - Requires no set up (automatically uses AVCaptureSessionPreset)
  - Can record multiple movie files
  - Supports max file size, duration, and minimum free disk space limits
  - Does not allow client access to video/audio buffers before writing
- AVAssetWriter
  - Requires set up of output settings
  - Is a one-shot writer
  - Allows client access to video/audio buffers before writing (via AVCaptureVideoDataOutput / AVCaptureAudioDataOutput)
  - Incurs more overhead than AVCaptureMovieFileOutput

AVAssetWriter

Sample video settings

- 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,
            AVVideoProfileLevelH264Main31, AVVideoProfileLevelKey,
            nil], AVVideoCompressionPropertiesKey,
        nil];
```

AVAssetWriter

Sample audio settings

- Settings are defined in <AVFoundation/AVAudioSettings.h>

```
AudioChannelLayout acl = {0};
acl.mChannelLayoutTag = kAudioChannelLayoutTagMono;

NSDictionary *outputSettings = [NSDictionary dictionaryWithObjectsAndKeys:
  [NSNumber numberWithInt:'aac '], AVFormatIDKey,
  [NSNumber numberWithInt:64000], AVEncoderBitRatePerChannelKey,
  [NSNumber numberWithInt:1], AVNumberOfChannelsKey,
  [NSNumber numberWithDouble:44100.], AVSampleRateKey,
  [NSData dataWithBytes:&layout length:sizeof(layout)], AVChannelLayoutKey,
  nil];
```

AVAssetWriter

Do's and don'ts

- DO:
  - Set `-expectsMediaDataInRealTime` to YES for each AVAssetWriterInput
  - Set AVAssetWriter's `movieFragmentInterval` to preserve recordings in the event of an interruption
  - Set AVCaptureVideoDataOutput's `-alwaysDiscardsLateVideoFrames` to NO
- DON'T:
  - Hold on to sample buffers outside the data output callbacks
  - Take a long time to process buffers in the data output callbacks

What Is a Movie Fragment?

- Fast start QuickTime movie

- Non fast start (captured) QuickTime movie

- QuickTime movie with movie fragments

Capture Use Cases We Will Cover

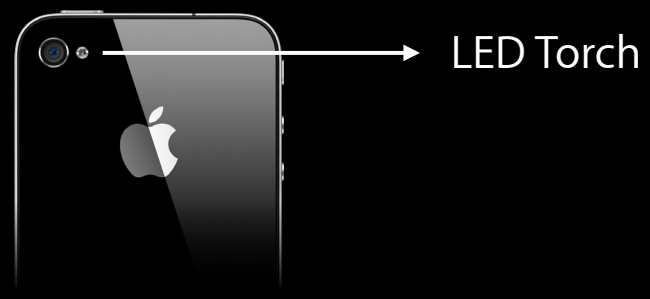
- Process video frames from the camera and render with OpenGL
- Process live video and audio, and write them to a QuickTime movie
- Scan video frames for patterns using the flash and VideoDataOutput
- Process captured still images with CoreImage

Demo

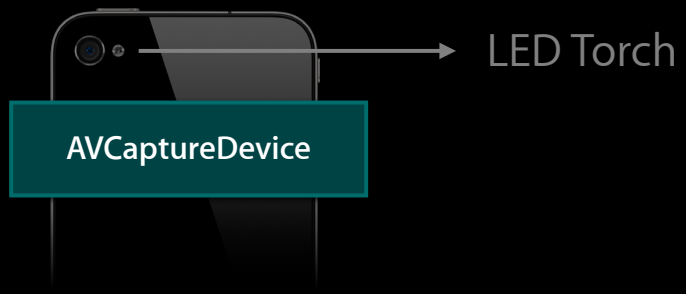
MorseMe

Valentin Bonnet
iOS Intern

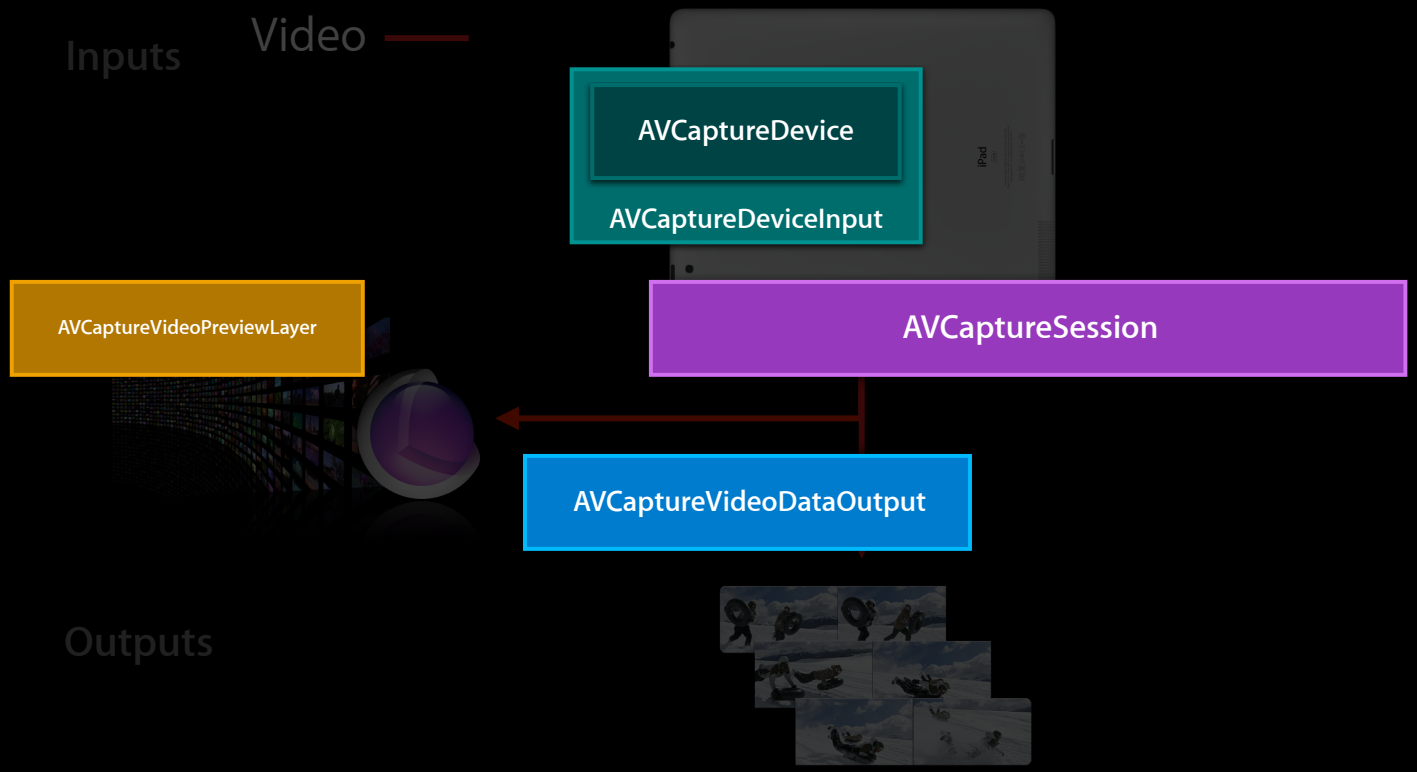
MorseMe (Sender)



MorseMe (Sender)



MorseMe


Inputs
Video
AVCaptureDevice
AVCaptureDeviceInput
AVCaptureVideoPreviewLayer
AVCaptureSession
AVCaptureVideoDataOutput
Outputs

MorseMe

Torch support

- Torch modes

`AVCaptureTorchModeOff`

`AVCaptureTorchModeOn`

`AVCaptureTorchModeAuto`

- Use `-hasTorch` to determine if the AVCaptureDevice has one
- Call `-lockForConfiguration:` before attempting to set the torch mode

Flashlight apps no longer need to run an AVCaptureSession to use the torch

```
if ( YES == [device lockForConfiguration:&error] ) {
    [device setTorchMode:AVCaptureTorchModeOn];
    [device unlockForConfiguration];
}
```

MorseMe

Torch API enhancements in iOS 5

- Torch availability accessors
  - Torch may become unavailable as phone gets too hot
  - Key-value observe `isTorchAvailable` to know when the unit gets too hot, and when it cools down enough
  - Key-value observe the `torchLevel` property to know when the torch illumination level is decreasing due to thermal duress

Capture Use Cases We Will Cover

- Process video frames from the camera and render with OpenGL
- Process live video and audio, and write them to a QuickTime movie
- Scan video frames for patterns using the flash and VideoDataOutput
- Process captured still images with CoreImage

Demo

‘StacheCam

Using AVCaptureStillImageOutput with CoreImage in iOS 5

‘StacheCam

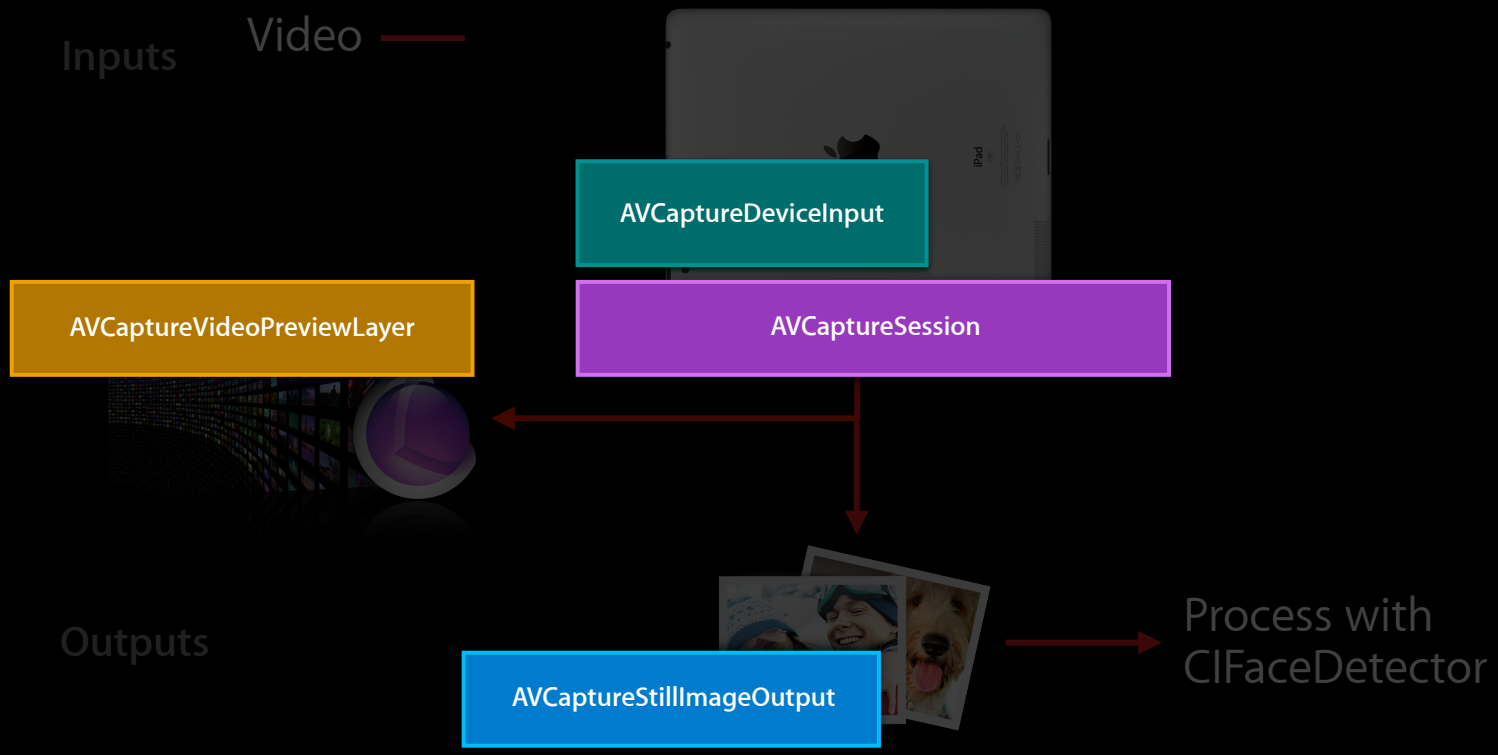
Inputs

Video

Outputs

Process with CIFaceDetector

‘StacheCam


Inputs
Video
AVCaptureDeviceInput
AVCaptureVideoPreviewLayer
AVCaptureSession
Outputs
AVCaptureStillImageOutput
Process with
CIFaceDetector

‘StacheCam

iOS 5 enhancements for still images

- AVCaptureStillImageOutput new property `isCapturingStillImage`
  - Key-value observe to know exactly when a still image is being taken
  - Handy for driving a shutter animation
- AVCaptureDevice new property `subjectAreaChangeMonitoringEnabled`
  - Allows you to lock focus or exposure, but still receive a notification when the scene has changed significantly, so you can re-focus / expose

‘StacheCam

iOS 5 enhancements for still images

- CoreImage CIFilters make their iOS debut
  - Red-eye reduction
  - “Auto-enhance”
- CIDectector finds faces and features
- CIImage interfaces with CVPixelBuffers
- Specify ‘BGRA’ output for CIImage compatibility

‘StacheCam

iOS 5 enhancements for still images


```
CVPixelBufferRef pb = CMSampleBufferGetImageBuffer(imageDataSampleBuffer);
CIImage *ciImage = [[CIImage alloc] pb options:nil];

NSDictionary *detectorOptions = [NSDictionary dictionaryWithObject:
             CIDetectorAccuracyLow forKey:CIDetectorAccuracy];
CIDetector *faceDetector = [CIDetector detectorOfType:CIDetectorTypeFace
                           context:nil options:detectorOptions];

NSArray *features = [faceDetector featuresInImage:ciImage
                                          options:imageOptions];
for (CIFaceFeature *face in features) {
    // iterate through the faces, find bounds,
    // left eye, right eye, mouth position
}
```

For More on CoreImage in iOS 5

Using Core Image on iOS & Mac OS X	Mission Thursday 2:00PM

Summary

iOS 5 AV foundation capture

- Gives you more CPU cycles
- Bridges CoreVideo and OpenGL
- Exposes more resolutions
- Adds more flexibility to still image capture
- Enables you to deliver even cooler apps

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
Introducing AV Foundation Capture For Lion	Pacific Heights Wednesday 3:15PM

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