# Direct Access to Video Encoding and Decoding

Session 513

David Eldred
Tech Lead, Frameworks with “Video” in their names

© 2014 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

# Introduction

Hardware encode and decode can improve user experience

# Introduction

Hardware encode and decode can improve user experience

- Better performance

# Introduction

Hardware encode and decode can improve user experience

- Better performance
- Increased efficiency

# Introduction

Hardware encode and decode can improve user experience

- Better performance
- Increased efficiency
- Extend battery life

# What You Will Learn

# What You Will Learn

Case Studies

# What You Will Learn

Case Studies

1. Displaying an H.264 stream in a layer in your application

# What You Will Learn

Case Studies

1. Displaying an H.264 stream in a layer in your application
2. Decoding an H.264 stream and accessing the decoded buffers

# What You Will Learn

Case Studies

1. Displaying an H.264 stream in a layer in your application
2. Decoding an H.264 stream and accessing the decoded buffers
3. Compressing a sequence of images into a movie file

# What You Will Learn

Case Studies

1. Displaying an H.264 stream in a layer in your application
2. Decoding an H.264 stream and accessing the decoded buffers
3. Compressing a sequence of images into a movie file
4. Compressing a sequence of images into an H.264 stream for the network

# What You Will Learn

Case Studies

1. Displaying an H.264 stream in a layer in your application
2. Decoding an H.264 stream and accessing the decoded buffers
3. Compressing a sequence of images into a movie file
4. Compressing a sequence of images into an H.264 stream for the network

Using multi-pass encoding in AVFoundation and Video Toolbox

# Media Interfaces Overview

## With a focus on video

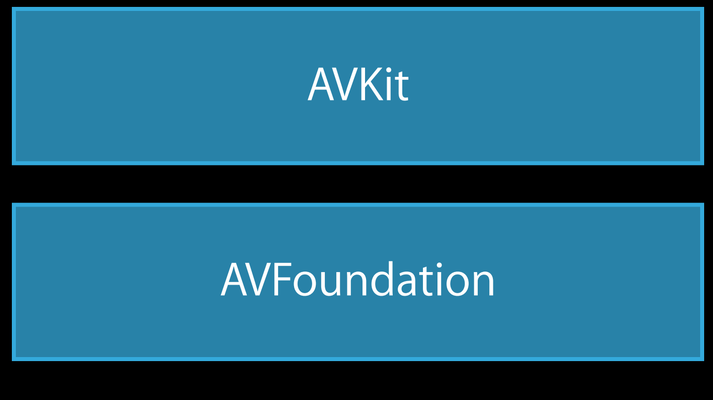
# Media Interfaces Overview

## With a focus on video

AVKit

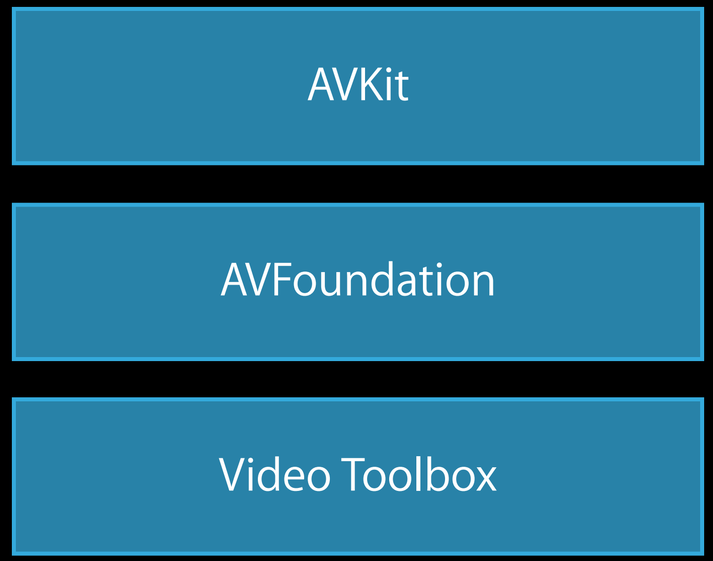
# Media Interfaces Overview

## With a focus on video



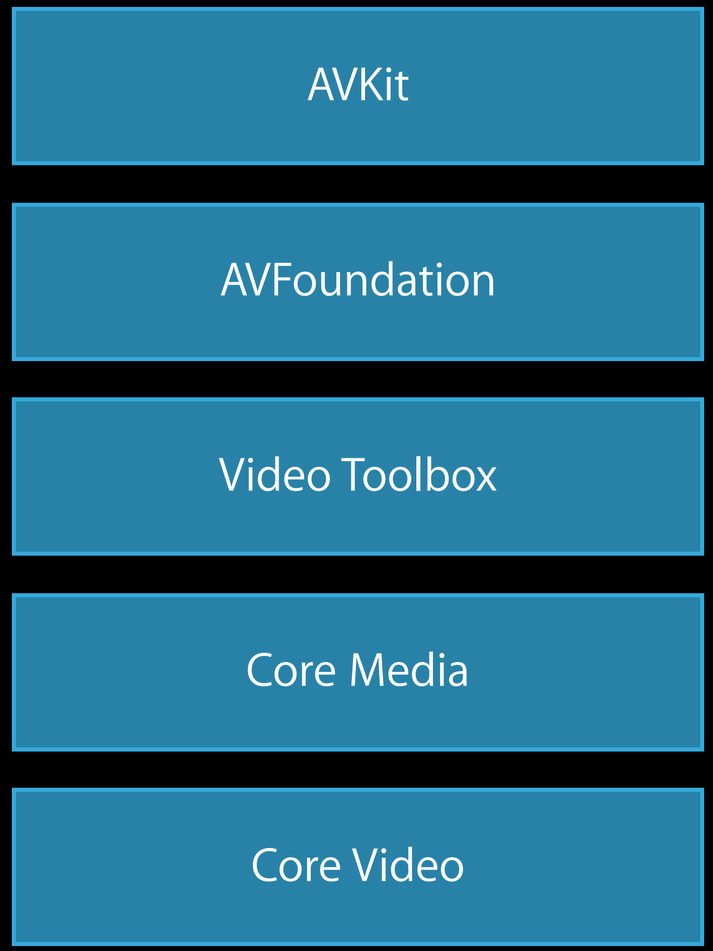
# Media Interfaces Overview

## With a focus on video

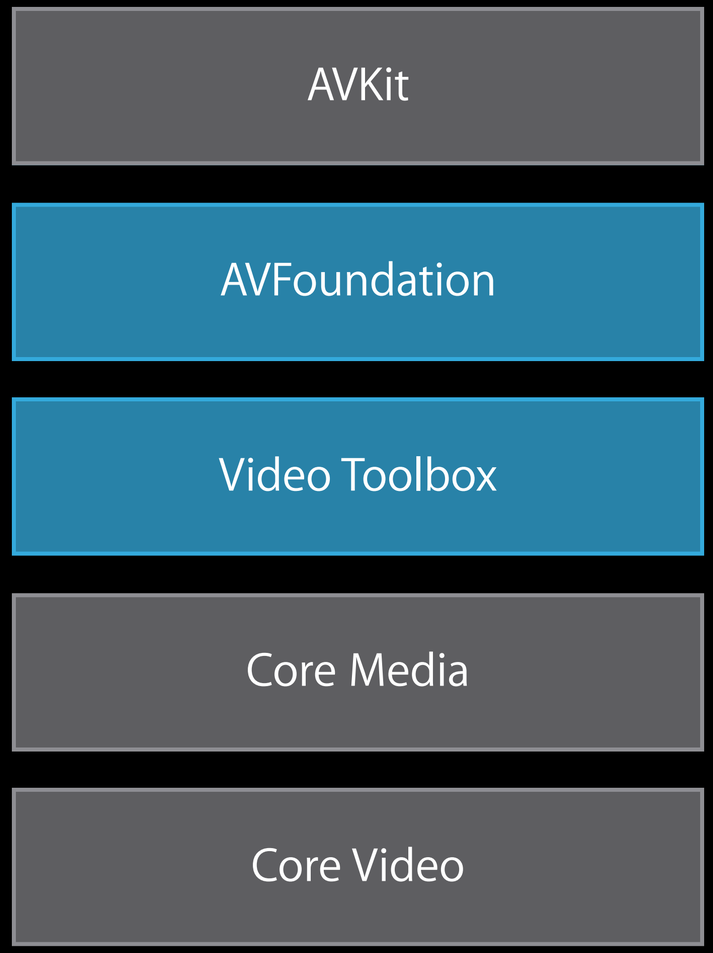


# Media Interfaces Overview

## With a focus on video



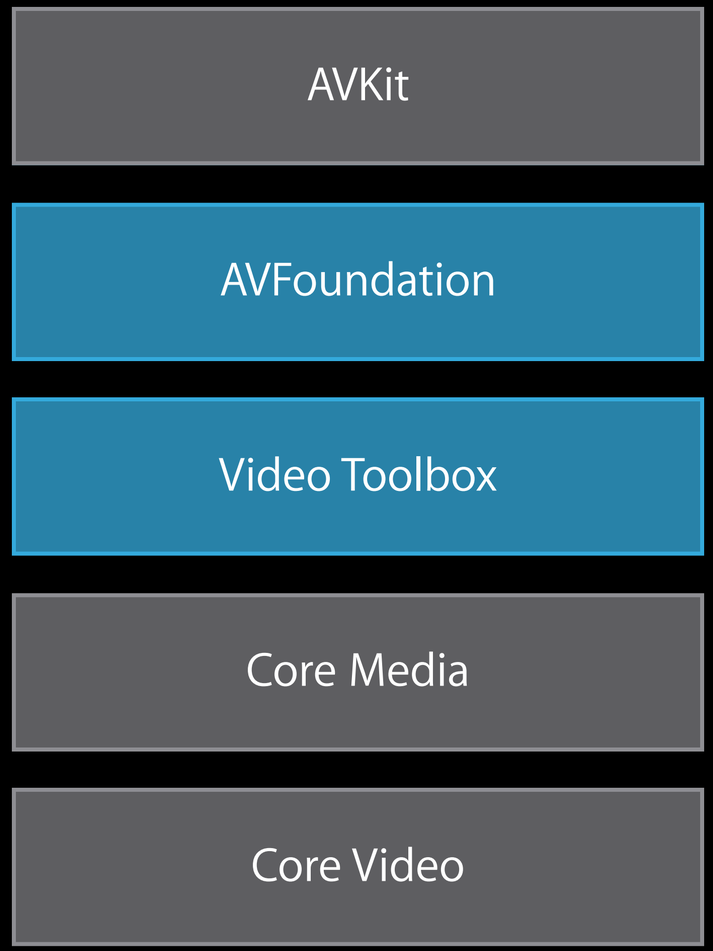
# Media Interface Focus



# Media Interface Focus

AVFoundation

- Decompress direct to display
- Compress directly to file



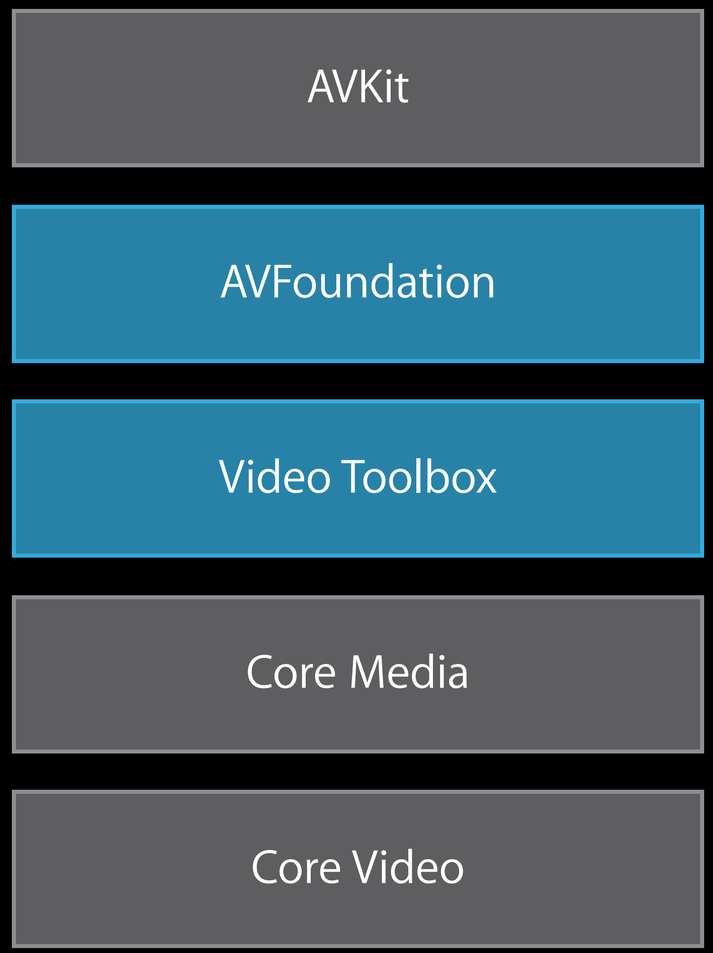
# Media Interface Focus

AVFoundation

- Decompress direct to display
- Compress directly to file

Video Toolbox

- Decompress to CVPixelBuffer
- Compress to CMSampleBuffer



# Media Interface Focus

AVKit

AVFoundation

Video Toolbox

# Hardware Codec Usage

AVKit

AVFoundation

Video Toolbox

# Hardware Codec Usage

| | iOS | OS X |
|---|---|---|
| AVKit | Always | When Available |
| AVFoundation | Always | When Available |
| Video Toolbox | Always | When Available and Requested |

# The Cast of Characters

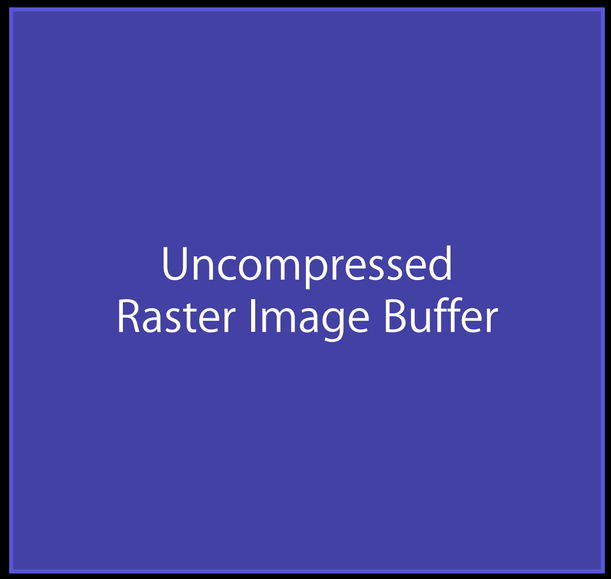
CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

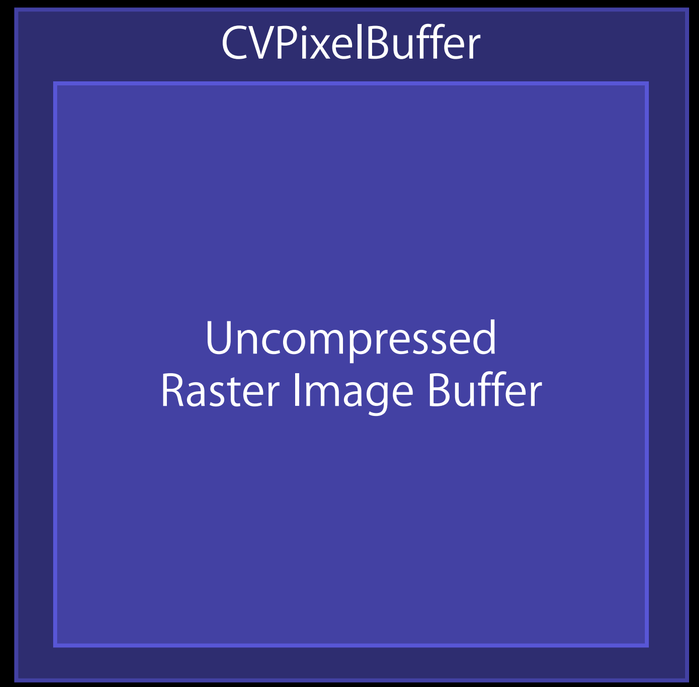
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



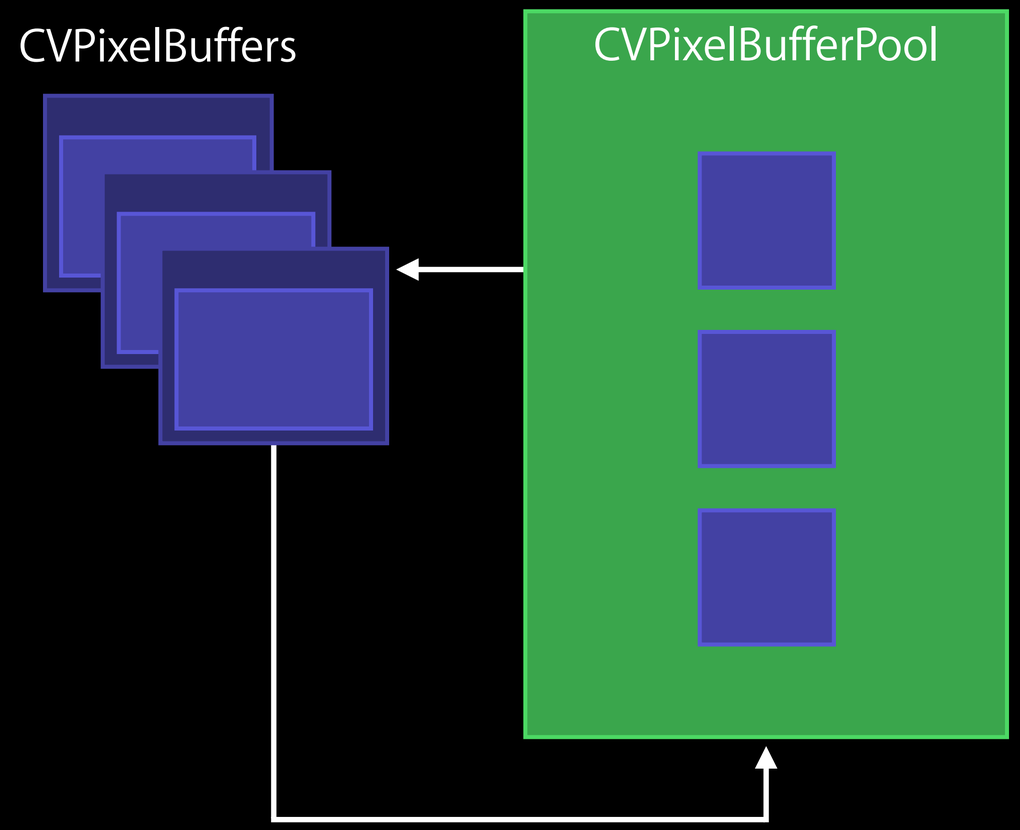
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

`CFDictionary` of requirements;
May Include:

- Width/height
- Pixel format type (e.g., 32BGRA, YCbCr420)
- Compatibility (e.g., OpenGL ES, Core Animation)

# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

64-bit Time Value (Numerator)
32-bit Time Scale (Denominator)

# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

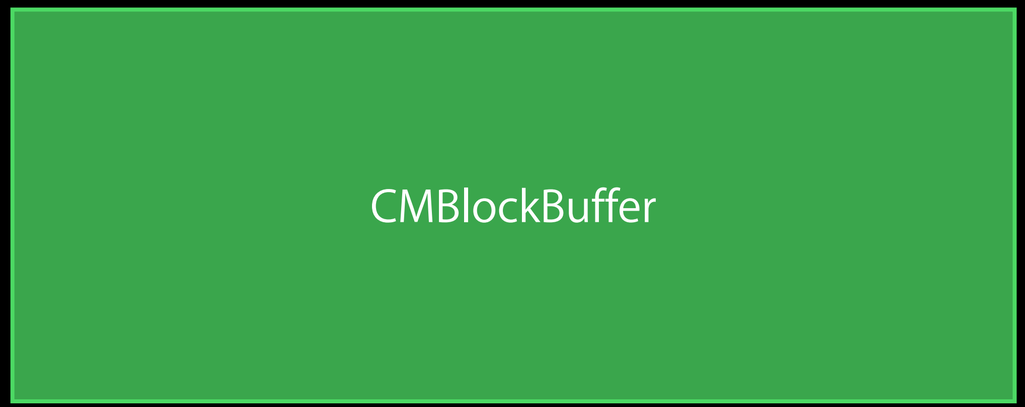
Width/Height

Format Type—(kCMPixelFormat_32BGRA, kCMVideoCodecType_H264,…)

Extensions—(Pixel Aspect Ratio, Color Space,…)

# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

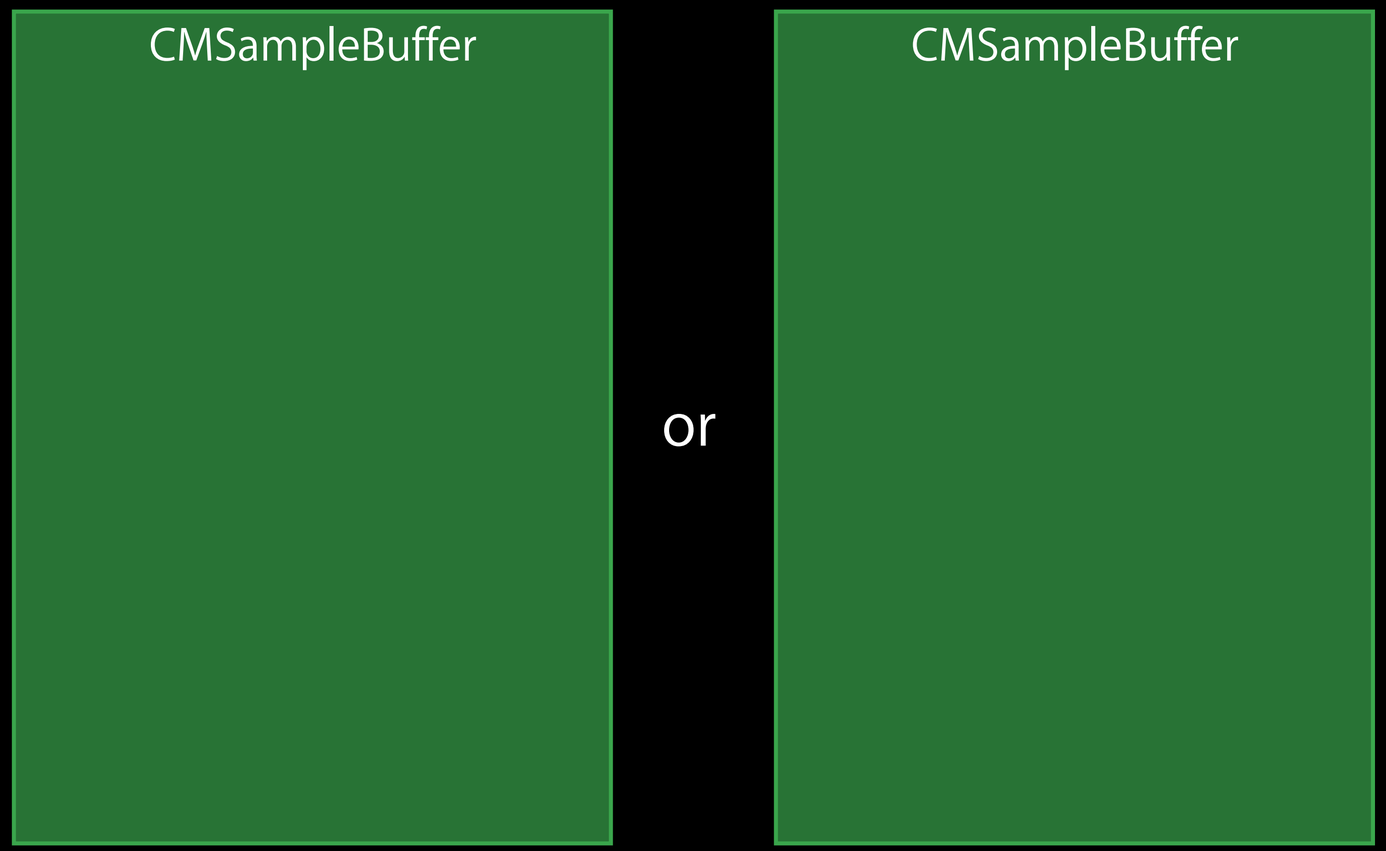


# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

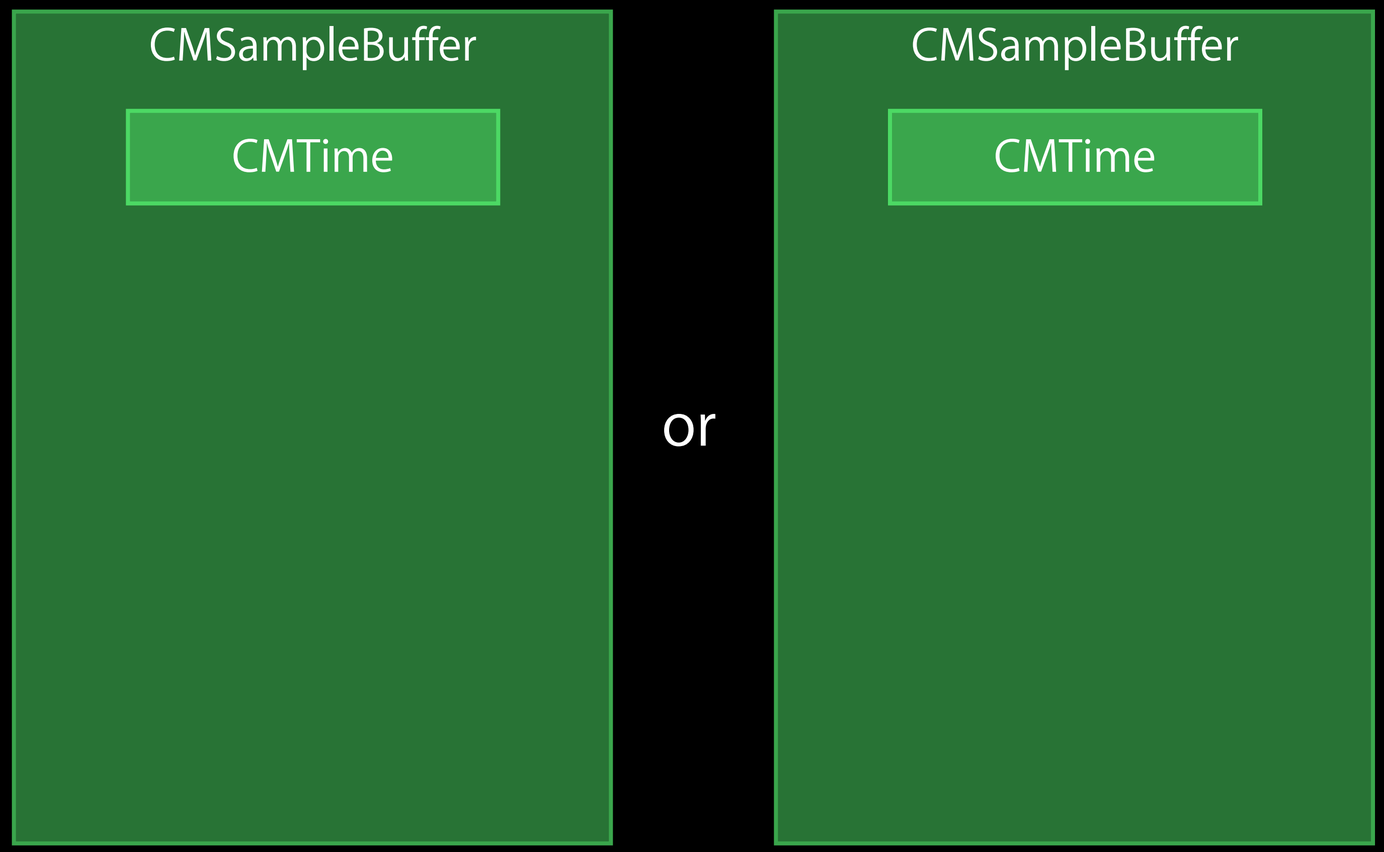
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



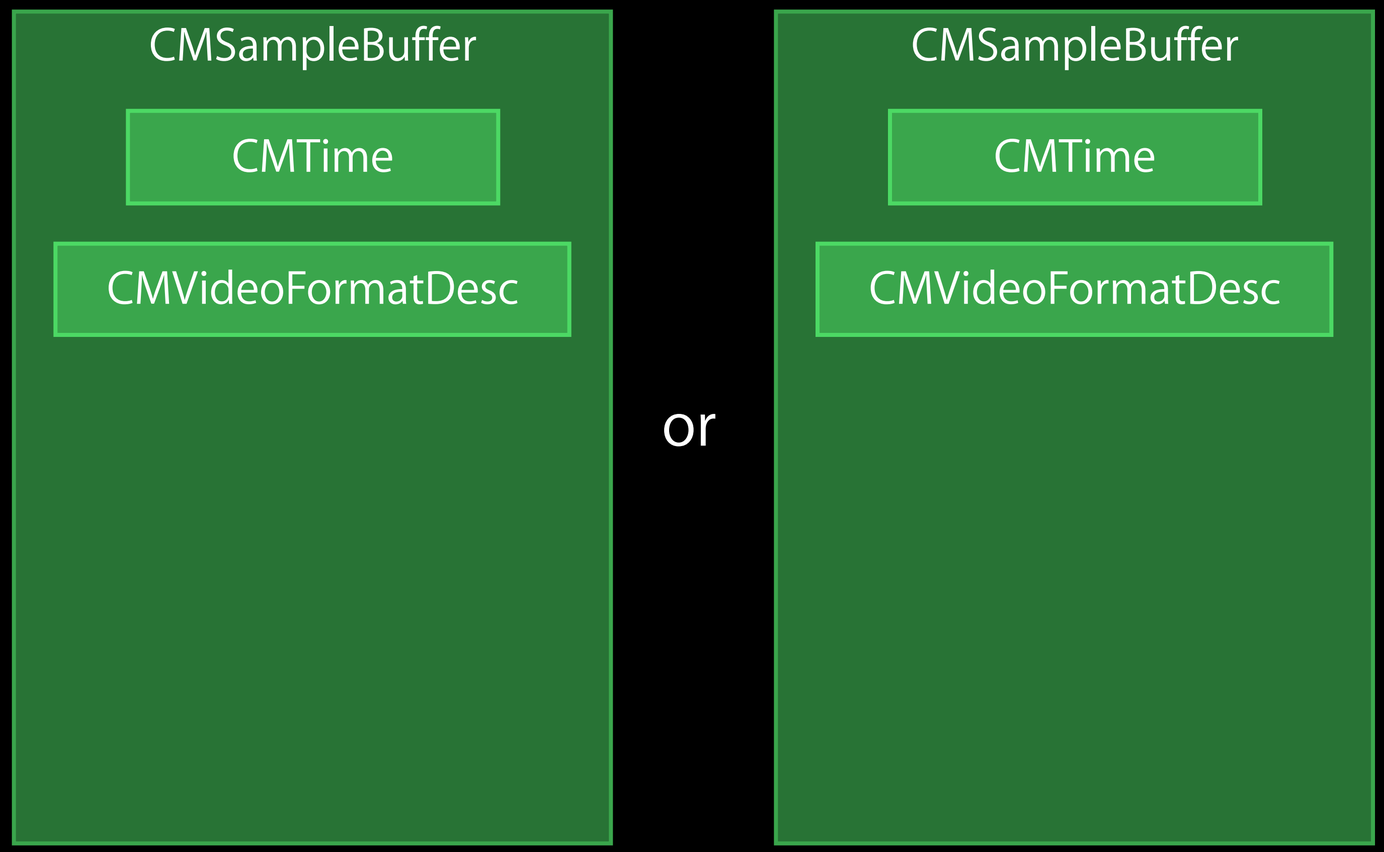
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



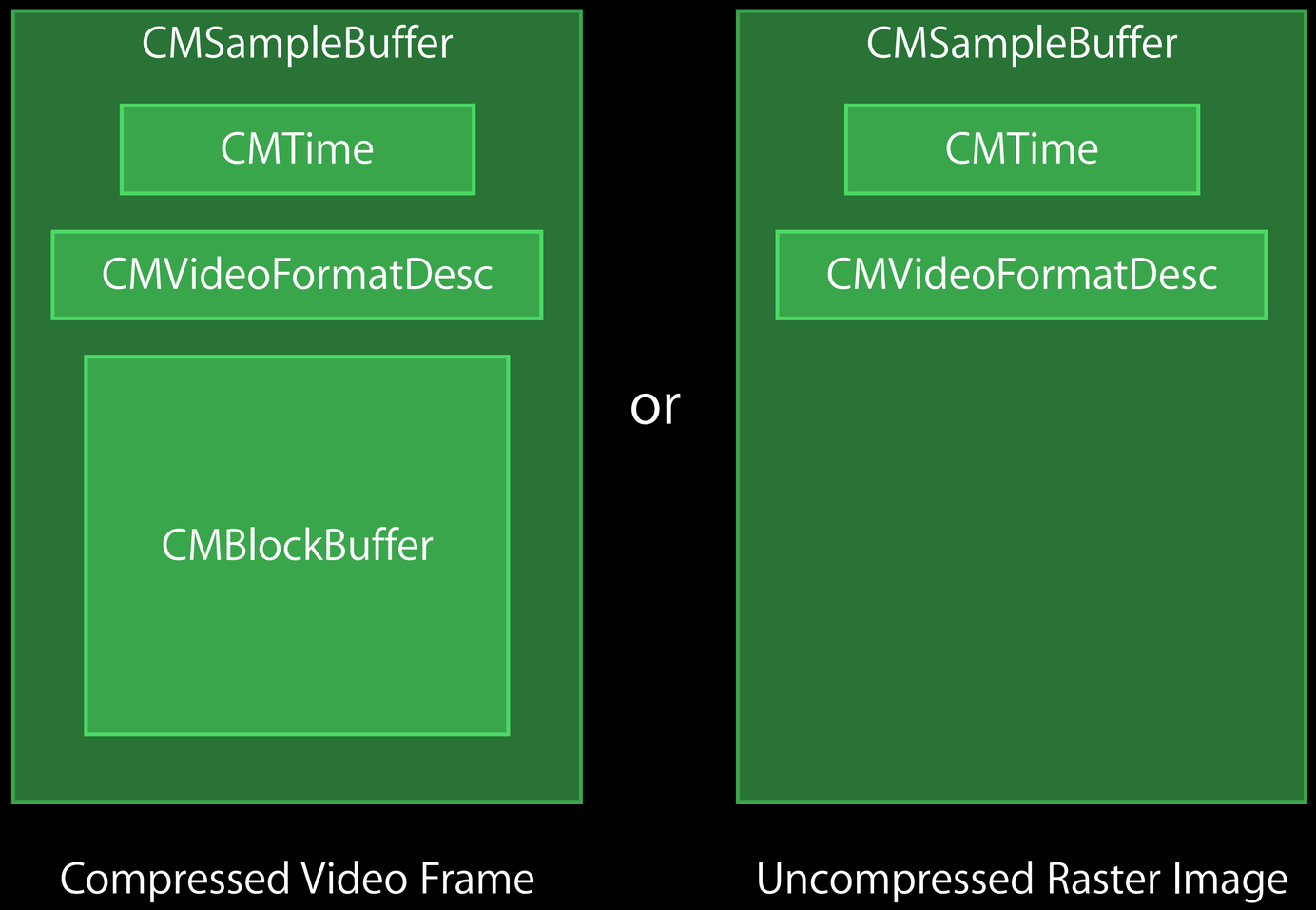
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



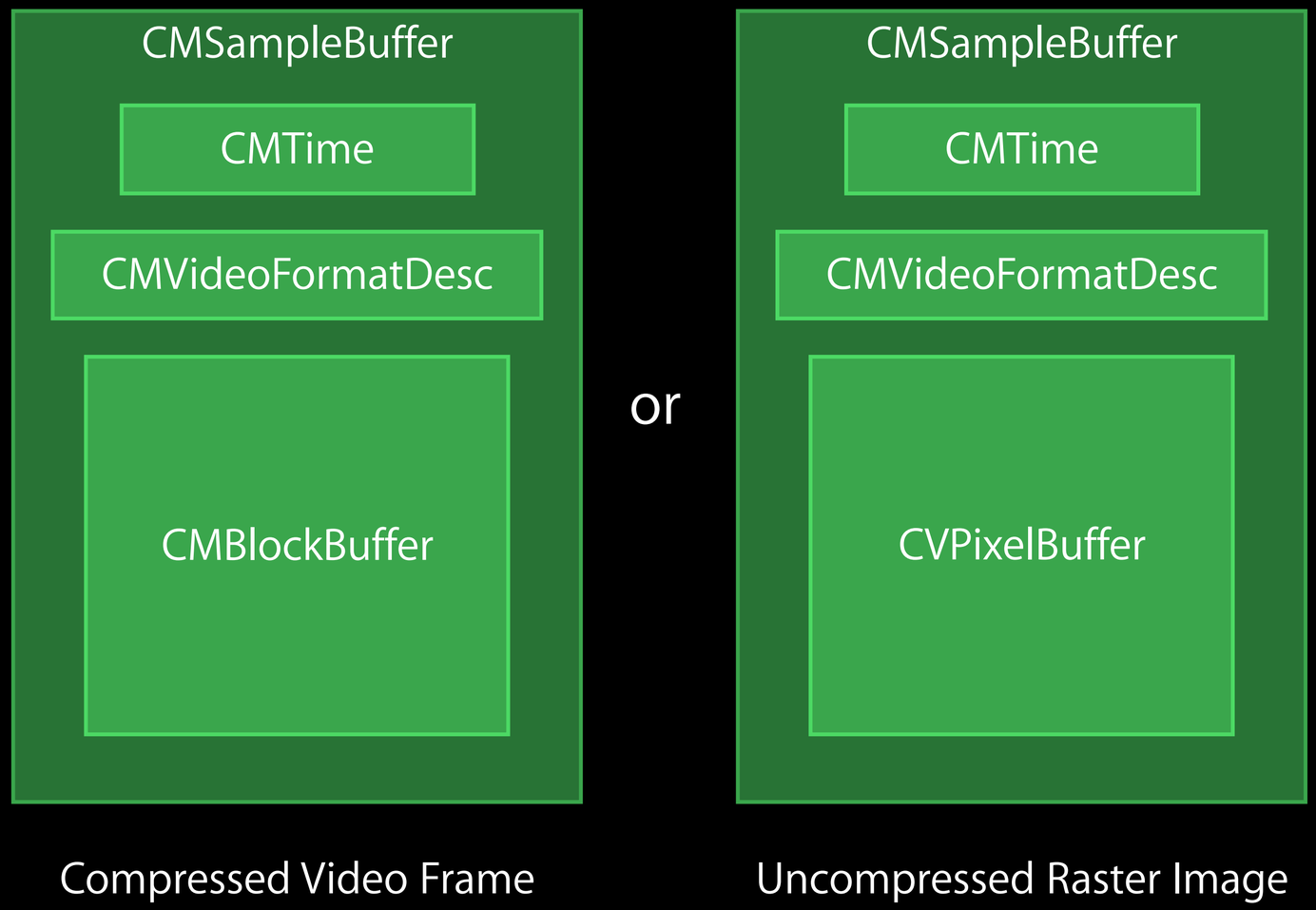
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase



# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

# The Cast of Characters

Wraps a source of time

A clock's time always increases

`CMClockGetHostTimeClock() wraps mach_absolute_time()`

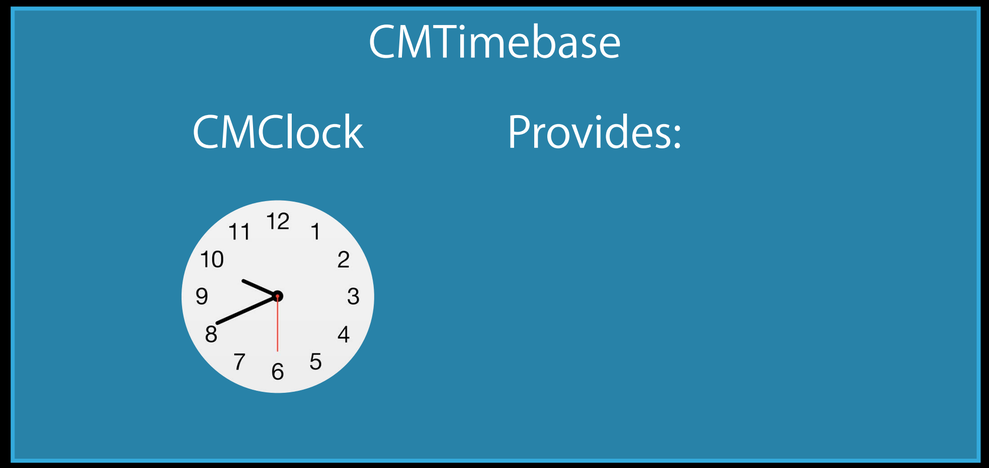
# The Cast of Characters

CVPixelBuffer
CVPixelBufferPool
pixelBufferAttributes
CMTime
CMVideoFormatDescription
CMBlockBuffer
CMSampleBuffer
CMClock
CMTimebase

CMClock

# The Cast of Characters

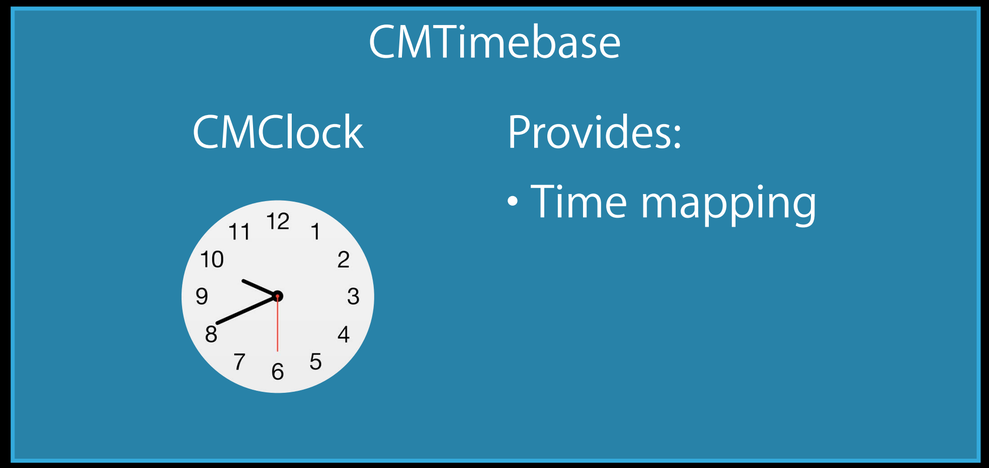
A controlled view onto a CMClock



# The Cast of Characters

A controlled view onto a CMClock

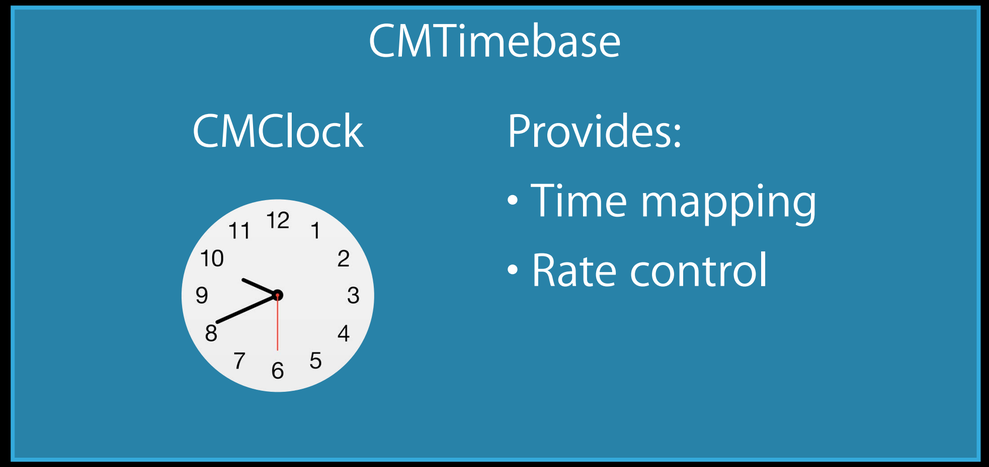
```
CMTimebaseSetTime(timebase, kCMTimeZero);
```



# The Cast of Characters

A controlled view onto a CMClock

```
CMTimebaseSetTime(timebase, kCMTimeZero);
CMTimebaseSetRate(timebase, 1.0);
```



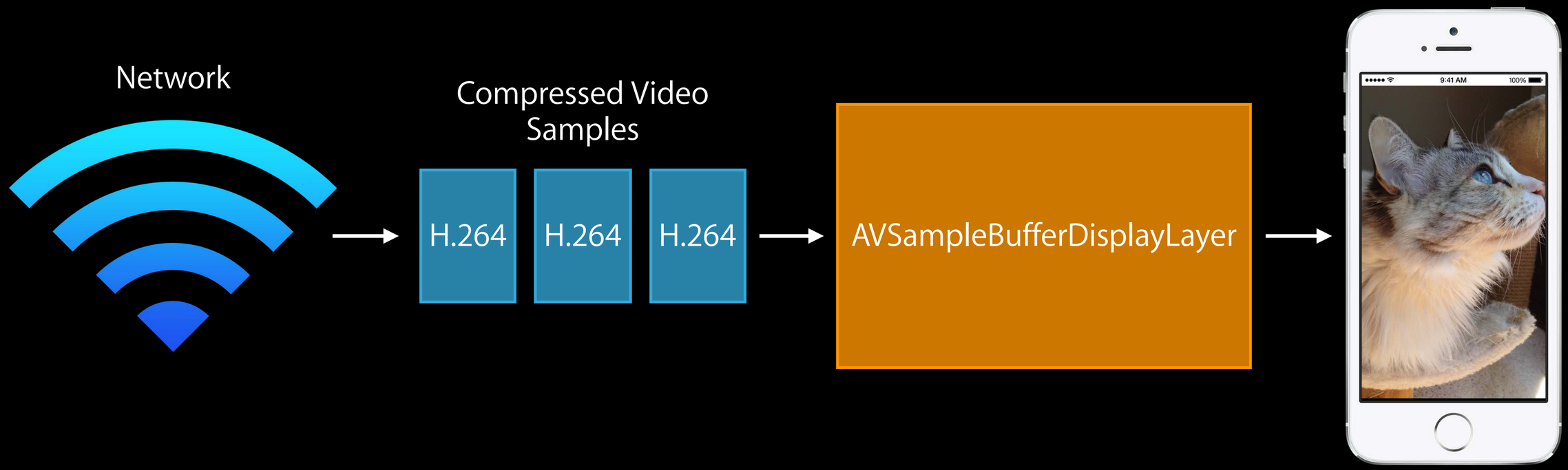
# Case One

## Displaying video from a network stream

# Case One Overview

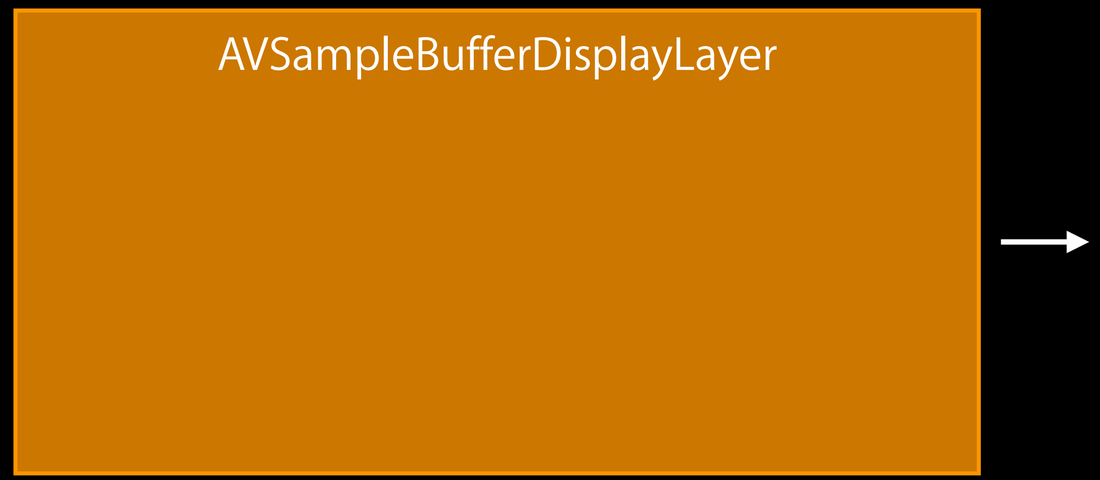
# Case One Overview

# Case One Overview



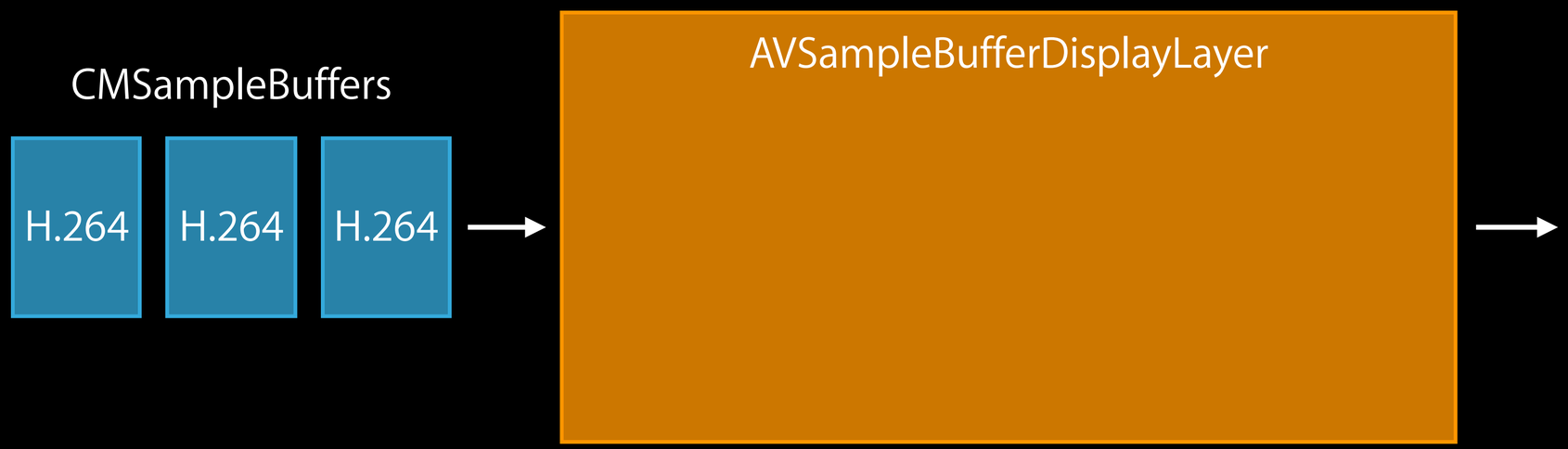
# AVSampleBufferDisplayLayer

## A closer look



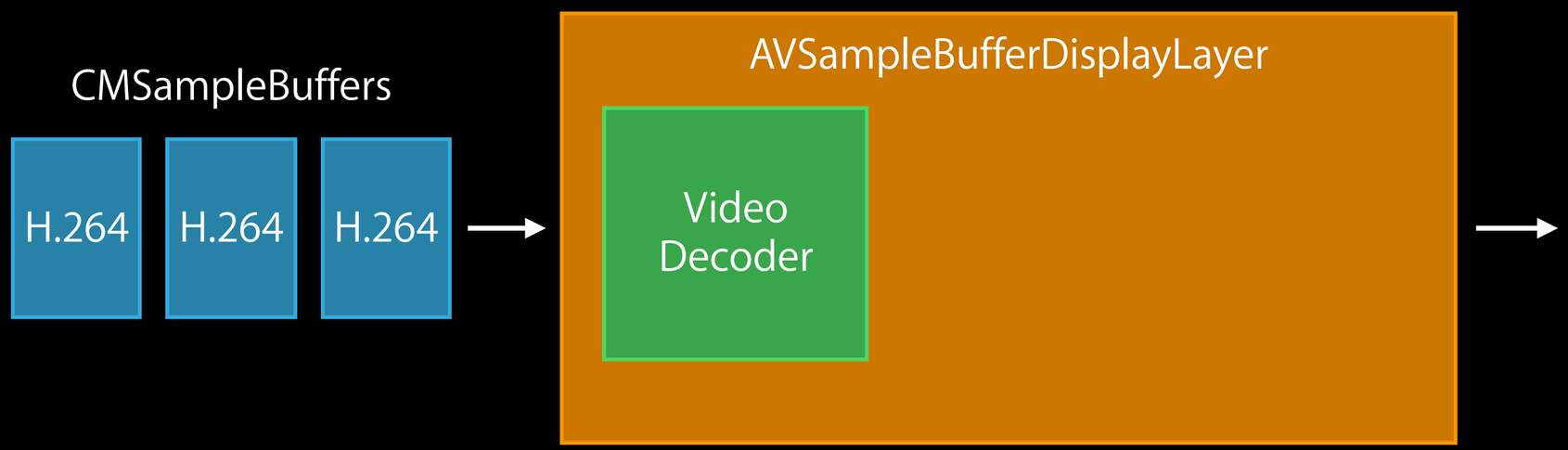
# AVSampleBufferDisplayLayer

## A closer look



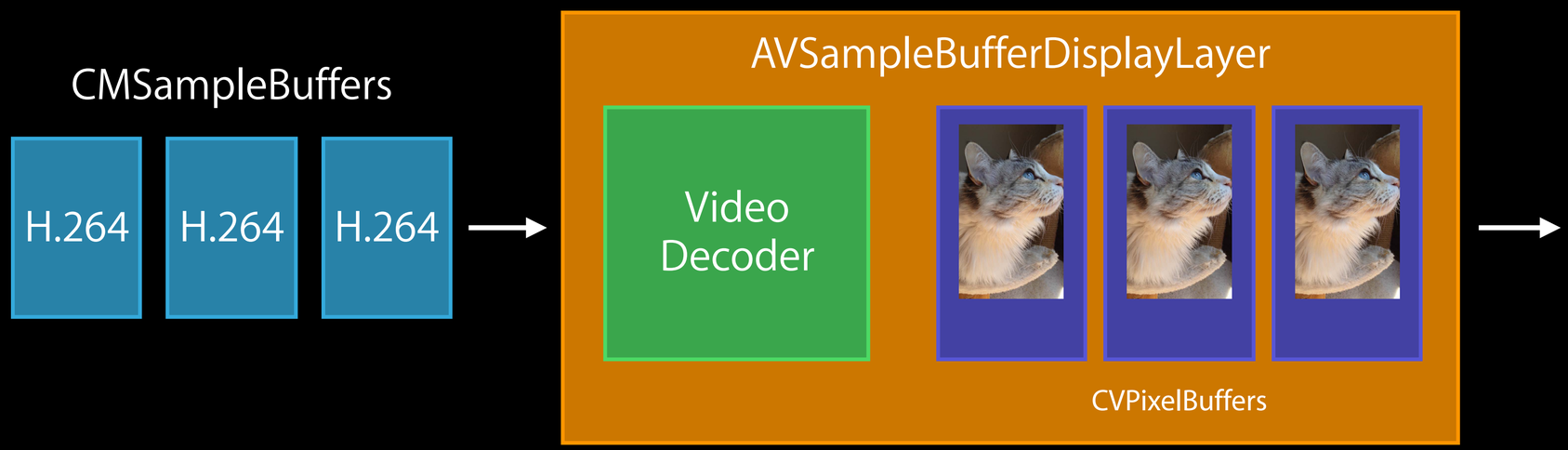
# AVSampleBufferDisplayLayer

## A closer look



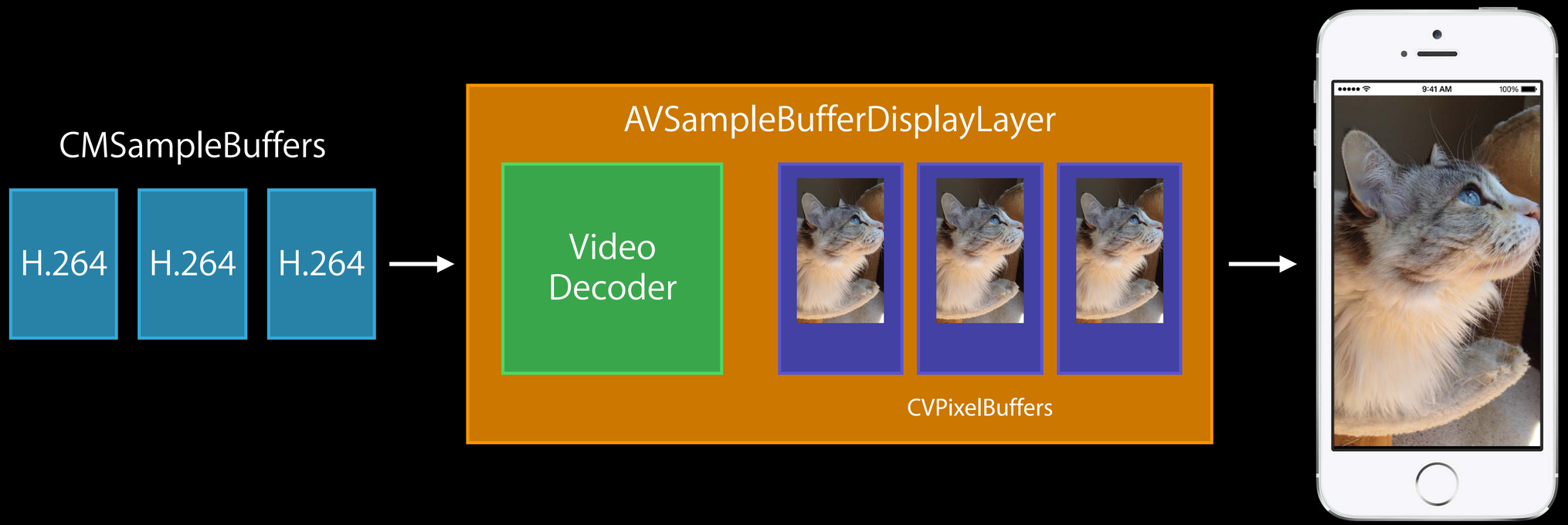
# AVSampleBufferDisplayLayer

## A closer look

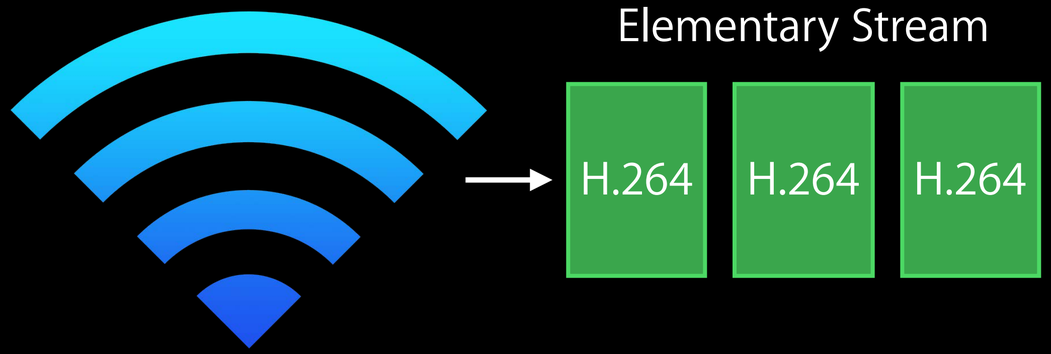


# AVSampleBufferDisplayLayer

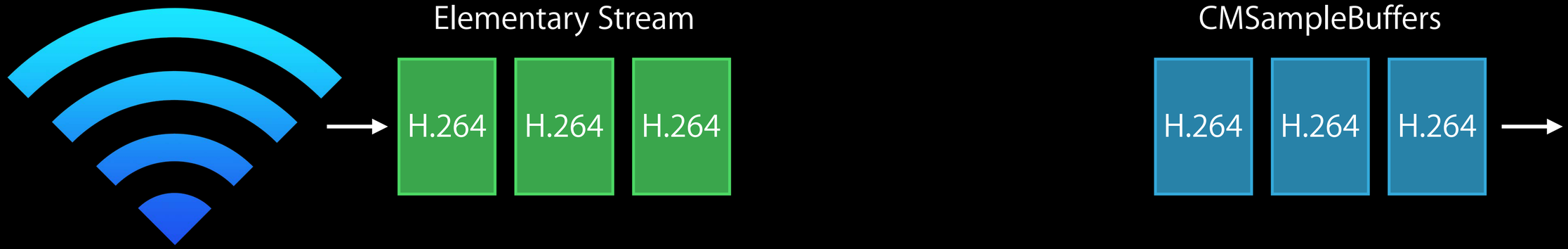
## A closer look



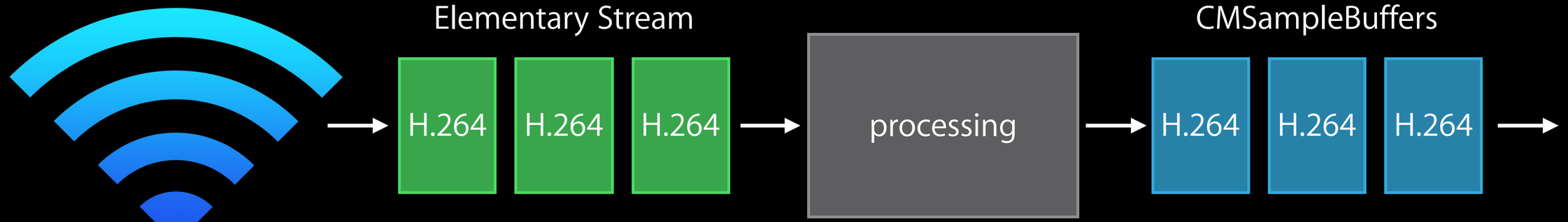
# AVSampleBufferDisplayLayer Input



# AVSampleBufferDisplayLayer Input



# AVSampleBufferDisplayLayer Input



# H.264 Syntax

Elementary Stream

MPEG-4

# H.264 Syntax

Elementary Stream

MPEG-4

# H.264 Syntax

## Network Abstraction Layer (NAL)

# H.264 Syntax

## Network Abstraction Layer (NAL)

H.264 stream consists of a sequence of NAL Units (NALUs)

# H.264 Syntax

## Network Abstraction Layer (NAL)

H.264 stream consists of a sequence of NAL Units (NALUs)


NAL Units may contain

# H.264 Syntax

## Network Abstraction Layer (NAL)

H.264 stream consists of a sequence of NAL Units (NALUs)

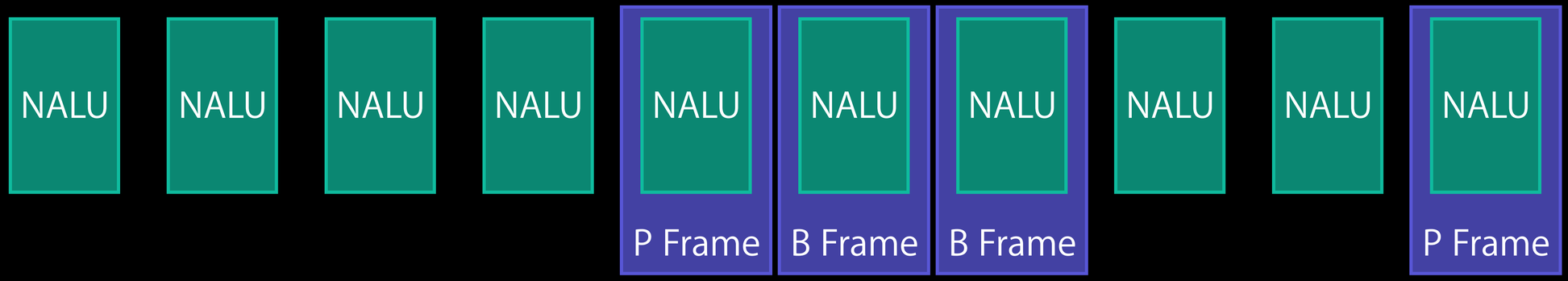

NAL Units may contain

- Video frame (or slice of video frame)

# H.264 Syntax

## Network Abstraction Layer (NAL)

H.264 stream consists of a sequence of NAL Units (NALUs)



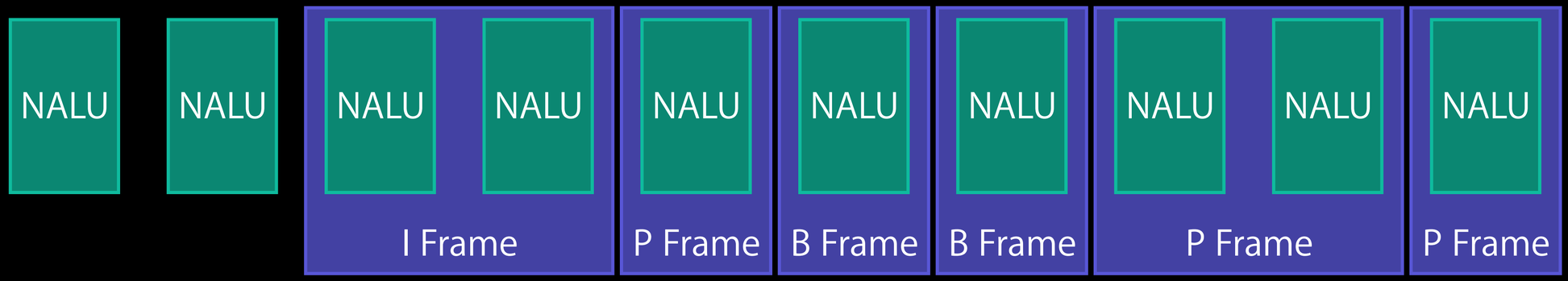

NAL Units may contain

- Video frame (or slice of video frame)

# H.264 Syntax

## Network Abstraction Layer (NAL)

H.264 stream consists of a sequence of NAL Units (NALUs)



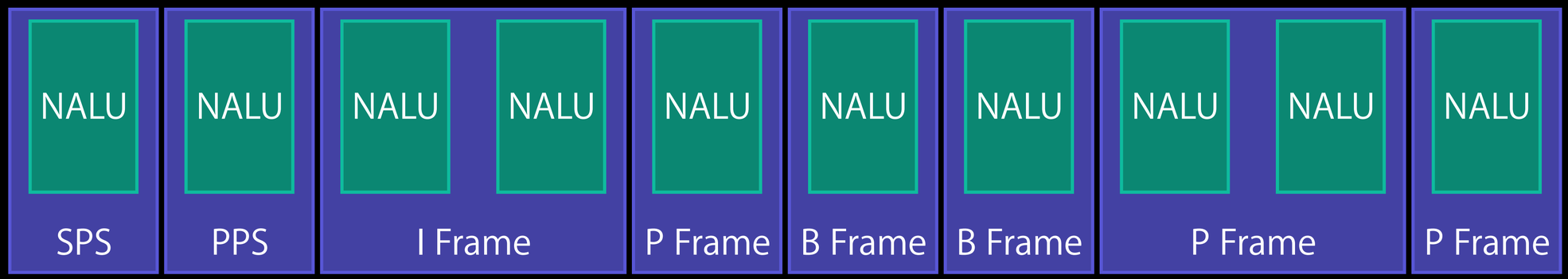

NAL Units may contain

- Video frame (or slice of video frame)

# H.264 Syntax

## Network Abstraction Layer (NAL)

H.264 stream consists of a sequence of NAL Units (NALUs)




NAL Units may contain

- Video frame (or slice of video frame)
- H.264 parameter sets
  - Sequence Parameter Set (SPS) and Picture Parameter Set (PPS)

# H.264 Syntax

## Parameter sets: SPS and PPS

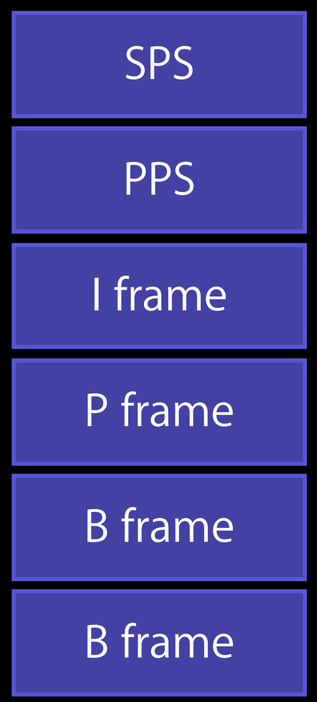
Elementary Stream

MPEG-4

# H.264 Syntax

## Parameter sets: SPS and PPS

Elementary Stream



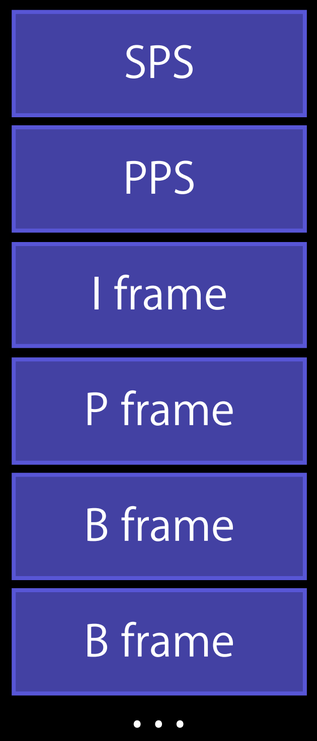

Parameter Sets in Stream

MPEG-4

# H.264 Syntax

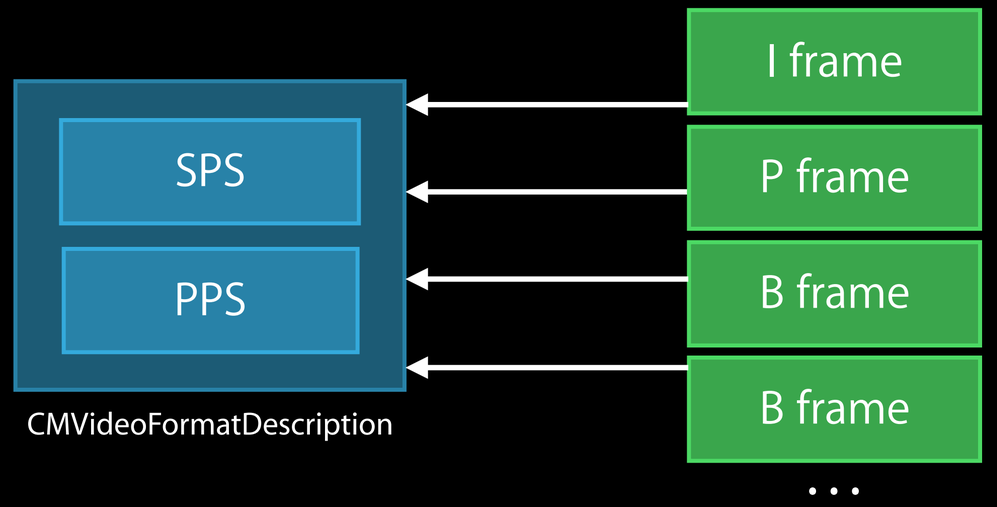
## Parameter sets: SPS and PPS

Elementary Stream




Parameter Sets in Stream

MPEG-4




Parameter Sets in Format Description

# H.264 Syntax

## Conversion

Elementary Stream

MPEG-4

# H.264 Syntax

## Conversion

Elementary Stream

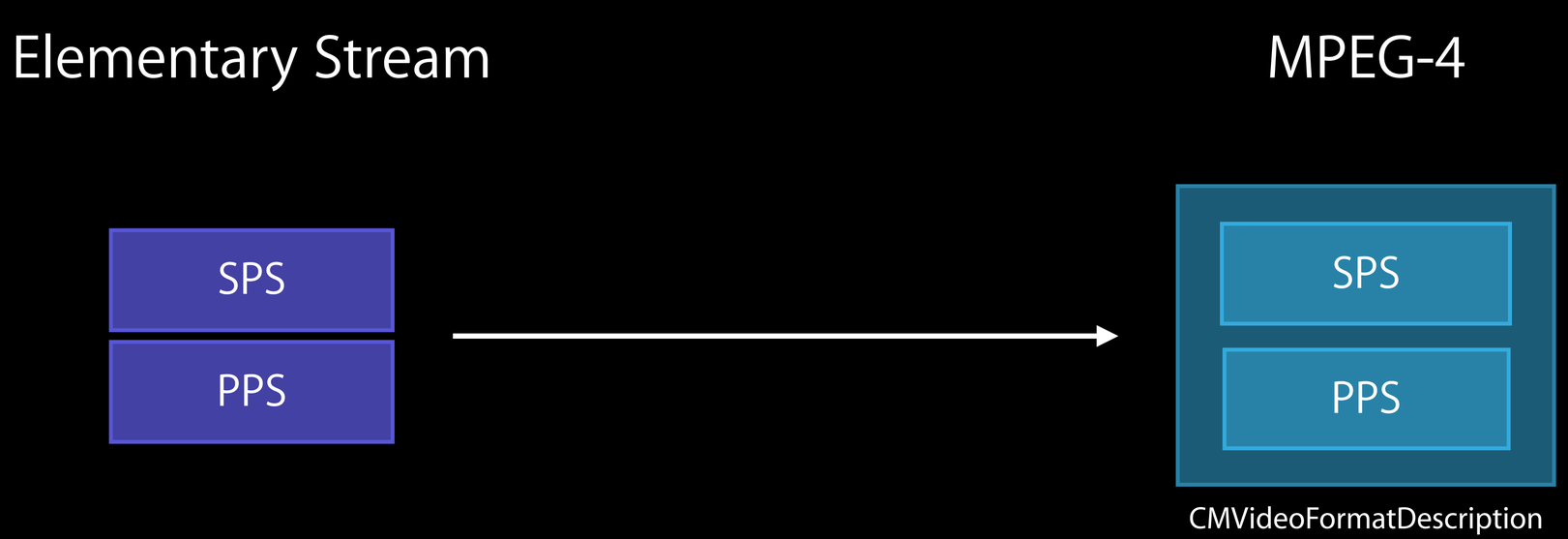
MPEG-4

SPS

PPS

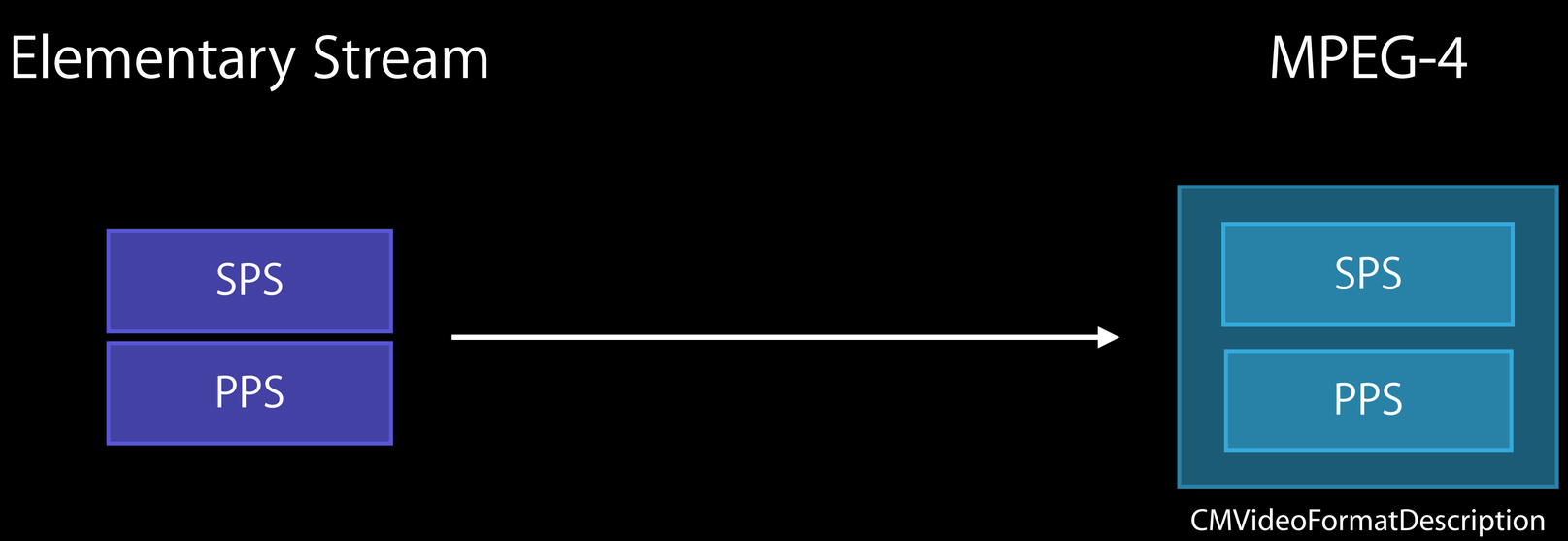
# H.264 Syntax

## Conversion



# H.264 Syntax

## Conversion




```
CMVideoFormatDescriptionCreateFromH264ParameterSets
```

# H.264 Syntax

## NAL Unit headers

Elementary Stream

MPEG-4

# H.264 Syntax

## NAL Unit headers

Elementary Stream

MPEG-4

```
00 00 01
```


3- or 4-Byte Header:
Start Code

# H.264 Syntax

## NAL Unit headers

### Elementary Stream

```
00 00 01
```


3- or 4-Byte Header:
Start Code

### MPEG-4

```
00 00 80 00
```


4-Byte Header:
Length

# Building a CMSampleBuffer

## NAL Unit conversion

00 00 01

# Building a CMSampleBuffer

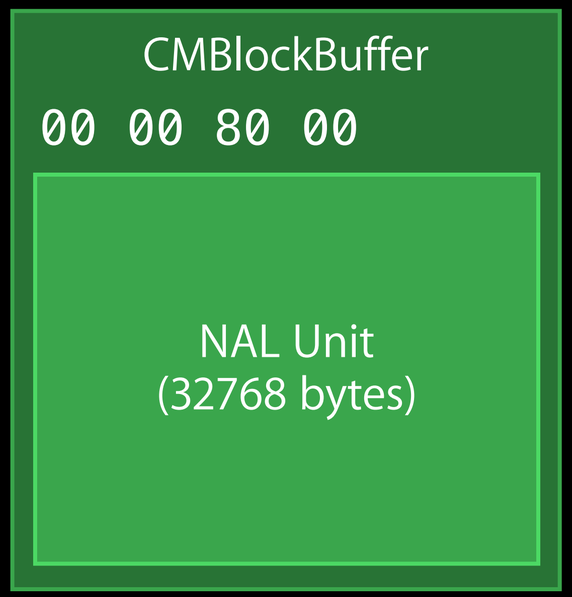
## NAL Unit conversion

00 00 80 00

NAL Unit
(32768 bytes)

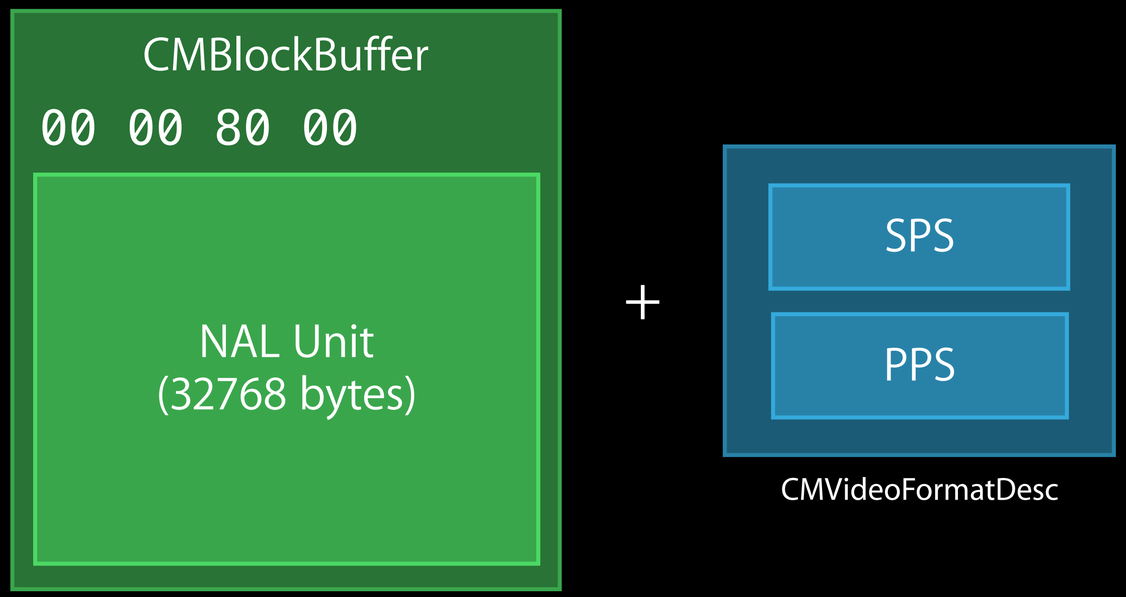
# Building a CMSampleBuffer

## NAL Unit conversion



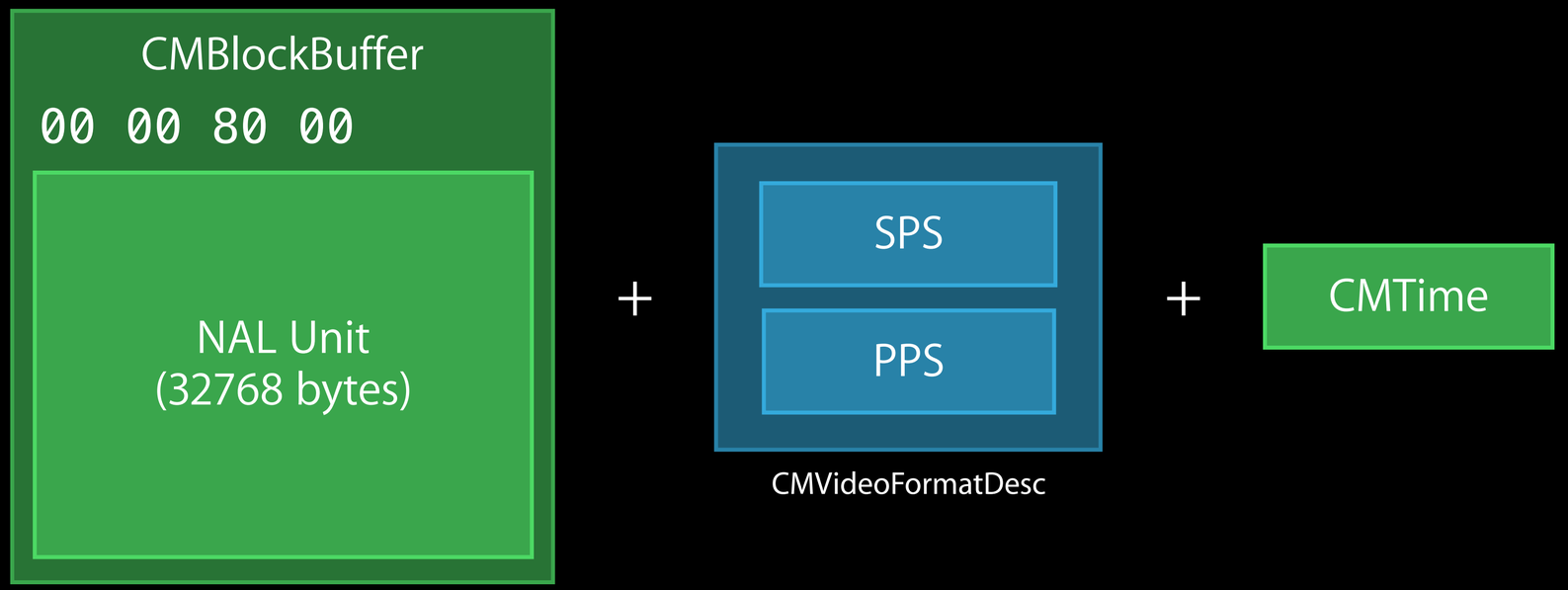
# Building a CMSampleBuffer

## NAL Unit conversion



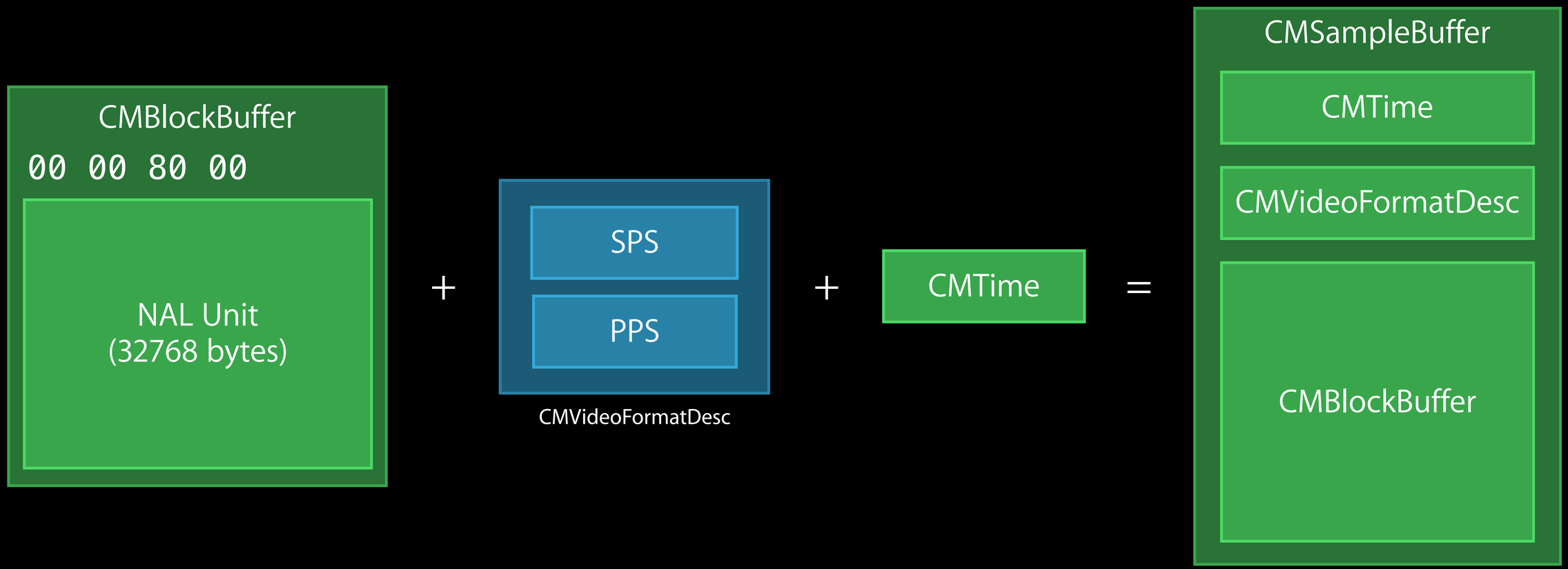
# Building a CMSampleBuffer

## NAL Unit conversion



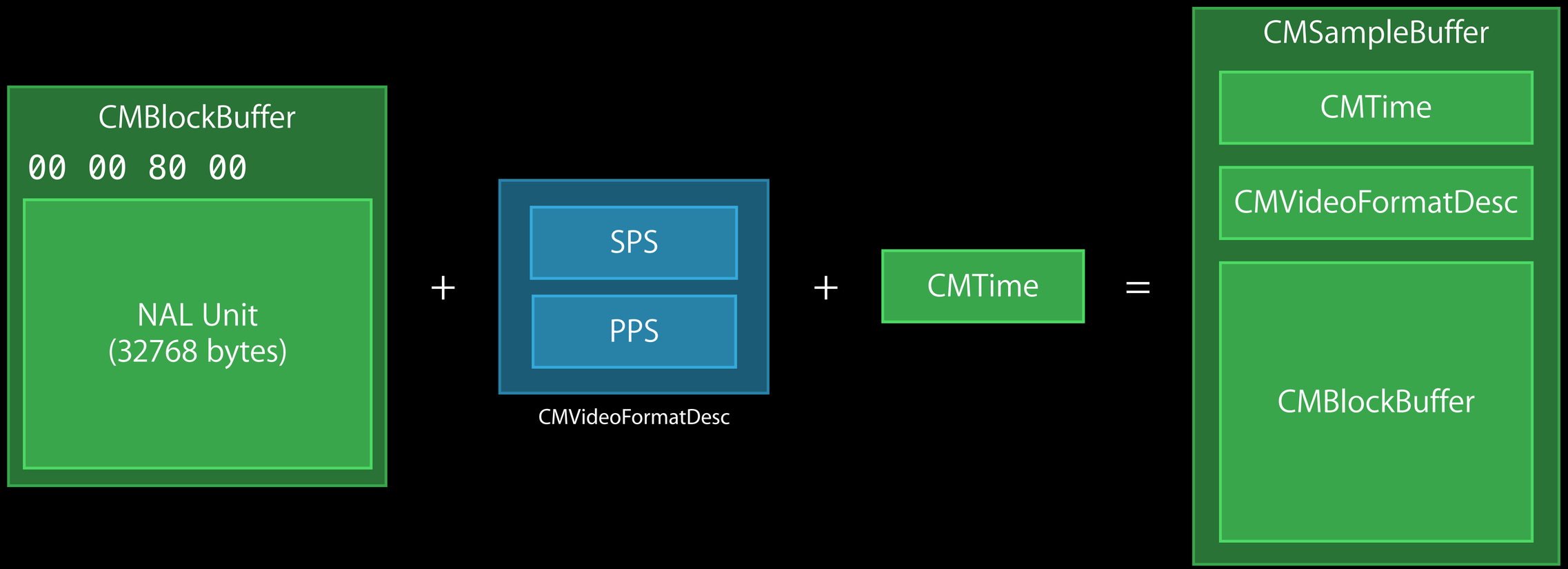
# Building a CMSampleBuffer

## NAL Unit conversion



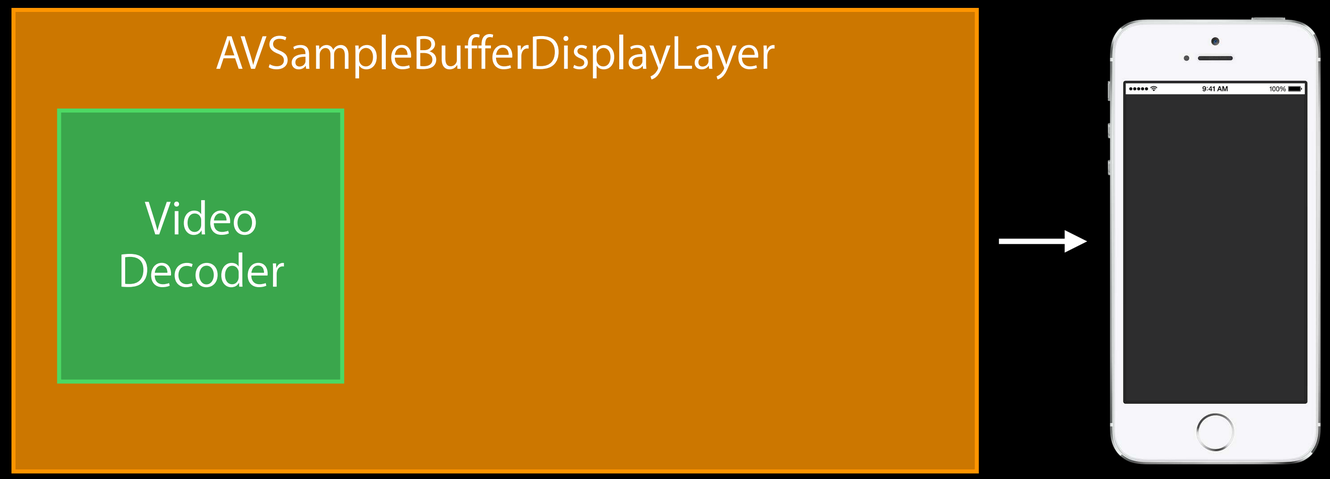
# Building a CMSampleBuffer

## NAL Unit conversion

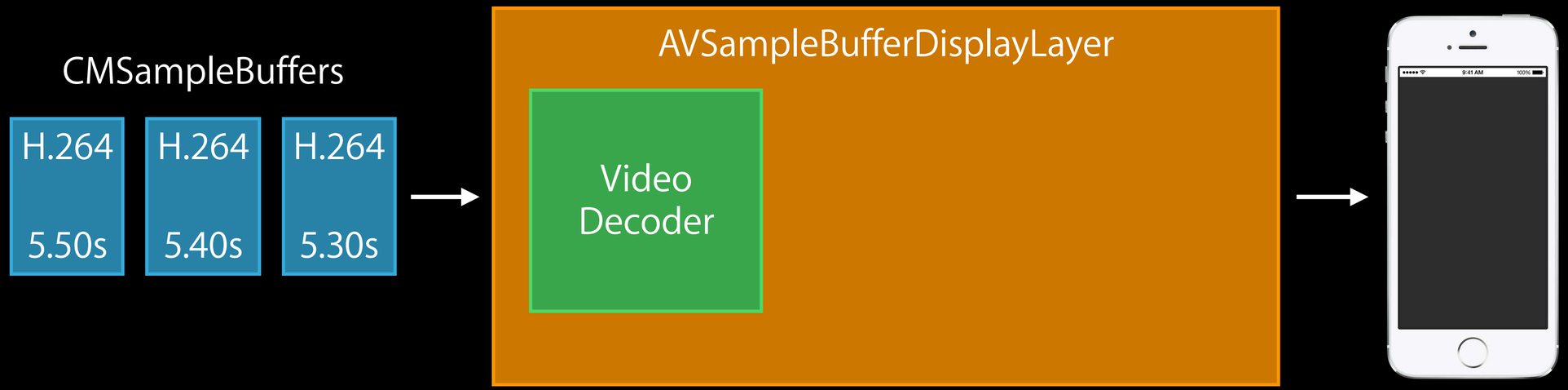



`CMSampleBufferCreate`

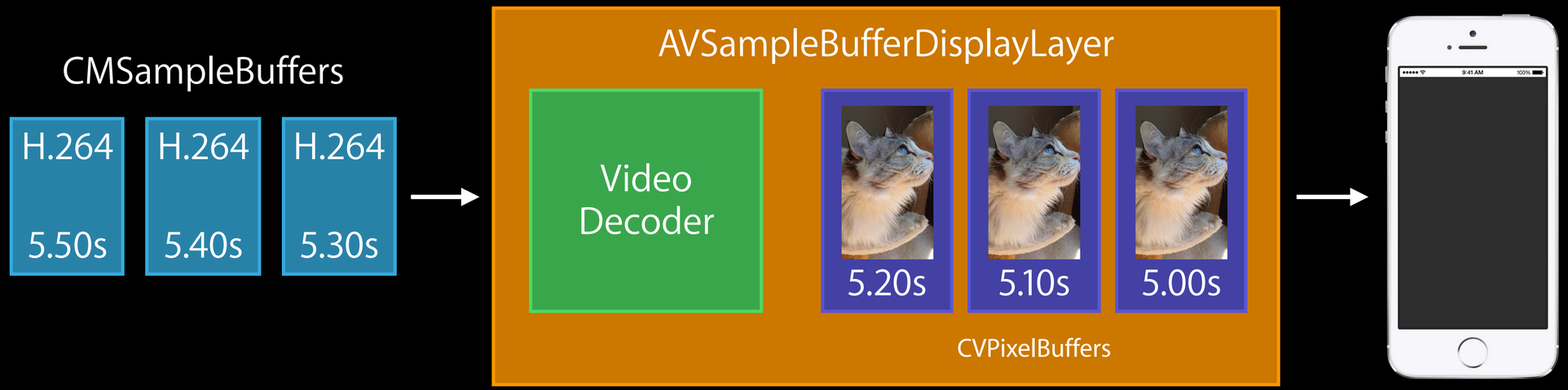
# AVSampleBufferDisplayLayer and Time



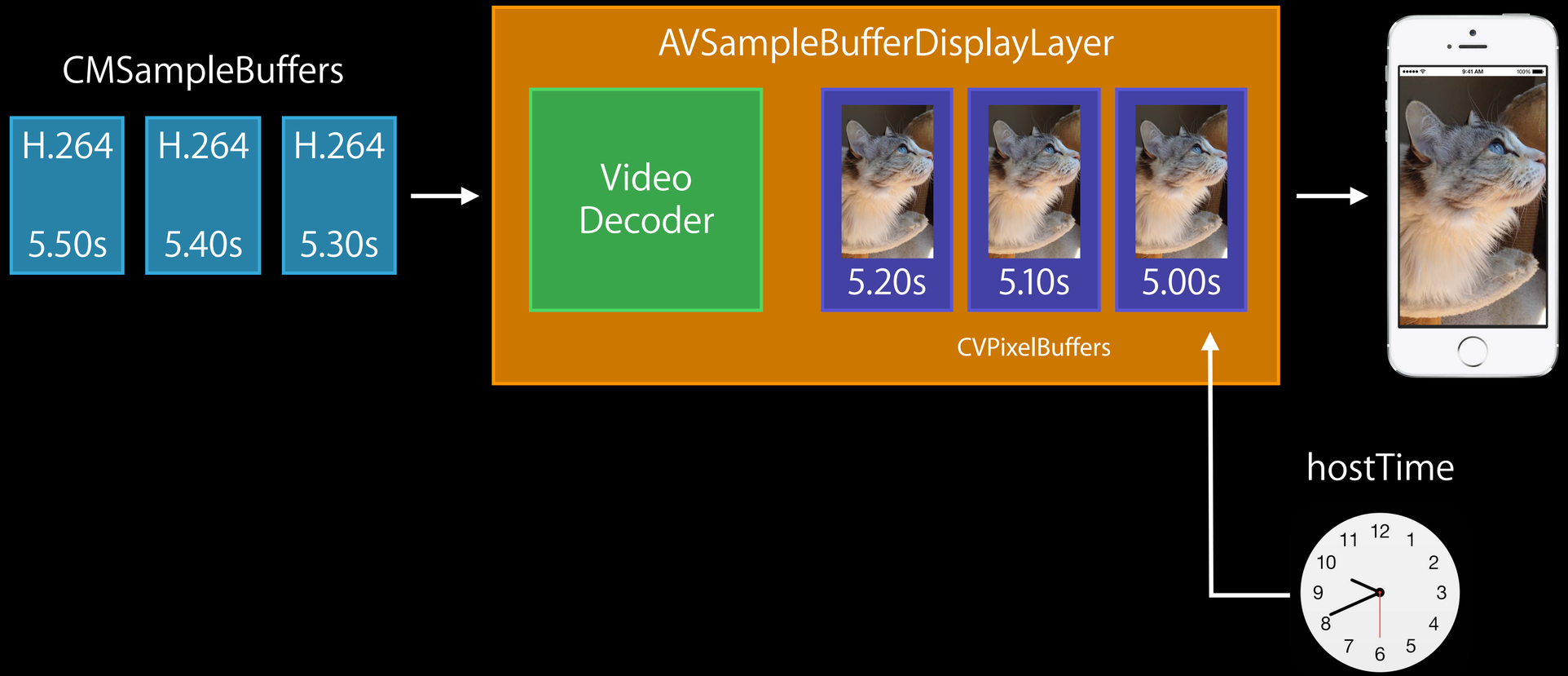
# AVSampleBufferDisplayLayer and Time



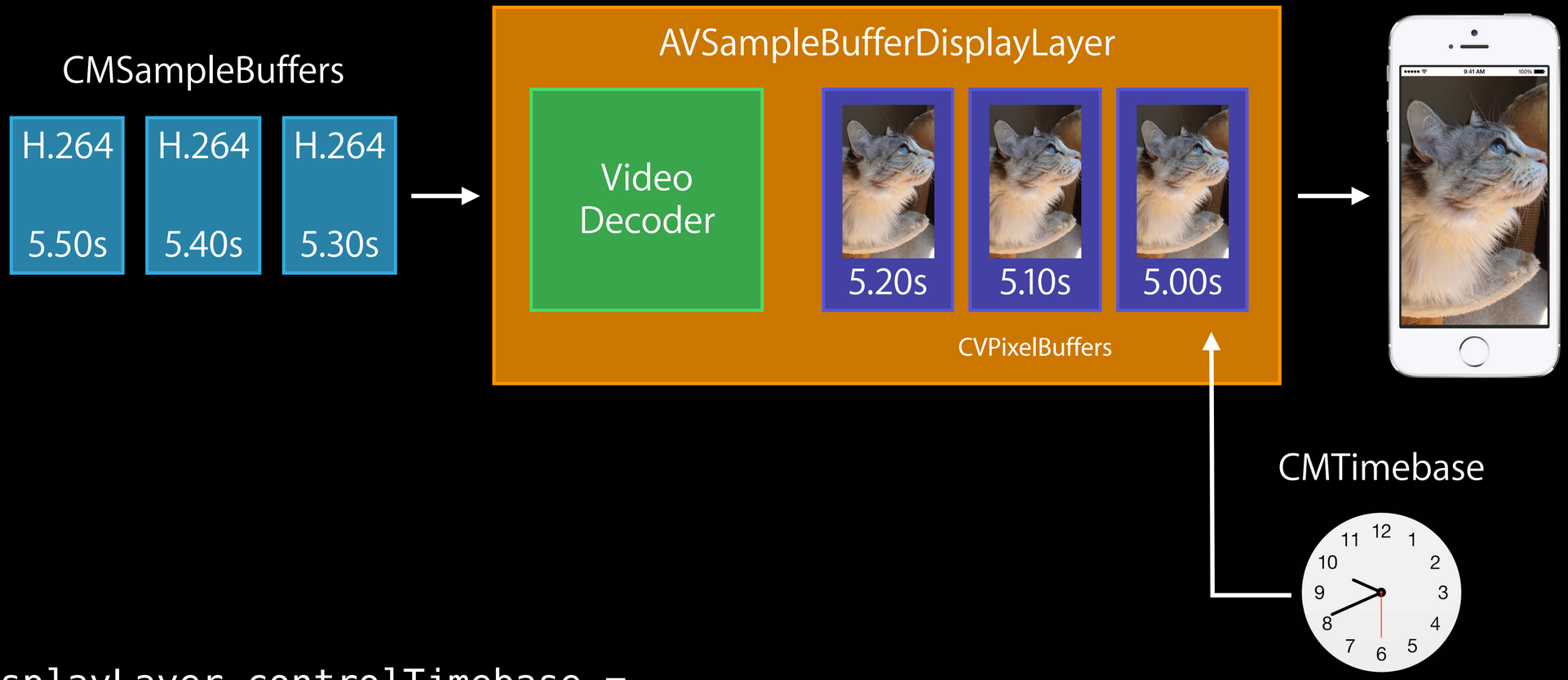
# AVSampleBufferDisplayLayer and Time



# AVSampleBufferDisplayLayer and Time



# AVSampleBufferDisplayLayer and Time




```
sbDisplayLayer.controlTimebase =
    CMTimebaseCreateWithMasterClock(CMClockGetHostTimeClock());
CMTimebaseSetTime(sbDisplayLayer.controlTimebase, CMTimeMake(5, 1));
CMTimebaseSetRate(sbDisplayLayer.controlTimebase, 1.0);
```

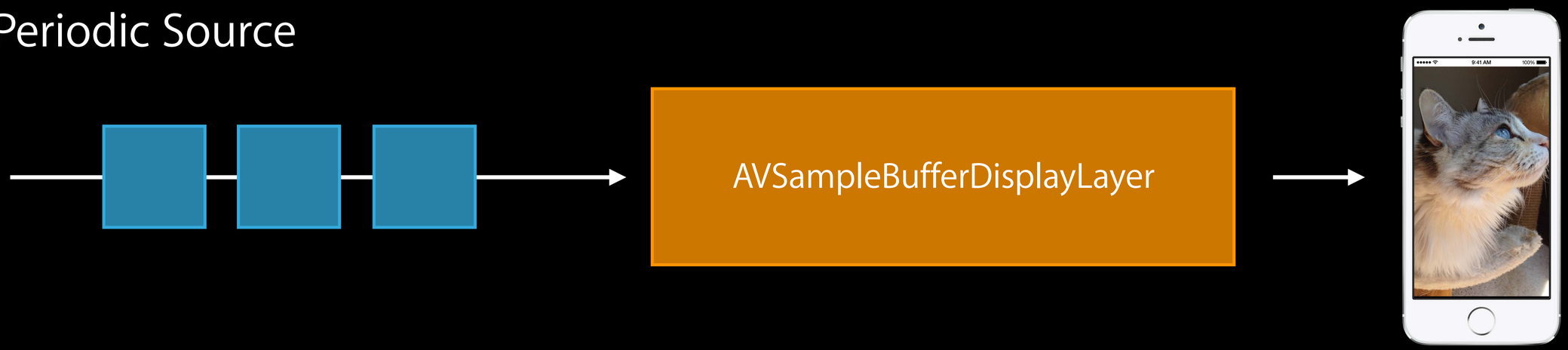
# Feeding AVSampleBufferDisplayLayer

Two scenarios

# Feeding AVSampleBufferDisplayLayer

## Two scenarios

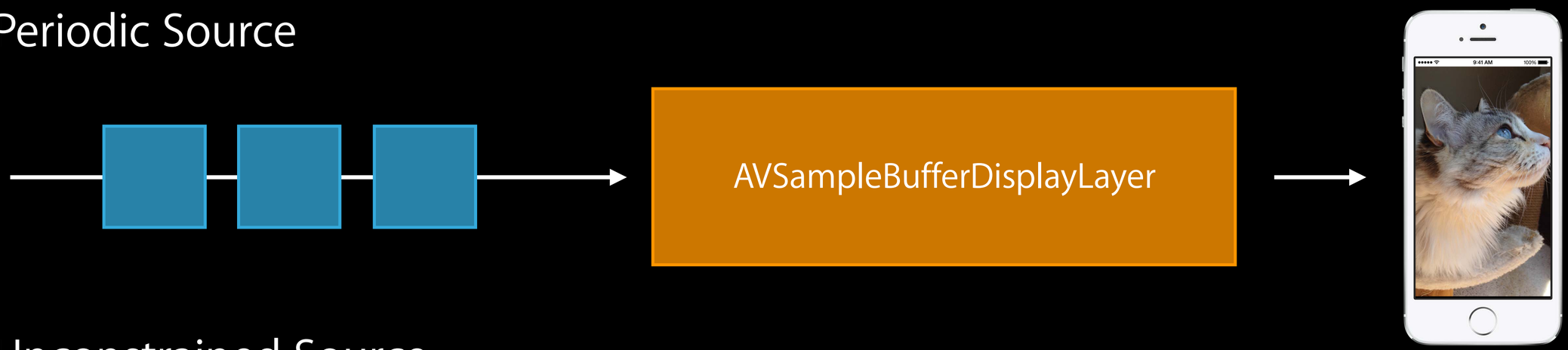
Periodic Source



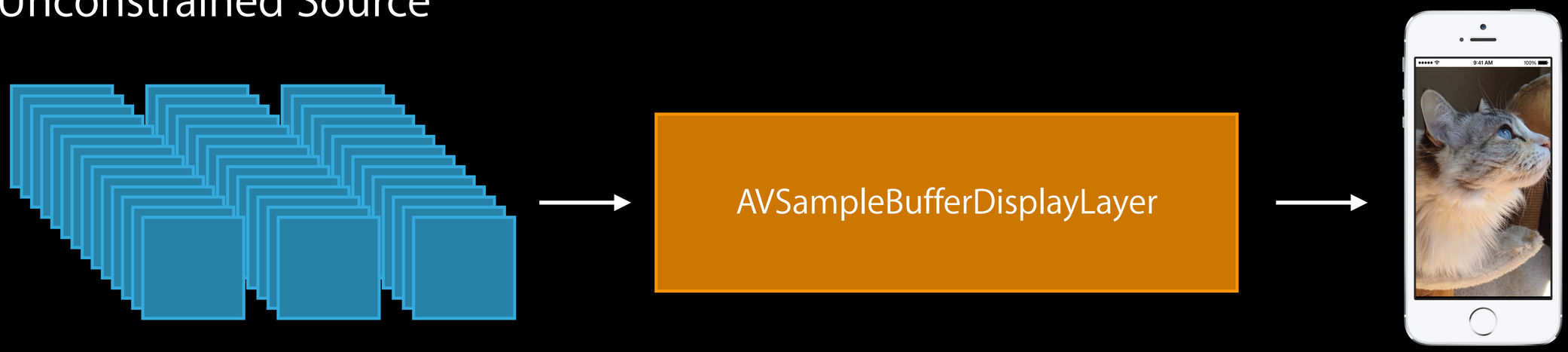
# Feeding AVSampleBufferDisplayLayer

## Two scenarios

Periodic Source

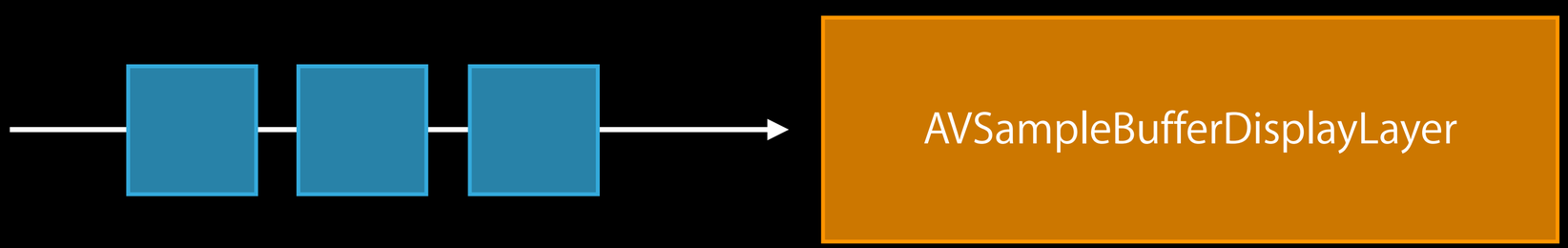



Unconstrained Source



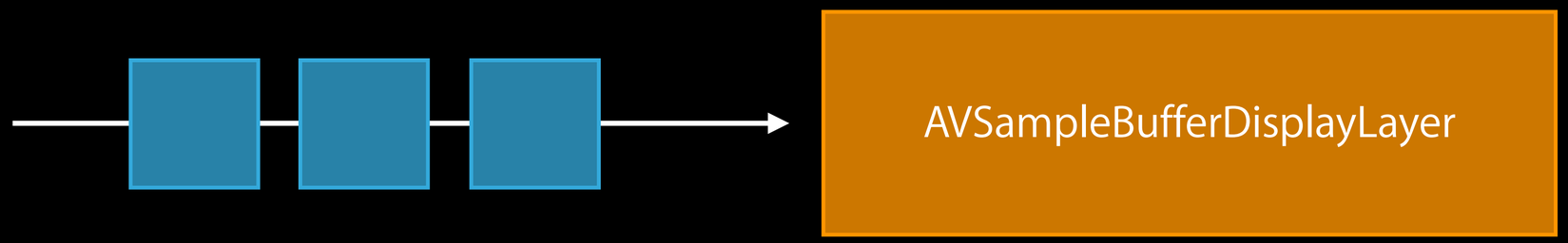
# Feeding AVSampleBufferDisplayLayer

## Periodic source



# Feeding AVSampleBufferDisplayLayer

## Periodic source



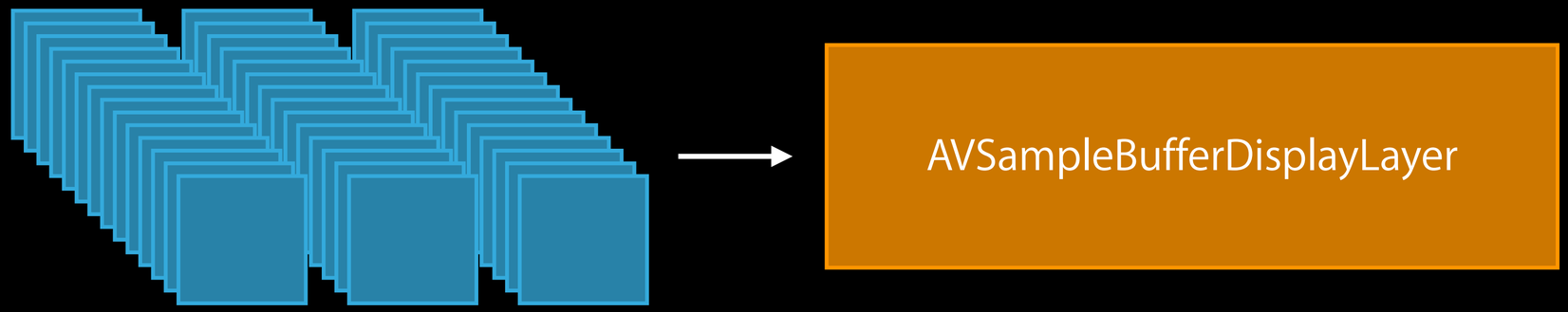

Frame arrival corresponds to display frequency

Enqueue with:

```
[sbDisplayLayer enqueueSampleBuffer:sbuf];
```

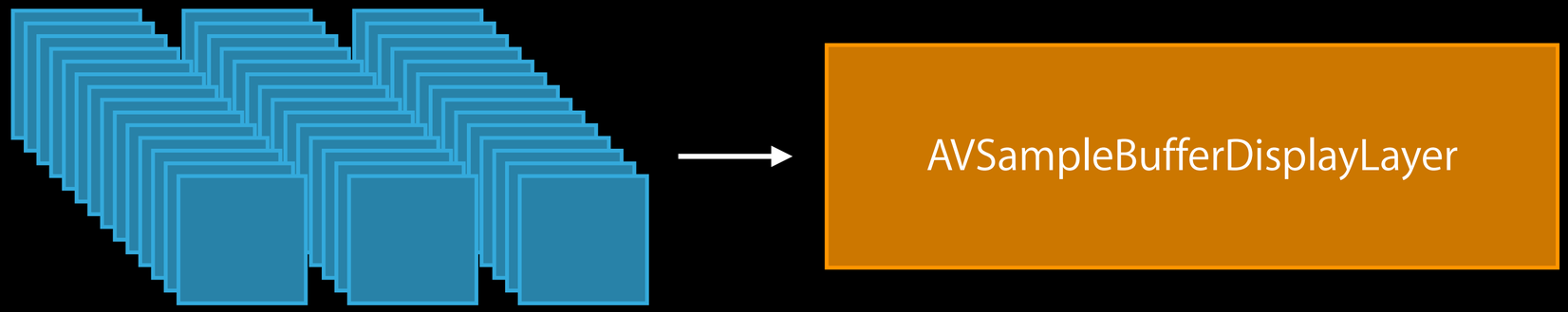
# Feeding AVSampleBufferDisplayLayer

## Unconstrained source



# Feeding AVSampleBufferDisplayLayer

## Unconstrained source

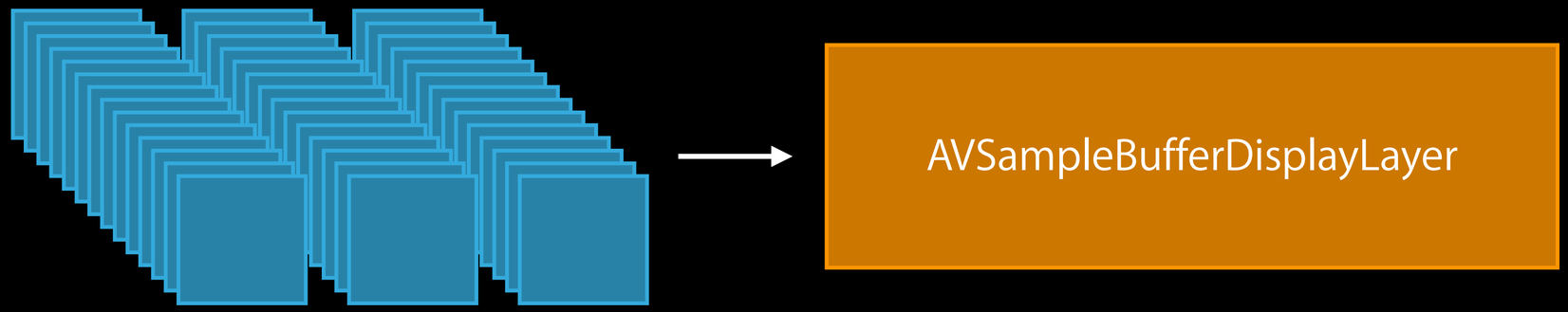



AVSampleBufferDisplayLayer throttles input:

```
[sbDisplayLayer requestMediaDataWhenReadyOnQueue:dispatchQueue usingBlock:^{
    while ([sbDisplayLayer isReadyForMoreMediaData]) {
        CMSampleBuffer sbuf = copyNextSBuf();
        [sbDisplayLayer enqueueSampleBuffer:sbuf];
        CFRelease(sbuf);
    }
}];
```

# Feeding AVSampleBufferDisplayLayer

## Unconstrained source

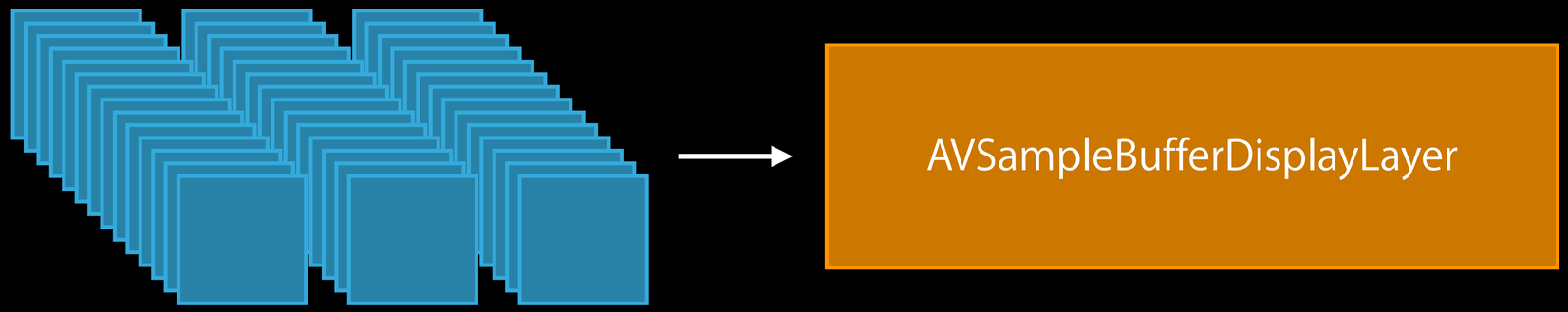



AVSampleBufferDisplayLayer throttles input:

```
[sbDisplayLayer requestMediaDataWhenReadyOnQueue:dispatchQueue usingBlock:^{
    while ([sbDisplayLayer isReadyForMoreMediaData]) {
        CMSampleBuffer sbuf = copyNextSBuf();
        [sbDisplayLayer enqueueSampleBuffer:sbuf];
        CFRelease(sbuf);
    }
}];
```

# Feeding AVSampleBufferDisplayLayer

## Unconstrained source

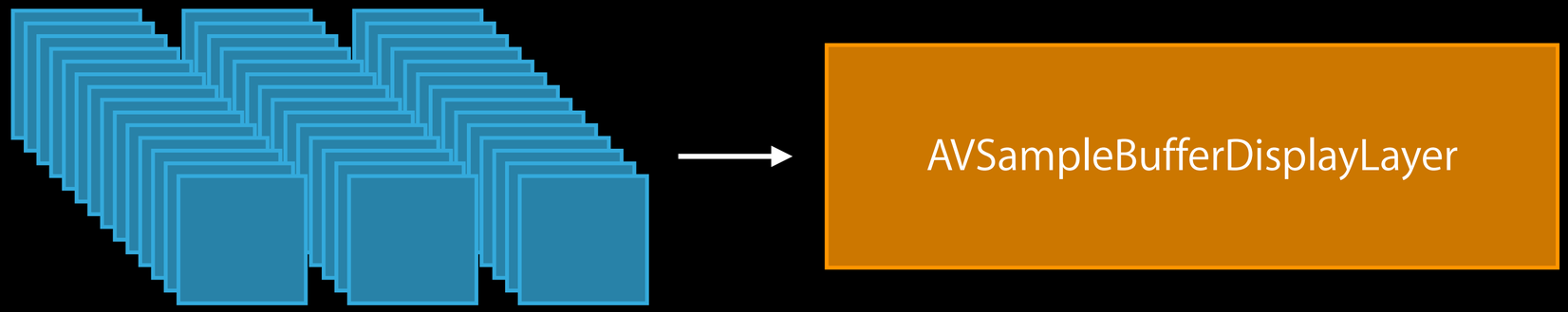



AVSampleBufferDisplayLayer throttles input:

```
[sbDisplayLayer requestMediaDataWhenReadyOnQueue:dispatchQueue usingBlock:^{
    while ([sbDisplayLayer isReadyForMoreMediaData]) {
        CMSampleBuffer sbuf = copyNextSBuf();
        [sbDisplayLayer enqueueSampleBuffer:sbuf];
        CFRelease(sbuf);
    }
}];
```

# Feeding AVSampleBufferDisplayLayer

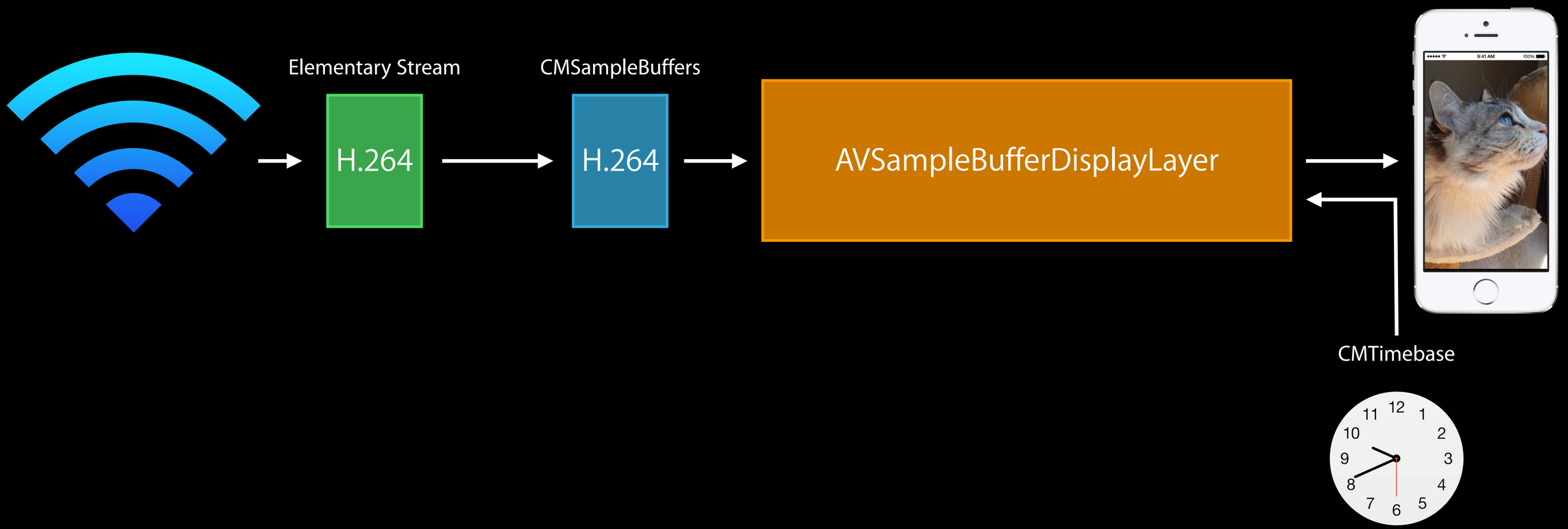
## Unconstrained source



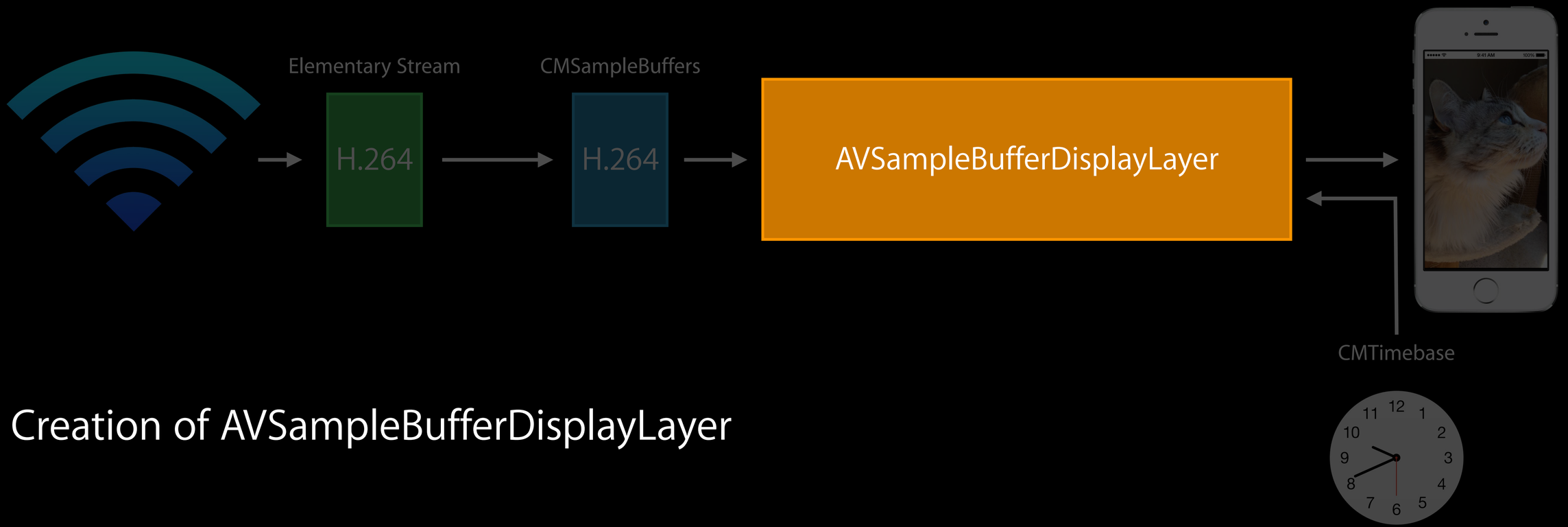

AVSampleBufferDisplayLayer throttles input:

```
[sbDisplayLayer requestMediaDataWhenReadyOnQueue:dispatchQueue usingBlock:^{
    while ([sbDisplayLayer isReadyForMoreMediaData]) {
        CMSampleBuffer sbuf = copyNextSBuf();
        [sbDisplayLayer enqueueSampleBuffer:sbuf];
        CFRelease(sbuf);
    }
}];
```

# AVSampleBufferDisplayLayer Summary

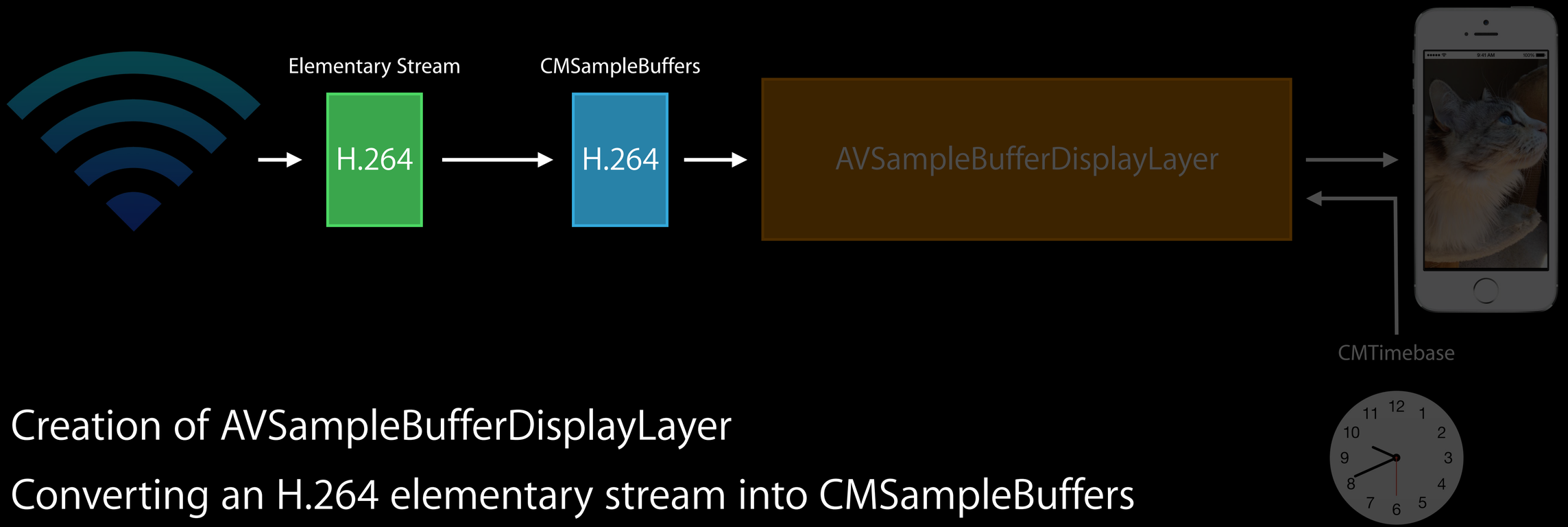


# AVSampleBufferDisplayLayer Summary




## Creation of AVSampleBufferDisplayLayer

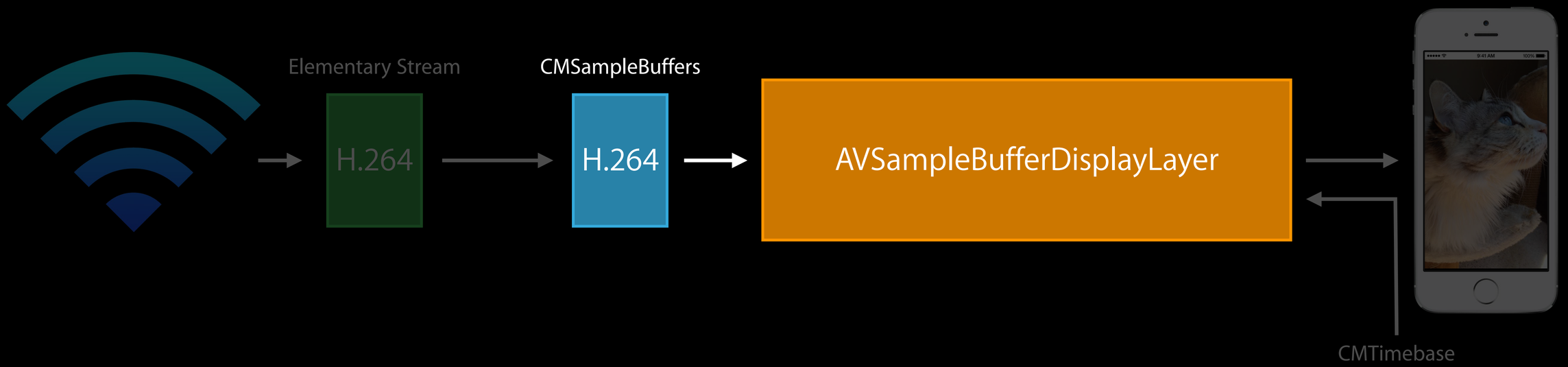
# AVSampleBufferDisplayLayer Summary




Creation of AVSampleBufferDisplayLayer

Converting an H.264 elementary stream into CMSampleBuffers

# AVSampleBufferDisplayLayer Summary

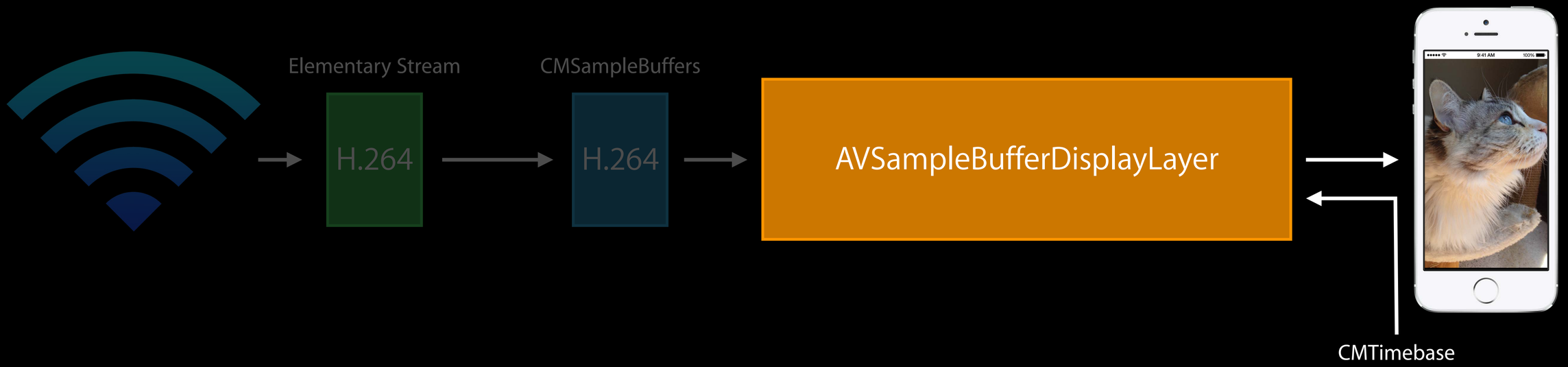



Creation of AVSampleBufferDisplayLayer

Converting an H.264 elementary stream into CMSampleBuffers

Providing CMSampleBuffers to AVSampleBufferDisplayLayer

# AVSampleBufferDisplayLayer Summary




Creation of AVSampleBufferDisplayLayer

Converting an H.264 elementary stream into CMSampleBuffers

Providing CMSampleBuffers to AVSampleBufferDisplayLayer

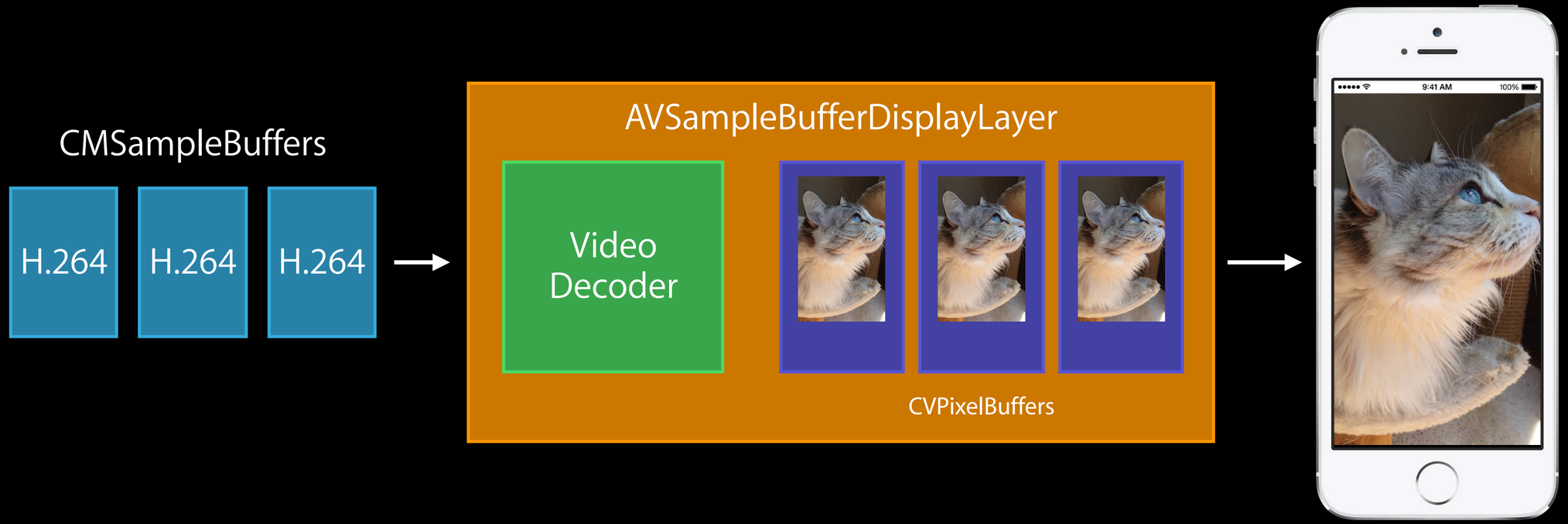
Using a custom CMTimebase with AVSampleBufferDisplayLayer

# Case Two

## Getting CVPixelBuffers from a compressed stream

# AVSampleBufferDisplayLayer

# AVSampleBufferDisplayLayer

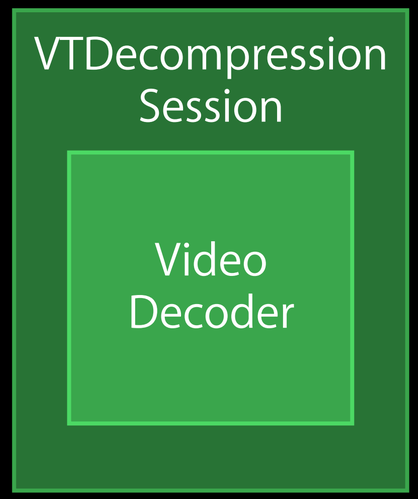


# VTDecompressionSession

## Getting access to the decoder

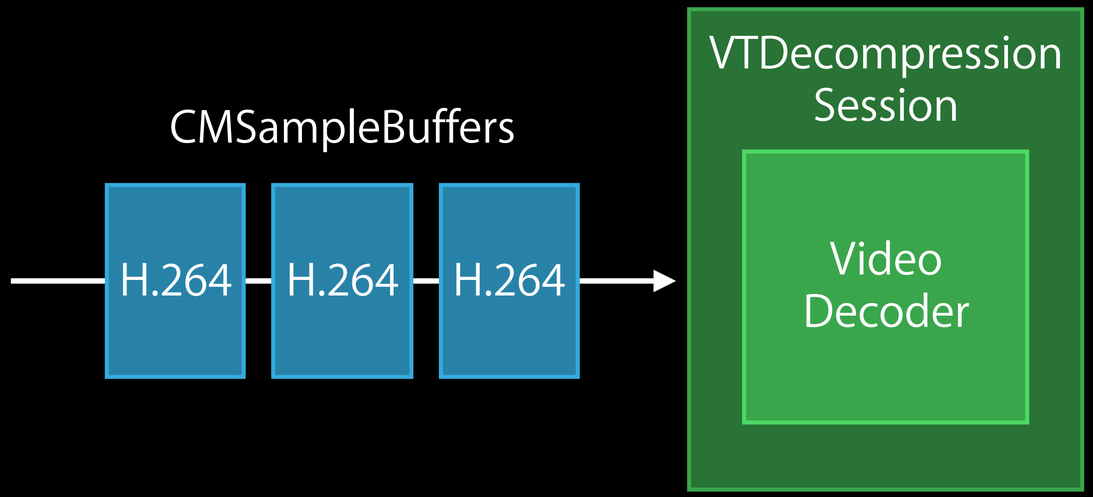
# VTDecompressionSession

## Getting access to the decoder



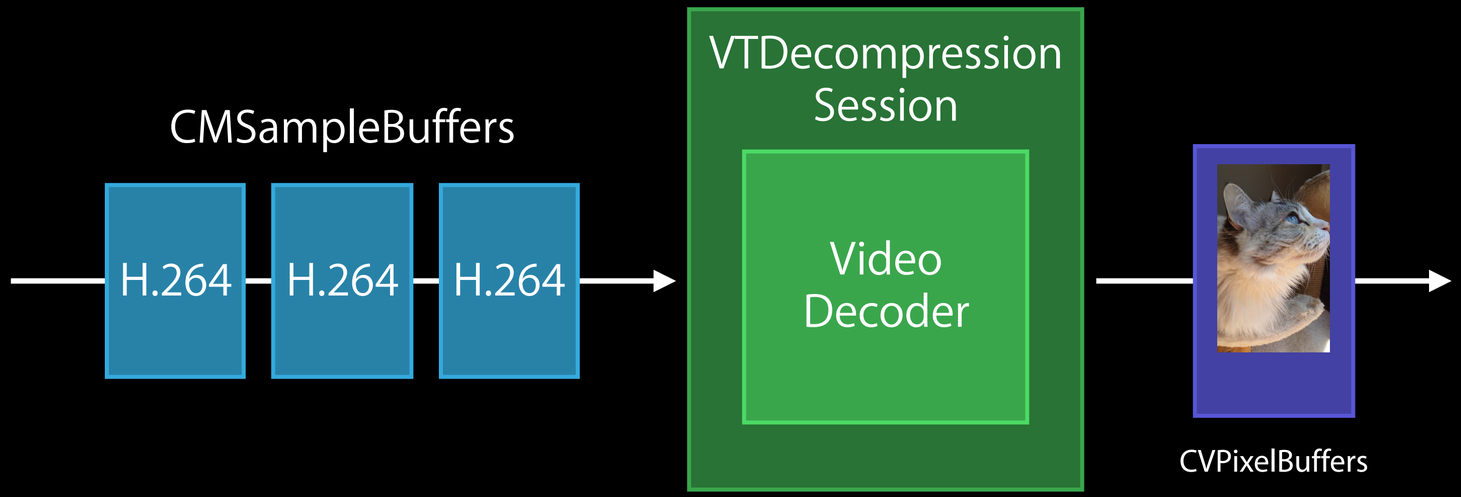
# VTDecompressionSession

## Getting access to the decoder



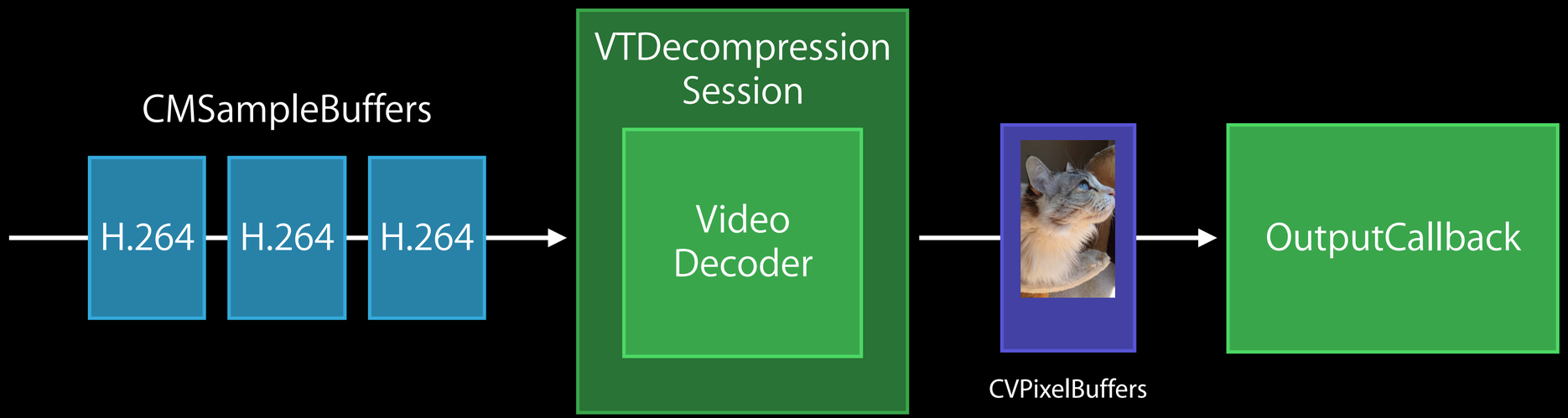
# VTDecompressionSession

## Getting access to the decoder



# VTDecompressionSession

## Getting access to the decoder



# Creating a VTDecompressionSession

What you need

# Creating a VTDecompressionSession

What you need

- Description of source data—CMVideoFormatDescription

# Creating a VTDecompressionSession

What you need

- Description of source data—CMVideoFormatDescription
- Requirements for output buffers—pixelBufferAttributes

# Creating a VTDecompressionSession

What you need

- Description of source data—CMVideoFormatDescription
- Requirements for output buffers—pixelBufferAttributes
- A VTDecompressionOutputCallback

# Requirements for Output CVPixelBuffers

## Creating a pixelBufferAttributes dictionary

# Requirements for Output CVPixelBuffers

## Creating a pixelBufferAttributes dictionary

Example: OpenGL ES render pipeline

# Requirements for Output CVPixelBuffers

## Creating a pixelBufferAttributes dictionary

Example: OpenGL ES render pipeline


To require OpenGL ES compatibility—

```
NSDictionary *destinationImageBufferAttributes =
    [NSDictionary dictionaryWithObjectsAndKeys:
     [NSNumber numberWithBool:YES],(id)kCVPixelBufferOpenGLESCompatibilityKey,
     nil];
```

# Requirements for Output CVPixelBuffers

## Creating a pixelBufferAttributes dictionary

Example: OpenGL ES render pipeline


To require OpenGL ES compatibility—

```
NSDictionary *destinationImageBufferAttributes =
    [NSDictionary dictionaryWithObjectsAndKeys:
     [NSNumber numberWithBool:YES],(id)kCVPixelBufferOpenGLESCompatibilityKey,
     nil];
```

# Optimizing Output

## Do not over specify

# Optimizing Output

## Do not over specify

kCVPixelBufferOpenGLESCompatibilityKey requested

# Optimizing Output

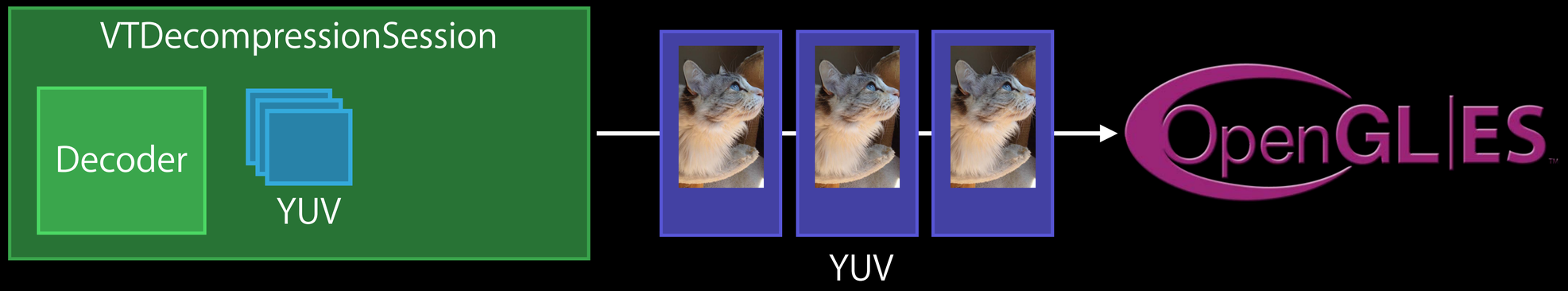
## Do not over specify

kCVPixelBufferOpenGLESCompatibilityKey requested

# Optimizing Output

## Do not over specify

kCVPixelBufferOpenGLESCompatibilityKey requested




kCVPixelBufferOpenGLESCompatibilityKey and ‘BGRA’ pixel format required

# Optimizing Output

## Do not over specify

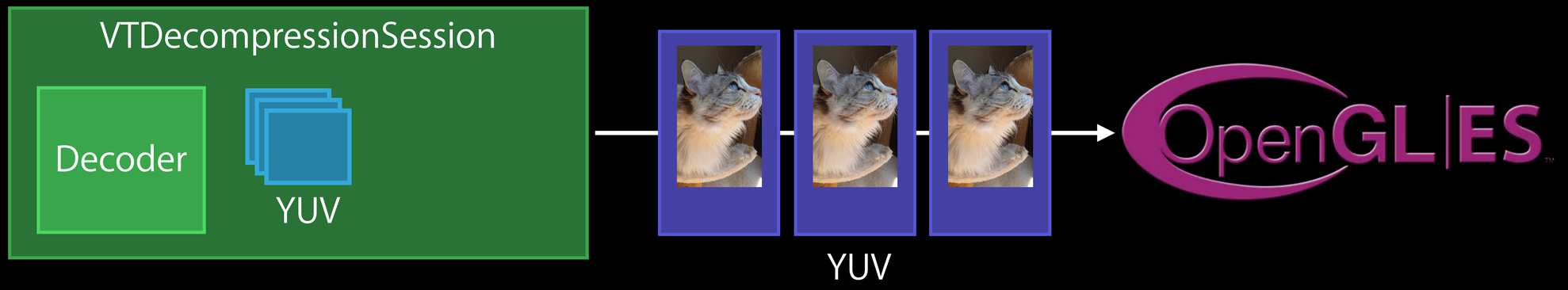
kCVPixelBufferOpenGLESCompatibilityKey requested


kCVPixelBufferOpenGLESCompatibilityKey and ‘BGRA’ pixel format required

# Optimizing Output

## Do not over specify

kCVPixelBufferOpenGLESCompatibilityKey requested

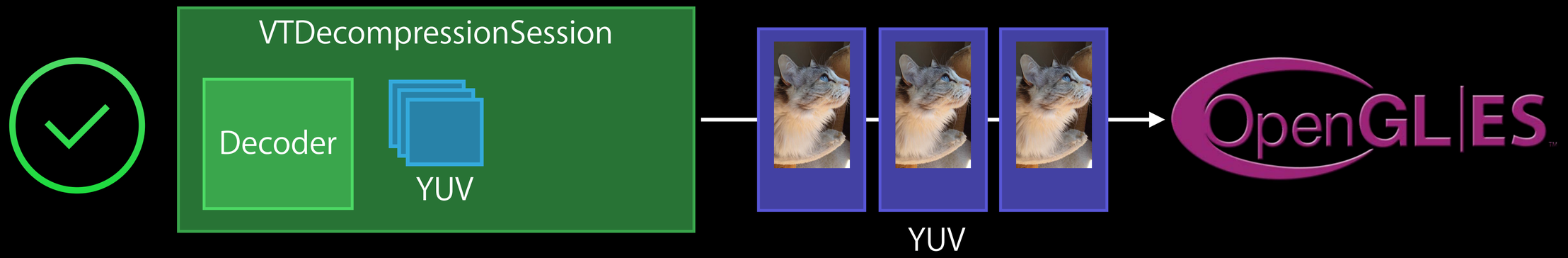



kCVPixelBufferOpenGLESCompatibilityKey and ‘BGRA’ pixel format required

# Optimizing Output

## Do not over specify

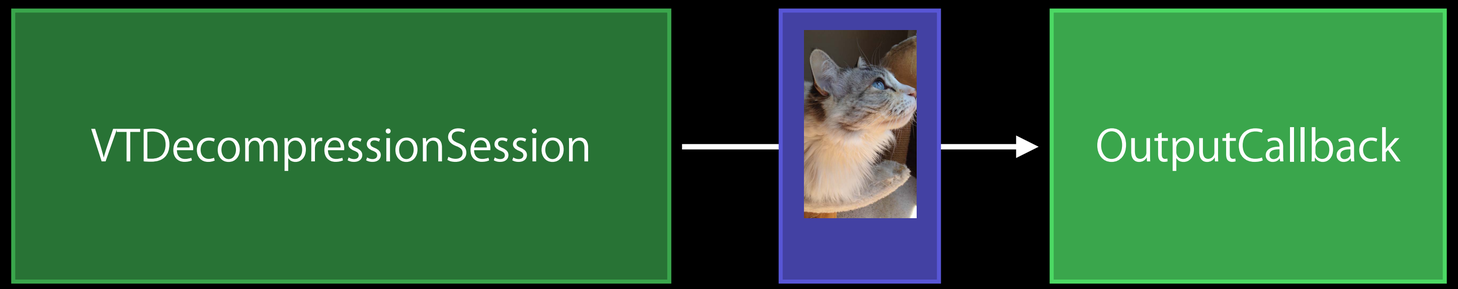
kCVPixelBufferOpenGLESCompatibilityKey requested



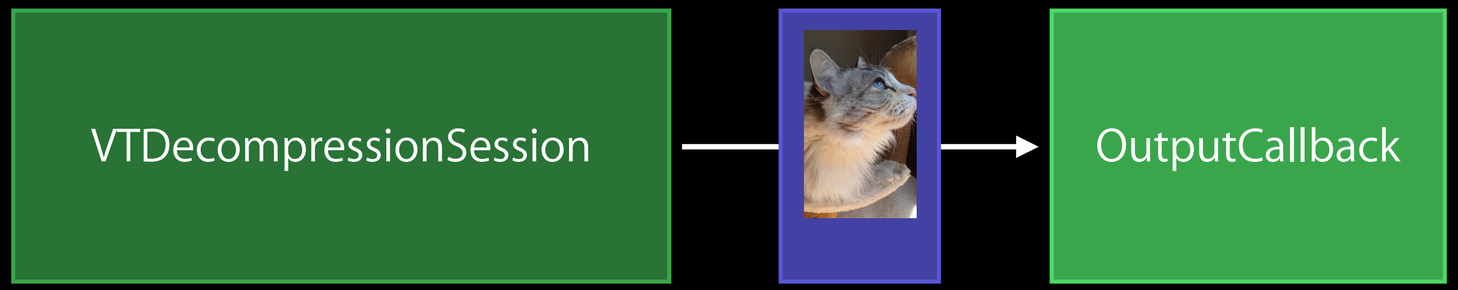

kCVPixelBufferOpenGLESCompatibilityKey and ‘BGRA’ pixel format required

# VTDecompressionOutputCallback

# VTDecompressionOutputCallback

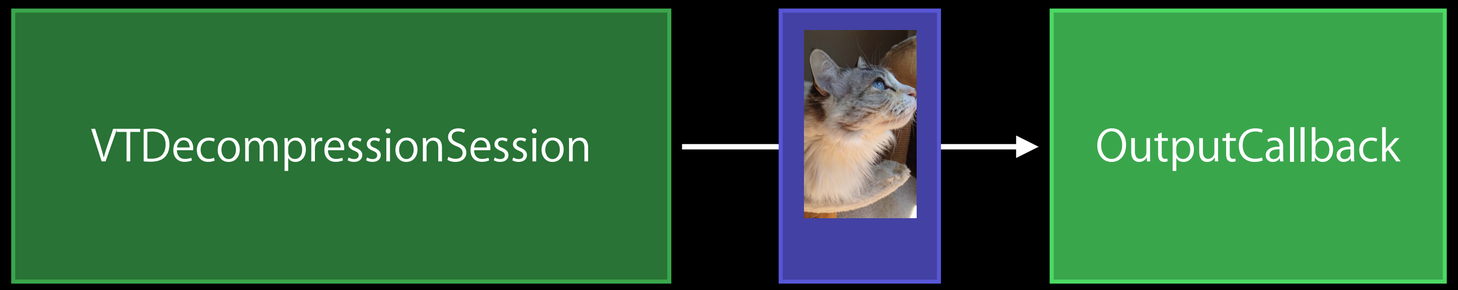


# VTDecompressionOutputCallback




VTDecompressionOutputCallback receives

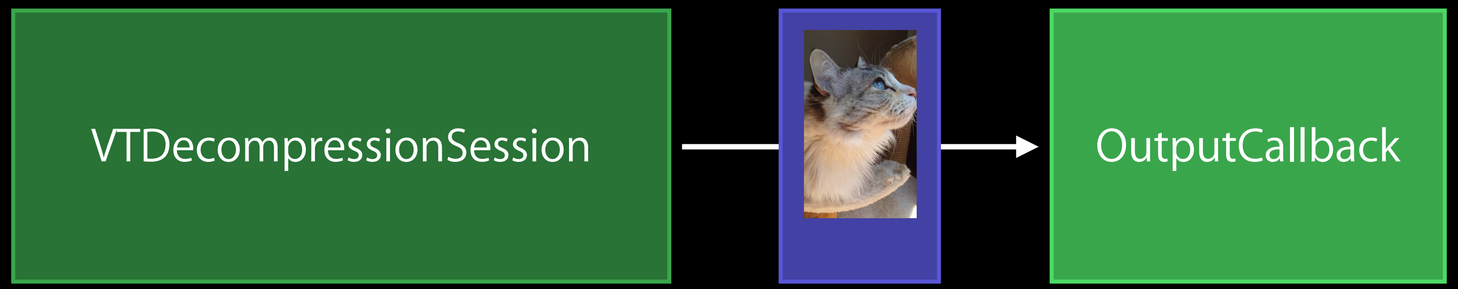
# VTDecompressionOutputCallback




VTDecompressionOutputCallback receives

- Output CVPixelBuffer

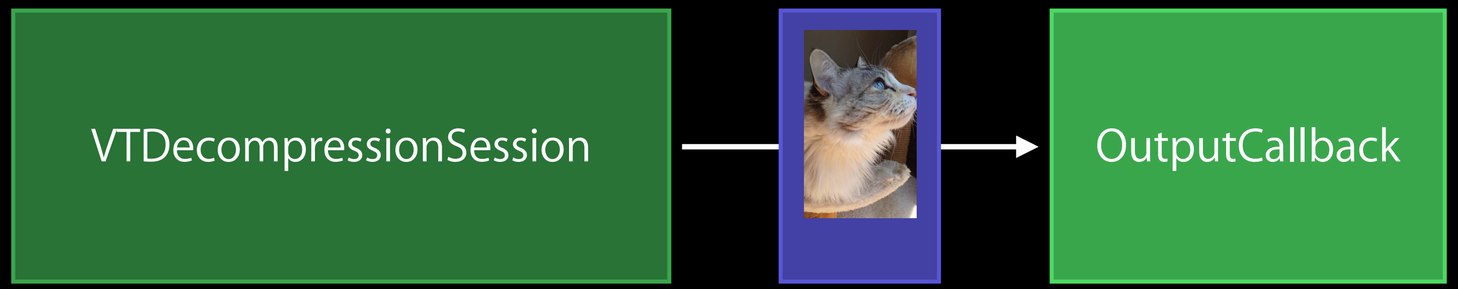
# VTDecompressionOutputCallback




VTDecompressionOutputCallback receives

- Output CVPixelBuffer
- Presentation time stamp

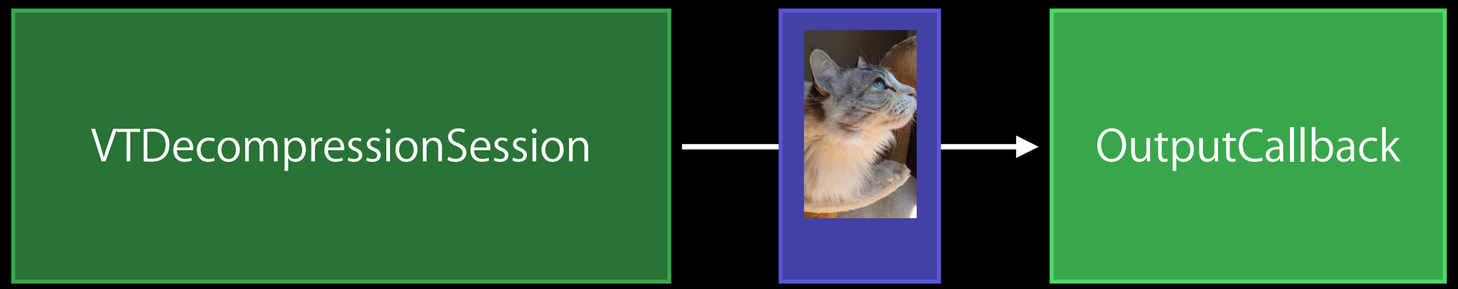
# VTDecompressionOutputCallback




VTDecompressionOutputCallback receives

- Output CVPixelBuffer
- Presentation time stamp
- Decompression error codes

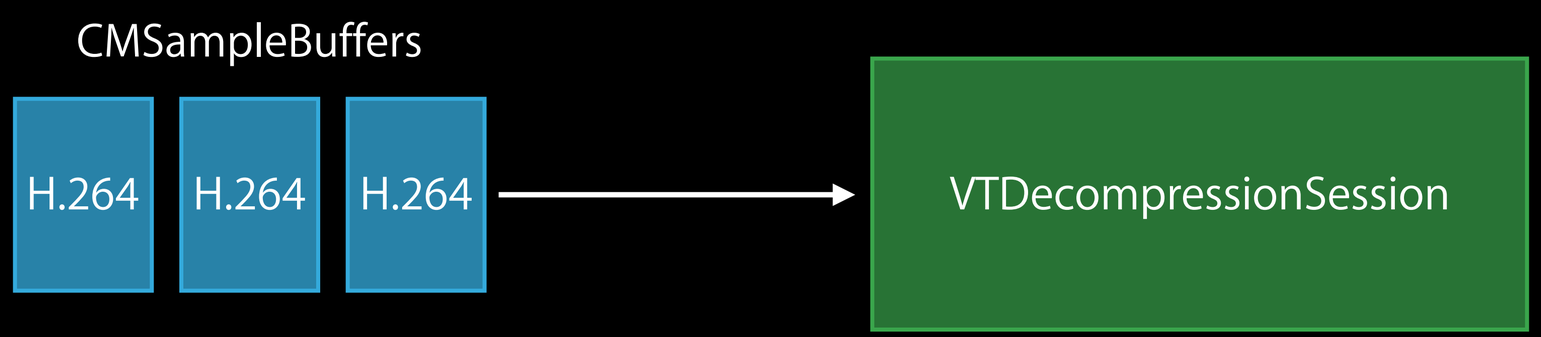
# VTDecompressionOutputCallback



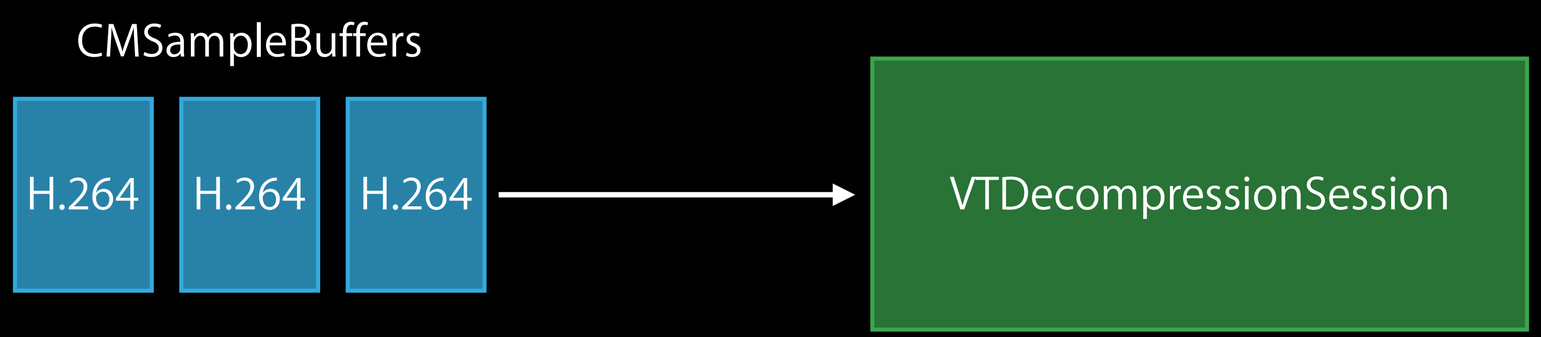

VTDecompressionOutputCallback receives

- Output CVPixelBuffer
- Presentation time stamp
- Decompression error codes
- Dropped frames

# Feeding VTDecompressionSession

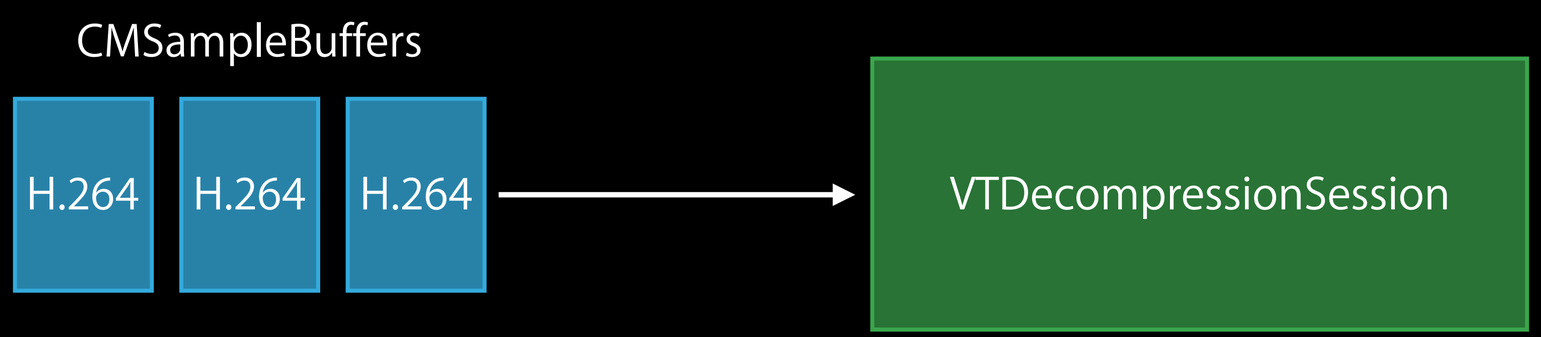


# Feeding VTDecompressionSession




```
err = VTDecompressionSessionDecodeFrame( session, sbuf,
          inFlags, refCon, &outFlags );
```

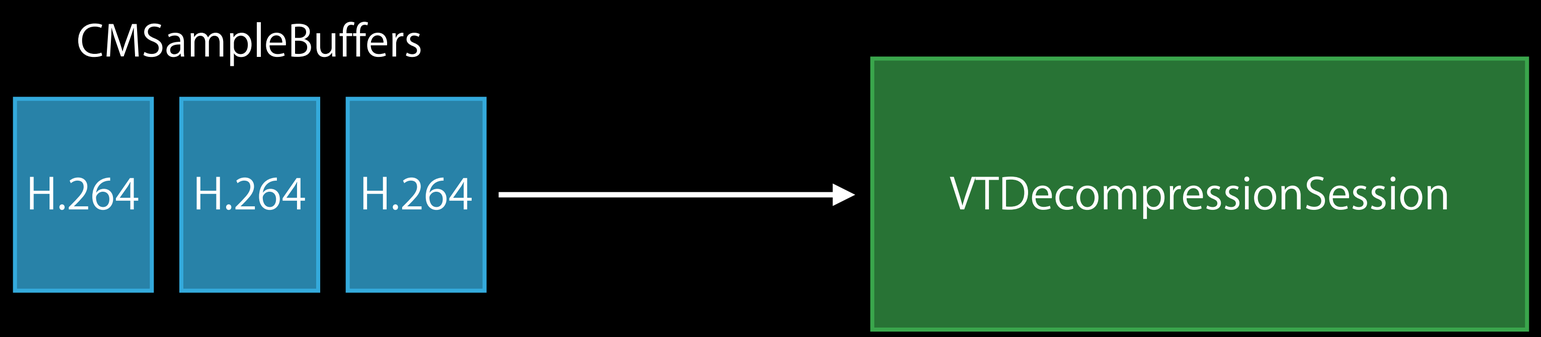
# Feeding VTDecompressionSession




```
err = VTDecompressionSessionDecodeFrame( session, sbuf,
          inFlags, refCon, &outFlags );
```

Wants CMSampleBuffers

# Feeding VTDecompressionSession

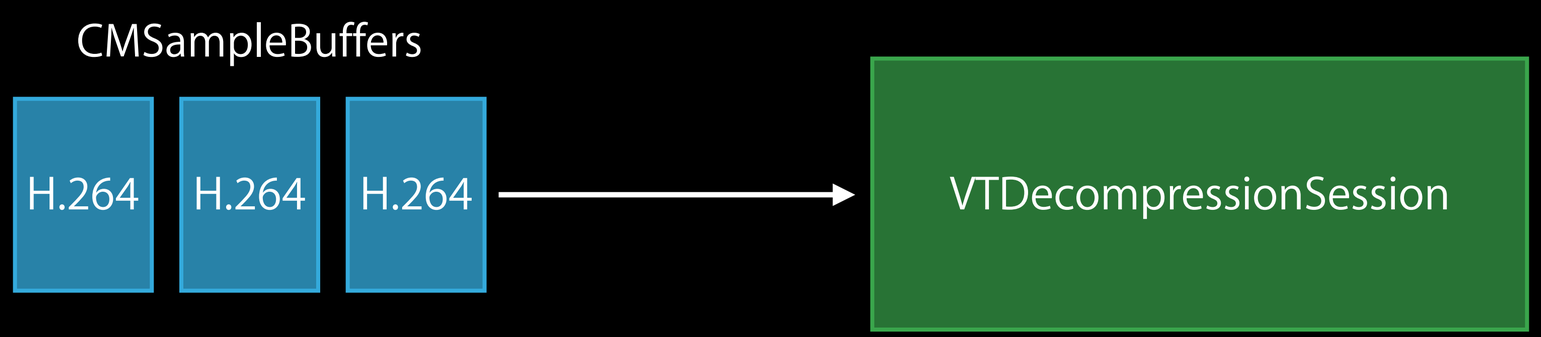



```
err = VTDecompressionSessionDecodeFrame( session, sbuf,
          inFlags, refCon, &outFlags );
```

Wants CMSampleBuffers

Decode order

# Feeding VTDecompressionSession



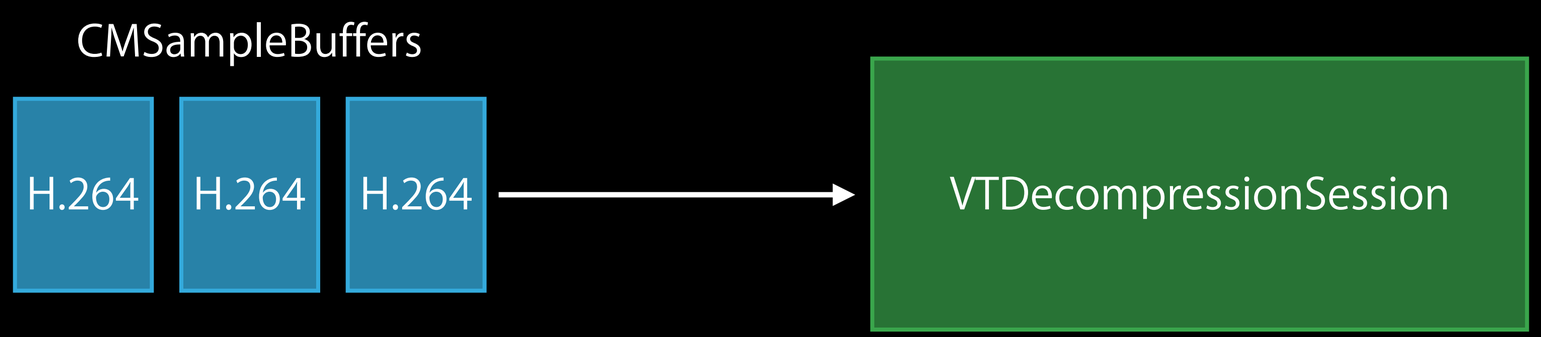

```
err = VTDecompressionSessionDecodeFrame( session, sbuf,
          inFlags, refCon, &outFlags );
```

Wants CMSampleBuffers

Decode order

Synchronous by default

# Feeding VTDecompressionSession




```
err = VTDecompressionSessionDecodeFrame( session, sbuf,
          inFlags, refCon, &outFlags );
```

Wants CMSampleBuffers

Decode order

Synchronous by default

Set kVTDecodeFrame_EnableAsynchronousDecompression for async

# Async Decompression

# Async Decompression

Decoder blocks when full—Decoder back pressure

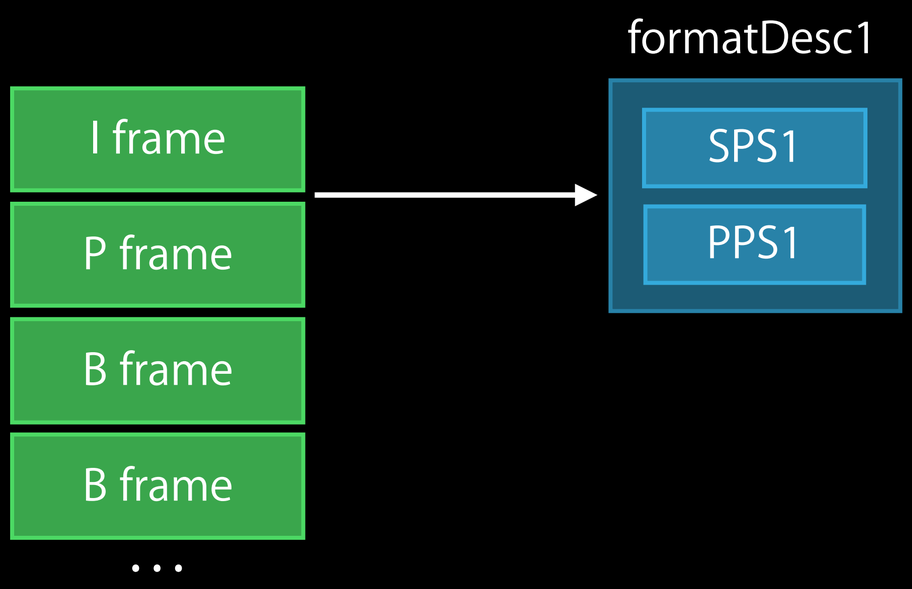
# Async Decompression

Decoder blocks when full—Decoder back pressure

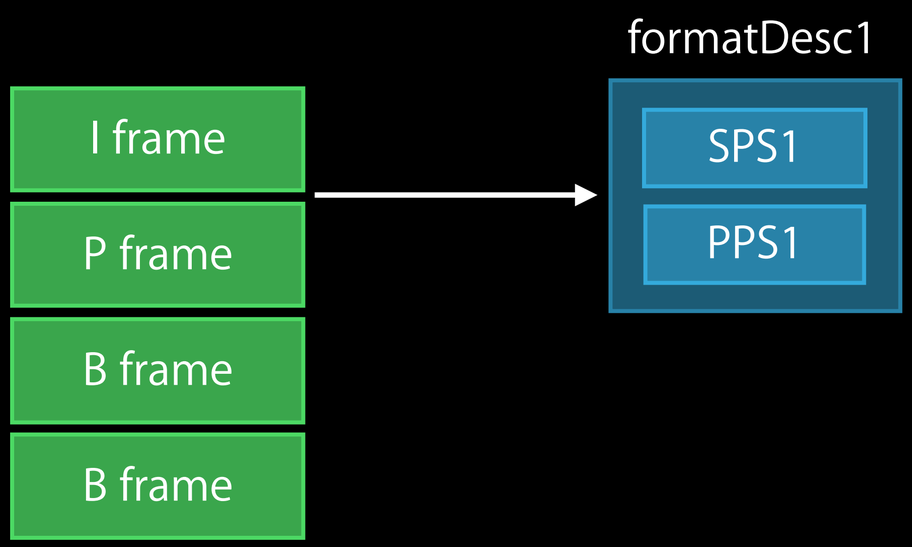
Finish async frames with VTDecompressionSessionWaitForAsynchronousFrames

# Changing CMVideoFormatDescription

# Changing CMVideoFormatDescription

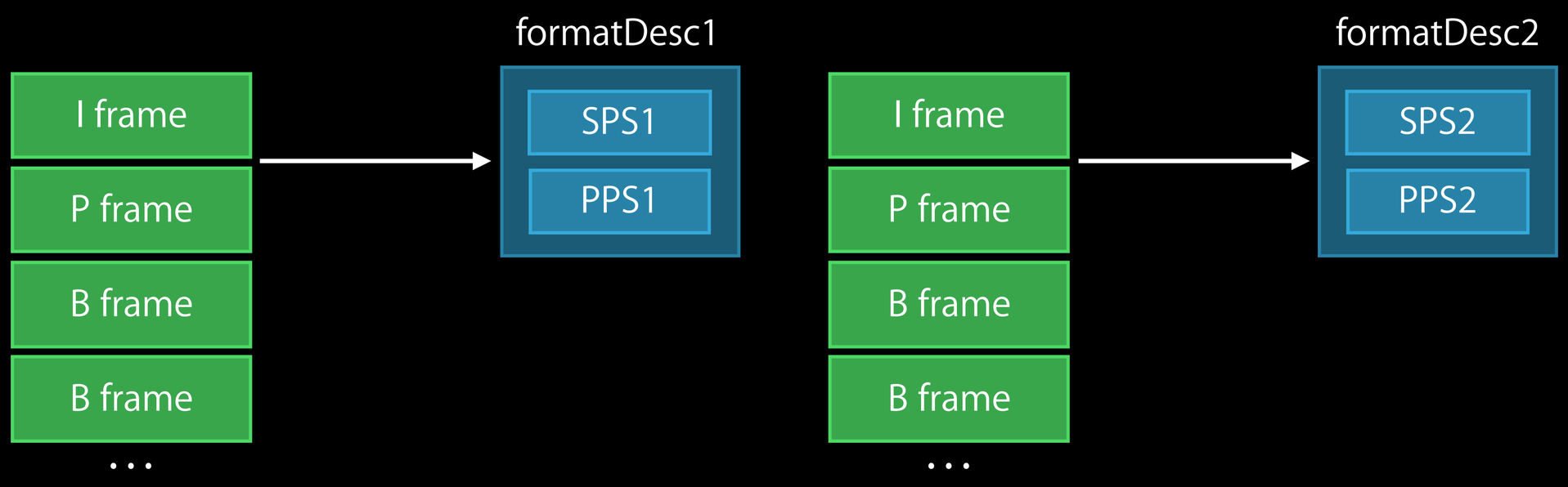


# Changing CMVideoFormatDescription



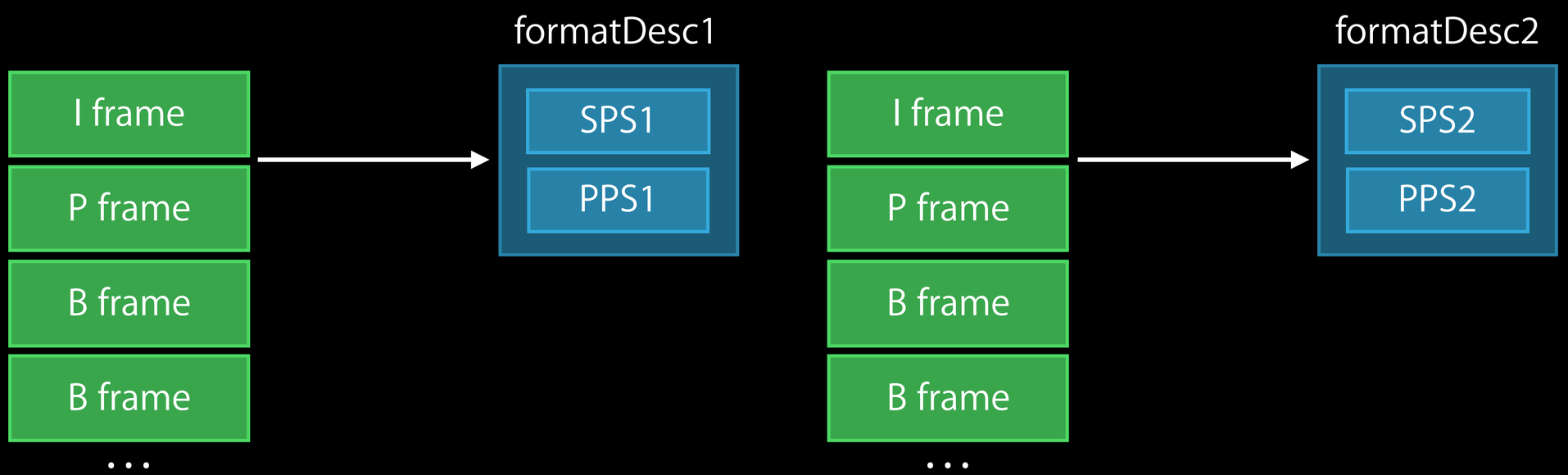

```
err = VTDecompressionSessionCreate( kCFAllocatorDefault, formatDesc1, … ,
                                    &session );
```

# Changing CMVideoFormatDescription




```
err = VTDecompressionSessionCreate( kCFAllocatorDefault, formatDesc1, … ,
                                    &session );
```

# Changing CMVideoFormatDescription




```
err = VTDecompressionSessionCreate( kCFAllocatorDefault, formatDesc1, … ,
                                    &session );

Boolean needNewSession = ( VTDecompressionSessionCanAcceptFormatDescription(
                                    session, formatDesc2 ) == false);
```

# VTDecompressionSession Summary

# VTDecompressionSession Summary

Creation of VTDecompressionSession

# VTDecompressionSession Summary

Creation of VTDecompressionSession

Make optimal decisions about output requirements

# VTDecompressionSession Summary

Creation of VTDecompressionSession

Make optimal decisions about output requirements

Run your VTDecompressionSession synchronously and asynchronously

# VTDecompressionSession Summary

Creation of VTDecompressionSession

Make optimal decisions about output requirements

Run your VTDecompressionSession synchronously and asynchronously

Handling changes in CMVideoFormatDescription

# Case Three

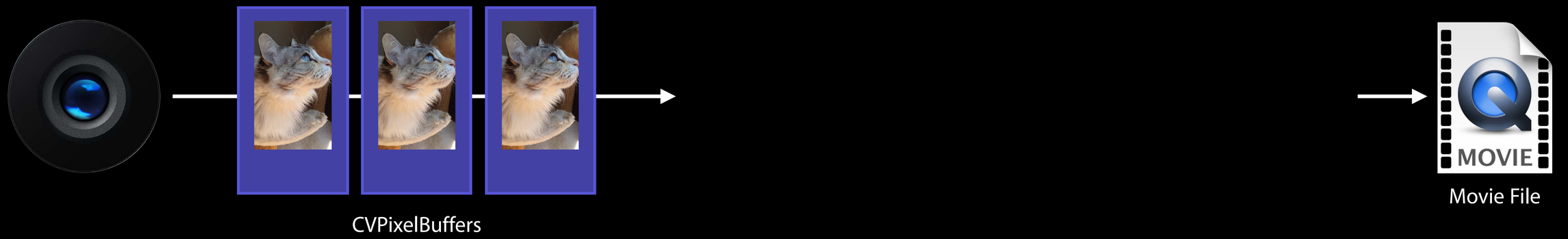
Compressing CVPixelBuffers into a file

# Compressing Video into a File

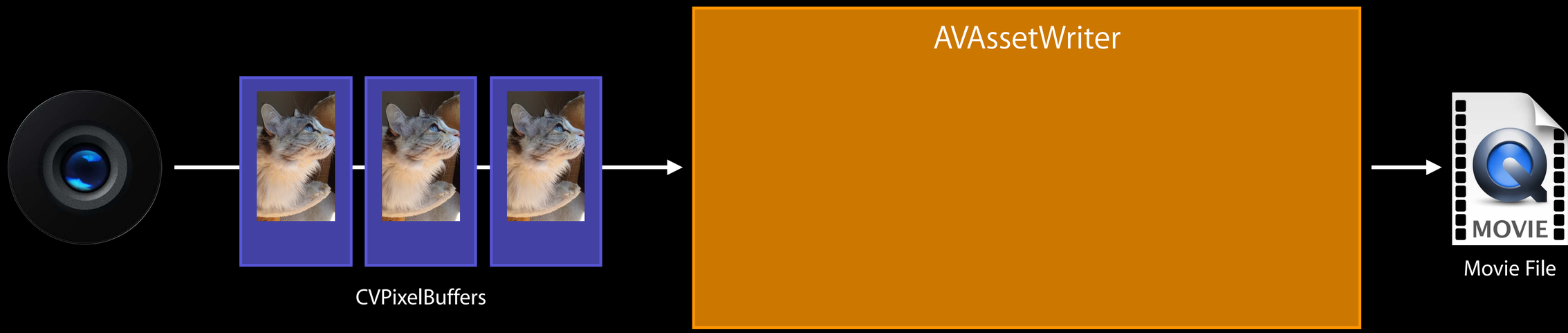
# Compressing Video into a File

CVPixelBuffers

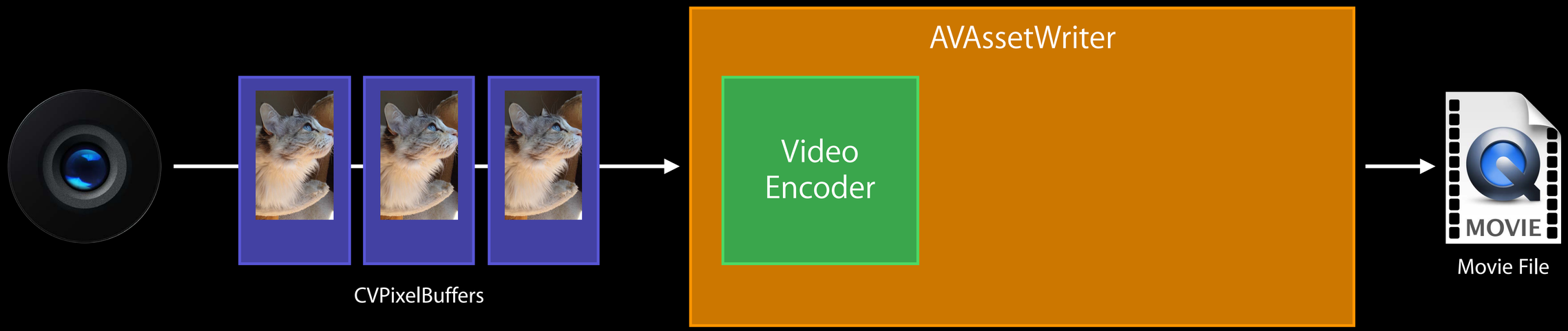
# Compressing Video into a File



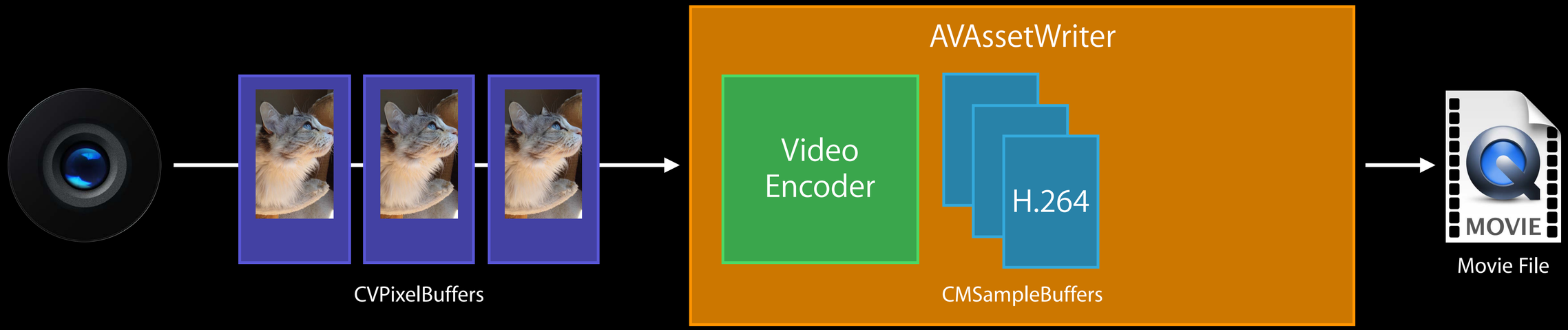
# Compressing Video into a File



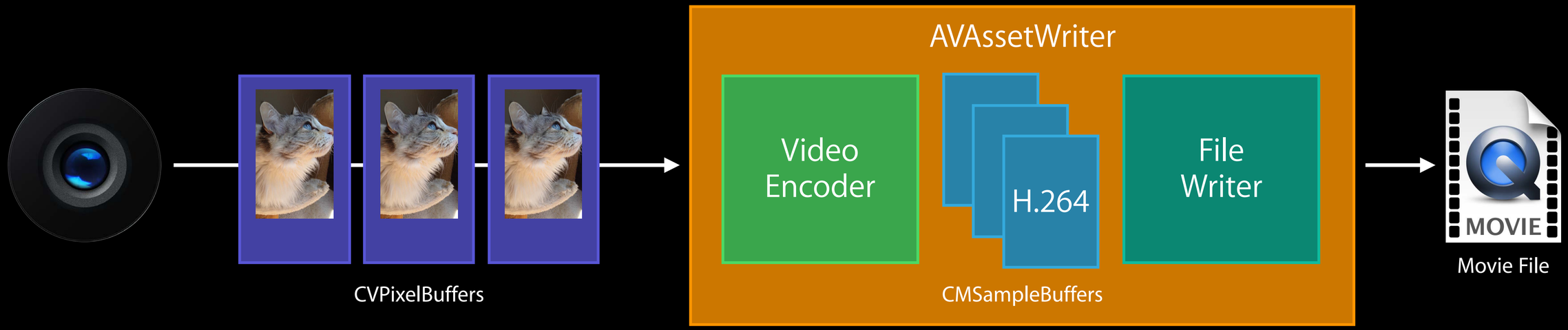
# Compressing Video into a File



# Compressing Video into a File



# Compressing Video into a File



# For More Details on AVAssetWriter

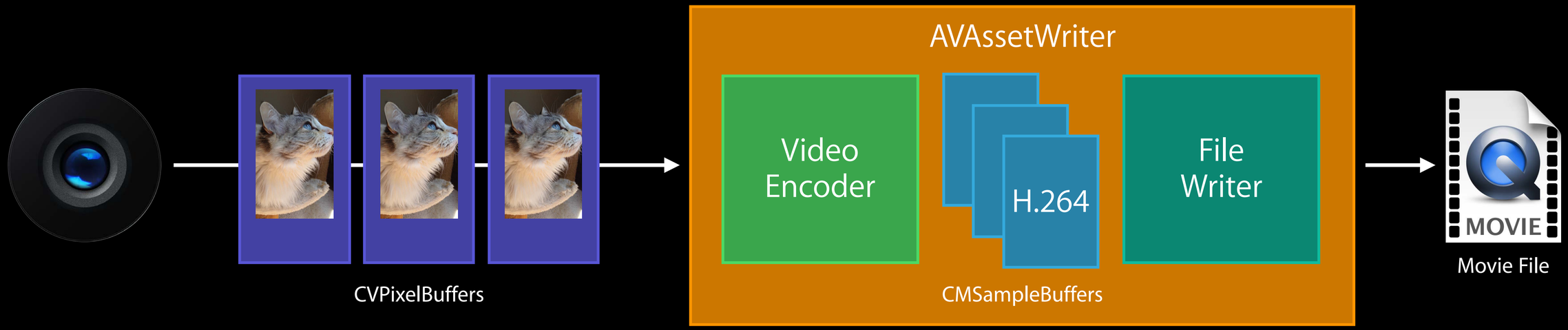
WWDC 2013—Moving to AVKit and AVFoundation

WWDC 2011—Working with Media in AVFoundation

# Case Four

Compressing CVPixelBuffers for the network

# Back Inside AVAssetWriter



# VTCompressionSession

## Getting access to the encoder

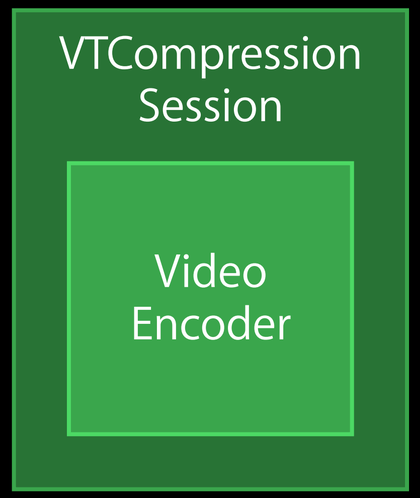
# VTCompressionSession

## Getting access to the encoder

Video
Encoder

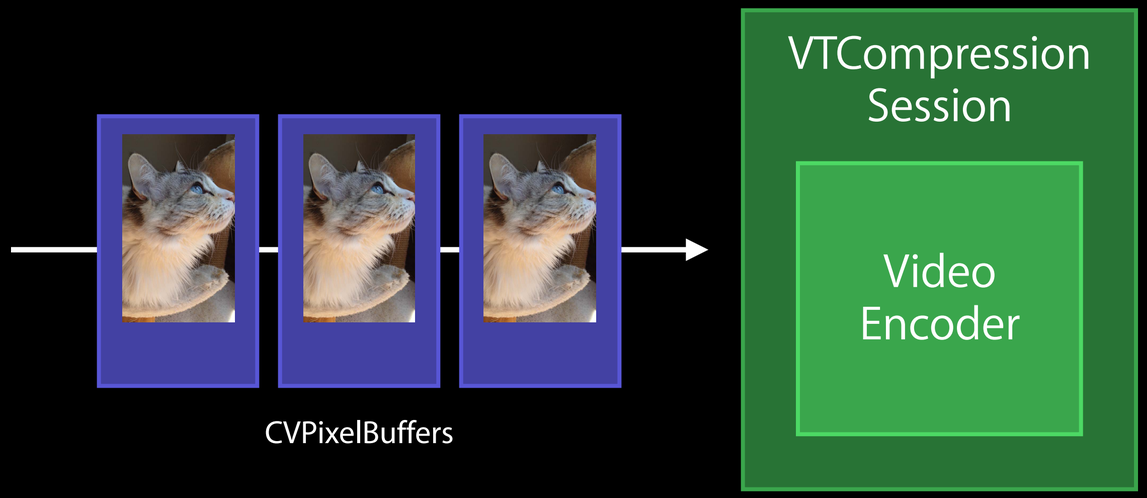
# VTCompressionSession

## Getting access to the encoder



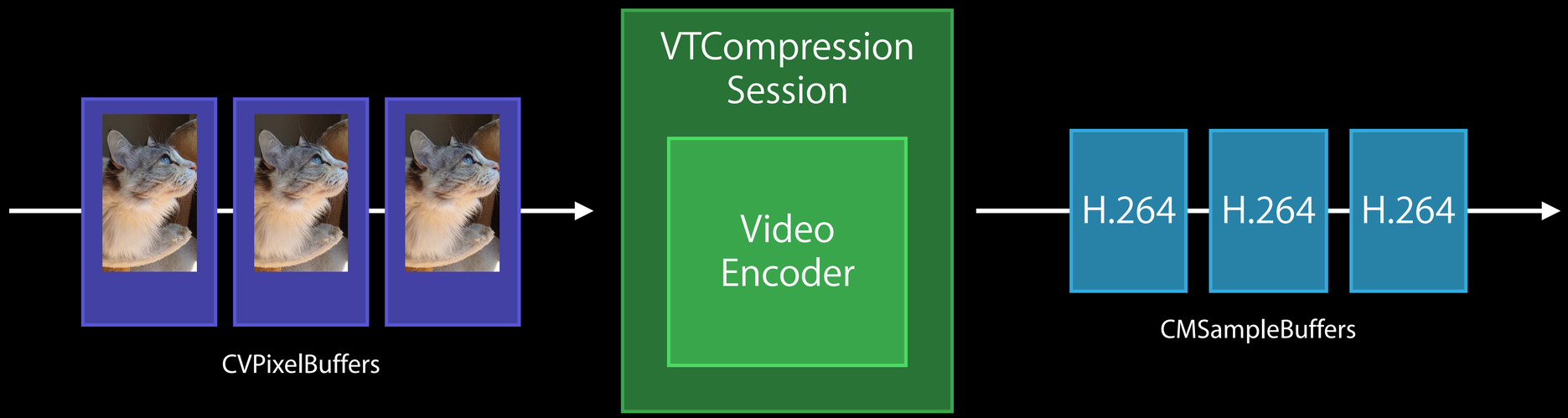
# VTCompressionSession

## Getting access to the encoder



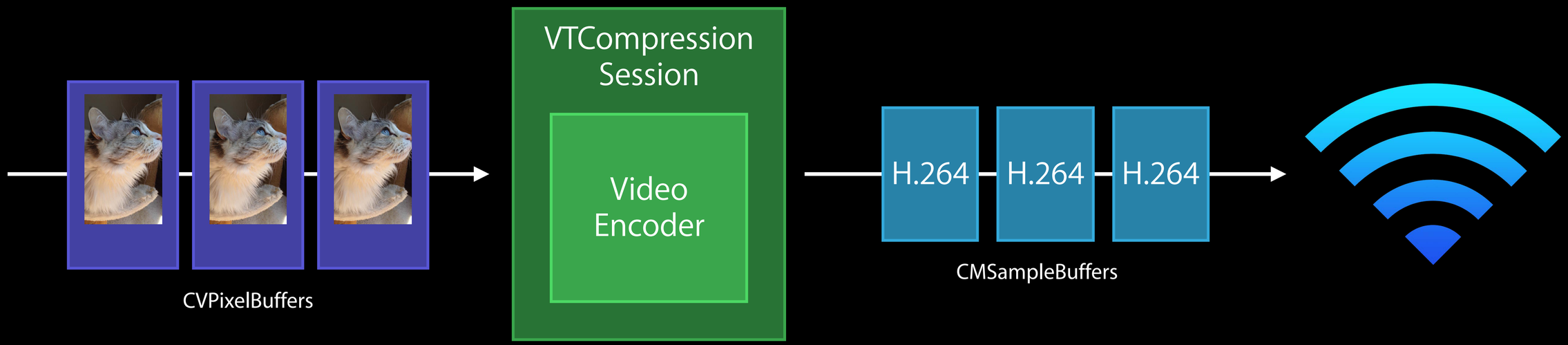
# VTCompressionSession

## Getting access to the encoder



# VTCompressionSession

## Getting access to the encoder



# Creating a VTCompressionSession

What you need:

# Creating a VTCompressionSession

What you need:

- Dimensions for compressed output

# Creating a VTCompressionSession

What you need:

- Dimensions for compressed output
- Format for compression (e.g., kCMVideoCodecType_H264)

# Creating a VTCompressionSession

What you need:

- Dimensions for compressed output
- Format for compression (e.g., kCMVideoCodecType_H264)
- PixelBufferAttributes describing source buffer requirements (optional)

# Creating a VTCompressionSession

What you need:

- Dimensions for compressed output
- Format for compression (e.g., kCMVideoCodecType_H264)
- PixelBufferAttributes describing source buffer requirements (optional)
- A VTCompressionOutputCallback

# Configuring VTCompressionSession

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

`kVTCompressionPropertyKey_AllowFrameReordering`

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
```

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
kVTCompressionPropertyKey_H264EntropyMode
```

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
kVTCompressionPropertyKey_H264EntropyMode
kVTH264EntropyMode_CAVLC/kVTH264EntropyMode_CABAC
```

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
kVTCompressionPropertyKey_H264EntropyMode
    kVTH264EntropyMode_CAVLC/kVTH264EntropyMode_CABAC
kVTCompressionPropertyKey_RealTime
```

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
kVTCompressionPropertyKey_H264EntropyMode
    kVTH264EntropyMode_CAVLC/kVTH264EntropyMode_CABAC
kVTCompressionPropertyKey_RealTime
kVTCompressionPropertyKey_ProfileLevel
```

# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
kVTCompressionPropertyKey_H264EntropyMode
    kVTH264EntropyMode_CAVLC/kVTH264EntropyMode_CABAC
kVTCompressionPropertyKey_RealTime
kVTCompressionPropertyKey_ProfileLevel
    for example: kVTProfileLevel_H264_Main_AutoLevel
```

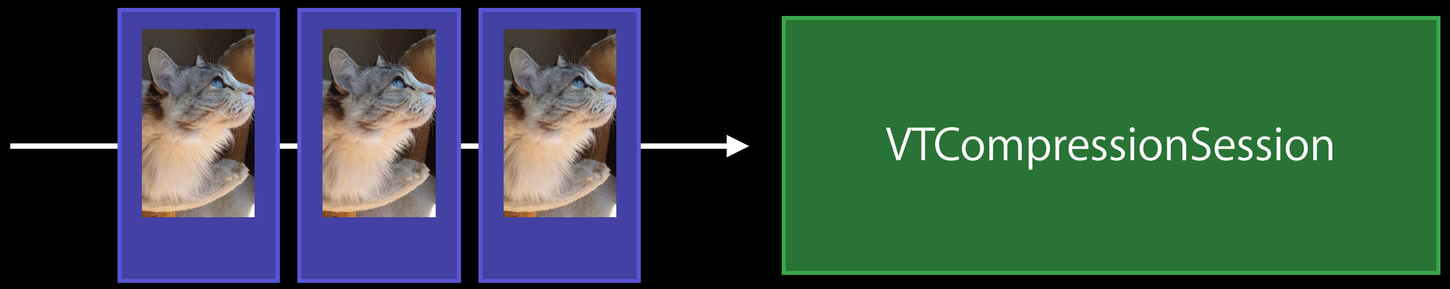
# Configuring VTCompressionSession

Configure compression using VTSessionSetProperty() calls

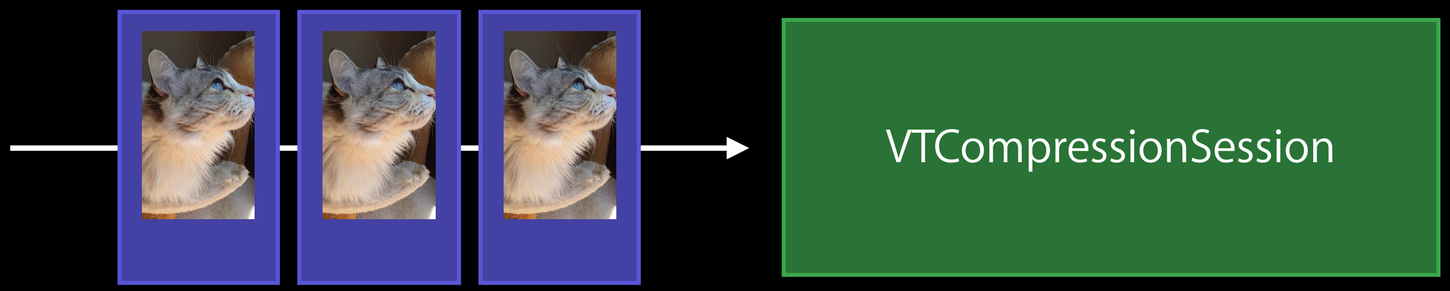
```
kVTCompressionPropertyKey_AllowFrameReordering
kVTCompressionPropertyKey_AverageBitRate
kVTCompressionPropertyKey_H264EntropyMode
    kVTH264EntropyMode_CAVLC/kVTH264EntropyMode_CABAC
kVTCompressionPropertyKey_RealTime
kVTCompressionPropertyKey_ProfileLevel
    for example: kVTProfileLevel_H264_Main_AutoLevel
```

...and many more

# Feeding VTCompressionSession

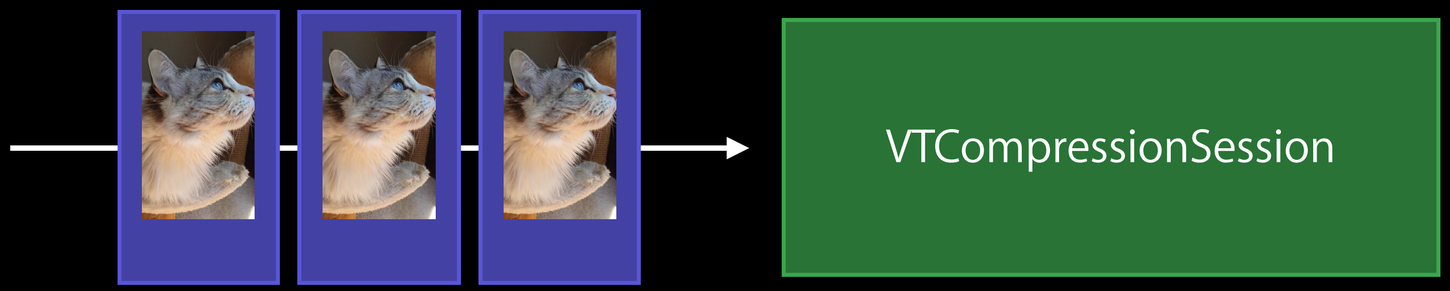


# Feeding VTCompressionSession




```
err = VTCompressionSessionEncodeFrame( session, pixelBuffer,
          presentationTime, … );
```

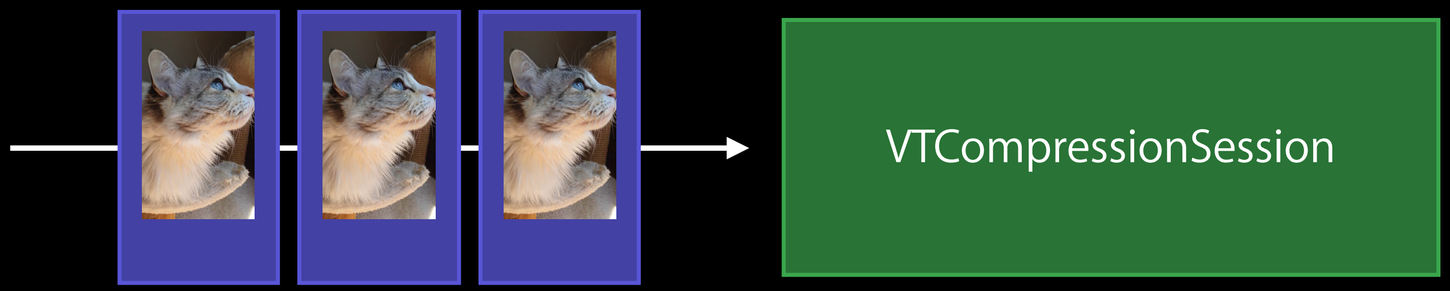
# Feeding VTCompressionSession




```
err = VTCompressionSessionEncodeFrame( session, pixelBuffer,
          presentationTime, … );
```

- Source frames as CVPixelBuffers with presentation time

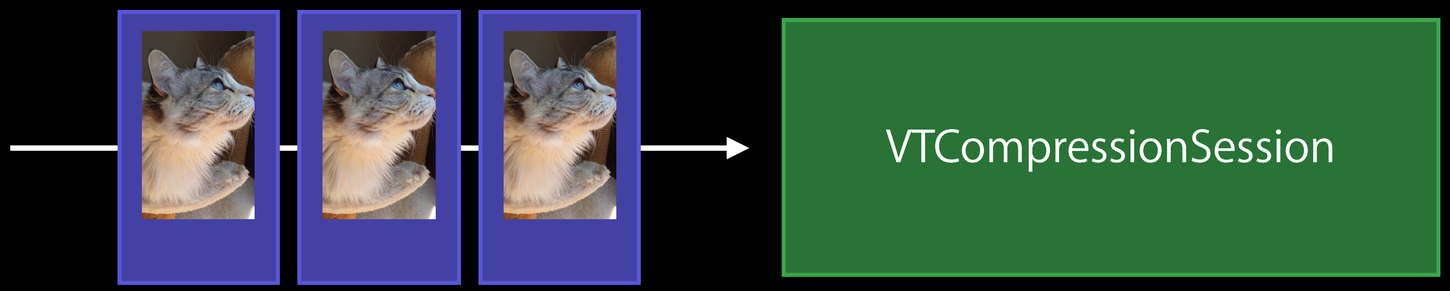
# Feeding VTCompressionSession




```
err = VTCompressionSessionEncodeFrame( session, pixelBuffer,
          presentationTime, … );
```

- Source frames as CVPixelBuffers with presentation time
- Presentation order

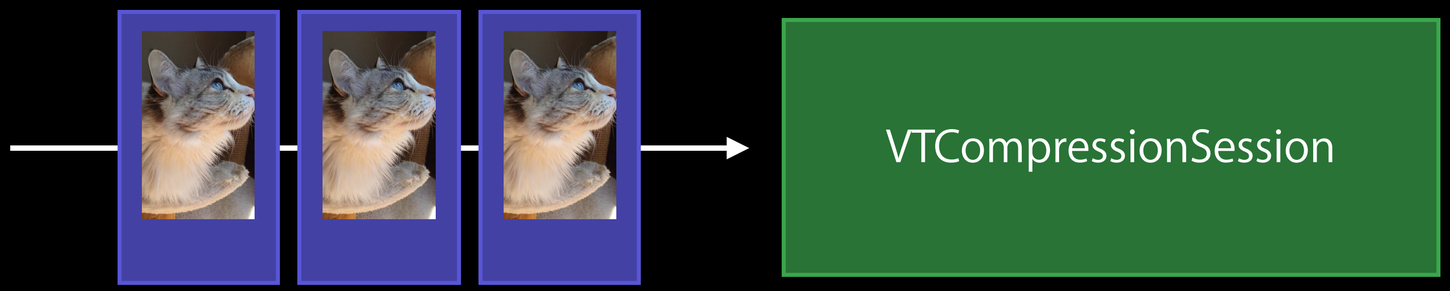
# Feeding VTCompressionSession




```
err = VTCompressionSessionEncodeFrame( session, pixelBuffer,
          presentationTime, … );
```

- Source frames as CVPixelBuffers with presentation time
- Presentation order
- Output may be delayed

# Feeding VTCompressionSession



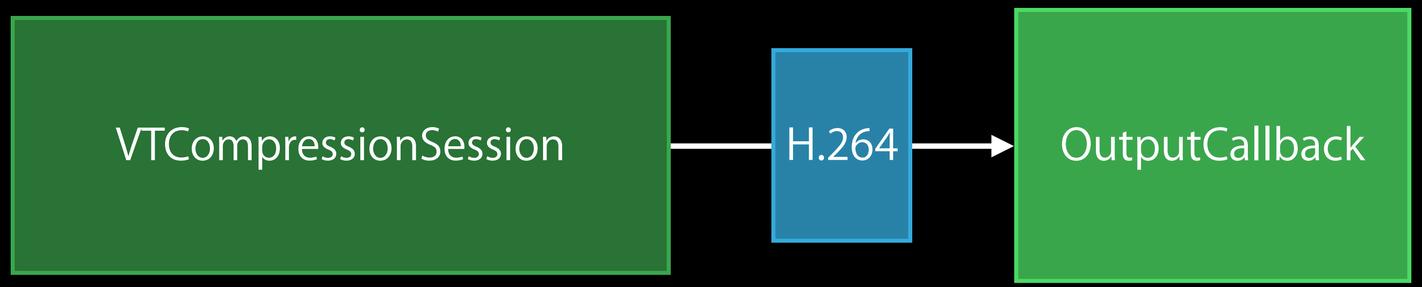

```
err = VTCompressionSessionEncodeFrame( session, pixelBuffer,
          presentationTime, … );
```

- Source frames as CVPixelBuffers with presentation time
- Presentation order
- Output may be delayed
- Use `VTCompressionSessionCompleteFrames()` to finish pending frames

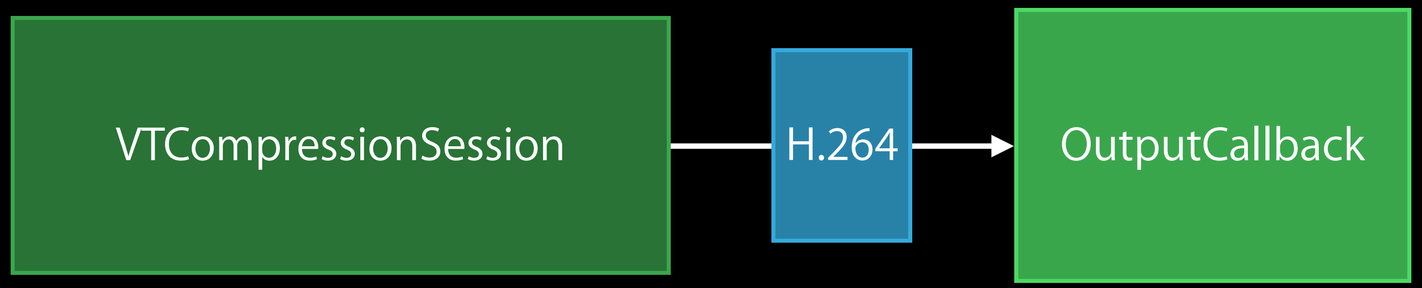
# VTCompressionOutputCallback

VTCompressionSession

# VTCompressionOutputCallback

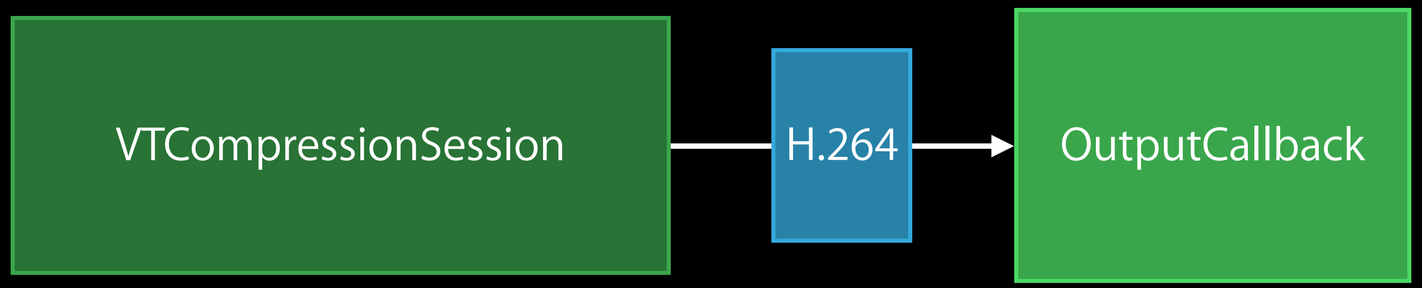


# VTCompressionOutputCallback




VTCompressionOutputCallback:

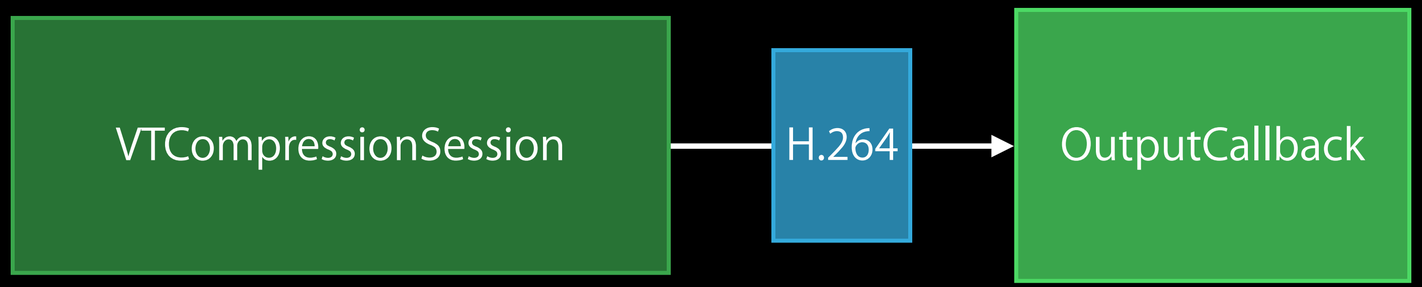
# VTCompressionOutputCallback




VTCompressionOutputCallback:

- Output CMSampleBuffer

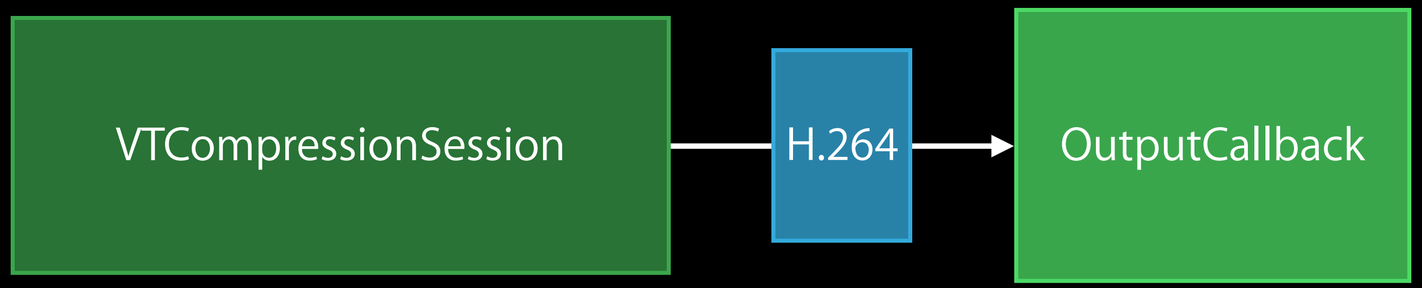
# VTCompressionOutputCallback




VTCompressionOutputCallback:

- Output CMSampleBuffer
- Compression error codes

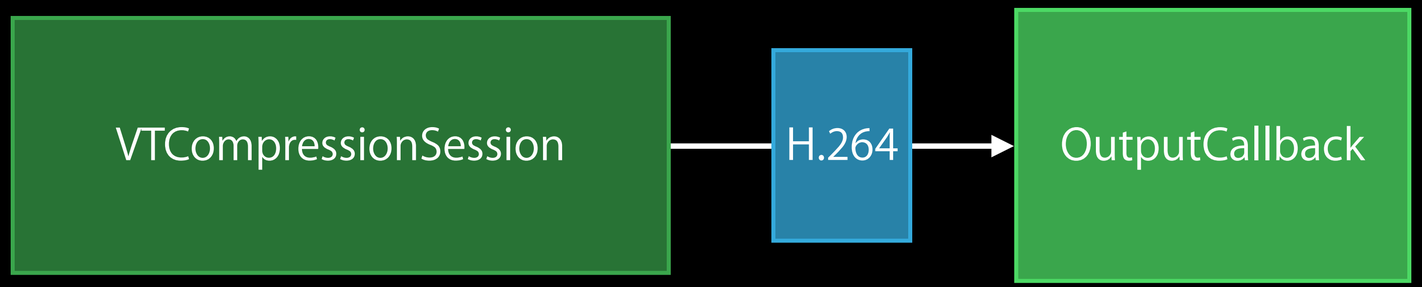
# VTCompressionOutputCallback




VTCompressionOutputCallback:

- Output CMSampleBuffer
- Compression error codes
- Dropped frames

# VTCompressionOutputCallback

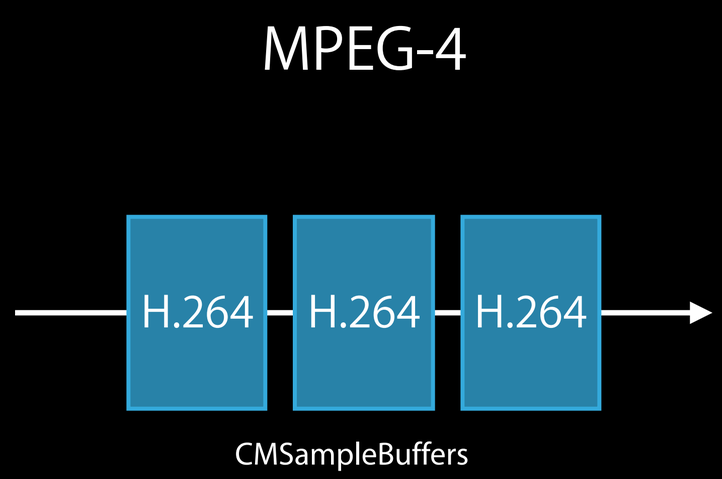



VTCompressionOutputCallback:

- Output CMSampleBuffer
- Compression error codes
- Dropped frames
- Frames emitted in decode order

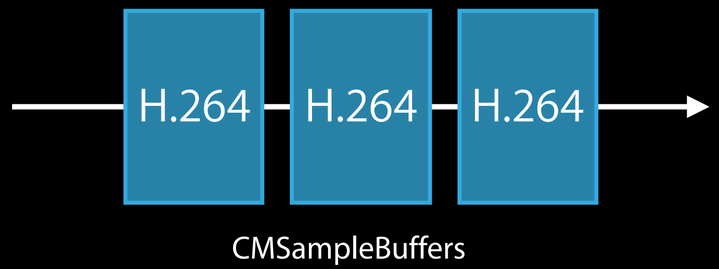
# CMSampleBuffers and Elementary Streams

# CMSampleBuffers and Elementary Streams



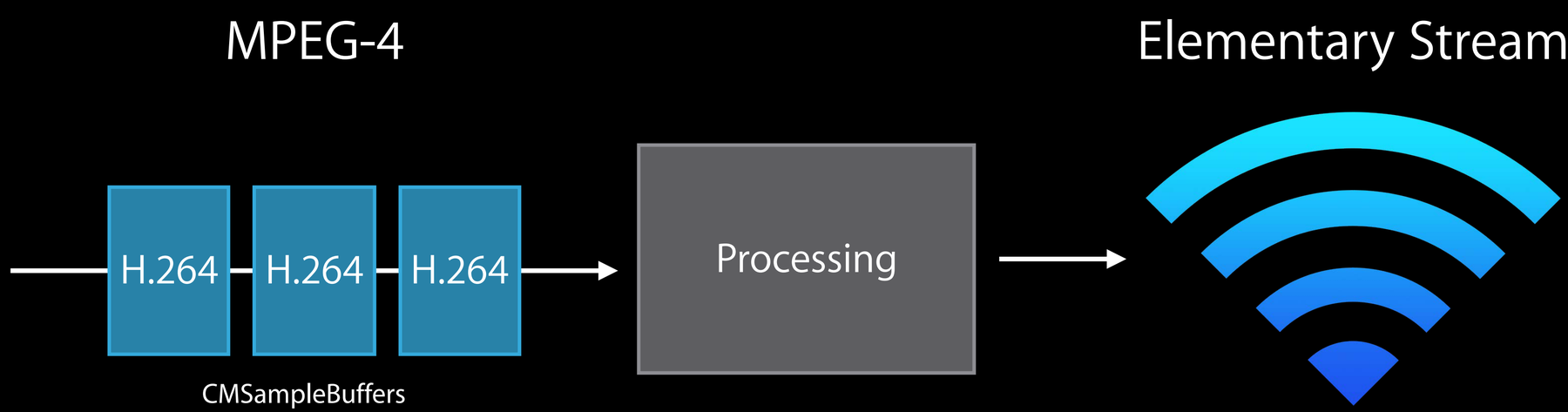
# CMSampleBuffers and Elementary Streams

MPEG-4




Elementary Stream

# CMSampleBuffers and Elementary Streams



# H.264 Syntax

## Conversion of parameter sets

MPEG-4

Elementary Stream

# H.264 Syntax

## Conversion of parameter sets

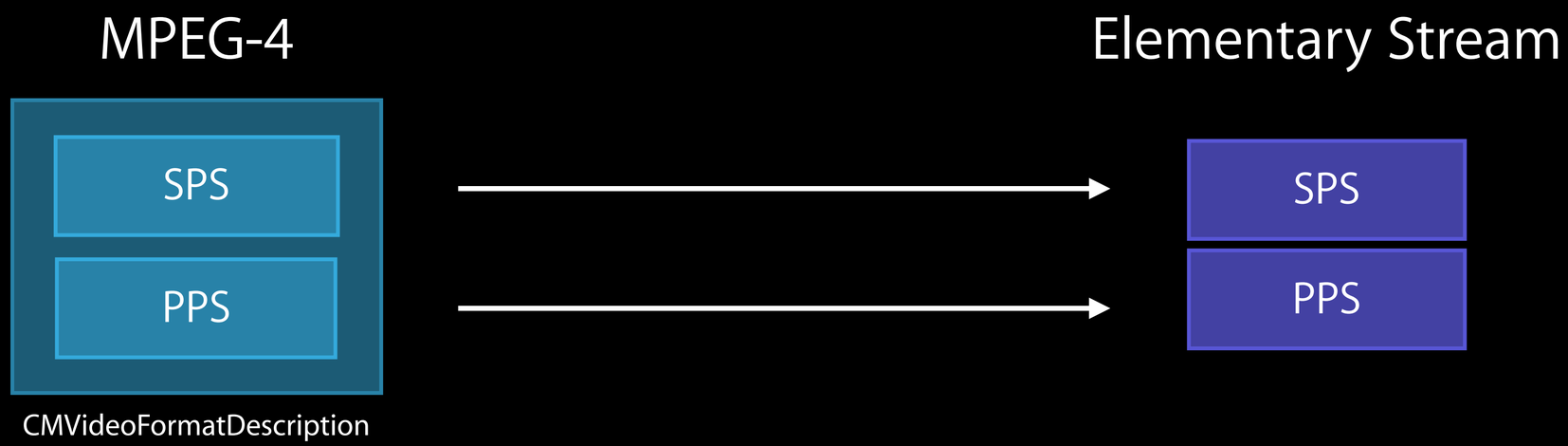
MPEG-4


CMVideoFormatDescription

Elementary Stream

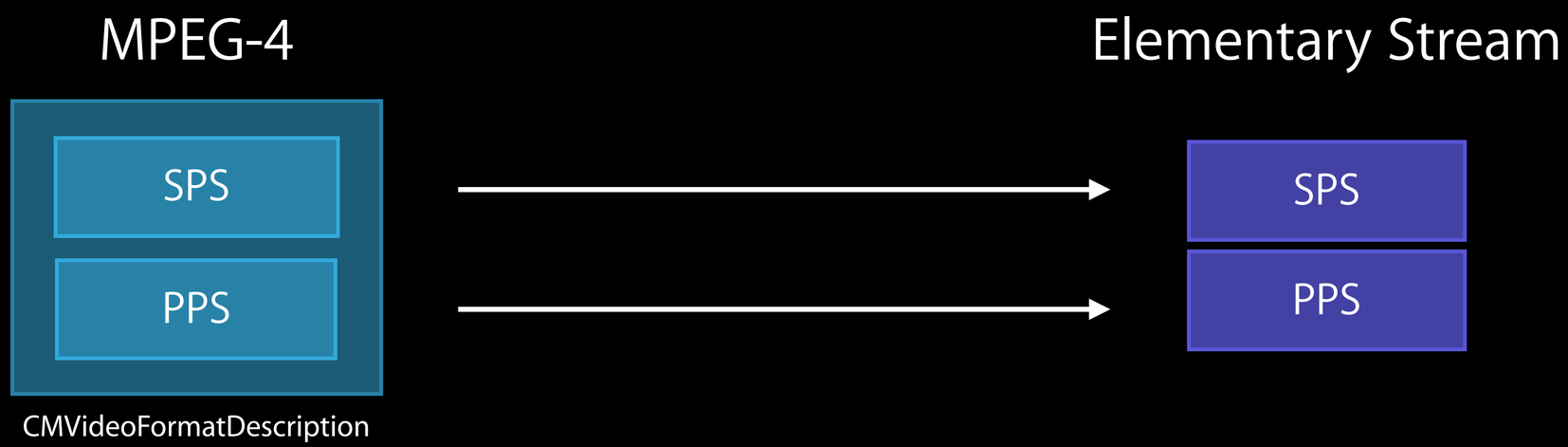
# H.264 Syntax

## Conversion of parameter sets



# H.264 Syntax

## Conversion of parameter sets




```
CMVideoFormatDescriptionGetH264ParameterSetAtIndex
```

# H.264 Syntax

## NAL Unit headers

MPEG-4

Elementary Stream

# H.264 Syntax

## NAL Unit headers

MPEG-4

Elementary Stream

00 00 80 00

I Frame
(32768 bytes)

4-Byte Header:
Length

# H.264 Syntax

## NAL Unit headers

MPEG-4

00 00 80 00

I Frame
(32768 bytes)

4-Byte Header:
Length

Elementary Stream

00 00 01

I Frame

3- or 4-Byte Header:
Start Code

# VTCompressionSession Summary

# VTCompressionSession Summary

Creation of VTCompressionSession

# VTCompressionSession Summary

Creation of VTCompressionSession

Configuring the compressor

# VTCompressionSession Summary

Creation of VTCompressionSession

Configuring the compressor

Providing CVPixelBuffers to VTCompressionSession

# VTCompressionSession Summary

Creation of VTCompressionSession

Configuring the compressor

Providing CVPixelBuffers to VTCompressionSession

Converting CMSampleBuffers into H.264 elementary stream packaging

# Multi-Pass Encoding

Erik Turnquist
Core Media Engineer

# Quality vs. Bit Rate

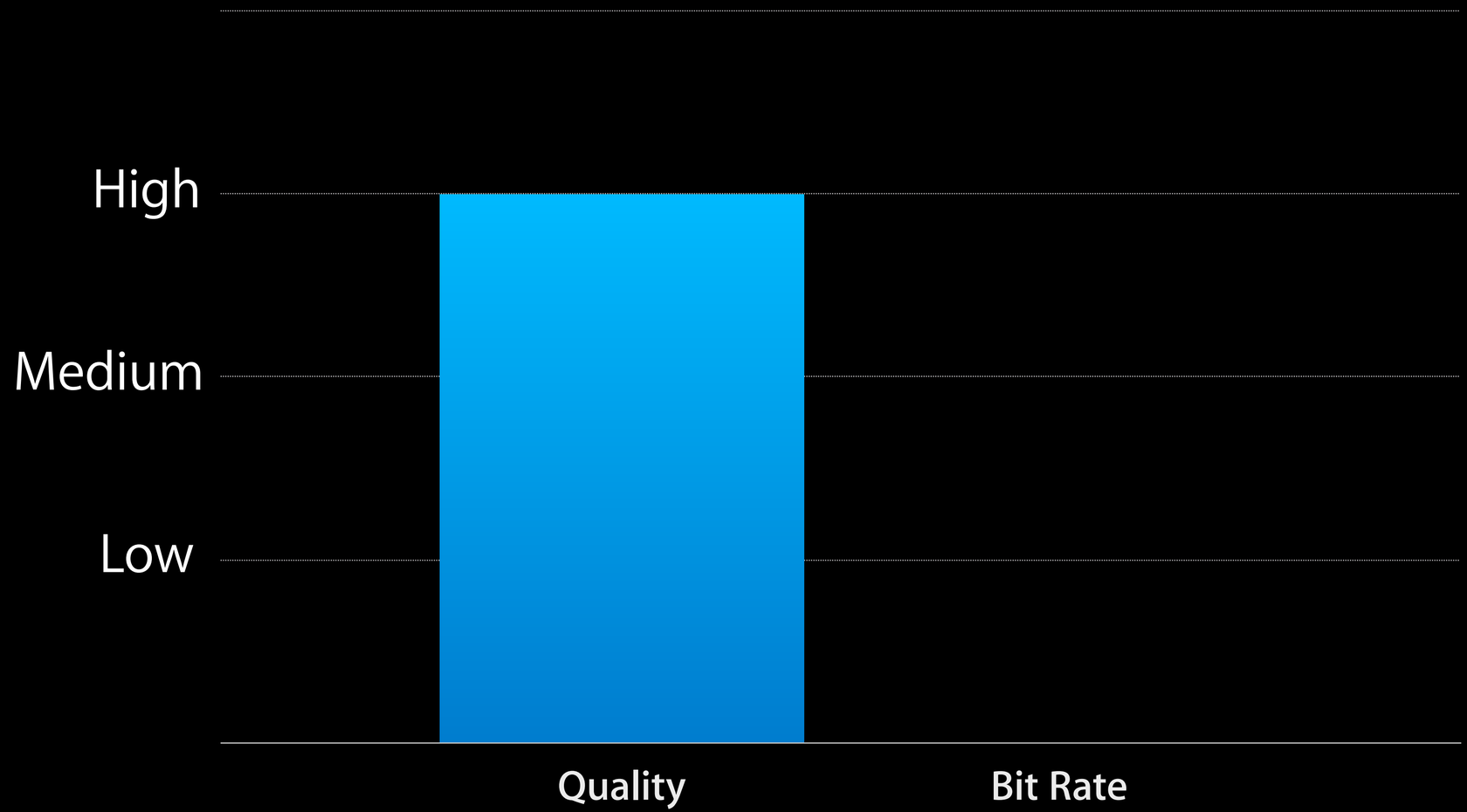
High

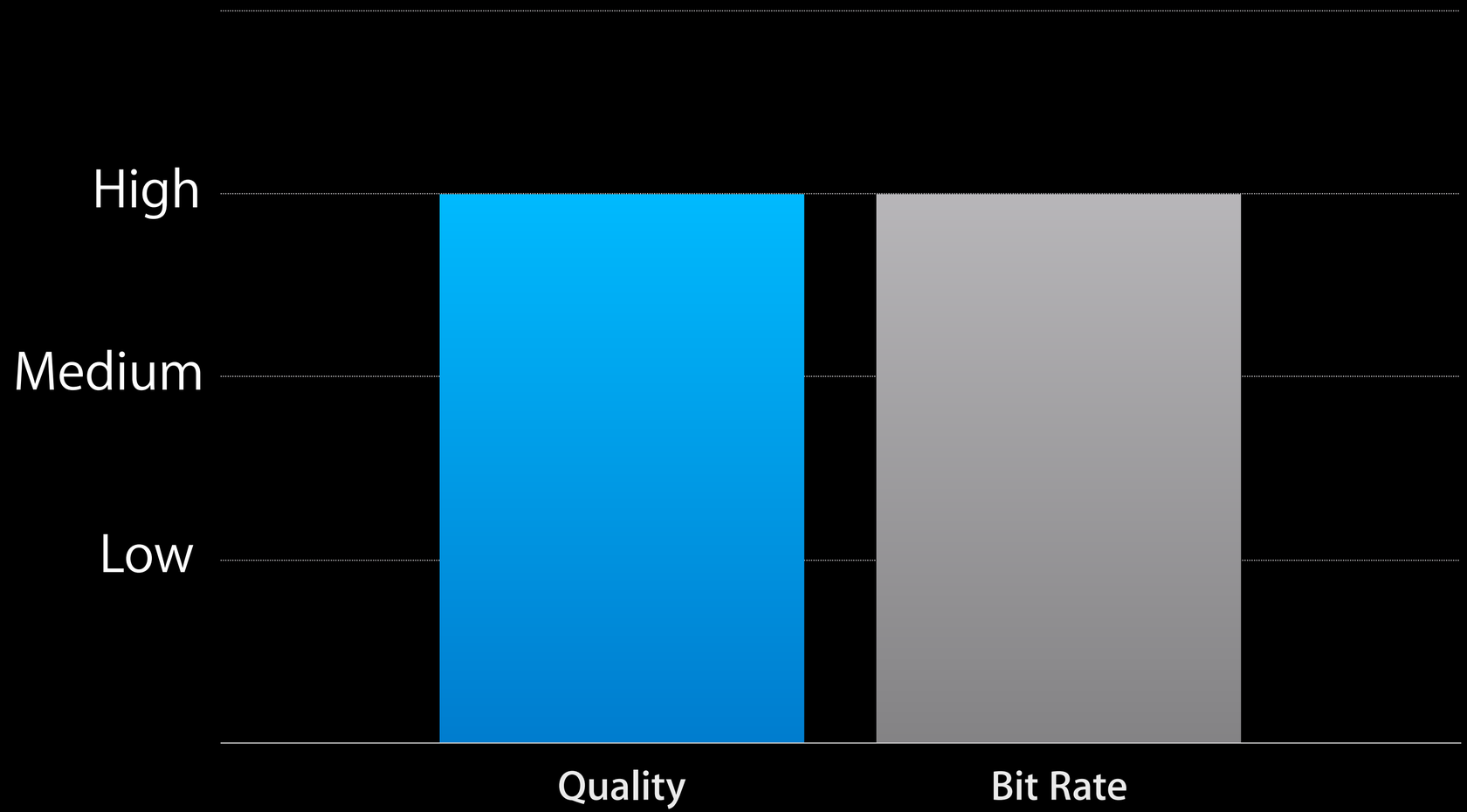
Medium

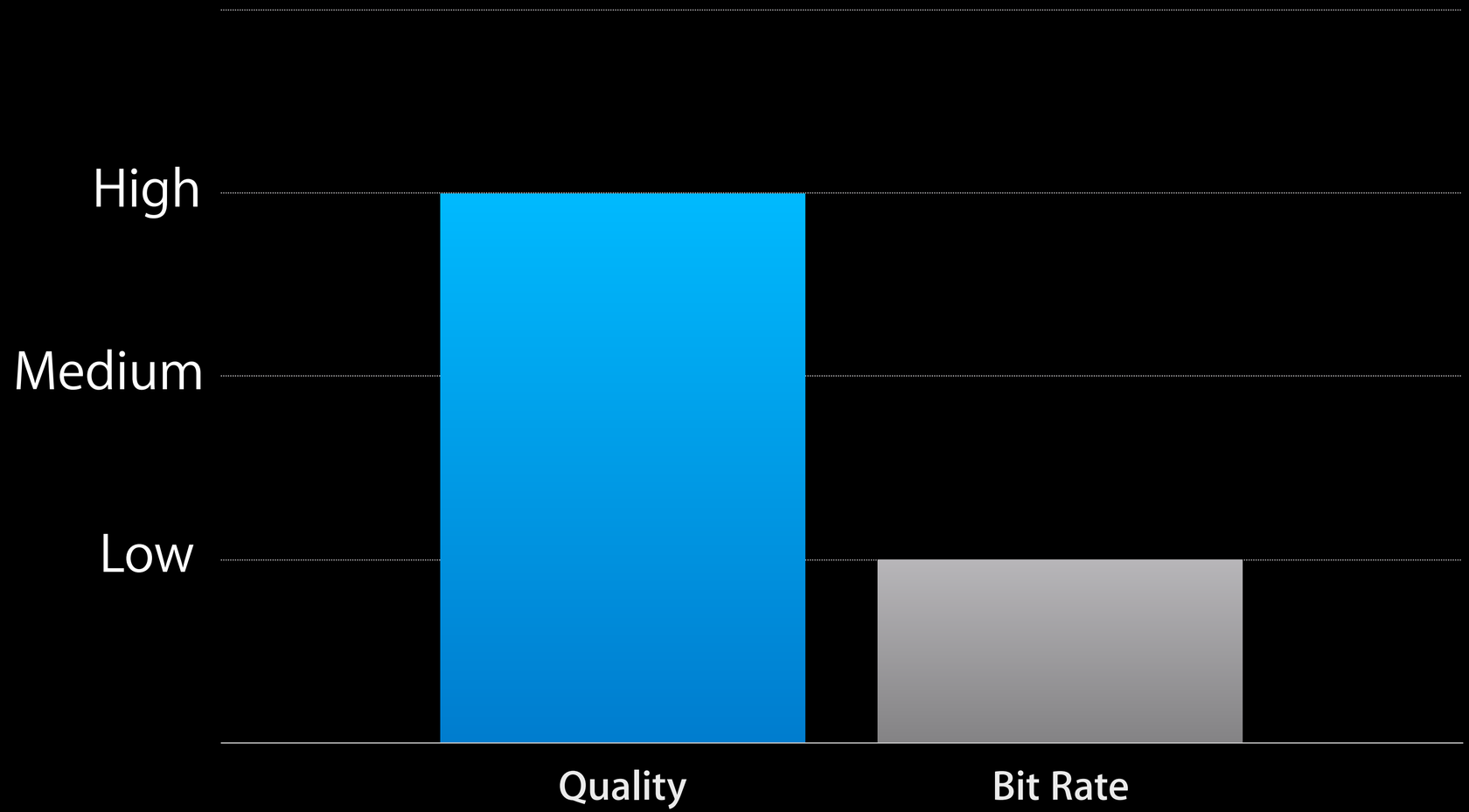
Low

Quality

Bit Rate


High
Medium
Low
Quality
Bit Rate

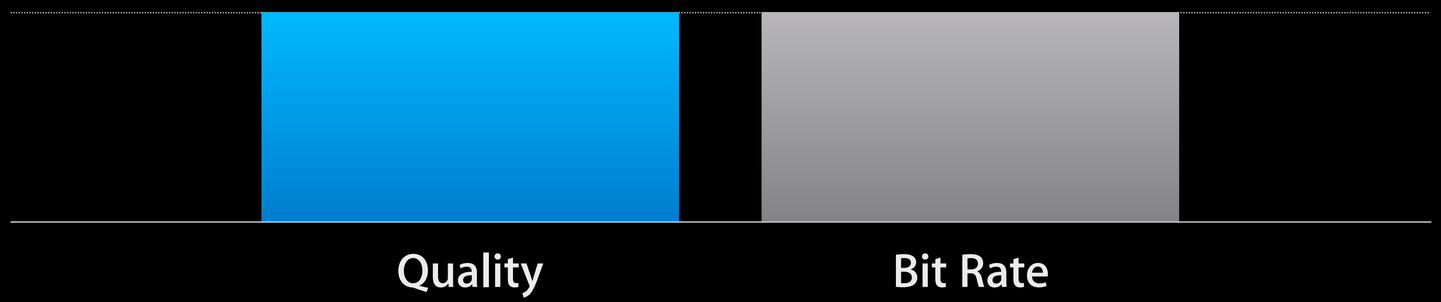

High
Medium
Low
Quality
Bit Rate

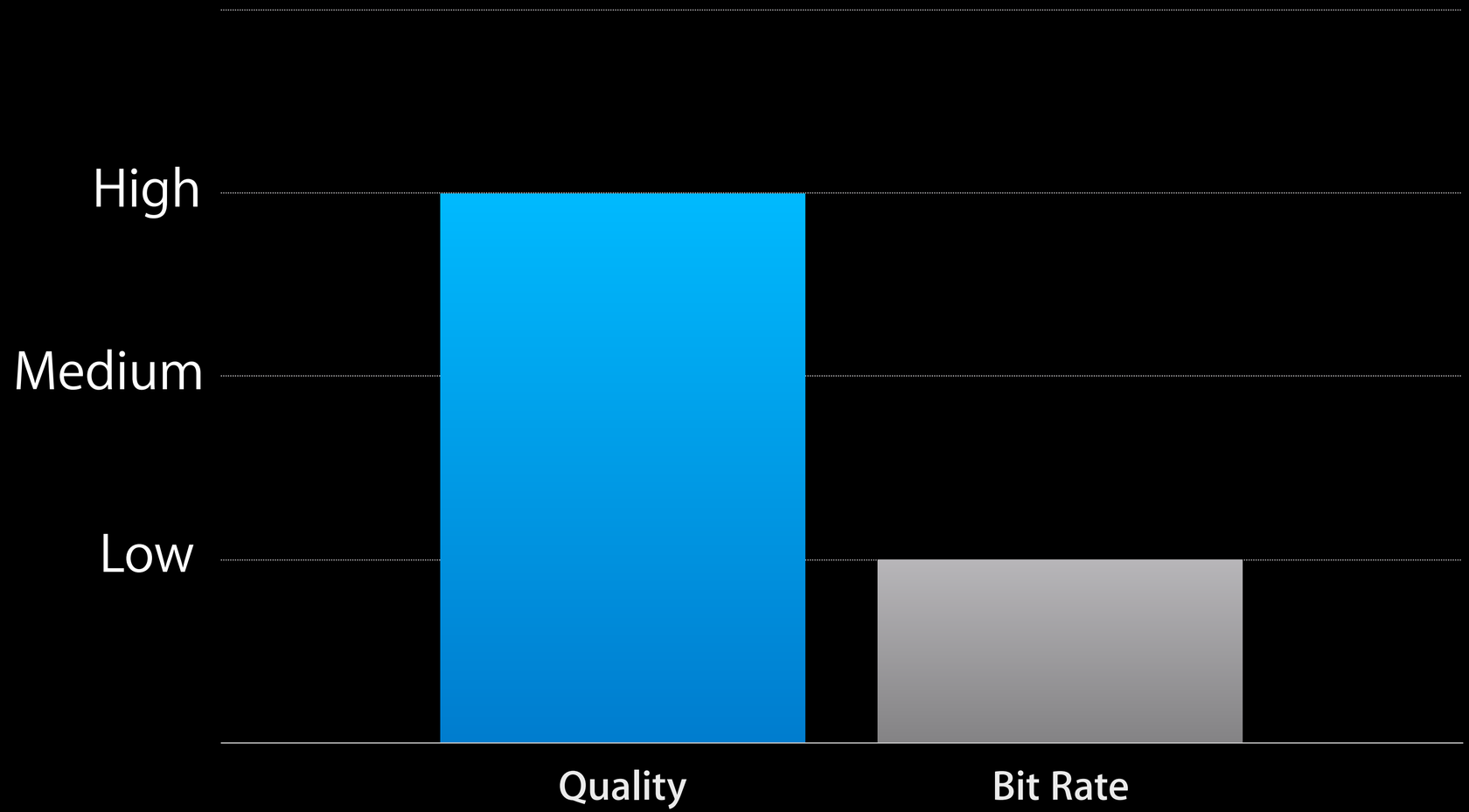

High
Medium
Low
Quality
Bit Rate

High

Medium

Low

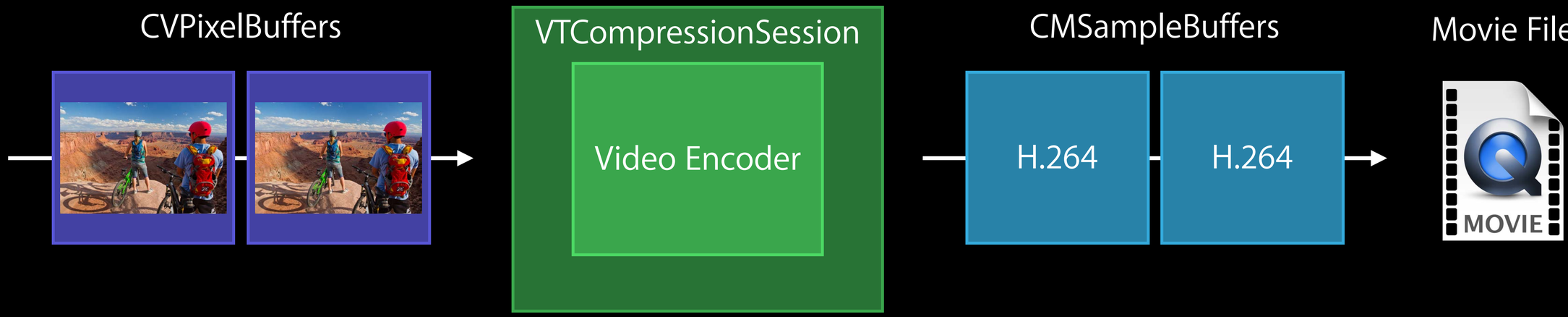



High
Medium
Low
Quality
Bit Rate

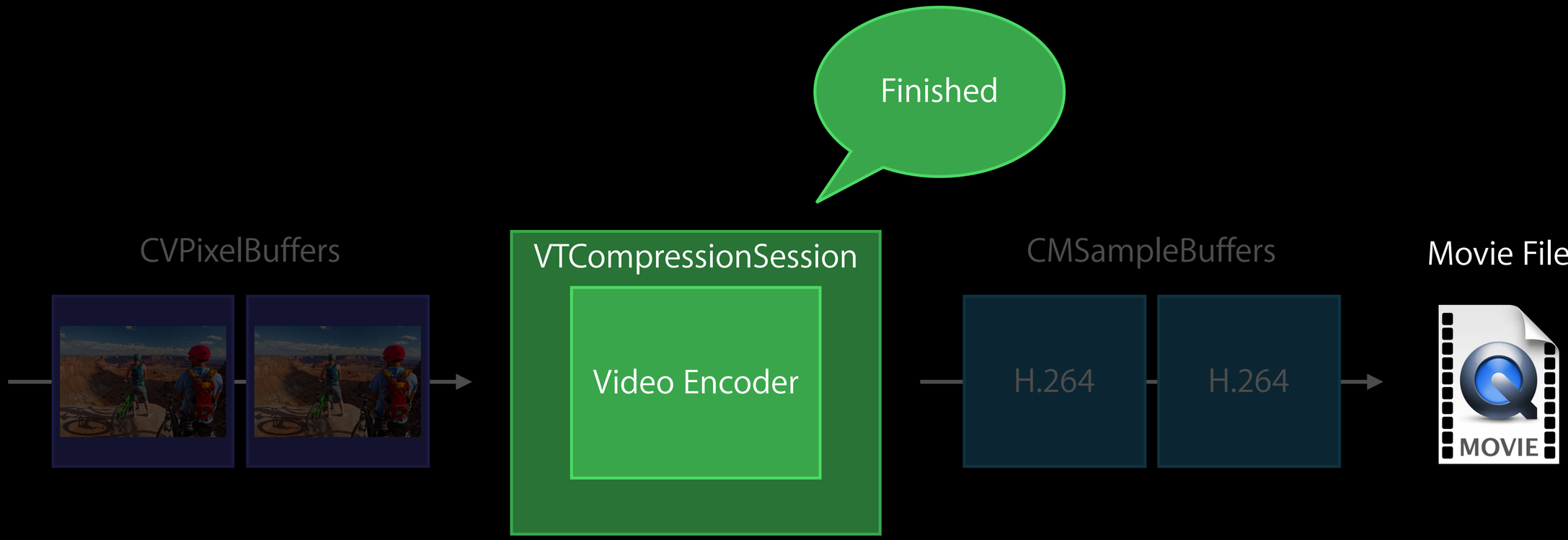
# What Is Multi-Pass Encoding?

# Single-Pass Encoding

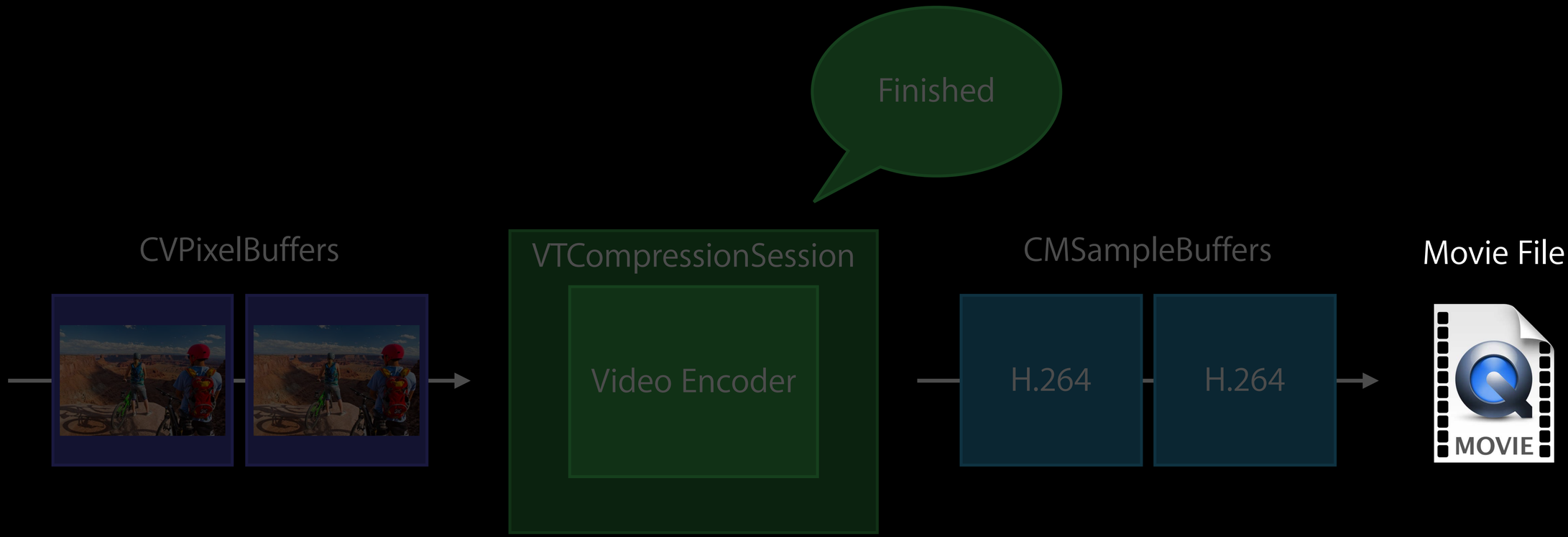
# Single-Pass Encoding



# Single-Pass Encoding

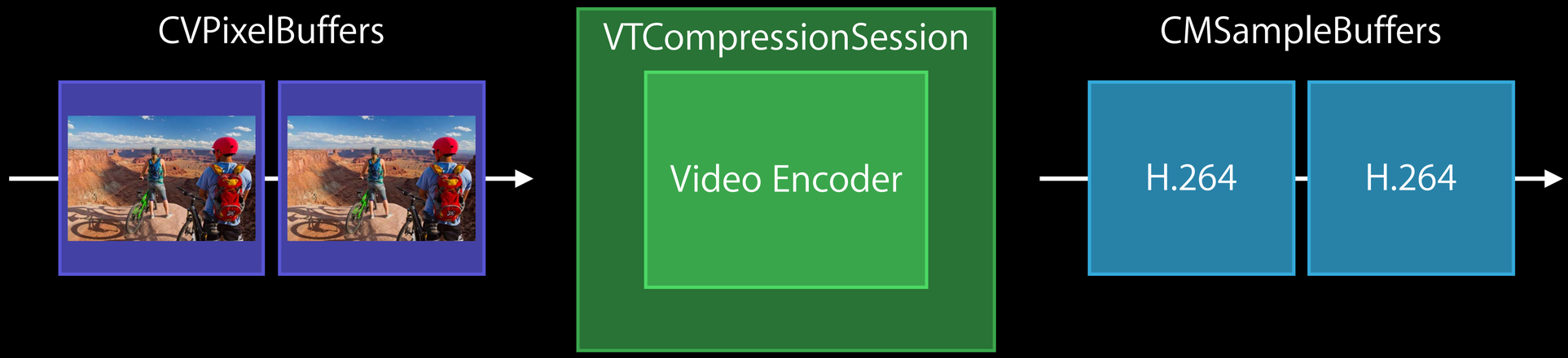


# Single-Pass Encoding

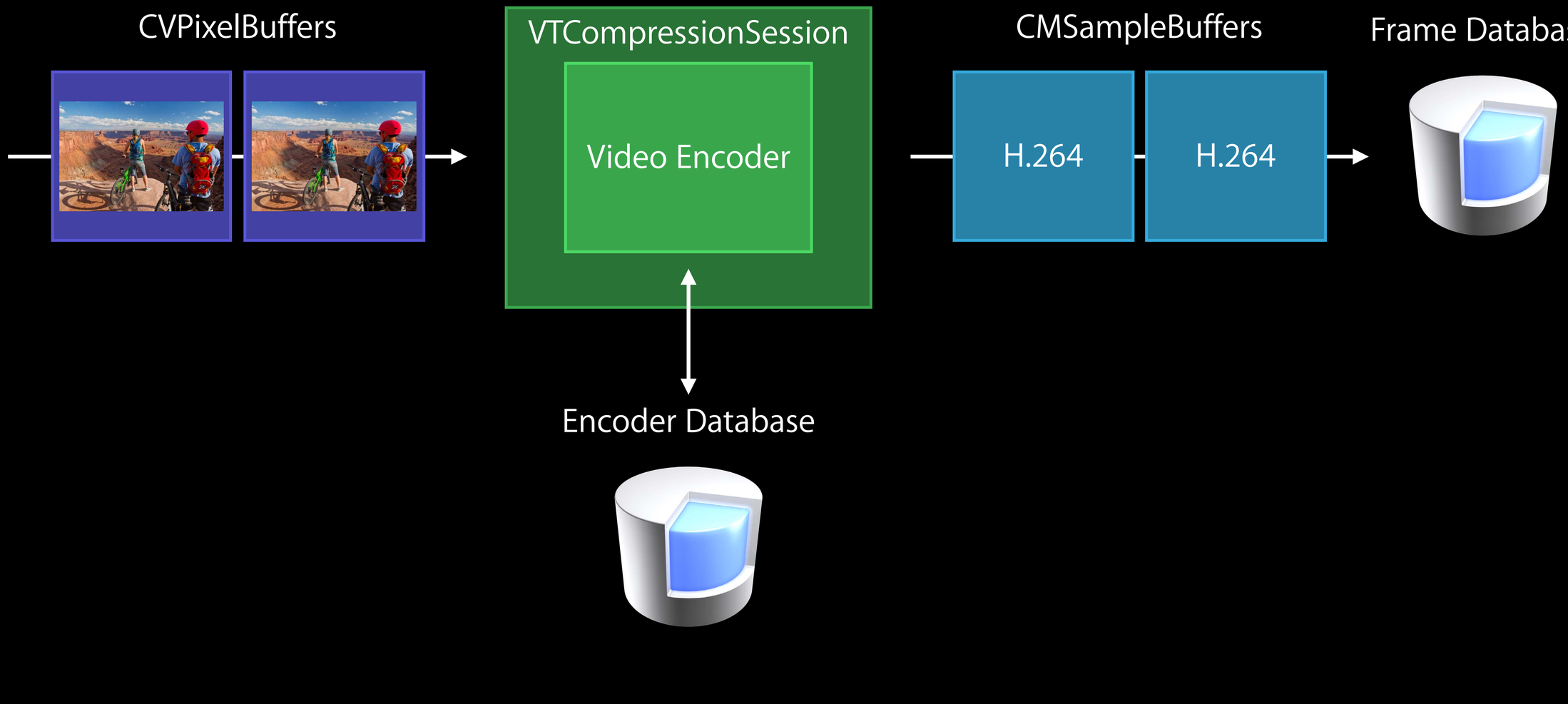


# Multi-Pass Encoding

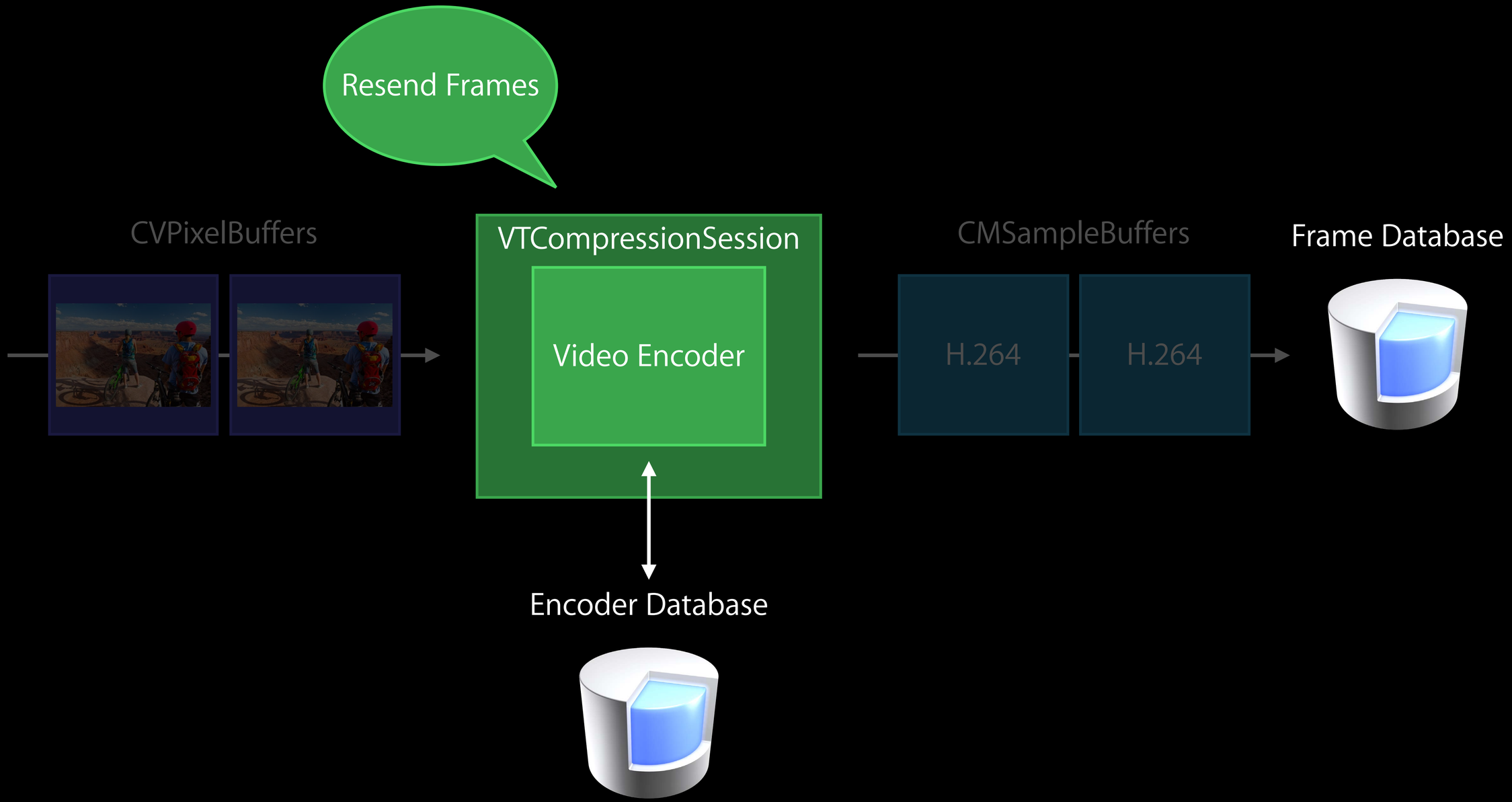
# Multi-Pass Encoding



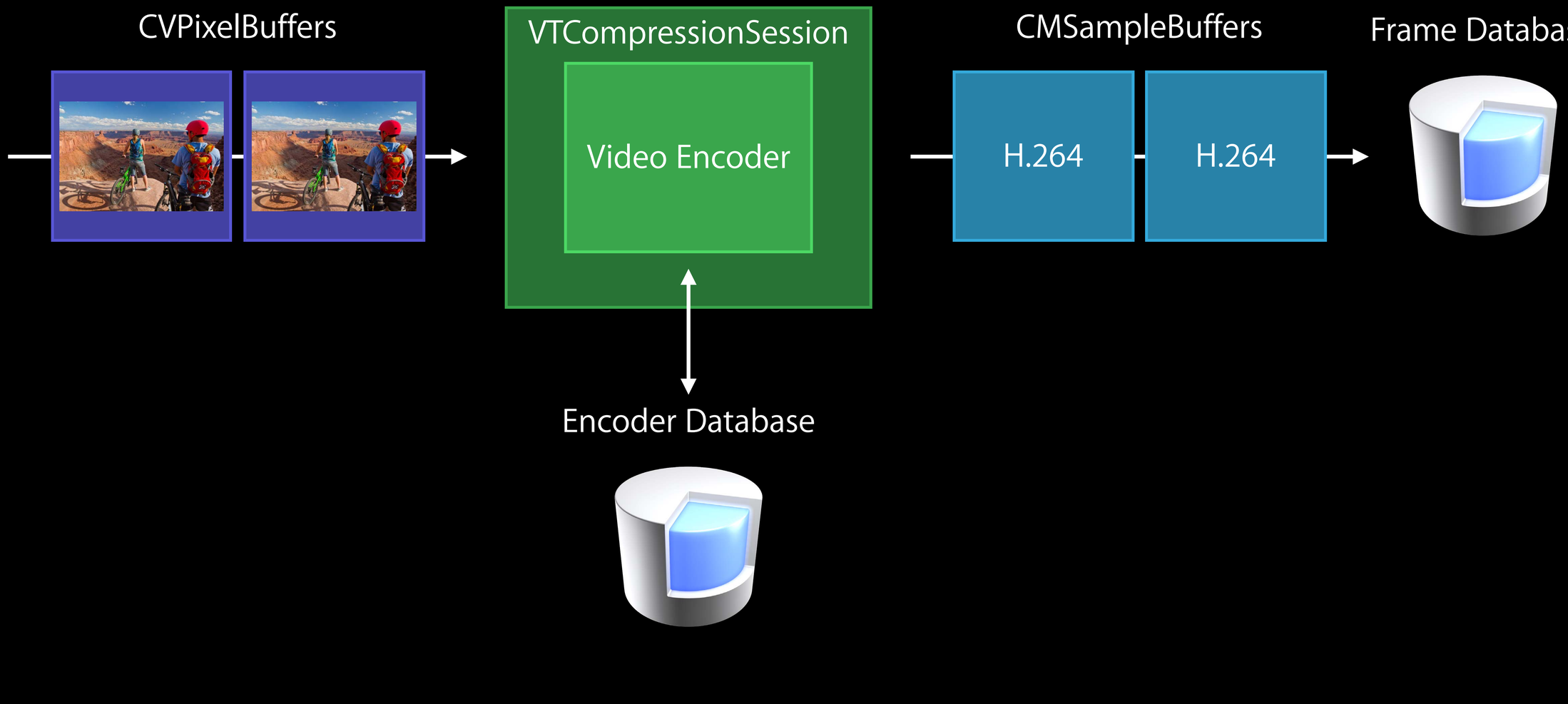
# Multi-Pass Encoding



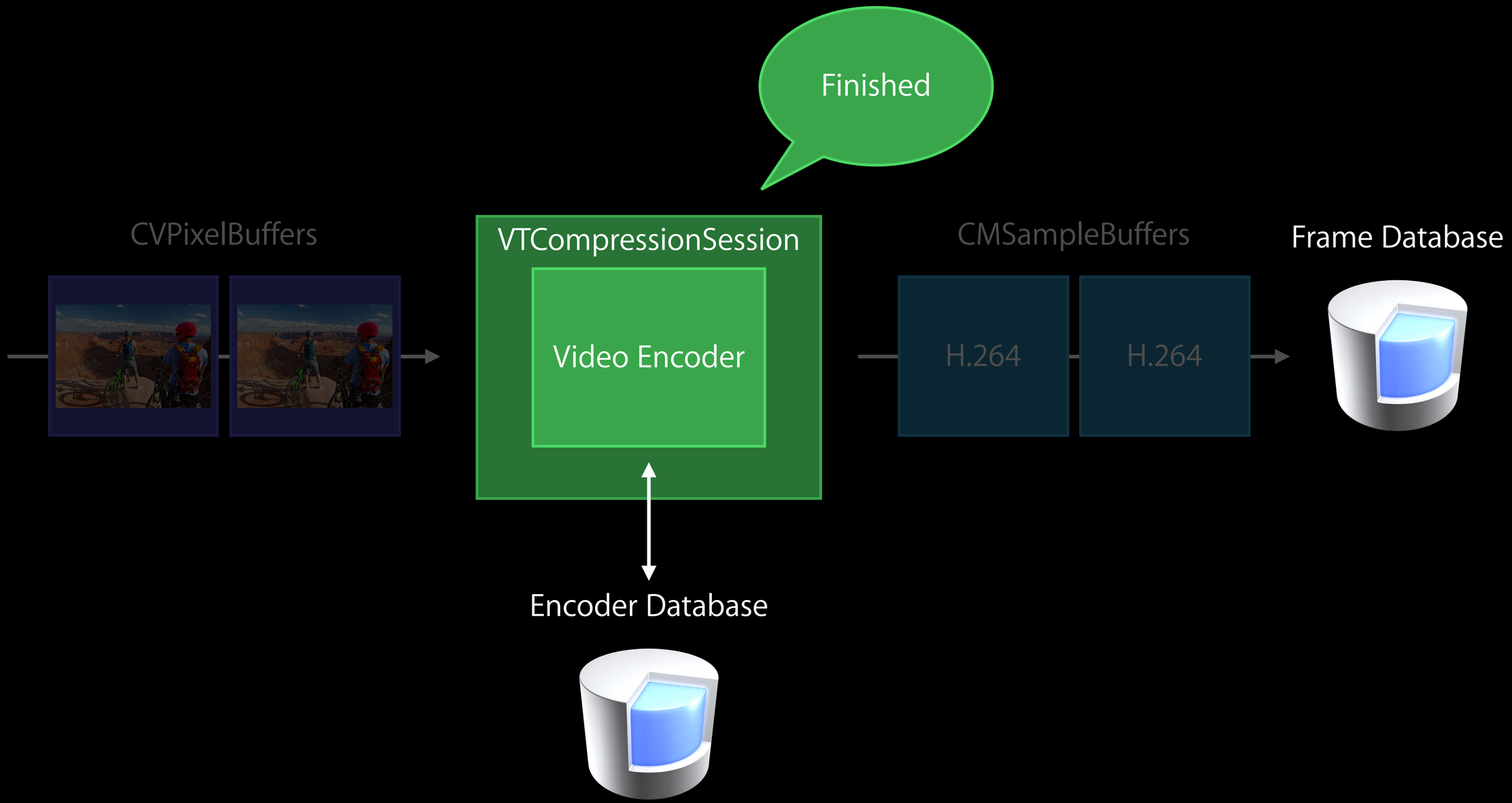
# Multi-Pass Encoding



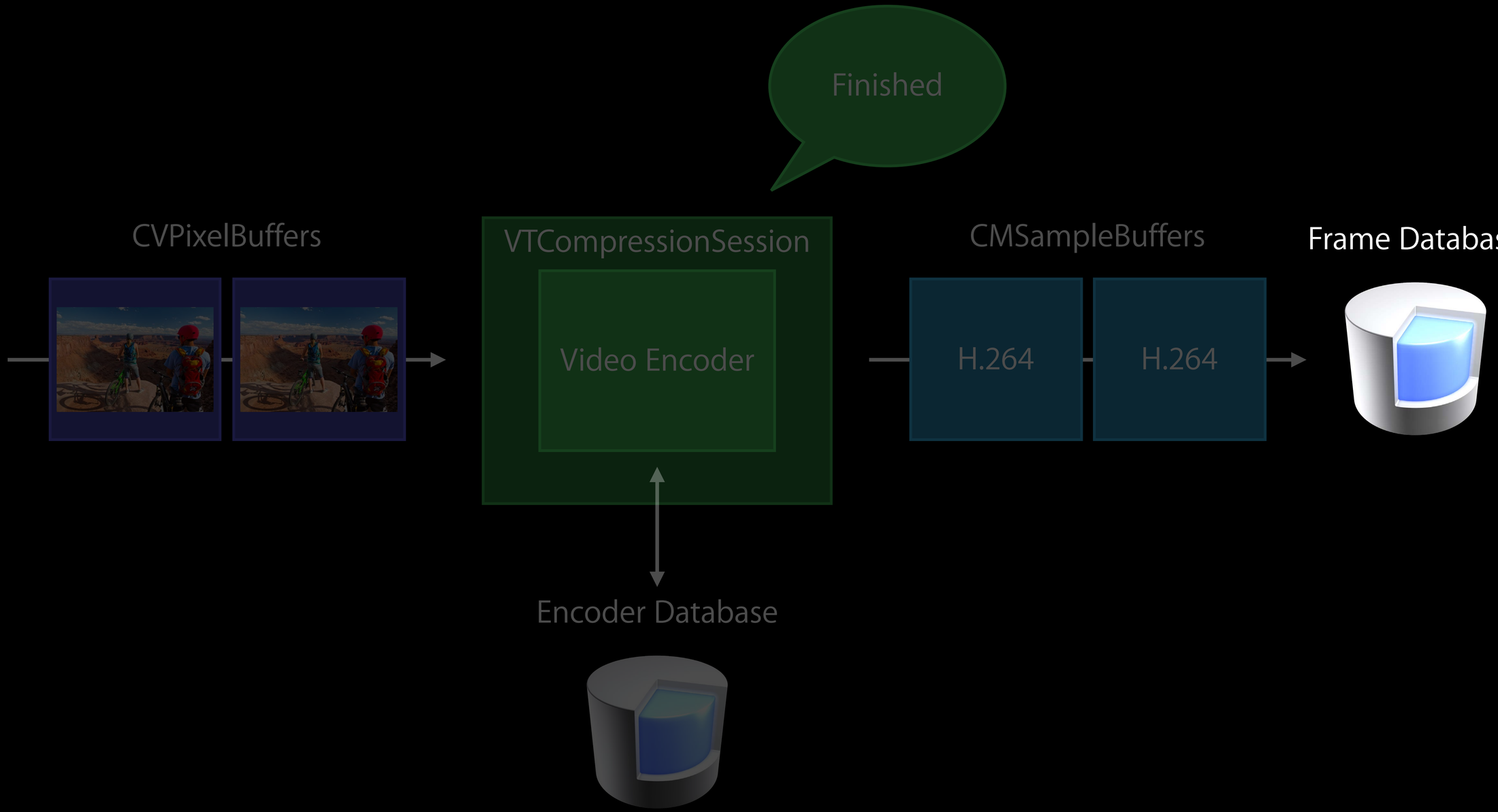
# Multi-Pass Encoding



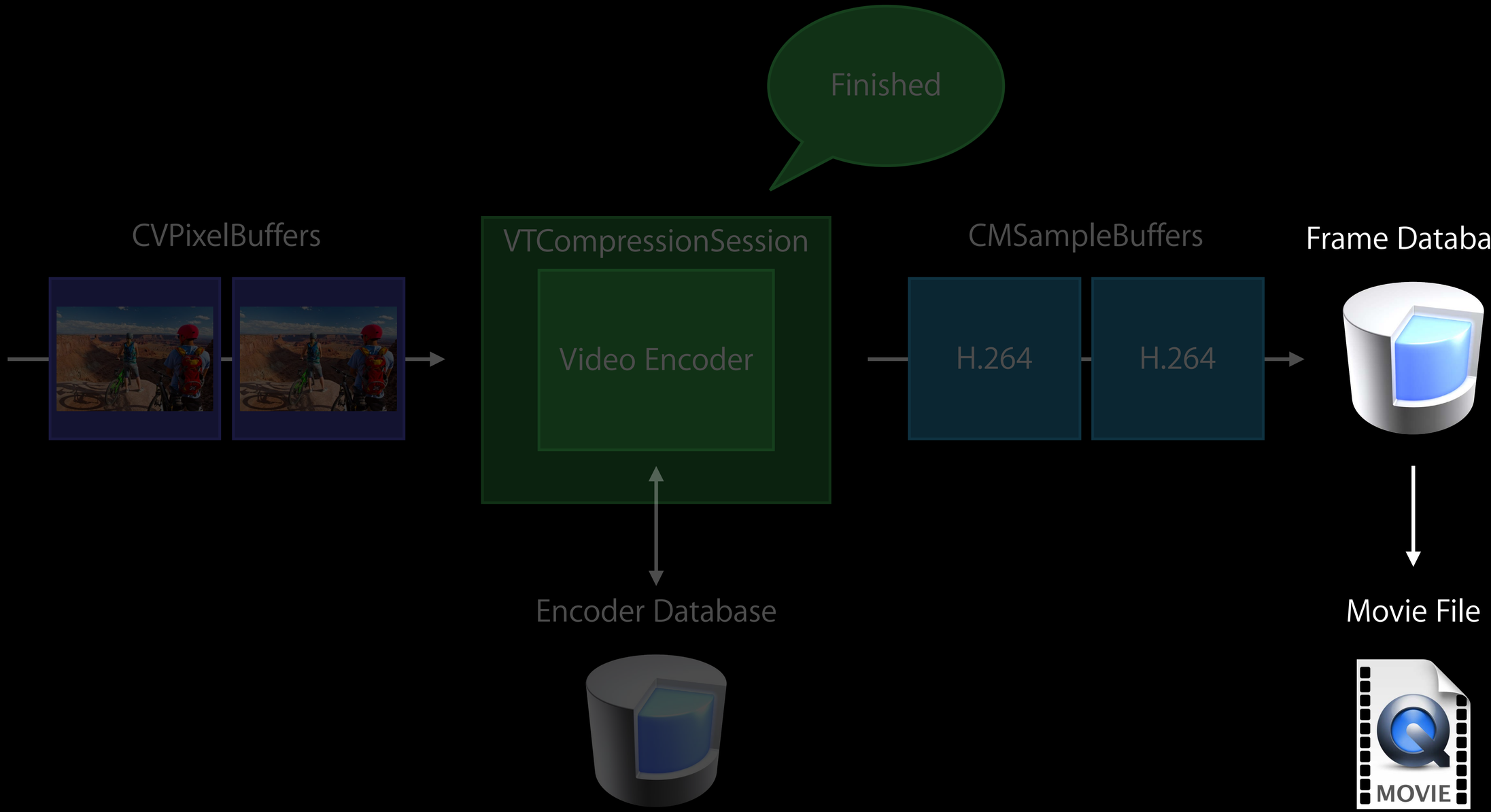
# Multi-Pass Encoding



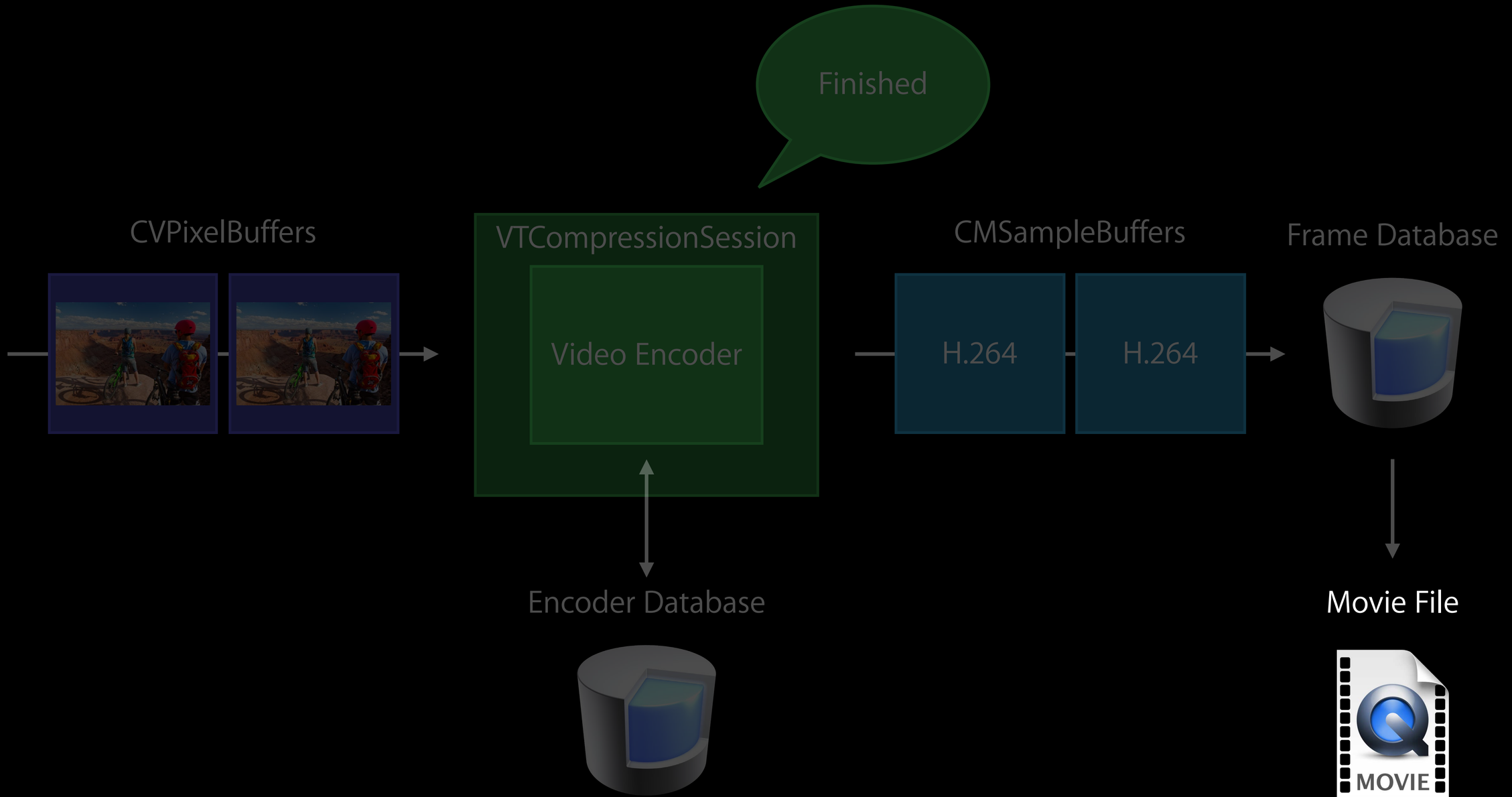
# Multi-Pass Encoding



# Multi-Pass Encoding



# Multi-Pass Encoding



# Multi-Pass Encoder Features

| | Single-Pass | Multi-Pass |
|---|---|---|

# Multi-Pass Encoder Features

| | Single-Pass | Multi-Pass |
|---|---|---|
| Hardware Accelerated | ✓ | ✓ |

# Multi-Pass Encoder Features

| | Single-Pass | Multi-Pass |
|---|---|---|
| Hardware Accelerated | ✓ | ✓ |
| Knowledge of Future | ✗ | ✓ |

# Multi-Pass Encoder Features

| | Single-Pass | Multi-Pass |
|---|---|---|
| Hardware Accelerated | ✓ | ✓ |
| Knowledge of Future | ✗ | ✓ |
| Change Decisions | ✗ | ✓ |

# Multi-Pass Encoder Features

| | Single-Pass | Multi-Pass |
|---|---|---|
| Hardware Accelerated | ✓ | ✓ |
| Knowledge of Future | ✗ | ✓ |
| Change Decisions | ✗ | ✓ |
| Optimal Quality per Bit | ✗ | ✓ |

# New APIs

AVKit

AVFoundation

Video Toolbox

Core Media

Core Video

# AVFoundation

New AVAssetExportSession property

Pass descriptions for AVAssetWriterInput

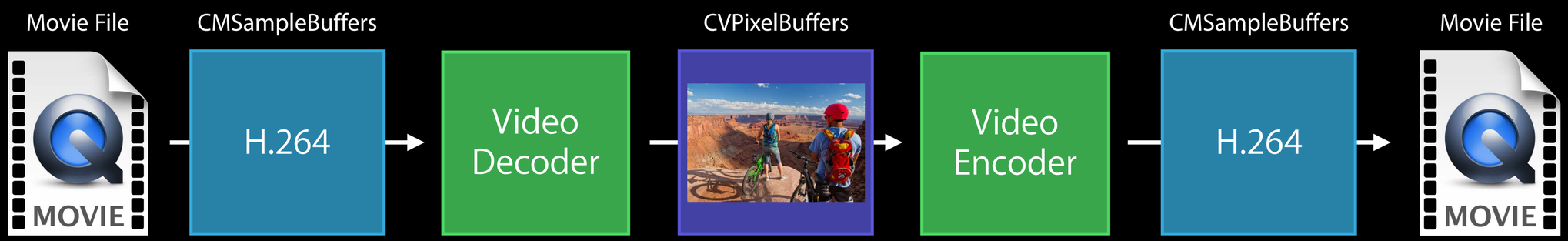
Reuse of AVAssetReaderOutput

# AVAssetExportSession

## Overview

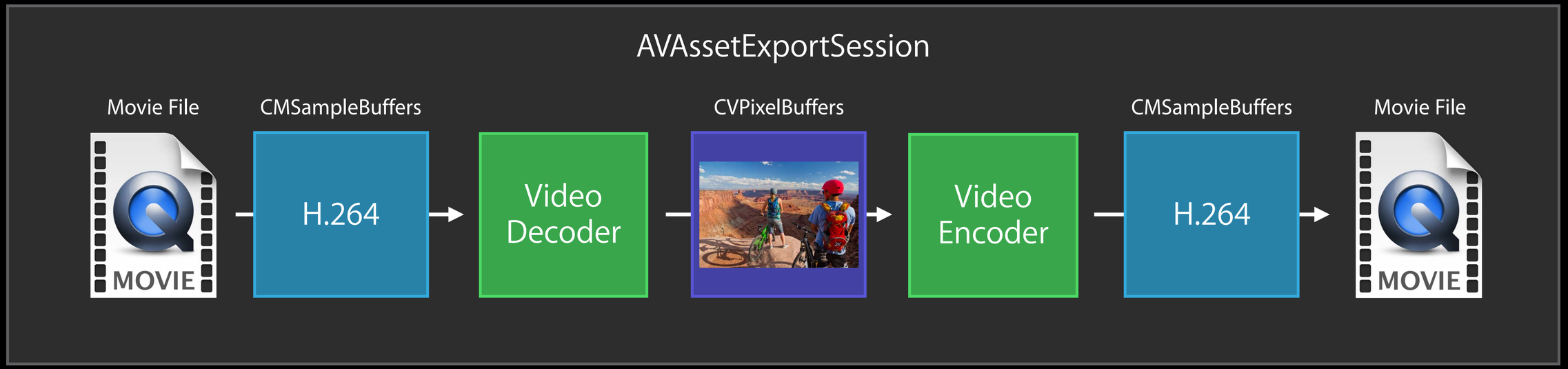
# AVAssetExportSession

## Overview



# AVAssetExportSession

## Overview



# AVAssetExportSession

## New APIs

# AVAssetExportSession

## New APIs


Multiple passes are taken care of automatically

# AVAssetExportSession

## New APIs


Multiple passes are taken care of automatically

- Falls back to single-pass if not supported

# AVAssetExportSession

## New APIs


Multiple passes are taken care of automatically

- Falls back to single-pass if not supported

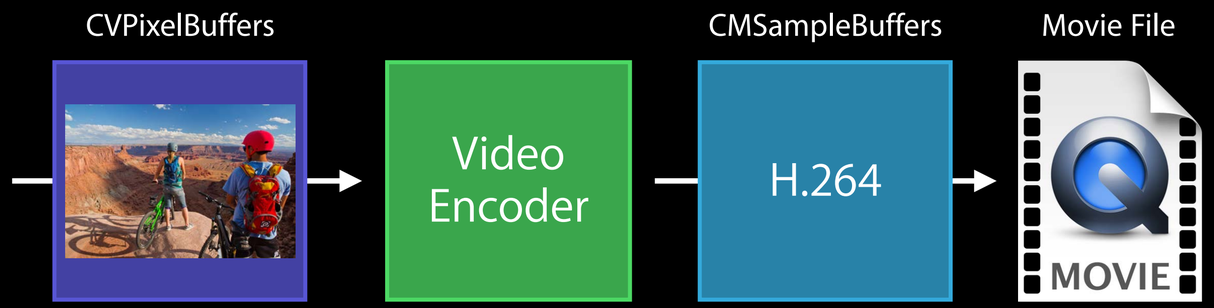
```
@property (nonatomic) BOOL canPerformMultiplePassesOverSourceMediaData;
```

# AVAssetWriter

## Overview

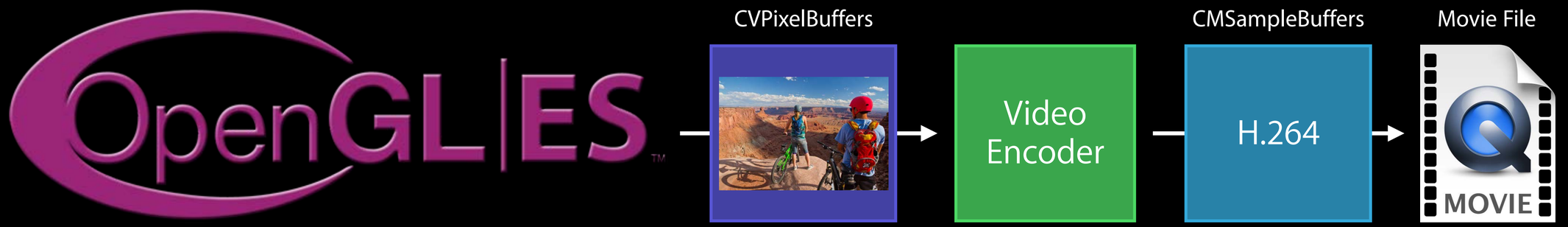
# AVAssetWriter

## Overview



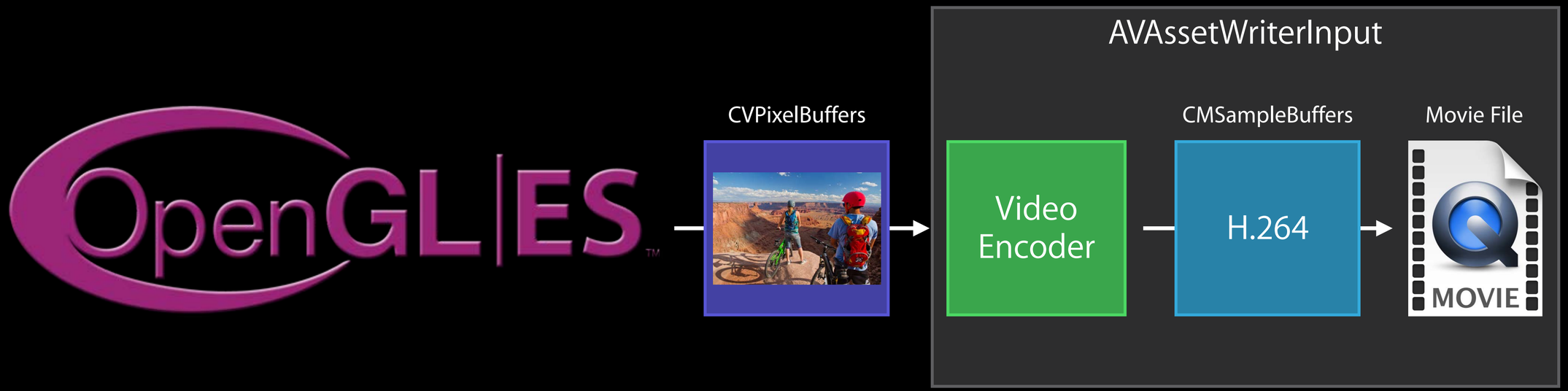
# AVAssetWriter

## Overview



# AVAssetWriter

## Overview

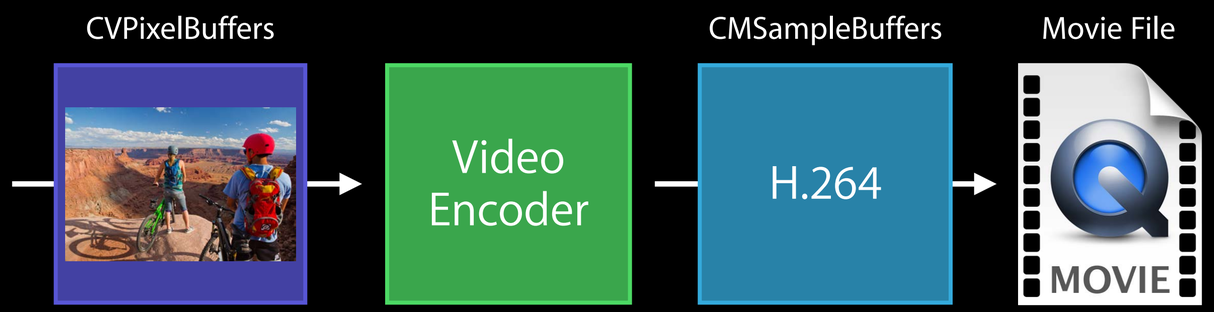


# AVAssetWriter

## Overview

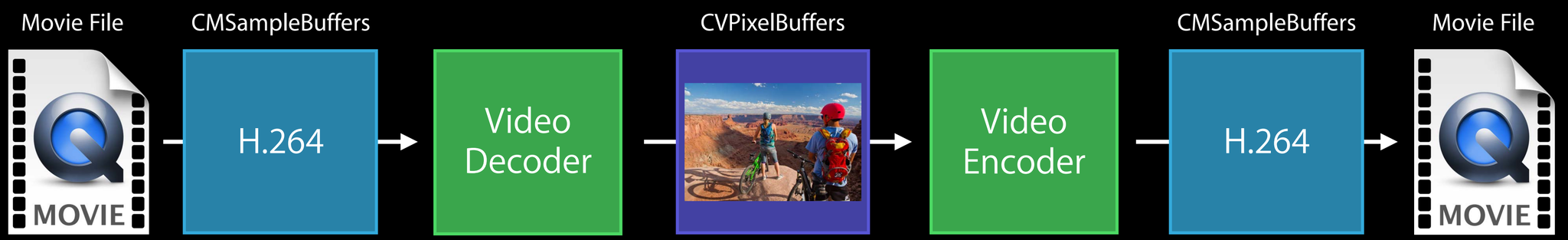
# AVAssetWriter

## Overview



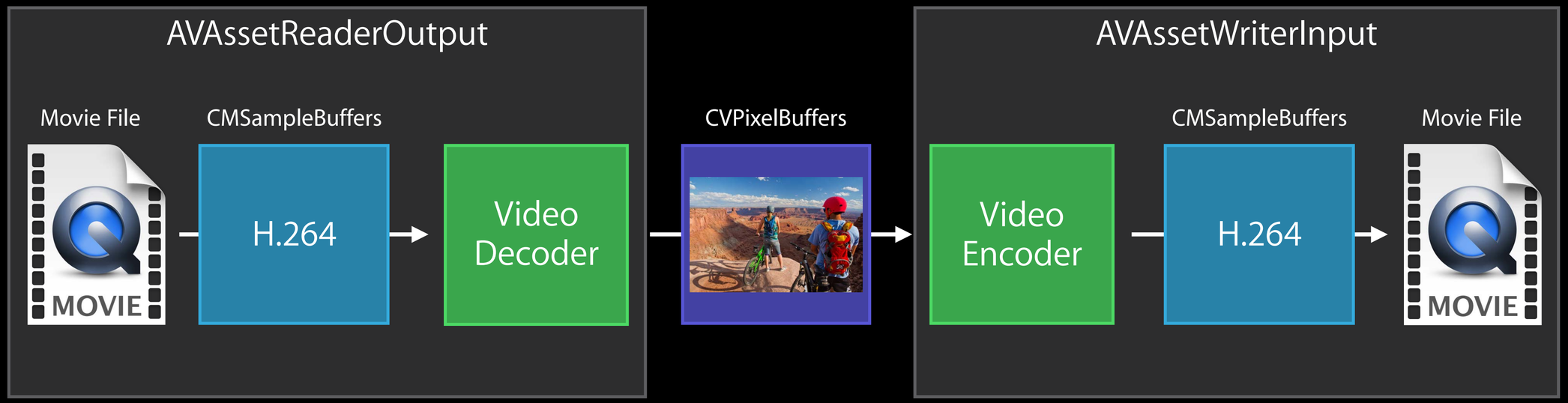
# AVAssetWriter

## Overview



# AVAssetWriter

## Overview



# AVAssetWriterInput

## New APIs

# AVAssetWriterInput

## New APIs


Enable multi-pass encoding if supported

```
@property (nonatomic) BOOL performsMultiPassEncodingIfSupported;
```

# AVAssetWriterInput

## New APIs


Enable multi-pass encoding if supported

```
@property (nonatomic) BOOL performsMultiPassEncodingIfSupported;
```

End current pass after appending samples

```
- (void)markCurrentPassAsFinished;
```

# AVAssetWriterInput

## New APIs


Enable multi-pass encoding if supported

```
@property (nonatomic) BOOL performsMultiPassEncodingIfSupported;
```

End current pass after appending samples

```
- (void)markCurrentPassAsFinished;
```

Triggers encoder analysis

# AVAssetWriterInput

## New APIs


Enable multi-pass encoding if supported

```
@property (nonatomic) BOOL performsMultiPassEncodingIfSupported;
```

End current pass after appending samples

```
- (void)markCurrentPassAsFinished;
```

Triggers encoder analysis

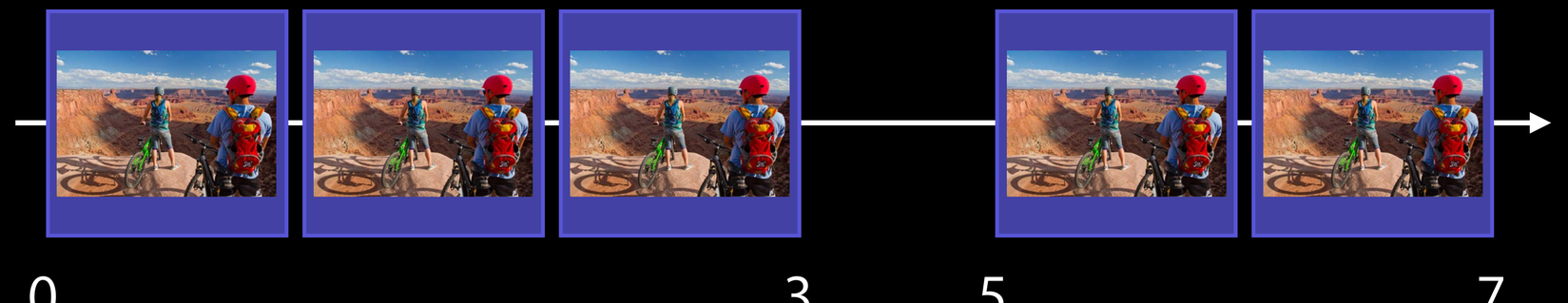
Encoder decides if it wants more passes and what time ranges

# AVAssetWriterInputPassDescription

NEW

# AVAssetWriterInputPassDescription

NEW

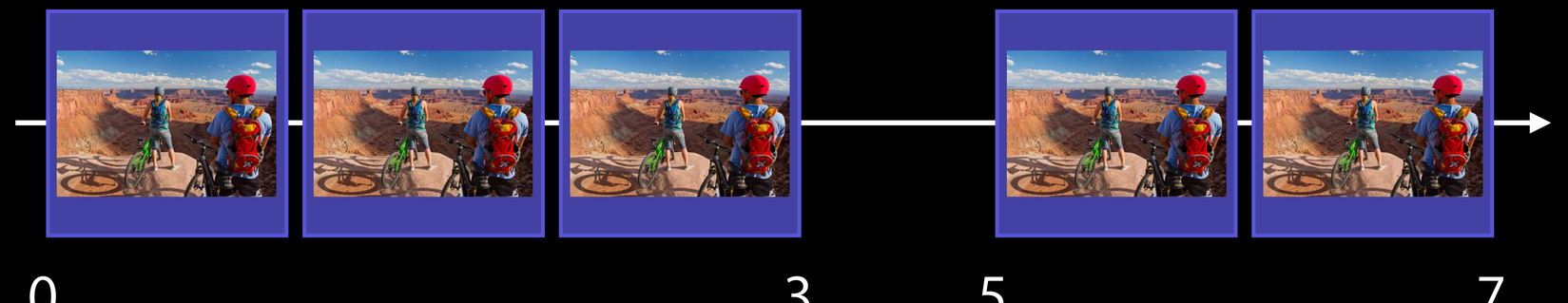



Encoder's request for samples in next pass

- May contain subsets of entire sequence

# AVAssetWriterInputPassDescription

NEW




Encoder's request for samples in next pass

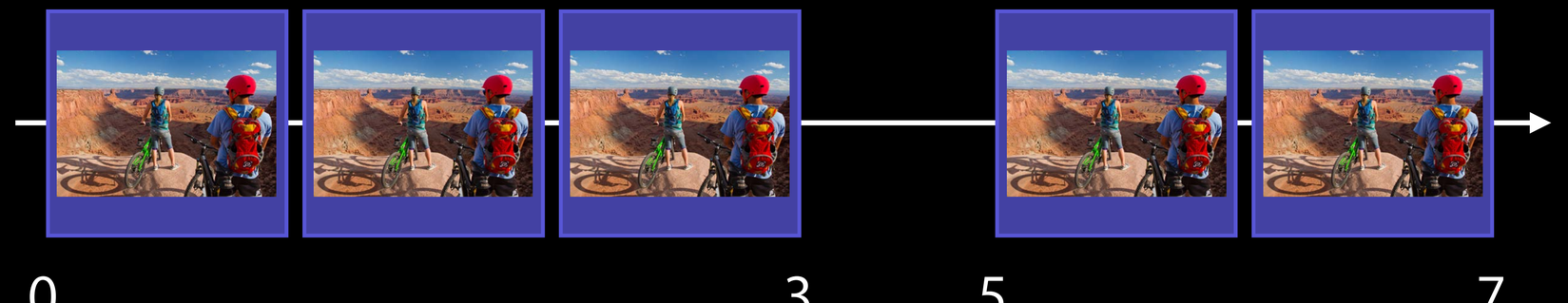
- May contain subsets of entire sequence

Query pass description time ranges

```
@property (nonatomic, readonly) NSArray *sourceTimeRanges;
```

# AVAssetWriterInputPassDescription

NEW




Encoder's request for samples in next pass

- May contain subsets of entire sequence

Query pass description time ranges

```
@property (nonatomic, readonly) NSArray *sourceTimeRanges;
```

Array of CMTimeRanges as NSValues

# AVAssetWriterInput

## Pass descriptions

# AVAssetWriterInput

## Pass descriptions


Block is called when encoder makes decision about next pass

```
– (void)respondToEachPassDescriptionOnQueue:(dispatch_queue_t)queue
                                 usingBlock:(dispatch_block_t)block;
```

# AVAssetWriterInput

## Pass descriptions


Block is called when encoder makes decision about next pass

```
– (void)respondToEachPassDescriptionOnQueue:(dispatch_queue_t)queue
                                 usingBlock:(dispatch_block_t)block;
```

Get the new description

```
@property (readonly) AVAssetWriterInputPassDescription *currentPassDescription;
```

# AVAssetWriterInput

## Sample

# AVAssetWriterInput

## Sample

```
[assetWriterInput respondToEachPassDescriptionOnQueue:queue usingBlock:^{
    AVAssetWriterInputPassDescription *pass = [assetWriterInput currentPassDescription];
    if (pass != nil) {
        // Reconfigure source to deliver samples
        [source deliverSamplesFromPassDescription:pass];

        // Ready to start next pass
        [assetWriterInput requestMediaDataWhenReadyOnQueue:queue usingBlock:block];
    } else {
        // Finished
        [assetWriterInput markAsFinished];
    }
}];
```

# AVAssetWriterInput

## Sample

```
[assetWriterInput respondToEachPassDescriptionOnQueue:queue usingBlock:^{
    AVAssetWriterInputPassDescription *pass = [assetWriterInput currentPassDescription];
    if (pass != nil) {
        // Reconfigure source to deliver samples
        [source deliverSamplesFromPassDescription:pass];
        // Ready to start next pass
        [assetWriterInput requestMediaDataWhenReadyOnQueue:queue usingBlock:block];
    } else {
        // Finished
        [assetWriterInput markAsFinished];
    }
}];
```

# AVAssetWriterInput

## Sample

```
[assetWriterInput respondToEachPassDescriptionOnQueue:queue usingBlock:^{
    AVAssetWriterInputPassDescription *pass = [assetWriterInput currentPassDescription];
    if (pass != nil) {
        // Reconfigure source to deliver samples
        [source deliverSamplesFromPassDescription:pass];
        // Ready to start next pass
        [assetWriterInput requestMediaDataWhenReadyOnQueue:queue usingBlock:block];
    } else {
        // Finished
        [assetWriterInput markAsFinished];
    }
}];
```

# AVAssetWriterInput

## Sample

```
[assetWriterInput respondToEachPassDescriptionOnQueue:queue usingBlock:^{
    AVAssetWriterInputPassDescription *pass = [assetWriterInput currentPassDescription];
    if (pass != nil) {
        // Reconfigure source to deliver samples
        [source deliverSamplesFromPassDescription:pass];

        // Ready to start next pass
        [assetWriterInput requestMediaDataWhenReadyOnQueue:queue usingBlock:block];
    } else {
        // Finished
        [assetWriterInput markAsFinished];
    }
}];
```

# AVAssetWriterInput

## Sample

```
[assetWriterInput respondToEachPassDescriptionOnQueue:queue usingBlock:^{
    AVAssetWriterInputPassDescription *pass = [assetWriterInput currentPassDescription];
    if (pass != nil) {
        // Reconfigure source to deliver samples
        [source deliverSamplesFromPassDescription:pass];

        // Ready to start next pass
        [assetWriterInput requestMediaDataWhenReadyOnQueue:queue usingBlock:block];
    } else {
        // Finished
        [assetWriterInput markAsFinished];
    }
}];
```

# AVAssetReaderOutput

## New APIs

# AVAssetReaderOutput

## New APIs


Prepare source for multi-pass

```
@property (nonatomic) BOOL supportsRandomAccess;
```

# AVAssetReaderOutput

## New APIs


Prepare source for multi-pass

```
@property (nonatomic) BOOL supportsRandomAccess;
```

Reconfigure source to deliver samples in time ranges

```
- (void)resetForReadingTimeRanges:(NSArray *)timeRanges;
```

# AVAssetReaderOutput

## New APIs


Prepare source for multi-pass

```
@property (nonatomic) BOOL supportsRandomAccess;
```

Reconfigure source to deliver samples in time ranges

```
- (void)resetForReadingTimeRanges:(NSArray *)timeRanges;
```

All passes have completed

```
- (void)markConfigurationAsFinal;
```

# AVAssetReader and AVAssetWriter

# AVAssetReader and AVAssetWriter

Enable AVAssetReaderOutput if AVAssetWriterInput support multi-pass

```
readerOutput.supportsRandomAccess = writerInput.canPerformMultiplePasses;
```

# AVAssetReader and AVAssetWriter

Enable AVAssetReaderOutput if AVAssetWriterInput support multi-pass

```
readerOutput.supportsRandomAccess = writerInput.canPerformMultiplePasses;
```

Reconfigure source to deliver samples for an AVAssetWriterInput

```
[readerOutput resetForReadingTimeRanges:passDescription.sourceTimeRanges];
```

# AVAssetReaderOutput

Sample

# AVAssetReaderOutput

## Sample

```
[assetWriterInput respondToEachPassDescriptionOnQueue:queue usingBlock:^{
    AVAssetWriterInputPassDescription *pass = [assetWriterInput currentPassDescription];
    if (currentPass != nil) {
        // Reconfigure source to deliver samples
        [readerOutput resetForReadingTimeRanges:pass.sourceTimeRanges];
        // Ready to start next pass
        [assetWriterInput requestMediaDataWhenReadyOnQueue:queue usingBlock:block];
    } else {
        // Finished
        [assetWriterInput markAsFinished];
    }
}];
```

AVKit

AVFoundation

Video Toolbox

Core Media

Core Video

# Video Toolbox

Encoder frame analysis database

- VTMultiPassStorage

Additions to VTCompressionSession

Compressed frame database

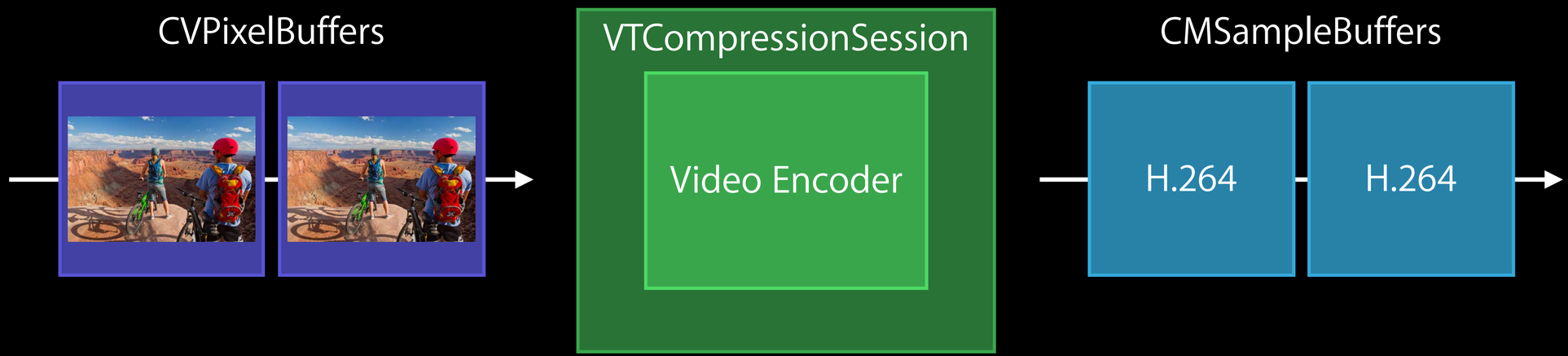
- VTFrameSilo

# Video Toolbox Multi-Pass

## Architecture

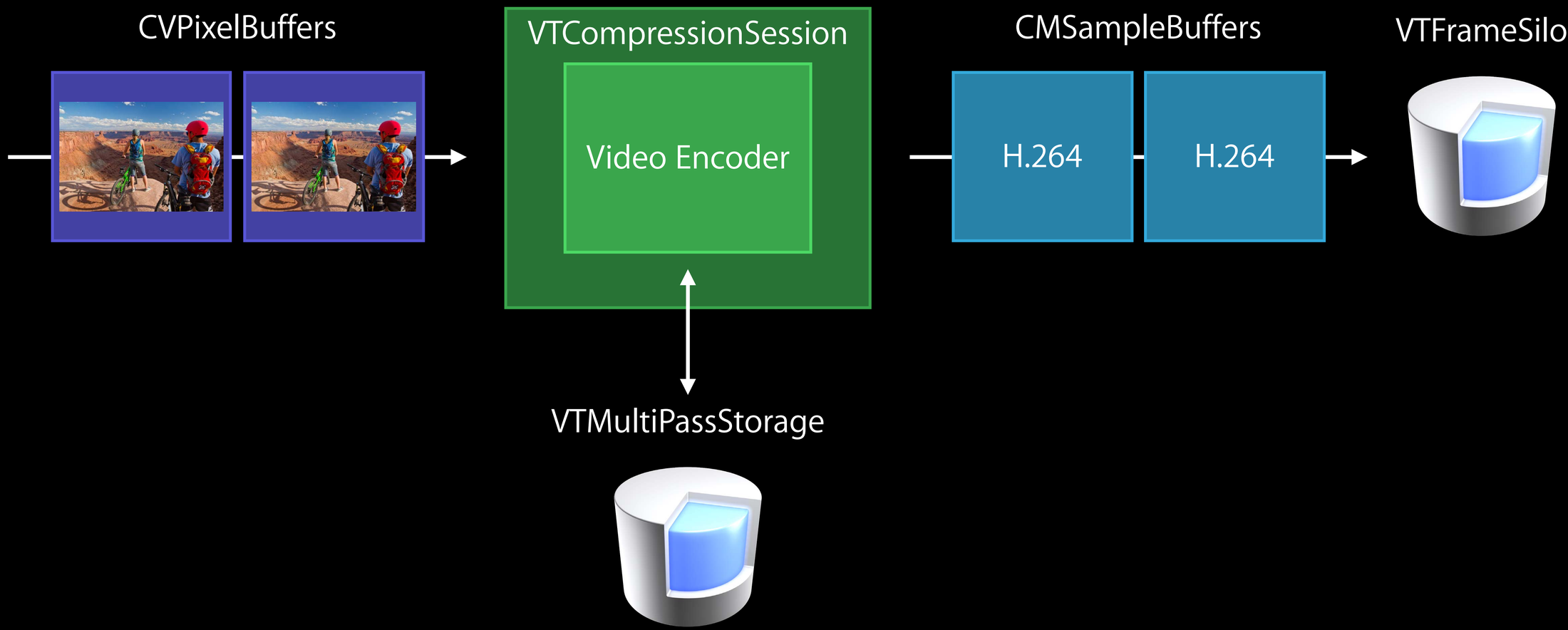
# Video Toolbox Multi-Pass

## Architecture



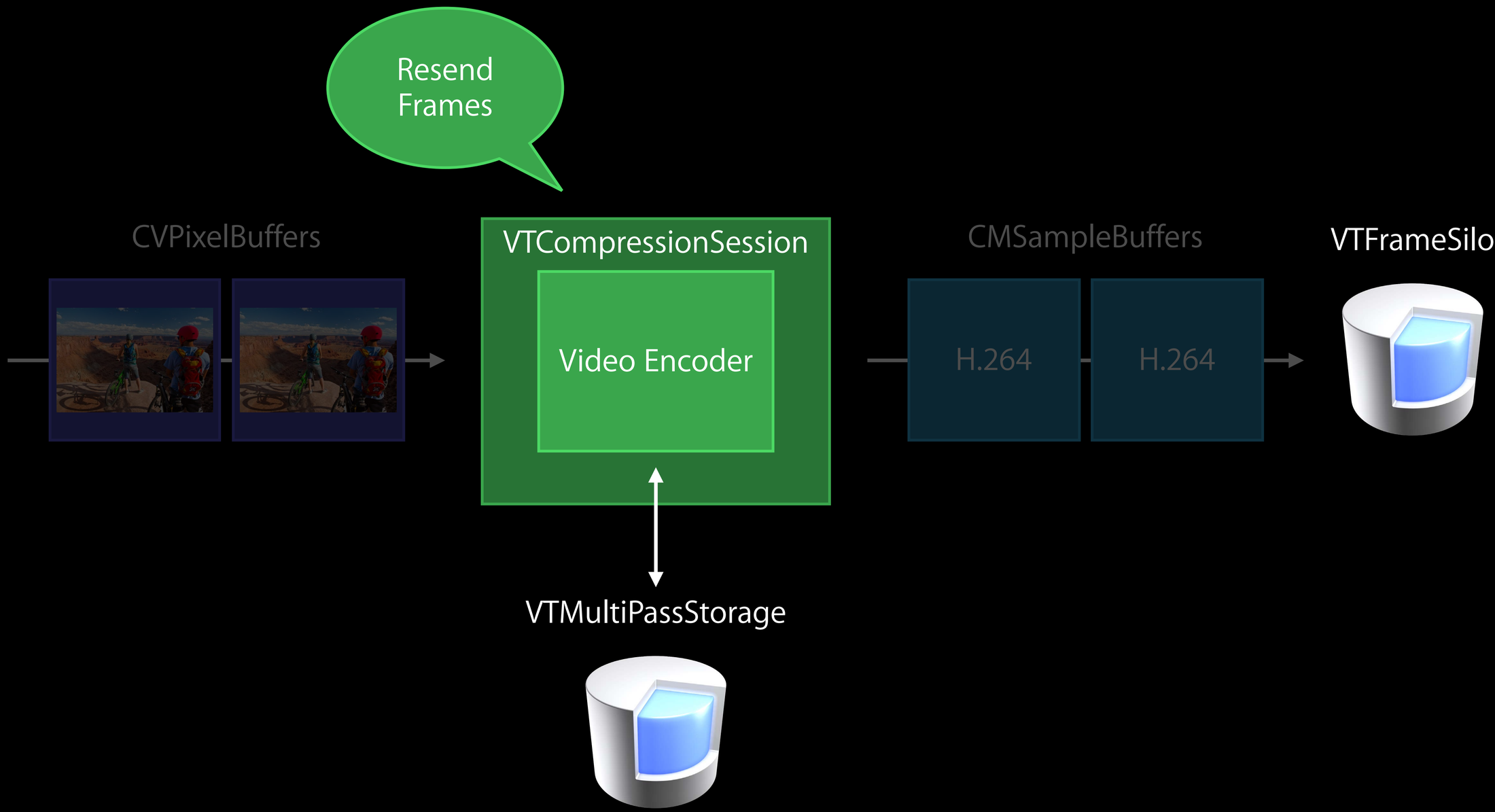
# Video Toolbox Multi-Pass

## Architecture



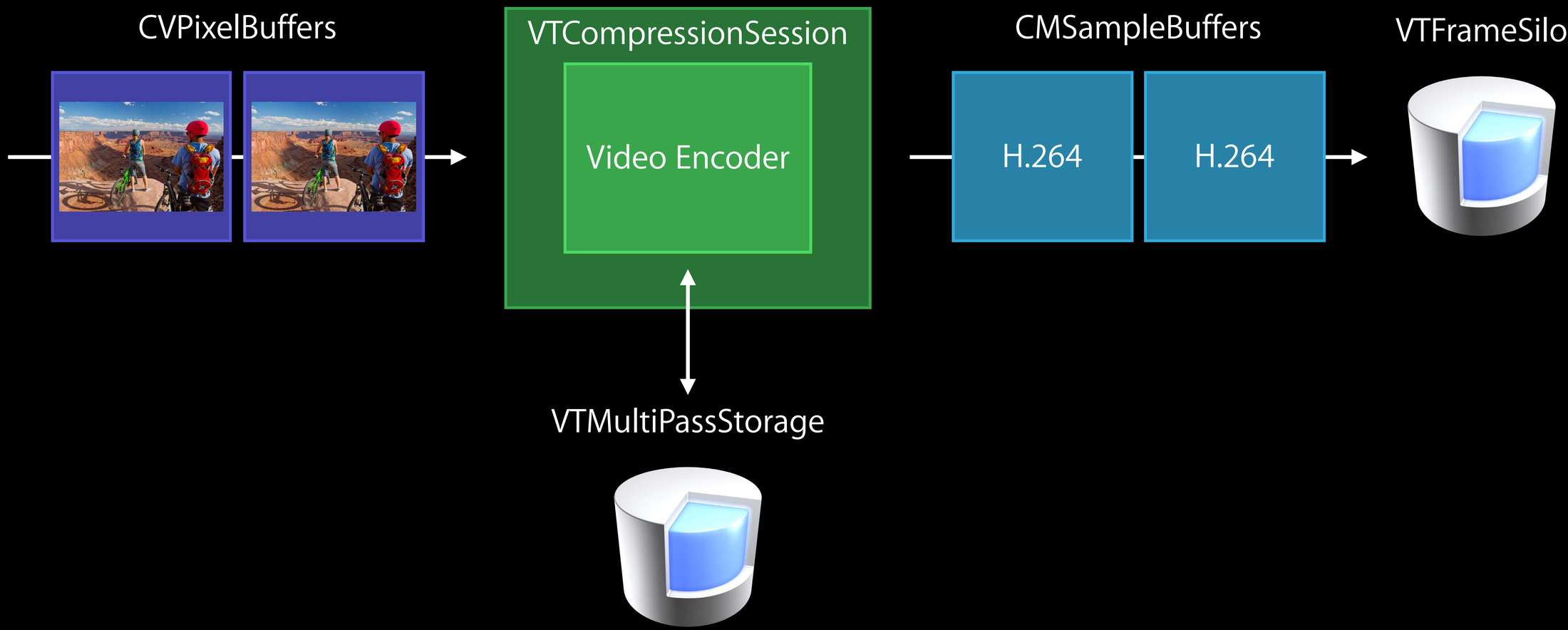
# Video Toolbox Multi-Pass

## Architecture



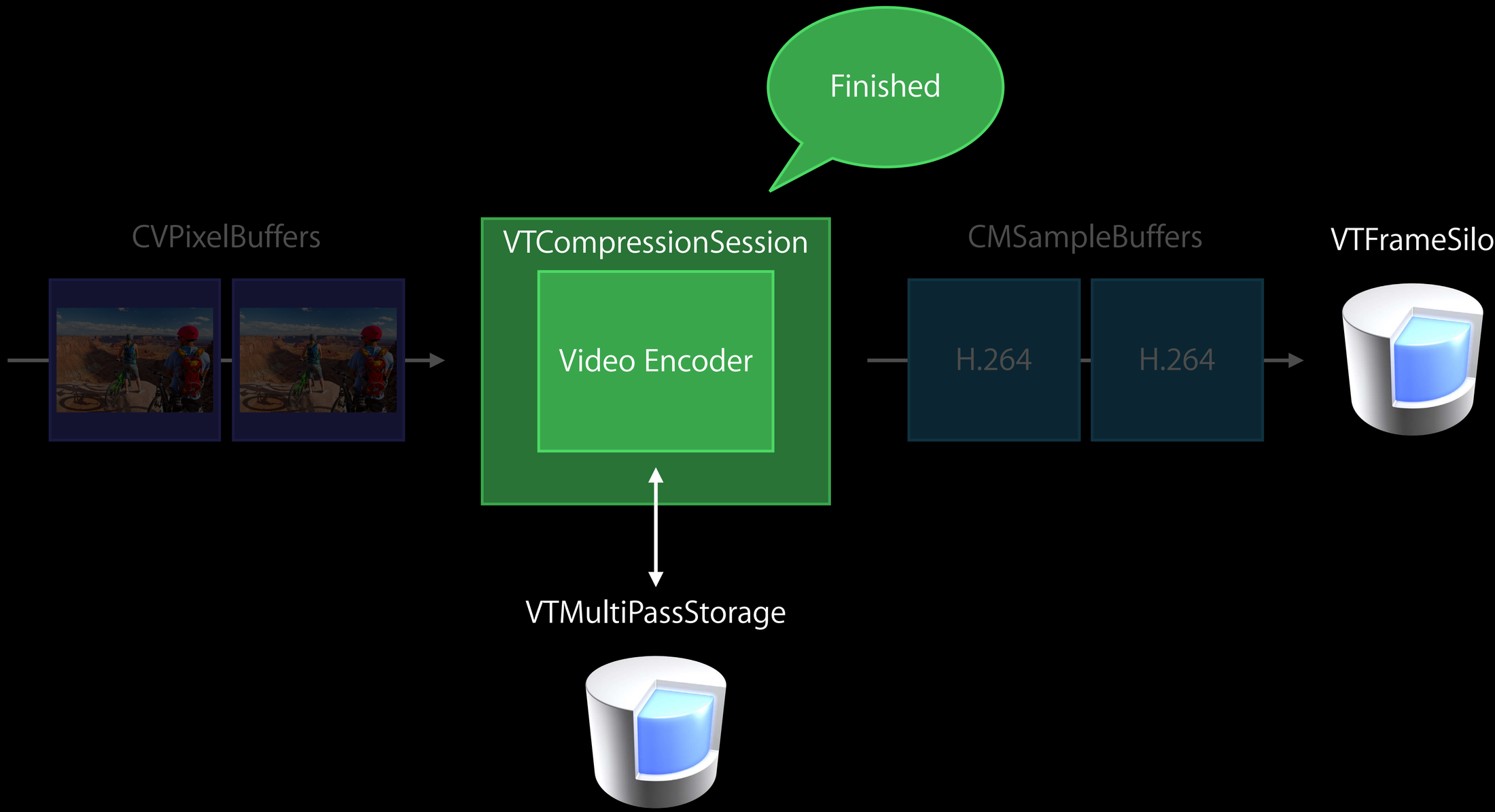
# Video Toolbox Multi-Pass

## Architecture



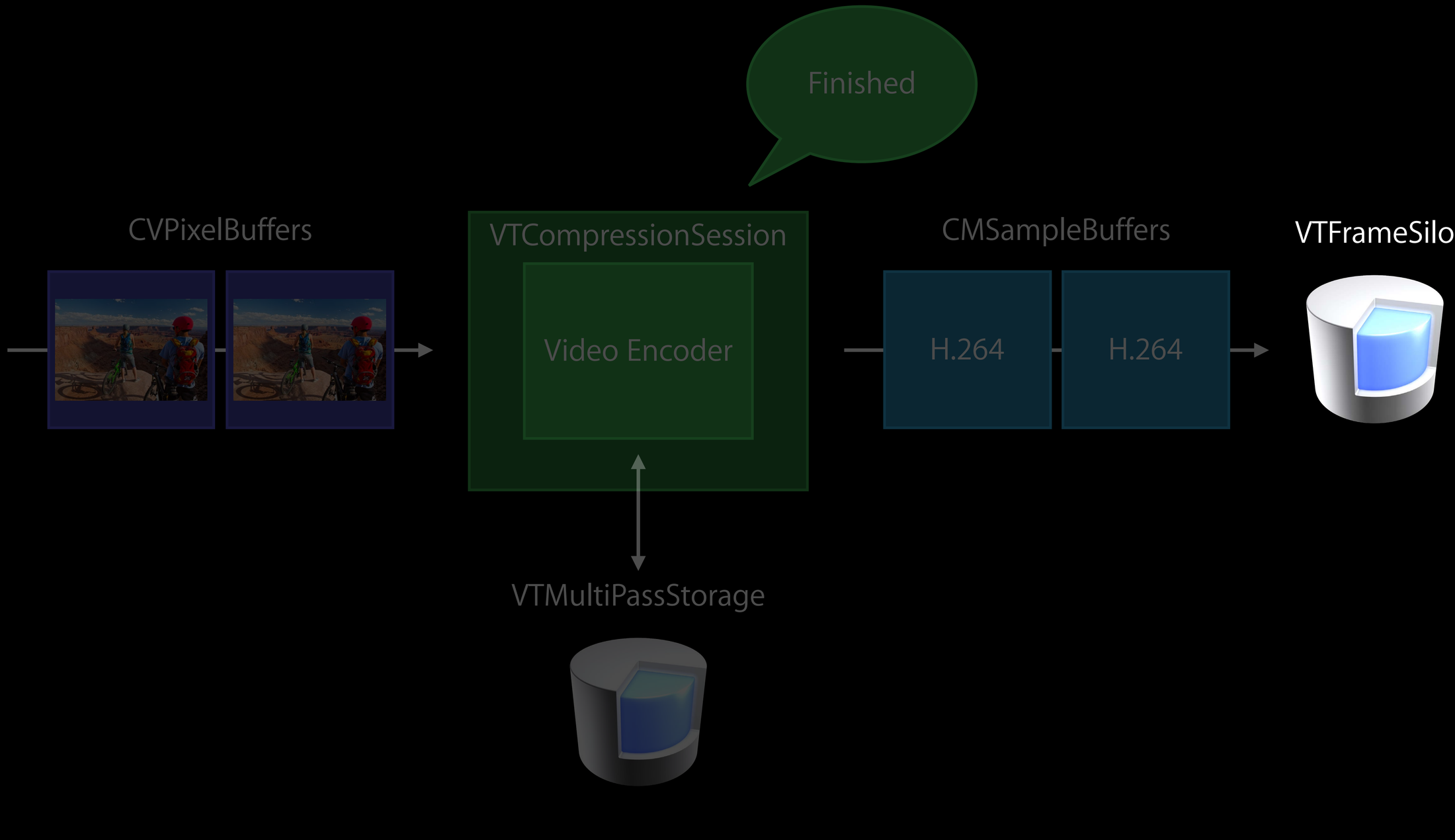
# Video Toolbox Multi-Pass

## Architecture



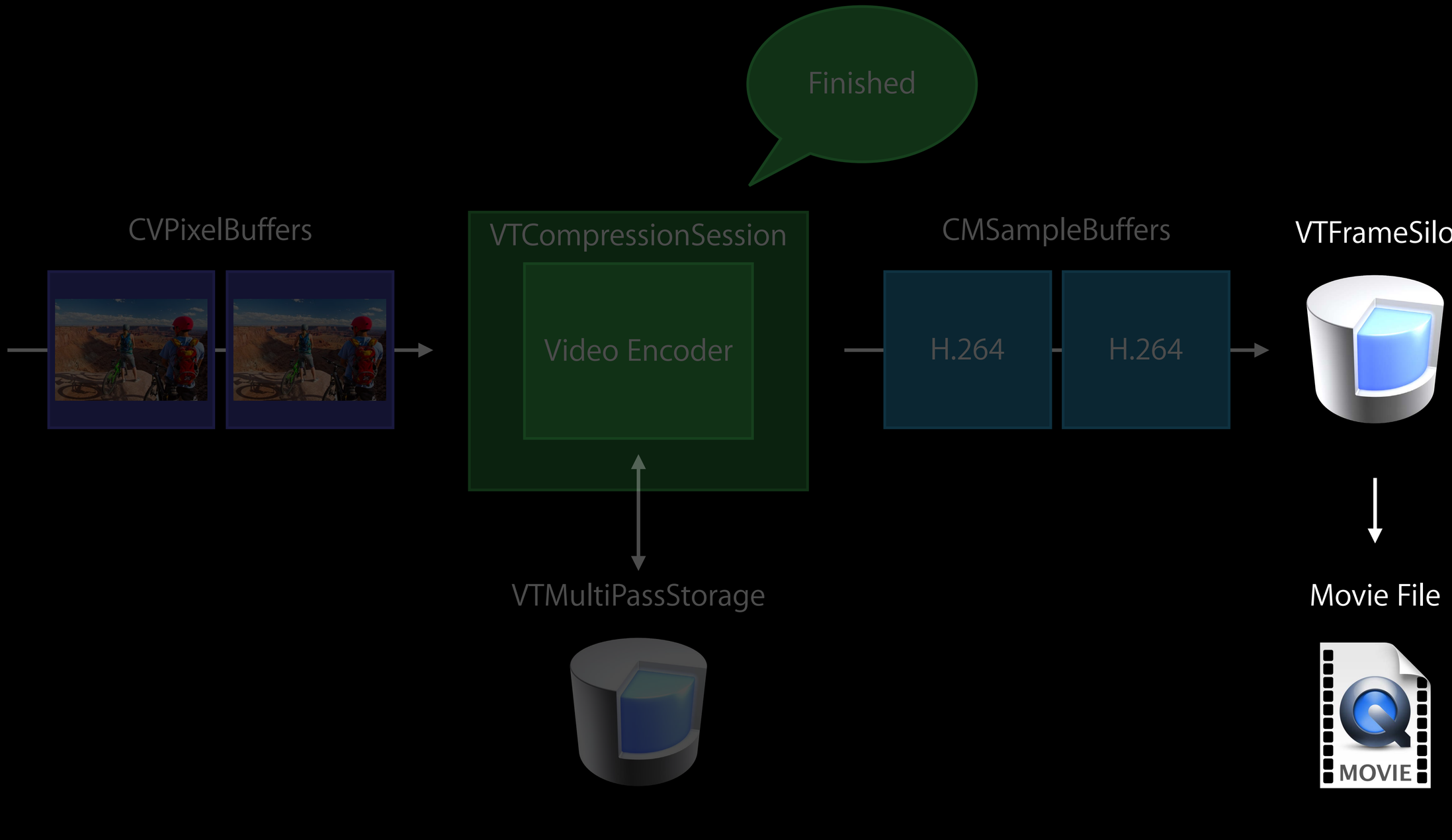
# Video Toolbox Multi-Pass

## Architecture



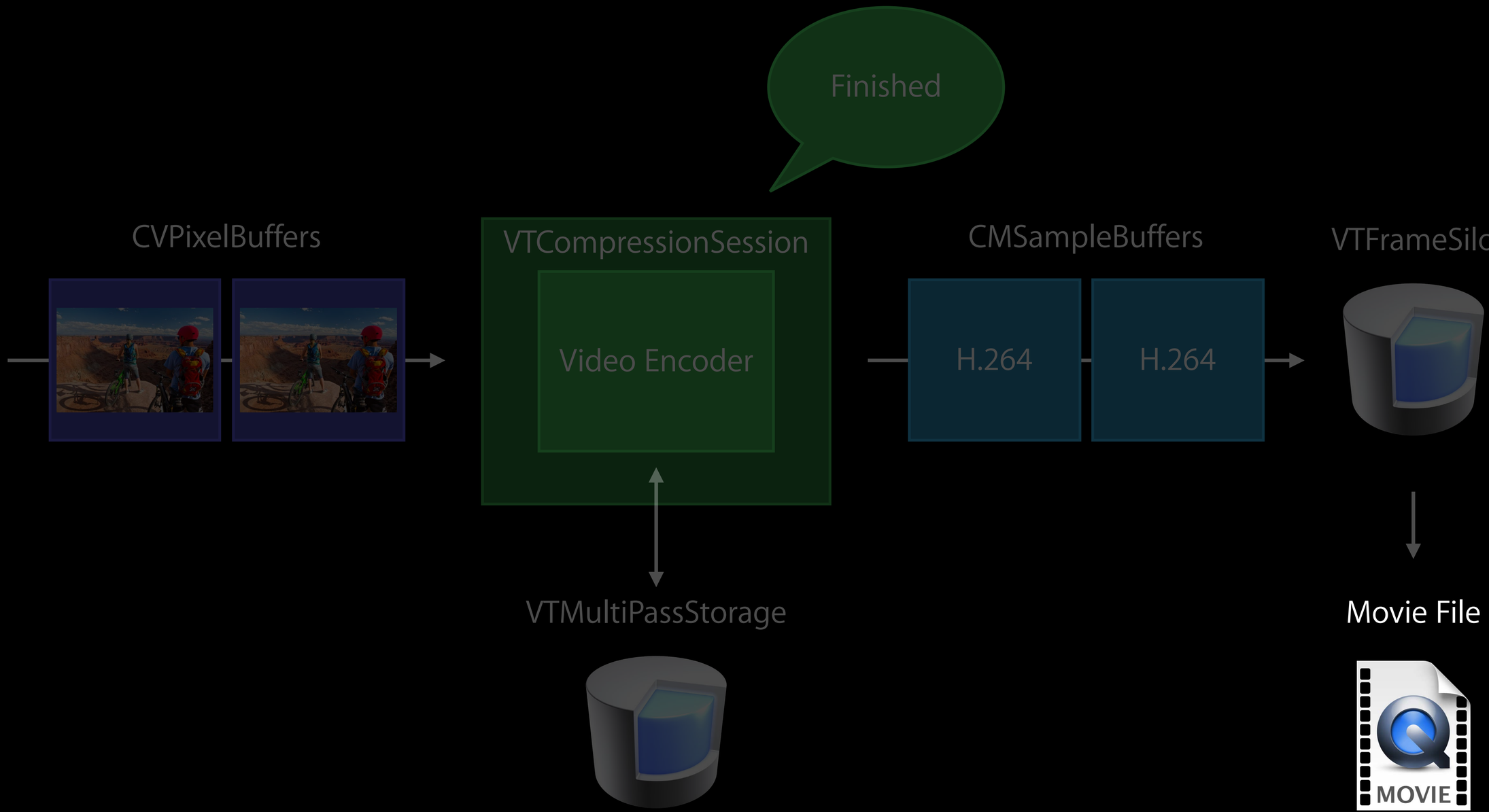
# Video Toolbox Multi-Pass

## Architecture

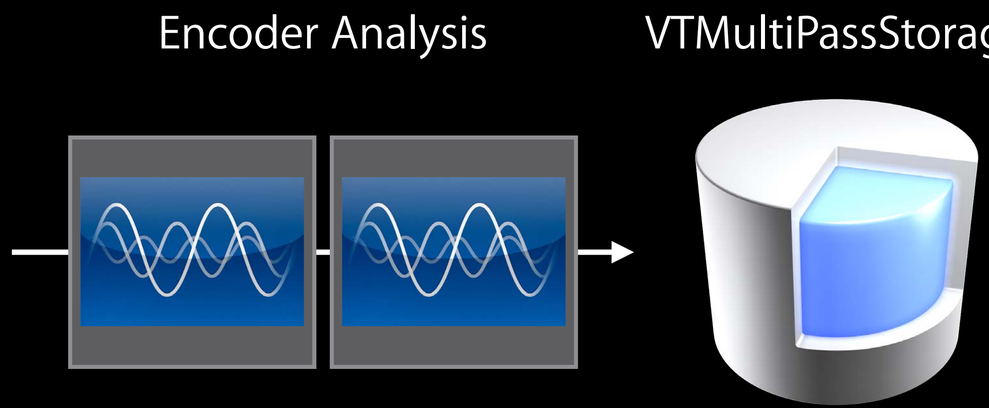


# Video Toolbox Multi-Pass

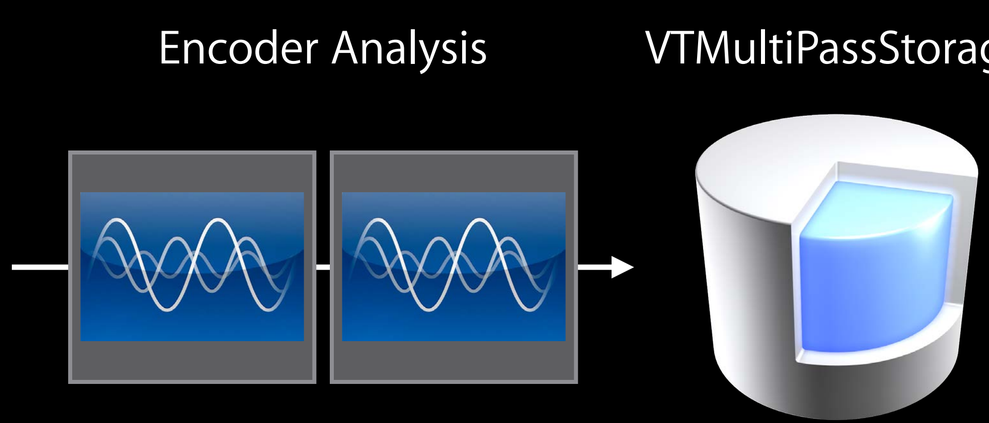
## Architecture



# VTMultiPassStorage



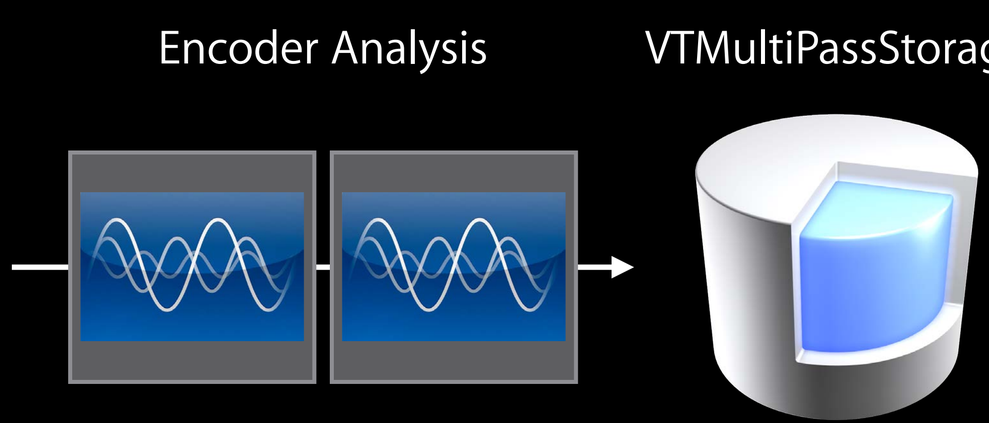
# VTMultiPassStorage




Create the encoder analysis storage

```
error = VTMultiPassStorageCreate( allocator, fileURL, timeRange,
                                  options, &storage );
```

# VTMultiPassStorage



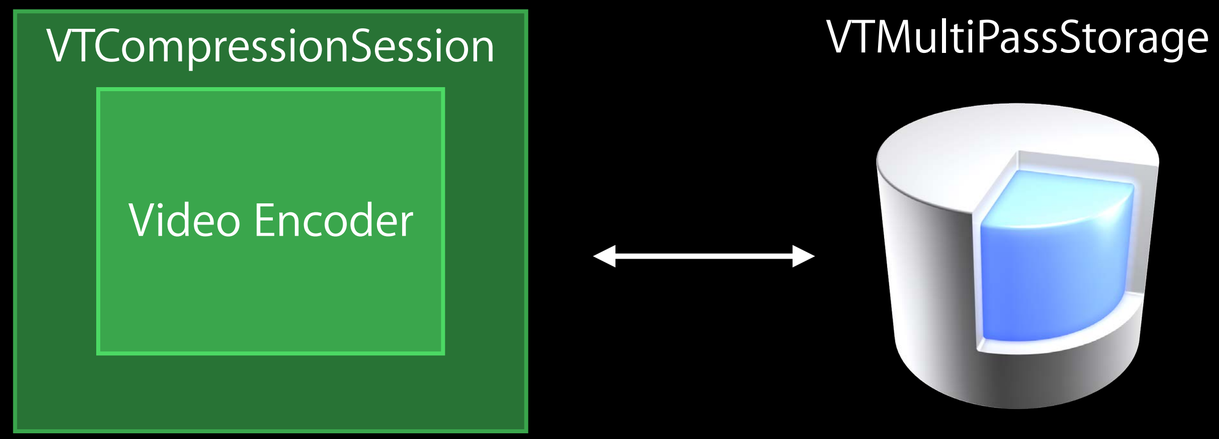

Create the encoder analysis storage

```
error = VTMultiPassStorageCreate( allocator, fileURL, timeRange,
                                  options, &storage );
```

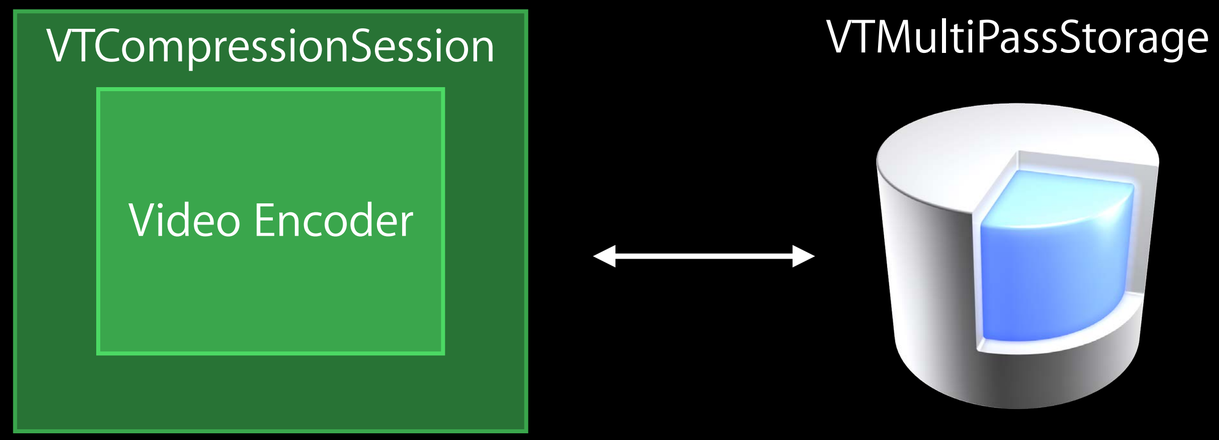
Close the file

```
error = VTMultiPassStorageClose( storage );
```

# VTCompressionSession



# VTCompressionSession




Enable multi-pass

```
error = VTSessionSetProperty( session,
            kVTCompressionPropertyKey_MultiPassStorage, multiPassStorage );
```

# VTCompressionSession

# VTCompressionSession

Begin a pass

```
error = VTCompressionSessionBeginPass( session, 0, NULL );
```

# VTCompressionSession

Begin a pass

```
error = VTCompressionSessionBeginPass( session, 0, NULL );
```

End a pass

- Asks the encoder if another pass can be performed

```
error = VTCompressionSessionEndPass( session,
                                     &furtherPassesRequested, NULL );
```

# VTCompressionSession

Begin a pass

```
error = VTCompressionSessionBeginPass( session, 0, NULL );
```

End a pass

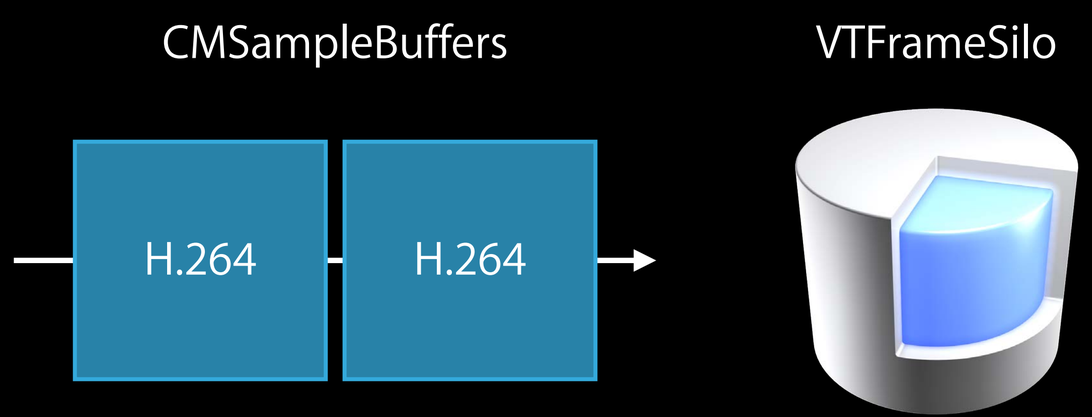
- Asks the encoder if another pass can be performed

```
error = VTCompressionSessionEndPass( session,
                                     &furtherPassesRequested, NULL );
```

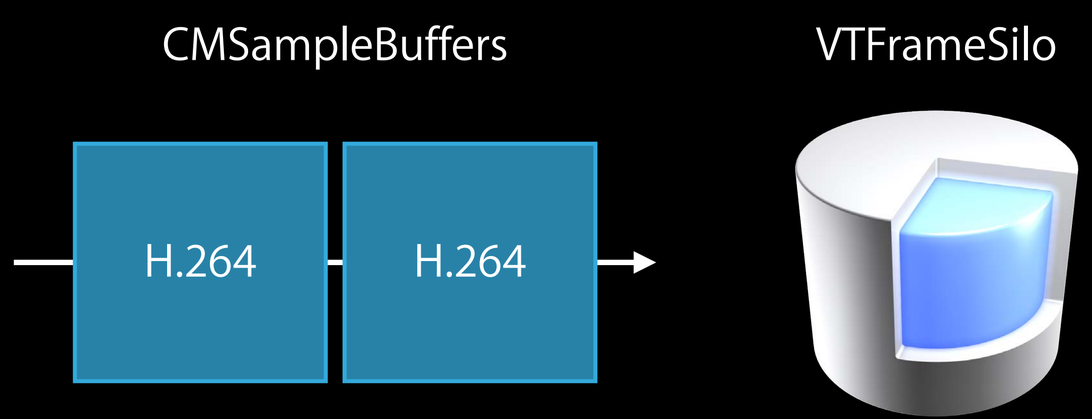
Retrieve time ranges for next pass

```
error = VTCompressionSessionGetTimeRangesForNextPass( session,
                                     &timeRangesCount, &timeRangeArray );
```

# VTFrameSilo



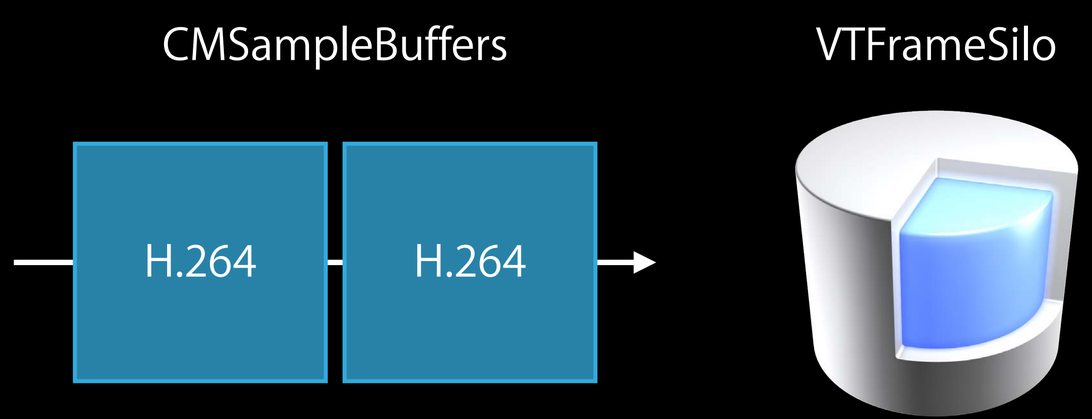
# VTFrameSilo




Create compressed frame storage

```
error = VTFrameSiloCreate( allocator, fileURL, timeRange, options, &silo );
```

# VTFrameSilo



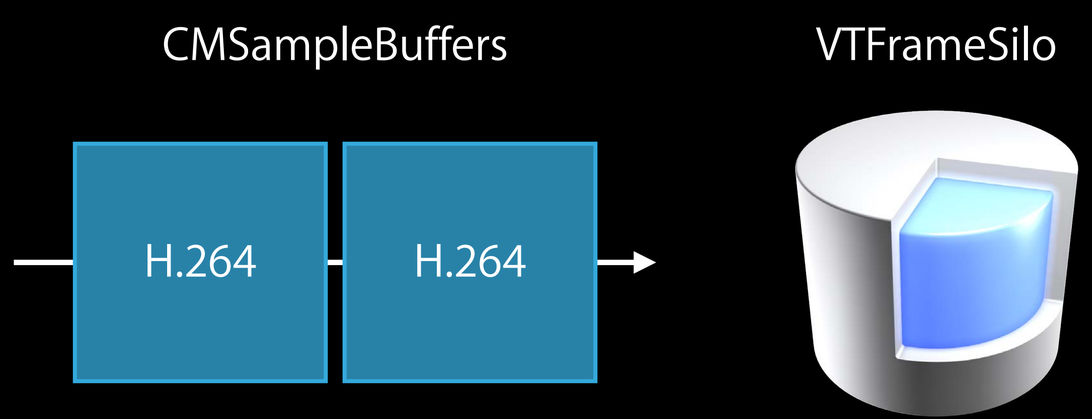

Create compressed frame storage

```
error = VTFrameSiloCreate( allocator, fileURL, timeRange, options, &silo );
```

Add a sample in VTCompressionOutputCallback

```
error = VTFrameSiloAddSampleBuffer( silo, sampleBuffer );
```

# VTFrameSilo




Create compressed frame storage

```
error = VTFrameSiloCreate( allocator, fileURL, timeRange, options, &silo );
```

Add a sample in VTCompressionOutputCallback

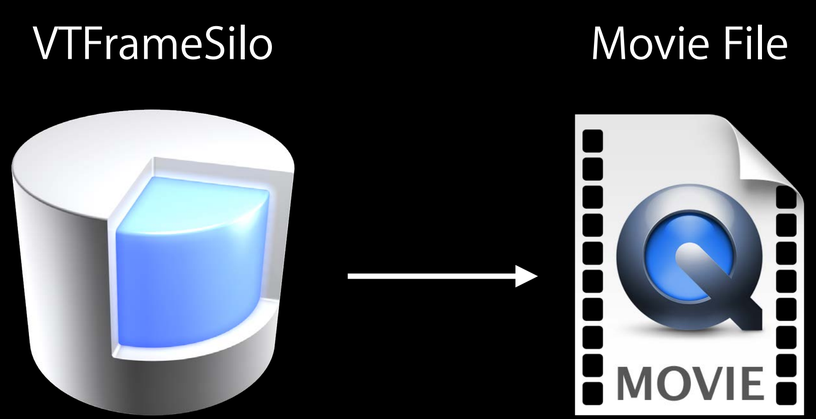
```
error = VTFrameSiloAddSampleBuffer( silo, sampleBuffer );
```

Prepare for next pass

```
error = VTFrameSiloSetTimeRangesForNextPass( silo,
                                             timeRangeCount, timeRangeArray );
```

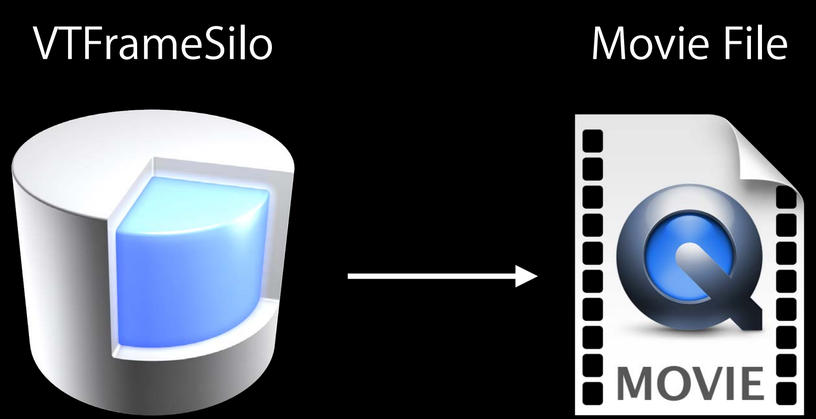
# VTFrameSilo

## File output



# VTFrameSilo

## File output




Retrieve samples for time range

```
error = VTFrameSiloCallBlockForEachSampleBuffer( silo,
            timeRange, ^(CMSampleBuffer sampleBuffer) {
      // append sampleBuffer to movie file
 });
```

# Considerations

# Use Case Considerations

Best Choice

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |
| Closer to Target Bit Rate | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |
| Closer to Target Bit Rate | Multi-Pass |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |
| Closer to Target Bit Rate | Multi-Pass |
| Okay to Take Longer | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |
| Closer to Target Bit Rate | Multi-Pass |
| Okay to Take Longer | Multi-Pass |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |
| Closer to Target Bit Rate | Multi-Pass |
| Okay to Take Longer | Multi-Pass |
| Your App | |

# Use Case Considerations

| | Best Choice |
|---|---|
| Real Time | Single-Pass |
| Minimum Power Use | Single-Pass |
| Minimum Temporary Storage | Single-Pass |
| Best Quality | Multi-Pass |
| Closer to Target Bit Rate | Multi-Pass |
| Okay to Take Longer | Multi-Pass |
| Your App | Experiment |

# Content Considerations

Best Choice

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |
| High Complexity | |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |
| High Complexity | Single/Multi |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |
| High Complexity | Single/Multi |
| Varying Complexity | |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |
| High Complexity | Single/Multi |
| Varying Complexity | Multi-Pass |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |
| High Complexity | Single/Multi |
| Varying Complexity | Multi-Pass |
| Your Content | |

# Content Considerations

| | Best Choice |
|---|---|
| Low Complexity | Single/Multi |
| High Complexity | Single/Multi |
| Varying Complexity | Multi-Pass |
| Your Content | Experiment |

# Summary

# Summary

AVFoundation provides powerful APIs to operate on media

# Summary

AVFoundation provides powerful APIs to operate on media

Video Toolbox APIs provide direct codec access

# Summary

AVFoundation provides powerful APIs to operate on media

Video Toolbox APIs provide direct codec access

Multi-pass can provide substantial quality improvements

# More Information

Evangelism
evangelism@apple.com

AVFoundation Documentation
AVFoundation Programming Guide
https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/AVFoundationPG/

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

# Related Sessions

| | | |
|---|---|---|
| • Mastering Modern Media Playback | Mission | Tuesday 11:30AM |
| • Harnessing Metadata in Audiovisual Media | Pacific Heights | Tuesday 2:00PM |
| • Camera Capture: Manual Controls | Marina | Wednesday 11:30AM |
| • Introducing the Photos Frameworks | Nob Hill | Thursday 10:15AM |

# Labs

| | | |
|---|---|---|
| • AVFoundation and Camera Capture Lab | Media Lab A | Thursday 2:00PM |

WWDC14