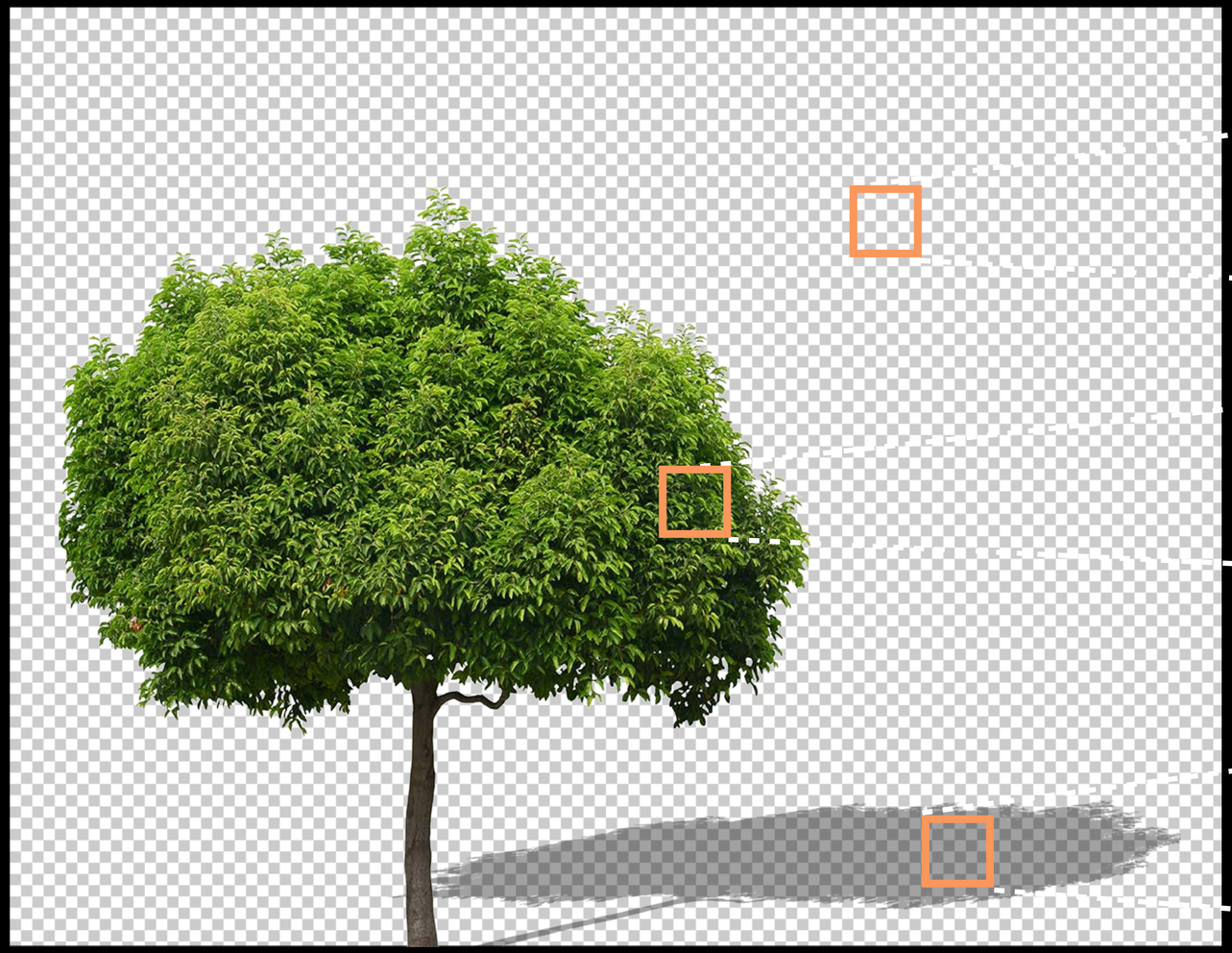
#WWDC19

HEVC Video with Alpha

Shiva Sundar, CoreMedia Engineering

Alpha Channels



0% opaque
(fully transparent)

100% opaque

50% opaque

Alpha Channels

0% opaque
(fully transparent)

100% opaque

50% opaque

Formats with Alpha Channels

	Still Image
High bitrate Lossless/Mezzanine	PNG, TIFF

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444
Flexible bitrate	HEIF	

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444
Flexible bitrate	HEIF / HEIFS	

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444
Flexible bitrate	HEIF / HEIFS	HEVC with Alpha

Platforms

iOS

iOS 13

tvOS

tvOS 13

macOS

macOS Catalina

Use Cases

Use Cases

Use Cases

Video Overlays

Use Cases

Video Overlays

Use Cases

Video Overlays

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Background Removal

Use Cases

Use Cases

Use Cases

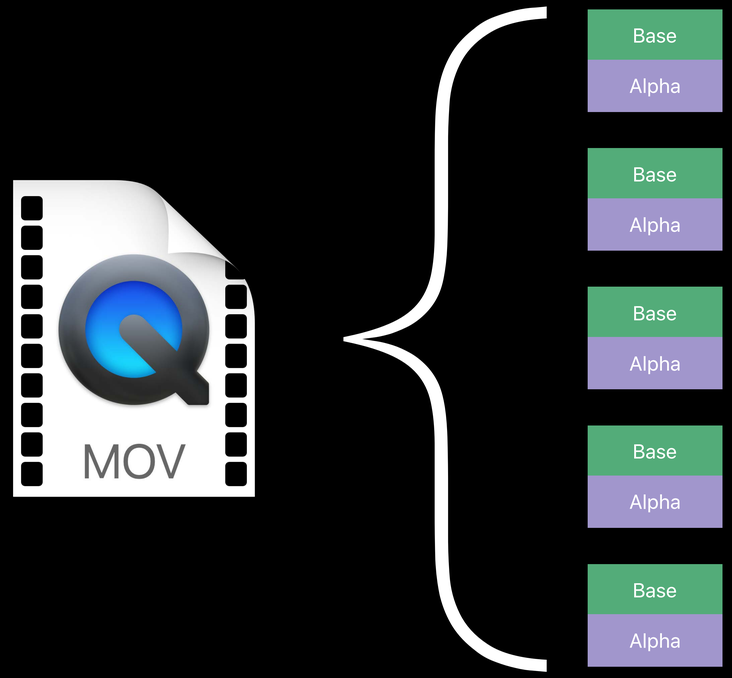
Use Cases

Use Cases

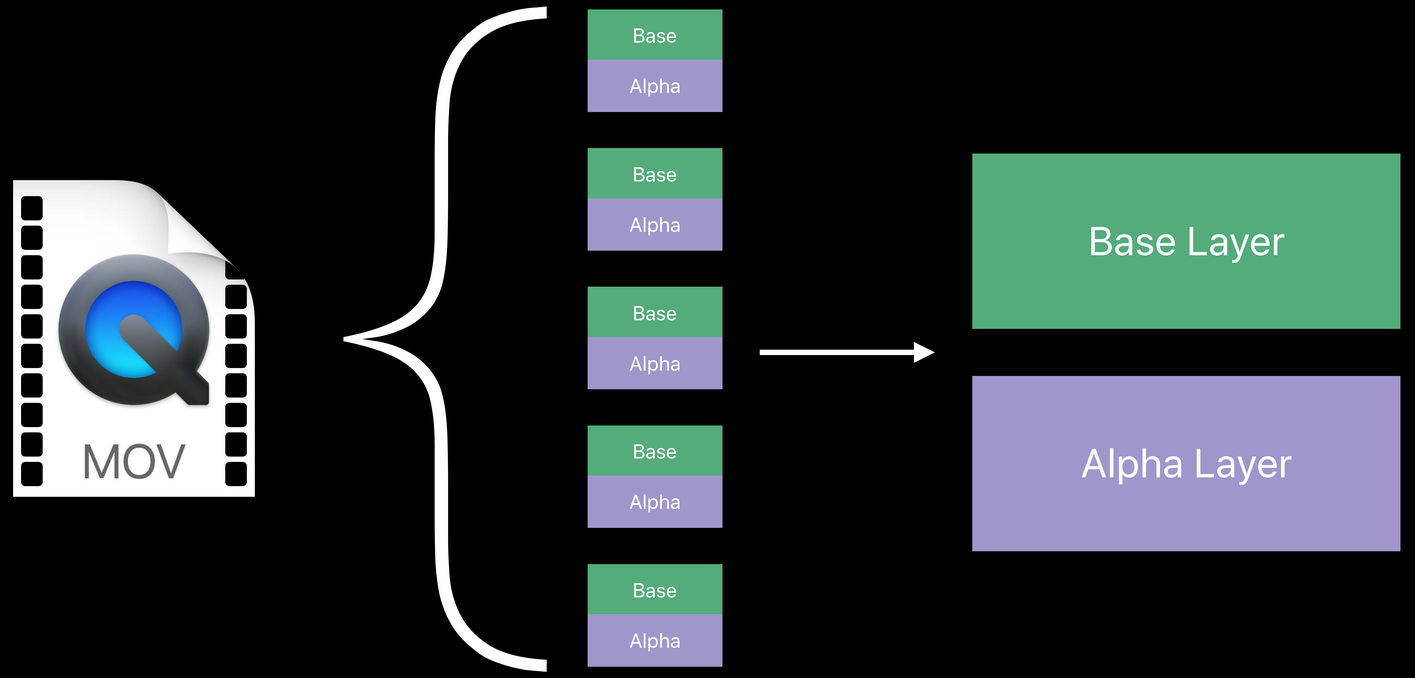
Use Cases

How it works

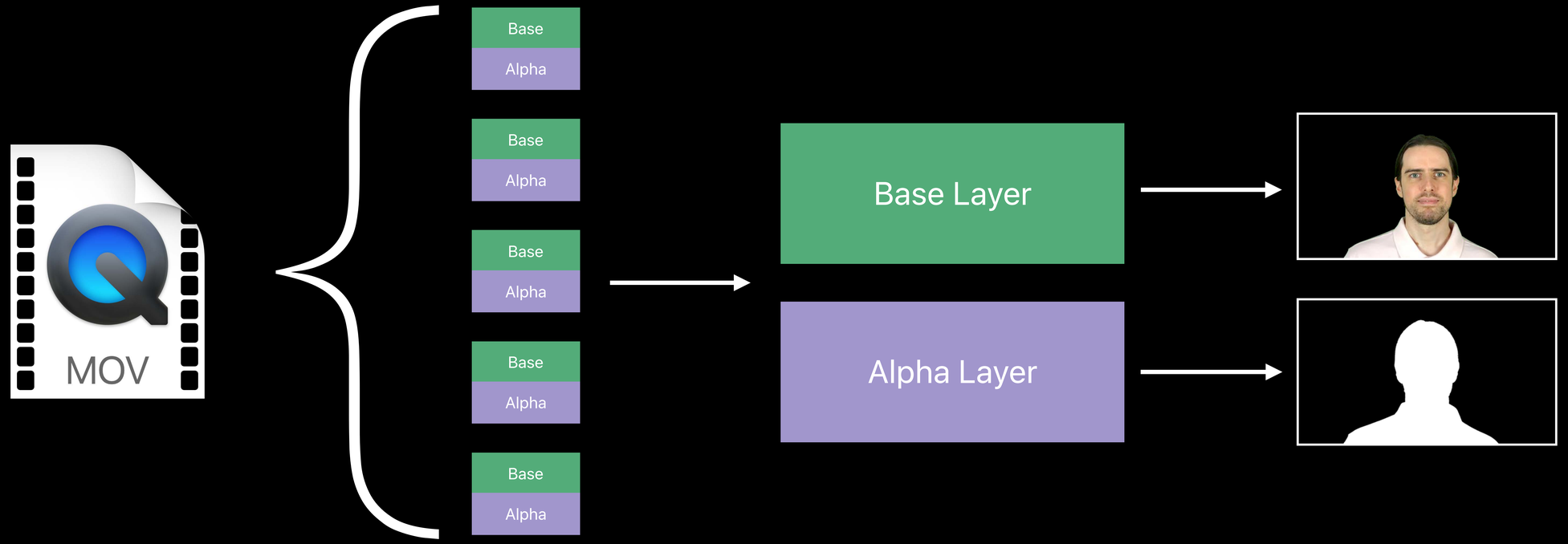
How it works



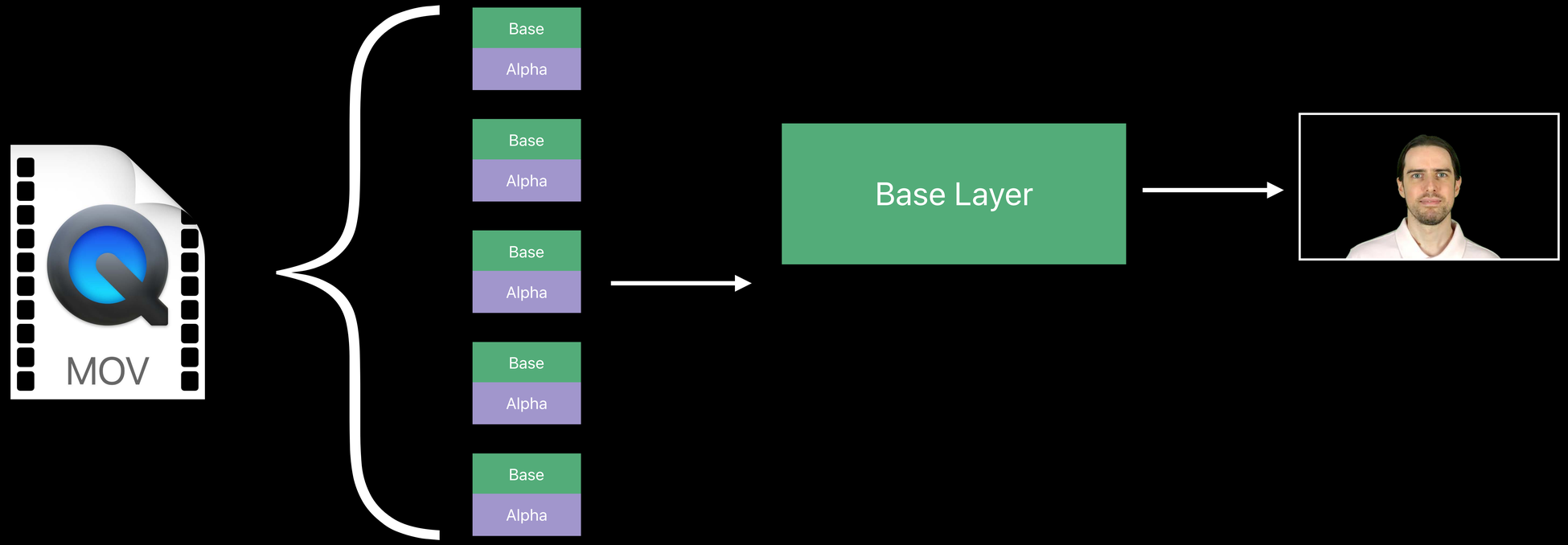
How it works


MOV
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base Layer
Alpha Layer

How it works


MOV
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base Layer
Alpha Layer

How it works


MOV
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base Layer

Encoding

Encoding

Encoding

Any Video with Alpha

HEVC with Alpha

```
AVAssetExportPresetHEVC…WithAlpha
```

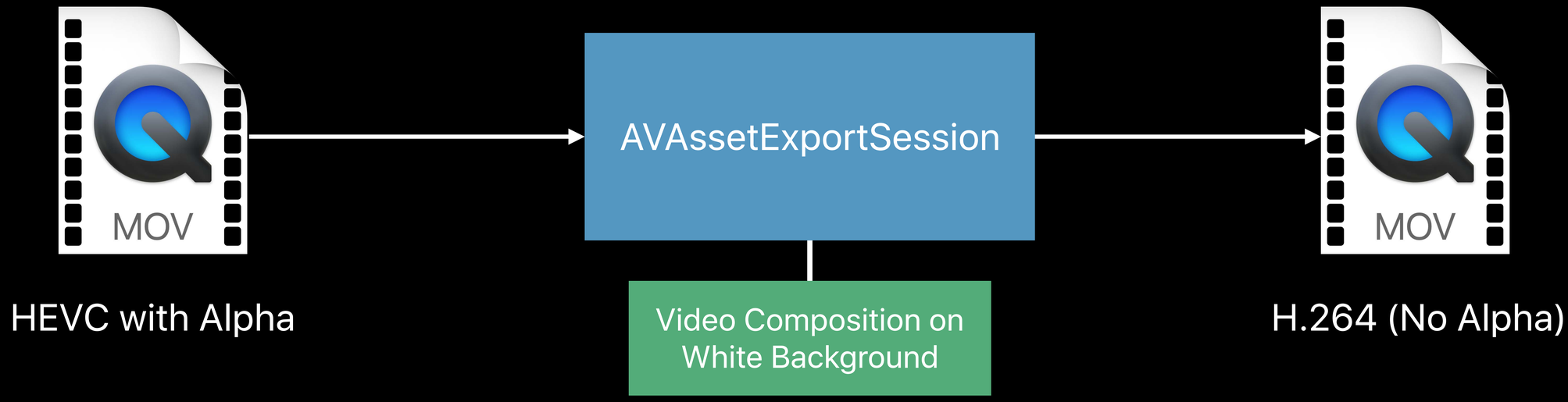
Encoding

Any Video with Alpha

HEVC with Alpha

```
AVAssetExportPresetHEVC...WithAlpha
```

Encoding




```
AVAssetExportPresetHighestQuality
```

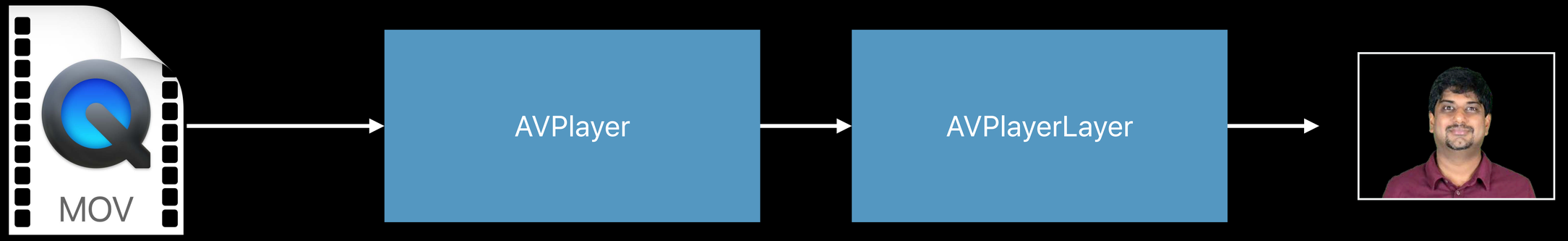
Encoding

Video Frames with Alpha

Sample Buffers

Playback

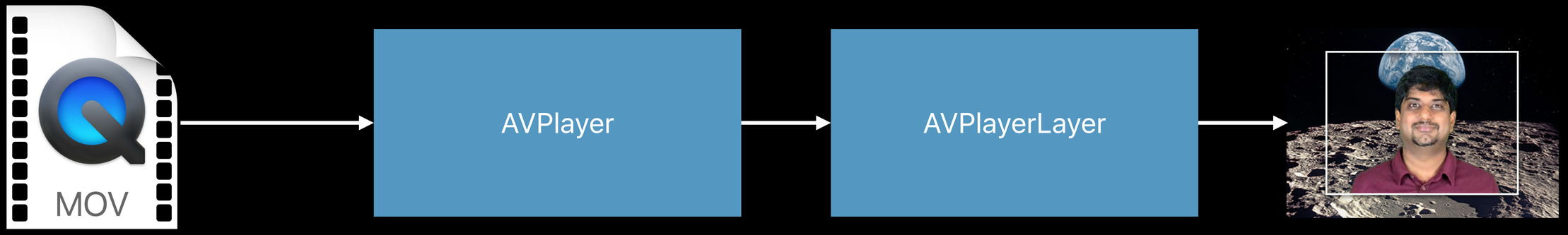
Playback



HEVC with Alpha

Displayed with
Transparent Background

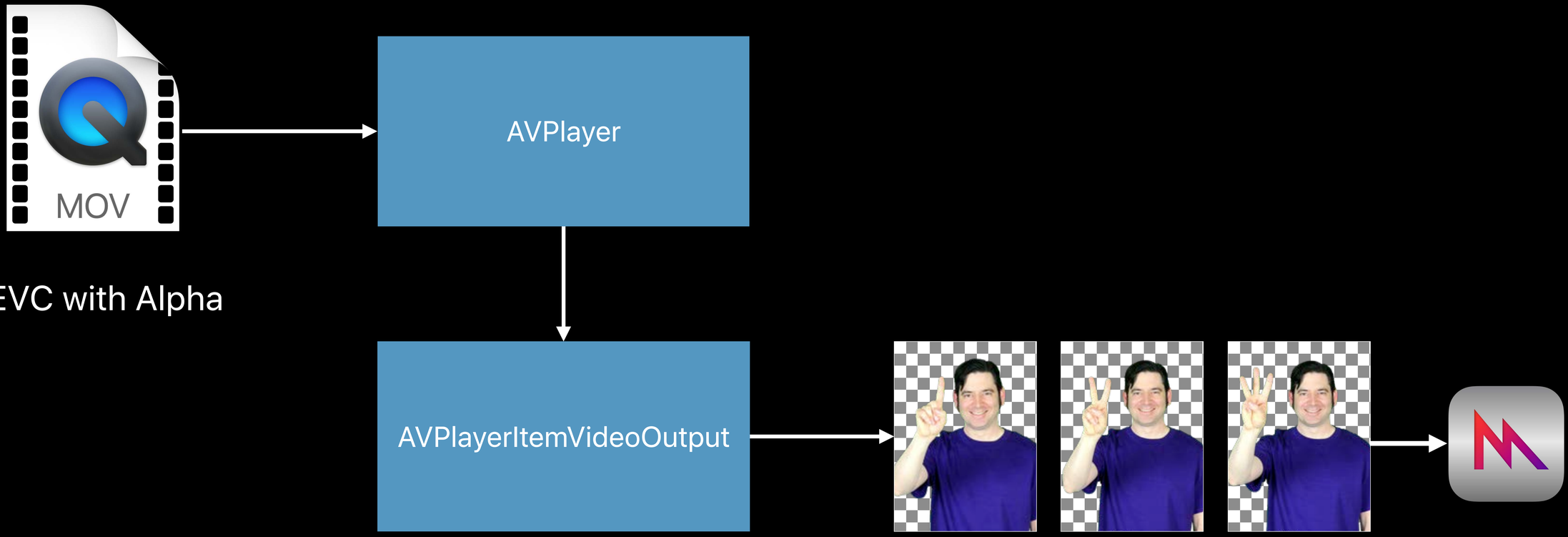
Playback



HEVC with Alpha

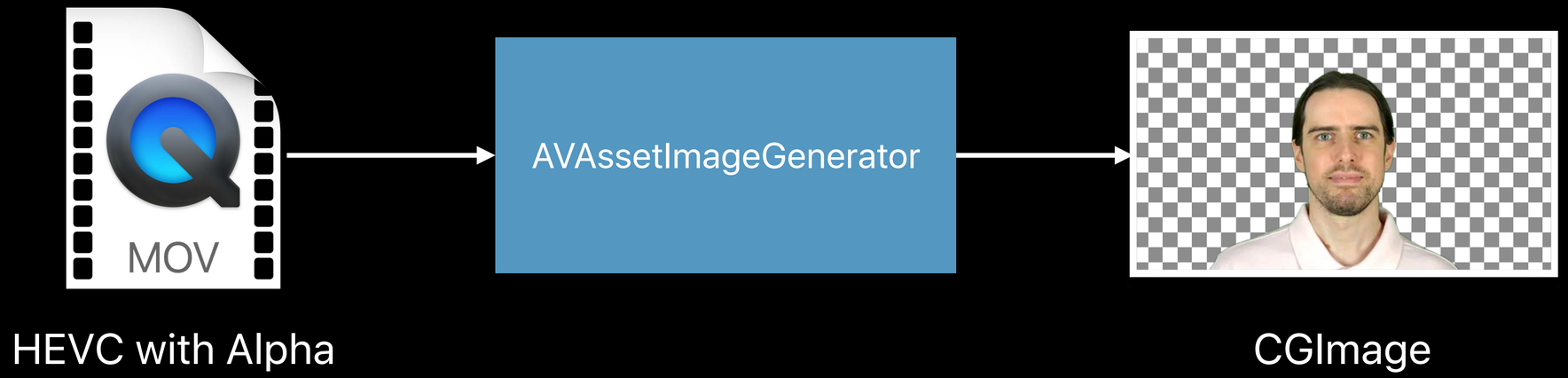
Displayed with
Transparent Background

Playback


MOV
AVPlayer
HEVC with Alpha
AVPlayerItemVideoOutput
Video Frames with Alpha

Decoding

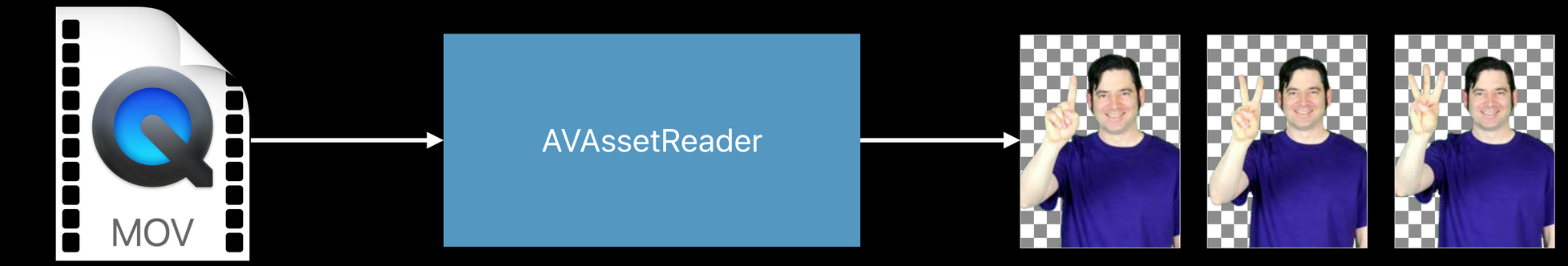
Decoding



HEVC with Alpha

CGImage

Decoding



HEVC with Alpha

Video Frames with Alpha

Decoding

Sample Buffers

Video Frames with Alpha

APIs

Encoding

AVAssetWriter

AVAssetExportSession

VTCompressionSession

Playback

AVPlayer

AVPlayerItemVideoOutput

Decoding

AVAssetImageGenerator

AVAssetReader

VTDecompressionSession

For Example

```
// AVAssetWriterInput - CodecType
import AVFoundation
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

```
// AVAssetWriterInput - CodecType
import AVFoundation
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Both Premultiplied Alpha and Straight Alpha are supported

Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Both Premultiplied Alpha and Straight Alpha are supported

Premultiplied Alpha is recommended

```swift
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
     AVVideoWidthKey: width,
     AVVideoHeightKey: height
     kVTCompressionPropertyKey_AlphaChannelMode:
        kVTAlphaChannelMode_PremultipliedAlpha
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
```

```swift
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
     AVVideoWidthKey: width,
     AVVideoHeightKey: height
     kVTCompressionPropertyKey_AlphaChannelMode:
        kVTAlphaChannelMode_PremultipliedAlpha
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
```

```swift
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
     AVVideoWidthKey: width,
     AVVideoHeightKey: height
     kVTCompressionPropertyKey_AlphaChannelMode:
        kVTAlphaChannelMode_PremultipliedAlpha
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
```

Bitrate and Quality

Can control bitrate of base layer (bits per second)

Can control quality of alpha layer (0 to 1)

```
// AVAssetWriterInput – BitRate and Quality tradeoffs
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey:
        [kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
         kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality]
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

```
// AVAssetWriterInput – BitRate and Quality tradeoffs
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey:
        [kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
         kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality]
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

```
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```

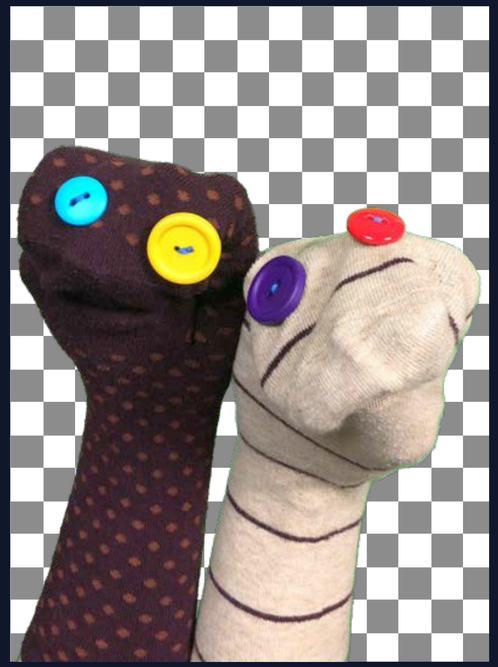
```
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```

```swift
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```

```swift
// AVAssetExportSession - compatibility
import AVFoundation
AVAssetExportSession.determineCompatibility(
    ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
    with: asset,
    outputFileType: .mov) { compatible in
    if compatible {
        // Export
    } else {
        print("Export Session failed compatibility check")
        // Handle failure
    }
}
```

```swift
// AVAssetExportSession - compatibility
import AVFoundation
AVAssetExportSession.determineCompatibility(
    ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
    with: asset,
    outputFileType: .mov) { compatible in
    if compatible {
        // Export
    } else {
        print("Export Session failed compatibility check")
        // Handle failure
    }
}
```

```
// AVAssetExportSession – backwards compatibility
import AVFoundation
// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace
let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
                                                 prototypeInstruction: prototypeInstruction)

// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
   // Handle completion
}
```



```
// AVAssetExportSession – backwards compatibility
import AVFoundation
// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace
let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
                                                 prototypeInstruction: prototypeInstruction)
```

```
// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
    // Handle completion
}
```

Interoperability Profile

Single video track with ‘hvc1’ codec type

- Two layers
  - Base layer: main profile; 4:2:0 video-range; nuh_layer_id 0
  - Alpha layer: main profile; 4:2:0 full-range, neutral chroma channel; nuh_layer_id 1
- Base and alpha layers share one VPS
  - VPS extension required — indicates presence of alpha channel
- Base and alpha layers have distinct PPS and SPS with different IDs
- Base and alpha layers must have identical frame types and dependency structure
- SEI alpha_channel_info message in hvcC in video format description
  - Indicates premultiplied alpha or straight alpha

HEVC Video with Alpha

Advanced lossy compression technology

Supported in iOS 13, tvOS 13, macOS Catalina

Hardware accelerated on recent devices

Integrated with media APIs

Integrated in Safari

More Information

developer.apple.com/wwdc19/506

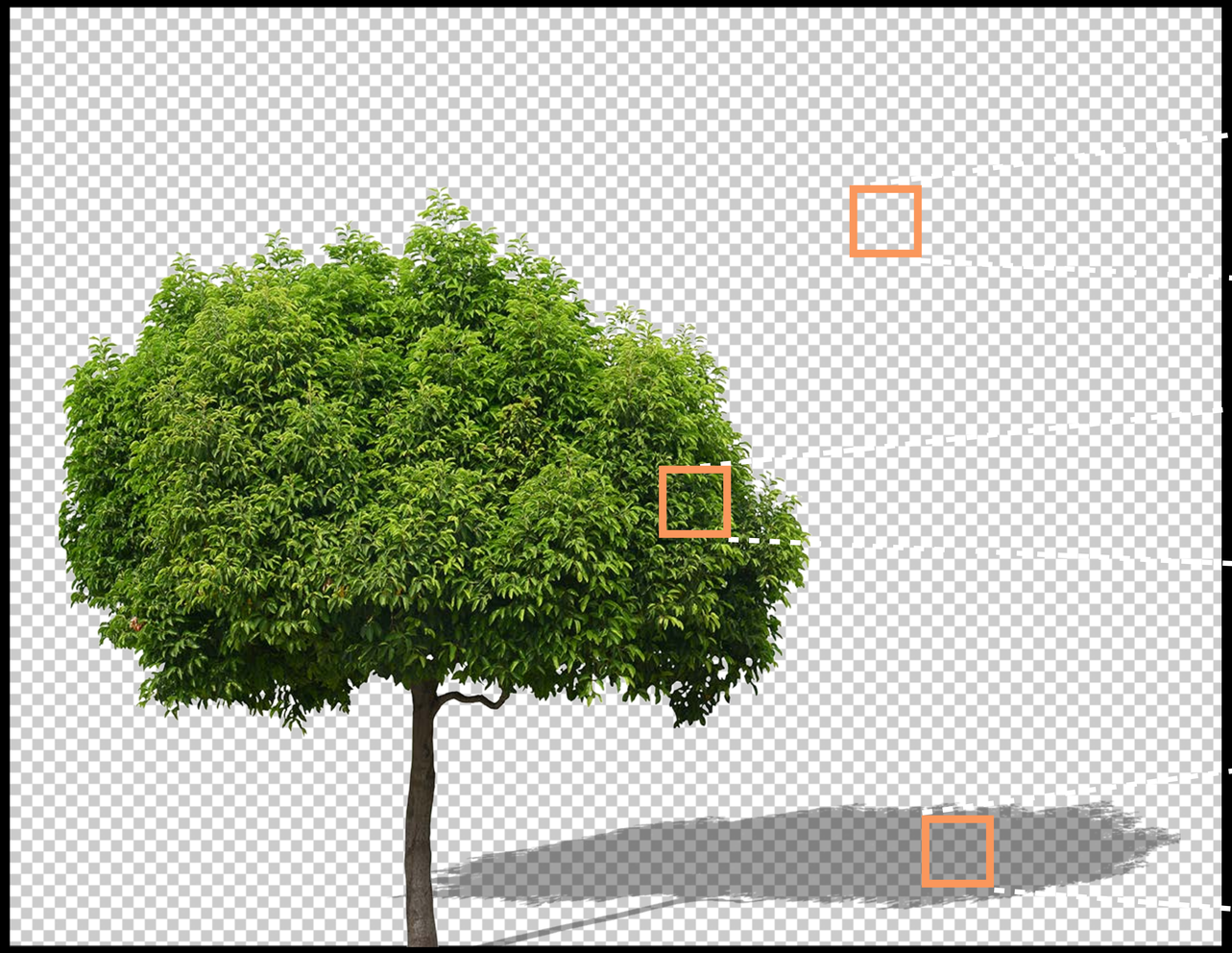

WWDC19

#WWDC19

HEVC Video with Alpha

Shiva Sundar, CoreMedia Engineering

Alpha Channels



0% opaque
(fully transparent)

100% opaque

50% opaque

Alpha Channels

0% opaque
(fully transparent)

100% opaque

50% opaque

Formats with Alpha Channels

	Still Image
High bitrate Lossless/Mezzanine	PNG, TIFF

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444
Flexible bitrate	HEIF	

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444
Flexible bitrate	HEIF / HEIFS	

Formats with Alpha Channels

	Still Image	Motion Video
High bitrate Lossless/Mezzanine	PNG, TIFF	Apple ProRes 4444
Flexible bitrate	HEIF / HEIFS	HEVC with Alpha

Platforms

iOS

iOS 13

tvOS

tvOS 13

macOS

macOS Catalina

Use Cases

Use Cases

Use Cases

Video Overlays

Use Cases

Video Overlays

Use Cases

Video Overlays

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Background Removal

Use Cases

Use Cases

Use Cases

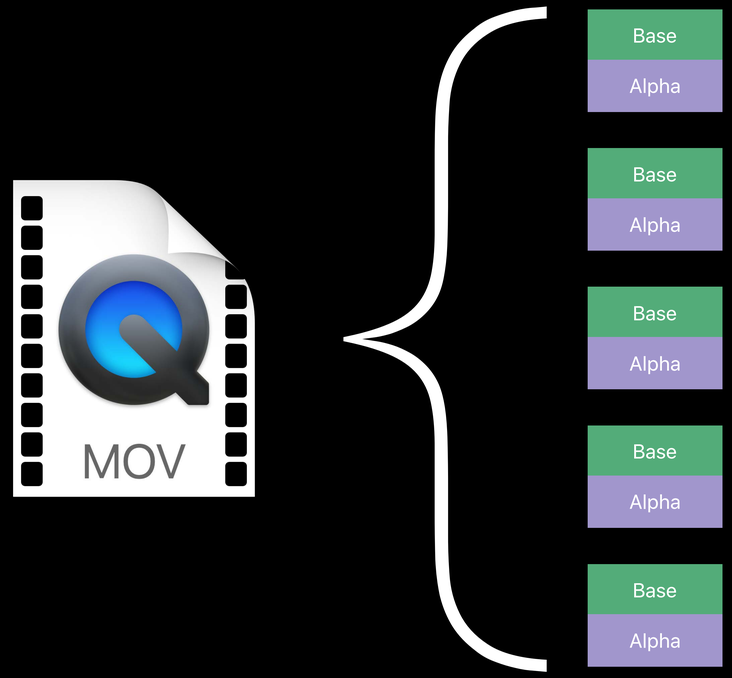
Use Cases

Use Cases

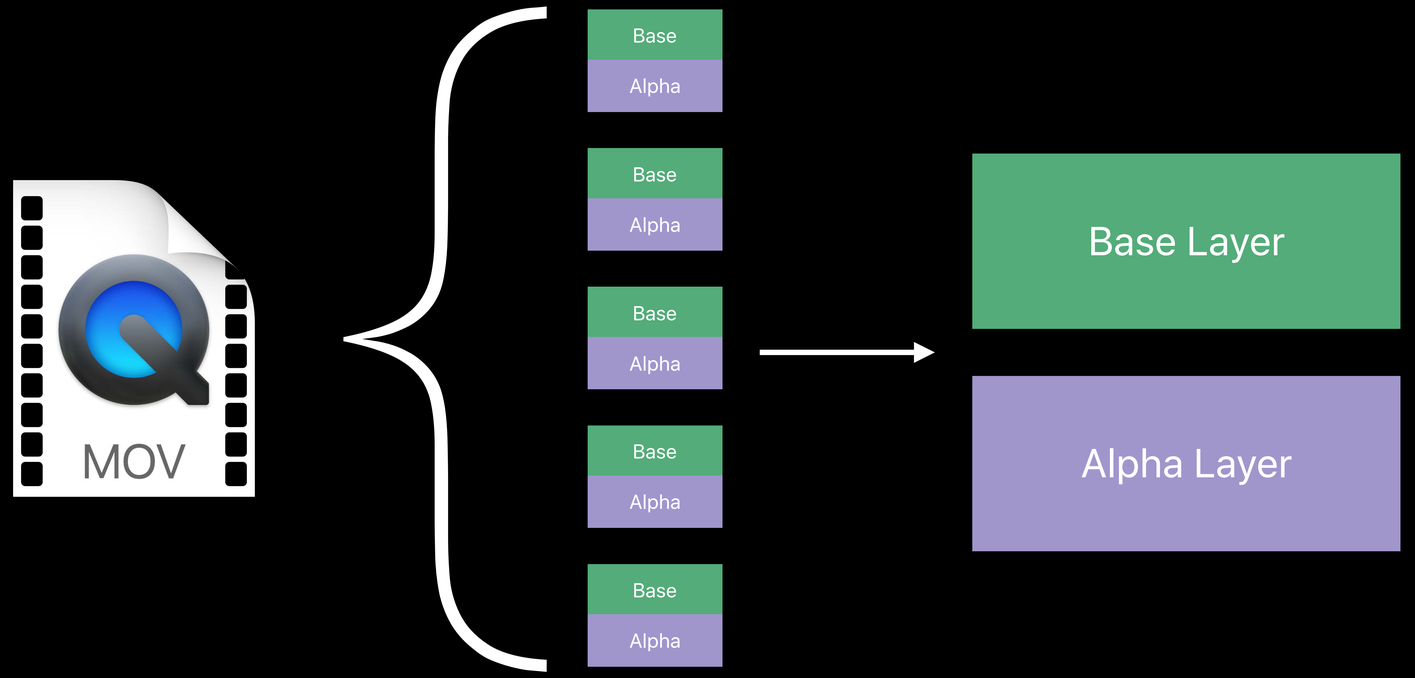
Use Cases

How it works

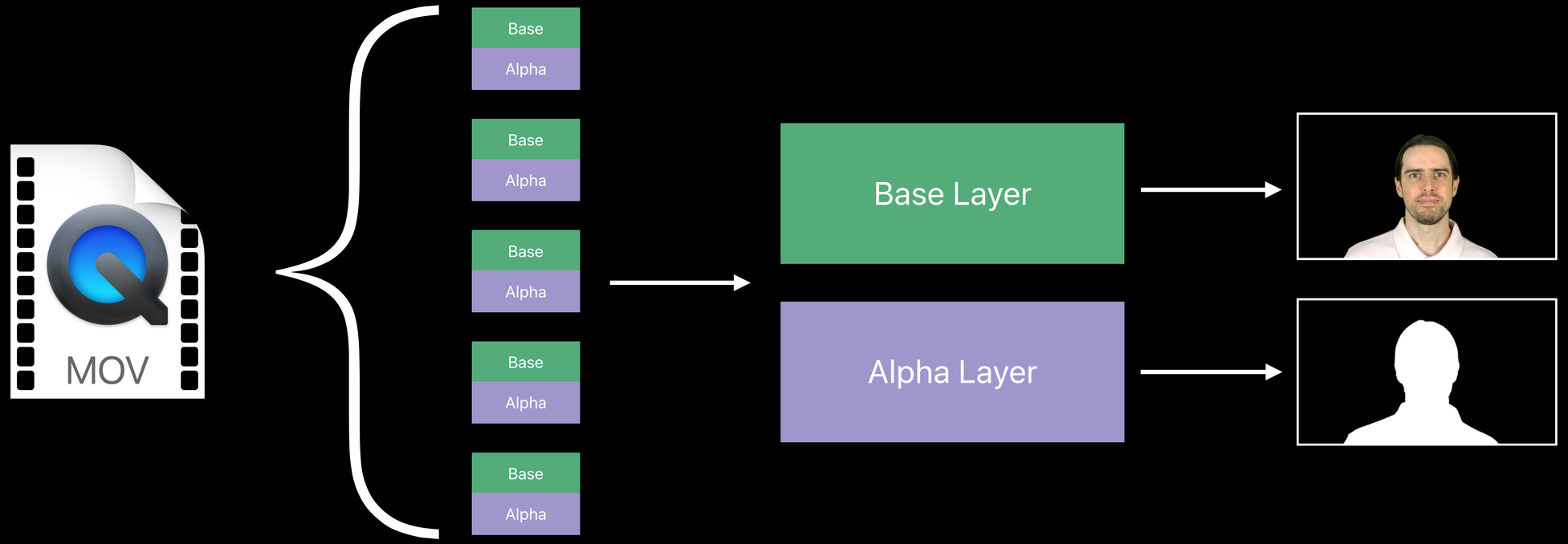
How it works



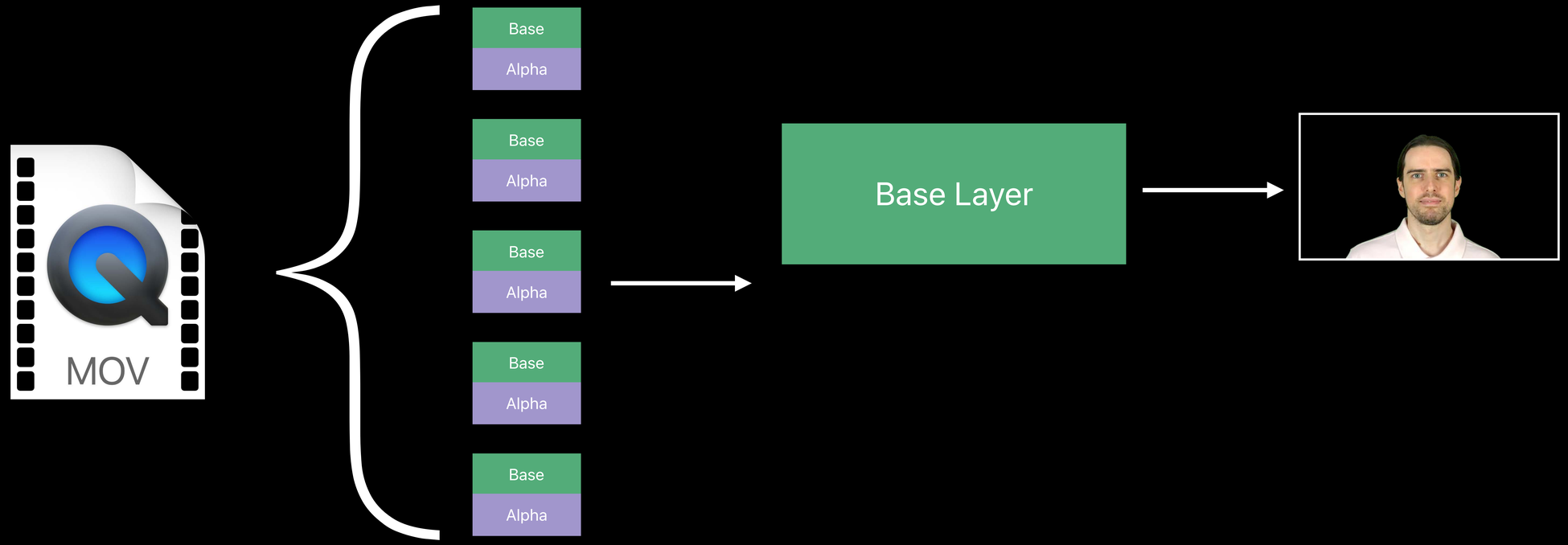
How it works


MOV
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base Layer
Alpha Layer

How it works


MOV
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base Layer
Alpha Layer

How it works


MOV
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base Layer

Encoding

Encoding

Encoding

Any Video with Alpha

HEVC with Alpha

```
AVAssetExportPresetHEVC…WithAlpha
```

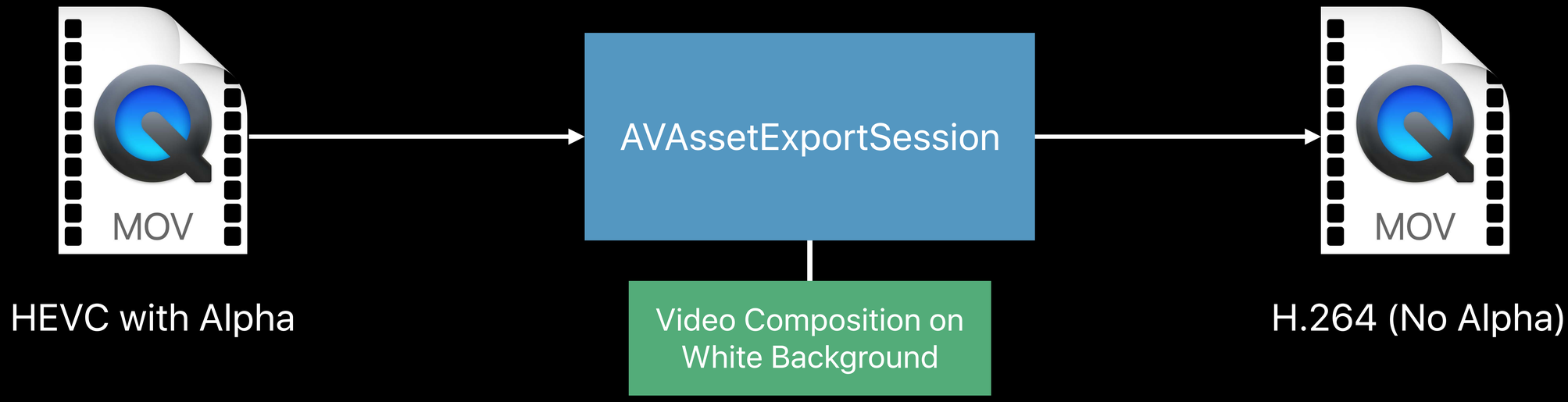
Encoding

Any Video with Alpha

HEVC with Alpha

```
AVAssetExportPresetHEVC...WithAlpha
```

Encoding




```
AVAssetExportPresetHighestQuality
```

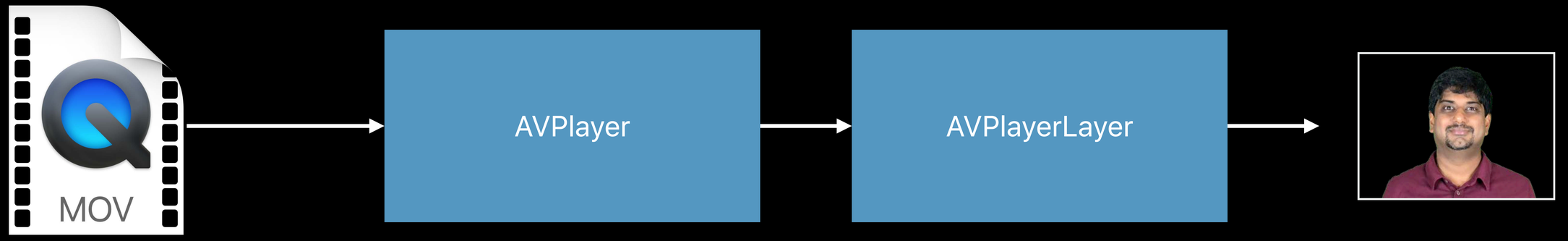
Encoding

Video Frames with Alpha

Sample Buffers

Playback

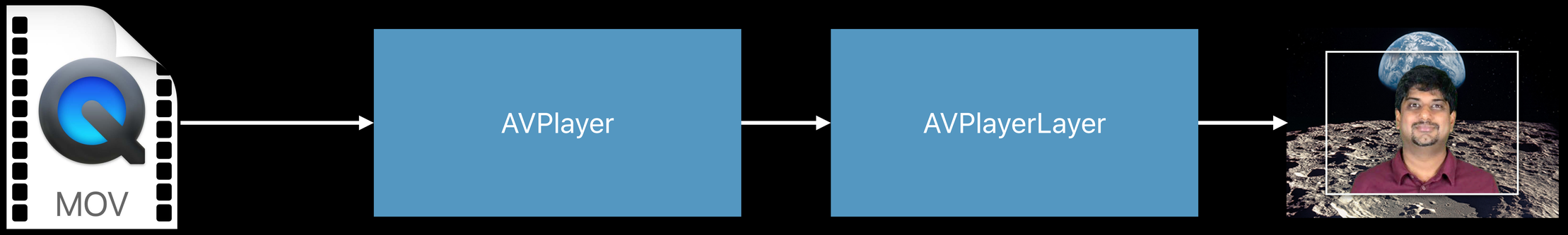
Playback



HEVC with Alpha

Displayed with
Transparent Background

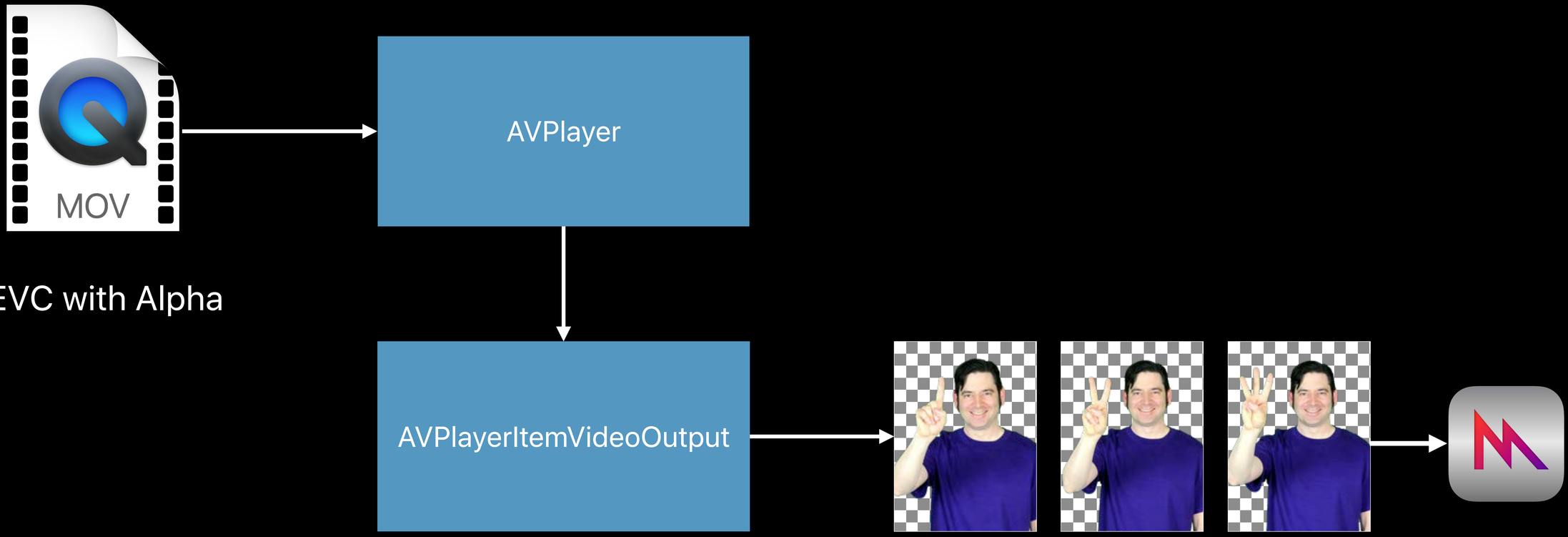
Playback



HEVC with Alpha

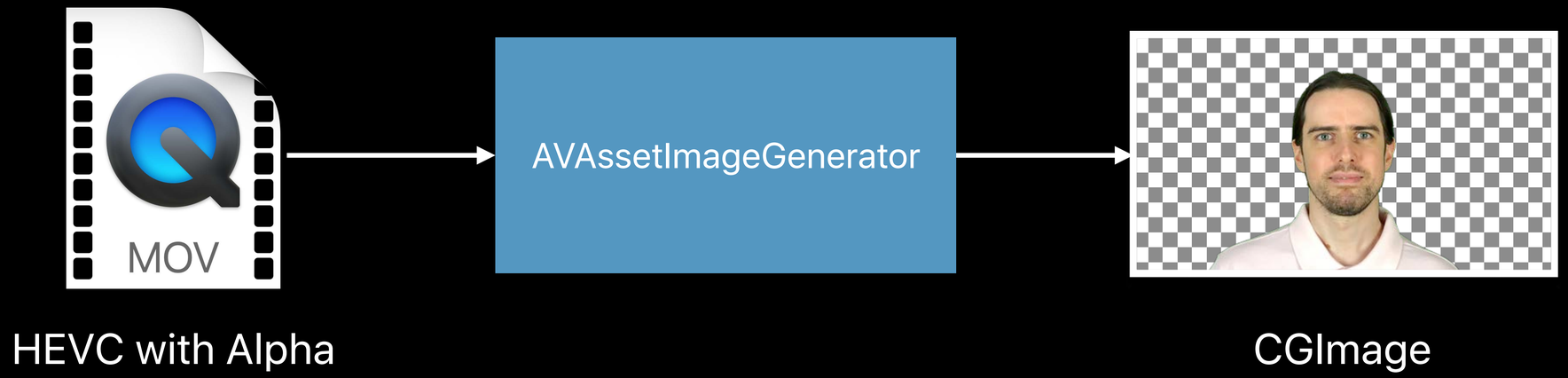
Displayed with
Transparent Background

Playback


MOV
AVPlayer
HEVC with Alpha
AVPlayerItemVideoOutput
Video Frames with Alpha

Decoding

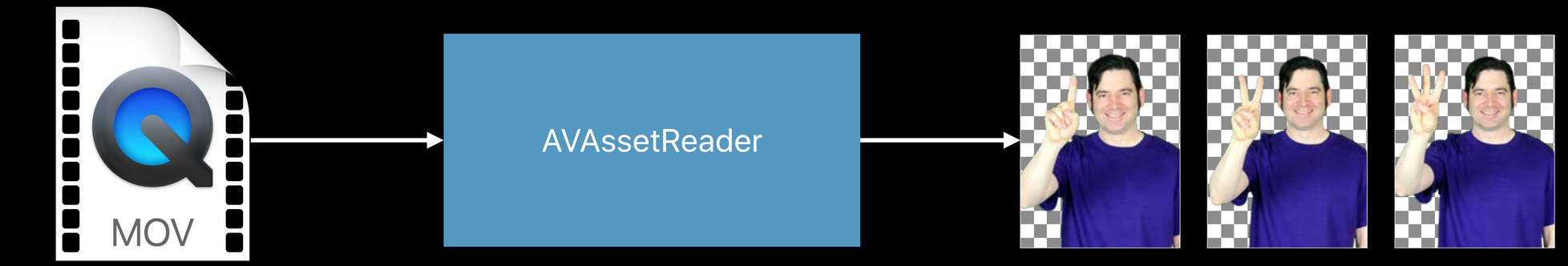
Decoding



HEVC with Alpha

CGImage

Decoding



HEVC with Alpha

Video Frames with Alpha

Decoding

Sample Buffers

Video Frames with Alpha

APIs

Encoding

AVAssetWriter

AVAssetExportSession

VTCompressionSession

Playback

AVPlayer

AVPlayerItemVideoOutput

Decoding

AVAssetImageGenerator

AVAssetReader

VTDecompressionSession

For Example

```
// AVAssetWriterInput - CodecType
import AVFoundation
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

```
// AVAssetWriterInput - CodecType
import AVFoundation
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Both Premultiplied Alpha and Straight Alpha are supported

Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Both Premultiplied Alpha and Straight Alpha are supported

Premultiplied Alpha is recommended

```swift
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
     AVVideoWidthKey: width,
     AVVideoHeightKey: height
     kVTCompressionPropertyKey_AlphaChannelMode:
        kVTAlphaChannelMode_PremultipliedAlpha
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
```

```swift
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
     AVVideoWidthKey: width,
     AVVideoHeightKey: height
     kVTCompressionPropertyKey_AlphaChannelMode:
        kVTAlphaChannelMode_PremultipliedAlpha
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
```

```swift
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
     AVVideoWidthKey: width,
     AVVideoHeightKey: height
     kVTCompressionPropertyKey_AlphaChannelMode:
        kVTAlphaChannelMode_PremultipliedAlpha
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
```

Bitrate and Quality

Can control bitrate of base layer (bits per second)

Can control quality of alpha layer (0 to 1)

```
// AVAssetWriterInput – BitRate and Quality tradeoffs
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey:
        [kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
         kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality]
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

```
// AVAssetWriterInput – BitRate and Quality tradeoffs
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey:
        [kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
         kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality]
    ] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
```

```
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```

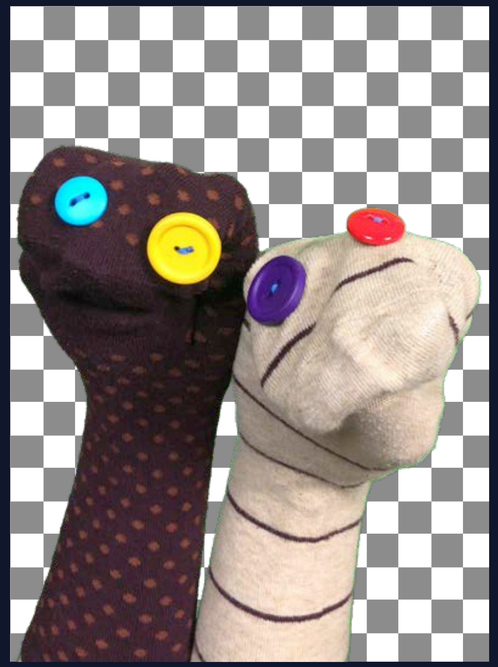
```
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```

```swift
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```

```swift
// AVAssetExportSession - compatibility
import AVFoundation
AVAssetExportSession.determineCompatibility(
    ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
    with: asset,
    outputFileType: .mov) { compatible in
    if compatible {
        // Export
    } else {
        print("Export Session failed compatibility check")
        // Handle failure
    }
}
```

```swift
// AVAssetExportSession - compatibility
import AVFoundation
AVAssetExportSession.determineCompatibility(
    ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
    with: asset,
    outputFileType: .mov) { compatible in
    if compatible {
        // Export
    } else {
        print("Export Session failed compatibility check")
        // Handle failure
    }
}
```

```
// AVAssetExportSession – backwards compatibility
import AVFoundation
// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace
let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
                                                 prototypeInstruction: prototypeInstruction)

// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
   // Handle completion
}
```



```
// AVAssetExportSession – backwards compatibility
import AVFoundation
// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace
let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
                                                 prototypeInstruction: prototypeInstruction)
```

```
// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
    // Handle completion
}
```

Interoperability Profile

Single video track with ‘hvc1’ codec type

- Two layers
  - Base layer: main profile; 4:2:0 video-range; nuh_layer_id 0
  - Alpha layer: main profile; 4:2:0 full-range, neutral chroma channel; nuh_layer_id 1
- Base and alpha layers share one VPS
  - VPS extension required — indicates presence of alpha channel
- Base and alpha layers have distinct PPS and SPS with different IDs
- Base and alpha layers must have identical frame types and dependency structure
- SEI alpha_channel_info message in hvcC in video format description
  - Indicates premultiplied alpha or straight alpha

HEVC Video with Alpha

Advanced lossy compression technology

Supported in iOS 13, tvOS 13, macOS Catalina

Hardware accelerated on recent devices

Integrated with media APIs

Integrated in Safari

More Information

https://developer.apple.com/wwdc19/506


WWDC19