

#WWDC19

# Advances in Camera Capture and Photo Segmentation

Brad Ford, Camera Software

Jacob Schack Vestergaard, Camera Software

David Hayward, Core Image

Multi-Camera Capture

Semantic Segmentation for Photos

# Multi-Camera Capture

**MultiCam**

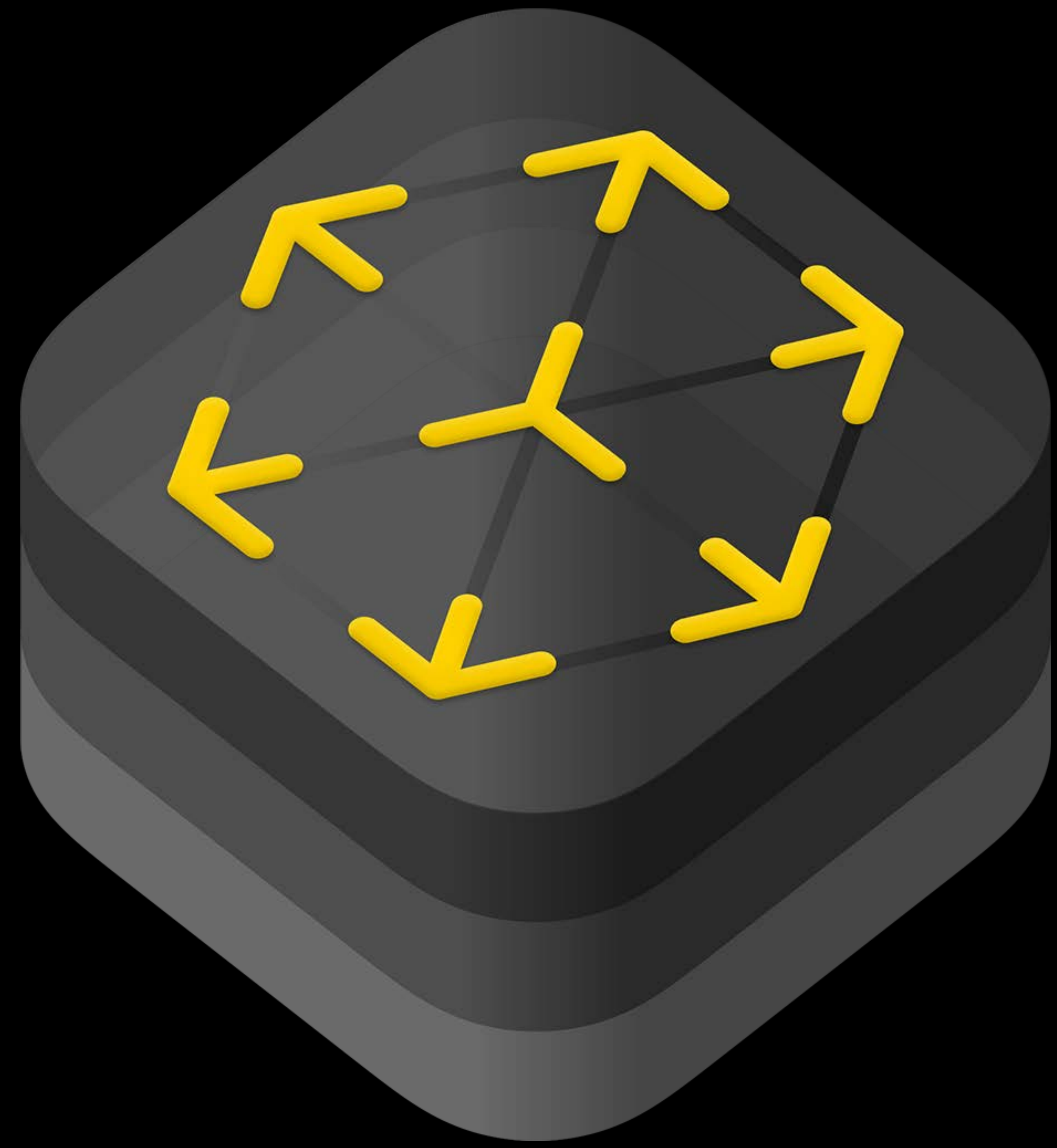












# Multi-Camera Capture Support by Platform



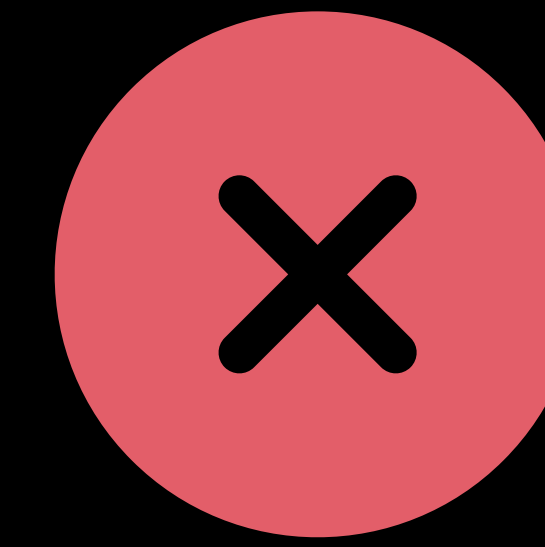
OS X Lion  
(2011)



# Multi-Camera Capture Support by Platform



OS X Lion  
(2011)



iOS 12  
(2018)

# Multi-Camera Capture Support





# Multi-Camera Capture Support



# Multi-Camera Capture Support





# Multi-Camera Capture

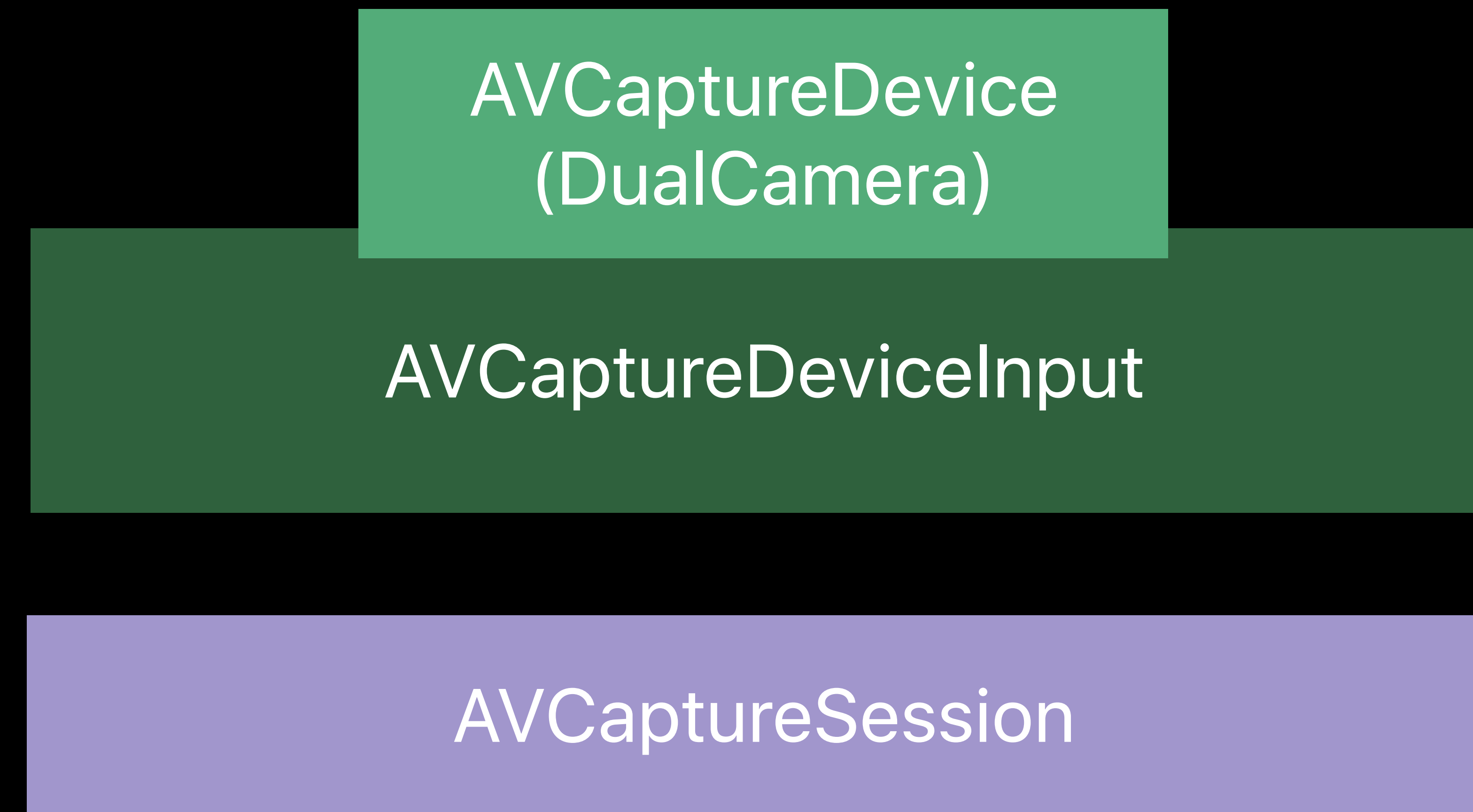
Session building

# **AV Foundation Capture Classes**

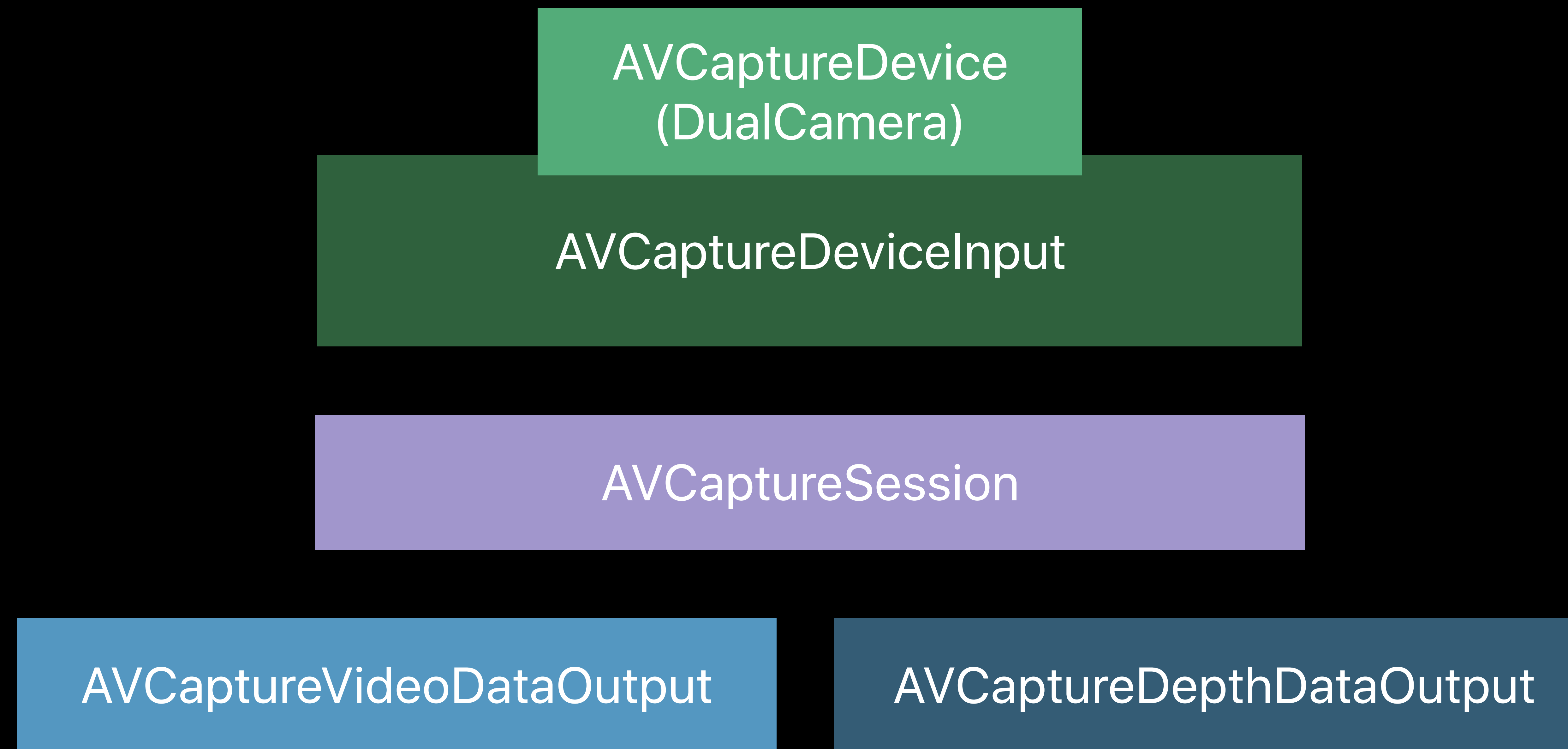
# AV Foundation Capture Classes

AVCaptureSession

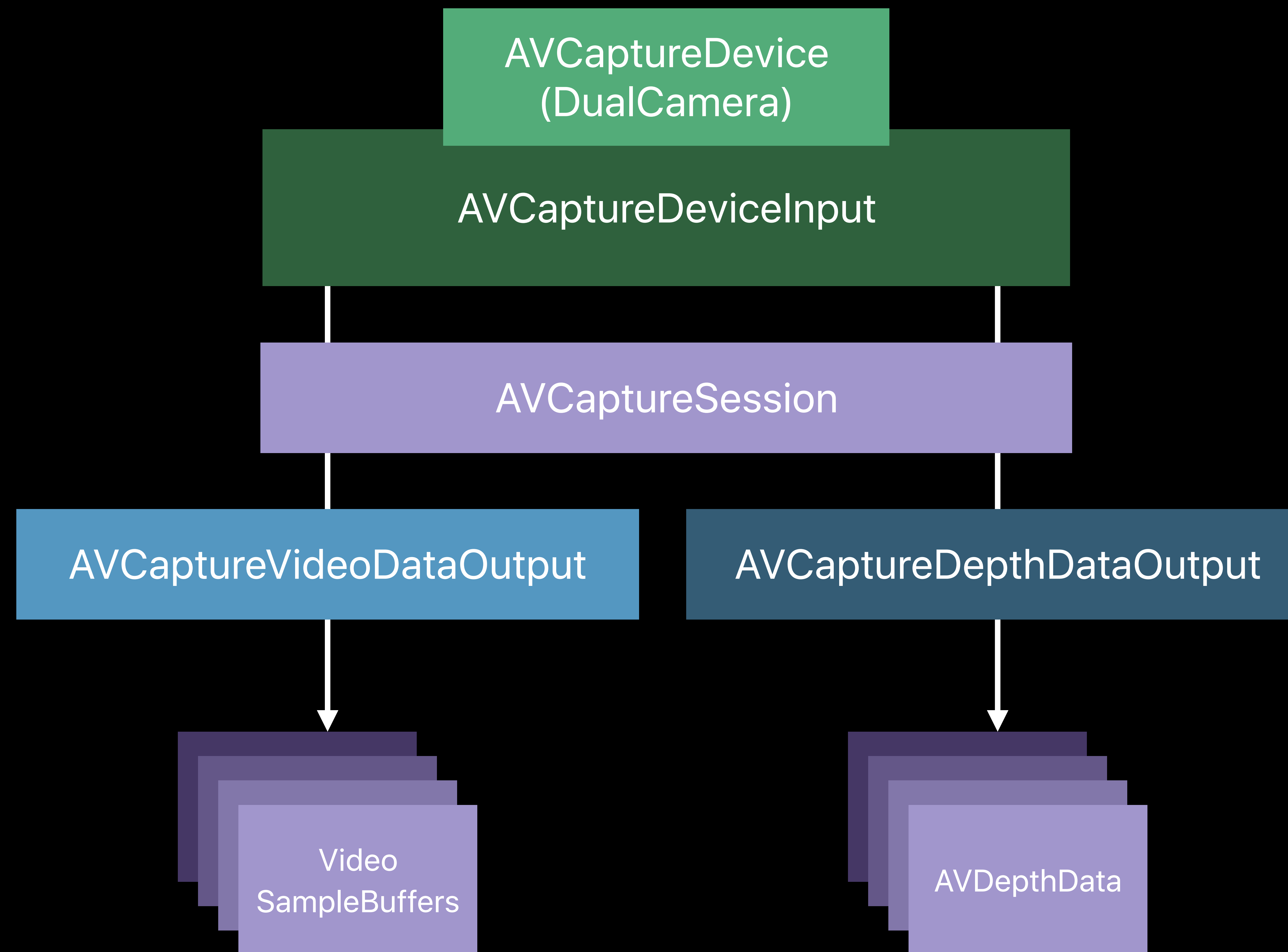
# AV Foundation Capture Classes



# AV Foundation Capture Classes



# AV Foundation Capture Classes



# Introducing `AVCaptureMultiCamSession`



NEW

Multiple `AVCaptureDeviceInputs`

Multiple `AVCaptureOutputs` of the same type

Multiple `AVCaptureVideoPreviewLayers`

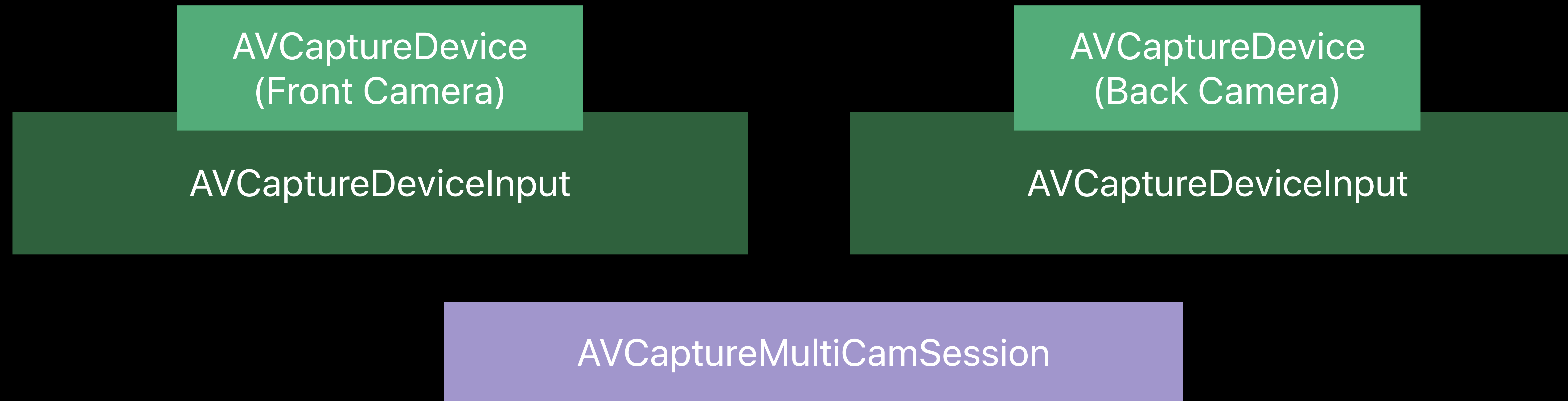
Not a replacement for `AVCaptureSession`

# AVCaptureMultiCamSession Example

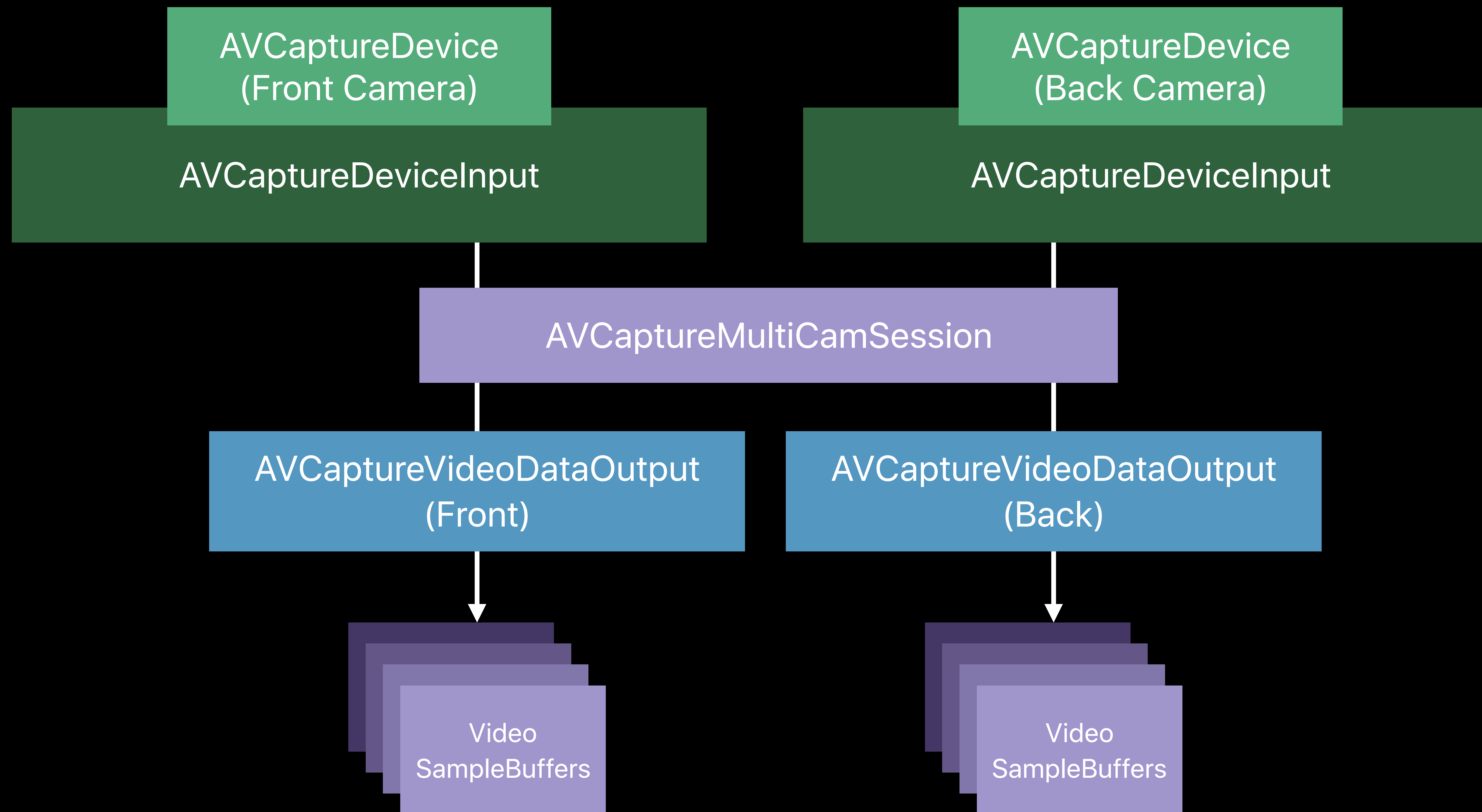
AVCaptureMultiCamSession



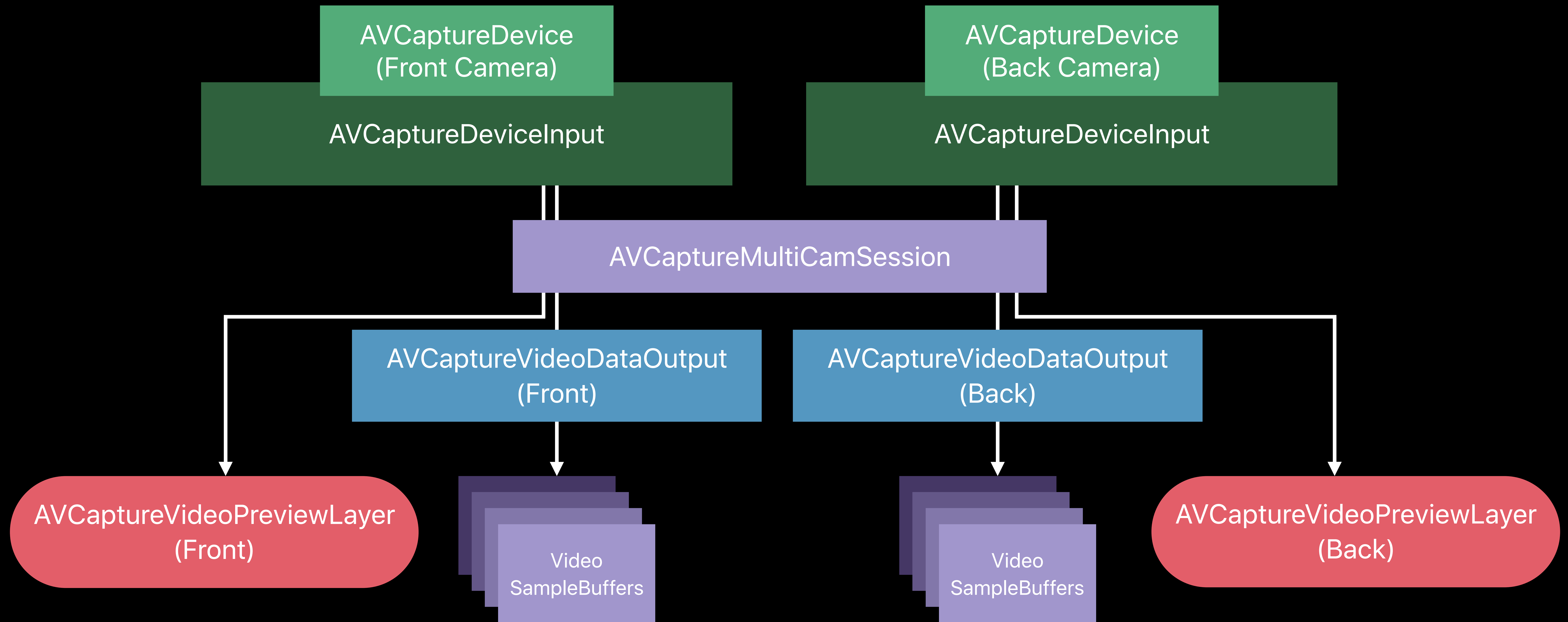
# AVCaptureMultiCamSession Example



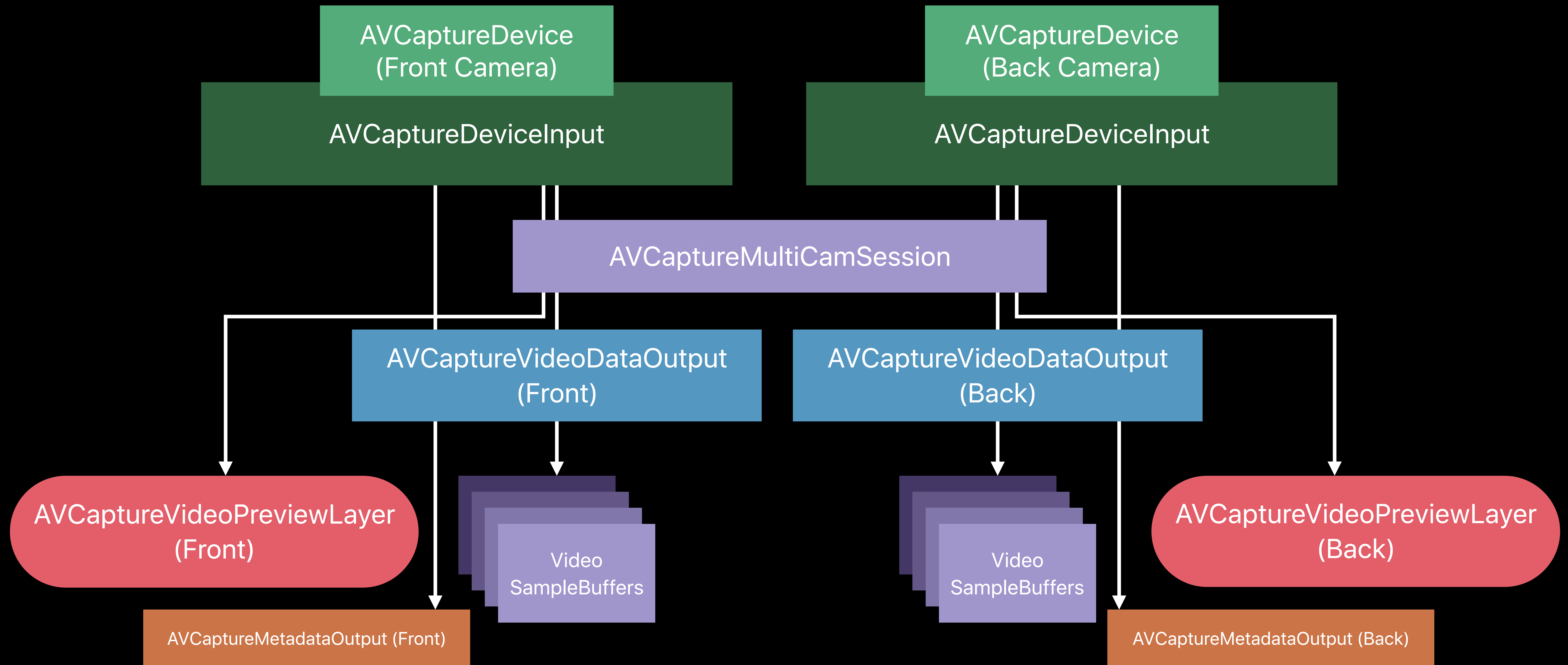
# AVCaptureMultiCamSession Example



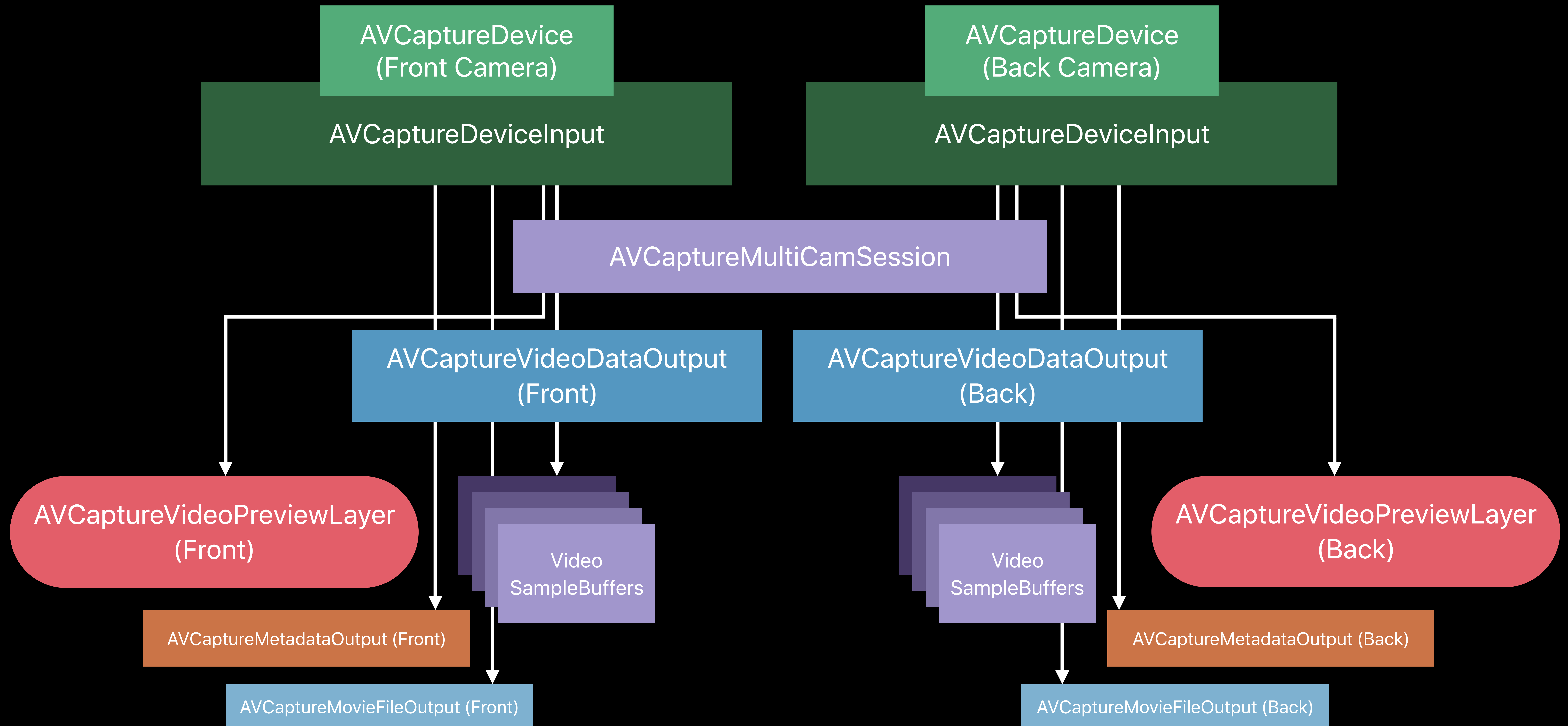
# AVCaptureMultiCamSession Example



# AVCaptureMultiCamSession Example

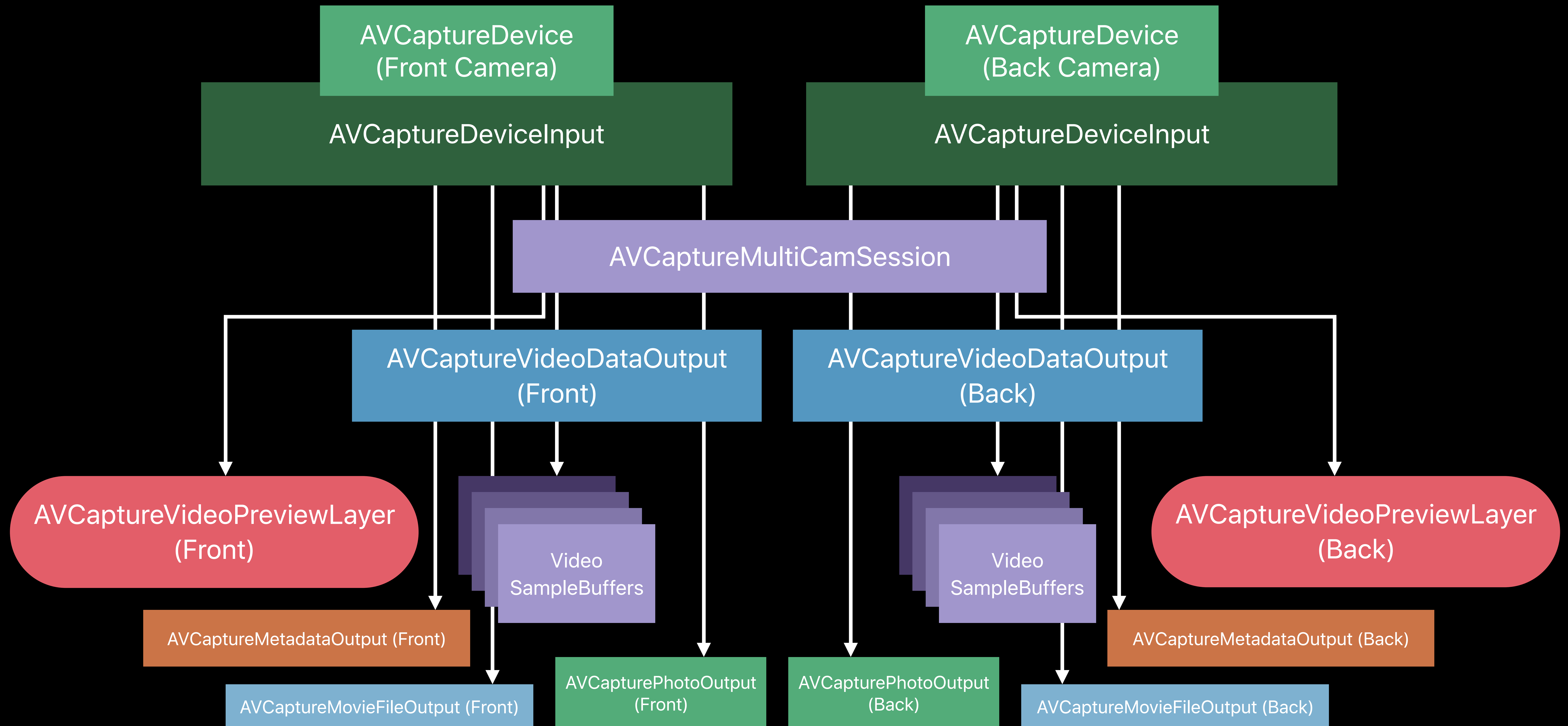


# AVCaptureMultiCamSession Example





# AVCaptureMultiCamSession Example

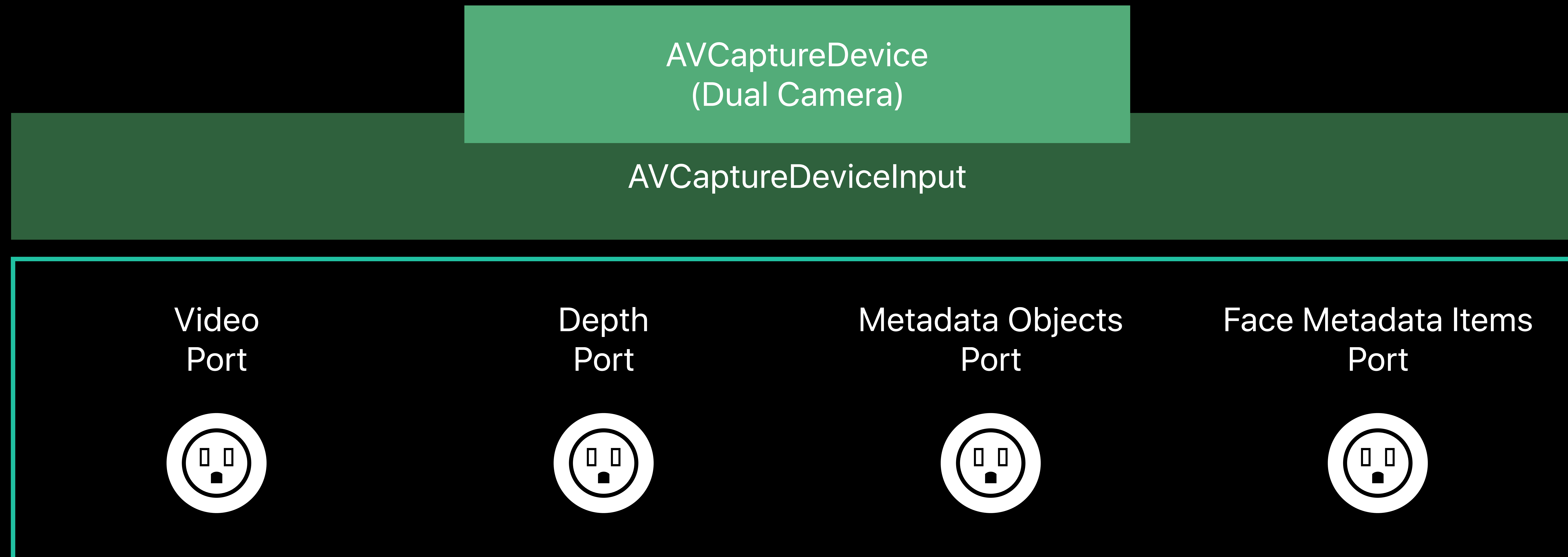


# Anatomy of a Connection

AVCaptureDevice  
(Dual Camera)

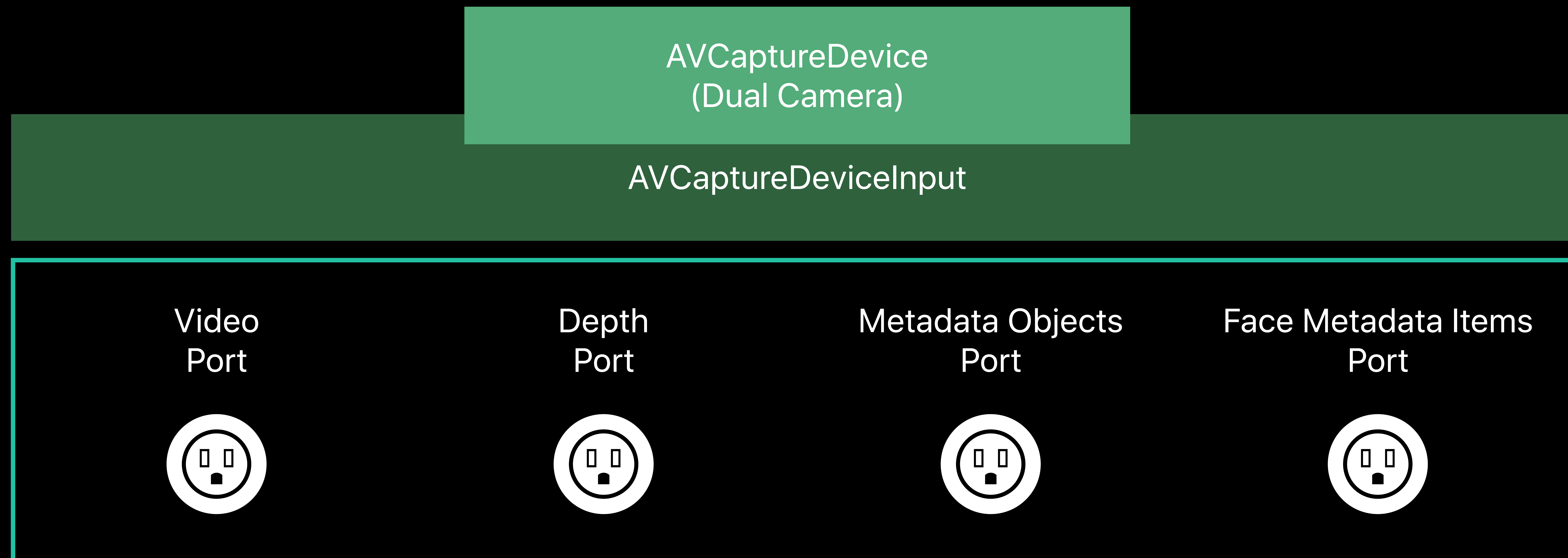
AVCaptureDeviceInput

# Anatomy of a Connection



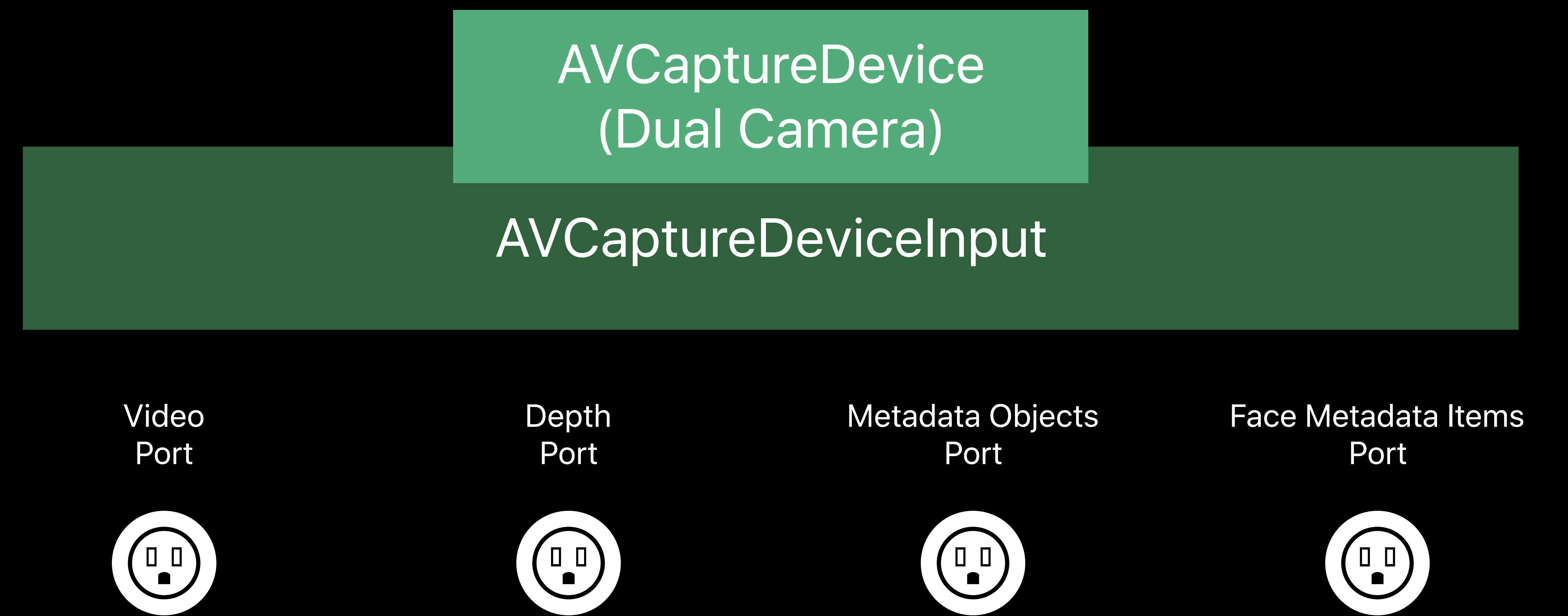


# Anatomy of a Connection



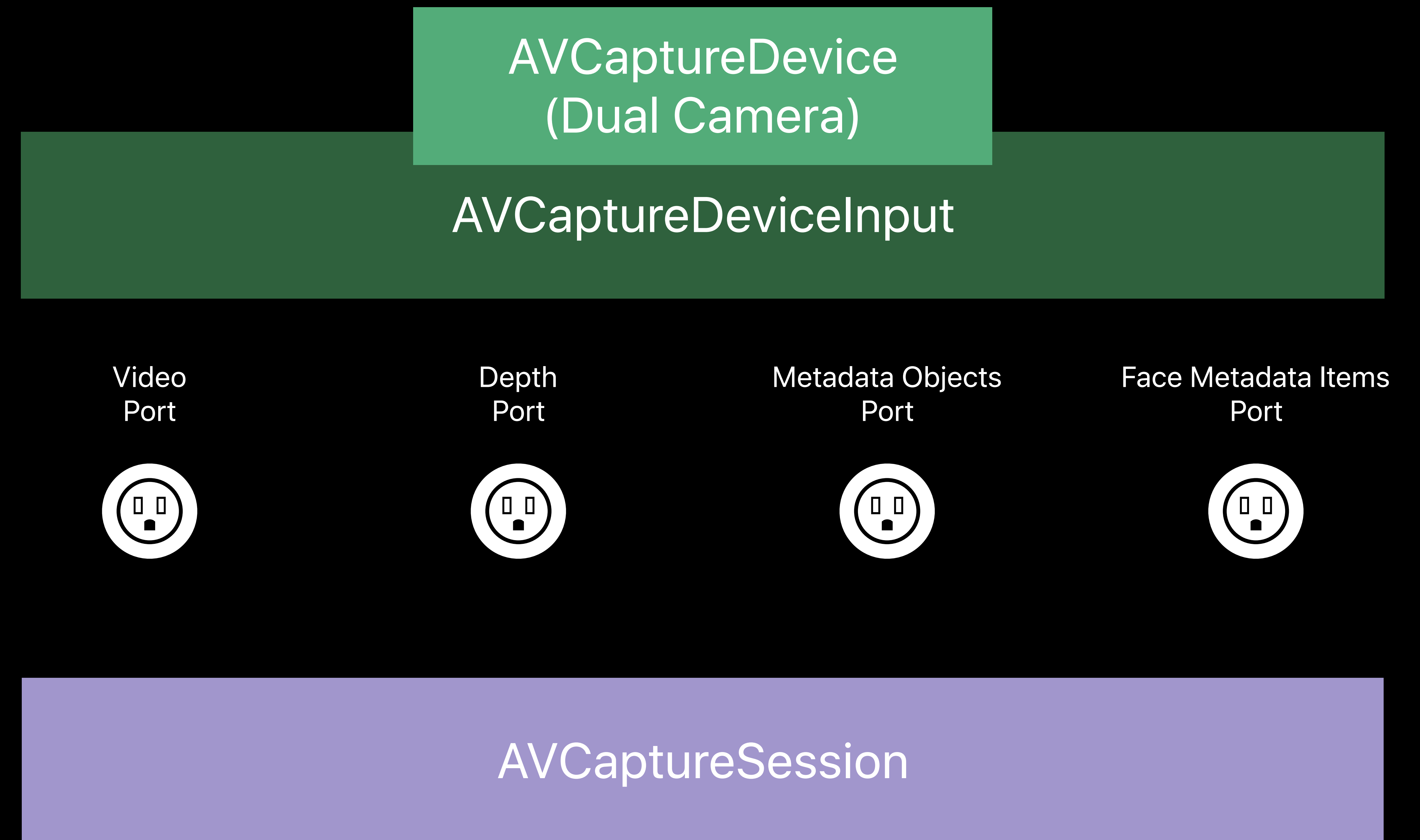
```
let ports: [AVCaptureInput.Port] = dualCameraDeviceInput.ports
```

# Anatomy of a Connection

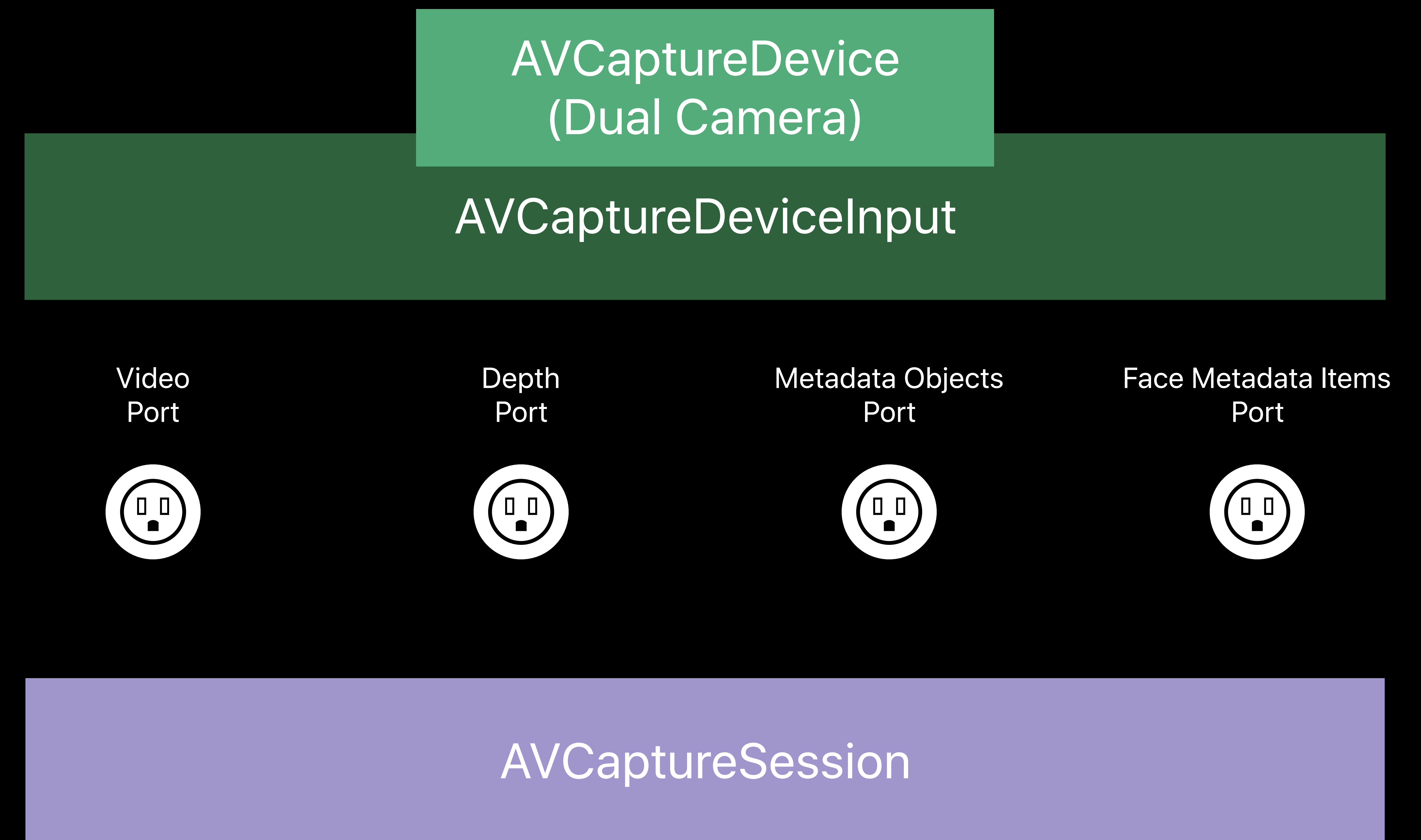


# Anatomy of a Connection

```
session.addInput(dualCameraInput)
```



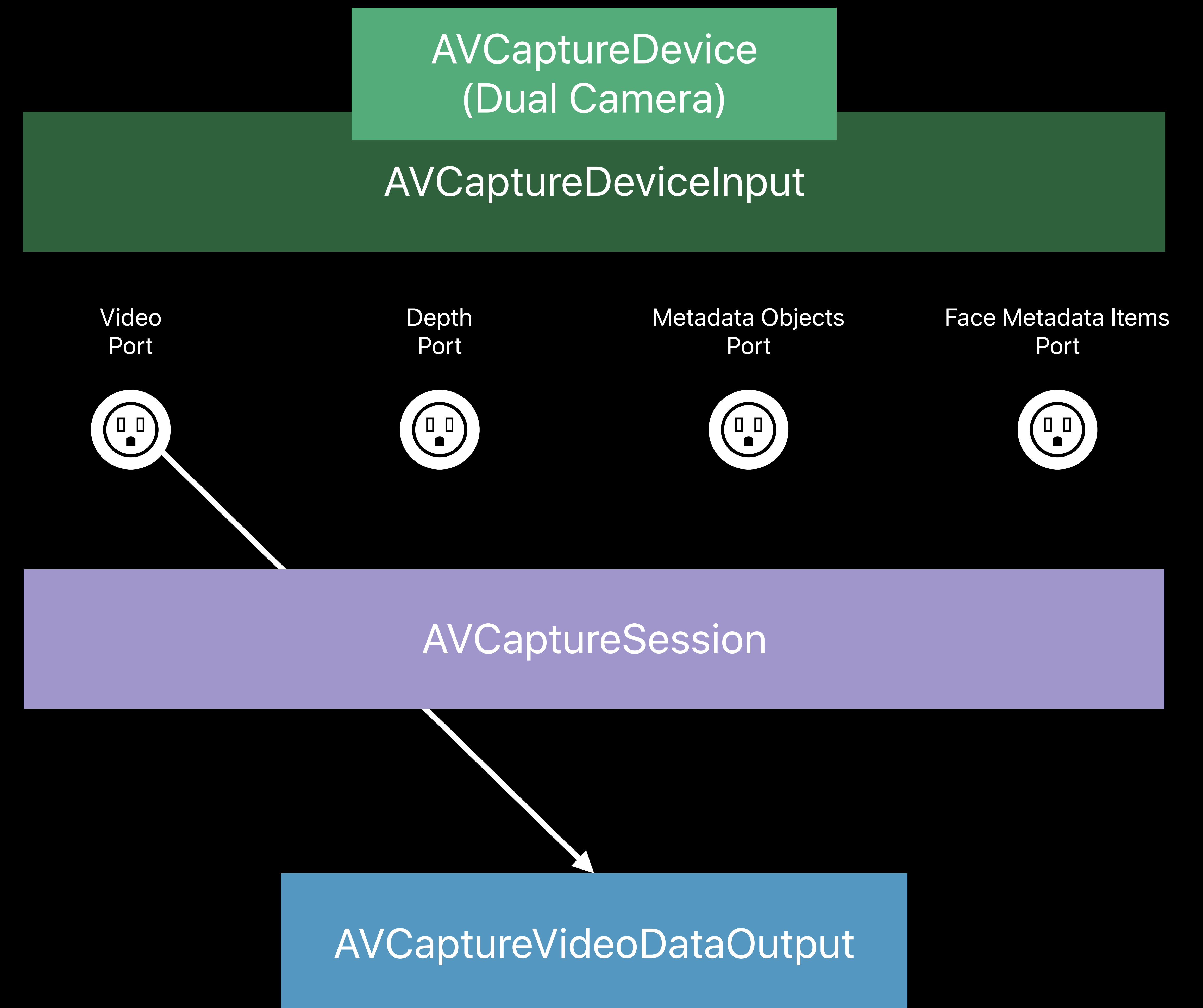
# Anatomy of a Connection



```
session.addOutput(videoDataOutput)
```

AVCaptureVideoDataOutput

# Anatomy of a Connection



# MultiCam Session Building Best Practices

When adding inputs and outputs

# MultiCam Session Building Best Practices

When adding inputs and outputs

- Use `addInputWithNoConnections` or `addOutputWithNoConnections`

# MultiCam Session Building Best Practices

When adding inputs and outputs

- Use `addInputWithNoConnections` or `addOutputWithNoConnections`

When adding video preview layers



# MultiCam Session Building Best Practices

When adding inputs and outputs

- Use `addInputWithNoConnections` or `addOutputWithNoConnections`

When adding video preview layers

- Use `AVCaptureVideoPreviewLayer.setSessionWithNoConnections()`

# MultiCam Session Building Best Practices

Then create and add explicit connections

```
let backCameraVideoConnection =  
    AVCaptureConnection(inputPorts: [backCameraVideoPort],  
                        output: backVideoDataOutput)  
  
session.addConnection(backCameraVideoConnection)
```

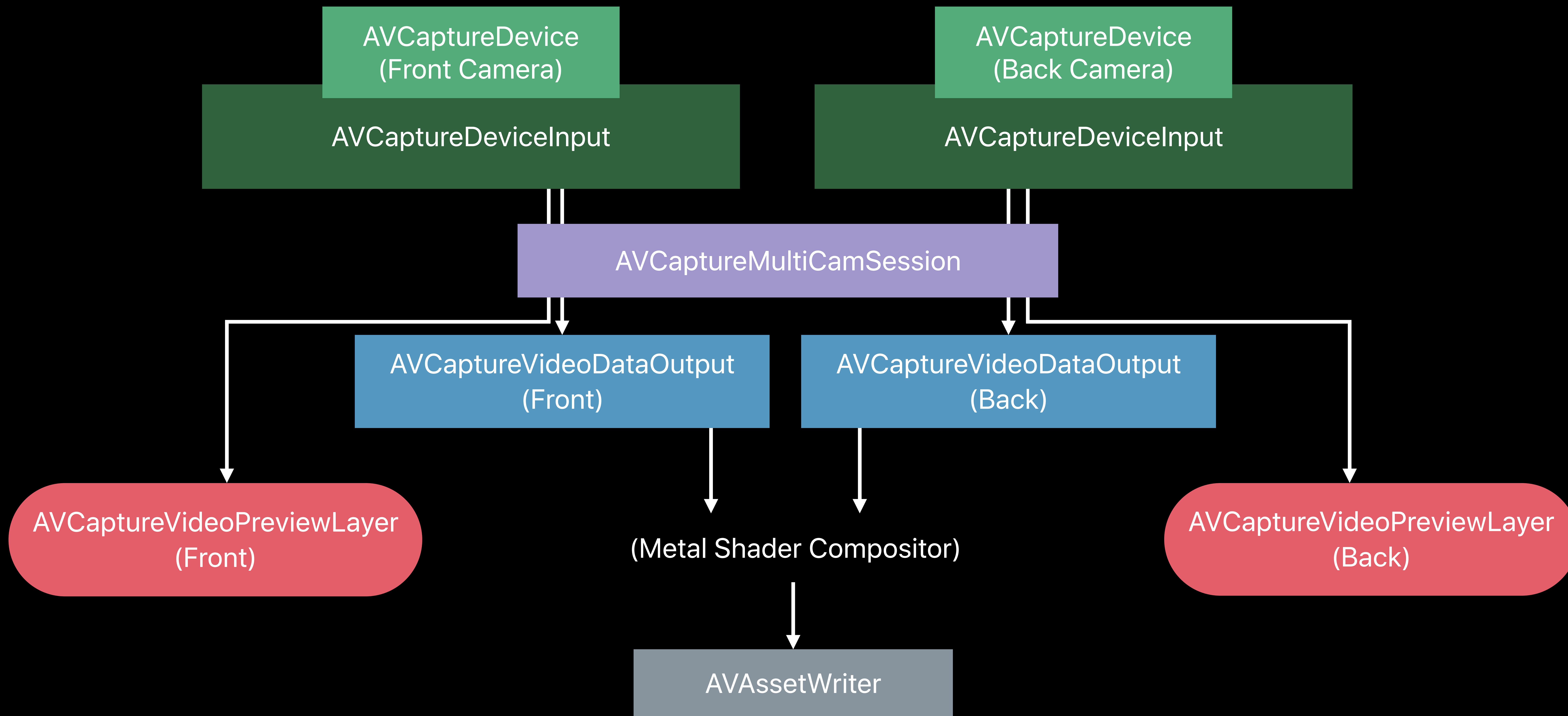


***Demo***

AVMultiCamPiP app

Nikolas Gelo, Camera Software

# AVMultiCamPiP Graph Topology

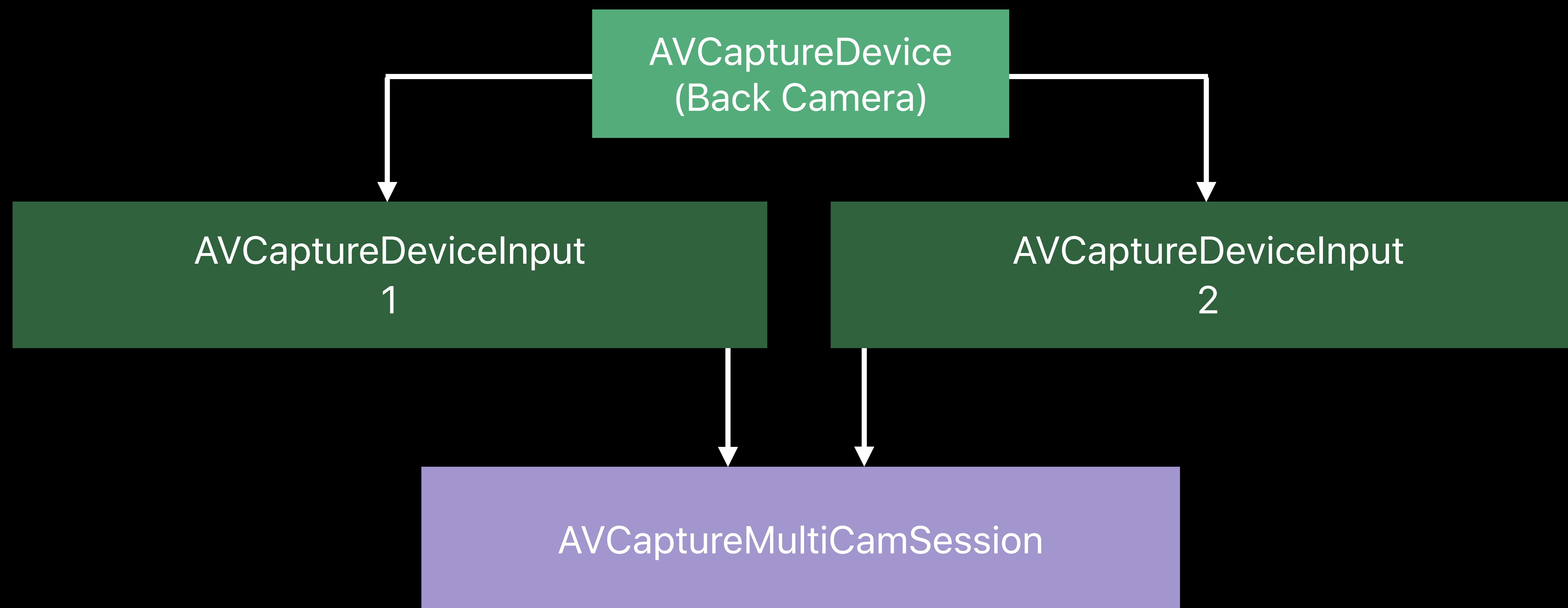
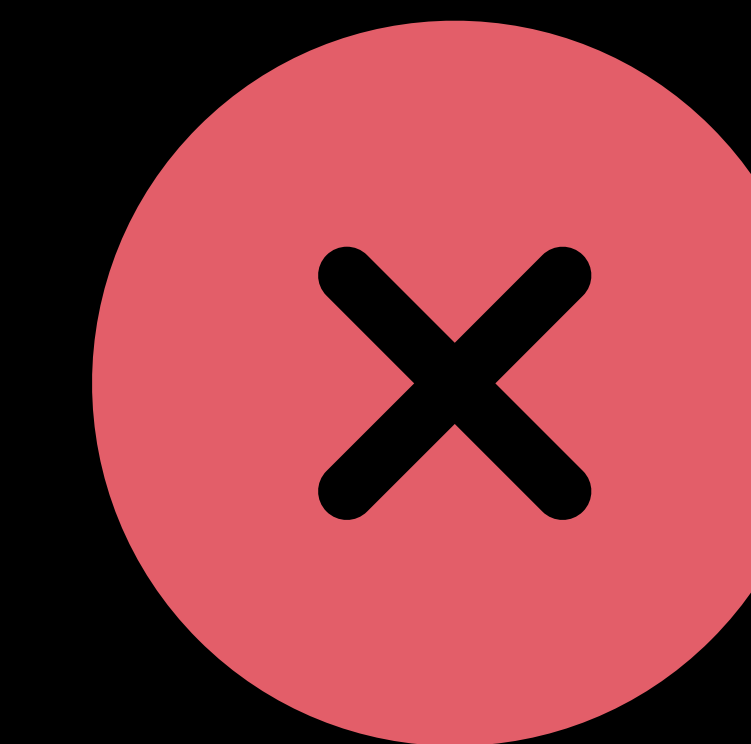




# Multi-Camera Capture

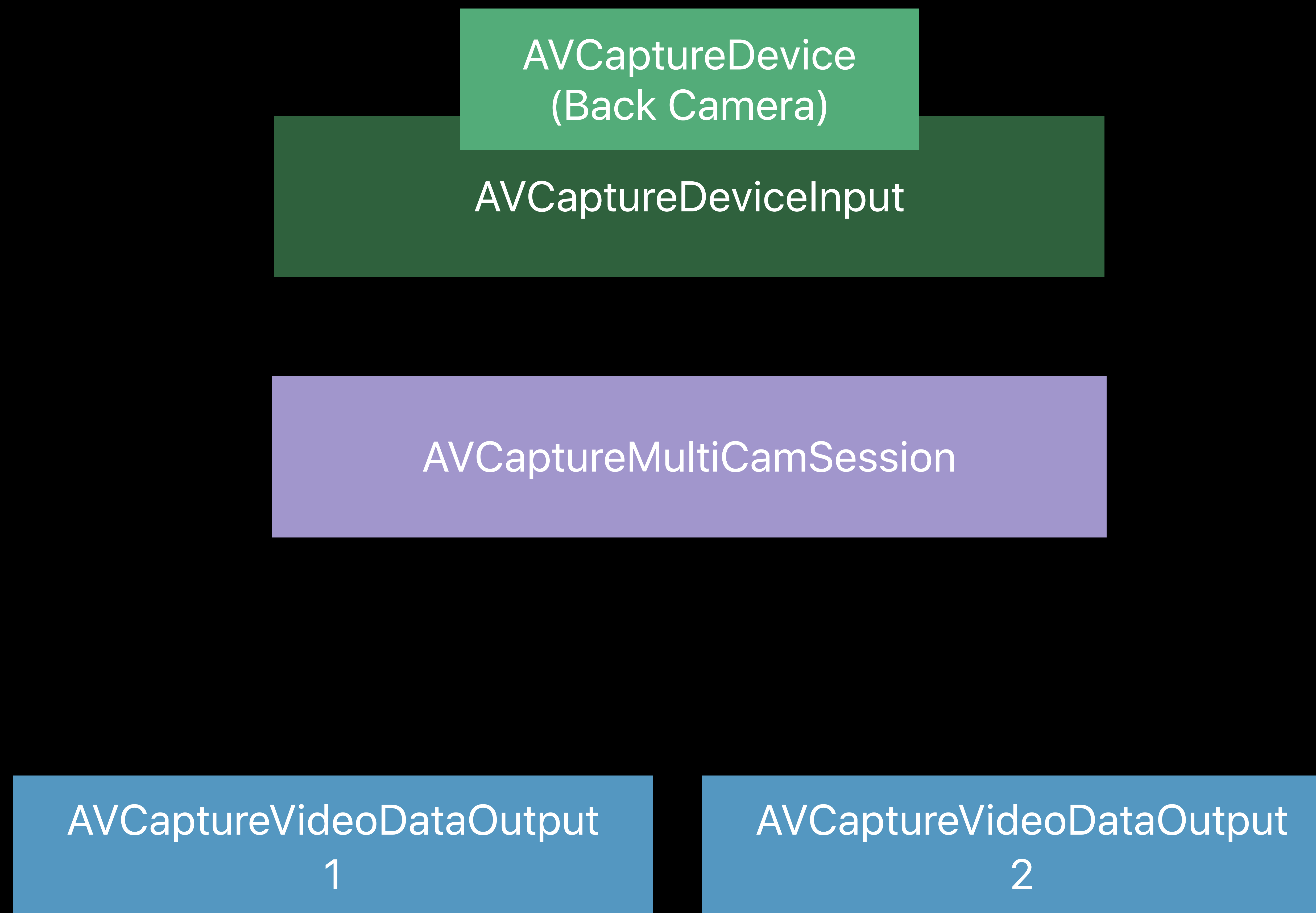
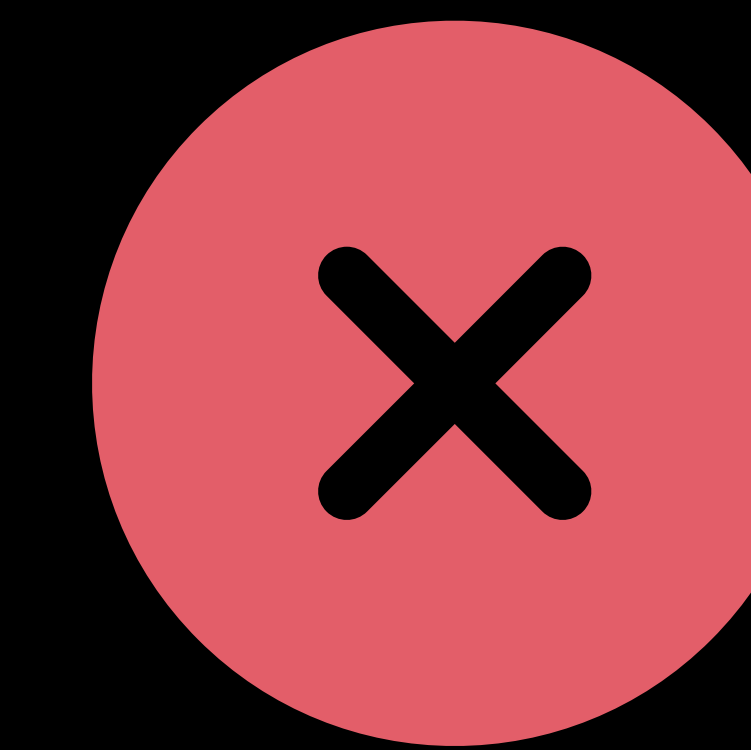
Limitations

# No Camera Cloning at the Input

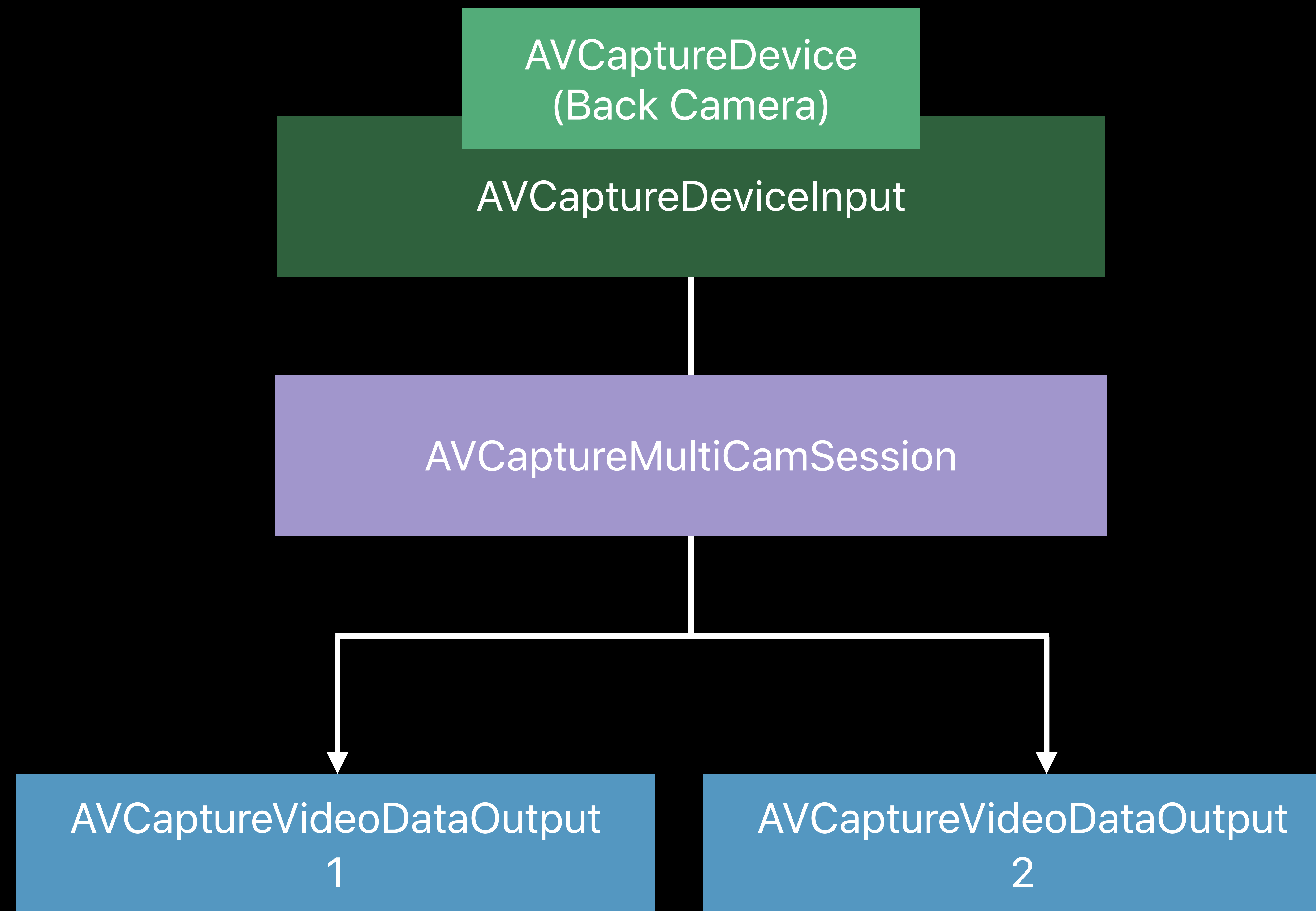
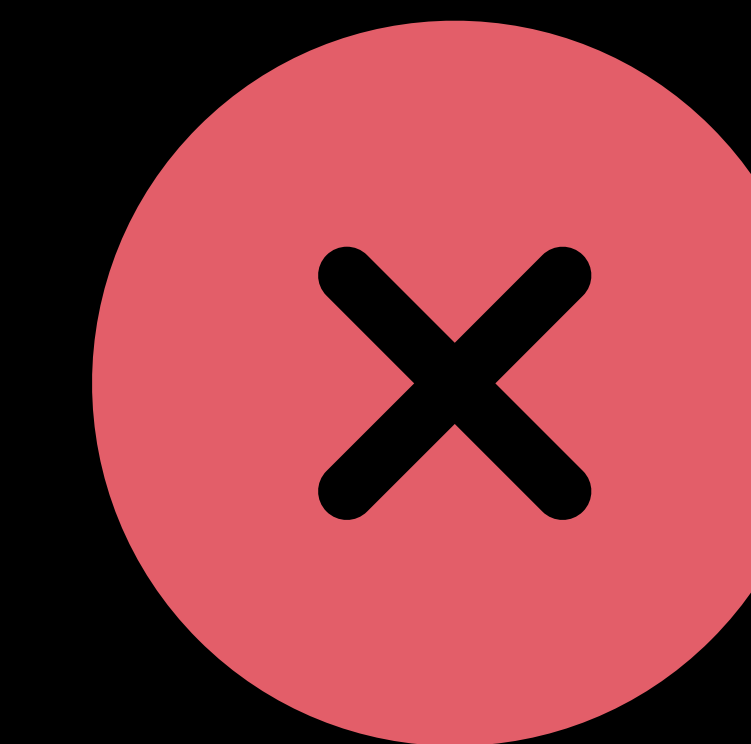




# No Split Personality Camera Output

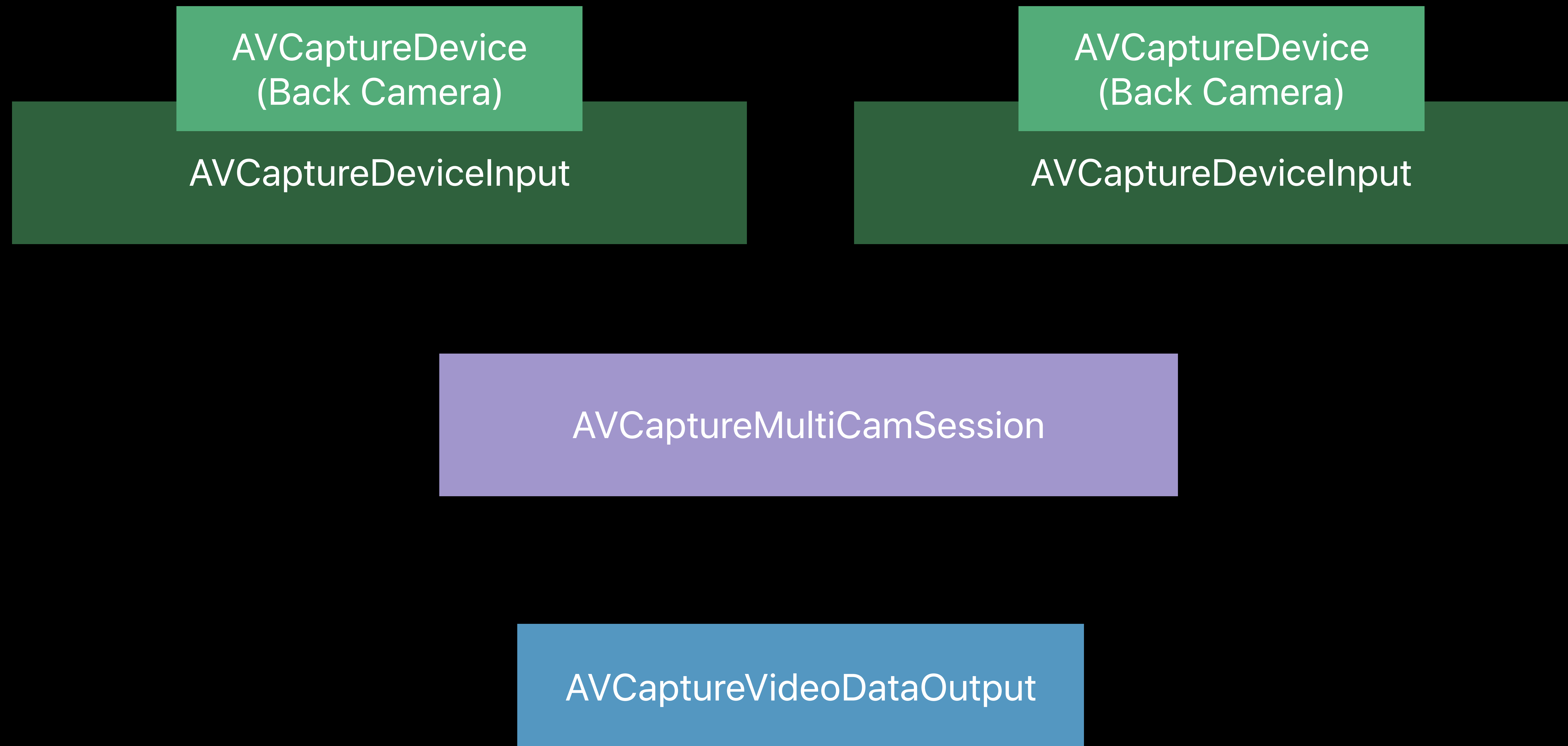
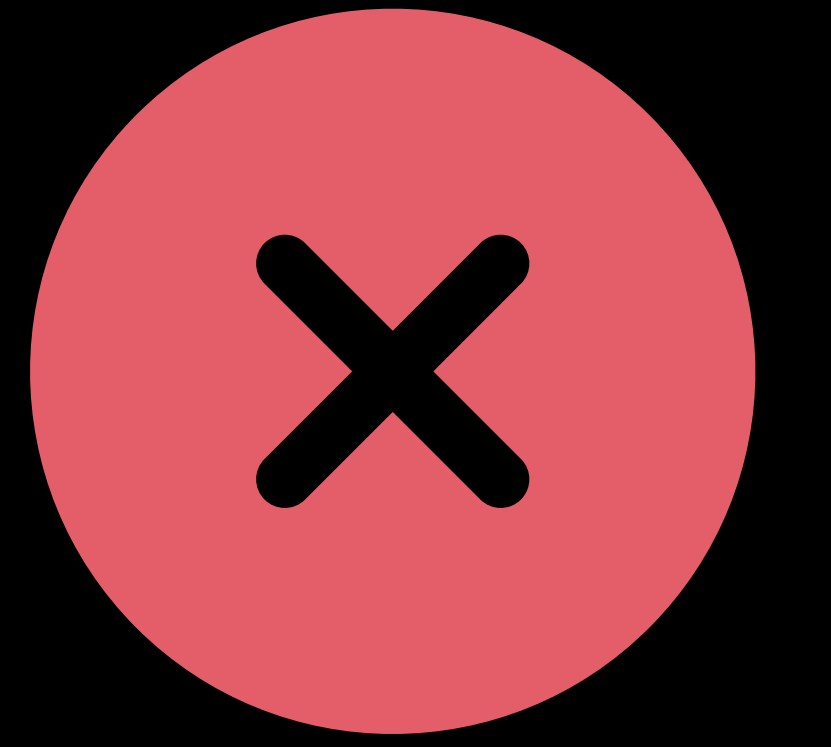


# No Split Personality Camera Output

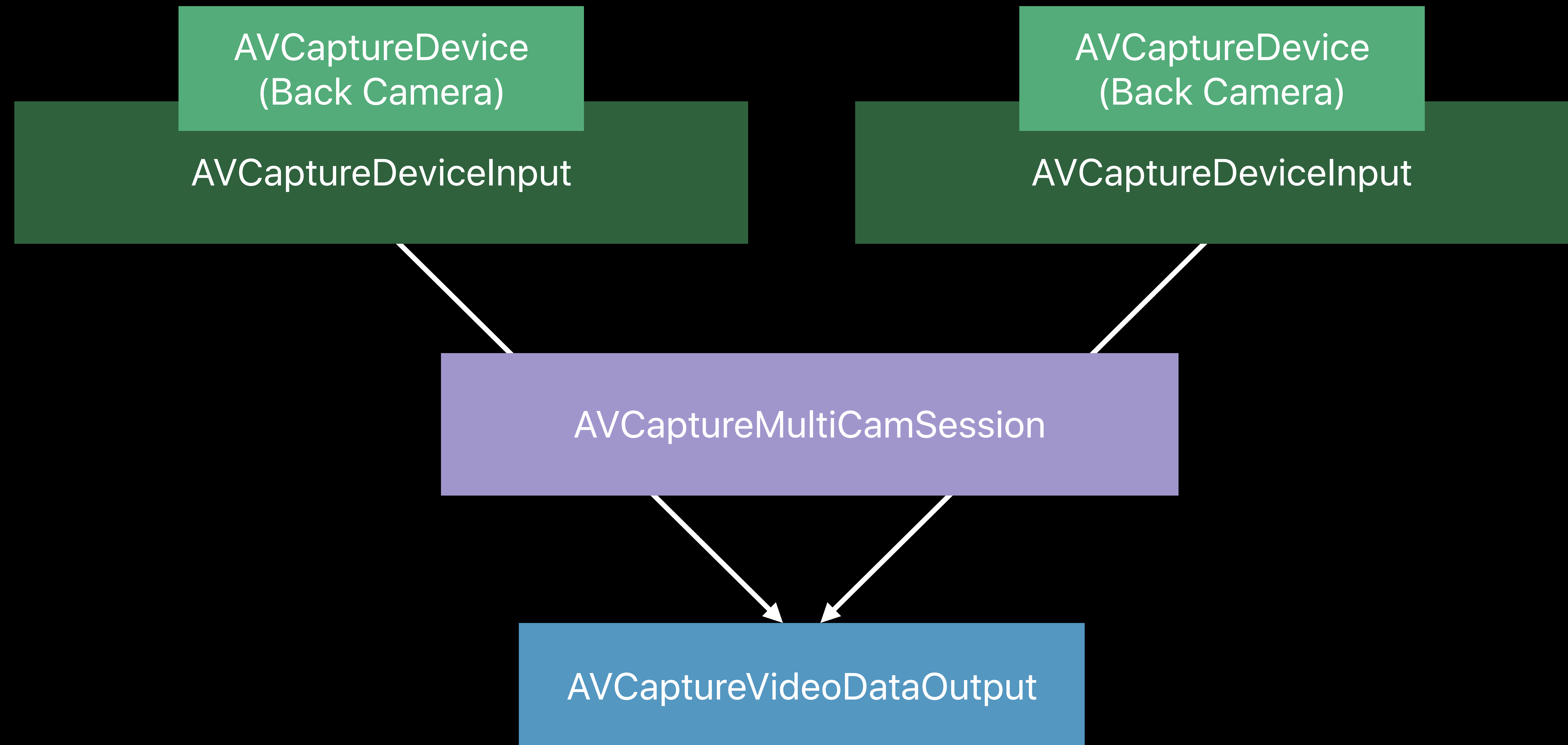
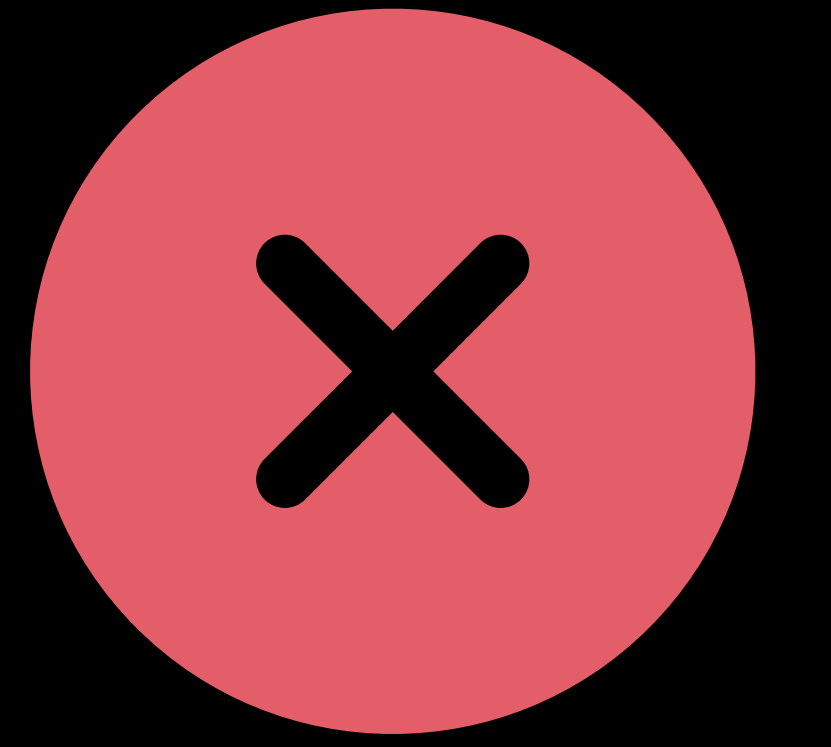




# No Multi-Camera Output Stuffing

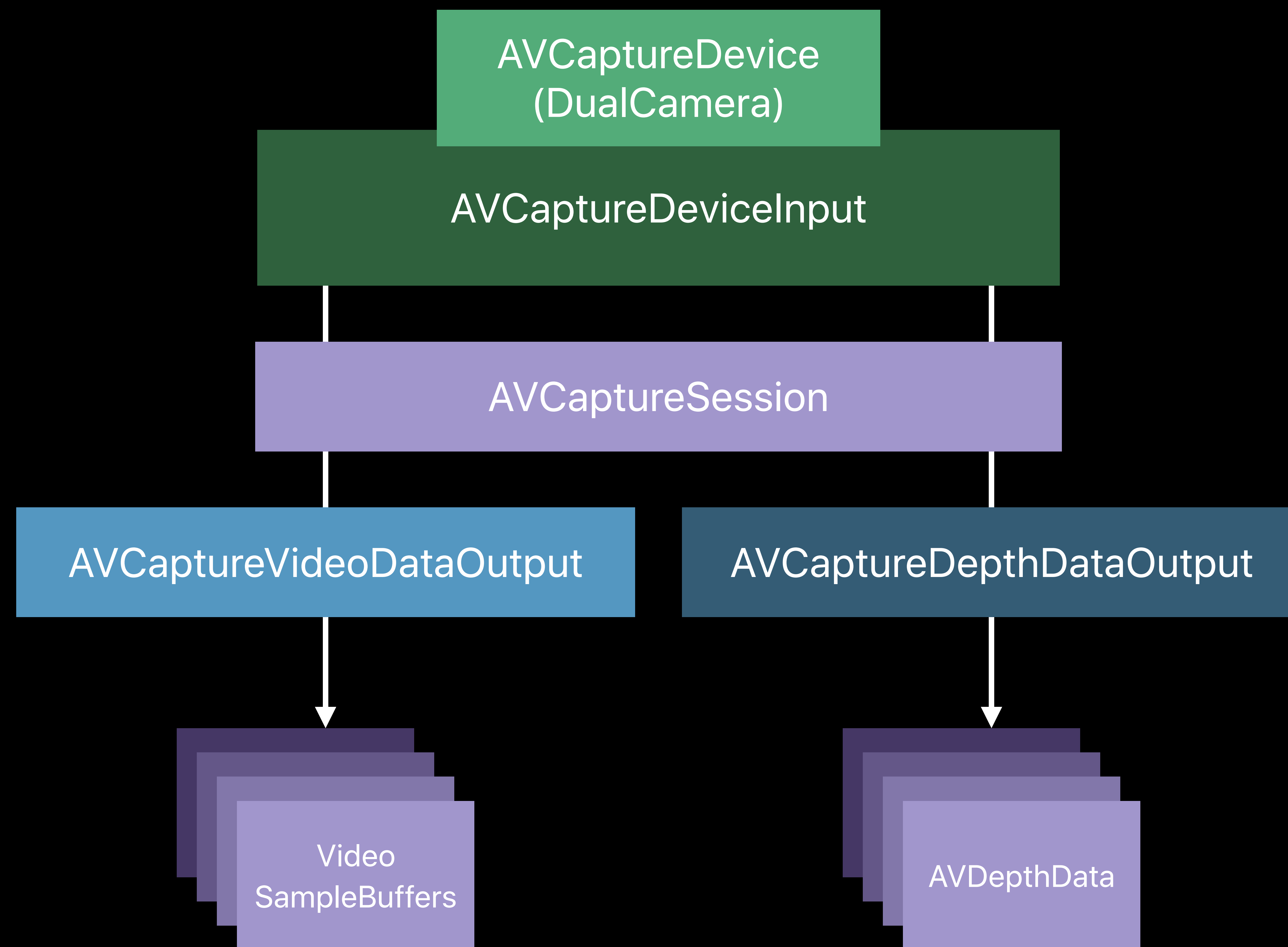


# No Multi-Camera Output Stuffing

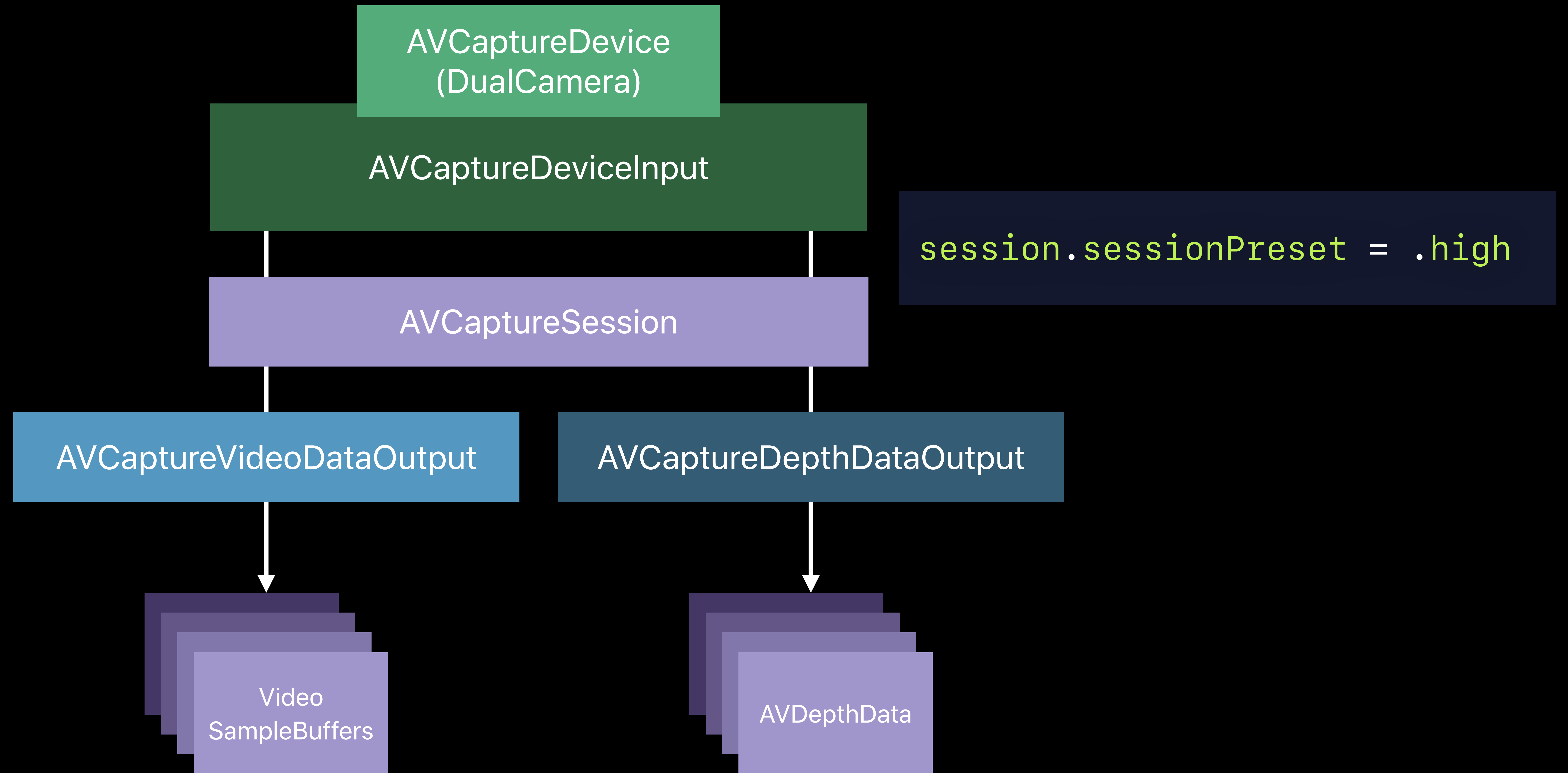




# Presets



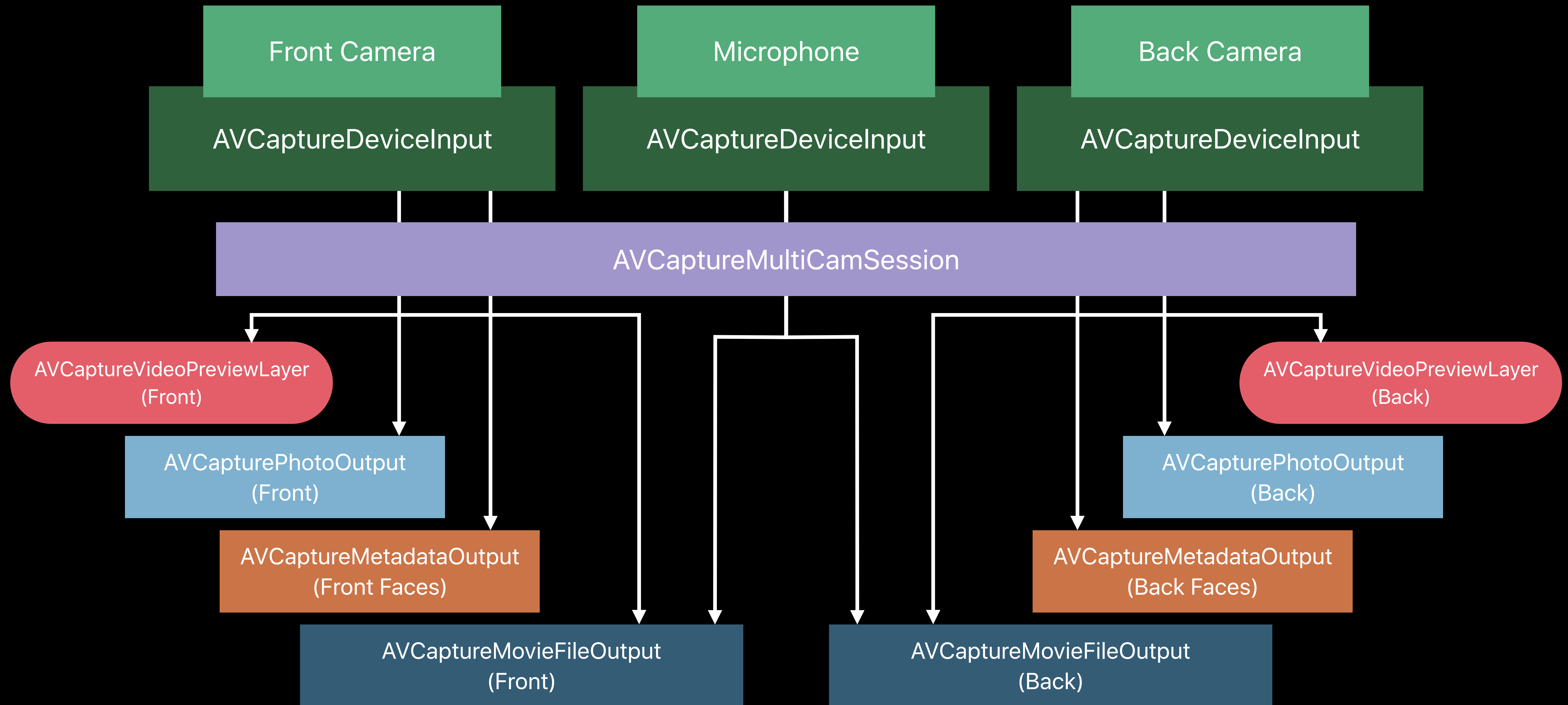
# Presets



**Presets? We Don't Need No Stinkin' Presets**

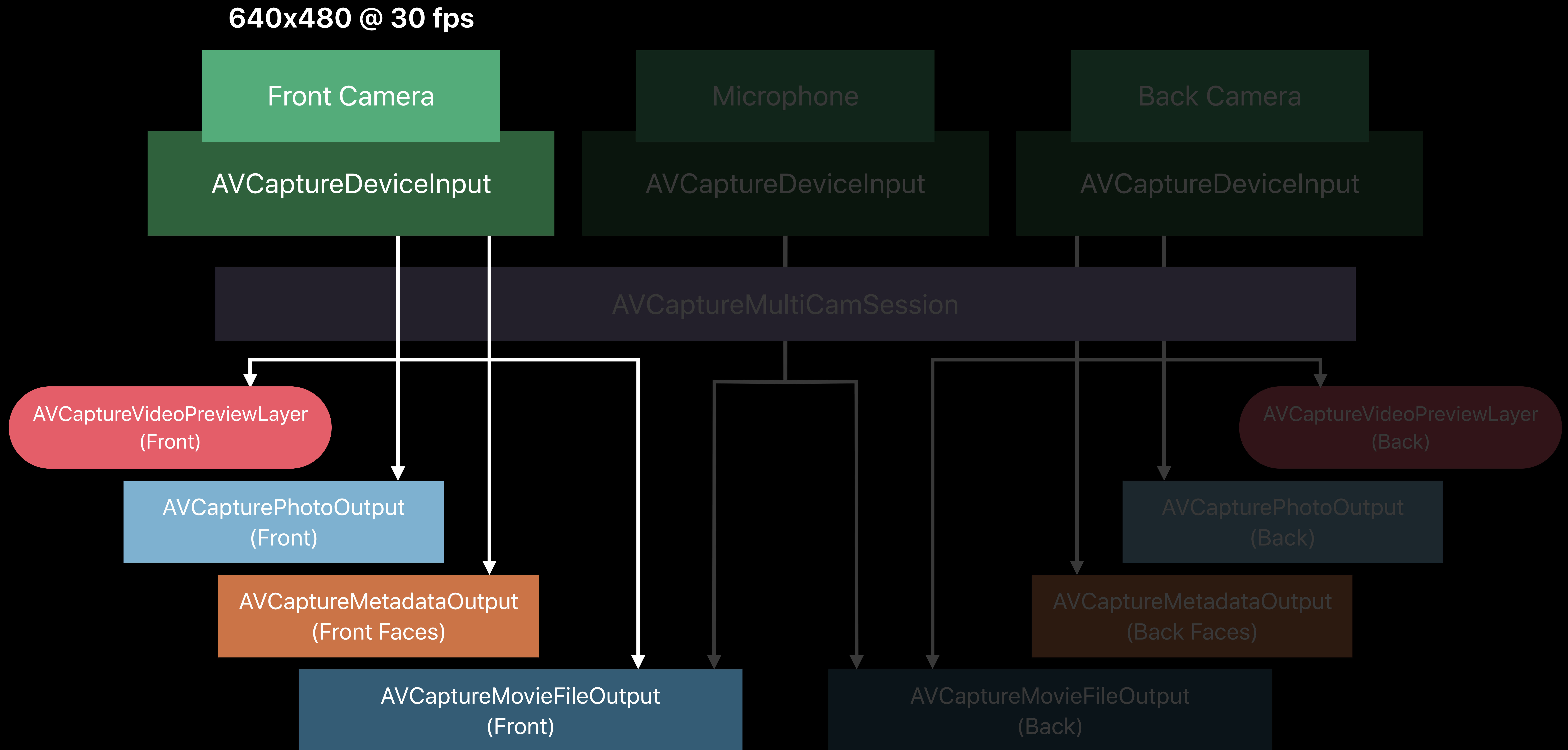


# Presets? We Don't Need No Stinkin' Presets



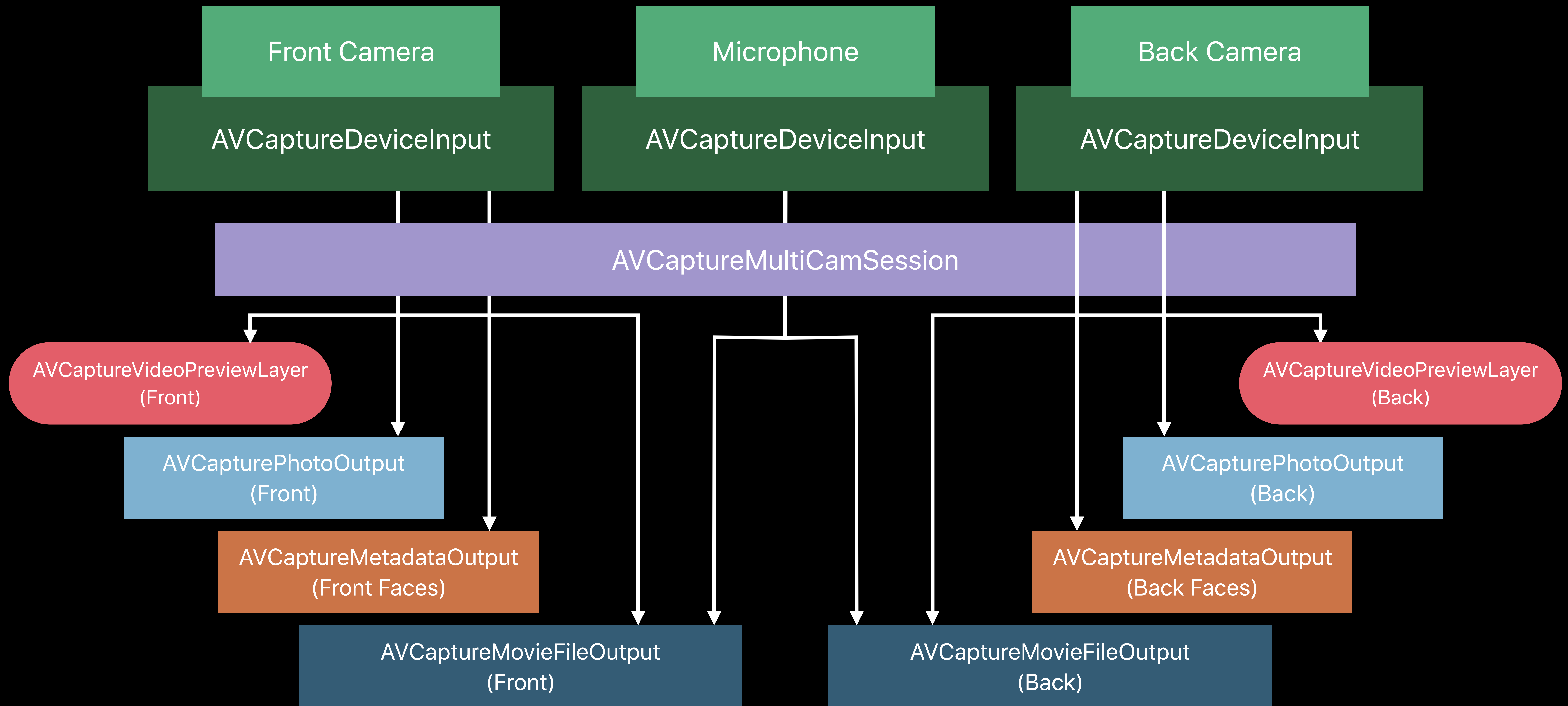


# Presets? We Don't Need No Stinkin' Presets

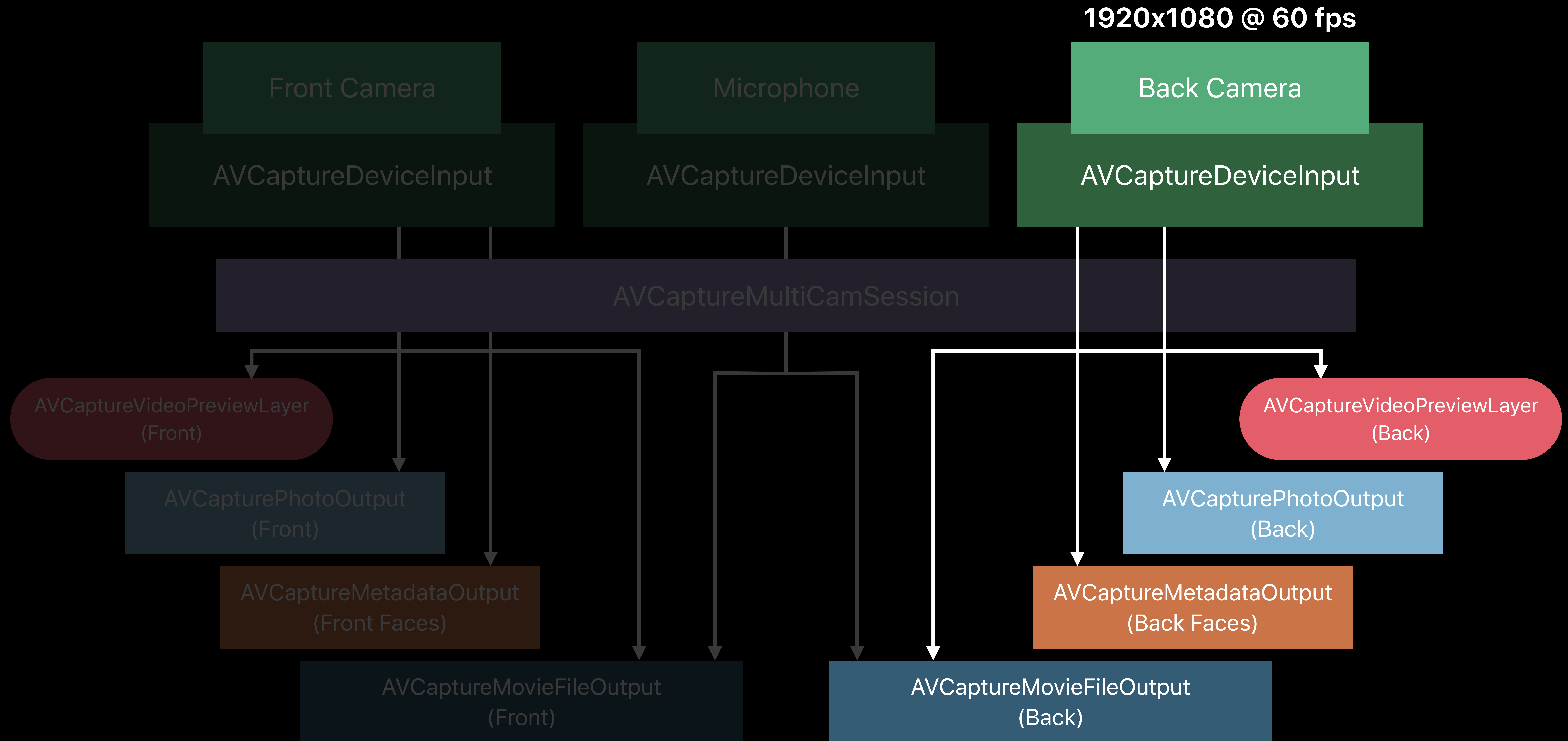




# Presets? We Don't Need No Stinkin' Presets



# Presets? We Don't Need No Stinkin' Presets



# Multi-Camera Capture

Cost functions



“There’s no such thing as a free lunch.”

- Somebody who wants you to pay for lunch

# Hardware Costs

Multiple cameras = multiple sensors

# Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to  $n$  pixels per clock



# Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to  $n$  pixels per clock

Contributors to hardware cost

# Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to  $n$  pixels per clock

Contributors to hardware cost

- Video resolution

# Hardware Costs

Multiple cameras = multiple sensors

One ISP, limited to  $n$  pixels per clock

Contributors to hardware cost

- Video resolution
- Max frame rate



# Hardware Costs

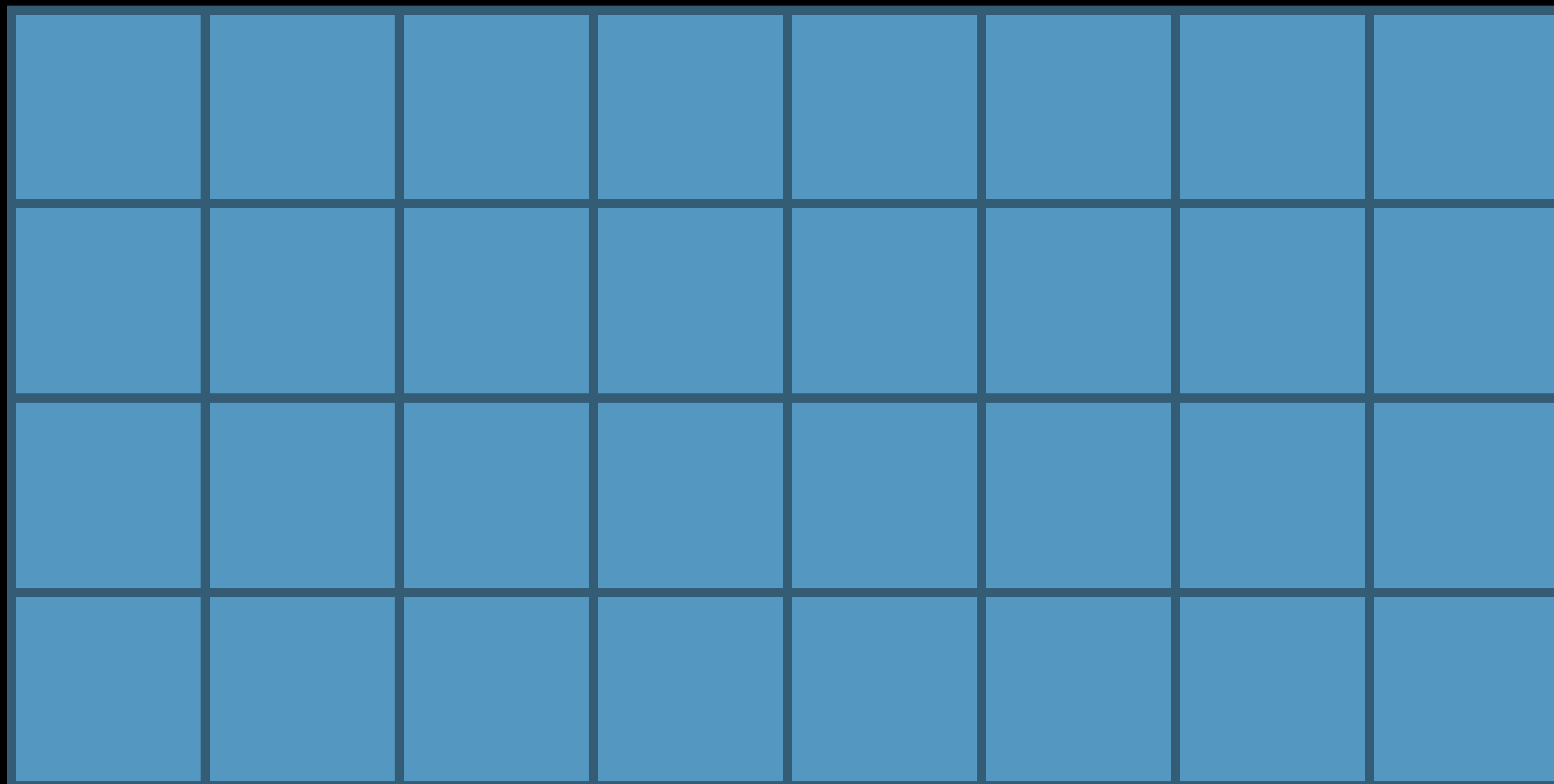
Multiple cameras = multiple sensors

One ISP, limited to  $n$  pixels per clock

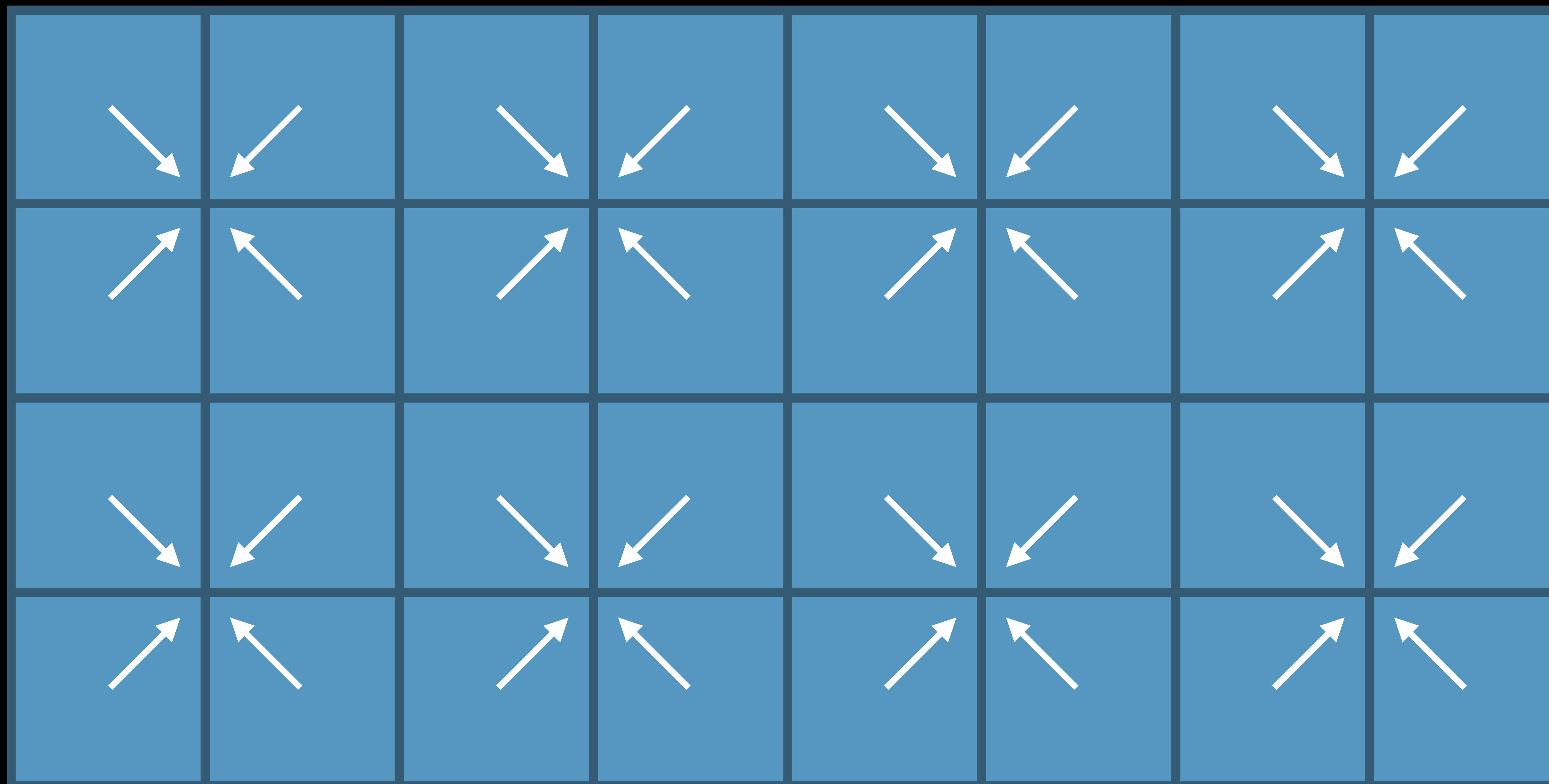
Contributors to hardware cost

- Video resolution
- Max frame rate
- Sensor "binning"

# Sensor Binning

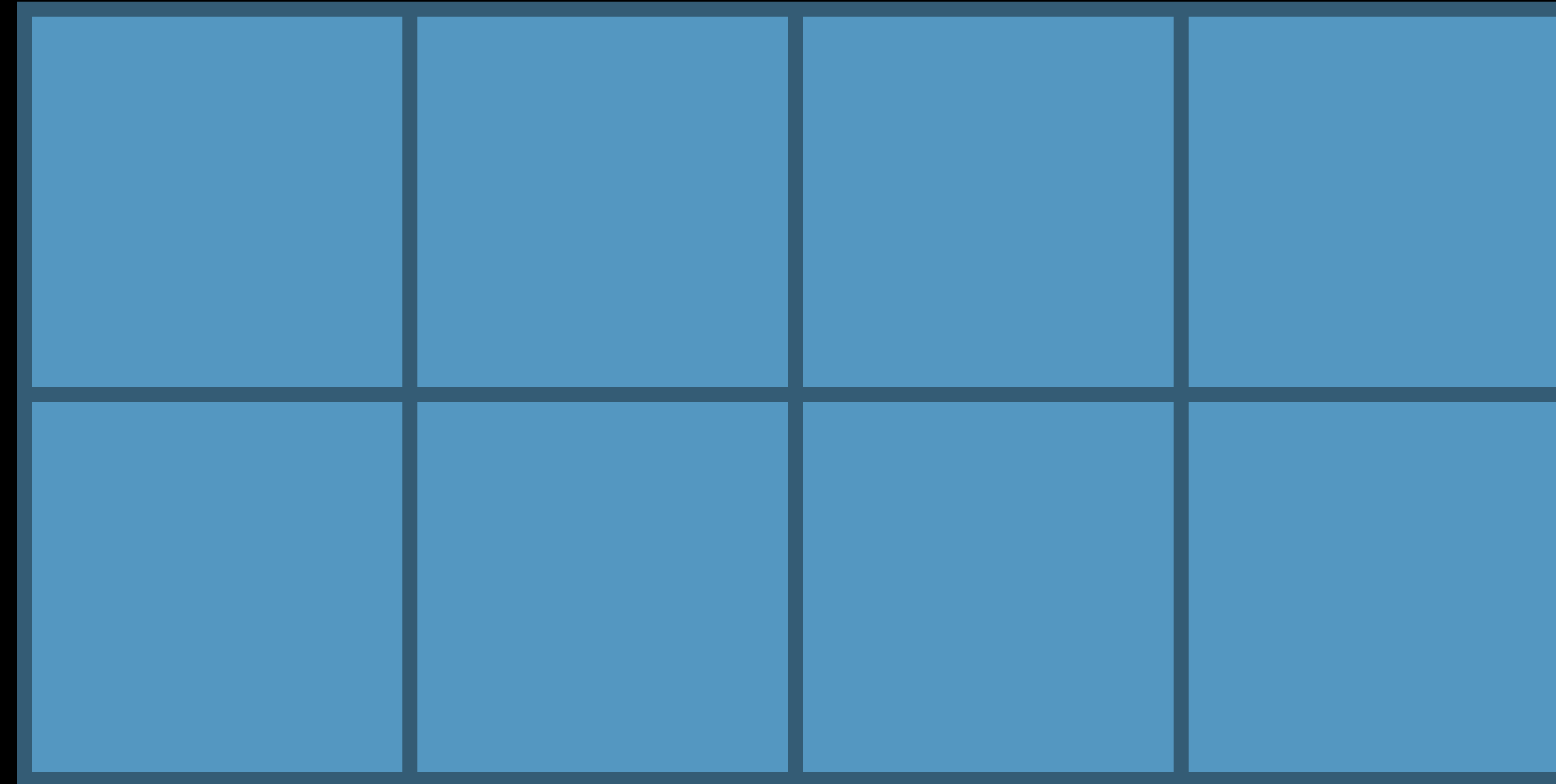


# Sensor Binning





# Sensor Binning



# Hardware Cost Reporting





# Hardware Cost Reporting

```
multicamSession.hardwareCost
```





# Hardware Cost Reporting

```
multicamSession.hardwareCost
```

$\geq 0.0$  &&  $\leq 1.0$  : Runnable





# Hardware Cost Reporting

```
multicamSession.hardwareCost
```

$\geq 0.0 \ \&\& \ \leq 1.0$  : Runnable

$\geq 1.0$  : No Bueno





# Hardware Cost Reporting

```
multicamSession.hardwareCost
```

$\geq 0.0 \ \&\& \ \leq 1.0$  : Runnable

$\geq 1.0$  : No Bueno

```
AVCaptureSessionRuntimeError = hardwareCostOverage
```





# How to Reduce Your Hardware Cost

# How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

# How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

Lower the camera max frame rate



# How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

~~Lower the camera max frame rate~~

# How to Reduce Your Hardware Cost

Lower the camera resolution

Choose a binned format

~~Lower the camera max frame rate~~

Set a max frame rate override at the device input

# How to Reduce Your Hardware Cost

NEW

Lower the camera resolution

Choose a binned format

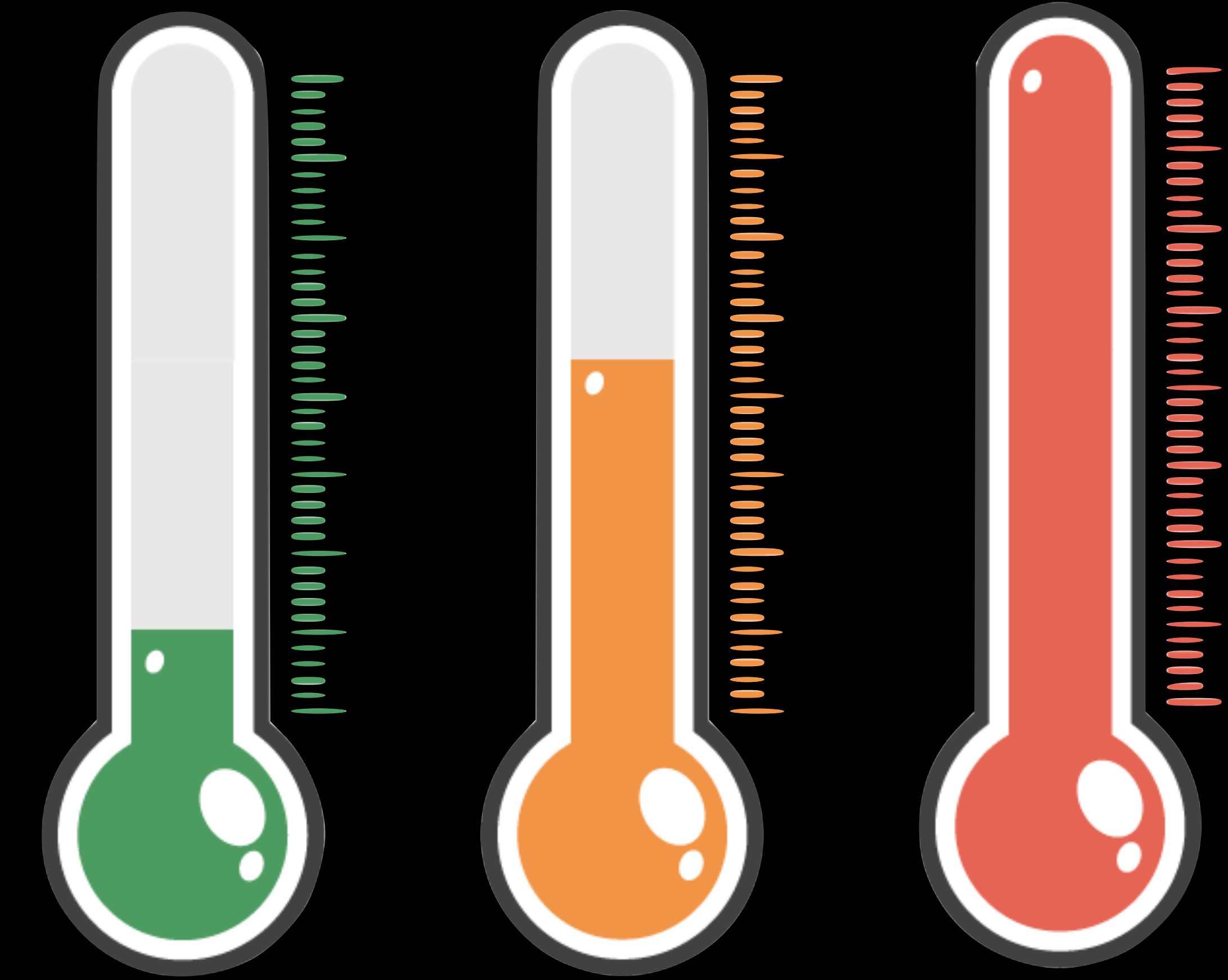
~~Lower the camera max frame rate~~

Set a max frame rate override at the device input

```
CMTIME thirtyFPS = CMTIMEMake( 1, 30 )  
deviceInput.videoMinFrameDurationOverride = thirtyFPS
```

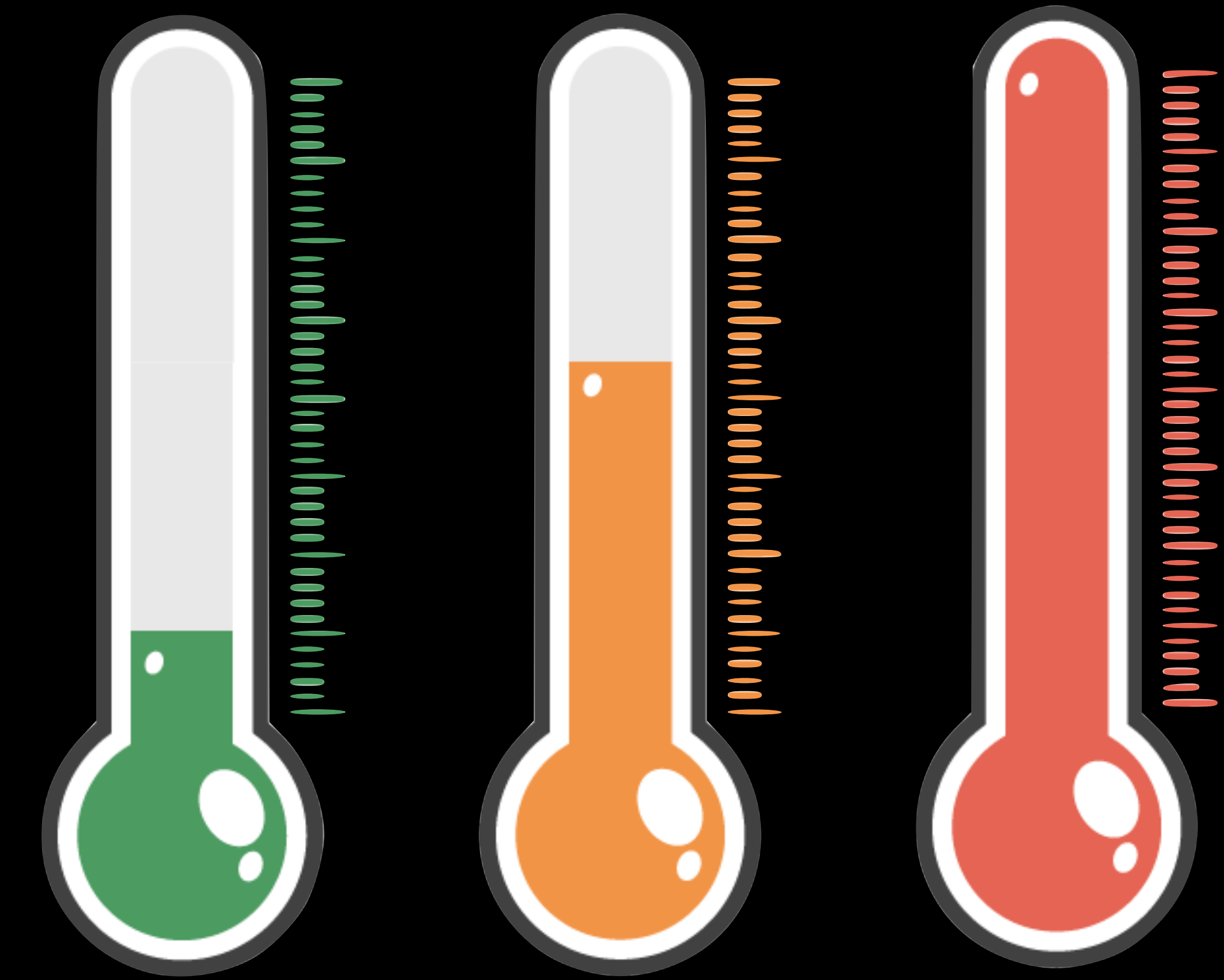


# System Pressure States



# System Pressure States

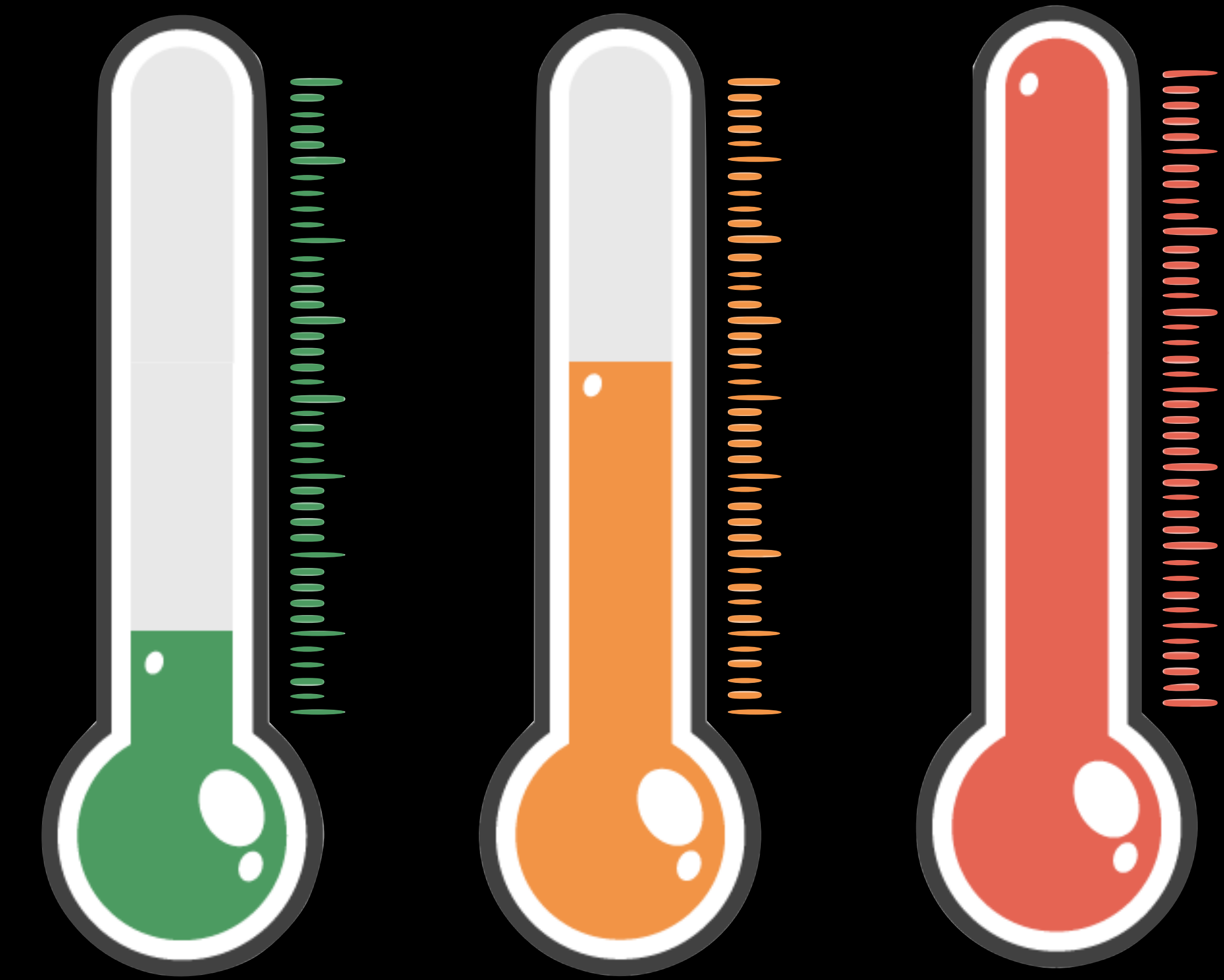
Camera System Pressure States  
introduced in iOS 11



# System Pressure States

Camera System Pressure States  
introduced in iOS 11

- System Temperature

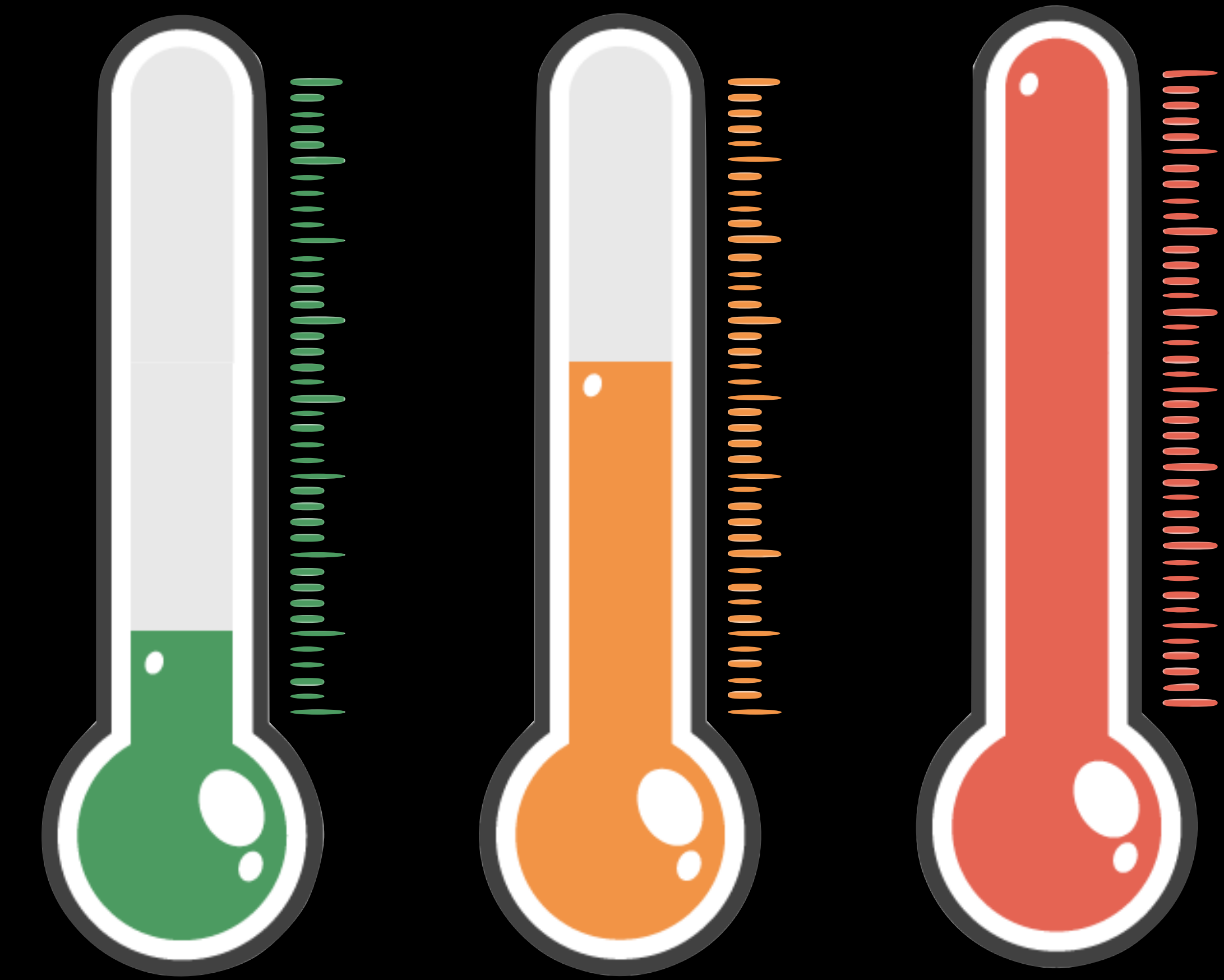




# System Pressure States

Camera System Pressure States  
introduced in iOS 11

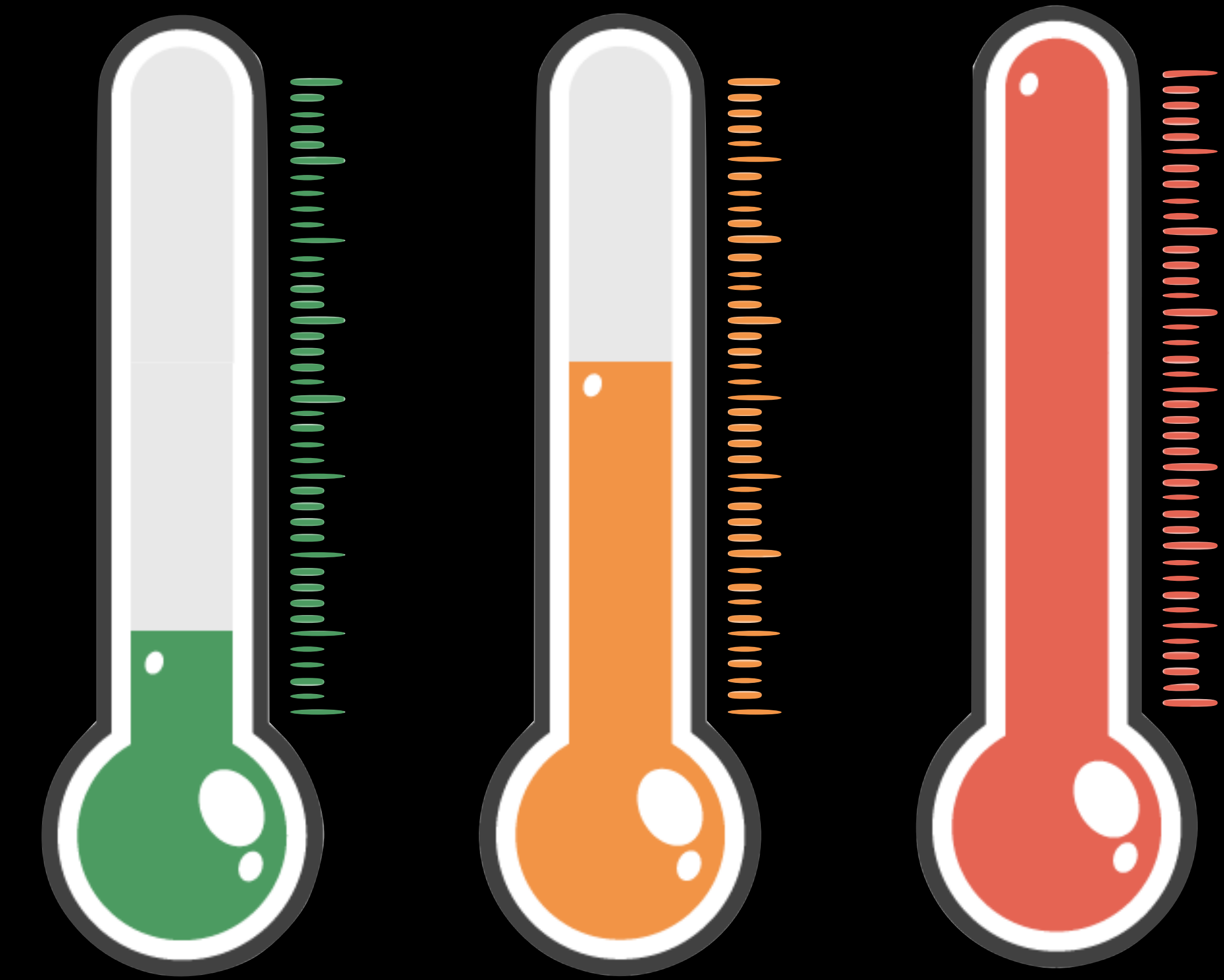
- System Temperature
- Peak Power Demands



# System Pressure States

Camera System Pressure States introduced in iOS 11

- System Temperature
- Peak Power Demands
- Infrared Projector Temperature



# System Pressure States

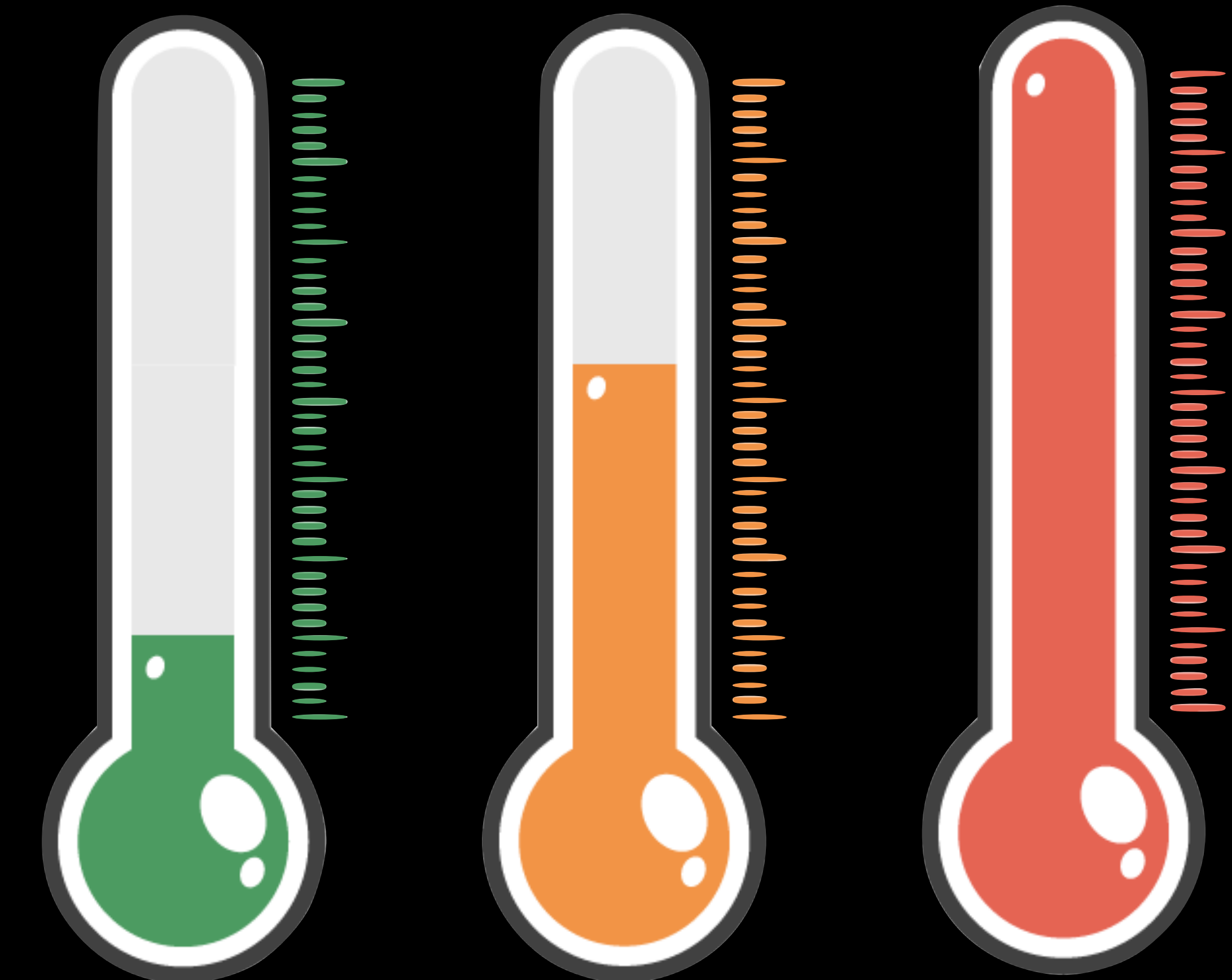
Nominal

Fair

Serious

Critical

Shutdown



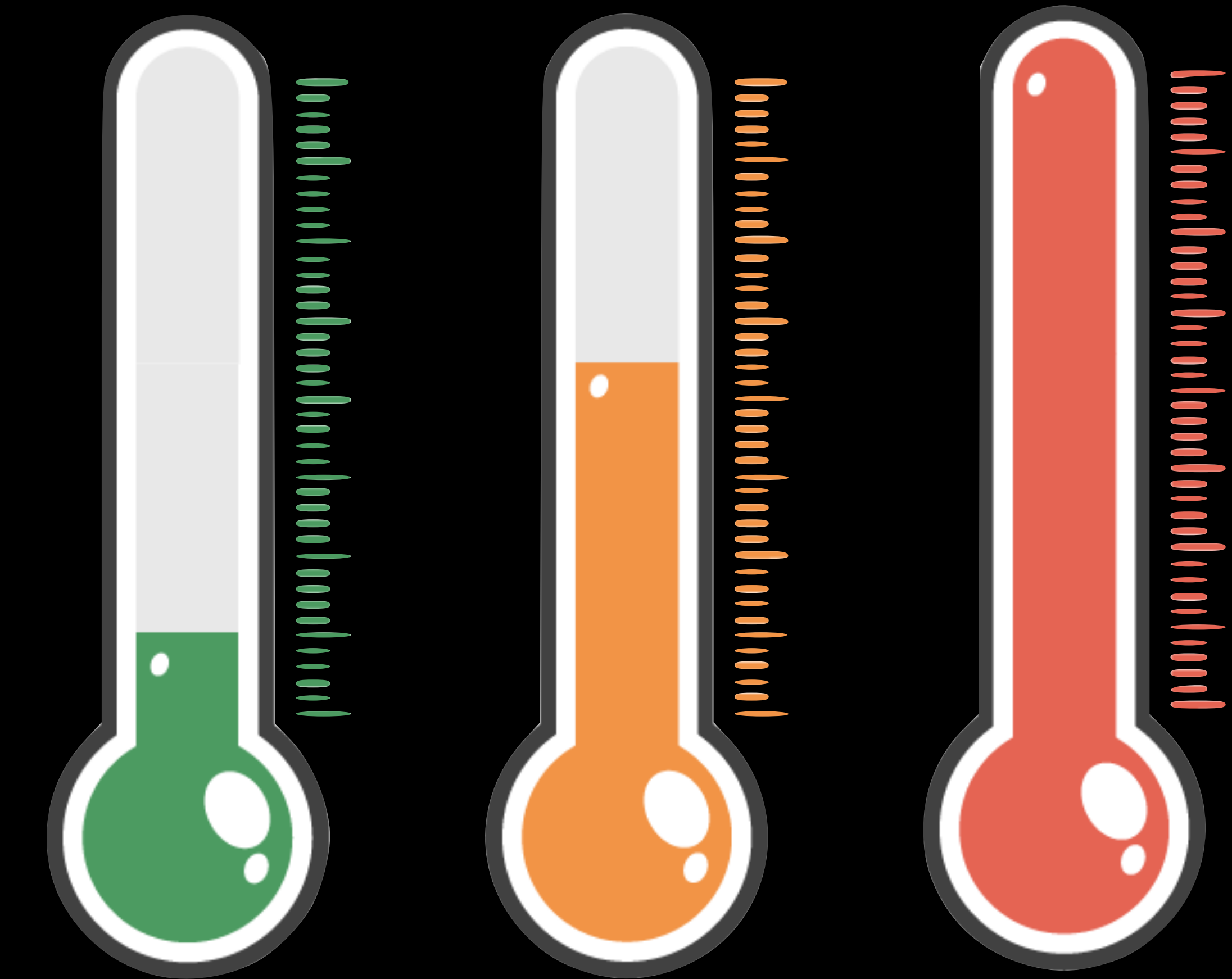


# System Pressure Cost

Camera system cost excluding all other factors

Contributors

- Same as hardware cost contributors
- VIS, OIS
- Smart HDR
- Infrared sensor and projector power for TrueDepth
- Microphone power

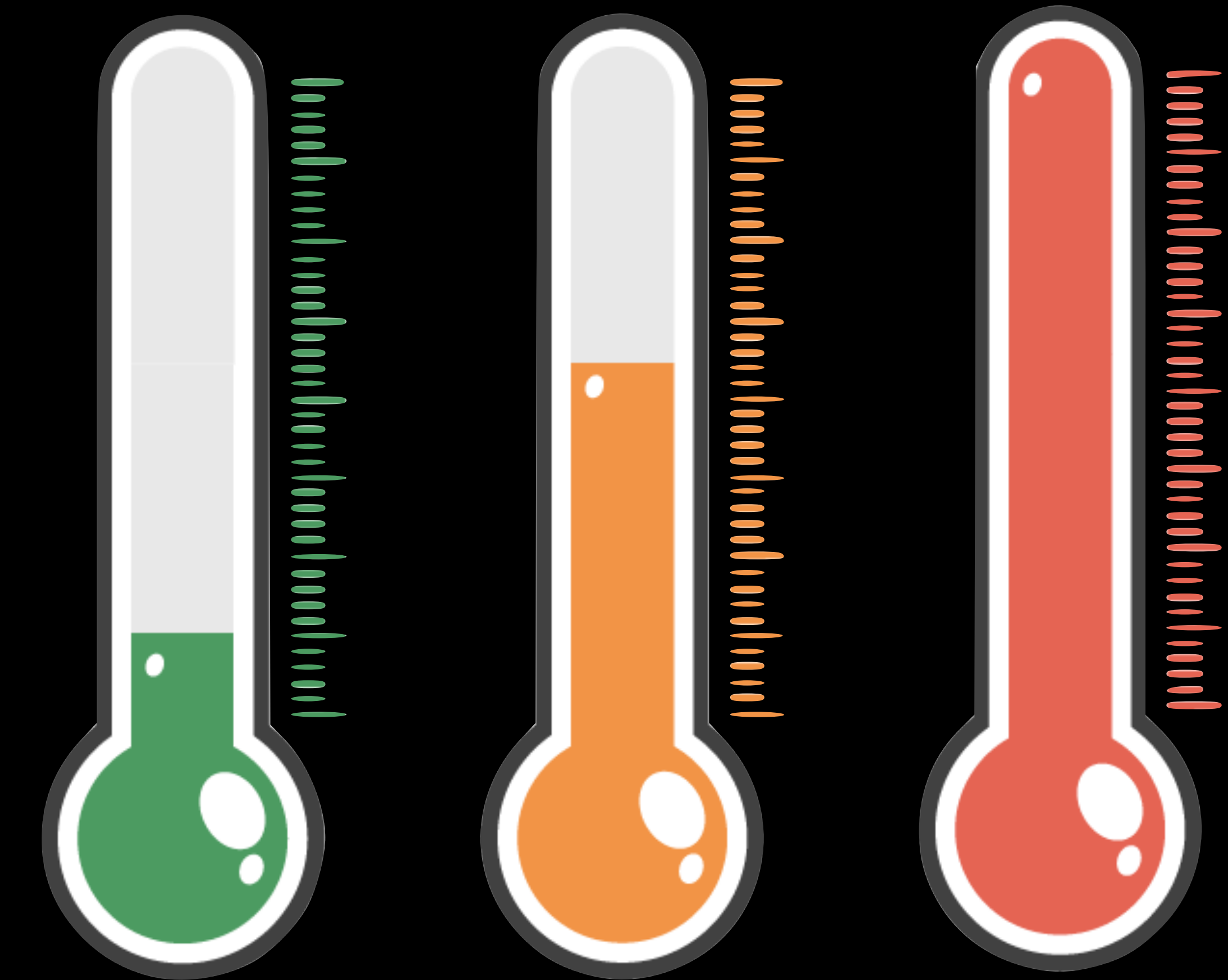


# System Pressure Cost Reporting

MultiCamSession tallies a system pressure cost

Independent of current system pressure state

Accounts only for factors the session knows about



# System Pressure Cost Reporting



# System Pressure Cost Reporting

```
multicamSession.systemPressureCost
```



# System Pressure Cost Reporting

```
multicamSession.systemPressureCost
```

■ < 1.0 : Runnable indefinitely

# System Pressure Cost Reporting



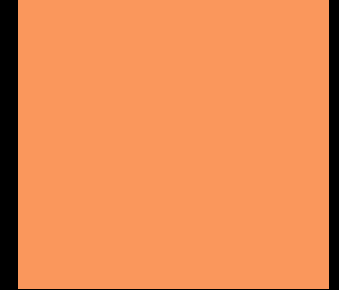
```
multicamSession.systemPressureCost
```

-   $< 1.0$  : Runnable indefinitely
-   $\geq 1.0 \ \&\& \ \leq 2.0$  : Runnable for 15 minutes



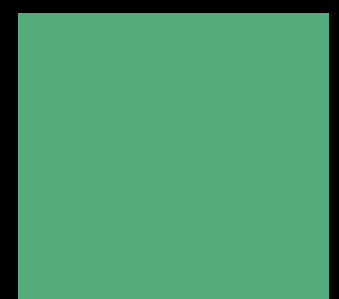
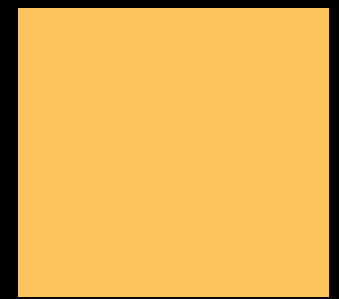
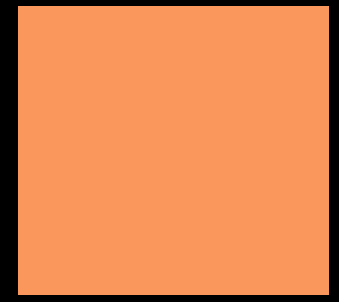
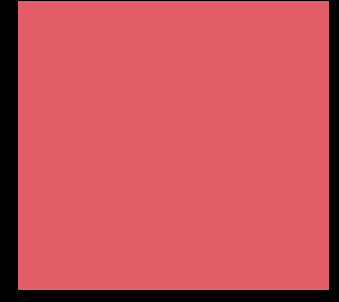
# System Pressure Cost Reporting

```
multicamSession.systemPressureCost
```

-   $< 1.0$  : Runnable indefinitely
-   $\geq 1.0 \ \&\& \ \leq 2.0$  : Runnable for 15 minutes
-   $\geq 2.0 \ \&\& \ \leq 3.0$  : Runnable for 10 minutes

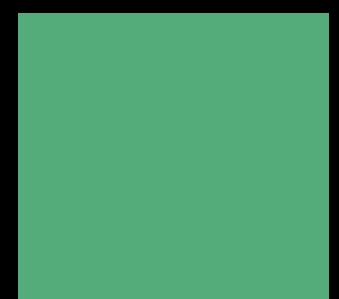
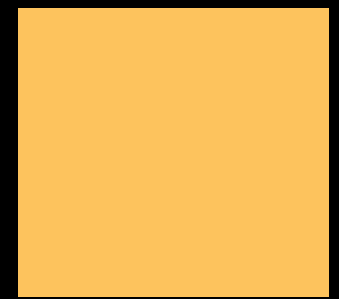
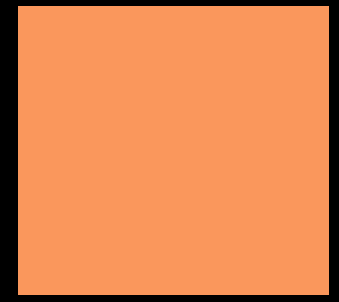
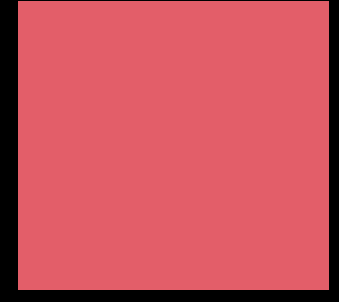
# System Pressure Cost Reporting

```
multicamSession.systemPressureCost
```

-   $< 1.0$  : Runnable indefinitely
-   $\geq 1.0 \ \&\& \ \leq 2.0$  : Runnable for 15 minutes
-   $\geq 2.0 \ \&\& \ \leq 3.0$  : Runnable for 10 minutes
-   $> 3.0$  : ¿Como se dice en fuego?

# System Pressure Cost Reporting

```
multicamSession.systemPressureCost
```

-   $< 1.0$  : Runnable indefinitely
-   $\geq 1.0 \ \&\& \ \leq 2.0$  : Runnable for 15 minutes
-   $\geq 2.0 \ \&\& \ \leq 3.0$  : Runnable for 10 minutes
-   $> 3.0$  : ¿Como se dice en fuego?

MultiCamSession will interrupt your session when system pressure = Shutdown



# How to Reduce System Pressure While Running

Lower one or more capture device frame rates

Throttle any GPU or CPU processing in your app code

Disable a camera (session keeps running!)

# How to Reduce System Pressure While Running

NEW

Lower one or more capture device frame rates

Throttle any GPU or CPU processing in your app code

Disable a camera (session keeps running!)

```
frontCameraInputVideoPort.enabled = false
```

# Unreported Costs

Unreported costs are unreported

Artificial constraints on devices and formats



# Supported Device Combinations (iPhone XS)

	Back Wide	Back Telephoto	Back Dual Camera	Front	Front TrueDepth
1			✓		
2	✓			✓	
3	✓	✓			
4	✓				✓
5		✓		✓	
6		✓			✓

```
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    print("MultiCam not supported on this device")
    setupResult = .multiCamNotSupported
    return
}

// Find the supported multicam device combinations

let deviceTypes = [AVCaptureDevice.DeviceType.builtInDualCamera,
                   AVCaptureDevice.DeviceType.builtInWideAngleCamera
                   AVCaptureDevice.DeviceType.builtInTelephotoCamera]
let session = AVCaptureDevice.DiscoverySession(deviceTypes: deviceTypes,
                                              mediaType: .video,
                                              position: .unspecified)

let multicamSupportedDeviceSets = session.supportedMultiCamDeviceSets
```

```
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    print("MultiCam not supported on this device")
    setupResult = .multiCamNotSupported
    return
}
```

```
// Find the supported multicam device combinations
```

```
let deviceTypes = [AVCaptureDevice.DeviceType.builtInDualCamera,
                  AVCaptureDevice.DeviceType.builtInWideAngleCamera
                  AVCaptureDevice.DeviceType.builtInTelephotoCamera]
let session = AVCaptureDevice.DiscoverySession(deviceTypes: deviceTypes,
                                              mediaType: .video,
                                              position: .unspecified)

let multicamSupportedDeviceSets = session.supportedMultiCamDeviceSets
```



```
guard AVCaptureMultiCamSession.isMultiCamSupported else {
    print("MultiCam not supported on this device")
    setupResult = .multiCamNotSupported
    return
}

// Find the supported multicam device combinations

let deviceTypes = [AVCaptureDevice.DeviceType.builtInDualCamera,
                   AVCaptureDevice.DeviceType.builtInWideAngleCamera
                   AVCaptureDevice.DeviceType.builtInTelephotoCamera]
let session = AVCaptureDevice.DiscoverySession(deviceTypes: deviceTypes,
                                              mediaType: .video,
                                              position: .unspecified)

let multicamSupportedDeviceSets = session.supportedMultiCamDeviceSets
```

# Supported MultiCam Formats (iPhone XS)

Resolution	Max FPS	Binned?	Hi-Res Stills
640x480	60	Yes	2016x1512
1280x720	60	Yes	2112x1188
1440x1080	60	Yes	2016x1512
1920x1080	30	No	4224x2376
1920x1080	60	Yes	2112x1188
1920x1440	30	No	4032x3024
1920x1440	60	Yes	2016x1512

# Supported MultiCam Formats (iPhone XS)

Resolution	Max FPS	Binned?	Hi-Res Stills
640x480	60	Yes	2016x1512
1280x720	60	Yes	2112x1188
1440x1080	60	Yes	2016x1512
1920x1080	30	No	4224x2376
1920x1080	60	Yes	2112x1188
1920x1440	30	No	4032x3024
1920x1440	60	Yes	2016x1512



# Supported MultiCam Formats (iPhone XS)

Resolution	Max FPS	Binned?	Hi-Res Stills
640x480	60	Yes	2016x1512
1280x720	60	Yes	2112x1188
1440x1080	60	Yes	2016x1512
<b>1920x1080</b>	<b>30</b>	<b>No</b>	<b>4224x2376</b>
1920x1080	60	Yes	2112x1188
1920x1440	30	No	4032x3024
1920x1440	60	Yes	2016x1512







# Supported MultiCam Formats (iPhone XS)

Resolution	Max FPS	Binned?	Hi-Res Stills
640x480	60	Yes	2016x1512
1280x720	60	Yes	2112x1188
1440x1080	60	Yes	2016x1512
1920x1080	30	No	4224x2376
1920x1080	60	Yes	2112x1188
<b>1920x1440</b>	<b>30</b>	<b>No</b>	<b>4032x3024</b>
1920x1440	60	Yes	2016x1512

```
// Find and activate the next smaller format that supports multicam

let formats = videoDeviceInput.device.formats
for format in formats.reversed() {
    if format.isMultiCamSupported {
        dims = CMVideoFormatDescriptionGetDimensions(format.formatDescription)
        if dims.width < activeWidth || dims.height < activeHeight {
            do {
                try videoDeviceInput.device.lockForConfiguration()
                videoDeviceInput.device.activeFormat = format
                videoDeviceInput.device.unlockForConfiguration()
                return true
            } catch {
                return false
            }
        }
    }
}
```

# Supported Session / App Configurations

	macOS	iOS
One app, one capture session, multiple cameras		
One app, multiple sessions, one or more cameras		
Multiple apps, multiple sessions, one or more cameras		



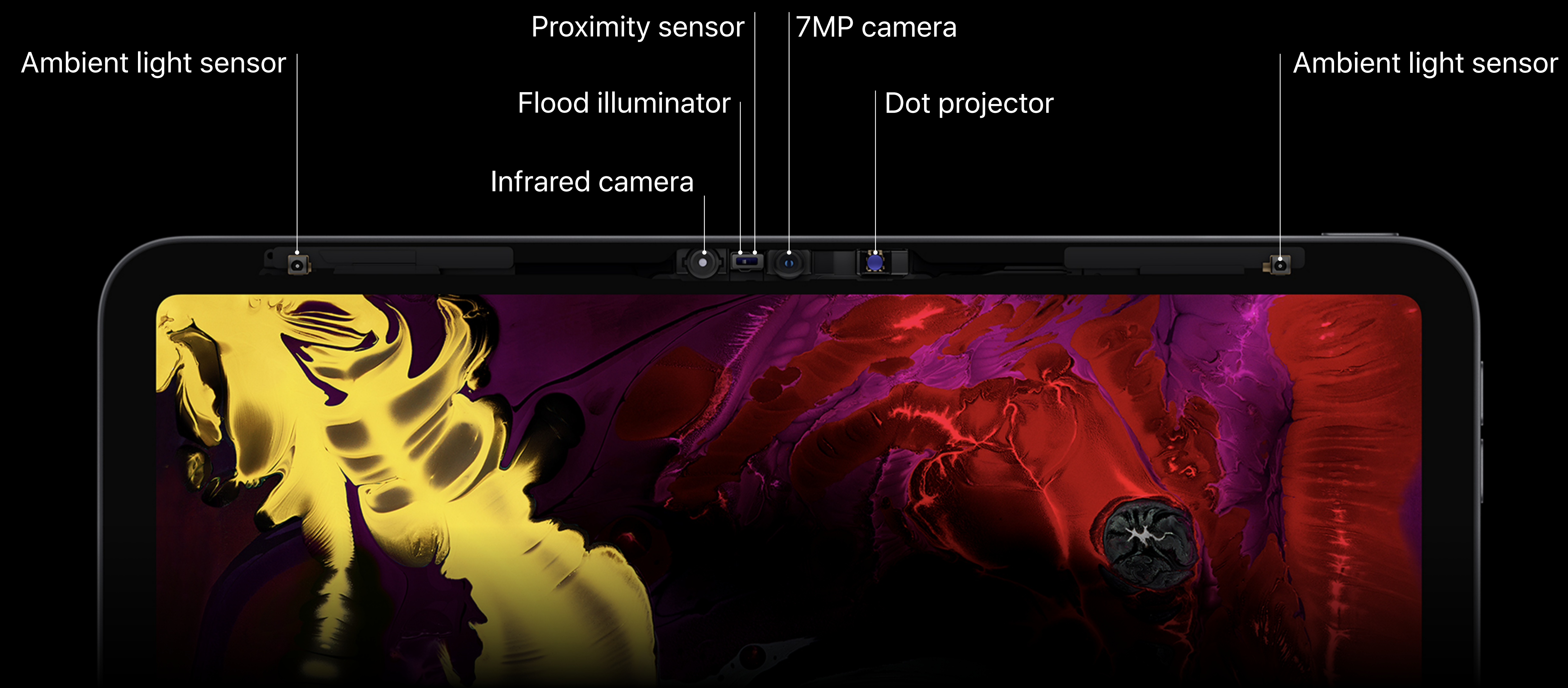
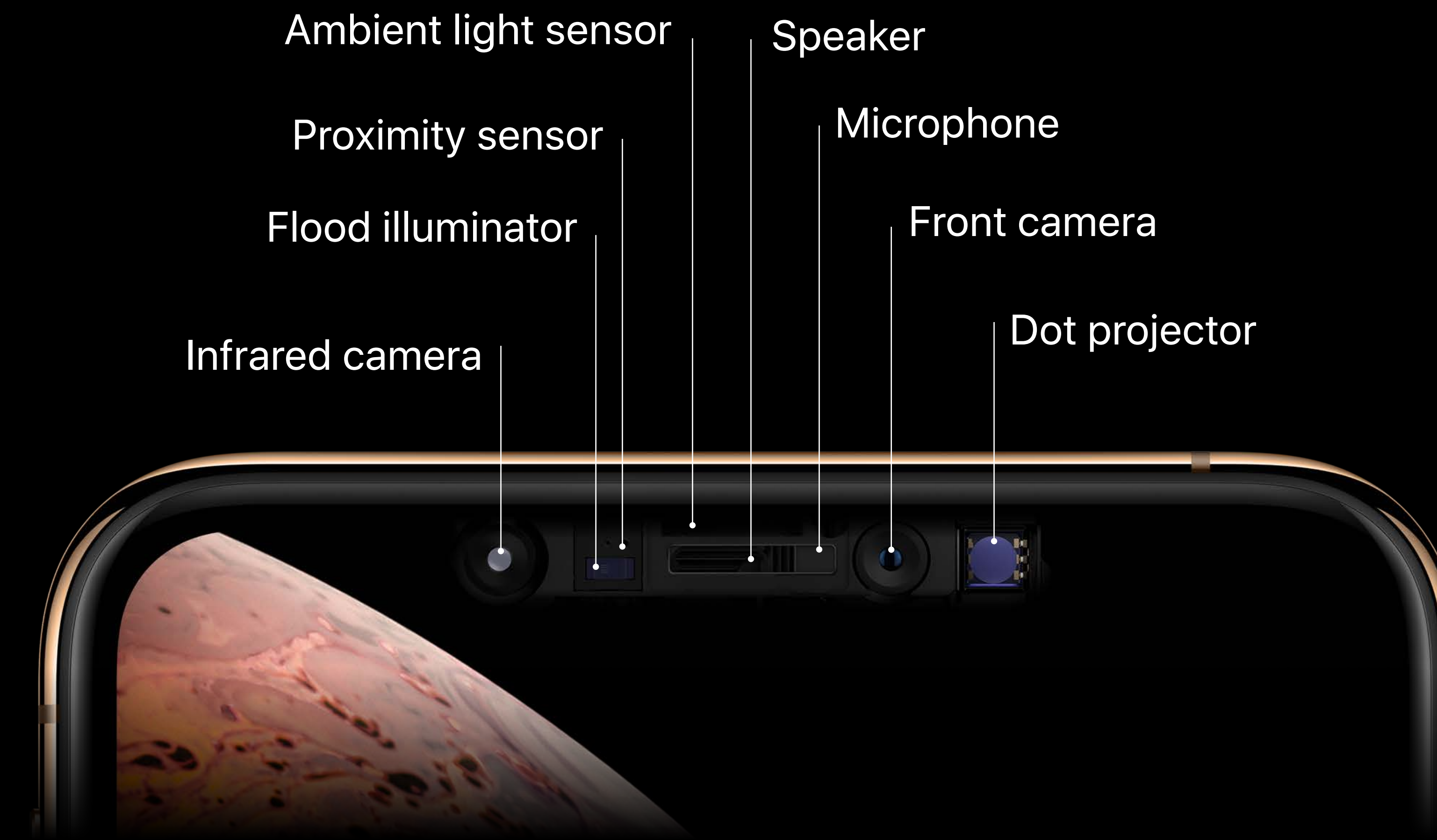
Be Responsible

# Multi-Camera Capture

Synchronized Streaming



# Virtual Cameras





# Virtual Cameras

DualCam presents one video stream at a time\*

Switches between cameras at different zoom levels

Generates disparity (depth) from wide and tele



\*AVCapturePhotoOutput dualCameraDualPhotoDelivery is the exception





NEW

```
// Virtual devices are made up of constituent devices
```

```
if aCameraDevice.isVirtualDevice == true
```

```
{
```

```
    let subCameras = aCameraDevice.constituentDevices
```

```
    for let subCamera in subCameras {
```

```
        print("Sub cameras: \(subCamera.localizedName)")
```

```
    }
```

```
}
```

# Synchronized Streaming

# Synchronized Streaming

When running a virtual device, its constituent devices share the same

# Synchronized Streaming

When running a virtual device, its constituent devices share the same

- Active resolution



# Synchronized Streaming

When running a virtual device, its constituent devices share the same

- Active resolution
- Frame rate

# Synchronized Streaming

When running a virtual device, its constituent devices share the same

- Active resolution
- Frame rate

They are synchronized

# Synchronized Streaming

When running a virtual device, its constituent devices share the same

- Active resolution
- Frame rate

They are synchronized

- Sensor read out matches frame centers

# Synchronized Streaming

When running a virtual device, its constituent devices share the same

- Active resolution
- Frame rate

They are synchronized

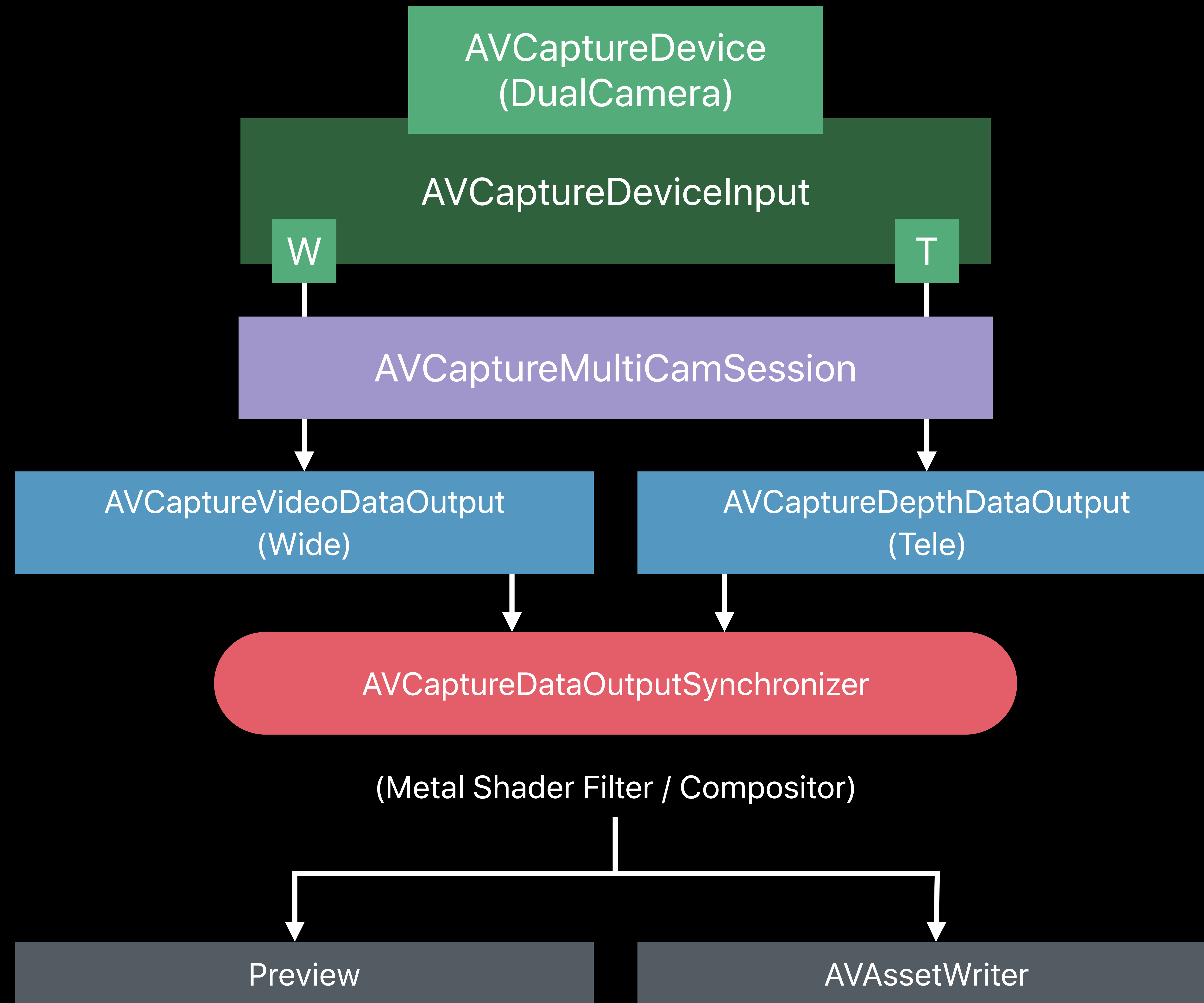
- Sensor read out matches frame centers
- Exposure / White Balance / Focus



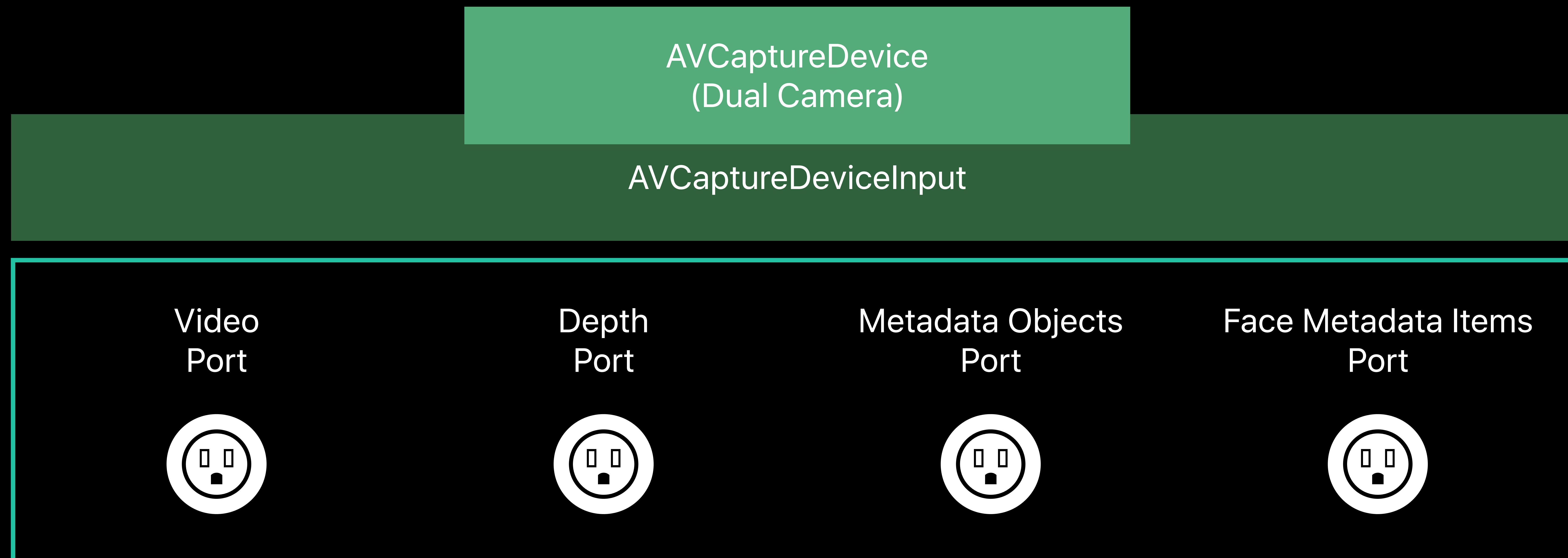
***Demo***

AVDualCam

# AVDualCam Graph Topology



# Any Port in a Storm?



```
let ports: [AVCaptureInput.Port] = dualCameraDeviceInput.ports
```

Virtual devices have secret ports.



```
// Constituent Device Port Discovery

guard let widePort = dualCameraInput.ports(for: .video,
                                           sourceDeviceType: .builtInWideAngleCamera,
                                           sourceDevicePosition: dualCamera.position).first,
let telePort = dualCameraInput.ports(for: .video,
                                      sourceDeviceType: .builtInTelephotoCamera,
                                      sourceDevicePosition: dualCamera.position).first
else {
    print("Could not obtain wide and telephoto camera input ports")
    return false
}
```

```
// Constituent Device Port Discovery
```

```
guard let widePort = dualCameraInput.ports(for: .video,  
                                             sourceDeviceType: .builtInWideAngleCamera,  
                                             sourceDevicePosition: dualCamera.position).first,  
    let telePort = dualCameraInput.ports(for: .video,  
                                           sourceDeviceType: .builtInTelephotoCamera,  
                                           sourceDevicePosition: dualCamera.position).first
```

```
else {  
    print("Could not obtain wide and telephoto camera input ports")  
    return false  
}
```

```
// Constituent Device Port Connections

let wideAngleCameraConnection = AVCaptureConnection(inputPorts: [widePort],
                                                    output: wideVideoDataOutput)
guard session.canAddConnection(wideAngleCameraConnection) else {
    print("Could not connect wide-angle video input to output")
    return false
}
session.addConnection(wideAngleCameraConnection)
```

# Dual Camera Homography Aids

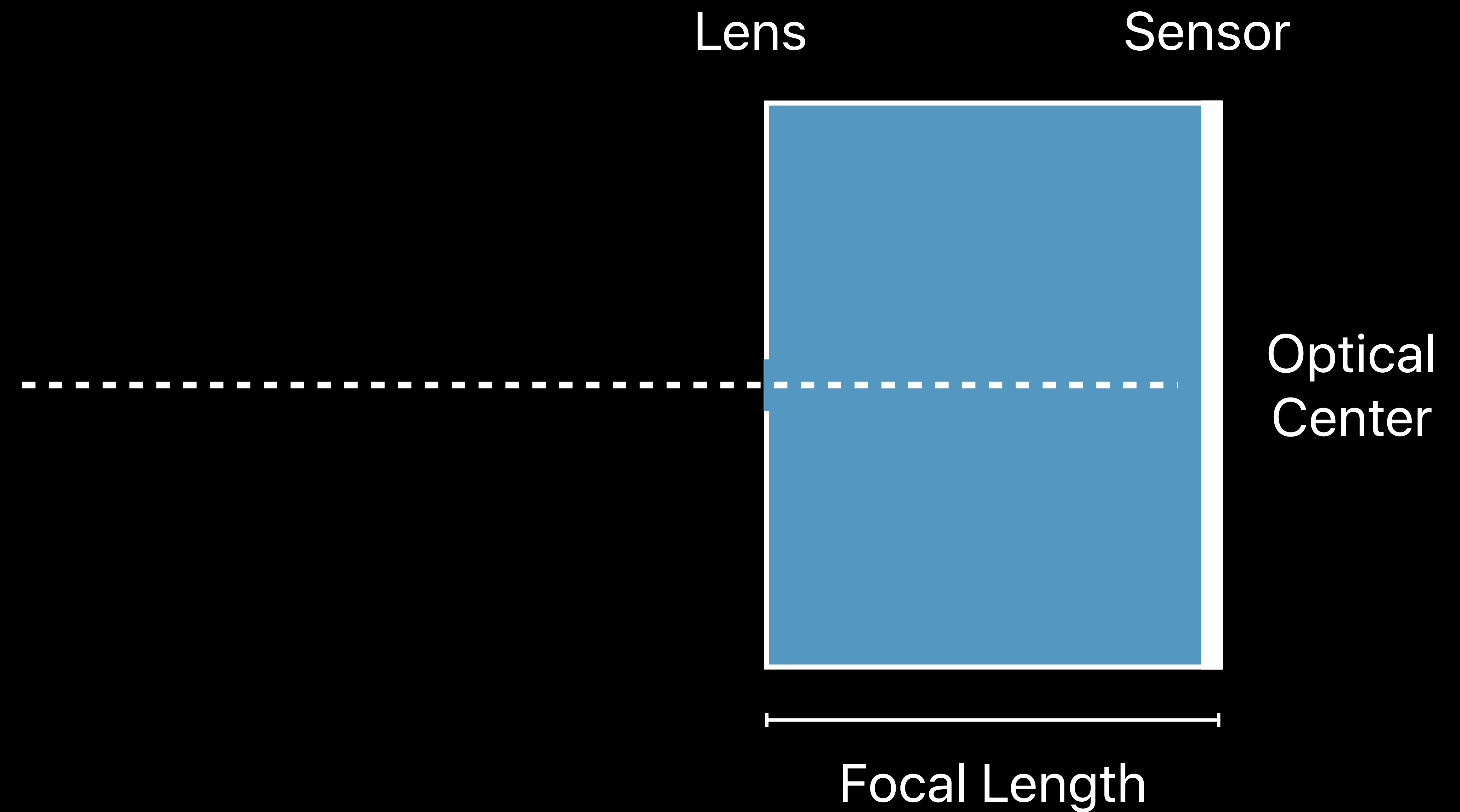
Camera Intrinsics

Camera Extrinsics



# Camera Intrinsics

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$



kCMSampleBufferAttachmentKey\_CameraIntrinsicMatrix

# Camera Extrinsics

NEW

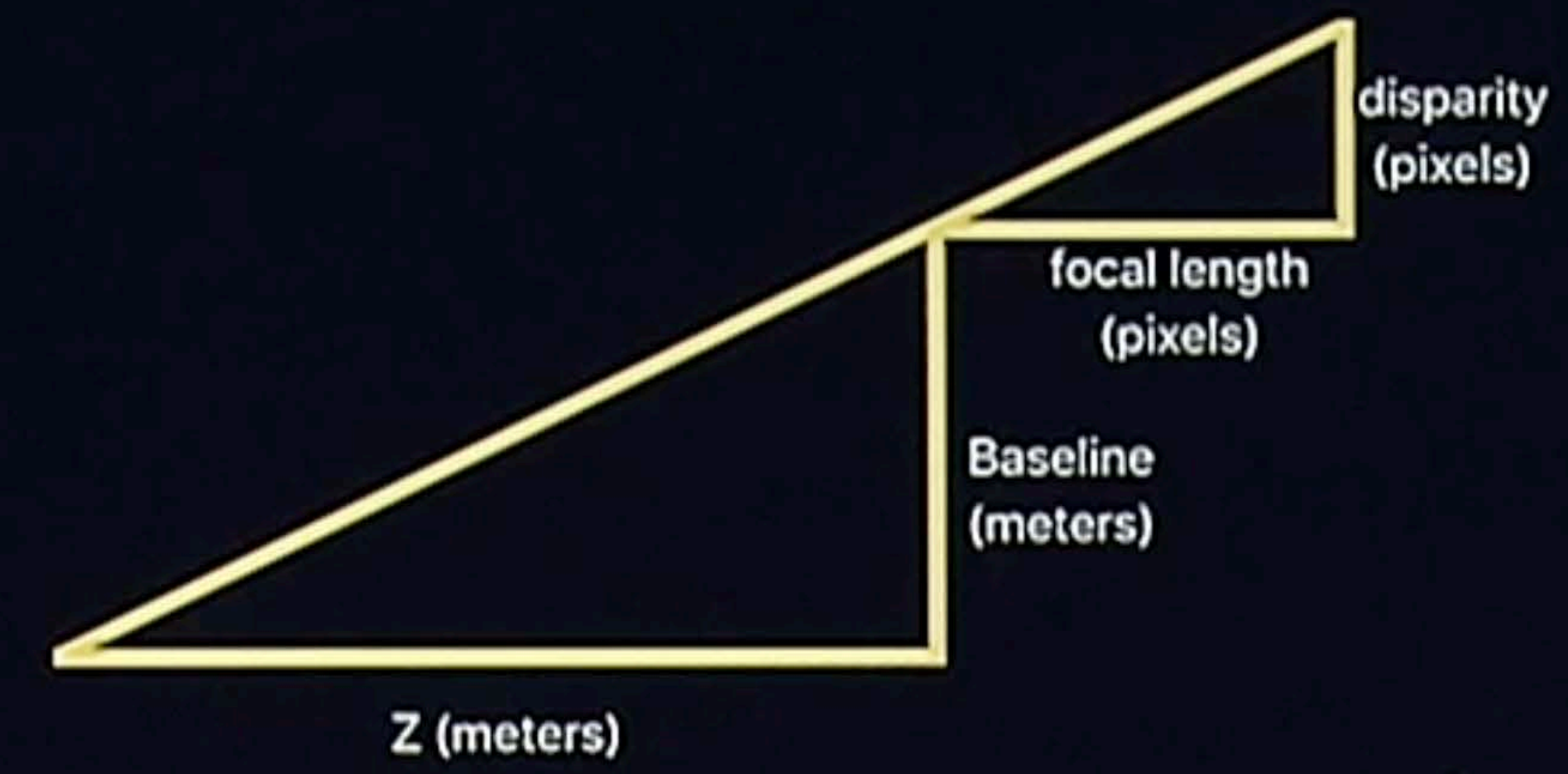
```
// Get extrinsics
if let wide = AVCaptureDevice.default(.builtInWideAngleCamera, for: nil, position: .back),
    let tele = AVCaptureDevice.default(.builtInTelephotoCamera, for: nil, position: .back) {
    self.extrinsics = AVCaptureDevice.extrinsicMatrix(from: tele, to: wide)
}
```

$$[R | t] = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{bmatrix}$$

Rotation                      Translation



## Removing Despair from Disparity



$$\frac{b}{z} = \frac{d}{fl}$$



# Multi-Camera Capture

## Multi-Microphone Capture





# Review: AVCaptureSession Mic Selection

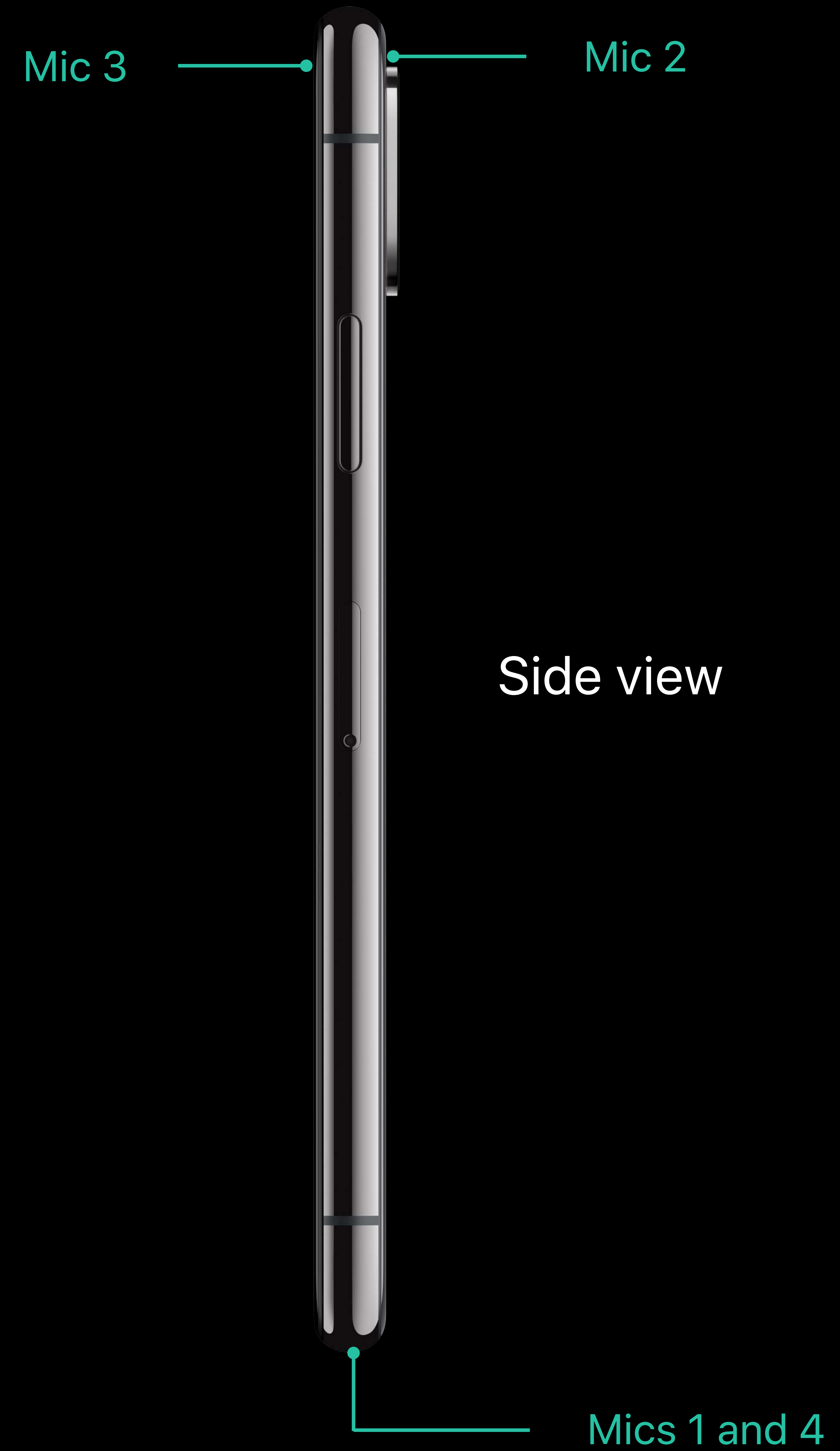
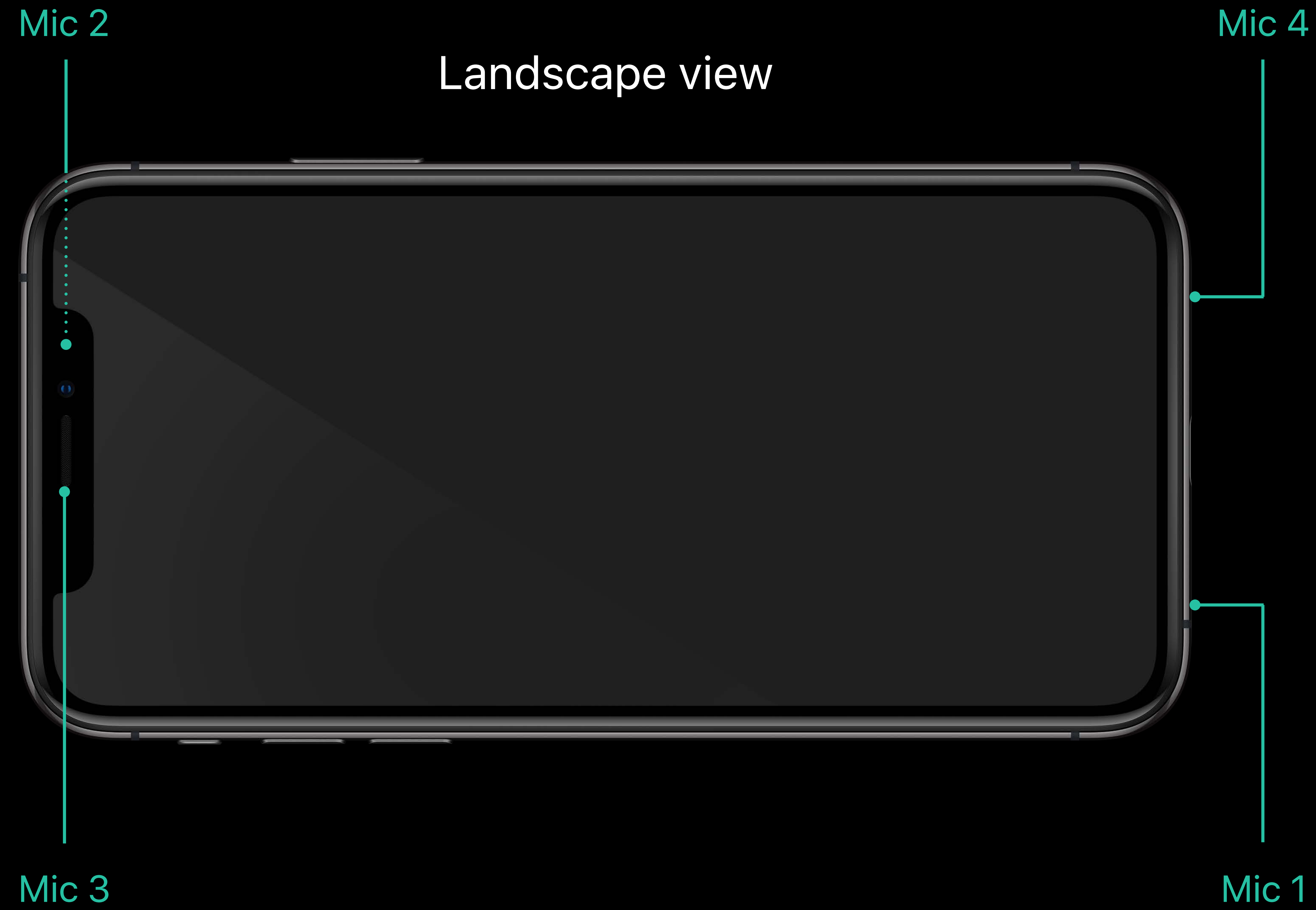
## Default behaviors

- Front camera in use = selects front microphone
- Back camera in use = selects back microphone
- Audio only session = selects omnidirectional mic (usually at the bottom)

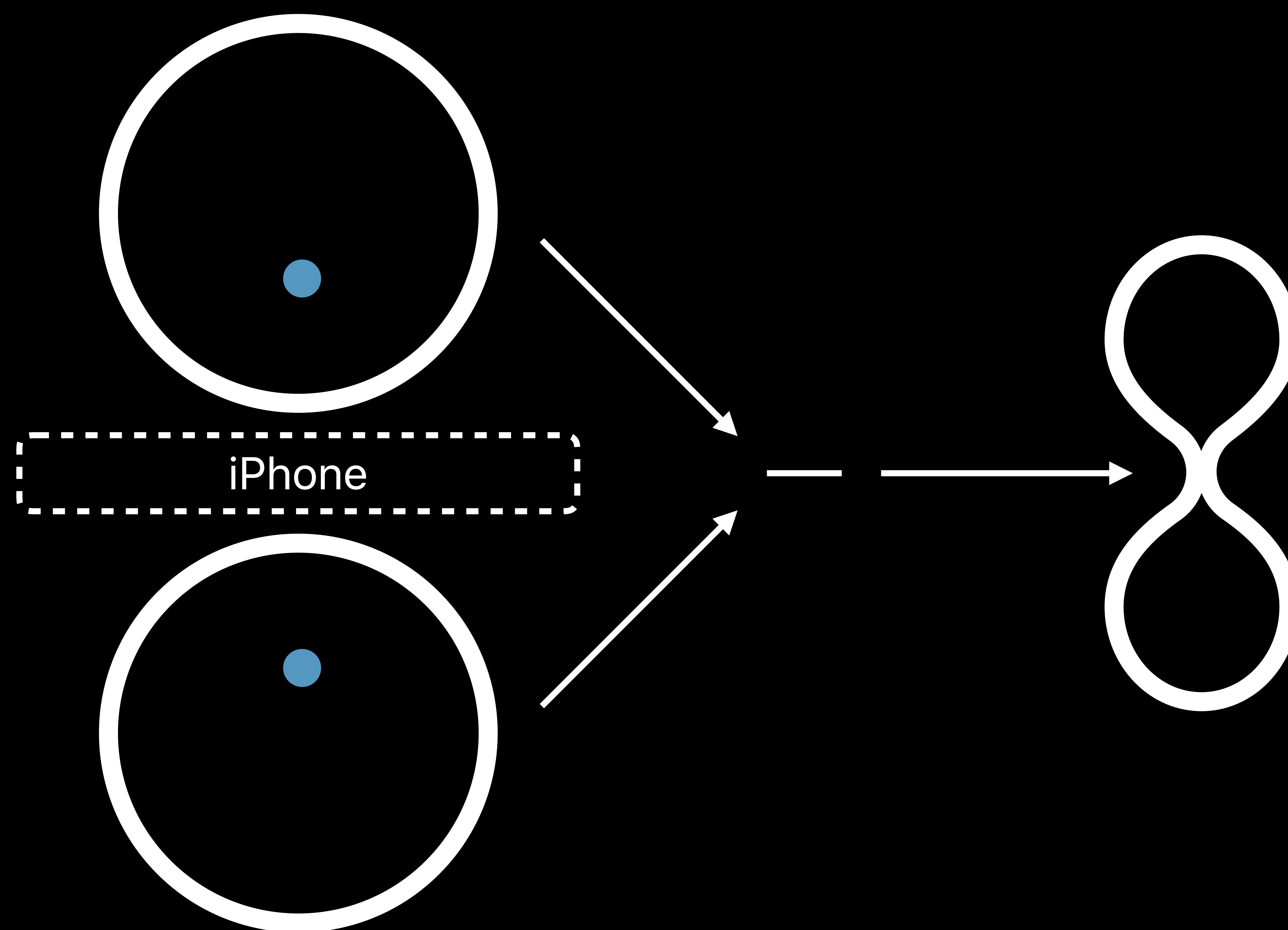
Power feature — app can use AVAudioSession to configure mics

There's no such thing as a "front mic".

# iPhone Microphone Arrays

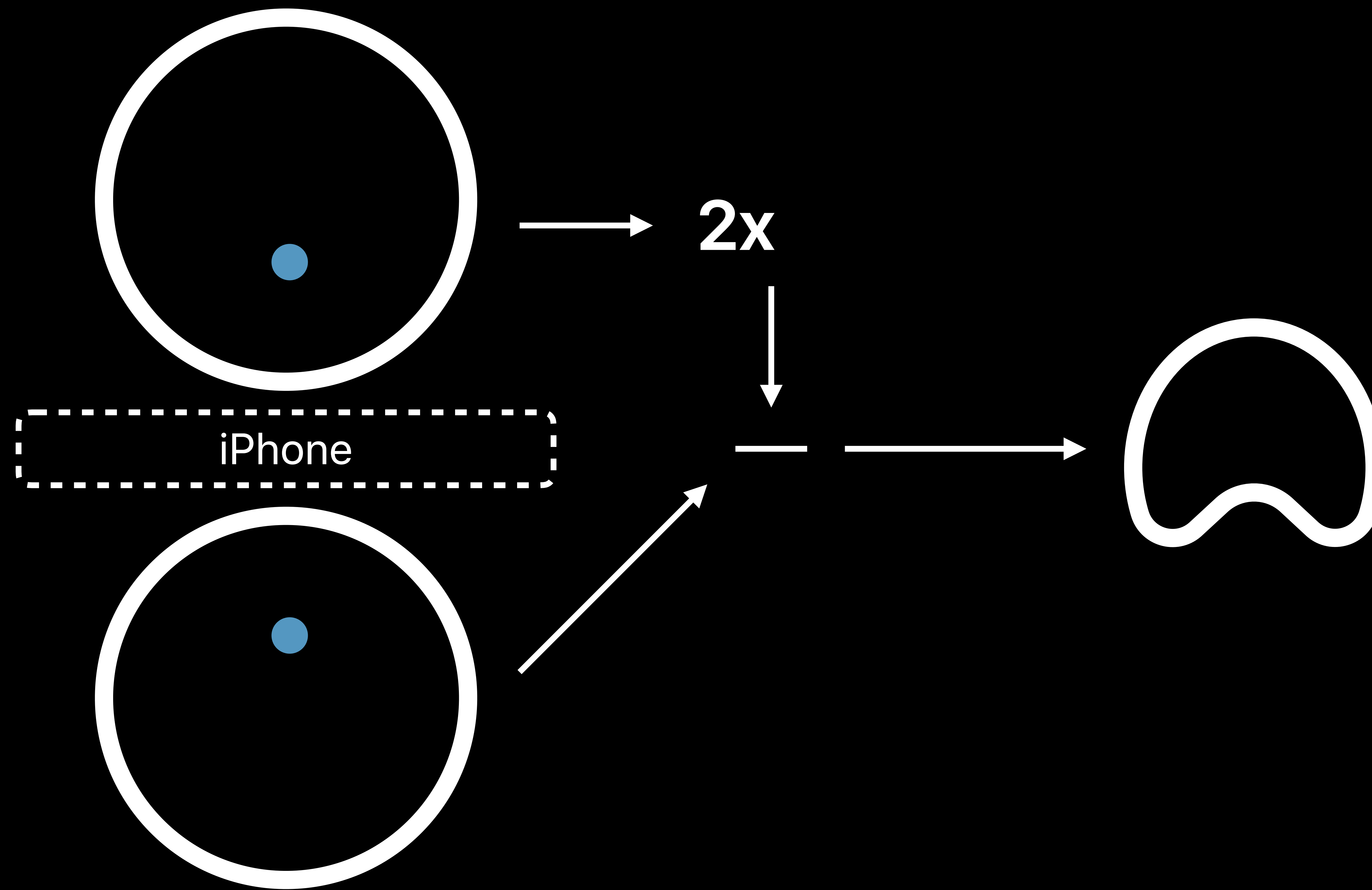


# Microphone Beam Forming

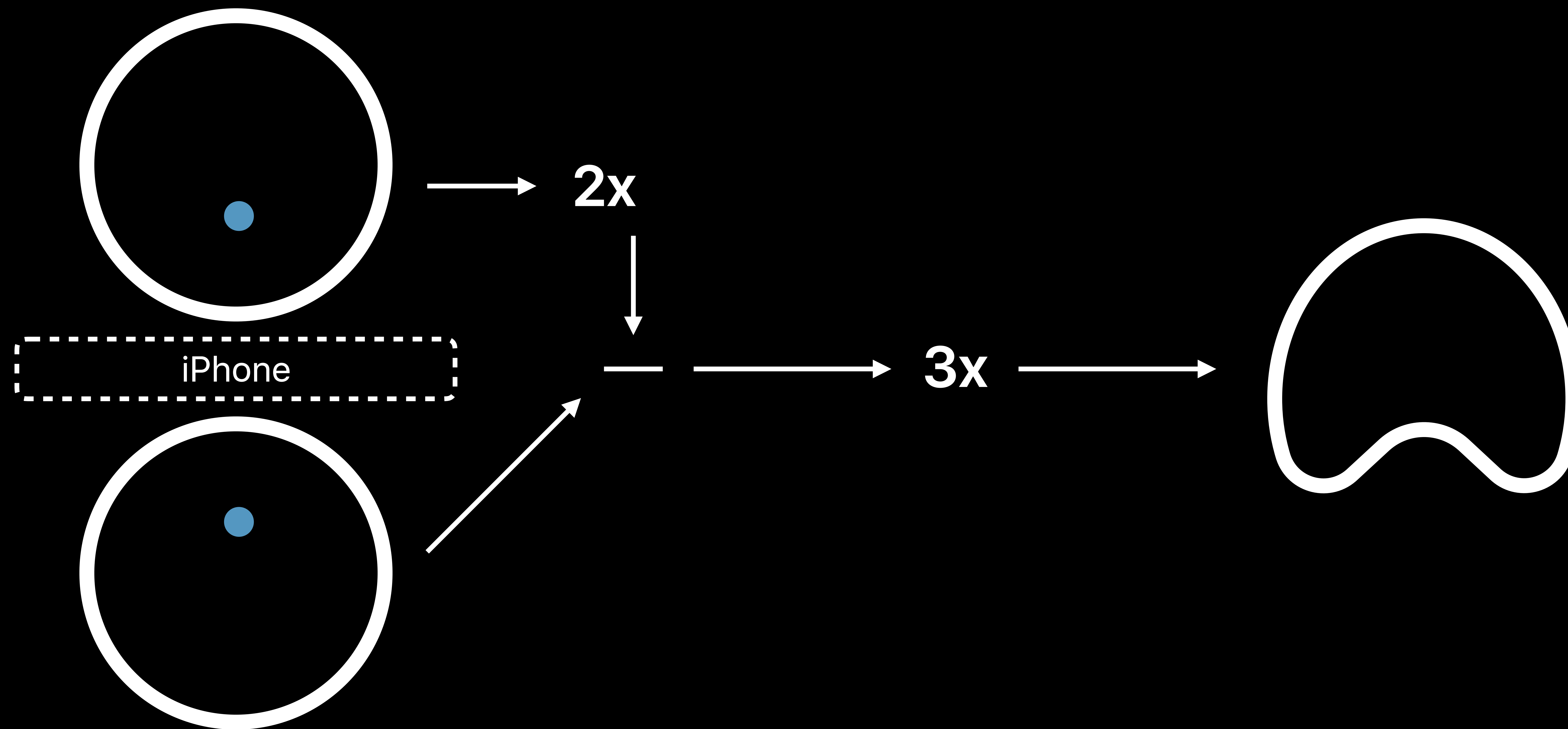




# Microphone Beam Forming



# Microphone Beam Forming



# AVCaptureSession Mic Input Port Behavior

AVCaptureDeviceInput (Microphone)

Primary Port  
(Front / Back / Omni)

# MultiCam Behavior: The Return of the Secret Ports

AVCaptureDeviceInput (Microphone)

Primary Port  
(Omni)

Back Beam-Formed

Back Beam-Formed



# Positional Audio Connections

```
let frontPort = micInput.ports(for: .audio,  
                               sourceDeviceType: micDevice.deviceType,  
                               sourceDevicePosition: .front).first
```

```
let backPort = micInput.ports(for: .audio,  
                               sourceDeviceType: micDevice.deviceType,  
                               sourceDevicePosition: .back).first
```

# Positional Audio Connections

```
let frontPort = micInput.ports(for: .audio,  
                               sourceDeviceType: micDevice.deviceType,  
                               sourceDevicePosition: .front).first
```

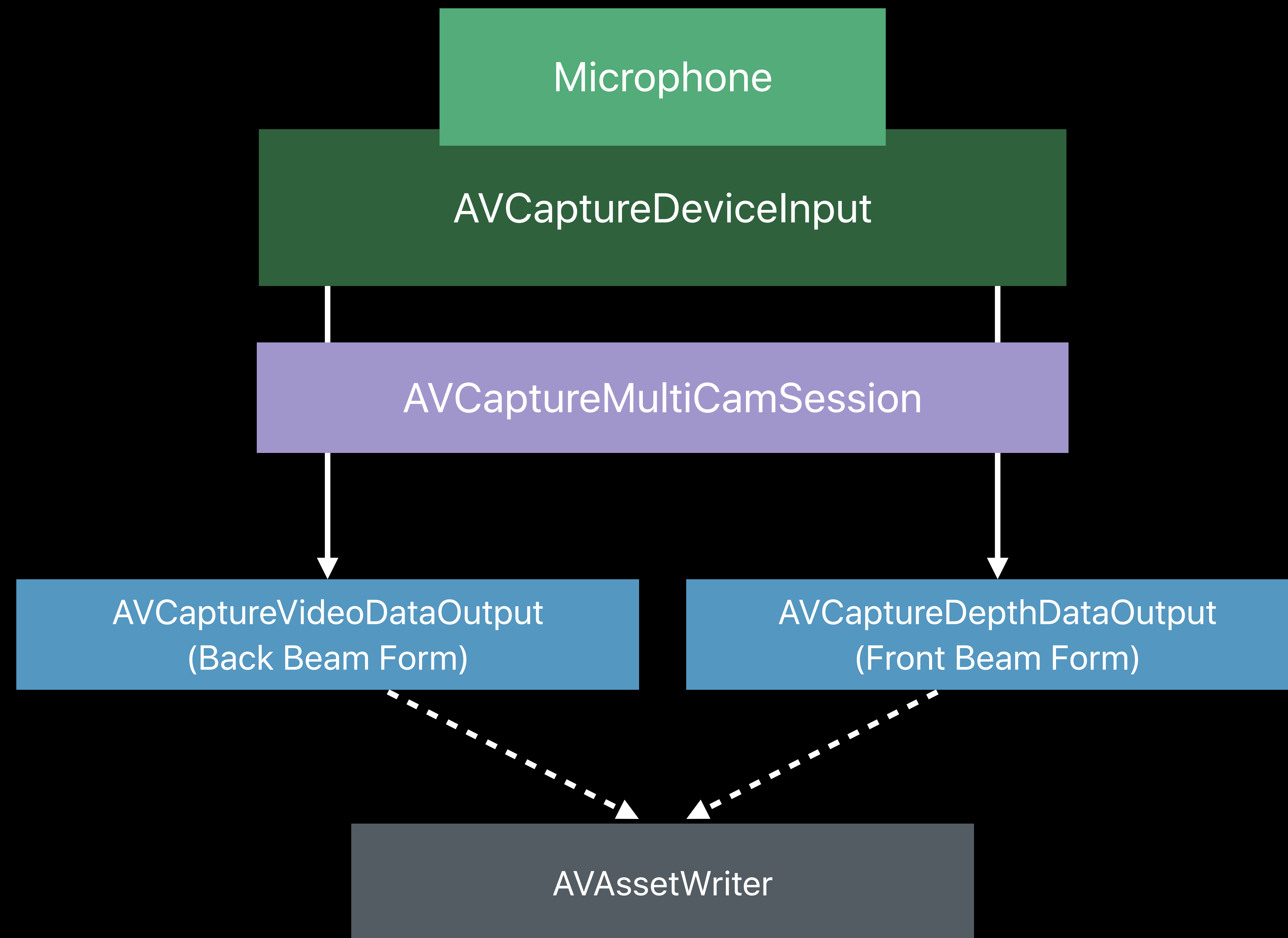
```
let backPort = micInput.ports(for: .audio,  
                               sourceDeviceType: micDevice.deviceType,  
                               sourceDevicePosition: .back).first
```

# Positional Audio Connections

```
let frontPort = micInput.ports(for: .audio,  
                               sourceDeviceType: micDevice.deviceType,  
                               sourceDevicePosition: .front).first
```

```
let backPort = micInput.ports(for: .audio,  
                               sourceDeviceType: micDevice.deviceType,  
                               sourceDevicePosition: .back).first
```

# Multi-Mic Beam Forming in AVMultiCamPiP





# Non Built-In Mic Behaviors

Beam-forming only works with built-in mics

External audio signal is duplicated to all audio input ports (omni, front, and back)

# Multi-Camera Capture Summary

# Multi-Camera Capture Summary

Use `AVCaptureMultiCamSession`

# Multi-Camera Capture Summary

Use `AVCaptureMultiCamSession`

Know its limitations



# Multi-Camera Capture Summary

Use `AVCaptureMultiCamSession`

Know its limitations

Thoughtfully handle hardware and system pressure costs

# Multi-Camera Capture Summary

Use `AVCaptureMultiCamSession`

Know its limitations

Thoughtfully handle hardware and system pressure costs

Use constituent device ports for synchronized camera streaming

# Multi-Camera Capture Summary

Use `AVCaptureMultiCamSession`

Know its limitations

Thoughtfully handle hardware and system pressure costs

Use constituent device ports for synchronized camera streaming

Use front, back, or omni mics simultaneously

# And Now For Something Completely Different

## Changes to AVCapturePhotoOutput

- Deprecation of auto still image stabilization
- Addition of photo quality versus speed hinting

Video coming soon to [developer.apple.com](https://developer.apple.com)



# Semantic Segmentation Mattes

Jacob Schack Vestergaard, Camera Software  
David Hayward, Core Image



PORTRAIT



DEPTH

$f4.5$



Cancel



Done



# iOS12: PortraitEffectsMatte





# iOS12: PortraitEffectsMatte





# Semantic Segmentation Mattes

NEW





# Semantic Segmentation Mattes

NEW





# Semantic Segmentation Mattes

NEW





# Semantic Segmentation Mattes

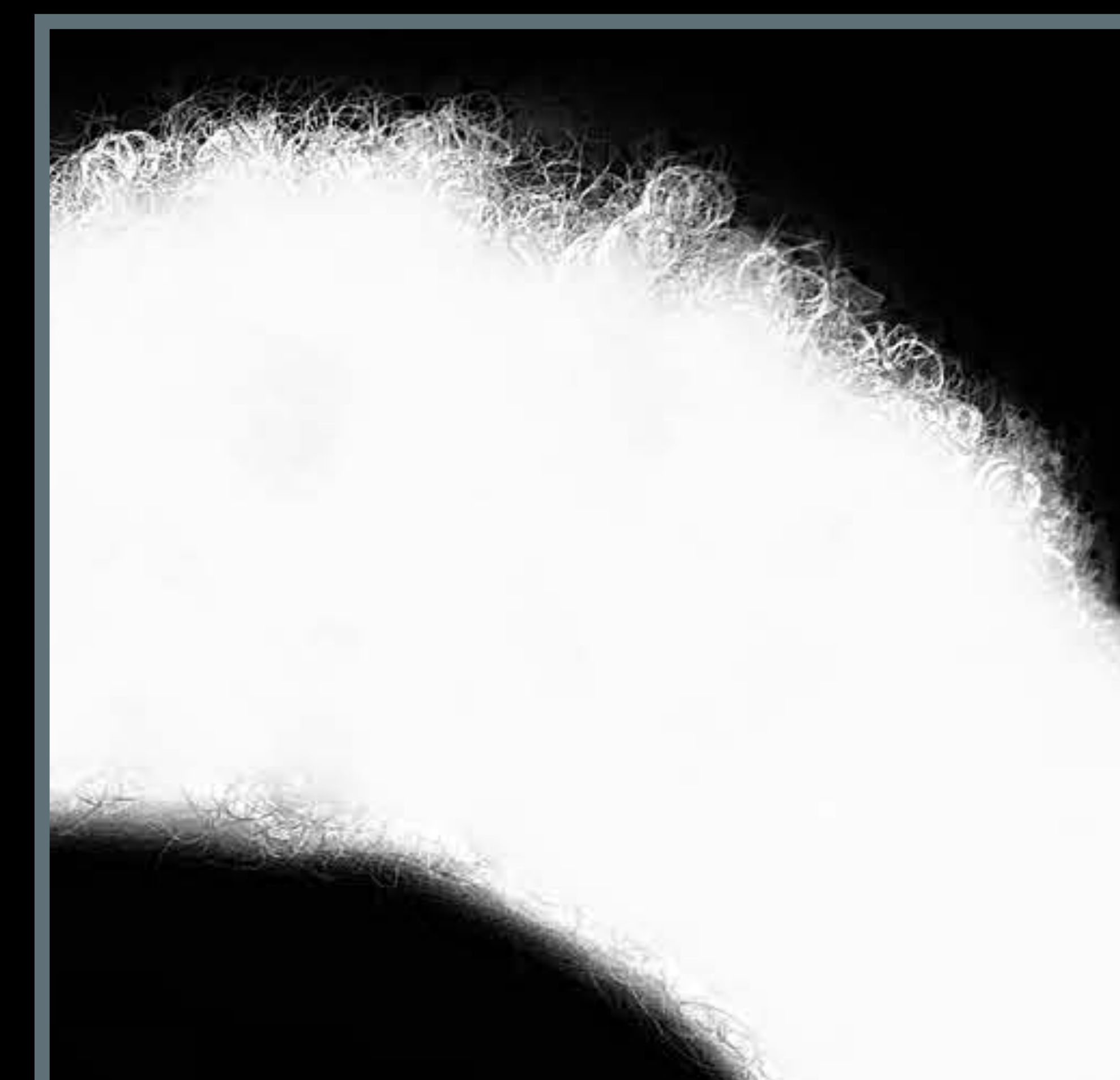
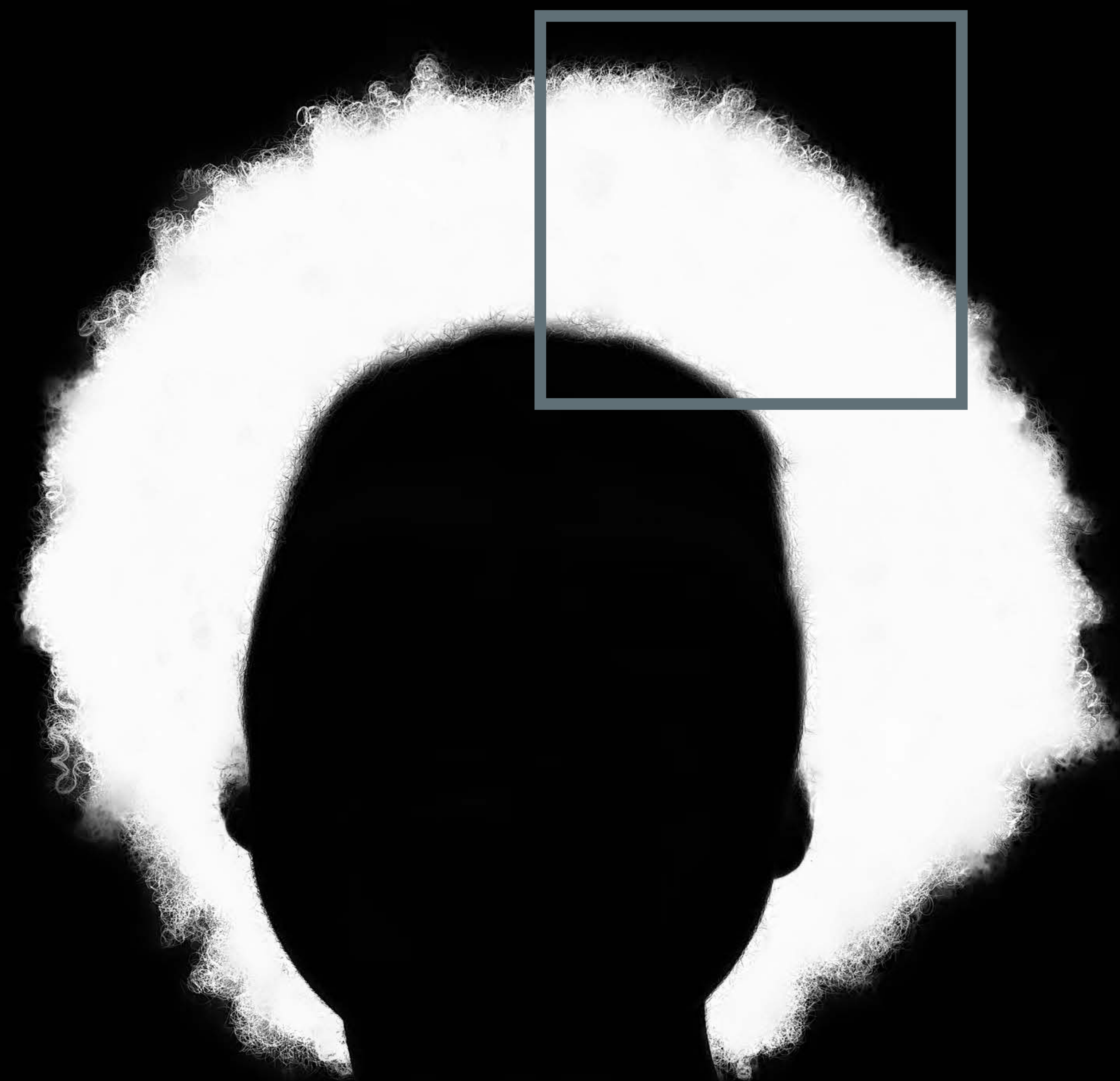
NEW





# Semantic Segmentation Mattes

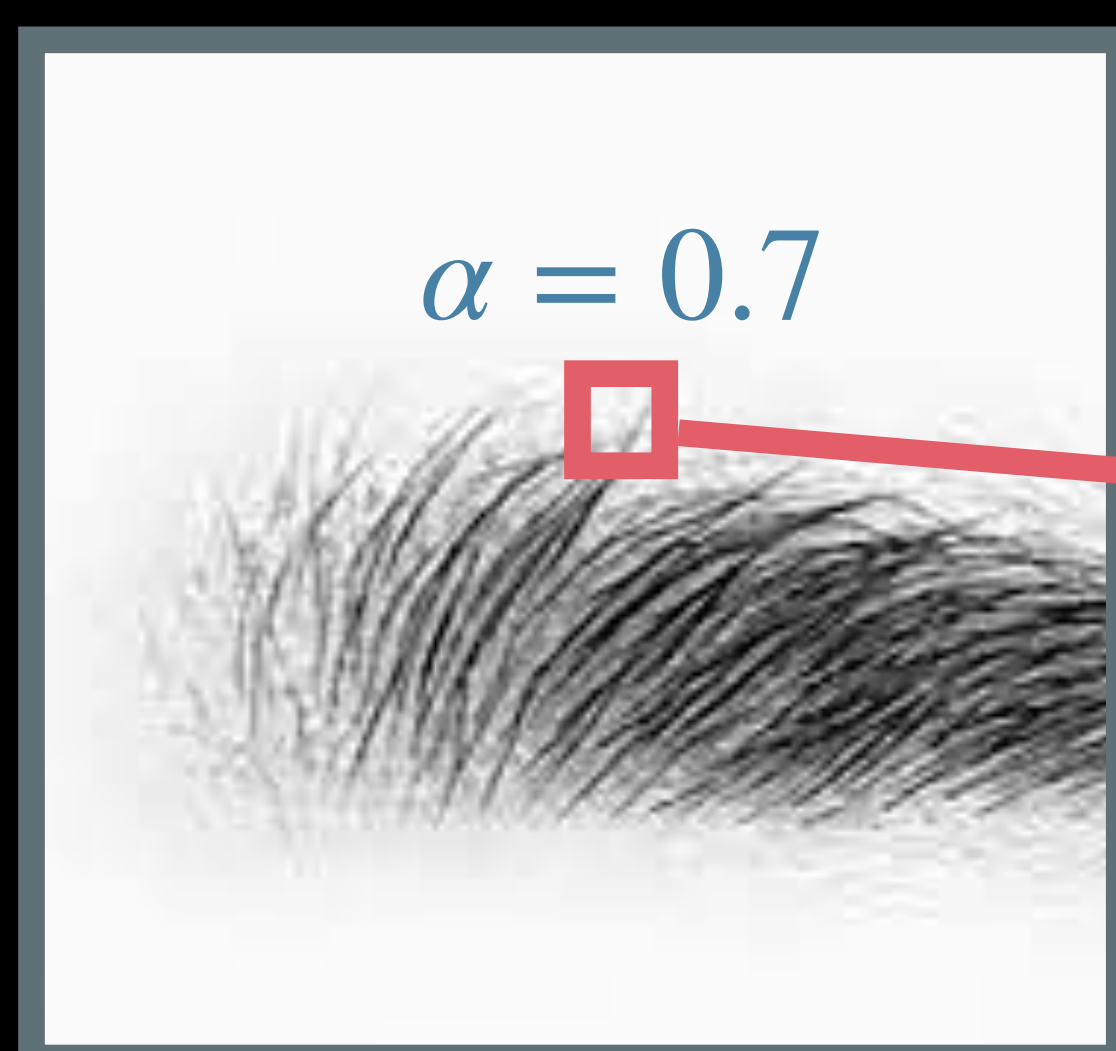
NEW





# Semantic Segmentation Mattes

NEW



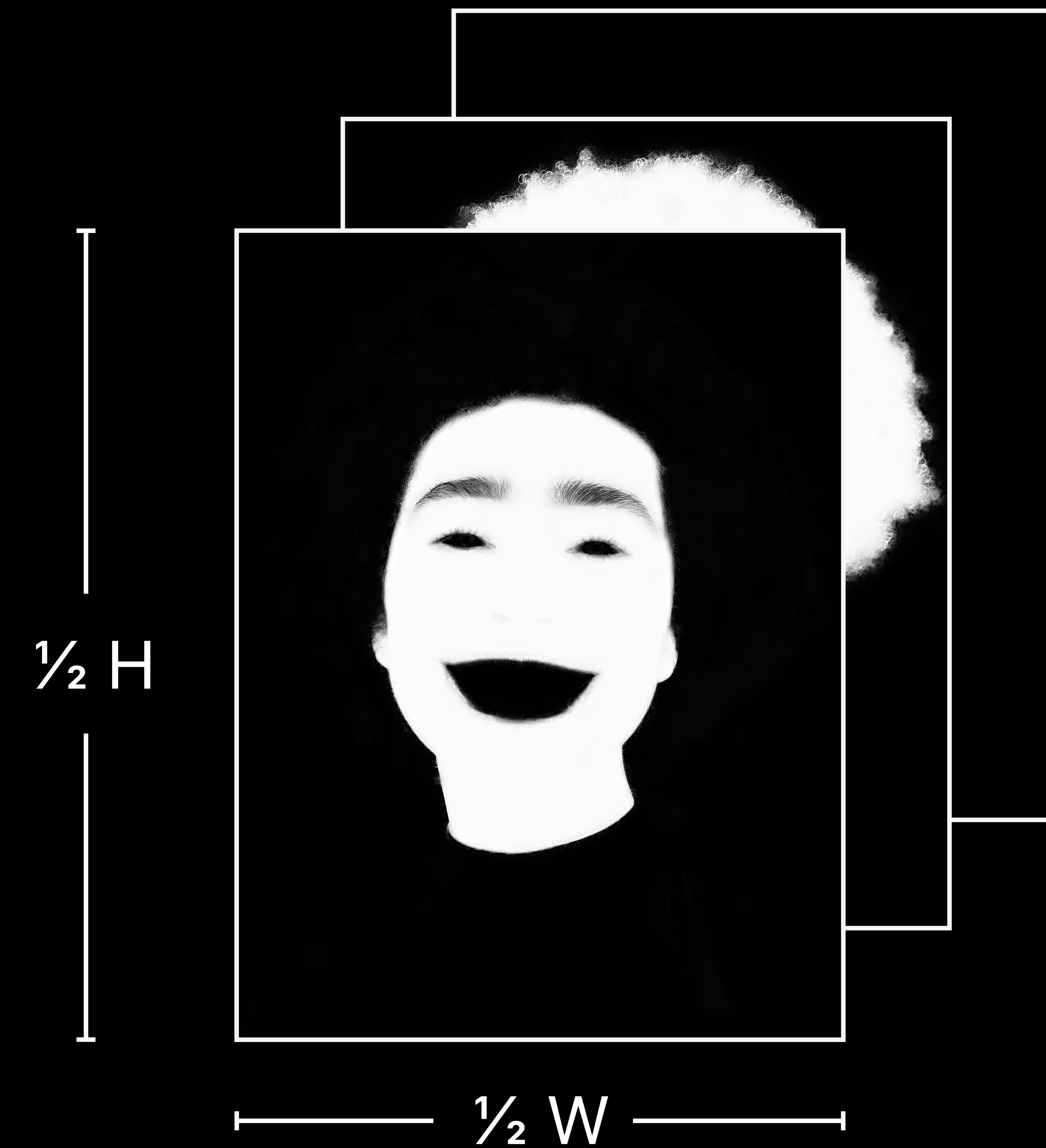


# Mattes Are Half-Size



W

H



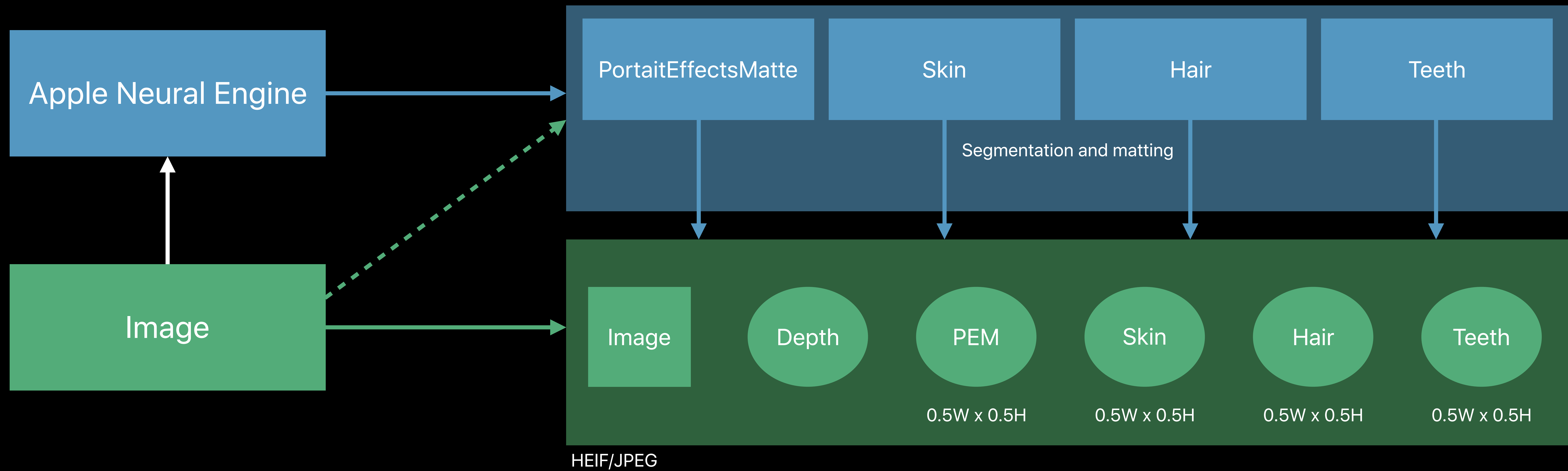




B I O N I C



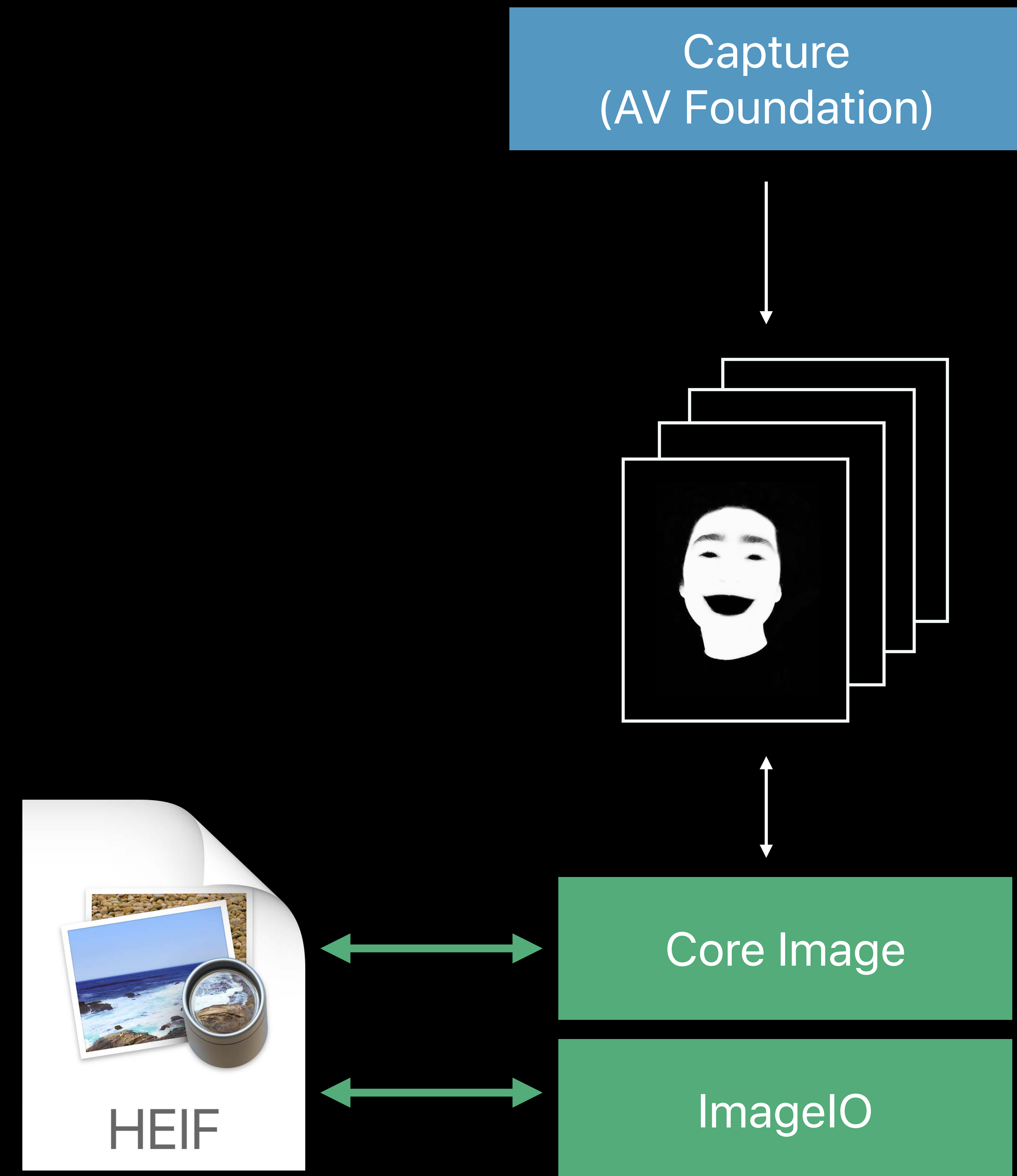
# Under the Hood



# Where Do the Mattes Come From?

Embedded in Portrait Mode captures

Write your own capture app and opt-in



# More Information

[developer.apple.com/wwdc19/249](https://developer.apple.com/wwdc19/249)

---

Capturing Depth in iPhone Photography

WWDC 2017

---

Introducing the Photos Frameworks

WWDC 2014

---

