

iOS Dev Accelerator

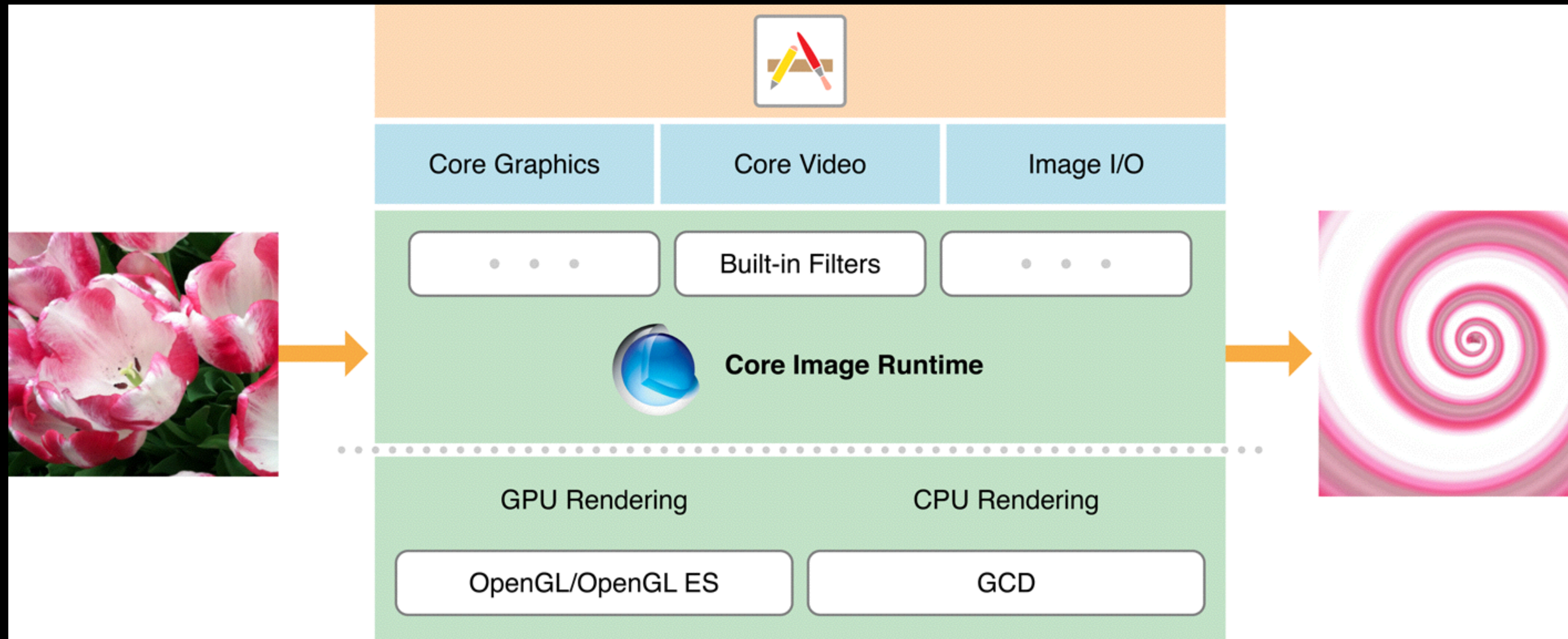
Week 3 Day 3

- Core Image
- AVFoundation

CoreImage

- “Core Image is an image processing and analysis technology designed to provide near real-time processing for still and video images”
- Can use either the GPU or CPU
- “Core Image hides the details of low-level graphic processing....You don’t need to know the details of OpenGL/ES to leverage the power of the GPU”

CoreImage



CoreImage Offerings

- Built-in image processing filters (90 on iOS)
- Feature detection capability
- Support for automatic image enhancement
- Ability to chain multiple filters together to create custom effects



guy using CoreImage

Filtering



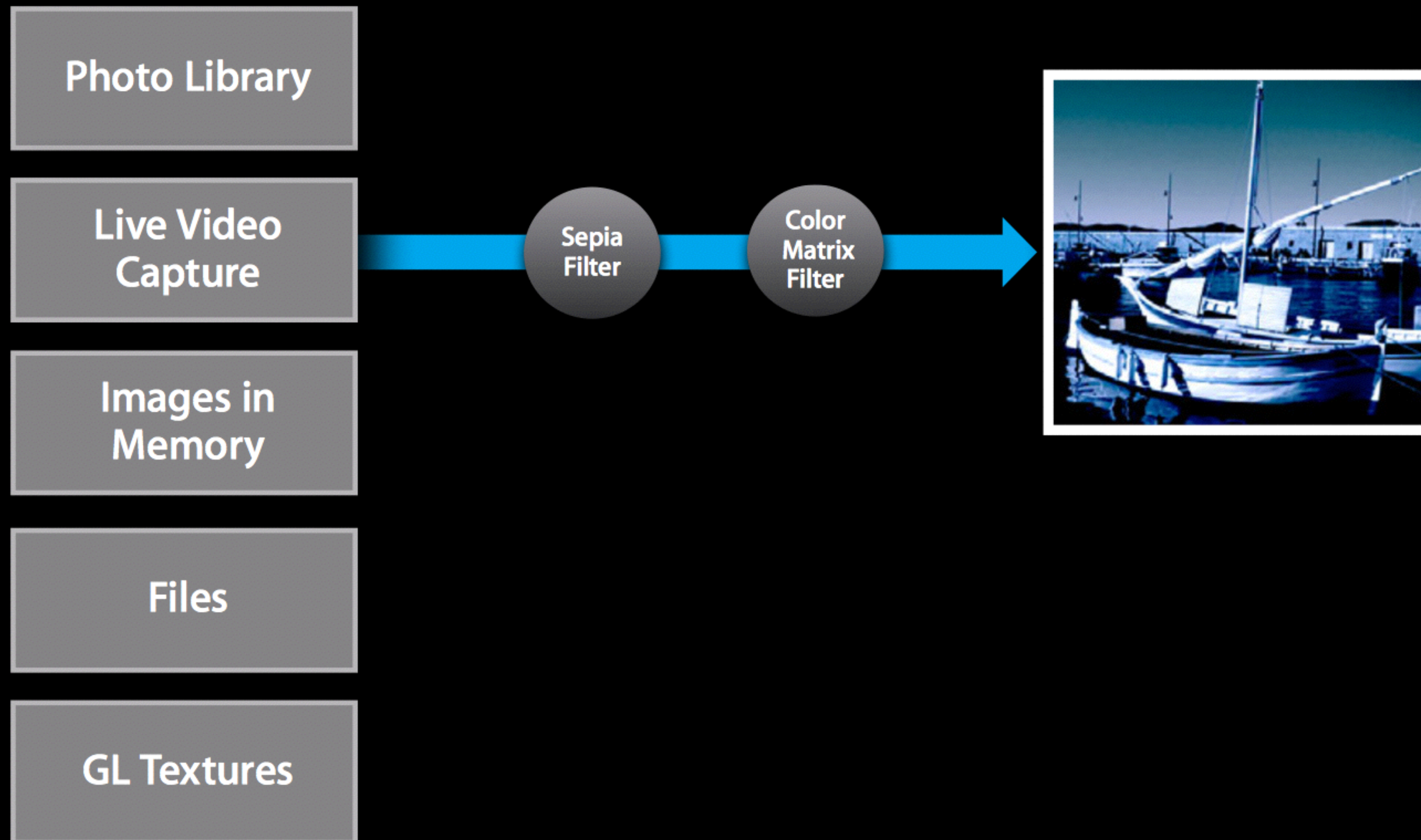
Original



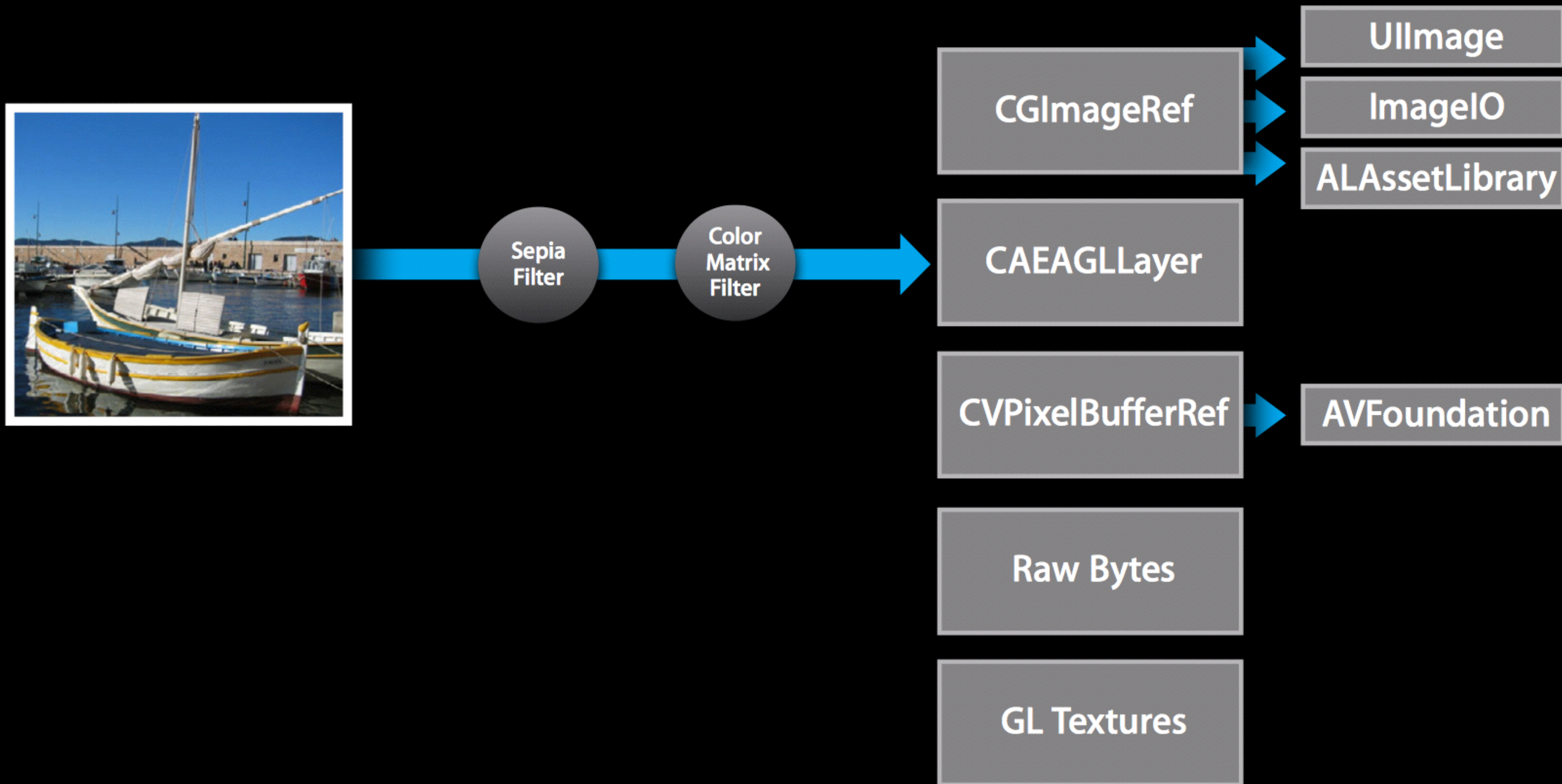
Result

- Filters applied on a per pixel basis
- Can be chained together

Filtering Inputs are Flexible



As are the Outputs



CIAdditionCompositing	CIColorPosterize	CI Gaussian Gradient	CI Minimum Compositing	CI Source In Compositing
CI Affine Clamp	CI Constant Color Generator	CI Glide Reflected Tile	CI Mod Transition	CI Source Out Compositing
CI Affine Tile	CI Copy Machine Transition	CI Gloom	CI Multiply Blend Mode	CI Source Over Compositing
CI Affine Transform	CI Crop	CI Hard Light Blend Mode	CI Multiply Compositing	CI Star Shine Generator
CI Bars Swipe Transition	CI Darken Blend Mode	CI Hatched Screen	CI Overlay Blend Mode	CI Straighten Filter
CI Blend With Mask	CI Difference Blend Mode	CI Highlight Shadow Adjust	CI Perspective Tile	CI Stripes Generator
CI Bloom	CI Disintegrate With Mask	CI Hole Distortion	CI Perspective Transform	CI Swipe Transition
CI Checkerboard Generator	CI Dissolve Transition	CI Hue Adjust	CI Pinch Distortion	CI Temperature And Tint
CI Circle Splash Distortion	CI Dot Screen	CI Hue Blend Mode	CI Pixellate	CI Tone Curve
CI Circular Screen	CI Eightfold Reflected Tile	CI Lanczos Scale Transform	CI Radial Gradient	CI Triangle Kaleidoscope
CI Color Blend Mode	CI Exclusion Blend Mode	CI Lighten Blend Mode	CI Random Generator	CI Twelfefold Reflected Tile
CI Color Burn Blend Mode	CI Exposure Adjust	CI Light Tunnel	CI Saturation Blend Mode	CI Twirl Distortion
CI Color Controls	CI False Color	CI Linear Gradient	CI Screen Blend Mode	CI Unsharp Mask
CI Color Cube	CI Flash Transition	CI Line Screen	CI Sepia Tone	CI Vibrance
CI Color Dodge Blend Mode	CI Fourfold Reflected Tile	CI Luminosity Blend Mode	CI Sharpen Luminance	CI Vignette
CI Color Invert	CI Fourfold Rotated Tile	CI Mask To Alpha	CI Sixfold Reflected Tile	CI Vortex Distortion
CI Color Map	CI Fourfold Translated Tile	CI Maximum Component	CI Sixfold Rotated Tile	CI White Point Adjust
CI Color Matrix	CI Gamma Adjust	CI Maximum Compositing	CI Soft Light Blend Mode	
CI Color Monochrome	CI Gaussian Blur	CI Minimum Component	CI Source Atop Compositing	



CIImage

- An Immutable object that represents the recipe for an Image
- Can represent a file from disk or the output of a CIFilter
- Multiple ways to create one:

```
var image = CIImage(contentsOfURL: url)
```

```
var anotherImage = CIImage(image: UIImage())
```

Also has inits from Raw bytes,
NSData, CGImage, PixelBuffers, etc

CIFilter

- Mutable object that represents a filter
- Produces an output image based on the input

```
var filter = CIFilter(name: "CISepiaTone")  
filter.setValue(image, forKey: kCIInputImageKey)  
filter.setValue(NSNumber(float: 0.8), forKey: @"inputIntensity")
```


CIText

- An object through which Core Image draws results
- Can be based on CPU or GPU

```
var context = CIText(options: nil)
var result = filter.valueForKey(kCIOutputImageKey) as CIImage
var imageRef = context.createCGImage(result, fromRect: result.extent()) as
    CGImageRef
var finalImage = UIImage(CGImage: imageRef)
```

CIKernel

- Custom Core Image Kernels are supported as of iOS 8
 - Built using Core Image Kernel language
 - similar to OpenGL ES2.0 shaders
 - used for advanced Core Image work
 - docs not published yet

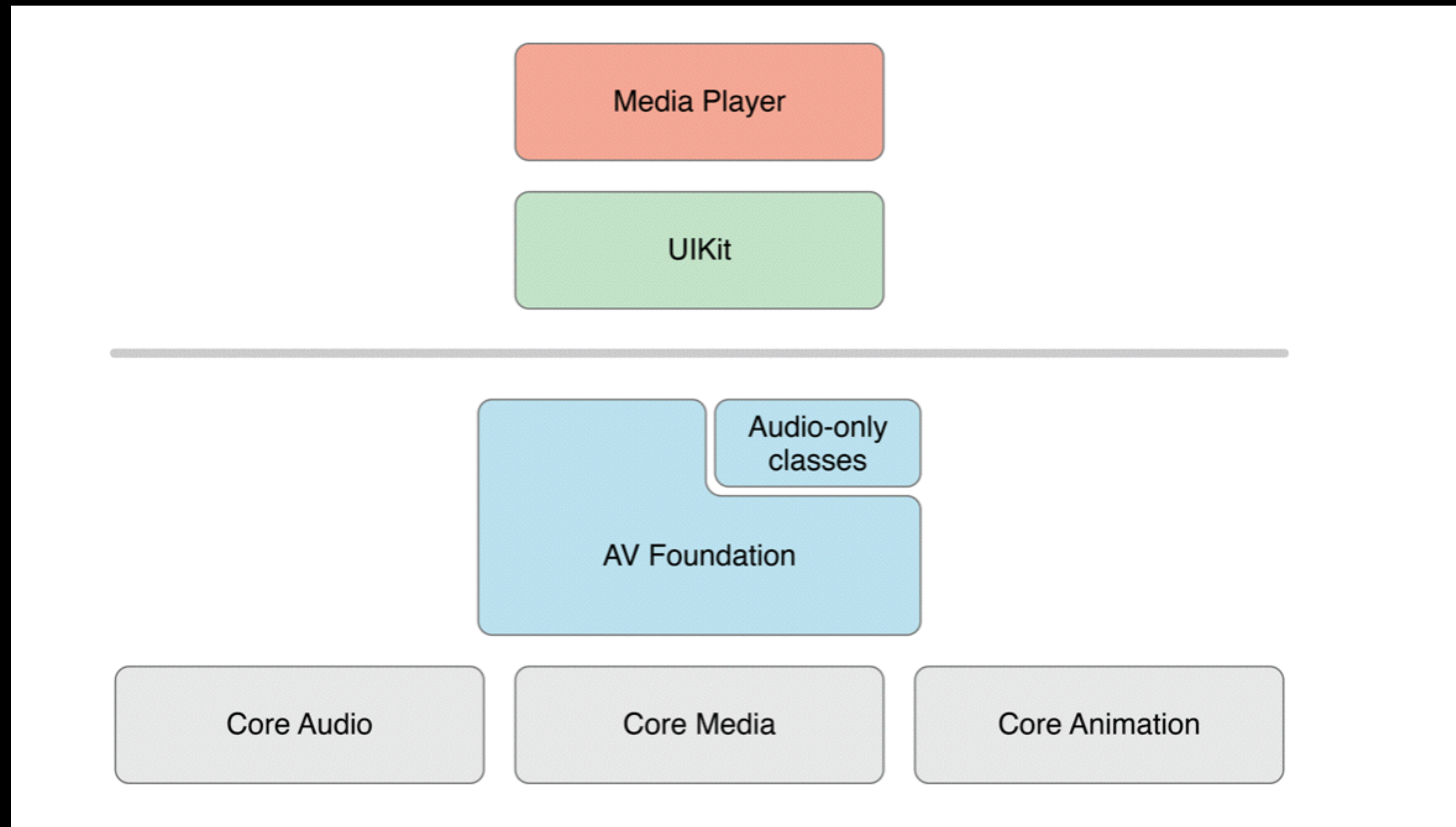
IBDesignable

- Create custom interface builder components
- Designable views must be in a separate framework
- Prefix your class declaration with @IBDesignable

IBInspectable

- Exposes attributes in `InterfaceBuilder`
- Prefix your variable declaration with `@IBInspectable`
- Add `didSet` method to the variable
- Set `self.property`, then update the view based on the changes

AVFoundation

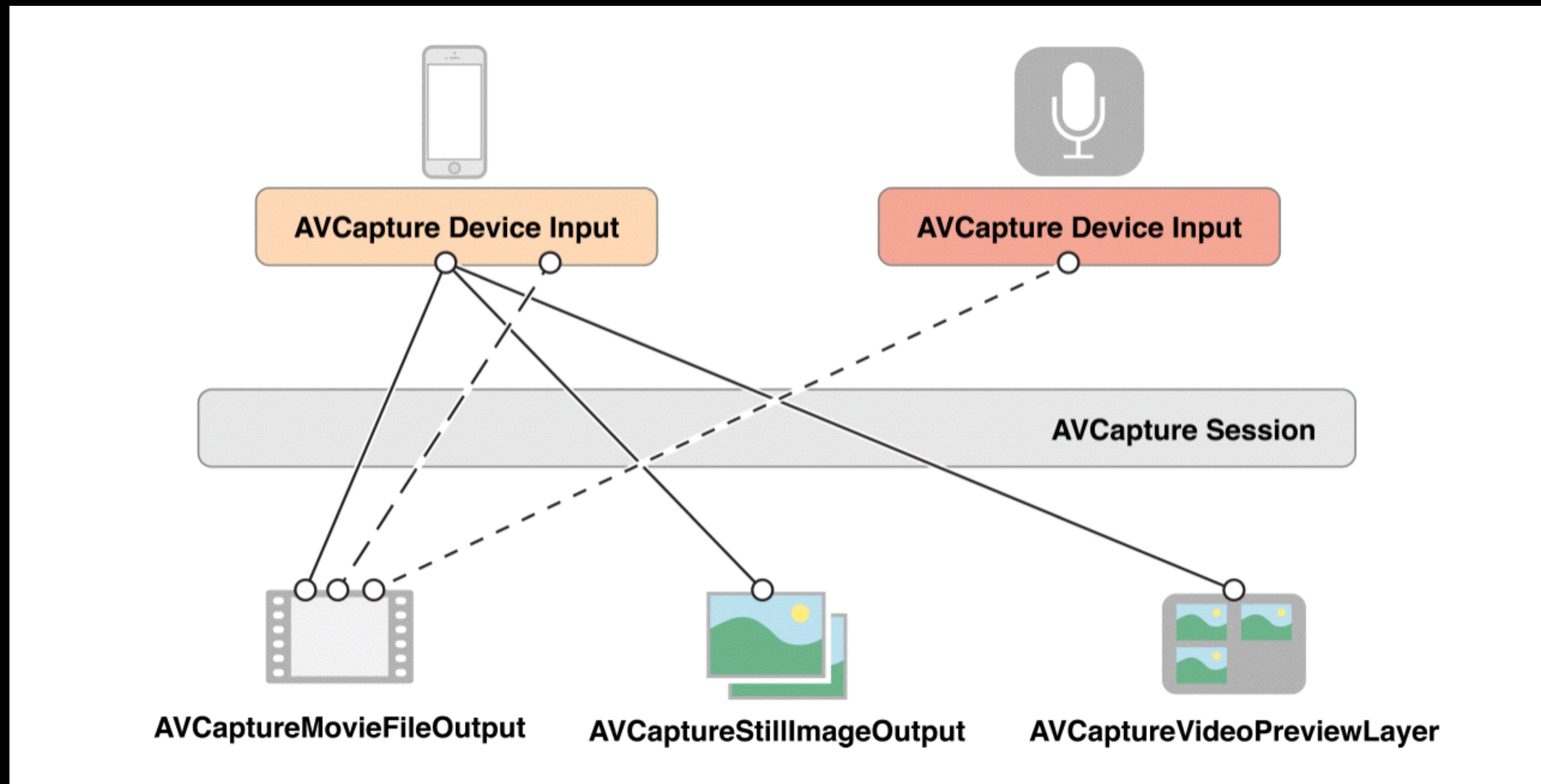


- A framework used to play and create time-based audiovisual media.

AVFoundation Assets

- The primary model class that AVFoundation uses to represent media is AVAsset.
- AVAsset is an aggregated representation of one or more pieces of media data.
- Provides info like the title,duration,natural size,etc.
- Each piece of media data inside the asset is considered a track.
- An asset or track that has been initialized may not ready to be used right away, so the API is highly asynchronous using callbacks.

AVFoundation Capturing



- To manage the capture of media, you create objects to represent inputs and outputs.
- You then use an instance of `AVCaptureSession` to coordinate the flow of data between them.

AVFoundation Capturing

- You will need the following objects setup for capture:
 - An an instance of `AVCaptureDevice` to represent the input device, like the phones camera or mic.
 - An instance of `AVCaptureInput` to configure the ports from the iput device.
 - An instance of `AVCaptureOutput` to manage the output to a movie file or still image.
 - An instance of `AVCaptureSession` to coordinate the flow of data from input to output.
 - An instance of `AVCaptureVideoPreviewLayer` to show the user a preview of what the camera is recording.

AVCaptureDevice

- Represents a physical capture device and the properties associated with the device.
- An instance of AVCapture devices allows you to configure the underlying device.
- Provides input data to an AVCaptureSession

AVDeviceInput

- A concrete sub-class of AVCaptureInput.
- Used to capture data from an AVCapture Device.
- initWithDevice:Error:

AVCaptureSession

- You use this class to coordinate the flow of data from input devices to outputs.
- use `addInput:` and `addOutput:` methods to add those streams.
- tell a session to `startRunning()` when everything is configured.
- Run all session setup and `startRunning` on a background queue because it is potentially blocking and we want to keep the interface responsive to the user.

AVMetadataObject

- Contains additional data about the image/frame of video
- AVMetadataFaceObject
 - defines features for a single detected face
 - properties include rollAngle, yawAngle, & faceID
- AVMetadataMachineReadableCodeObject
 - features of a detected 1d or 2d barcode