What's New in Core Image

Session 510

David Hayward Engineering Manager
Tony Chu Engineer
Alexandre Naaman Lead Engineer

What We Will Cover Today

A Brief Introduction to Core Image

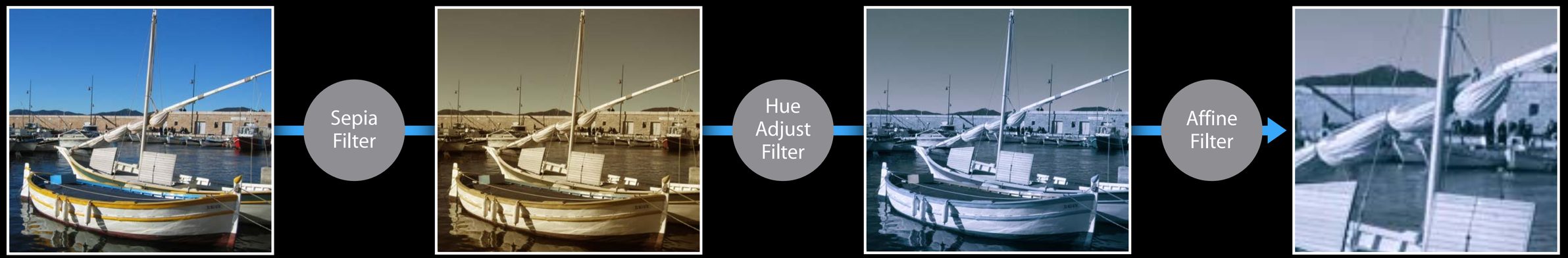
What's New in Core Image

Bridging Core Image with Other Frameworks

A Brief Introduction to Core Image

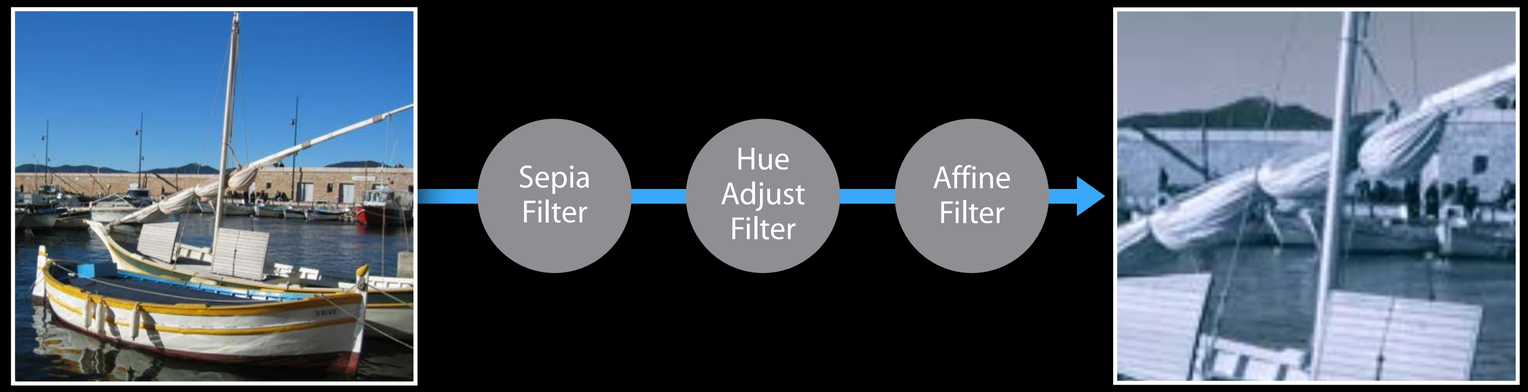
Key Concepts

Filters can be chained together for complex effects



Key Concepts

Intermediate images are lightweight objects

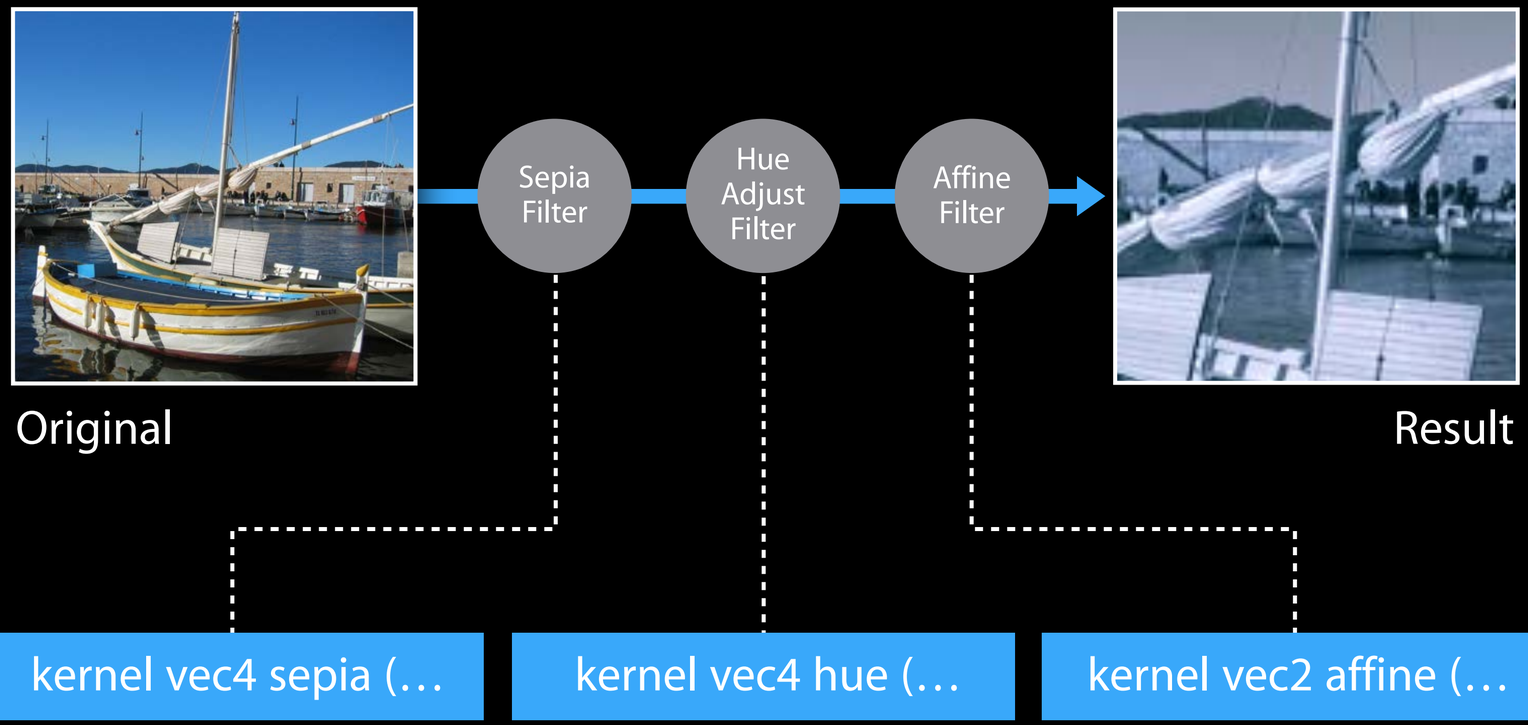


Original

Result

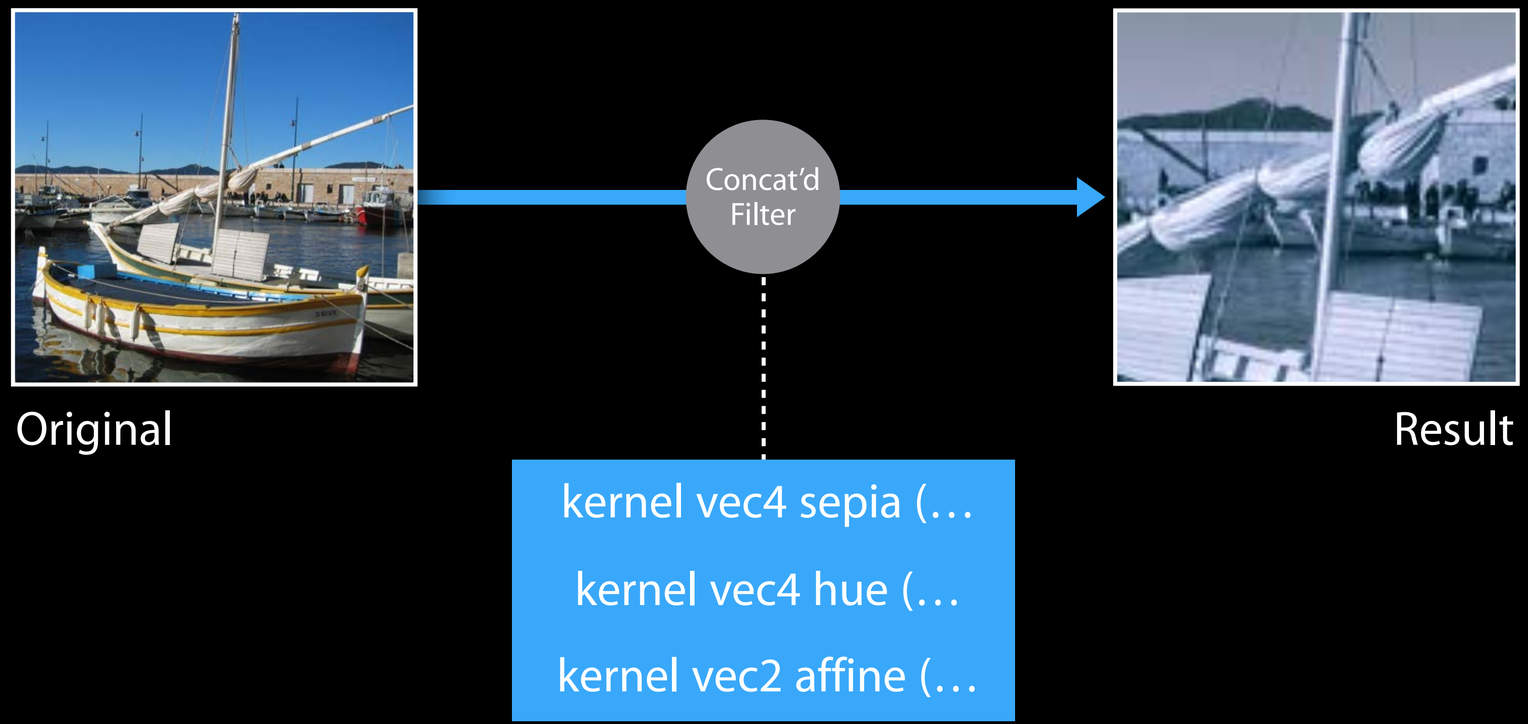
Key Concepts

Each filter has one or more kernel functions



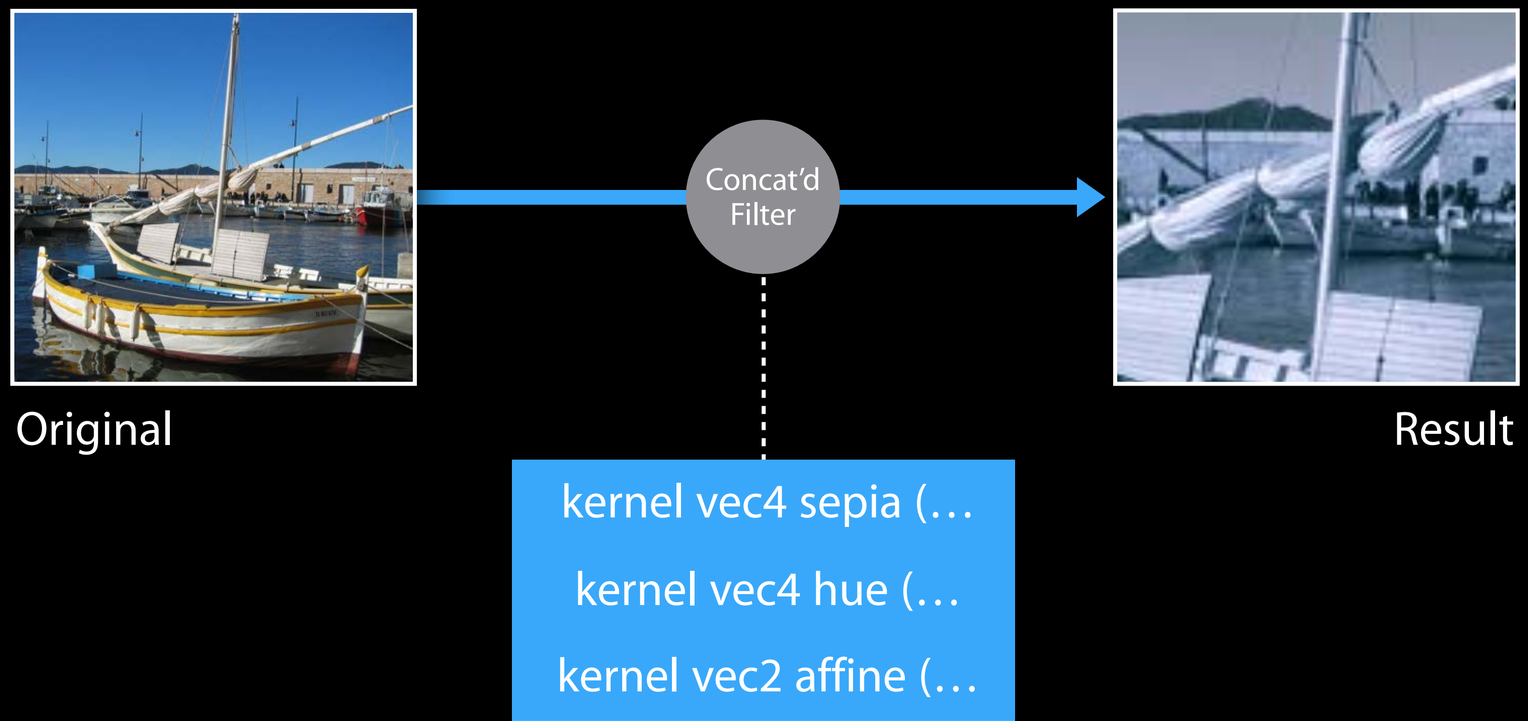
Key Concepts

Kernels concatenated into programs to minimize buffers



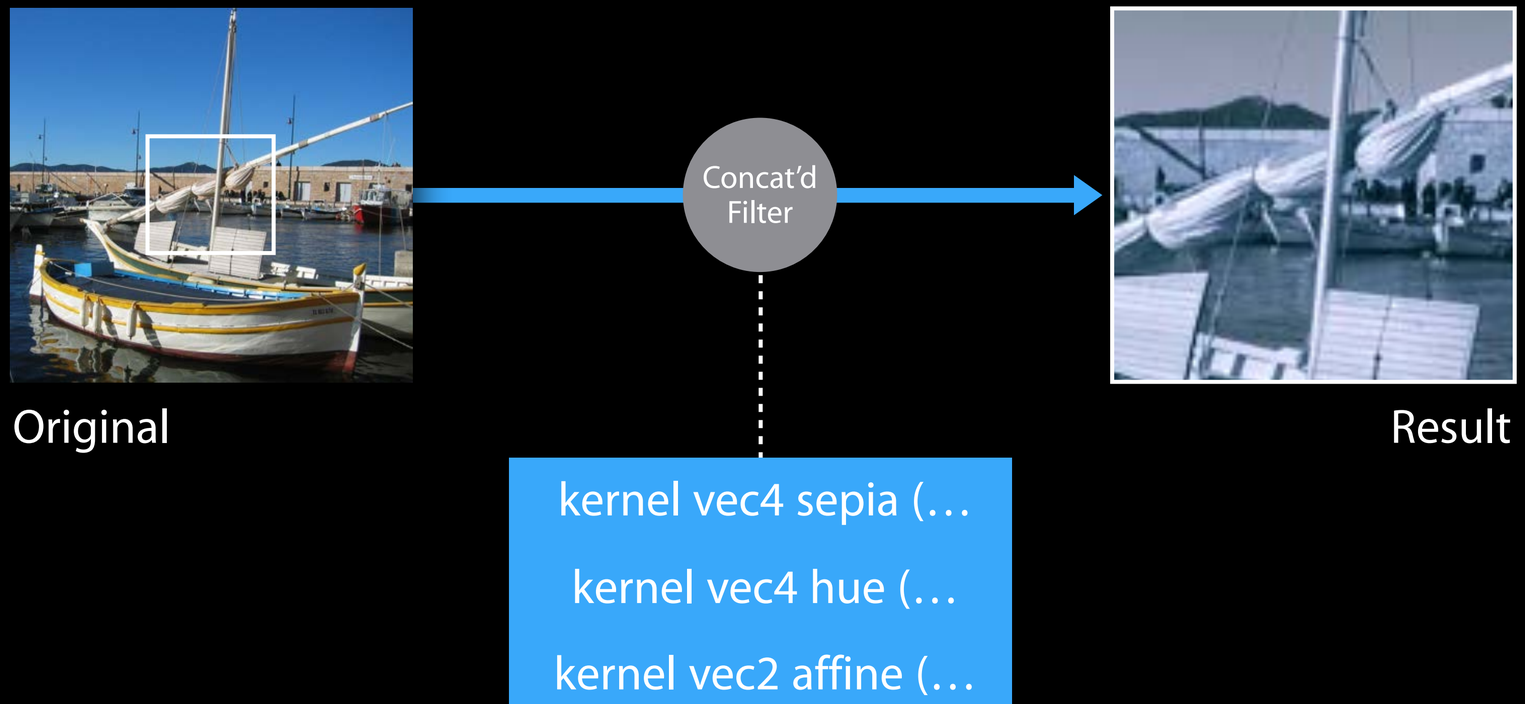
Key Concepts

Region of Interest functions enable large image renders



Key Concepts

Region of Interest functions enable large image renders



Core Image Classes

CIKernel

- Represents a program written in Core Image's language

CIFilter

- Has mutable input parameters
- Uses one or more CIKernels to create a new image based on inputs

CIImage

- An immutable object that represents the recipe for an image

CIContext

- An object through which Core Image draws results

What's New in Core Image

What's New in Core Image

Metal

Filters

Detectors

Color management

Kernel class and language

What's New in Core Image

NEW

Metal

Filters

Detectors

Color management

Kernel class and language

Unified implementation

Metal Integration

NEW

Metal Textures can be an input to Core Image

Metal Textures can be the output of Core Image

Core Image can use Metal to render filters

Some CIFilters use Metal Performance Shaders

Same Built-in Filters Across Platforms

Same Built-in Filters Across Platforms

200 built-in filters on both platforms

Same Built-in Filters Across Platforms

NEW

200 built-in filters on both platforms

40 new filters for iOS Core Image

- Comic Effect, CMYK Halftone, Droste, Page Curl
- Median, Edges, Noise Reduction
- Reduction filters such as Area Maximum, Column Average

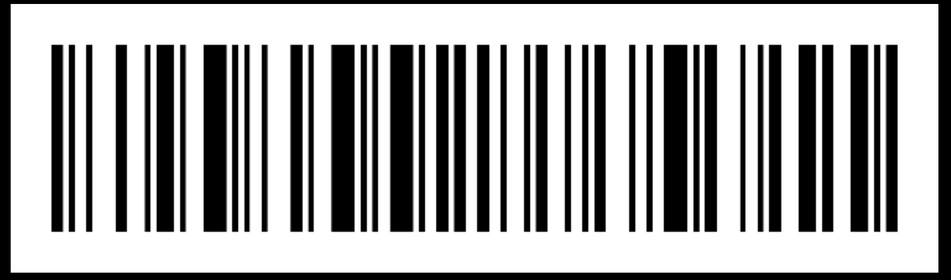

9:41 AM
100%
CIFunHouse
Messages
Safari
Mail
Music


9:41 AM
100%
CIFunHouse
Messages
Safari
Mail
Music

Two New Built-in CIFilters

PDF417 and Code128 barcode generators

NEW



New CIDetector

CIFaceDetector

CIBarcodeDetector

CIRectangleDetector

New CIDetector

NEW

CIFaceDetector

CIBarcodeDetector

CIRectangleDetector

CITextDetector


9:41 AM
100%
CIFunHouse
Messages
Safari
Mail
Music


9:41 AM
100%
CIFunHouse
Messages
Safari
Mail
Music

Full Color Management

Now supported on iOS

Supports ICC-based CGColorSpaceRef for input or output

Correct Rendering of TIFFs and JPGs tagged with colorspace


iPad
9:41 AM
100%
Rothaus
Rothaus


iPad
9:41 AM
100%
Rothaus
Rothaus

New CIKernel Classes

CIColorKernels and CIWarpKernels now on OS X

CIColorKernel and CIWarpKernel subclasses makes it easier to write common filters

```
kernel vec4 blendWithMask( sampler fore, sampler back, sampler mask)
{
    vec4 f = sample (fore, samplerCoord (fore));
    vec4 b = sample (back, samplerCoord (back));
    vec4 m = sample (mask, samplerCoord (mask));
    return mix (b, f, m.g);
}
```

New CIKernel Classes

CIColorKernels and CIWarpKernels now on OS X

CIColorKernel and CIWarpKernel subclasses makes it easier to write common filters

```
kernel vec4 blendWithMask( sampler fore, sampler back, sampler mask)
{
    vec4 f = sample (fore, samplerCoord (fore));
    vec4 b = sample (back, samplerCoord (back));
    vec4 m = sample (mask, samplerCoord (mask));
    return mix (b, f, m.g);
}
```

```
kernel vec4 blendWithMask(__sample fore, __sample back, __sample mask)
{
    return mix (back, fore, mask.g);
}
```

Improved CIKernel Language

Richer language features on OS X

Based on LLVM technologies

New language features (if, for, while)

CIKernels in existing apps should work

Stricter compiler errors if your app is linked with OS X El Capitan

Improved CIKernel Language

Richer language features on OS X

```
kernel vec4 motionBlur (sampler image, vec2 dir, float count)
{
    vec2 dc = destCoord();
    vec4 result = vec4(0.0);

    for (float i=0.0; i < count; i++)
        result += sample(image, samplerTransform (image, dc + dir * i));

    return result / count;
}
```

Improved CIKernel Language

Richer language features on OS X

```
kernel vec4 motionBlur (sampler image, vec2 dir, float count)
{
    vec2 dc = destCoord();
    vec4 result = vec4(0.0); float div = 0.0;

    for (float i=0.0; i < count; i++) {
        vec4 s = sample(image, samplerTransform (image, dc + dir * i));
        if (s.a < 1.0) break;
        result += s; div += 1.0;
    }
    return result / div;
}
```

The Goal of the CIKernel Language

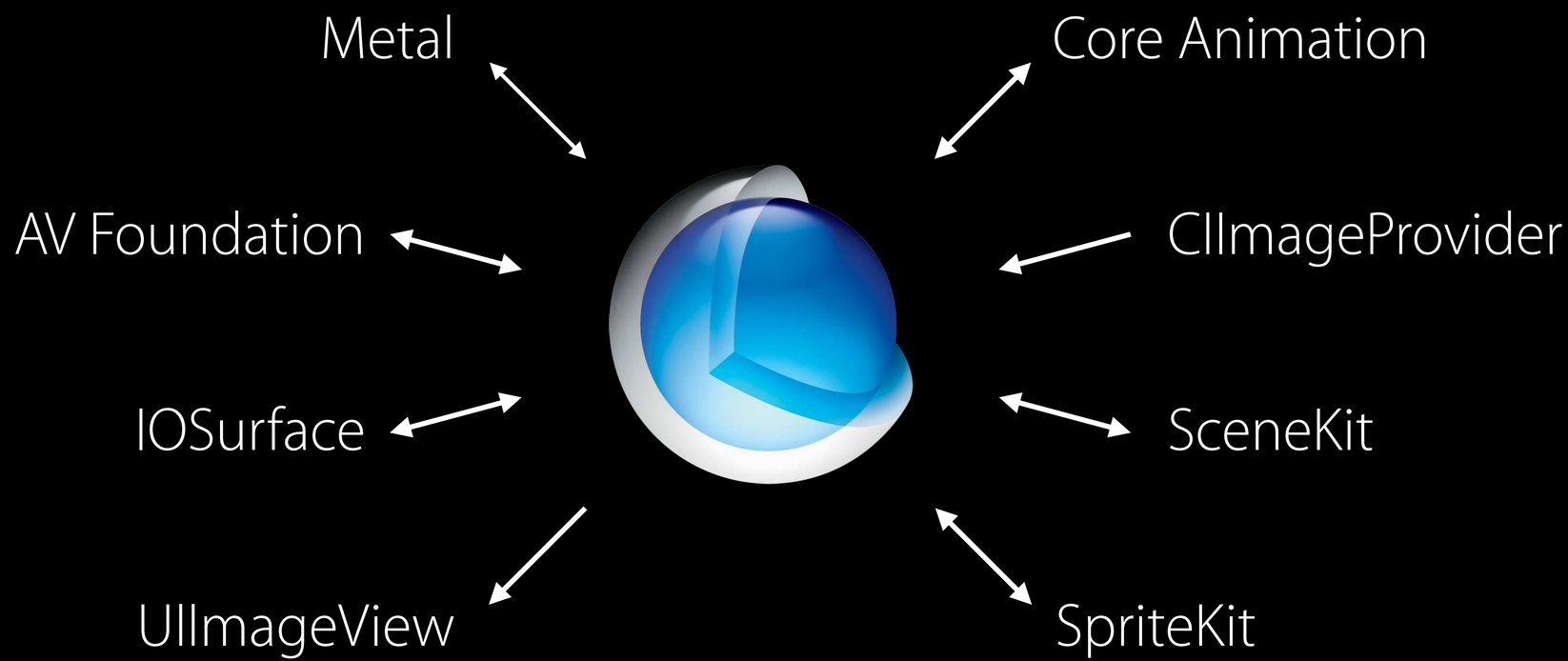
Write kernels once and run everywhere regardless of:

- System: iOS or OS X
- Size: destCoord() and samplerTransform() enable automatic tiling
- Renderer: Metal, OpenCL, OpenGL, OpenGL ES

Bridging Core Image with Other Frameworks

Interoperability

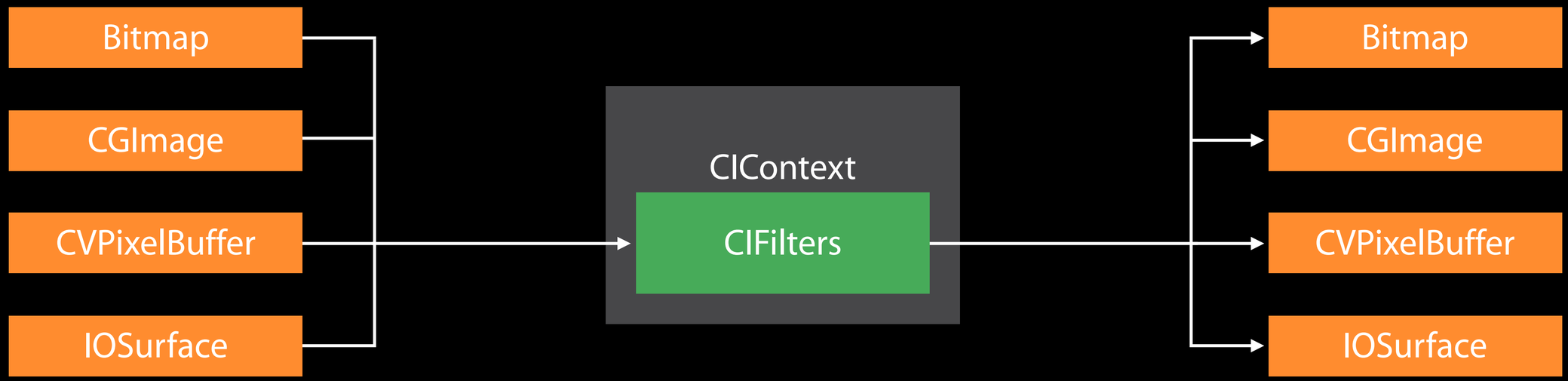
Interoperability



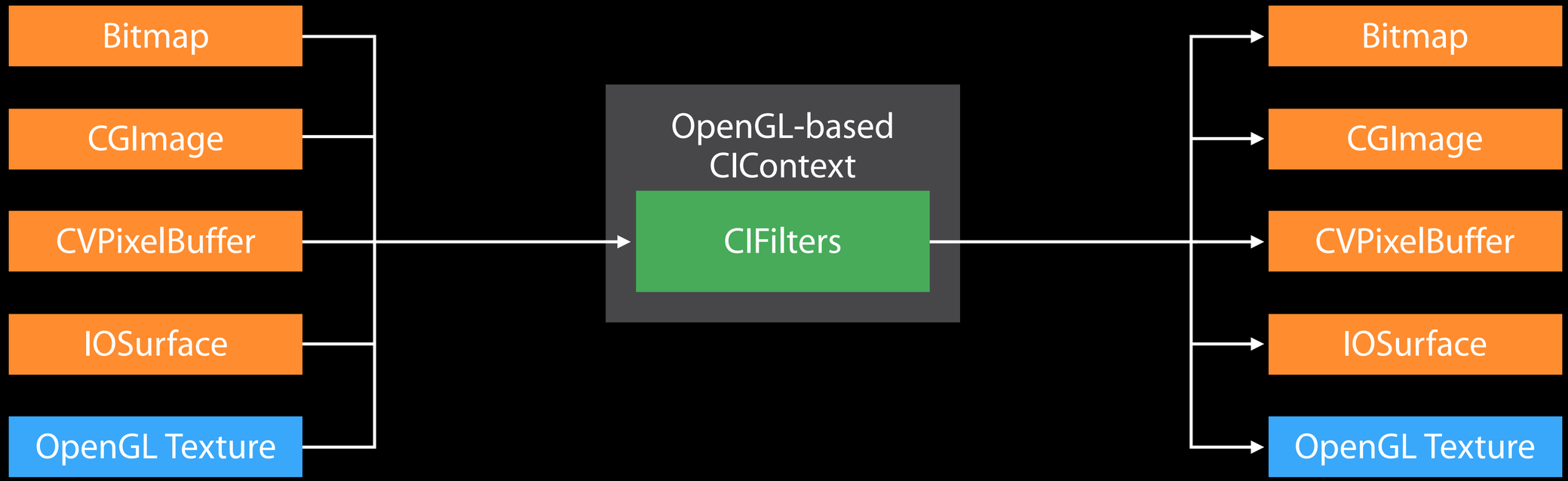
Core Image and Metal

Tony Chu Engineer

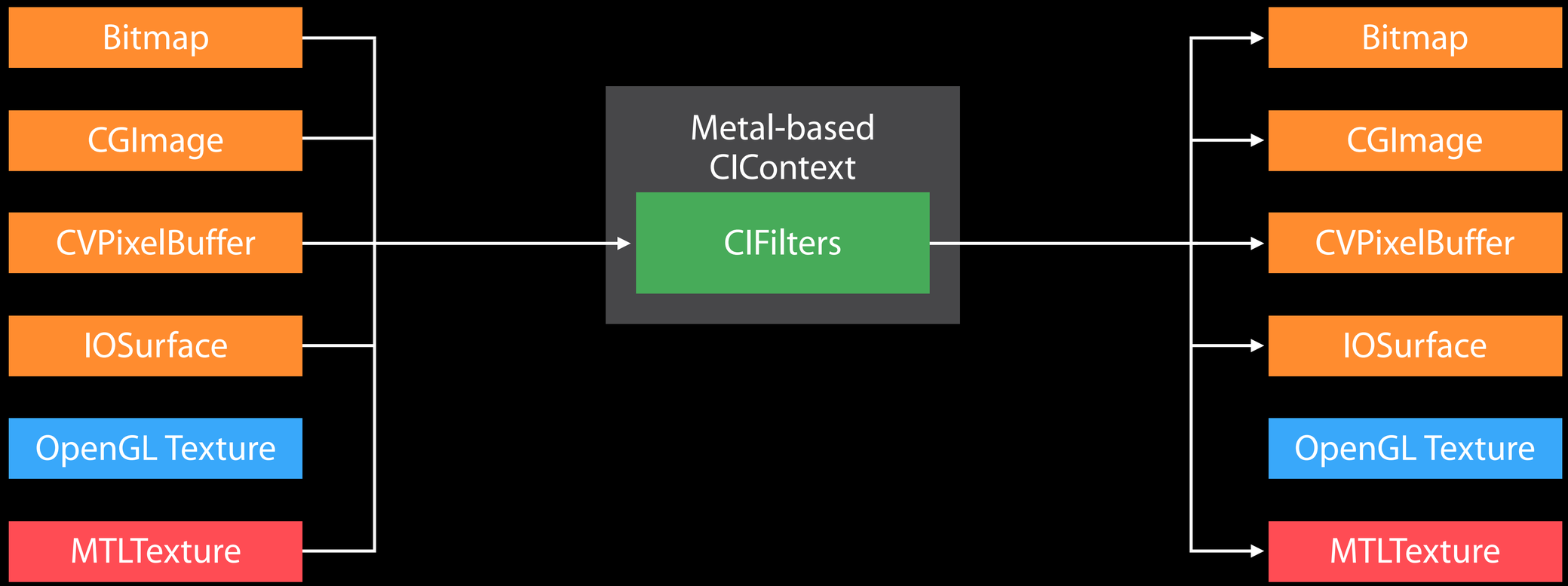
Using Metal with Core Image



Using Metal with Core Image



Using Metal with Core Image



CIImage Metal API

For input textures

NEW

```
init(MTLTexture texture: MTLTexture,
     options: [String: AnyObject]?)
```

CIContext Metal API

NEW

```
init(MTLDevice device: MTLDevice)

init(MTLDevice device: MTLDevice,
     options: [String: AnyObject]?)
```

CIContext Metal API

For output textures

NEW

```
func render(image: CIImage,
            toMTLTexture texture: MTLTexture,
            commandBuffer cb: MTLCommandBuffer?,
            bounds r: CGRect,
            colorSpace cs: CGColorSpace)
```

CIContext Metal API

For output textures

NEW

```
func render(image: CIImage,
            toMTLTexture texture: MTLTexture,
            commandBuffer cb: MTLCommandBuffer?,
            bounds r: CGRect,
            colorSpace cs: CGColorSpace)
```

CIContext Metal API

For output textures

NEW

```
func render(image: CIImage,
            toMTLTexture texture: MTLTexture,
            commandBuffer cb: MTLCommandBuffer?,
            bounds r: CGRect,
            colorSpace cs: CGColorSpace)
```

If commandBuffer is nil, Core Image will:

- Create one internally
- Encode commands to it
- Commit it before returning

CIContext Metal API

For output textures

NEW

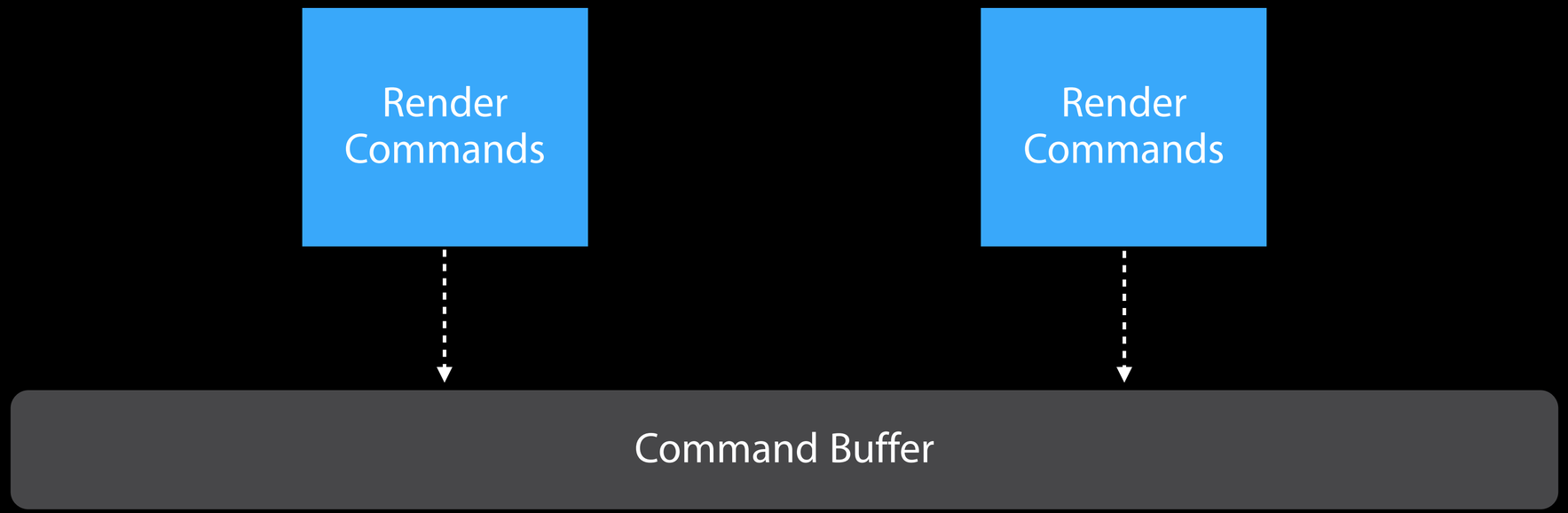
```
func render(image: CIImage,
            toMTLTexture texture: MTLTexture,
            commandBuffer cb: MTLCommandBuffer?,
            bounds r: CGRect,
            colorSpace cs: CGColorSpace)
```

If commandBuffer is provided, Core Image will:

- Encode commands to it
- Return without committing

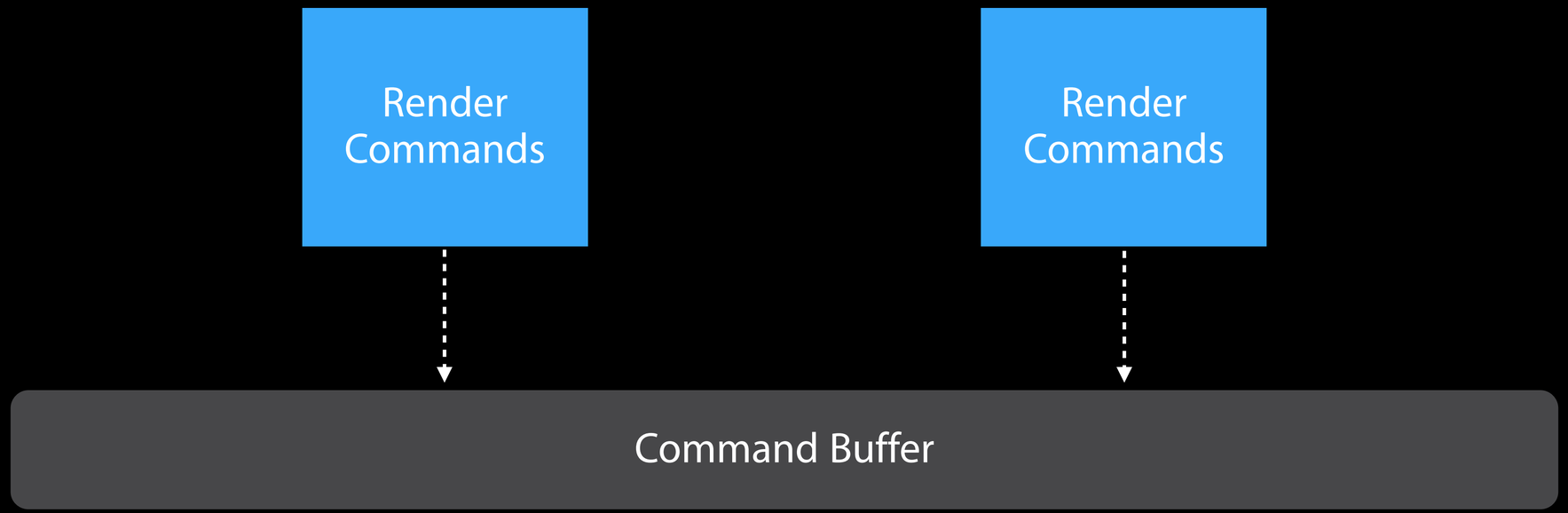
Seamless Integration with Metal

Encoding Metal commands



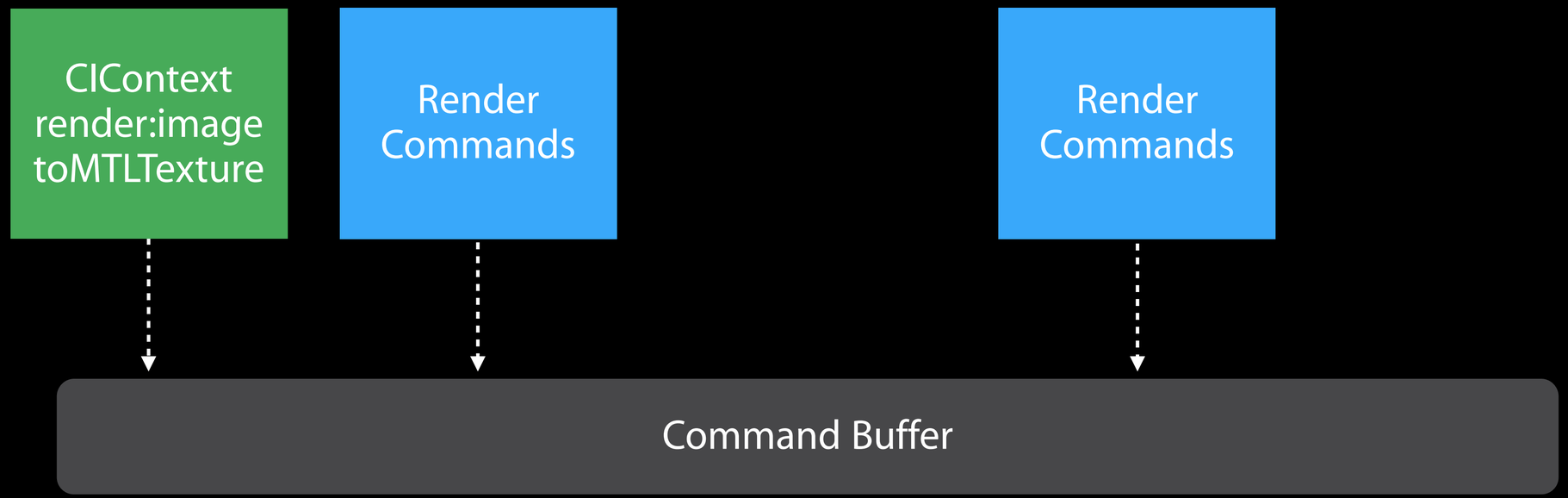
Seamless Integration with Metal

Core Image filters can be inserted anywhere



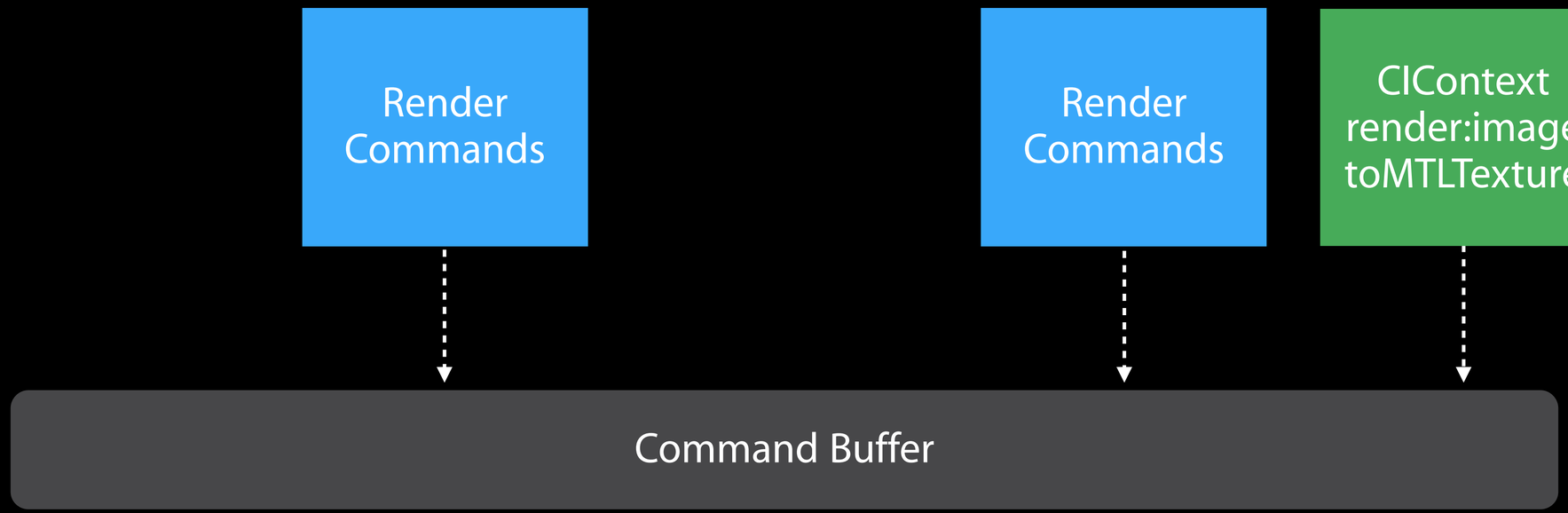
Seamless Integration with Metal

Core Image filters can be inserted anywhere



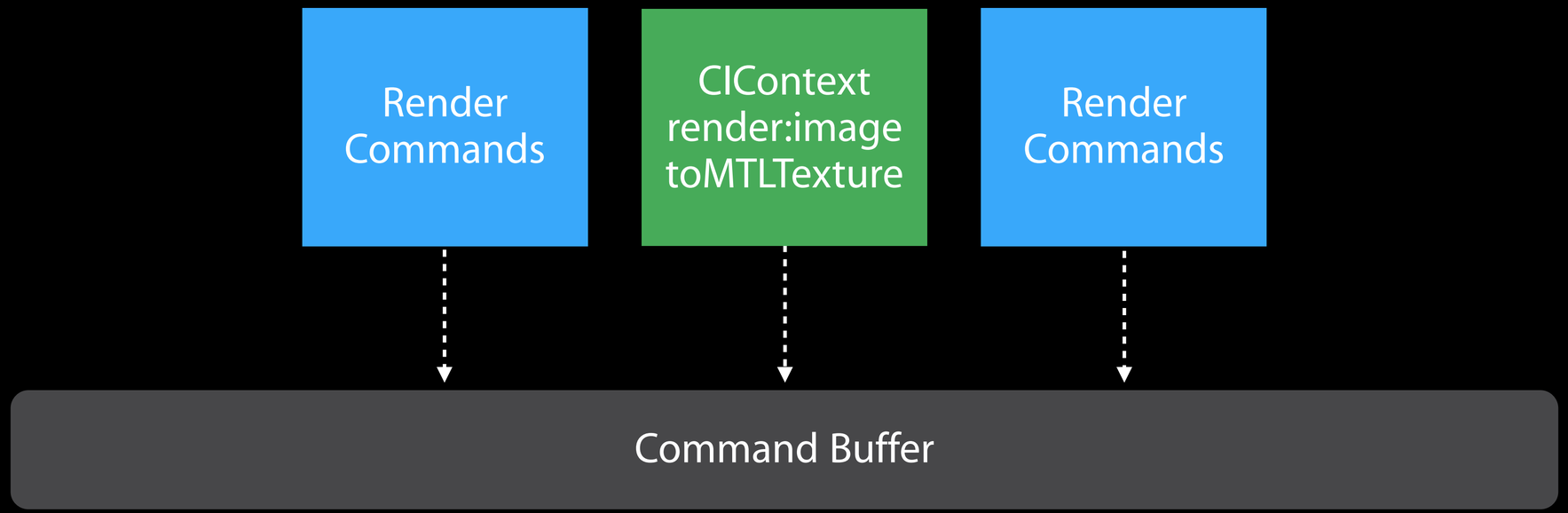
Seamless Integration with Metal

Core Image filters can be inserted anywhere



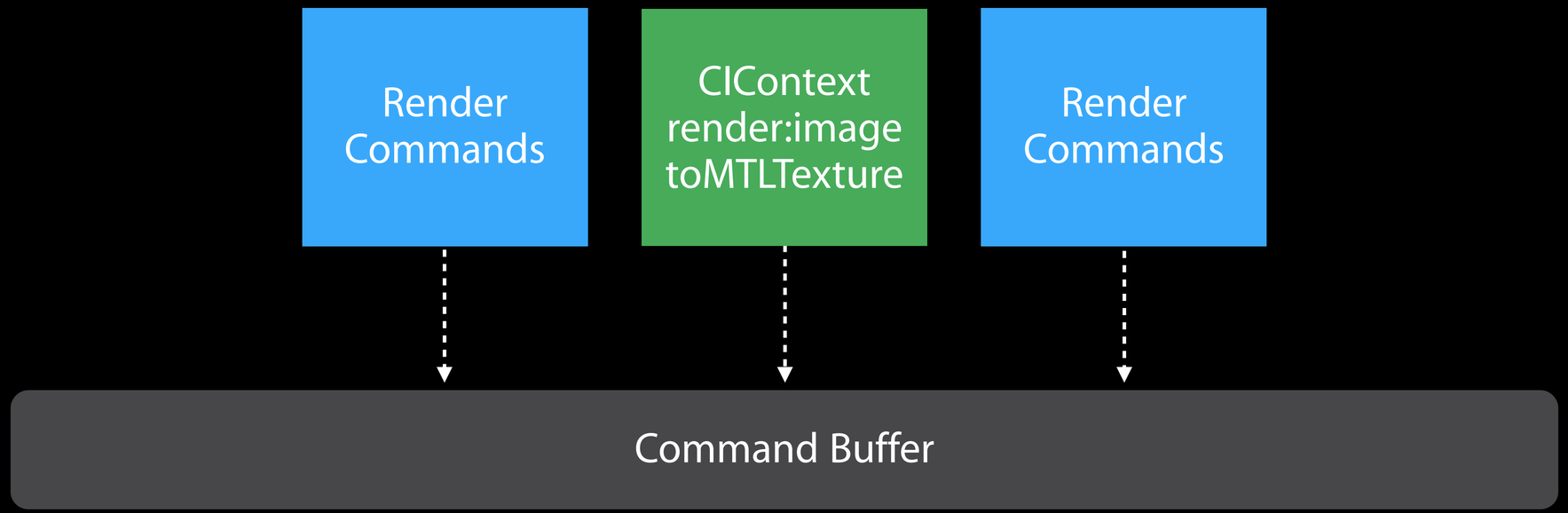
Seamless Integration with Metal

Core Image filters can be inserted anywhere



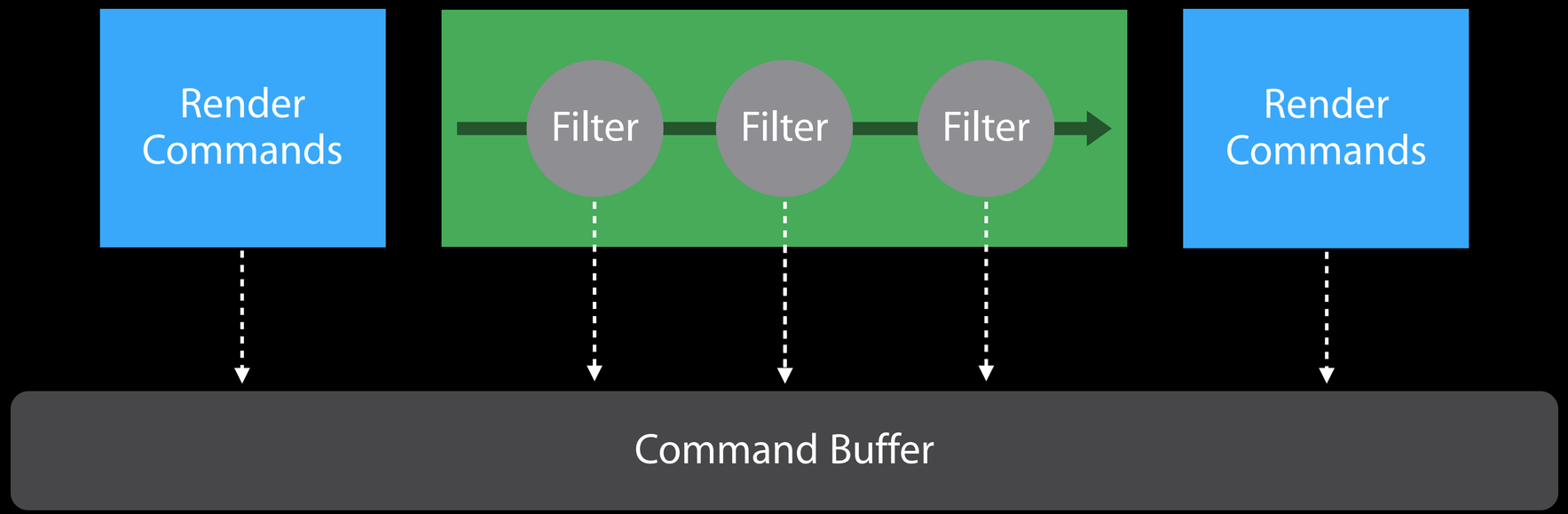
Seamless Integration with Metal

Core Image filters can be inserted anywhere



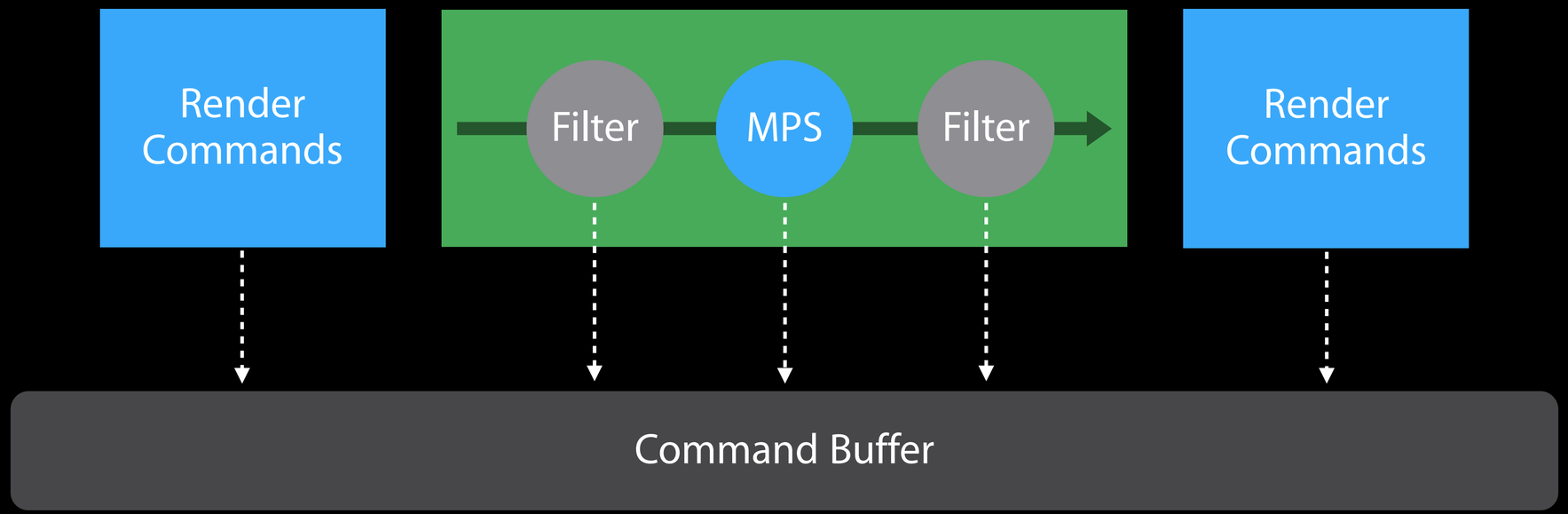
Seamless Integration with Metal

Core Image will encode commands for each CIFilter



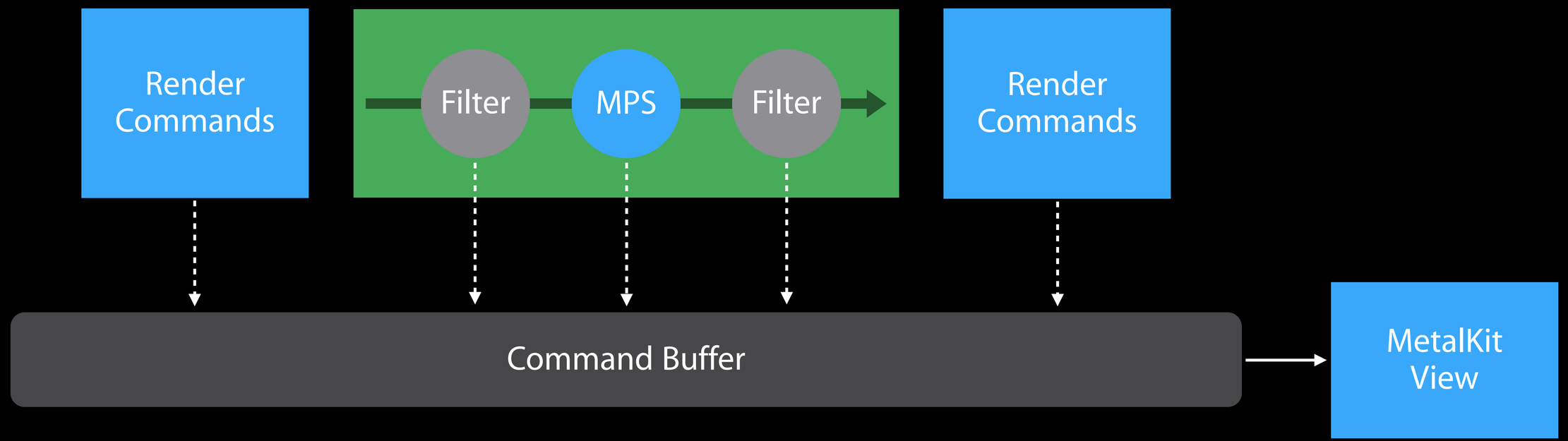
Seamless Integration with Metal

Some built-in filters use Metal Performance Shaders



Seamless Integration with Metal

Drawing final Metal Texture to a MTKView



Rendering Core Image to a MetalKit View

Setting up the view

```
override func viewDidLoad()
{
    super.viewDidLoad()

    // setup view properties
    let view = self.view as! MTKView
    view.delegate = self
    view.framebufferOnly = false

    context = CIContext(MTLDevice: device)
}
```

Rendering Core Image to a MetalKit View

Setting up the view

```
override func viewDidLoad()
{
    super.viewDidLoad()

    // setup view properties
    let view = self.view as! MTKView
    view.delegate = self
    view.framebufferOnly = false

    context = CIContext(MTLDevice: device)
}
```

Rendering Core Image to a MetalKit View

Setting up the view

```
override func viewDidLoad()
{
    super.viewDidLoad()

    // setup view properties
    let view = self.view as! MTKView
    view.delegate = self
    view.framebufferOnly = false

    context = CIContext(MTLDevice: device)
}
```

Rendering Core Image to a MetalKit View

Drawing into the view

```
func drawInView(view: MTKView)
{
    let commandBuffer = commandQueue.commandBuffer()
    var image = CIImage(MTLTexture: srcTexture!, options: nil)
    image = image.imageByApplyingFilter(“CIGaussianBlur”,
        withInputParameters: [kCIInputRadiusKey: 50])
    let outputTexture = view.currentDrawable?.texture
    context.render(image, toMTLTexture: outputTexture!,
        commandBuffer: commandBuffer, bounds: image.extent, colorSpace: cs)

    commandBuffer.presentDrawable(view.currentDrawable!)
    commandBuffer.commit()
}
```

Rendering Core Image to a MetalKit View

Drawing into the view

```
func drawInView(view: MTKView)
{
    let commandBuffer = commandQueue.commandBuffer()
    var image = CIImage(MTLTexture: srcTexture!, options: nil)
    image = image.imageByApplyingFilter(“CIGaussianBlur”,
        withInputParameters: [kCIInputRadiusKey: 50])
    let outputTexture = view.currentDrawable?.texture
    context.render(image, toMTLTexture: outputTexture!,
        commandBuffer: commandBuffer, bounds: image.extent, colorSpace: cs)

    commandBuffer.presentDrawable(view.currentDrawable!)
    commandBuffer.commit()
}
```

Rendering Core Image to a MetalKit View

Drawing into the view

```
func drawInView(view: MTKView)
{
    let commandBuffer = commandQueue.commandBuffer()
    var image = CIImage(MTLTexture: srcTexture!, options: nil)
    image = image.imageByApplyingFilter(“CIGaussianBlur”,
        withInputParameters: [kCIInputRadiusKey: 50])
    let outputTexture = view.currentDrawable?.texture
    context.render(image, toMTLTexture: outputTexture!,
        commandBuffer: commandBuffer, bounds: image.extent, colorSpace: cs)

    commandBuffer.presentDrawable(view.currentDrawable!)
    commandBuffer.commit()
}
```

Rendering Core Image to a MetalKit View

Drawing into the view

```
func drawInView(view: MTKView)
{
    let commandBuffer = commandQueue.commandBuffer()
    var image = CIImage(MTLTexture: srcTexture!, options: nil)
    image = image.imageByApplyingFilter(“CIGaussianBlur”,
        withInputParameters: [kCIInputRadiusKey: 50])
    let outputTexture = view.currentDrawable?.texture
    context.render(image, toMTLTexture: outputTexture!,
        commandBuffer: commandBuffer, bounds: image.extent, colorSpace: cs)

    commandBuffer.presentDrawable(view.currentDrawable!)
    commandBuffer.commit()
}
```

Rendering Core Image to a MetalKit View

Drawing into the view

```
func drawInView(view: MTKView)
{
    let commandBuffer = commandQueue.commandBuffer()
    var image = CIImage(MTLTexture: srcTexture!, options: nil)
    image = image.imageByApplyingFilter(“CIGaussianBlur”,
        withInputParameters: [kCIInputRadiusKey: 50])
    let outputTexture = view.currentDrawable?.texture
    context.render(image, toMTLTexture: outputTexture!,
        commandBuffer: commandBuffer, bounds: image.extent, colorSpace: cs)

    commandBuffer.presentDrawable(view.currentDrawable!)
    commandBuffer.commit()
}
```

Core Image and AV Foundation

AV Foundation

NEW

Now it is easy to use Core Image within your AV Foundation app

AV Foundation

NEW

Now it is easy to use Core Image within your AV Foundation app

- Core Image integrated with AVVideoComposition class

AV Foundation

NEW

Now it is easy to use Core Image within your AV Foundation app

- Core Image integrated with AVVideoComposition class
- Automatic color management
  - Can be disabled

AV Foundation

NEW

Now it is easy to use Core Image within your AV Foundation app

- Core Image integrated with AVVideoComposition class
- Automatic color management
  - Can be disabled
- Examples:
  - Exporting an AVAsset applying CIFilters
  - Playback an AVAsset applying CIFilters

Exporting an AVAsset Applying CIFilters

The custom CIFilter: Image

Exporting an AVAsset Applying CIFilters

The custom CIFilter: Image + Sepia

Exporting an AVAsset Applying CIFilters

The custom CIFilter: Image + Sepia + Noise

Exporting an AVAsset Applying CIFilters

The custom CIFilter: Image + Sepia + Noise + Scratches

Exporting an AVAsset Applying CIFilters

The custom CIFilter

```
class OldeFilm : CIFilter
{
    var inputImage: CIImage?
    var inputTime: NSNumber?
}
```

Exporting an AVAsset Applying CIFilters

Creating a filtered composition

```
let vidComp = AVVideoComposition(asset: avAsset,
    applyingCIFiltersWithHandler: {
        request in

        let seconds = CMTimeGetSeconds(request.compositionTime)

        let filtered = request.sourceImage.imageByApplyingFilter("OldeFilm",
               withInputParameters: [kCIInputTimeKey: seconds])

        request.finishWithImage(filtered, context: nil)
    })
```

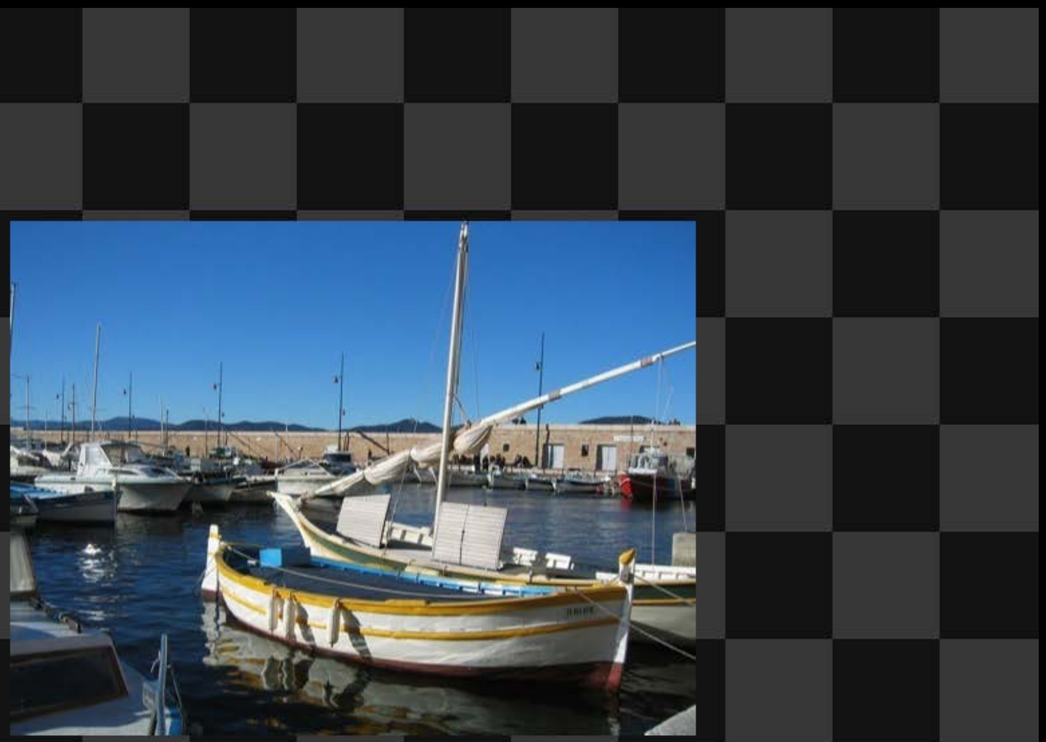
Exporting an AVAsset Applying CIFilters

Creating a filtered composition without color management

```
let cicontext = CIContext(options: [kCIContextWorkingColorSpace: NSNull()])
let vidComp = AVVideoComposition(asset: avAsset,
    applyingCIFiltersWithHandler: {
        request in

        let seconds = CMTimeGetSeconds(request.compositionTime)

        let filtered = request.sourceImage.imageByApplyingFilter("OldeFilm",
               withInputParameters: [kCIInputTimeKey: seconds])

        request.finishWithImage(filtered, context: cicontext)
    })
```

Exporting an AVAsset Applying CIFilters

Convolution filters with unclamped edges



Exporting an AVAsset Applying CIFilters

Convolution filters with unclamped edges

Exporting an AVAsset Applying CIFilters

Convolution filters with clamped edges

```
let vidComp = AVVideoComposition(asset: avAsset,
    applyingCIFiltersWithHandler: {
        request in
        filtered = request.sourceImage.imageByClampingToExtent();

        filtered = filtered.imageByApplyingFilter("CIGaussianBlur",
                    withInputParameters: [kCIInputRadiusKey: 100])

        filtered = filtered.imageByCroppingToRect(request.sourceImage.extent)
        request.finishWithImage(filtered, context: cicontext)
    })
```

Exporting an AVAsset Applying CIFilters

Convolution filters with clamped edges

```
let vidComp = AVVideoComposition(asset: avAsset,
    applyingCIFiltersWithHandler: {
        request in
        filtered = request.sourceImage.imageByClampingToExtent();
        filtered = filtered.imageByApplyingFilter("CIGaussianBlur",
                    withInputParameters: [kCIInputRadiusKey: 100])

        filtered = filtered.imageByCroppingToRect(request.sourceImage.extent)
        request.finishWithImage(filtered, context: cicontext)
    })
```

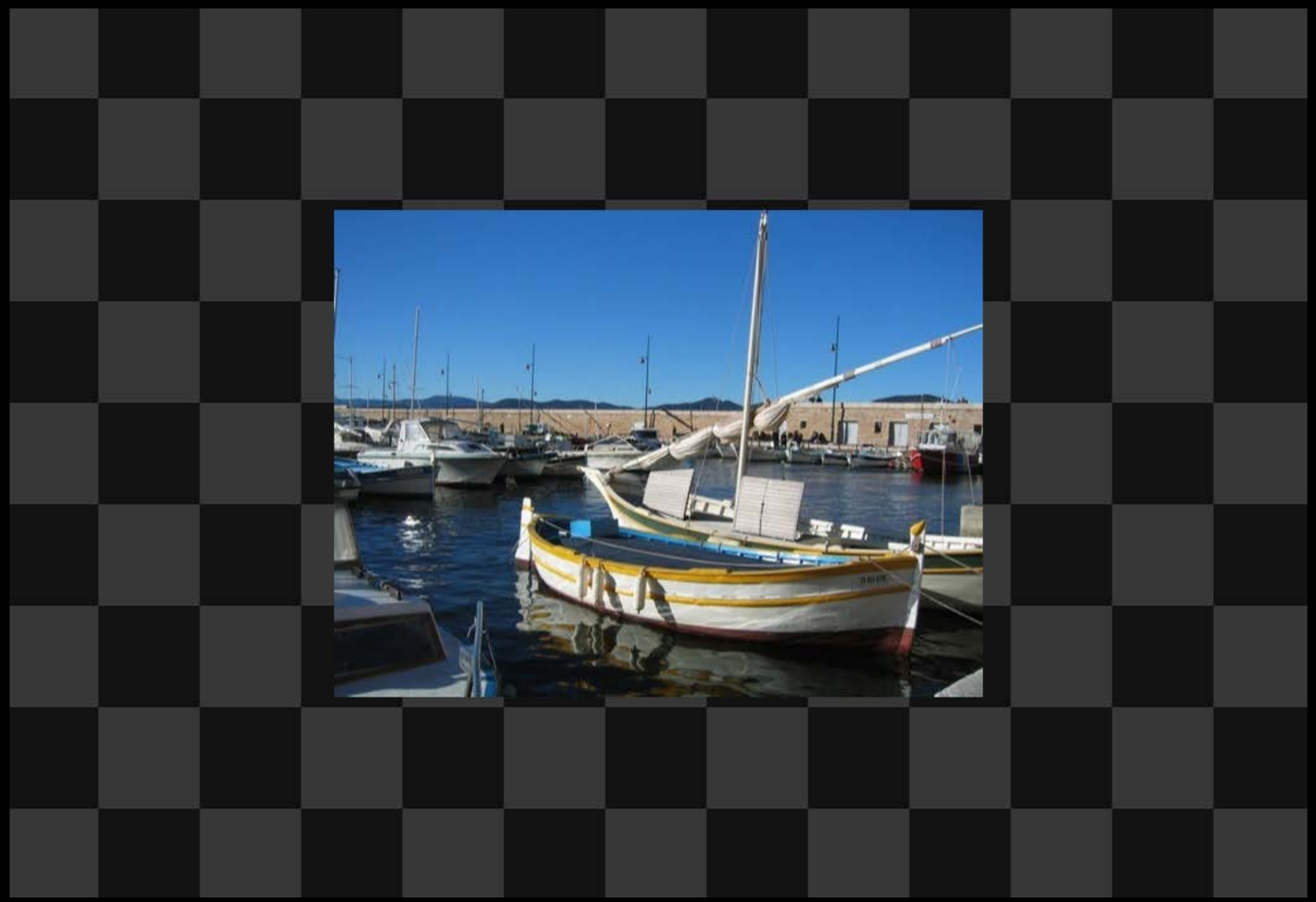
Exporting an AVAsset Applying CIFilters

Convolution filters with clamped edges

```
let vidComp = AVVideoComposition(asset: avAsset,
    applyingCIFiltersWithHandler: {
        request in
        filtered = request.sourceImage.imageByClampingToExtent();

        filtered = filtered.imageByApplyingFilter("CIGaussianBlur",
                    withInputParameters: [kCIInputRadiusKey: 100])

        filtered = filtered.imageByCroppingToRect(request.sourceImage.extent)
        request.finishWithImage(filtered, context: cicontext)
    })
```

Exporting an AVAsset Applying CIFilters

Convolution filters with clamped edges



Exporting an AVAsset Applying CIFilters

Convolution filters with clamped edges

Exporting an AVAsset Applying CIFilters

Exporting the composition

```
let export = AVAssetExportSession(asset: avAsset,
                presetName: AVAssetExportPreset1920x1080)

export.outputFileType = AVFileTypeQuickTimeMovie
export.outputURL = outURL
export.videoComposition = vidComp

NSFileManager.defaultManager().removeItemAtURL(outURL)

export.exportAsynchronouslyWithCompletionHandler()
```

Playback an AVAsset Applying CIFilters

Creating a filtered composition

```
let vidComp = AVVideoComposition(asset: avAsset,
    applyingCIFiltersWithHandler: {

        // same as earlier example

})
```

Playback an AVAsset Applying CIFilters

Make the AVPlayerItem and the AVPlayer

```
let playerItem = AVPlayerItem(asset: avAsset)

playerItem.videoComposition = vidComp

let player = AVPlayer(playerItem: playerItem)

player.play()
```


9:41 AM
100%
CIFunHouse
CoreImageIn...
Messages
Safari
Mail
Music


9:41 AM
100%
CIFunHouse
CoreImageIn...
Messages
Safari
Mail
Music

Core Image Providers

Alexandre Naaman Lead Engineer

CIImageProvider

Allows input image in a CIFilter graph to be provided by a callback

Callback is not called until the image is rendered

Supports tiling

Handles caching with purgeability for you

CIImageProvider

```
let myProvider = TileProvider()

let ciimg = CIImage(imageProvider: myProvider,
                    size:pixelsWide, pixelsHigh,
                    format: kCIFormatBGRA8,
                    colorSpace: cs,
                    options: [kCIImageProviderTileSize: tileSize])
```

CIImageProvider

```
class TileProvider {

  func provideImageData(data: UnsafeMutablePointer<Void>,
                        bytesPerRow rowbytes: Int,
                        origin x: Int, _ y: Int,
                        size width: Int, _ height: Int,
                        userInfo info: AnyObject?) {

    // your code here

  }
}
```

Core Image and View Classes

View Classes and Core Image

High Level

Easy to use

Low Level

Better performance, more control

UIImageView

GLK View | MTKView

Custom Class CAEGLLayer backed | Custom Class CAMetalLayer backed

Core Image and UIImageView

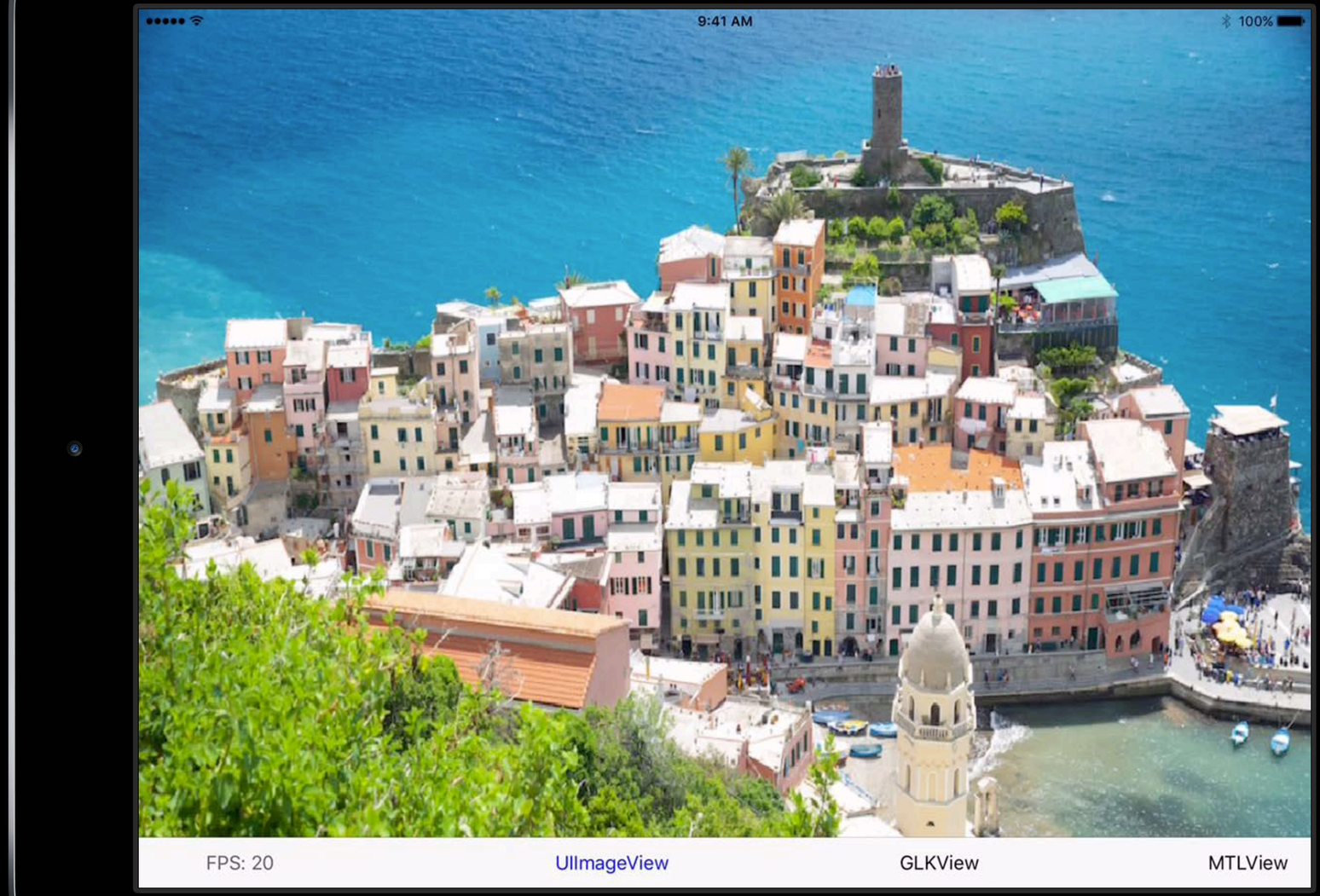
UIImageView

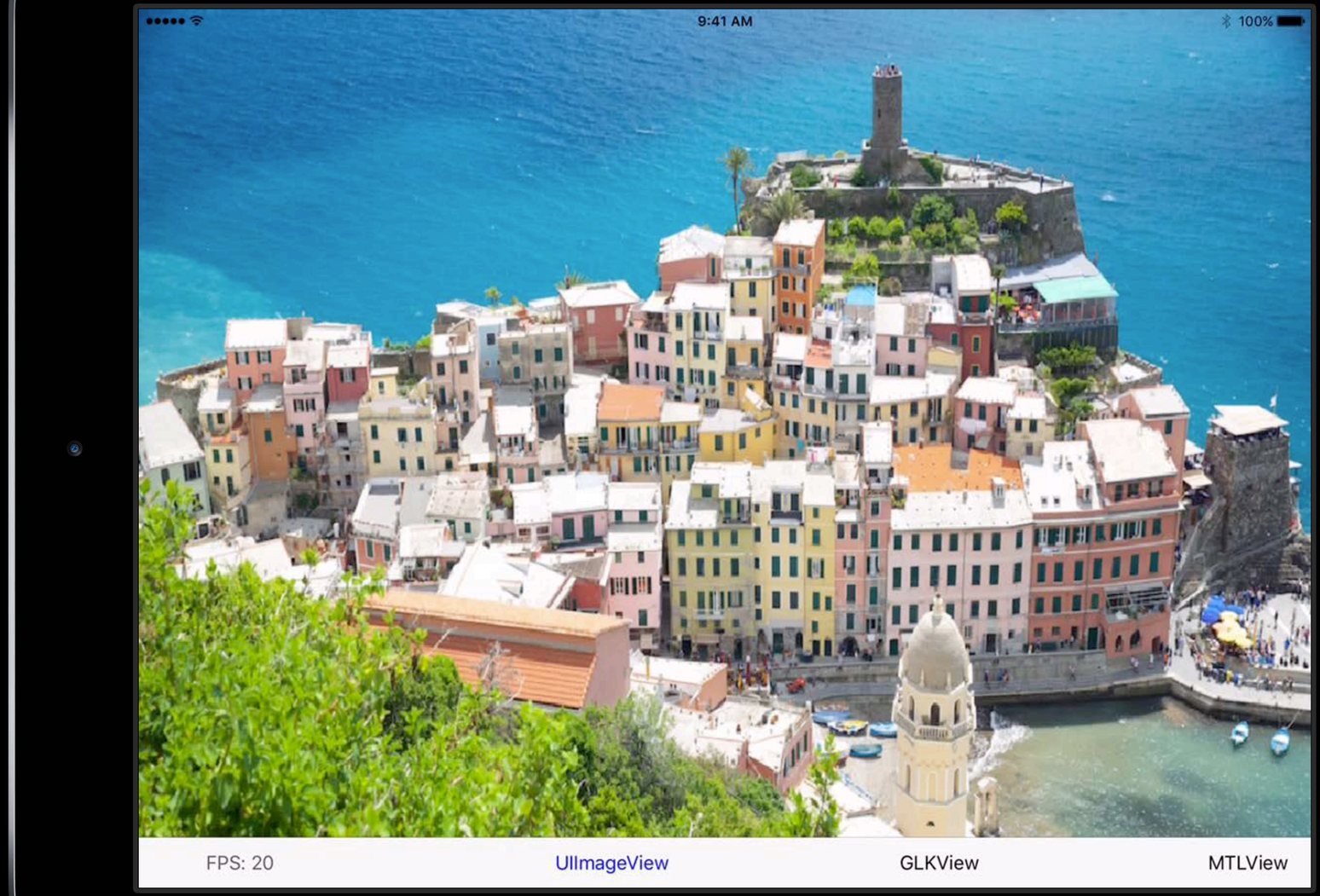
It is very easy to use CIImages with UIImageView

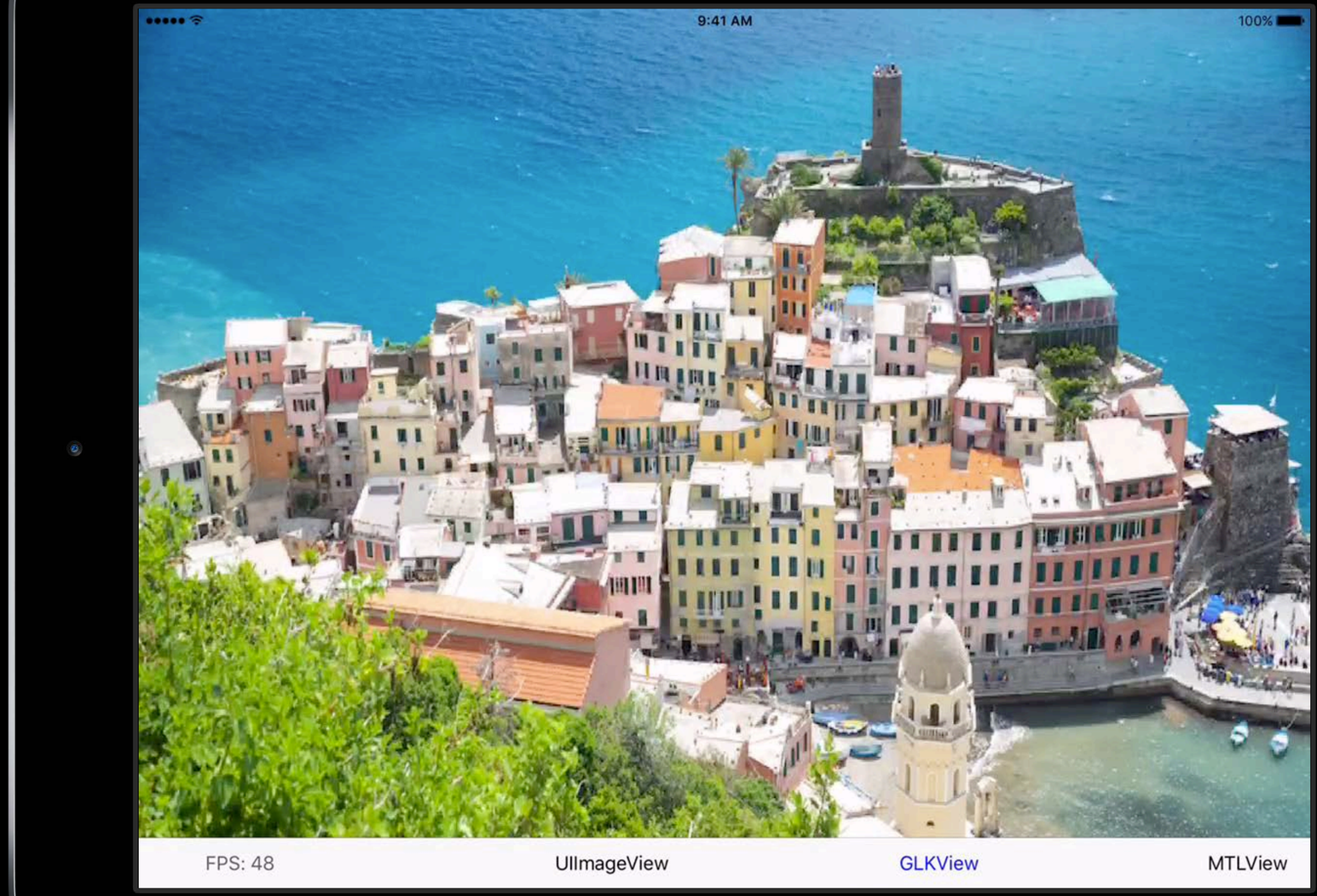
```
imageView.image = UIImage(CIImage: ciimage)
```

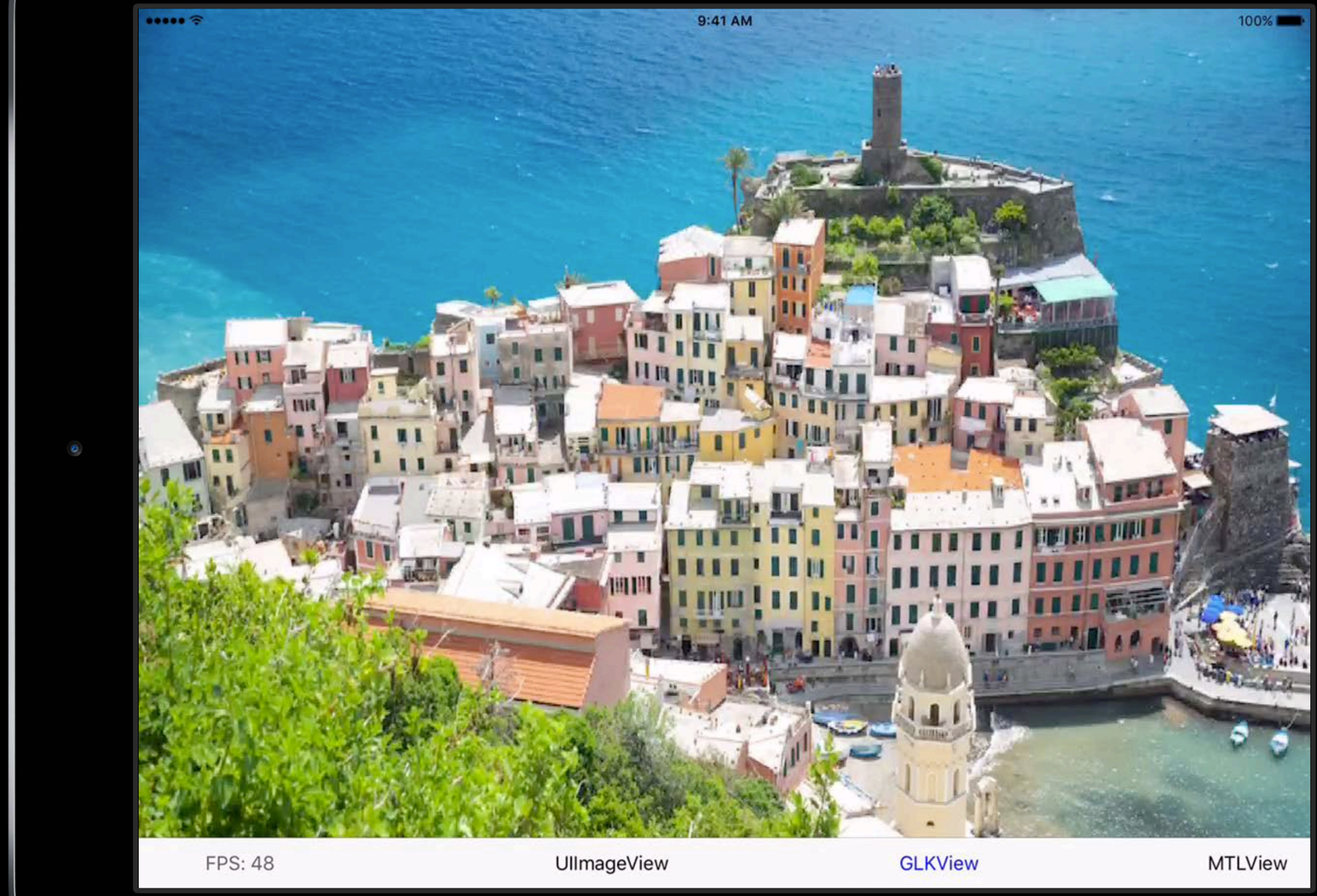
But its performance is not optimal

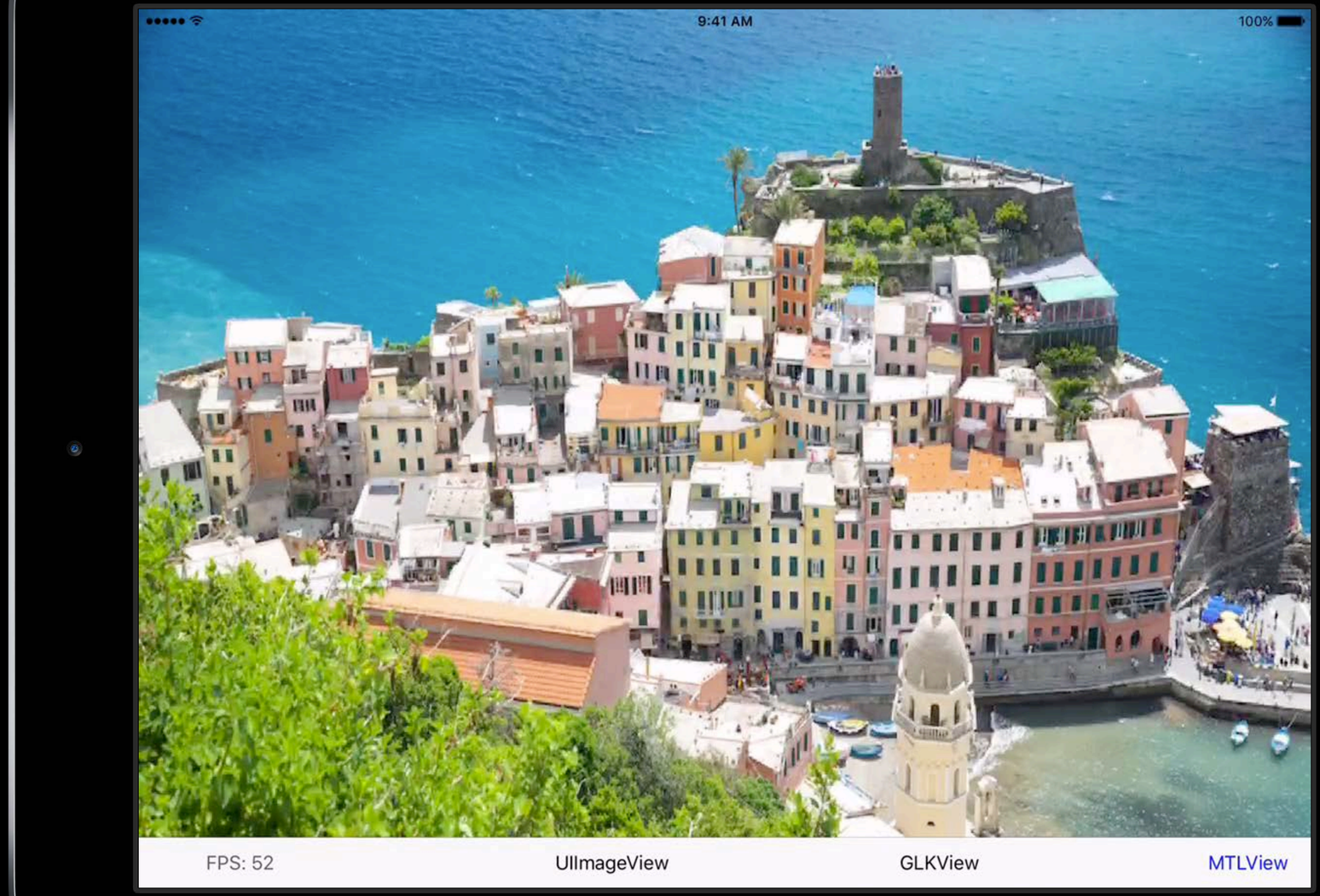
- Because the image is rendered twice

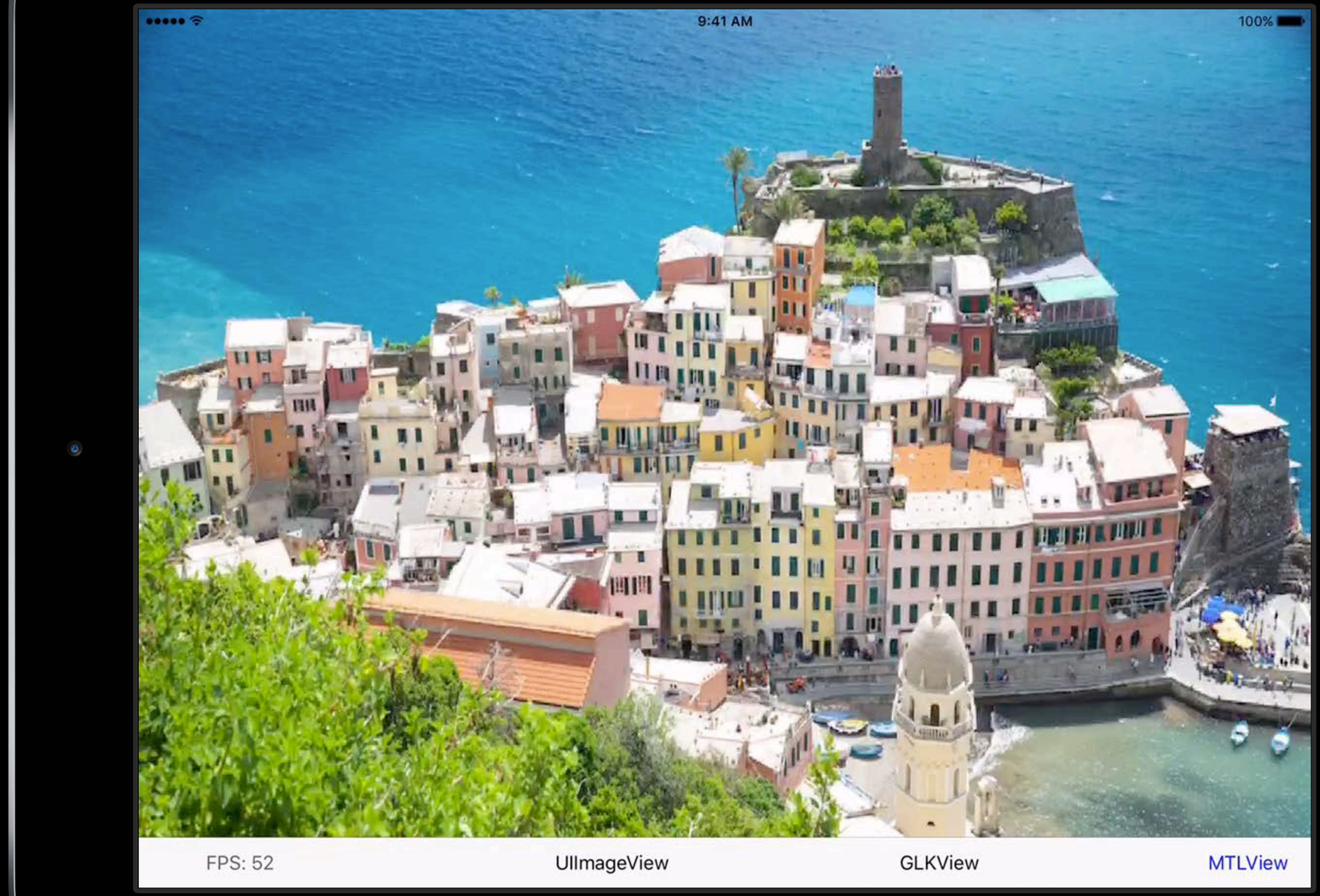

9:41 AM
100%
FPS: 20
UIImageView
GLKView
MTLView


9:41 AM
100%
FPS: 20
UIImageView
GLKView
MTLView


9:41 AM
100%
FPS: 48
UIImageView
GLKView
MTLView


9:41 AM
100%
FPS: 48
UIImageView
GLKView
MTLView


9:41 AM
100%
FPS: 52
UIImageView
GLKView
MTLView


9:41 AM
100%
FPS: 52
UIImageView
GLKView
MTLView

Core Image and Core Animation

CALayer

Can set filter array on a NSView's CALayer

```
let filter = CIFilter(name: ”CIPixellate”,
  withInputParameters:[kCIInputScaleKey: 20.0])

view.layer = CALayer()
view.wantsLayer = true
view.layerUsesCoreImageFilters = true
view.layer?.filters = [filter]
```

CALayer

Can set filter array on a NSView's CALayer

```
let filter = CIFilter(name: "CIPixellate",
  withInputParameters:[kCIInputScaleKey: 20.0])

view.layer = CALayer()
view.wantsLayer = true
view.layerUsesCoreImageFilters = true
view.layer?.filters = [filter]
```

CALayer

On iOS use GLKView or OpenGL ES directly


```
class MyGLKView : GLKView {

}

class MyGLView : UIView {
    override class func layerClass() -> AnyClass { return CAEAGLLayer.self }
}
```

Core Image and IOSurface

IOSurface

IOSurfaceRef advantages:

- Purgeability, locking semantics, efficient moving between devices

Core Image supports many surfaces pixel formats (eg. 420, 444, RGBAh)

On iOS, the benefits of IOSurface can be achieved via CVPixelBuffers

```
var pixelBuffer : UnsafeMutablePointer<Unmanaged<CVPixelBuffer>?>
CVPixelBufferCreate(nil, width: width, height: height,
   pixelFormatType: kCVPixelFormatType_32RGBA,
   pixelFormatAttributes: [kCVPixelBufferIOSurfacePropertiesKey: []],
   pixelBufferOut: pixelBuffer)
```

CVPixelBufferPools use this trick

IOSurface

IOSurfaceRef advantages:

- Purgeability, locking semantics, efficient moving between devices

Core Image supports many surfaces pixel formats (eg. 420, 444, RGBAh)

On iOS, the benefits of IOSurface can be achieved via CVPixelBuffers

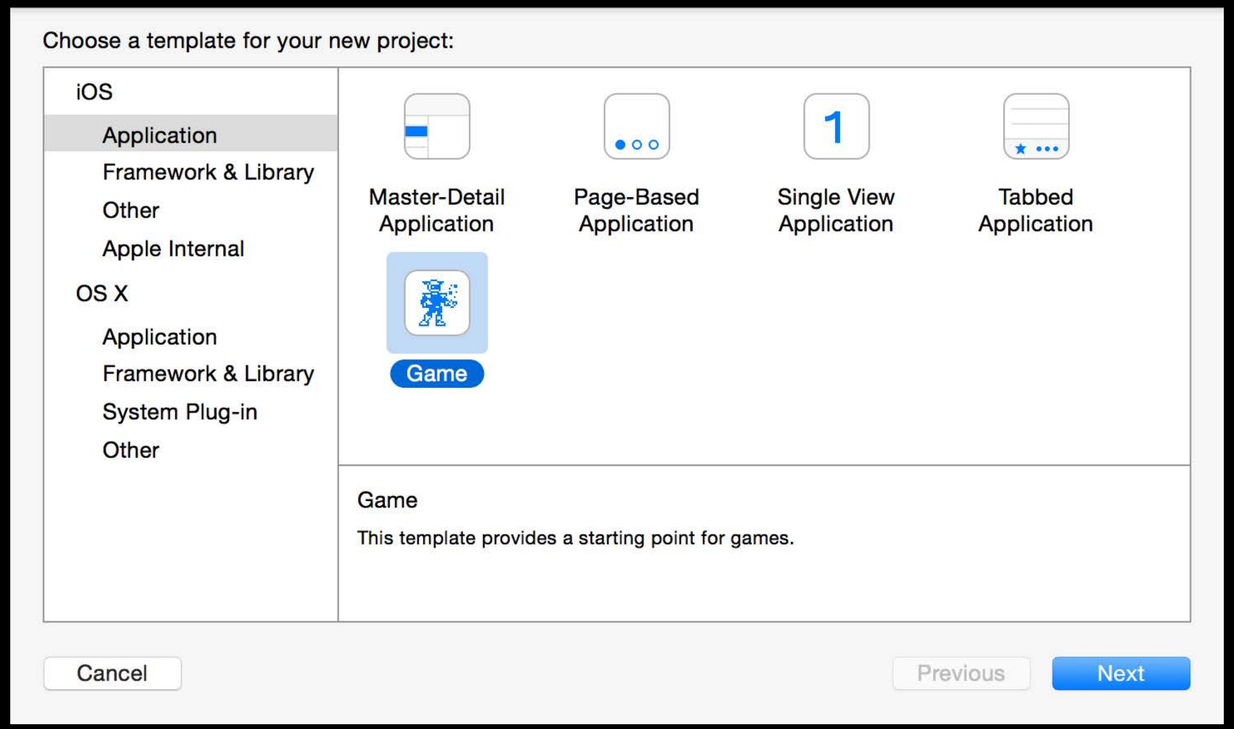
```
var pixelBuffer : UnsafeMutablePointer<Unmanaged<CVPixelBuffer>?>
CVPixelBufferCreate(nil, width: width, height: height,
    pixelFormatType: kCVPixelFormatType_32RGBA,
    pixelFormatAttributes: [kCVPixelBufferIOSurfacePropertiesKey: []],
    pixelBufferOut: pixelBuffer)
```

CVPixelBufferPools use this trick

Core Image and SpriteKit

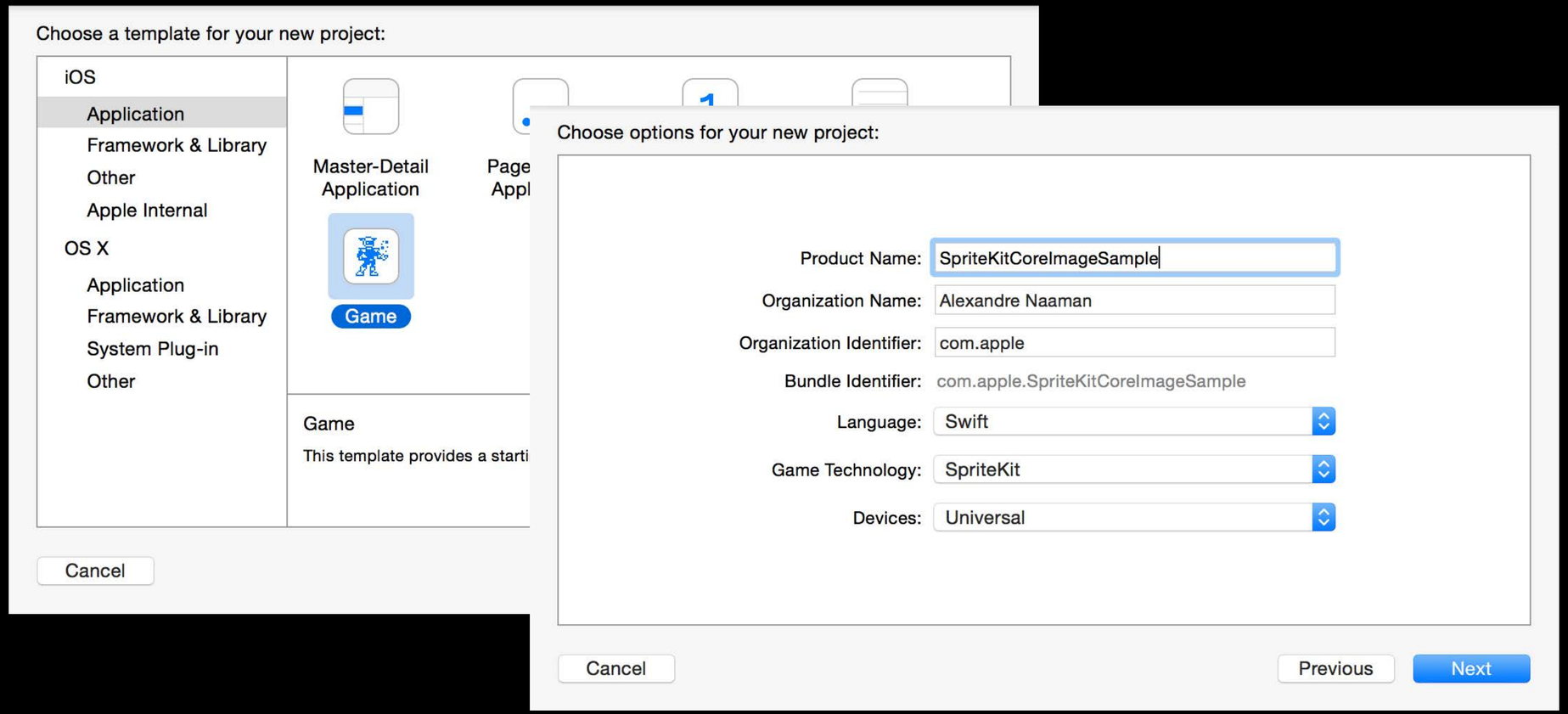
SpriteKit

In Xcode create a new SpriteKit game template (for OS X or iOS)



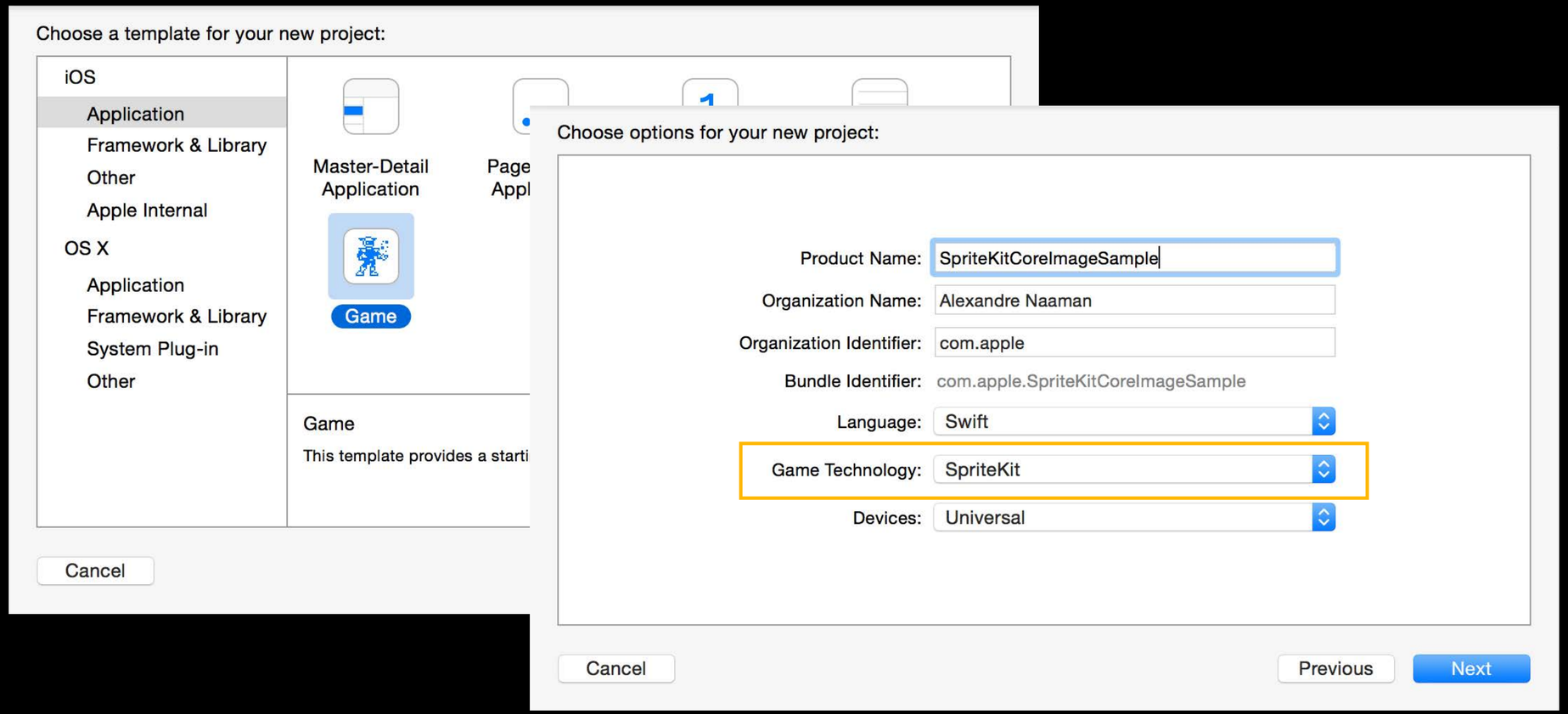
SpriteKit

In Xcode create a new SpriteKit game template (for OS X or iOS)



SpriteKit

In Xcode create a new SpriteKit game template (for OS X or iOS)




Hello, World!
1 node 60.0 fps


Hello, World!
1 node 60.0 fps

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...

        self.addChild(sprite)
    }
}
```

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...
```

```
        }
}
```

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...

        let effect = SKEffectNode()

                }
}
```

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...

        let effect = SKEffectNode()
        effect.addChild(sprite)



        }
}
```

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...

        let effect = SKEffectNode()
        effect.addChild(sprite)
        effect.shouldEnableEffects = true




        }
}
```

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...

        let effect = SKEffectNode()
        effect.addChild(sprite)
        effect.shouldEnableEffects = true
        effect.filter = CIFilter(name: "CIPixellate",
                                 withInputParameters: [kCIInputScaleKey: 20.0])



        }
}
```

SpriteKit

Now modify touchesBegan() in GameScene.swift

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
{
        ...

        let effect = SKEffectNode()
        effect.addChild(sprite)
        effect.shouldEnableEffects = true
        effect.filter = CIFilter(name: "CIPixellate",
                                 withInputParameters: [kCIInputScaleKey: 20.0])

        self.addChild(effect)

        }
}
```

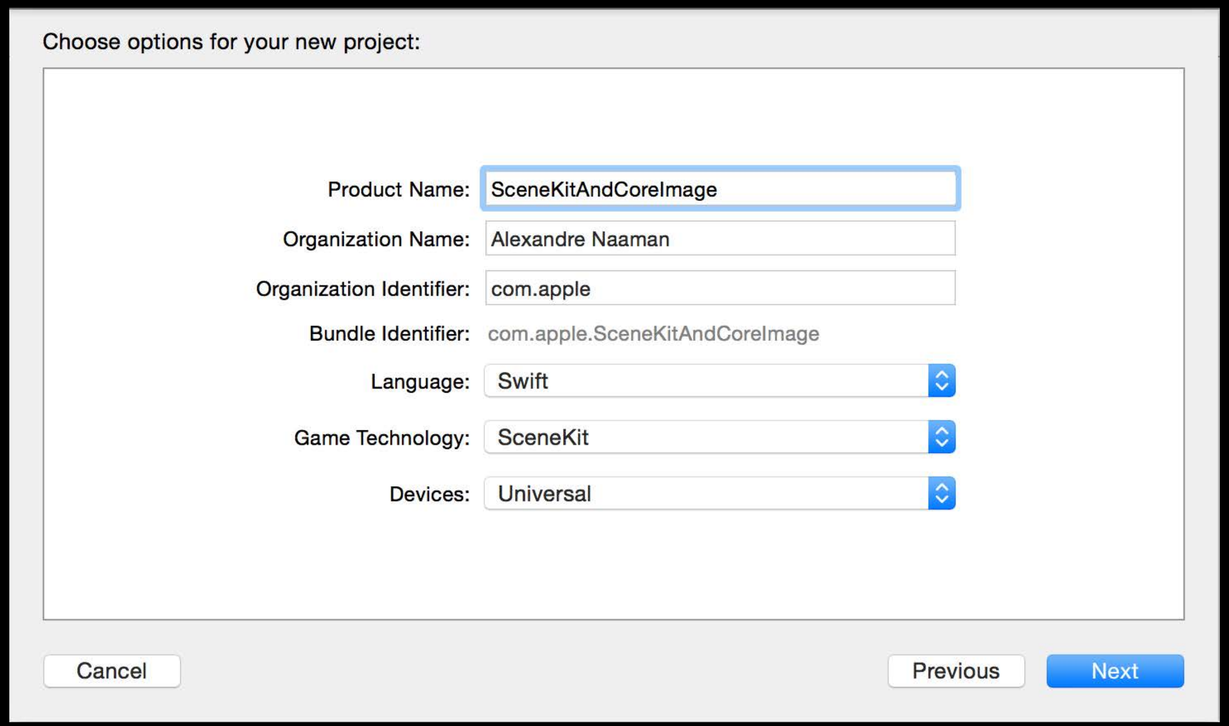

Hello, World!
1 node 60.0 fps


Hello, World!
1 node 60.0 fps

Core Image and SceneKit

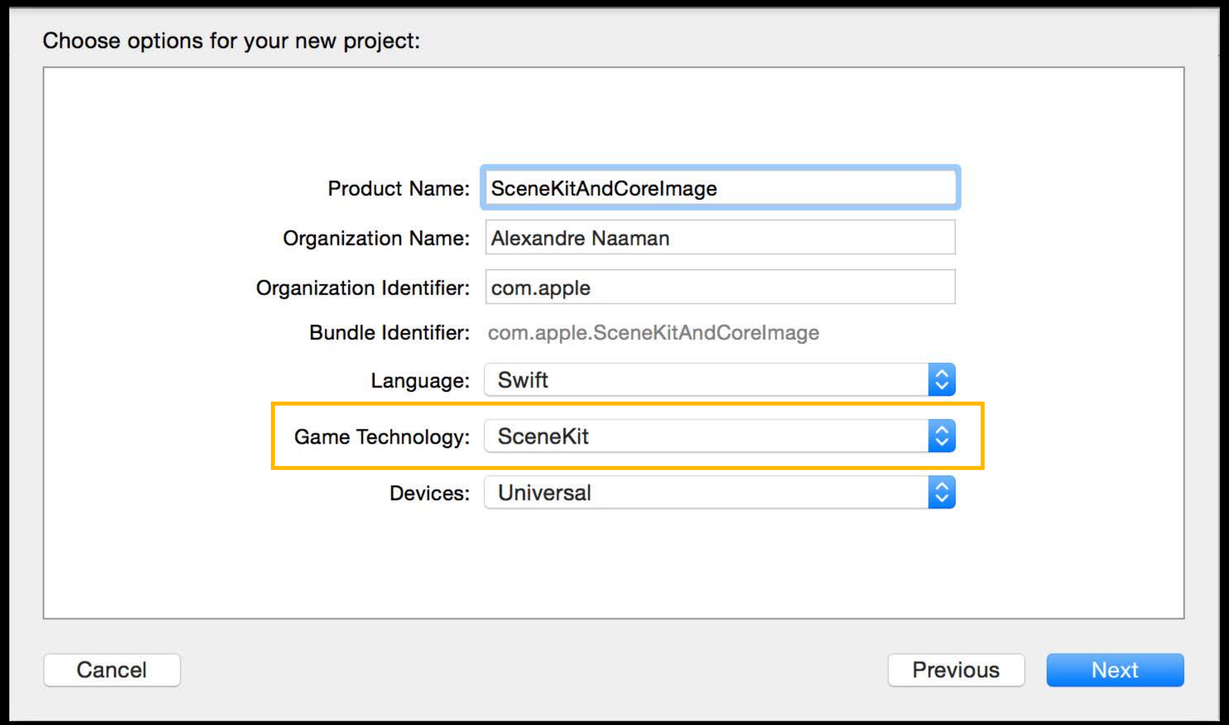
SceneKit

In Xcode create a new SceneKit game template (for OS X or iOS)



SceneKit

In Xcode create a new SceneKit game template (for OS X or iOS)




+
Mt 60fps
◆2 ▲2.87K •


+
Mt 60fps
◆2 ▲2.87K •

SceneKit

Modify viewDidLoad() inside of GameViewController.swift

```
// retrieve the ship node
let ship = scene.rootNode.childNodeWithName("ship", recursively: true)!
```

SceneKit

Modify viewDidLoad() inside of GameViewController.swift

```
// retrieve the ship node
let ship = scene.rootNode.childNodeWithName("ship", recursively: true)!

let pixellate = CIFilter(name: ”CIPixellate”,
                withInputParameters: [kCIInputScaleKey: 20.0])
ship.filters = [pixellate]
```


+
0fps
◆2 ▲2.81K ·8.42K ●


+
0fps
◆2 ▲2.81K ·8.42K ●

SceneKit

Filter properties on are animatable with Core Animation

```
let animation = CABasicAnimation(keyPath:
                        "filters.\(pixellate.name).\(kCIInputScaleKey)")
animation.toValue = 50
animation.fromValue = 0
animation.autoreverses = true
animation.repeatCount = FLT_MAX
animation.duration = 2.0
animation.timingFunction = CAMediaTimingFunction(name:
                                    kCAMediaTimingFunctionEaseInEaseOut)
ship.addAnimation(animation, forKey: nil)
```


0fps
◆2 ▲2.81K ·8.42K

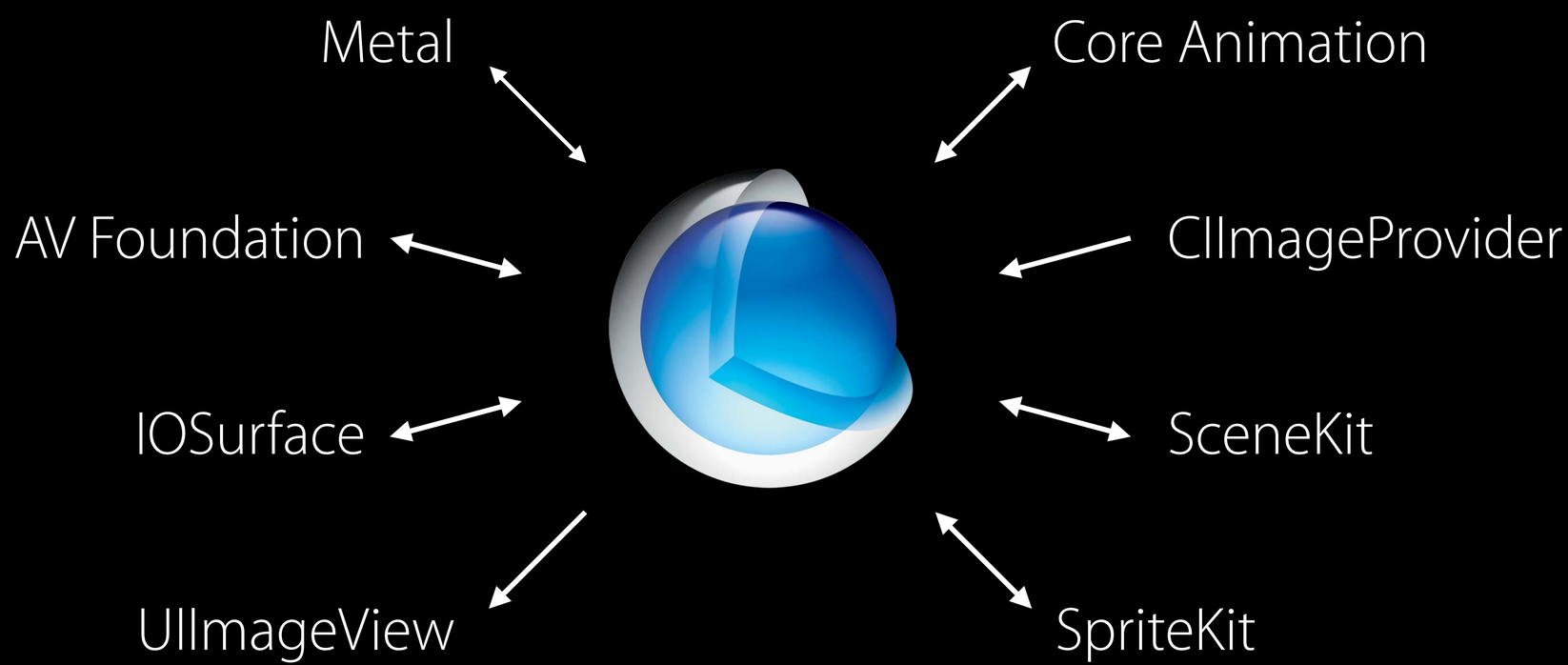

0fps
◆2 ▲2.81K ·8.42K


Score 0
1:56 Time
GL 52fps
◆ 40 ▲ 51K


Score 0
1:56 Time
GL 52fps
◆ 40 ▲ 51K

Final Thoughts

Final Thoughts



More Information

Technical Support

Apple Developer Forums

http://developer.apple.com/forums

Developer Technical Support

http://developer.apple.com/support/technical

General Inquiries

Stephen Chick, Evangelist

chick@apple.com

Related Sessions

Editing Movies in AV Foundation	Nob Hill	Wednesday 3:30PM
What's New In Metal, Part 2	Nob Hill	Thursday 9:00AM

Related Labs

Core Image Lab	Graphics, Games, and Media Lab B	Friday 11:00AM
AVKit and AV Foundation Lab	Graphics, Games, and Media Lab B	Friday 1:30PM

WWDC15