

Advances in Core Image

Filters, Metal, Vision, and More

Session 510

David Hayward, Core Image Manager

Agenda

A brief overview of Core Image

Highlights of what's new this year

Details on new APIs and features

A Brief Overview of Core Image

A Very Brief Introduction to Core Image

A simple, high-performance API to apply filters to images



Original CImage

Sepia
Filter

Hue
Filter

Contrast
Filter



Output CImage

A Very Brief Introduction to Core Image

Automatically tiles if images are large or graph is complex



Original CImage

Sepia
Filter

Hue
Filter

Contrast
Filter



Output CImage

A Very Brief Introduction to Core Image

Automatically tiles if only a region of the output is rendered



Original CImage

Sepia
Filter

Hue
Filter

Contrast
Filter



Output CImage

A Very Brief Introduction to Core Image

Each CIFilter has one or more CIKernel functions



Original CImage

Sepia
Filter

Hue
Filter

Contrast
Filter



Output CImage

kernel vec4 sepia ()

kernel vec4 hue ()

kernel vec4 contrast ()

A Very Brief Introduction to Core Image

Multiple CIKernels are concatenated to improve performance



Original CIImage

Concatenated
Program



Output CIImage

kernel vec4 sepia ()

kernel vec4 hue ()

kernel vec4 contrast ()

Summary of What's New

Our goal for this release is to enable developers to get better:

Summary of What's New

Our goal for this release is to enable developers to get better:



Performance

Summary of What's New

Our goal for this release is to enable developers to get better:



Performance



Information

Summary of What's New

Our goal for this release is to enable developers to get better:

Performance

Information

Functionality

Summary of What's New

Our goal for this release is to enable developers to get better:

Performance

Write CIKernels in Metal
CIRenderDestination API

Information

Functionality

Summary of What's New

Our goal for this release is to enable developers to get better:

Performance

Write CIKernels in Metal
CIRenderDestination API

Information

CIRenderInfo API
Xcode Quick Looks

Functionality

Summary of What's New

Our goal for this release is to enable developers to get better:

Performance

Write CIKernels in Metal
CIRenderDestination API

Information

CIRenderInfo API
Xcode Quick Looks

Functionality

New Filters
Barcode Support
Depth Support

Summary of What's New

Our goal for this release is to enable developers to get better:

Performance

Write CIKernels in Metal
CIRenderDestination API

Information

CIRenderInfo API
Xcode Quick Looks

Functionality

New Filters
Barcode Support
Depth Support

New Built-In ClFilters

196 Built-In Filters

196 Built-In Filters

AccordionFoldTransition	ColorBurnBlendMode	DisparityToDepth	HueSaturationValueGradient	PageCurlTransition	SoftLightBlendMode
AdditionCompositing	ColorClamp	DisplacementDistortion	Kaleidoscope	PageCurlWithShadowTransition	SourceAtopCompositing
AffineClamp	ColorControls	DissolveTransition	LabDeltaE	ParallelogramTile	SourceInCompositing
AffineTile	ColorCrossPolynomial	DivideBlendMode	LanczosScaleTransform	PDF417BarcodeGenerator	SourceOutCompositing
AffineTransform	ColorCube	DotScreen	LenticularHaloGenerator	PerspectiveCorrection	SourceOverCompositing
AreaAverage	ColorCubesMixedWithMask	Droste	LightenBlendMode	PerspectiveTile	SpotColor
AreaHistogram	ColorCubeWithColorSpace	EdgePreserveUpsampleFilter	LightTunnel	PerspectiveTransform	SpotLight
AreaMaximum	ColorCurves	Edges	LinearBurnBlendMode	PerspectiveTransformWithExtent	SRGBToneCurveToLinear
AreaMaximumAlpha	ColorDodgeBlendMode	EdgeWork	LinearDodgeBlendMode	PhotoEffectChrome	StarShineGenerator
AreaMinimum	ColorInvert	EightfoldReflectedTile	LinearGradient	PhotoEffectFade	StraightenFilter
AreaMinimumAlpha	ColorMap	ExclusionBlendMode	LinearToSRGBToneCurve	PhotoEffectInstant	StretchCrop
AreaMinMaxRed	ColorMatrix	ExposureAdjust	LineOverlay	PhotoEffectMono	StripesGenerator
AttributedTextImageGenerator	ColorMonochrome	FalseColor	LineScreen	PhotoEffectNoir	SubtractBlendMode
AztecCodeGenerator	ColorPolynomial	FlashTransition	LuminosityBlendMode	PhotoEffectProcess	SunbeamsGenerator
BarcodeGenerator	ColorPosterize	FourfoldReflectedTile	MaskedVariableBlur	PhotoEffectTonal	SwipeTransition
BarsSwipeTransition	ColumnAverage	FourfoldRotatedTile	MaskToAlpha	PhotoEffectTransfer	TemperatureAndTint
BicubicScaleTransform	ComicEffect	FourfoldTranslatedTile	MaximumComponent	PinchDistortion	TextImageGenerator
BlendWithAlphaMask	ConstantColorGenerator	GammaAdjust	MaximumCompositing	PinLightBlendMode	Thermal
BlendWithMask	Convolution3X3	GaussianBlur	MedianFilter	Pixellate	ToneCurve
Bloom	Convolution5X5	GaussianGradient	MinimumComponent	Pointillize	TorusLensDistortion
BokehBlur	Convolution7X7	GlassDistortion	MinimumCompositing	QRCodeGenerator	TriangleKaleidoscope
BoxBlur	Convolution9Horizontal	GlassLozenge	ModTransition	RadialGradient	TriangleTile
BumpDistortion	Convolution9Vertical	GlideReflectedTile	MorphologyGradient	RandomGenerator	TwelvefoldReflectedTile
BumpDistortionLinear	CopyMachineTransition	Gloom	MorphologyMaximum	RippleTransition	TwirlDistortion
CheckerboardGenerator	Crop	HardLightBlendMode	MorphologyMinimum	RowAverage	UnsharpMask
CircleSplashDistortion	Crystallize	HatchedScreen	MotionBlur	SaturationBlendMode	Vibrance
CircularScreen	DarkenBlendMode	HeightFieldFromMask	MultiplyBlendMode	ScreenBlendMode	Vignette
CircularWrap	DepthBlurEffect	HexagonalPixellate	MultiplyCompositing	SepiaTone	VignetteEffect
Clamp	DepthOfField	HighlightShadowAdjust	NinePartStretched	ShadedMaterial	VortexDistortion
CMYKHalftone	DepthToDisparity	HistogramDisplayFilter	NinePartTiled	SharpenLuminance	WhitePointAdjust
Code128BarcodeGenerator	DifferenceBlendMode	HoleDistortion	NoiseReduction	SixfoldReflectedTile	XRay
ColorBlendMode	DiscBlur	HueAdjust	OpTile	SixfoldRotatedTile	ZoomBlur
	DisintegrateWithMaskTransition	HueBlendMode	OverlayBlendMode	SmoothLinearGradient	

New Built-In CIFilters

Some that are useful for Depth filtering

CIDepthToDisparity / CIDisparityToDepth

CI MorphologyMinimum / Maximum / Gradient

CIColorCubesMixedWithMask

CIAreaMinMaxRed

New Built-In CIFilters

Some that are useful for Depth filtering

CIDepthToDisparity / CIDisparityToDepth

CI MorphologyMinimum / Maximum / Gradient

CIColorCubesMixedWithMask

CIAreaMinMaxRed

CIDepthBlurEffect

New Built-In CIFilters

Some that are useful for Depth filtering

CIDepthToDisparity / CIDisparityToDepth

CI MorphologyMinimum / Maximum / Gradient

CIColorCubesMixedWithMask

CIAreaMinMaxRed

CI DepthBlurEffect

New Built-in CIFilters

Some that are often requested additions

CITextImageGenerator

CIColorCurves

CILabDeltaE

CIbicubicScaleTransform

CIBarcodeGenerator

New Built-in CIFilters

Some that are improved

CIHueBlendMode

ClSaturationBlendMode

ClColorBlendMode

ClLuminosityBlendMode

ClLinearBurnBlendMode

New Built-in CIFilters

Some that are improved

CIHueBlendMode

ClSaturationBlendMode

ClColorBlendMode

ClLuminosityBlendMode

ClLinearBurnBlendMode

The demosaic and noise reduction filters used for RAW files

Writing CJKernels in Metal

Tony Chu, Core Image Engineer

Writing CIKernels



Original CImage

Sepia
Filter

Hue
Filter

Contrast
Filter



Output CImage

kernel vec4 sepia ()

kernel vec4 hue ()

kernel vec4 contrast ()

Writing CIKernels



Original CImage

Sepia
Filter

Hue
Filter

Contrast
Filter



Output CImage

kernel vec4 sepia ()

kernel vec4 hue ()

kernel vec4 contrast ()

Writing CIKernels

Previously, kernels are written in CIKernel Language

Writing CIKernels

Previously, kernels are written in CIKernel Language

- Based on GLSL

Writing CLKernels

Previously, kernels are written in CLKernel Language

- Based on GLSL
- Language extensions to enable automatic tiling and subregion rendering

Writing CIKernels

Previously, kernels are written in CIKernel Language

- Based on GLSL
- Language extensions to enable automatic tiling and subregion rendering

```
destCoord()
```


Writing CTKernels

Previously, kernels are written in CTKernel Language

- Based on GLSL
- Language extensions to enable automatic tiling and subregion rendering

```
destCoord()
```

```
samplerTransform(sampler src, vec2 p)
```

Writing CIColors

Previously, kernels are written in CIColor Language

- Based on GLSL
- Language extensions to enable automatic tiling and subregion rendering

```
destCoord()  
samplerTransform(sampler src, vec2 p)  
sample(sampler src, vec2 p)
```

Writing CIColorKernels

Previously, kernels are written in CIColorKernel Language

- Based on GLSL
- Language extensions to enable automatic tiling and subregion rendering

```
destCoord()  
samplerTransform(sampler src, vec2 p)  
sample(sampler src, vec2 p)
```

- Translated, concatenated, and compiled at run-time to Metal or GLSL

Compiling CIKernels on First Render



- Translate CIKernels
- Concatenate CIKernels
- Compile to GPU Code

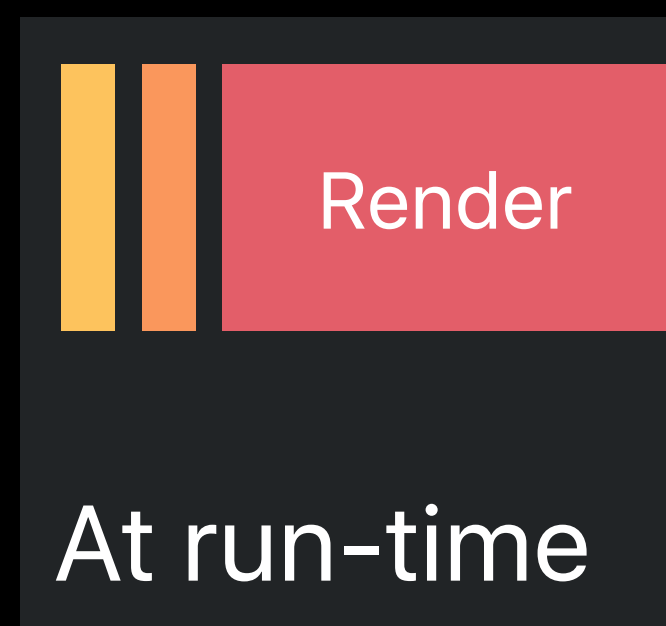
Compiling CIKernels on First Render



- Translate CIKernels
- Concatenate CIKernels
- Compile to GPU Code

Compiling CIKernels on First Render

At build-time (in Xcode)



- Translate CIKernels
- Concatenate CIKernels
- Compile to GPU Code

Writing CIColors in Metal



NEW

Now, you can write CIColors directly in Metal Shading Language

Writing CKernels in Metal



NEW

Now, you can write CKernels directly in Metal Shading Language

Benefits

Writing CLKernels in Metal



NEW

Now, you can write CLKernels directly in Metal Shading Language

Benefits

- Precompiled at build-time with error diagnostics

Writing CLKernels in Metal



NEW

Now, you can write CLKernels directly in Metal Shading Language

Benefits

- Precompiled at build-time with error diagnostics
- More modern language features

Writing CLKernels in Metal



NEW

Now, you can write CLKernels directly in Metal Shading Language

Benefits

- Precompiled at build-time with error diagnostics
- More modern language features
- Still supports concatenation and tiling

Writing CLKernels in Metal



NEW

Now, you can write CLKernels directly in Metal Shading Language

Benefits

- Precompiled at build-time with error diagnostics
- More modern language features
- Still supports concatenation and tiling
- Can be mixed with traditional CLKernels not written in Metal

Writing CIColors in Metal



NEW

Now, you can write CIColors directly in Metal Shading Language

Benefits

- Precompiled at build-time with error diagnostics
- More modern language features
- Still supports concatenation and tiling
- Can be mixed with traditional CIColors not written in Metal

Supported on iOS (for A8 or newer devices), macOS, and tvOS

How to Create Metal CIColors

How to Create Metal CIColorKernels

1. Write CIColorKernel in Metal shader file

How to Create Metal CIColorKernels

1. Write CIColorKernel in Metal shader file
2. Compile and link Metal shader file

How to Create Metal CIColors

1. Write CIColor in Metal shader file
2. Compile and link Metal shader file
3. Initialize CIColor with function from Metal library

How to Create Metal CIColorKernels

1. Write CIColorKernel in Metal shader file
2. Compile and link Metal shader file
3. Initialize CIColorKernel with function from Metal library

CIKernel Metal Library



NEW

New header file containing CIKernel extensions to the Metal Shading Language

CIKernel Metal Library



NEW

New header file containing CIKernel extensions to the Metal Shading Language

- CIKernel data types

CIKernel Metal Library



NEW

New header file containing CIKernel extensions to the Metal Shading Language

- CIKernel data types

`destination, sampler, sample_t`

CIKernel Metal Library



NEW

New header file containing CIKernel extensions to the Metal Shading Language

- CIKernel data types

 - `destination, sampler, sample_t`

- CIKernel functions

CIKernel Metal Library



NEW

New header file containing CIKernel extensions to the Metal Shading Language

- CIKernel data types

`destination, sampler, sample_t`

- CIKernel functions

`premultiply, unpremultiply, srgb_to_linear, linear_to_srgb, compare, cossin, sincos`

```
// CIKernelMetalLib.h

namespace coreimage {

    struct destination {
        float2 coord() const;
    };

    struct sampler {
        float2 transform(float2 p) const;
        float2 coord() const;
        float4 sample(float2 p) const;
        float4 extent() const;
    };

    // ...
}
```

```
// CIKernelMetalLib.h
```

```
namespace coreimage {
```

```
    struct destination {  
        float2 coord() const;  
    };
```

```
    struct sampler {  
        float2 transform(float2 p) const;  
        float2 coord() const;  
        float4 sample(float2 p) const;  
        float4 extent() const;  
    };
```

```
    // ...
```

```
}
```

```
// CIKernelMetalLib.h
```

```
namespace coreimage {
```

```
    struct destination {  
        float2 coord() const;  
    };
```

```
    struct sampler {  
        float2 transform(float2 p) const;  
        float2 coord() const;  
        float4 sample(float2 p) const;  
        float4 extent() const;  
    };
```

```
    // ...
```

```
}
```

Destination and Samplers

CIKernel Language

Metal

Get destination coordinate	<code>destCoord()</code>	<code>dest.coord()</code>
Transform coordinate to sampler space	<code>samplerTransform(src, p)</code>	<code>src.transform(p)</code>
Get destination coordinate in sampler space	<code>samplerCoord(src)</code>	<code>src.coord()</code>
Sample from source image	<code>sample(src, p)</code>	<code>src.sample(p)</code>
Get extent of source image	<code>samplerExtent(src)</code>	<code>src.extent()</code>

Destination and Samplers

CIKernel Language

Metal

Get destination coordinate	<code>destCoord()</code>	<code>dest.coord()</code>
Transform coordinate to sampler space	<code>samplerTransform(src, p)</code>	<code>src.transform(p)</code>
Get destination coordinate in sampler space	<code>samplerCoord(src)</code>	<code>src.coord()</code>
Sample from source image	<code>sample(src, p)</code>	<code>src.sample(p)</code>
Get extent of source image	<code>samplerExtent(src)</code>	<code>src.extent()</code>


```
// Metal CIWarpKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float2 myWarp (destination dest) {
    float2 p = dest.coord();
    // do something
    return p;
}

}}
```

```
// Metal CIWarpKernel Source
```

```
#include <metal_stdlib>
```

```
using namespace metal;
```

```
#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h
```

```
extern "C" { namespace coreimage {
```

```
float2 myWarp (destination dest) {
```

```
    float2 p = dest.coord();
```

```
    // do something
```

```
    return p;
```

```
}
```

```
}}
```

```
// Metal CIWarpKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float2 myWarp (destination dest) {
    float2 p = dest.coord();
    // do something
    return p;
}

}}
```

```
// Metal CIWarpKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float2 myWarp (destination dest) {
    float2 p = dest.coord();
    // do something
    return p;
}

}}
```

```
// Metal CIWarpKernel Source
```

```
#include <metal_stdlib>
```

```
using namespace metal;
```

```
#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h
```

```
extern "C" { namespace coreimage {
```

```
float2 myWarp (destination dest) {
```

```
    float2 p = dest.coord();
```

```
    // do something
```

```
    return p;
```

```
}
```

```
}}
```

```
kernel vec2 myWarp () {
```

```
    vec2 p = destCoord();
```

```
    // do something
```

```
    return p;
```

```
}
```

```
// Metal CIColorKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float4 myColor (sample_t s) {
    // do something
    return s;
}

}}
```

```
// Metal CIColorKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float4 myColor (sample_t s) {
    // do something
    return s;
}

}}
```

```
// Metal CIColorKernel Source
```

```
#include <metal_stdlib>
```

```
using namespace metal;
```

```
#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h
```

```
extern "C" { namespace coreimage {
```

```
float4 myColor (sample_t s) {  
    // do something  
    return s;  
}
```

```
kernel vec4 myColor (__sample s) {  
    // do something  
    return s;  
}
```

```
}}
```



```
// Metal CIKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float4 myKernel (sampler src) {
    float4 s = src.sample(src.coord());
    // do something
    return s;
}

}}
```

```
// Metal CIKernel Source

#include <metal_stdlib>
using namespace metal;

#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h

extern "C" { namespace coreimage {

float4 myKernel (sampler src) {
    float4 s = src.sample(src.coord());
    // do something
    return s;
}

}}
```

```
// Metal CIKernel Source
```

```
#include <metal_stdlib>
```

```
using namespace metal;
```

```
#include <CoreImage/CoreImage.h> // includes CIKernelMetalLib.h
```

```
extern "C" { namespace coreimage {
```

```
float4 myKernel (sampler src) {  
    float4 s = src.sample(src.coord());  
    // do something  
    return s;  
}
```

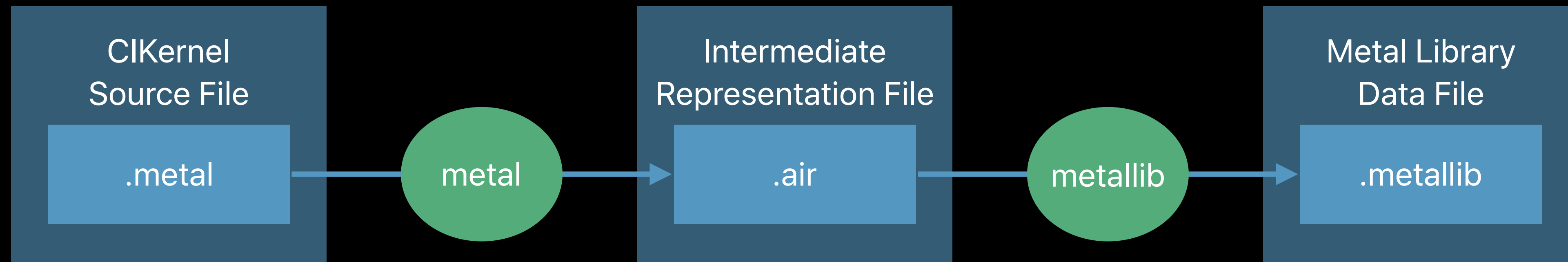
```
}}
```

```
kernel vec4 myKernel (sampler src) {  
    vec4 s = sample(src, samplerCoord(src));  
    // do something  
    return s;  
}
```

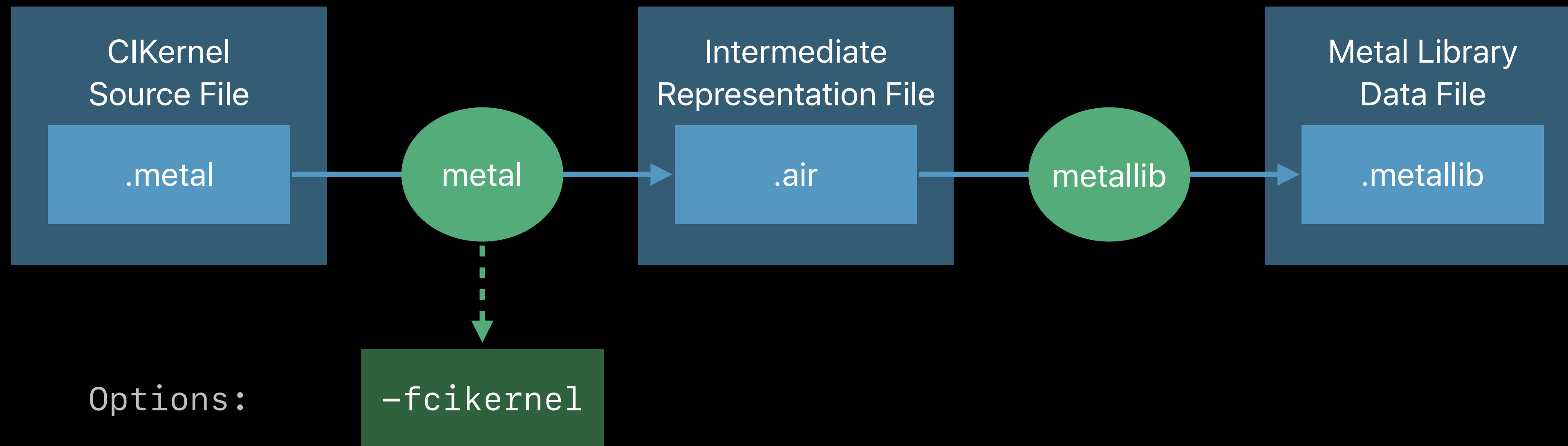
How to Create Metal CIColorKernels

1. Write CIColorKernel in Metal shader file
2. Compile and link Metal shader file
3. Initialize CIColorKernel with function from Metal library

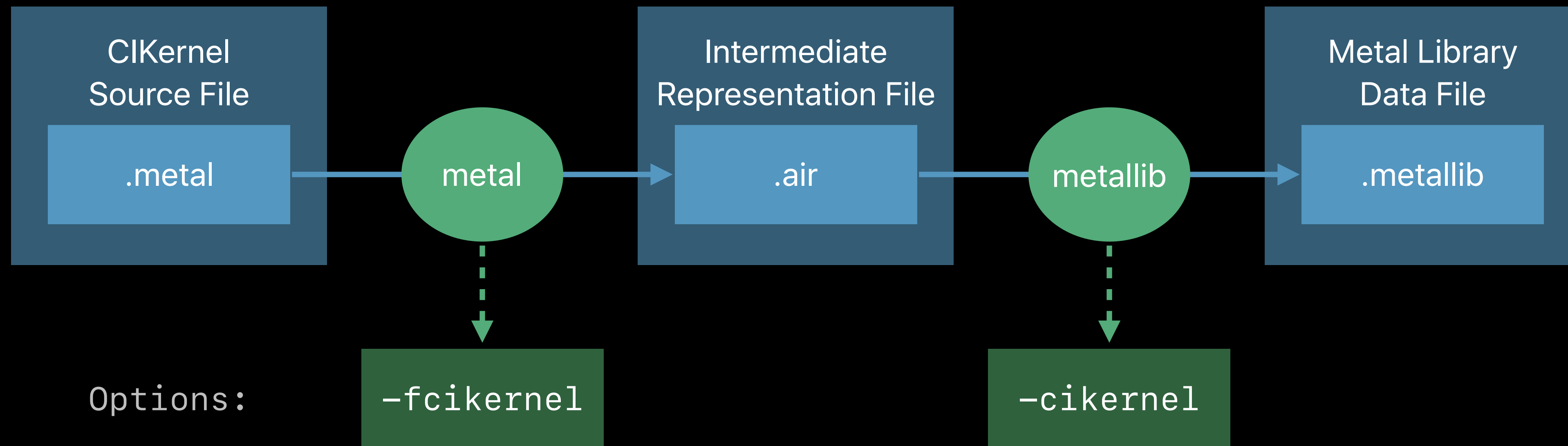
Compiling and Linking CIKernels



Compiling and Linking CIKernels



Compiling and Linking CIKernels





General

Capabilities

Resource Tags

Info

Build Settings

Build Phases

Build Rules

PROJECT

CIKernelDemo

TARGETS

CIKernelDemo

Basic

Customized

All

Combined

Levels



Q Metal



▼ Metal Compiler - Build Options

Setting

CIKernelDemo

Enable fast math

No

Header Search Paths

Ignore Warnings

No

Metal language revision



Optimization Level



Other Metal Compiler Flags

Preprocessor Definitions

▼ Produce debugging information

<Multiple values>

Debug

Yes

Release

No

Treat Warnings as Errors

No



Filter





General

Capabilities

Resource Tags

Info

Build Settings

Build Phases

Build Rules

PROJECT

CIKernelDemo

TARGETS

CIKernelDemo

Basic

Customized

All

Combined

Levels



Q Metal



▼ Metal Compiler - Build Options

Setting

CIKernelDemo

Enable fast math

No

Header Search Paths

Ignore Warnings

No

Metal language revision



Optimization Level



Other Metal Compiler Flags

Preprocessor Definitions

▼ Produce debugging information

<Multiple values>

Debug

Yes

Release

No

Treat Warnings as Errors

No

+ - Filter



PROJECT

CIKernelDemo

TARGETS

CIKernelDemo

Basic

Customized

All

Combined

Levels

+

Metal

▼ Metal Compiler - Build Options

Setting

CIKernelDemo

Enable fast math

No

Header Search Paths

Ignore Warnings

No

Metal language revision

◇

Optimization Level

◇

Other Metal Compiler Flags

Preprocessor Definitions

▼ Produce debugging information

<Multiple values>

Debug

Yes

Release

No

Trea

```
xcrun metal -fcikernel MyKernels.metal -o MyKernels.air
xcrun metallib -cikernel MyKernels.air -o MyKernels.metallib
```



Filter



How to Create Metal CIColors

1. Write CIColor in Metal shader file
2. Compile and link Metal shader file
3. Initialize CIColor with function from Metal library

New CIKernel API



NEW

```
init(functionName name: String, fromMetalLibraryData data: Data) throws
```

```
init(functionName name: String, fromMetalLibraryData data: Data,  
      outputPixelFormat format: CIFORMat) throws
```

New CIKernel API

NEW

```
init(functionName name: String, fromMetalLibraryData data: Data) throws  
init(functionName name: String, fromMetalLibraryData data: Data,  
      outputPixelFormat format: CIFORMat) throws
```

```
// Example of initializing CIKernels  
  
let url = Bundle.main.url(forResource: "default", withExtension: "metallib")!  
let data = try! Data(contentsOf: url)  
  
let kernel = try? CIKernel(functionName: "myKernel", fromMetalLibraryData: data)
```

New CIKernel API

NEW

```
init(functionName name: String, fromMetalLibraryData data: Data) throws  
init(functionName name: String, fromMetalLibraryData data: Data,  
      outputPixelFormat format: CIFORMat) throws
```

```
// Example of initializing CIKernels
```

```
let url = Bundle.main.url(forResource: "default", withExtension: "metallib")!
```

```
let data = try! Data(contentsOf: url)
```

```
let kernel = try? CIKernel(functionName: "myKernel", fromMetalLibraryData: data)
```

```
let warpKernel = try? CIWarpKernel(functionName: "myWarp", fromMetalLibraryData: data)
```

```
let colorKernel = try? CIColorKernel(functionName: "myColor", fromMetalLibraryData: data)
```

New CIRenderDestination API

New CIRenderDestination API

A consistent API to render to different destination types

- IOSurface
- CVPixelBuffer
- Metal Texture
- OpenGL Texture
- Memory buffer

New CIRenderDestination API

A consistent API to render to different destination types

- IOSurface
- CVPixelBuffer
- Metal Texture
- OpenGL Texture
- Memory buffer

Returns immediately if render fails—with an error

New CIRenderDestination API

Allow setting of common properties for the destination

- Alpha mode behavior
- Clamping mode behavior
- Destination colorspace

New CIRenderDestination API

Allow setting of common properties for the destination

- Alpha mode behavior
- Clamping mode behavior
- Destination colorspace

And some advanced properties

- Dithering
- Blending

New CIRenderDestination API

Allow setting of common properties for the destination

- Alpha mode behavior
- Clamping mode behavior
- Destination colorspace

And some advanced properties

- Dithering
- Blending

Reduces the need for multiple CIContexts

New CIRenderDestination API

Performance Benefits

Existing CGContext APIs for rendering to IOSurface or CVPixelBuffer

- Return to caller when render is completed

New CIRenderDestination API

Performance Benefits

Existing CGContext APIs for rendering to IOSurface or CVPixelBuffer

- Return to caller when render is completed

New CIRenderDestination APIs

- Return to caller when work is issued

New CIRenderDestination API

Performance Benefits

Existing CGContext APIs for rendering to IOSurface or CVPixelBuffer

- Return to caller when render is completed

New CIRenderDestination APIs

- Return to caller when work is issued

This allows CPU and GPU work to be pipelined efficiently

```
// Clear Then Render Foreground Blended with Background

func renderImagePair (foregroundImage : CIImage, backgroundImage : CIImage,
    blend : CIBlendKernel = CIBlendKernel.sourceOver
    toSurface : IOSurface)
{
    let dest = CIRenderDestination(ioSurface: toSurface)

    let ctx = getContext() // don't create a context each time

    try? ctx.startTask(toClear: dest)

    try? ctx.startTask(toRender: backgroundImage, to: dest)

    dest.blendKernel = blend // use one of 37 built-in blend kernels or create your own

    try? ctx.startTask(toRender: foregroundImage, to: dest).waitUntilCompleted()
}
```



```
// Clear Then Render Foreground Blended with Background

func renderImagePair (foregroundImage : CIImage, backgroundImage : CIImage,
    blend : CIBlendKernel = CIBlendKernel.sourceOver
    toSurface : IOSurface)
{
    let dest = CIRenderDestination(ioSurface: toSurface)

    let ctx = getContext() // don't create a context each time

    try? ctx.startTask(toClear: dest)

    try? ctx.startTask(toRender: backgroundImage, to: dest)

    dest.blendKernel = blend // use one of 37 built-in blend kernels or create your own

    try? ctx.startTask(toRender: foregroundImage, to: dest).waitUntilCompleted()
}
```

```
// Clear Then Render Foreground Blended with Background

func renderImagePair (foregroundImage : CIImage, backgroundImage : CIImage,
    blend : CIBlendKernel = CIBlendKernel.sourceOver
    toSurface : IOSurface)
{
    let dest = CIRenderDestination(ioSurface: toSurface)

    let ctx = getContext() // don't create a context each time

    try? ctx.startTask(toClear: dest)

    try? ctx.startTask(toRender: backgroundImage, to: dest)

    dest.blendKernel = blend // use one of 37 built-in blend kernels or create your own

    try? ctx.startTask(toRender: foregroundImage, to: dest).waitUntilCompleted()
}
```

```
// Clear Then Render Foreground Blended with Background
```

```
func renderImagePair (foregroundImage : CIImage, backgroundImage : CIImage,  
    blend : CIBlendKernel = CIBlendKernel.sourceOver  
    toSurface : IOSurface)
```

```
{
```

```
    let dest = CIRenderDestination(ioSurface: toSurface)
```

```
    let ctx = getContext() // don't create a context each time
```

```
    try? ctx.startTask(toClear: dest)
```

```
    try? ctx.startTask(toRender: backgroundImage, to: dest)
```

```
    dest.blendKernel = blend // use one of 37 built-in blend kernels or create your own
```

```
    try? ctx.startTask(toRender: foregroundImage, to: dest).waitUntilCompleted()
```

```
}
```

```
// Clear Then Render Foreground Blended with Background
```

```
func renderImagePair (foregroundImage : CIImage, backgroundImage : CIImage,
```

```
    blend : CIBlendKernel = CIBlendKernel.sourceOver
```

```
    toSurface : IOSurface)
```

```
{
```

```
    let dest = CIRenderDestination(ioSurface: toSurface)
```

```
    let ctx = getContext() // don't create a context each time
```

```
    try? ctx.startTask(toClear: dest)
```

```
    try? ctx.startTask(toRender: backgroundImage, to: dest)
```

```
    dest.blendKernel = blend // use one of 37 built-in blend kernels or create your own
```

```
    try? ctx.startTask(toRender: foregroundImage, to: dest).waitUntilCompleted()
```

```
}
```

```
// Clear Then Render Foreground Blended with Background

func renderImagePair (foregroundImage : CIImage, backgroundImage : CIImage,
    blend : CIBlendKernel = CIBlendKernel.sourceOver
    toSurface : IOSurface)
{
    let dest = CIRenderDestination(ioSurface: toSurface)

    let ctx = getContext() // don't create a context each time

    try? ctx.startTask(toClear: dest)

    try? ctx.startTask(toRender: backgroundImage, to: dest)

    dest.blendKernel = blend // use one of 37 built-in blend kernels or create your own

    try? ctx.startTask(toRender: foregroundImage, to: dest).waitUntilCompleted()
}
```

Rendering to Metal Drawable Textures

```
let ctx = getContext() // don't create a context each time

let texture = currentDrawable.texture

let dest = CIRenderDestination(texture, commandBuffer: cmdBuffer)

try? ctx.startTask(toRender: image, to: dest)
```

Rendering to Metal Drawable Textures

```
let ctx = getContext() // don't create a context each time
```

```
let texture = currentDrawable.texture
```

```
let dest = CIRenderDestination(texture, commandBuffer: cmdBuffer)
```

```
try? ctx.startTask(toRender: image, to: dest)
```

Rendering to Metal Drawable Textures

```
let ctx = getContext() // don't create a context each time
```

```
let texture = currentDrawable.texture
```

```
let dest = CIRenderDestination(texture, commandBuffer: cmdBuffer)
```

```
try? ctx.startTask(toRender: image, to: dest)
```


Rendering to Metal Drawable Textures

```
let ctx = getContext() // don't create a context each time

while stillDrawing {
    let texture = currentDrawable.texture

    let dest = CIRenderDestination(texture, commandBuffer: cmdBuffer)

    try? ctx.startTask(toRender: image, to: dest)
}
```

Rendering to Metal Drawable Textures

CPU

GPU



Rendering to Metal Drawable Textures

CPU

GPU

Get drawable



Rendering to Metal Drawable Textures

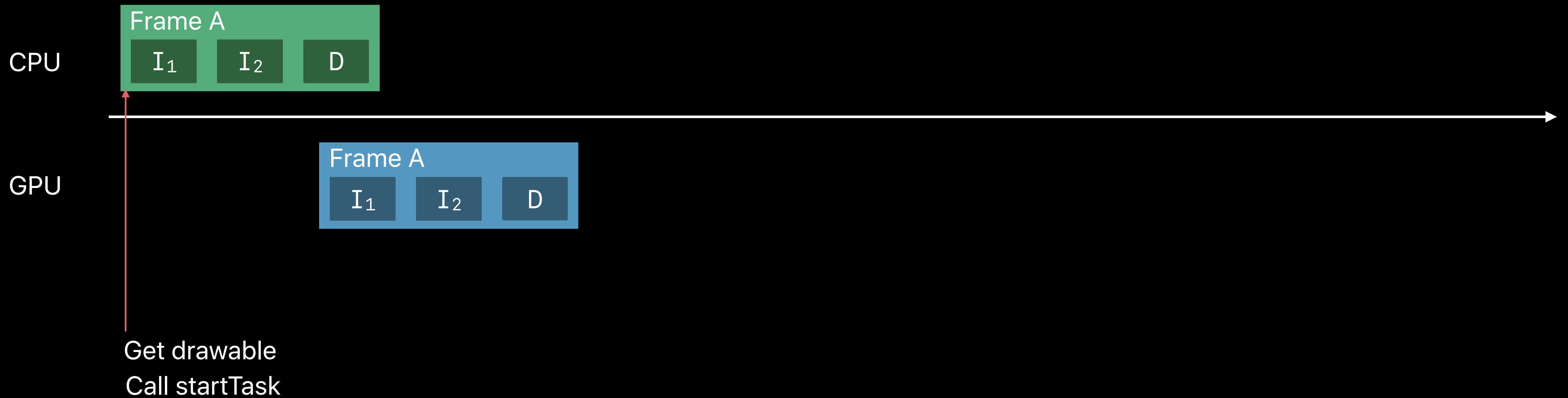
CPU

GPU

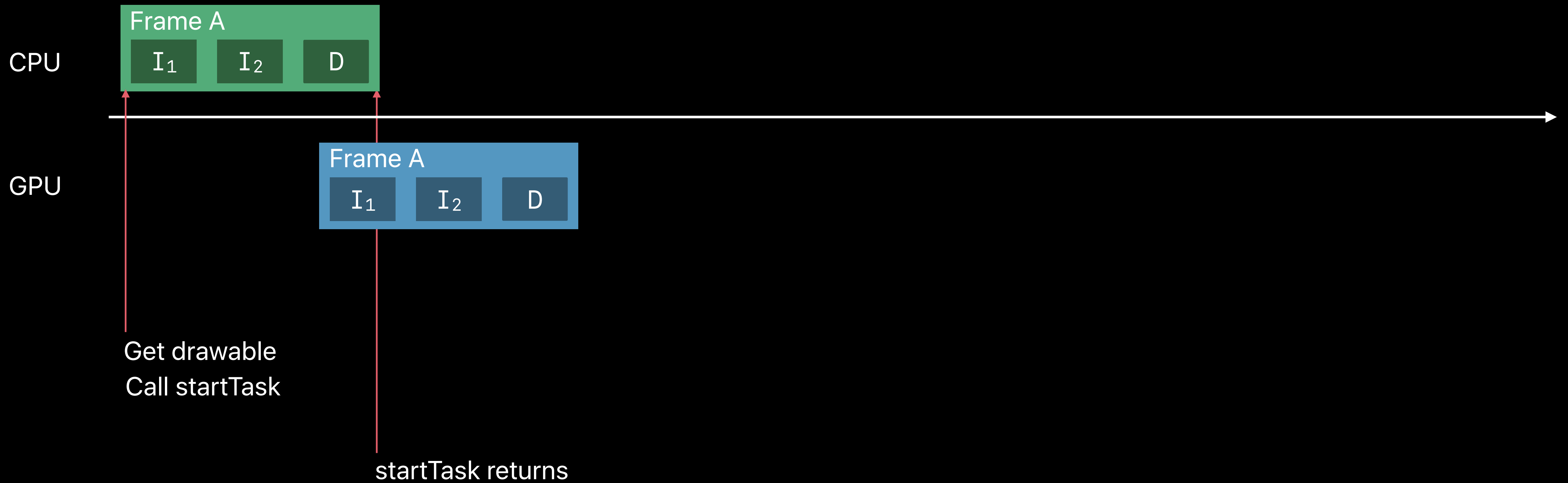


Get drawable
Call startTask

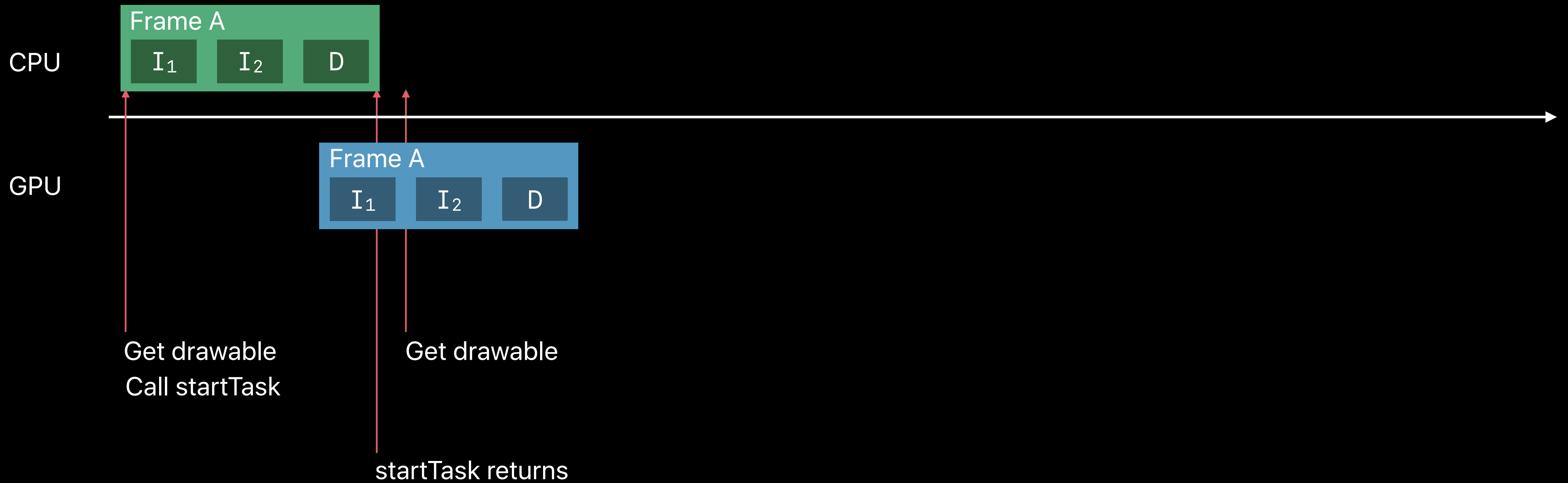
Rendering to Metal Drawable Textures



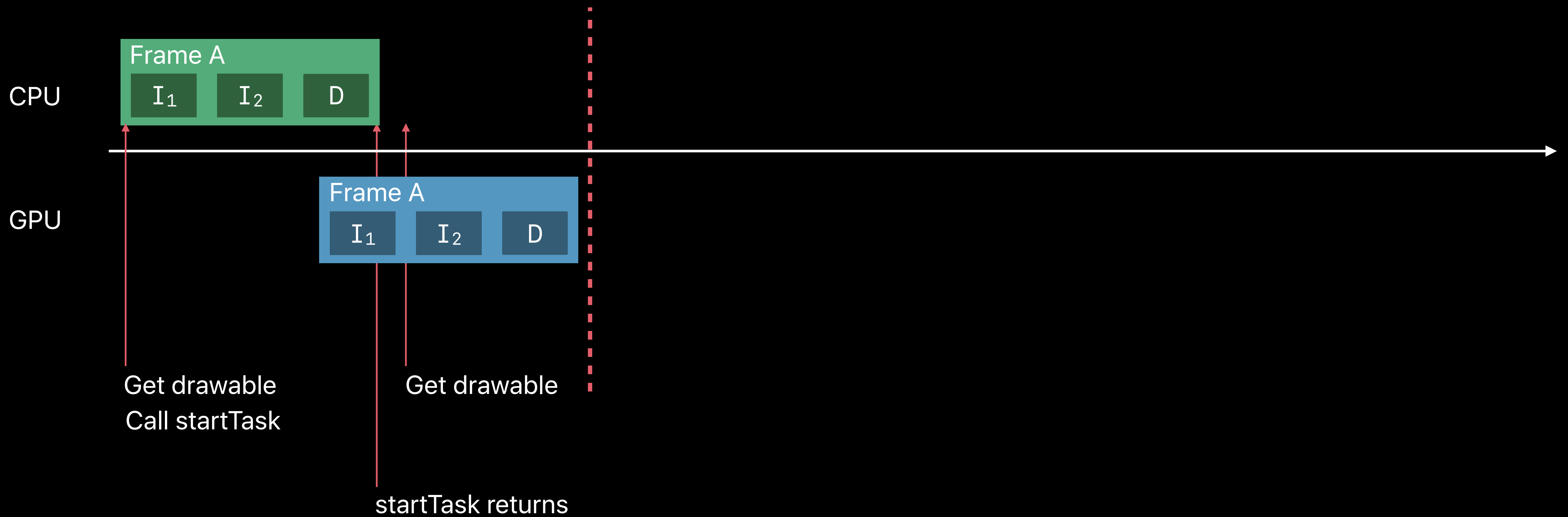
Rendering to Metal Drawable Textures



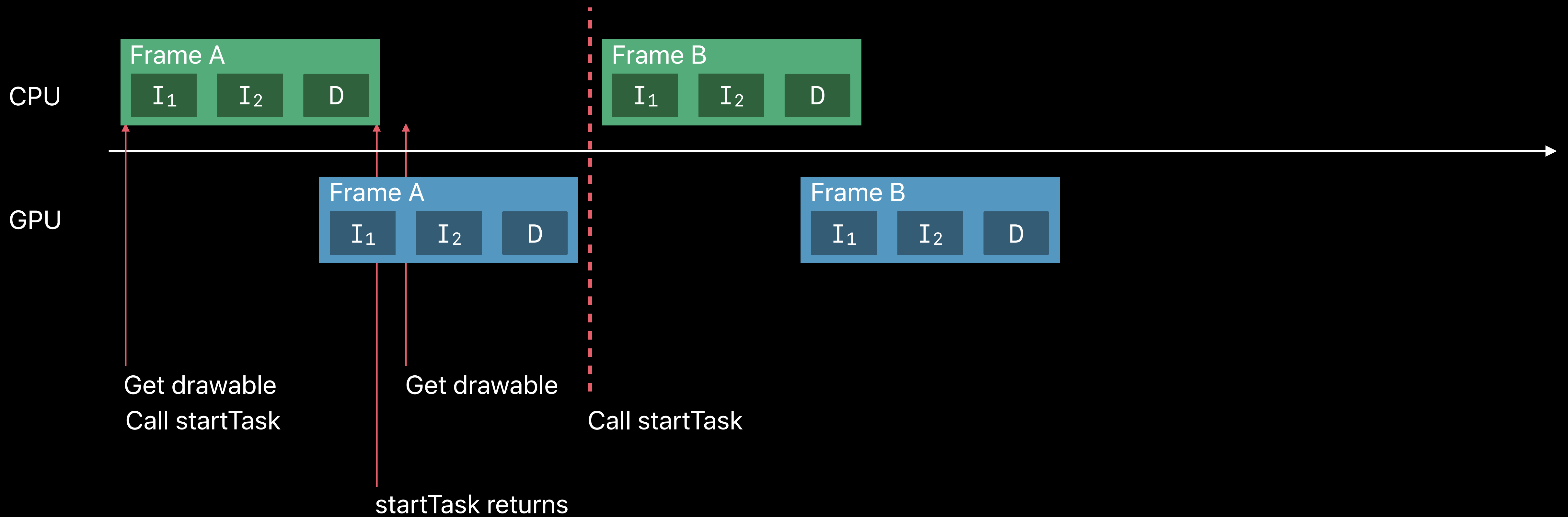
Rendering to Metal Drawable Textures



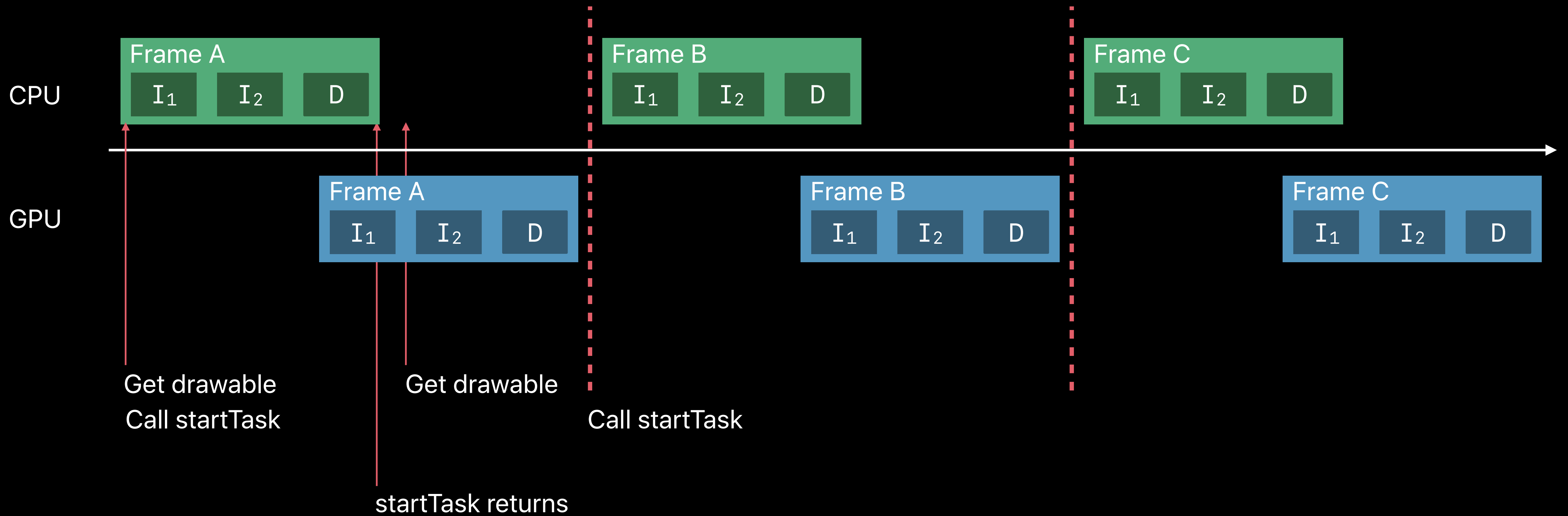
Rendering to Metal Drawable Textures



Rendering to Metal Drawable Textures



Rendering to Metal Drawable Textures



Rendering to Metal Drawable Textures Improved



```
let ctx = getContext() // don't create a context each time

while stillDrawing {
    let dest = CIRenderDestination(
        width: 1024,
        height: 768,
        pixelFormat: MTLPixelFormat.rgb8Unorm,
        commandBuffer: cmdBuffer) { () -> MTLTexture in
        return currentDrawable.texture
    })

    try? ctx.startTask(toRender: image, to: dest)
}
```

Rendering to Metal Drawable Textures Improved



```
let ctx = getContext() // don't create a context each time

while stillDrawing {
  let dest = CIRenderDestination(
    width: 1024,
    height: 768,
    pixelFormat: MTLPixelFormat.rgb8Unorm,
    commandBuffer: cmdBuffer) { () -> MTLTexture in
    return currentDrawable.texture
  })

  try? ctx.startTask(toRender: image, to: dest)
}
```

Rendering to Metal Drawable Textures Improved



```
let ctx = getContext() // don't create a context each time

while stillDrawing {
    let dest = CIRenderDestination(
        width: 1024,
        height: 768,
        pixelFormat: MTLPixelFormat.rgb8Unorm,
        commandBuffer: cmdBuffer) { () -> MTLTexture in
        return currentDrawable.texture
    })

    try? ctx.startTask(toRender: image, to: dest)
}
```

Rendering to Metal Drawable Textures Improved

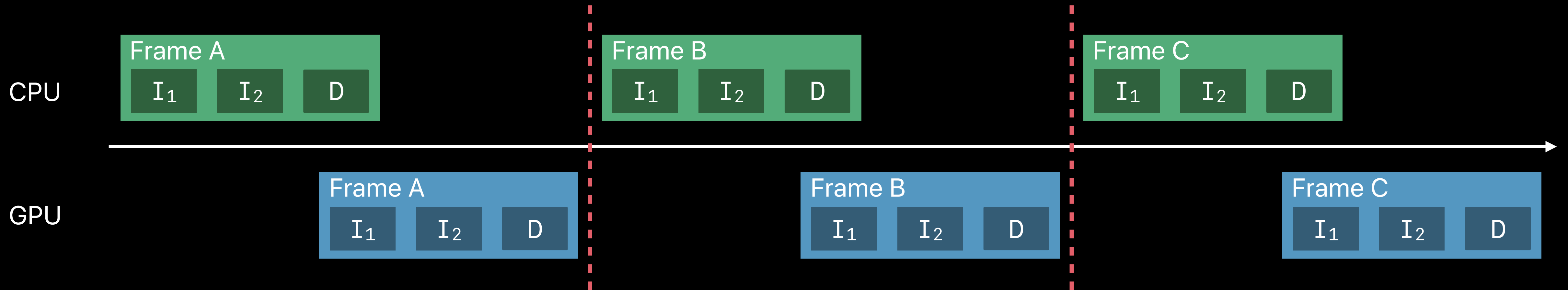


```
let ctx = getContext() // don't create a context each time

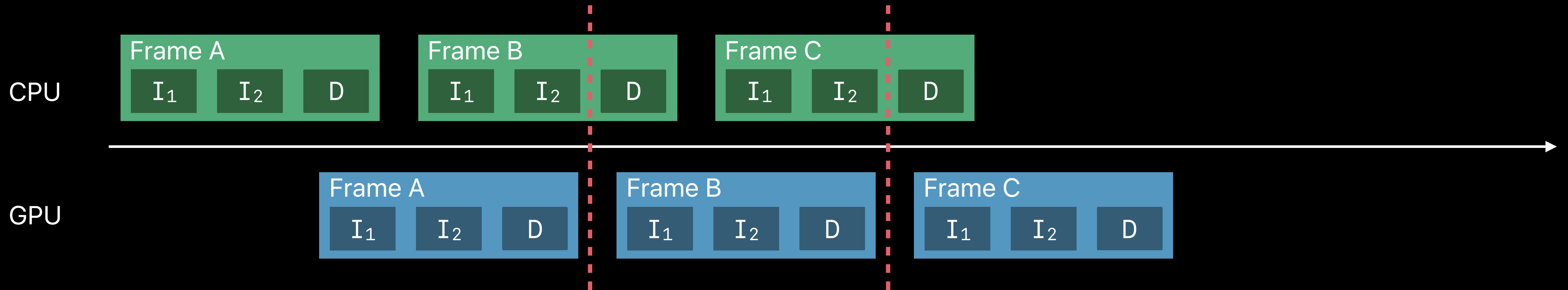
while stillDrawing {
    let dest = CIRenderDestination(
        width: 1024,
        height: 768,
        pixelFormat: MTLPixelFormat.rgb8Unorm,
        commandBuffer: cmdBuffer) { () -> MTLTexture in
        return currentDrawable.texture
    })

    try? ctx.startTask(toRender: image, to: dest)
}
```

Rendering to Metal Drawable Textures Improved



Rendering to Metal Drawable Textures Improved



Looking Inside Core Image

Quick Look support and other runtime information

Detailed Image and Render Information

See how Core Image works internally

CIRenderTask returns a CIRenderInfo object when complete

passCount

kernelExecutionTime

kernelPixelsProcessed

Detailed Image and Render Information

Awesome Core Image Quick Looks in Xcode

Detailed Image and Render Information

Awesome Core Image Quick Looks in Xcode

CIImage

Shows the image graph as constructed

Detailed Image and Render Information

Awesome Core Image Quick Looks in Xcode

CIImage

Shows the image graph as constructed

CIRenderTask

Shows how Core Image optimized the graph

Detailed Image and Render Information

Awesome Core Image Quick Looks in Xcode

CIImage

Shows the image graph as constructed

CIRenderTask

Shows how Core Image optimized the graph

CIRenderInfo

Shows concatenation, timing, and caching

CIImage Quick Look

```
// load image and apply orientation
var image = CIImage(
    contentsOf: url
    options: [kCIImageApplyOrientationProperty: true])

// downsample by 2x
image = image.applying(CGAffineTransform(scaleX:0.5, y:0.5))
```

CIImage Quick Look

```
// load image and apply orientation
var image = CIImage(
    contentsOf: url
    options: [kCIImageApplyOrientationProperty: true])

// downsample by 2x
image = image.applying(CGAffineTransform(scaleX:0.5, y:0.5))
```

▶ (CIImage)0x0000000100e61630 

CIImage Quick Look

```
// load image and apply orientation
```

```
var image = CIImage(  
    contentsOf: url  
    options: [kCIImageApplyOrientationProperty: true])
```

```
// downsample by 2x
```

```
image = image.applying(CGAffineTransform(scaleX:0.5, y:0.5))
```

```
▶ (CIImage)0x0000000100e61630 👁
```



CIImage «0x100e61630»

affine
0.5 0 0
0 0.5 0
extent=[0 0 1512 2016]
opaque

affine
0 -1 0
1 0 4032
extent=[0 0 3824 4032]
opaque

affine
1 0 0
0 -1 3824
extent=[0 0 4032 3824]
opaque

colormatch
display_P3_to_workingSpace
extent=[0 0 4032 3824]
opaque

IOSurface 0x100f48340(189) seed:1 YCC420f 601 alpha_one
extent=[0 0 4032 3824]
opaque

CIImage <0x100e61630>

affine
0.5 0 0
0 0.5 0
extent=[0 0 1512 2016]
opaque

affine
0 -1 0
1 0 4032
extent=[0 0 3024 4032]
opaque

affine
1 0 0
0 -1 3024
extent=[0 0 4032 3024]
opaque

colormatch
Display P3_to_workingspace
extent=[0 0 4032 3024]
opaque

IOSurface 0x100f483d0(189) seed:1 YCC420f 601 alpha_one
extent=[0 0 4032 3024]
opaque

CIImage <0x100e61630>

affine
0.5 0 0
0 0.5 0
extent=[0 0 1512 2016]
opaque

affine
0 -1 0
1 0 4032
extent=[0 0 3024 4032]
opaque

affine
1 0 0
0 -1 3024
extent=[0 0 4032 3024]
opaque

colormatch
Display P3_to_workingspace
extent=[0 0 4032 3024]
opaque

IOSurface 0x100f483d0(418) seed:1 YCC420f 601 alpha_one
extent=[0 0 4032 3024]
opaque

UIImage <0x100e61630>

affine
0.5 0 0
0 0.5 0
extent=[0 0 1512 2016]
opaque

affine
0 -1 0
1 0 4032
extent=[0 0 3024 4032]
opaque

affine
1 0 0
0 -1 3024
extent=[0 0 4032 3024]
opaque

colormatch
Display P3_to_workingspace
extent=[0 0 4032 3024]
opaque

IOSurface 0x100f483d0(189) seed:1 YCC420f 601 alpha_one
extent=[0 0 4032 3024]
opaque

CIImage <0x100e61630>

```
affine
0.5  0  0
0  0.5  0
extent=[0 0 1512 2016]
opaque
```

```
affine
0  -1  0
1  0  4032
extent=[0 0 3024 4032]
opaque
```

```
affine
1  0  0
0  -1  3024
extent=[0 0 4032 3024]
opaque
```

```
colormatch
Display P3_to_workingspace
extent=[0 0 4032 3024]
opaque
```

```
IOSurface 0x100f483d0(189) seed:1 YCC420f 601 alpha_one
extent=[0 0 4032 3024]
opaque
```

CIImage Quick Look

```
// load disparity image and apply orientation
var disparity = CIImage(
    contentsOf: url,
    options: [kCIImageAuxiliaryDisparity: true,
              kCIImageApplyOrientationProperty: true])

// cubic upsample by 2x
disparity = disparity.applyingFilter("CIBicubicScaleTransform", withInputParameters: [...])

// adjust disparity mask and blend foreground and background
let mask = disparity.applyingFilter("CIColorControls",
    withInputParameters: [kCIInputContrastKey: 2.0])
mask = mask.applyingFilter("CIColorClamp")
```

CIImage Quick Look

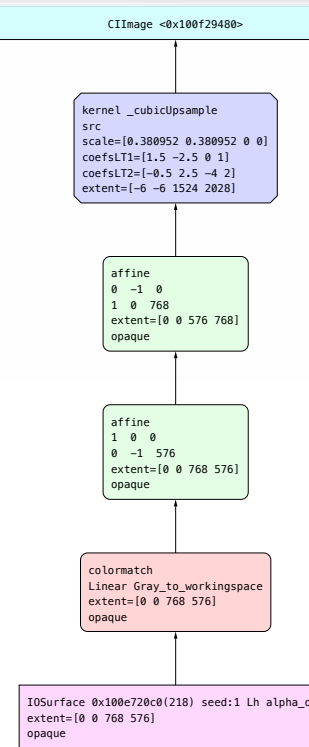
```
// load disparity image and apply orientation
var disparity = CIImage(
    contentsOf: url,
    options: [kCIImageAuxiliaryDisparity: true,
              kCIImageApplyOrientationProperty: true])

// cubic upsample by 2x
disparity = disparity.applyingFilter("CIBicubicScaleTransform", withInputParameters: [...])
▶ (CIImage) 0x00000000100f29480 👁
// adjust disparity mask and blend foreground and background
let mask = disparity.applyingFilter("CIColorControls",
    withInputParameters: [kCIInputContrastKey: 2.0])
mask = mask.applyingFilter("CIColorClamp")
```

CIImage Quick Look

```
// load disparity image and apply orientation
var disparity = CIImage(
  contentsOf: url,
  options: [kCIImageAuxiliaryDisparity: true,
            kCIImageApplyOrientationProperty: true])

// cubic upsample by 2x
disparity = disparity.applyingFilter("CIBicubicScaleTransform
  (CIImage) 0x0000000100f29480 🗙
// adjust disparity mask and blend foreground and background
let mask = disparity.applyingFilter("CIColorControls",
  withInputParameters: [kCIInputContrastKey: 2.0])
mask = mask.applyingFilter("CIColorClamp")
```



CIImage <0x100f29480>

kernel _cubicUpsample
src
scale=[0.380952 0.380952 0 0]
coefsLT1=[1.5 -2.5 0 1]
coefsLT2=[-0.5 2.5 -4 2]
extent=[-6 -6 1524 2028]

affine
0 -1 0
1 0 768
extent=[0 0 576 768]
opaque

affine
1 0 0
0 -1 576
extent=[0 0 768 576]
opaque

colormatch
Linear Gray_to_workingspace
extent=[0 0 768 576]
opaque

IOSurface 0x100e720c0(218) seed:1 Lh alpha_one
extent=[0 0 768 576]
opaque

CIImage <0x100f29480>

kernel _cubicUpsample
src
scale=[0.380952 0.380952 0 0]
coefsLT1=[1.5 -2.5 0 1]
coefsLT2=[-0.5 2.5 -4 2]
extent=[-6 -6 1524 2028]

affine
0 -1 0
1 0 768
extent=[0 0 576 768]
opaque

affine
1 0 0
0 -1 576
extent=[0 0 768 576]
opaque

colormatch
Linear Gray_to_workingspace
extent=[0 0 768 576]
opaque

IOSurface 0x100e720c0(428) seed:1 Lh alpha_one
extent=[0 0 768 576]
opaque

CIImage <0x100f29480>

```
kernel _cubicUpsample  
src  
scale=[0.380952 0.380952 0 0]  
coefsLT1=[1.5 -2.5 0 1]  
coefsLT2=[-0.5 2.5 -4 2]  
extent=[-6 -6 1524 2028]
```

```
affine  
0 -1 0  
1 0 768  
extent=[0 0 576 768]  
opaque
```

```
affine  
1 0 0  
0 -1 576  
extent=[0 0 768 576]  
opaque
```

```
colormatch  
Linear Gray_to_workingspace  
extent=[0 0 768 576]  
opaque
```

```
IOSurface 0x100e720c0(218) seed:1 Lh alpha_one  
extent=[0 0 768 576]  
opaque
```

CIImage <0x100f29480>

kernel _cubicUpsample
src
scale=[0.380952 0.380952 0 0]
coefsLT1=[1.5 -2.5 0 1]
coefsLT2=[-0.5 2.5 -4 2]
extent=[-6 -6 1524 2028]

affine
0 -1 0
1 0 768
extent=[0 0 576 768]
opaque

affine
1 0 0
0 -1 576
extent=[0 0 768 576]
opaque

colormatch
Linear Gray_to_workingspace
extent=[0 0 768 576]
opaque

IOSurface 0x100e720c0(218) seed:1 Lh alpha_one
extent=[0 0 768 576]
opaque

CIImage Quick Look

```
// adjustment for foreground
let fg = image.applyingFilter("CIPhotoEffectFade")

// adjustment for background
let bg = image.applyingFilter("CIPhotoEffectNoir")

// combine foreground and background using mask
let output = fg.applyingFilter("CIBlendWithMask",
    withInputParameters: [ kCIInputBackgroundImageKey: bg,
                          kCIInputMaskImageKey: mask])
```

CIImage Quick Look

```
// adjustment for foreground
let fg = image.applyingFilter("CIPhotoEffectFade")

// adjustment for background
let bg = image.applyingFilter("CIPhotoEffectNoir")

// combine foreground and background using mask
let output = fg.applyingFilter("CIBlendWithMask",
    ▶ (CIImage) 0x00000000100e341d0 👁 tBackgroundImageKey: bg,
    kCIInputMaskImageKey: mask])
```

CIImage Quick Look

```
// adjustment for foreground
```

```
let fg = image.applyingFilter("CIPhotoEffectFade")
```

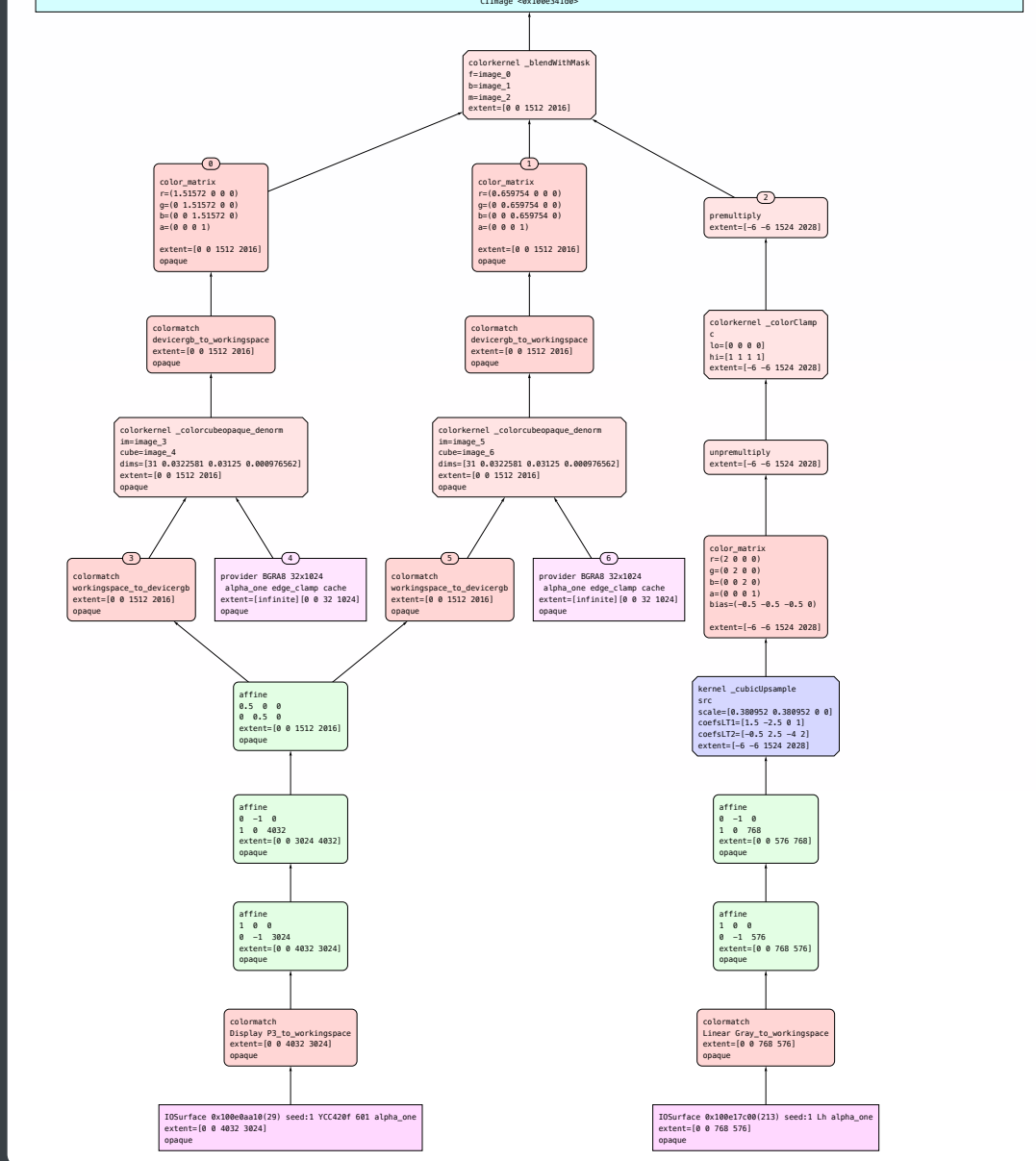
```
// adjustment for background
```

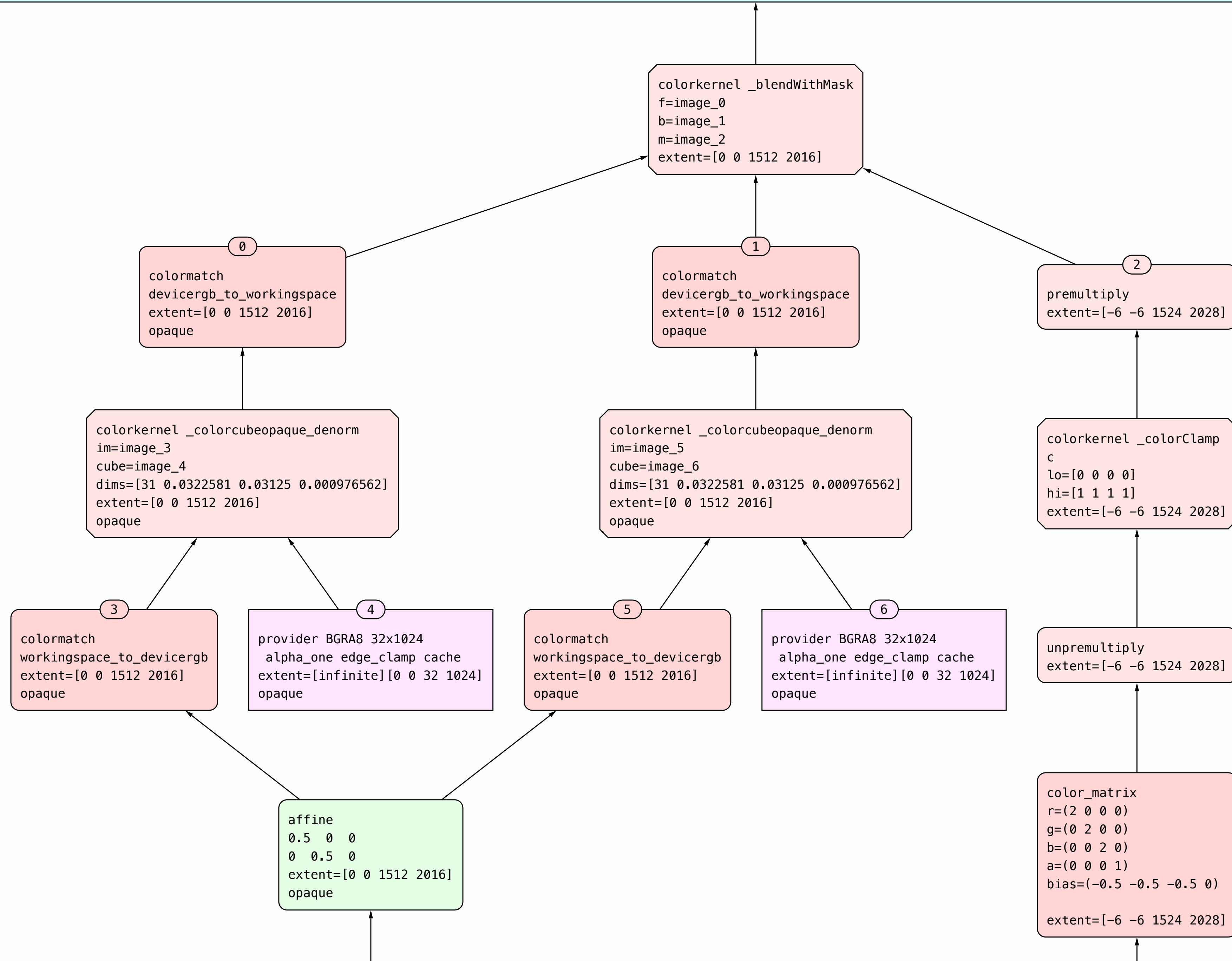
```
let bg = image.applyingFilter("CIPhotoEffectNoir")
```

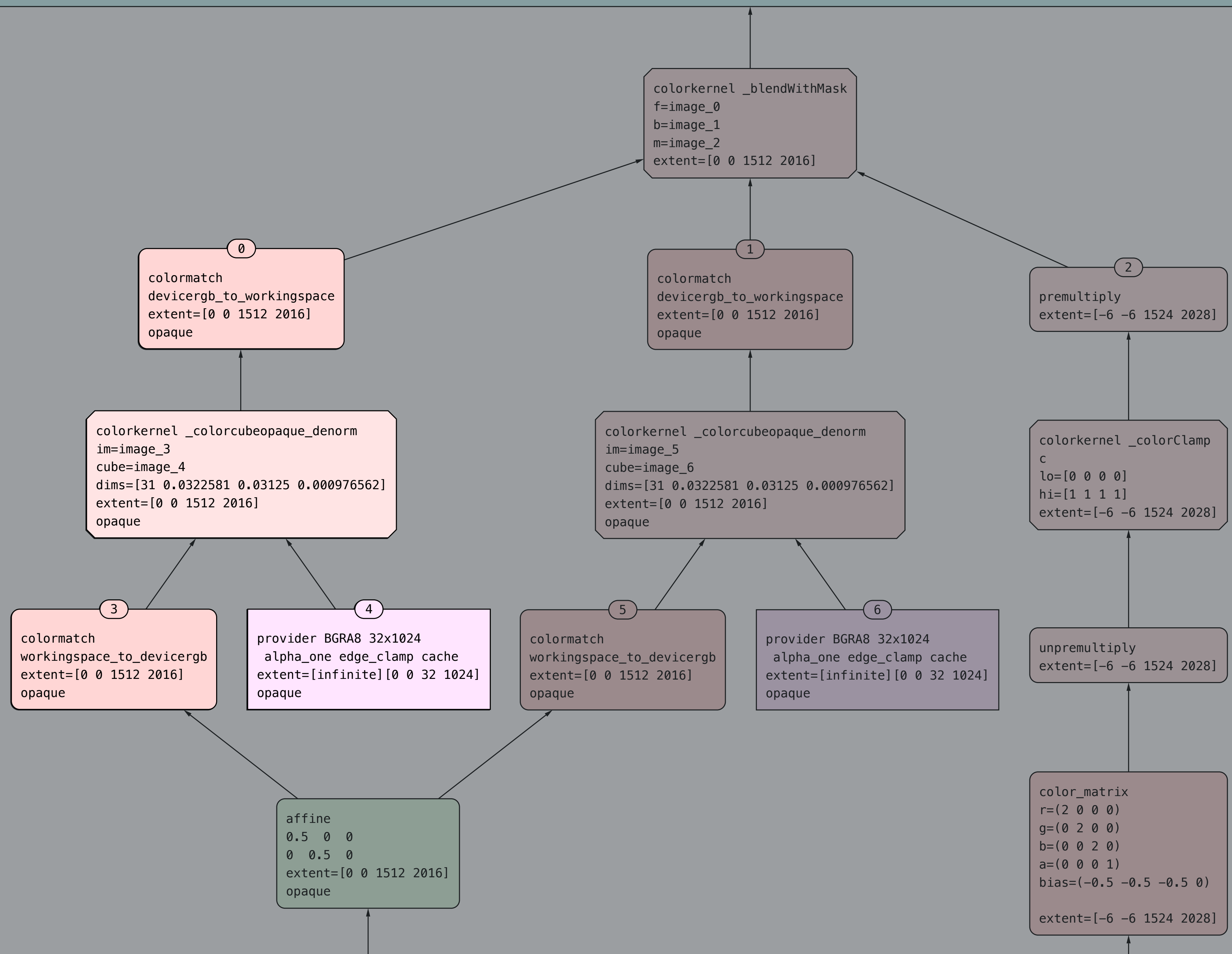
```
// combine foreground and background using mask
```

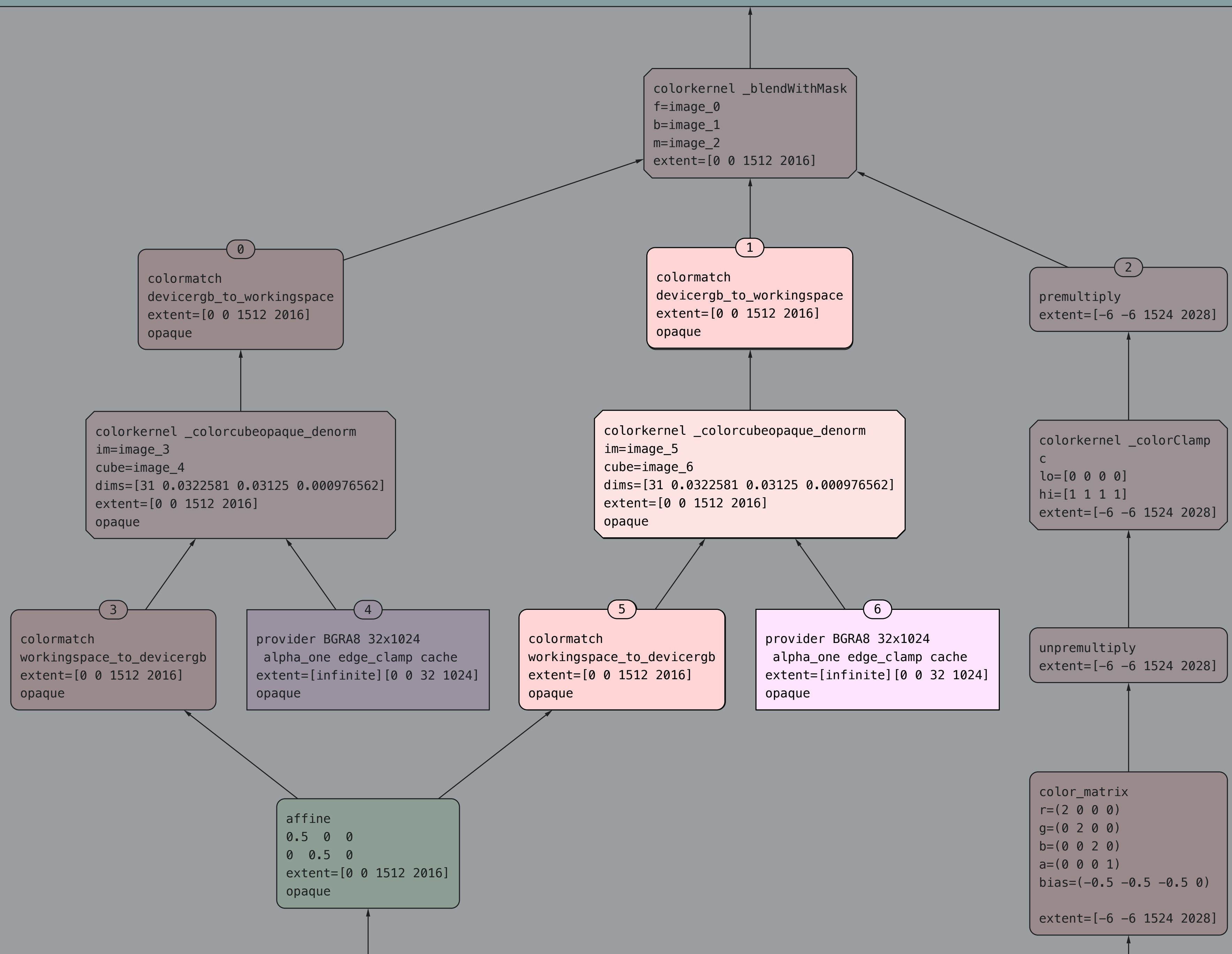
```
let output = fg.applyingFilter("CIBlendWithMask",
```

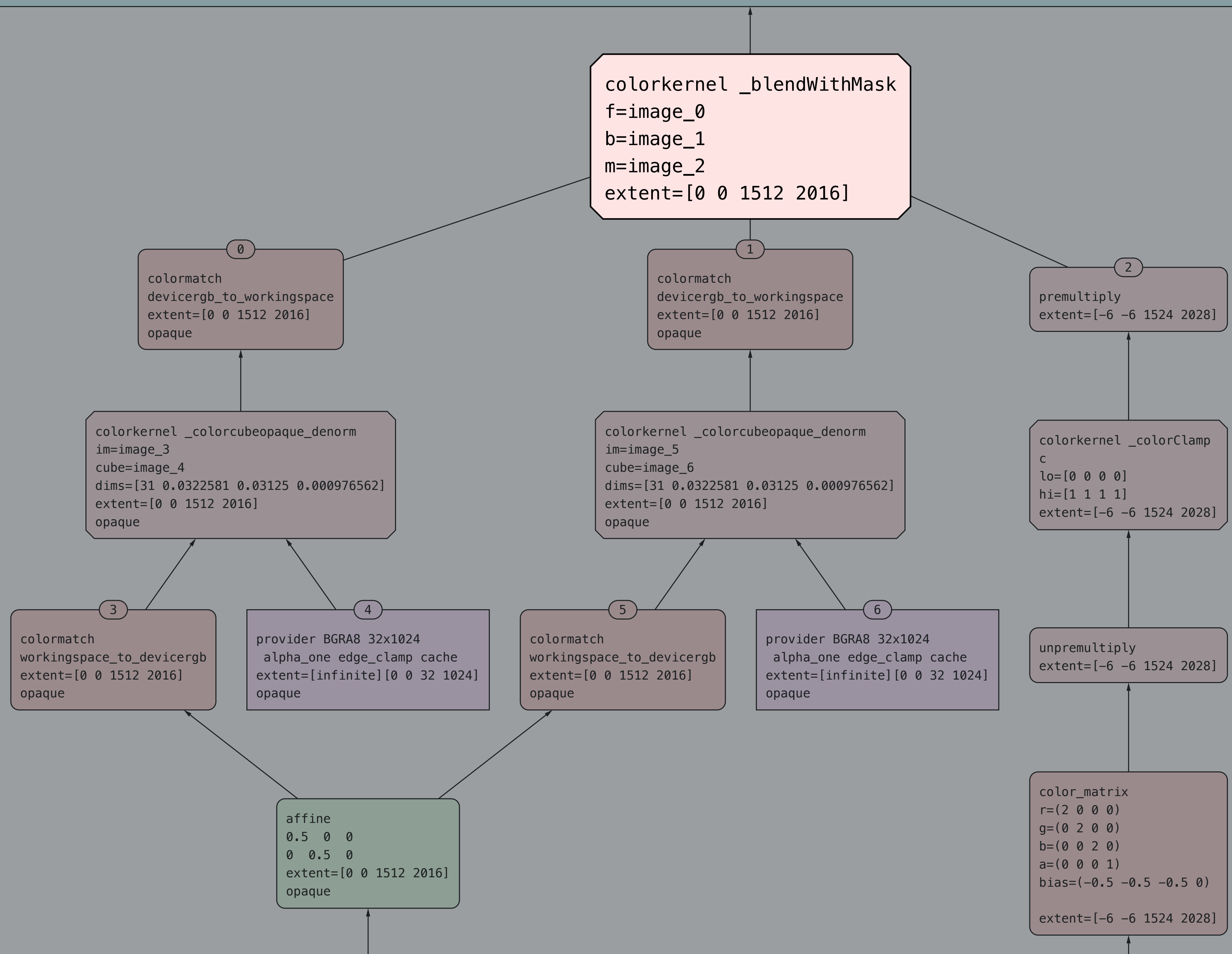
```
  (CIImage) 0x0000000100e341d0 tBackgroundImageKey: bg,  
  kCIInputMaskImageKey: mask])
```











CIRenderTask Quick Look

```
// render output to an IOSurface
let dest = CIRenderDestination(ioSurface: surface);



let context = self.context;

guard
    let task = try? context.startTask(toRender: output, to: dest),
    let info = try? task.waitUntilCompleted()
else {
    // handle render failure
}
```

CIRenderTask Quick Look

```
// render output to an IOSurface
let dest = CIRenderDestination(ioSurface: surface);

let context = self.context;

guard
    let task = try? context.startTask(toRender: output, to: dest),
    let  (CIRenderTask) 0x00000000102806b20  ed()
else {
    // handle render failure
}
```

CIRenderTask Quick Look

```
// render output to an IOSurface
```

```
let dest = CIGRenderDestination(ioSurface: surface);
```

```
let context = self.context;
```

```
guard
```

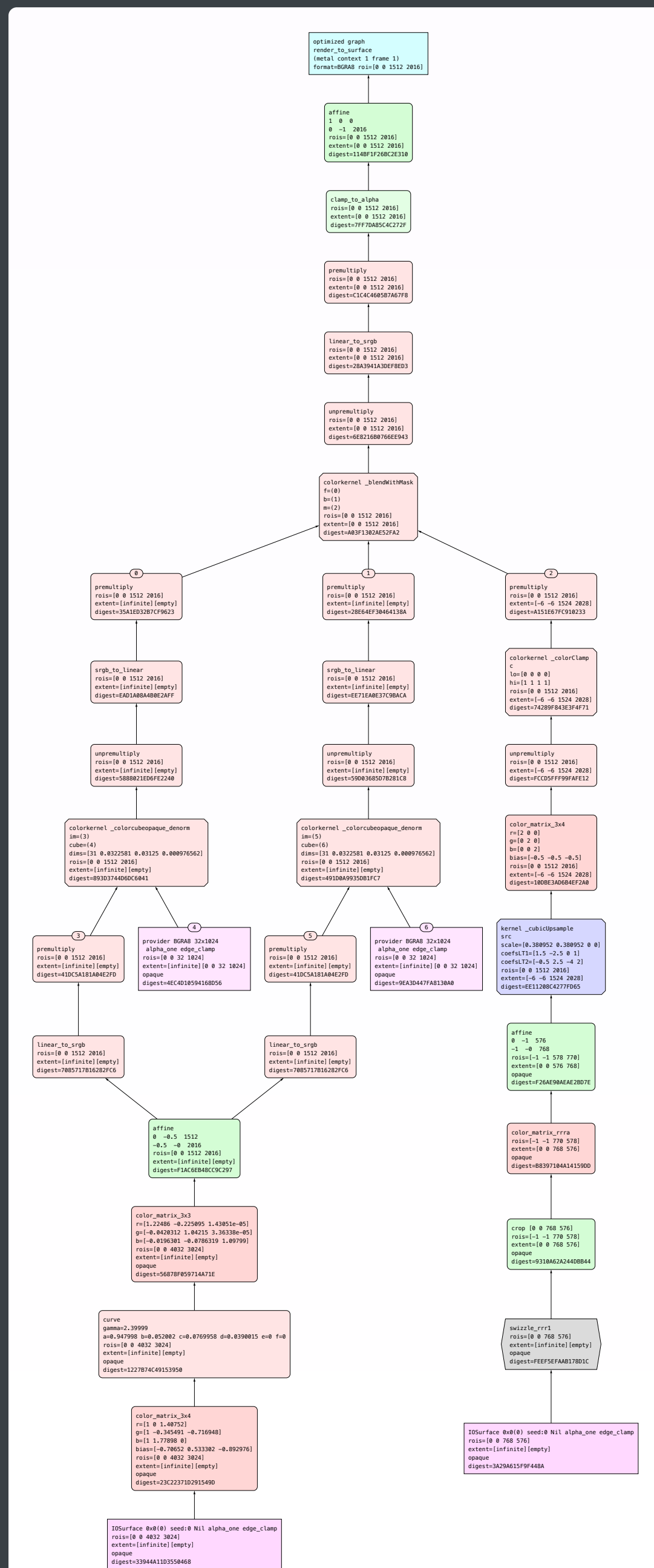
```
let task = try? context.startTask(toRender: output, to: dest),
```

```
let (CIGRenderTask) 0x00000000102806b20
```

```
else {
```

```
// handle render failure
```

```
}
```



```
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=7085717B16282FC6
```

```
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=7085717B16282FC6
```

```
affine
0 -0.5 1512
-0.5 -0 2016
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=F1AC6EB48CC9C297
```

```
color_matrix_3x3
r=[1.22486 -0.225095 1.43051e-05]
g=[-0.0420312 1.04215 3.36338e-05]
b=[-0.0196301 -0.0786319 1.09799]
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=56878F059714A71E
```

```
curve
gamma=2.39999
a=0.947998 b=0.052002 c=0.0769958 d=0.0390015 e=0 f=0
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=1227B74C49153950
```

```
color_matrix_3x4
r=[1 0 1.40752]
g=[1 -0.345491 -0.716948]
```

```
rois=[-1 -1 578 770]
extent=[0 0 576 768]
opaque
digest=F26AE90AEAE2BD7E
```

```
color_matrix_rrra
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=B8397104A14159DD
```

```
crop [0 0 768 576]
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=9310A62A244DBB44
```

```
swizzle_rrr1
rois=[0 0 768 576]
extent=[infinite][empty]
opaque
digest=FEEF5EFAAB178D1C
```

```
IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp
```



```
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=7085717B16282FC6
```

```
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=7085717B16282FC6
```

```
affine
0 -0.5 1512
-0.5 -0 2016
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=F1AC6EB48CC9C297
```

```
color_matrix_3x3
r=[1.22486 -0.225095 1.43051e-05]
g=[-0.0420312 1.04215 3.36338e-05]
b=[-0.0196301 -0.0786319 1.09799]
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=CFB3EC1C5985E7E0
```

```
curve
gamma=2.39999
a=0.947998 b=0.052002 c=0.0769958 d=0.0390015 e=0 f=0
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=6BDBF61A00035E59
```

```
color_matrix_3x4
r=[1 0 1.40752]
g=[1 -0.345491 -0.716948]
```

```
rois=[-1 -1 578 770]
extent=[0 0 576 768]
opaque
digest=F26AE90AEAE2BD7E
```

```
color_matrix_rrra
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=B8397104A14159DD
```

```
crop [0 0 768 576]
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=9310A62A244DBB44
```

```
swizzle_rrr1
rois=[0 0 768 576]
extent=[infinite][empty]
opaque
digest=FEEF5EFAAB178D1C
```

```
IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp
```


rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=7085717B16282FC6

rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=7085717B16282FC6

affine
0 -0.5 1512
-0.5 -0 2016
rois=[0 0 1512 2016]
extent=[infinite][empty]
digest=F1AC6EB48CC9C297

color_matrix_3x3
r=[1.22486 -0.225095 1.43051e-05]
g=[-0.0420312 1.04215 3.36338e-05]
b=[-0.0196301 -0.0786319 1.09799]
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=56878F059714A71E

curve
gamma=2.39999
a=0.947998 b=0.052002 c=0.0769958 d=0.0390015 e=0 f=0
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=1227B74C49153950

color_matrix_3x4
r=[1 0 1.40752]
g=[1 -0.345491 -0.716948]

rois=[-1 -1 578 770]
extent=[0 0 576 768]
opaque
digest=F26AE90AEAE2BD7E

color_matrix_rrra
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=B8397104A14159DD

crop [0 0 768 576]
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=9310A62A244DBB44

swizzle_rrr1
rois=[0 0 768 576]
extent=[infinite][empty]
opaque
digest=FEEF5EFAAB178D1C

IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp

```
optimized graph
render_to_surface
(metal context 1 frame 1)
format=BGRA8 roi=[0 0 1512 2016]
```

```
affine
1 0 0
0 -1 2016
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=114BF1F26BC2E310
```

```
clamp_to_alpha
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=7FF7DA85C4C272F
```

```
premultiply
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=C1C4C4605B7A67F8
```

```
linear_to_srgb
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=28A3941A3DEF8ED3
```

```
unpremultiply
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=6E8216B0766EE943
```

```
optimized graph
render_to_surface
(metal context 1 frame 1)
format=BGRA8 roi=[0 0 1512 2016]
```

```
affine
1 0 0
0 -1 2016
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=114BF1F26BC2E310
```

```
clamp_to_alpha
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=7FF7DA85C4C272F
```

```
premultiply
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=C1C4C4605B7A67F8
```

```
linear_to_srgb
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=28A3941A3DEF8ED3
```

```
unpremultiply
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=6E8216B0766EE943
```

CIRenderInfo Quick Look

```
// render output to an IOSurface
let dest = CIRenderDestination(ioSurface: surface);

let context = self.context;

guard
    let task = try? context.startTask(toRender: output, to: dest),
    let info = try? task.waitUntilCompleted()
else {
    // handle render failure
}
```

CIRendererInfo Quick Look

```
// render output to an IOSurface
let dest = CIRenderDestination(ioSurface: surface);

let context = self.context;

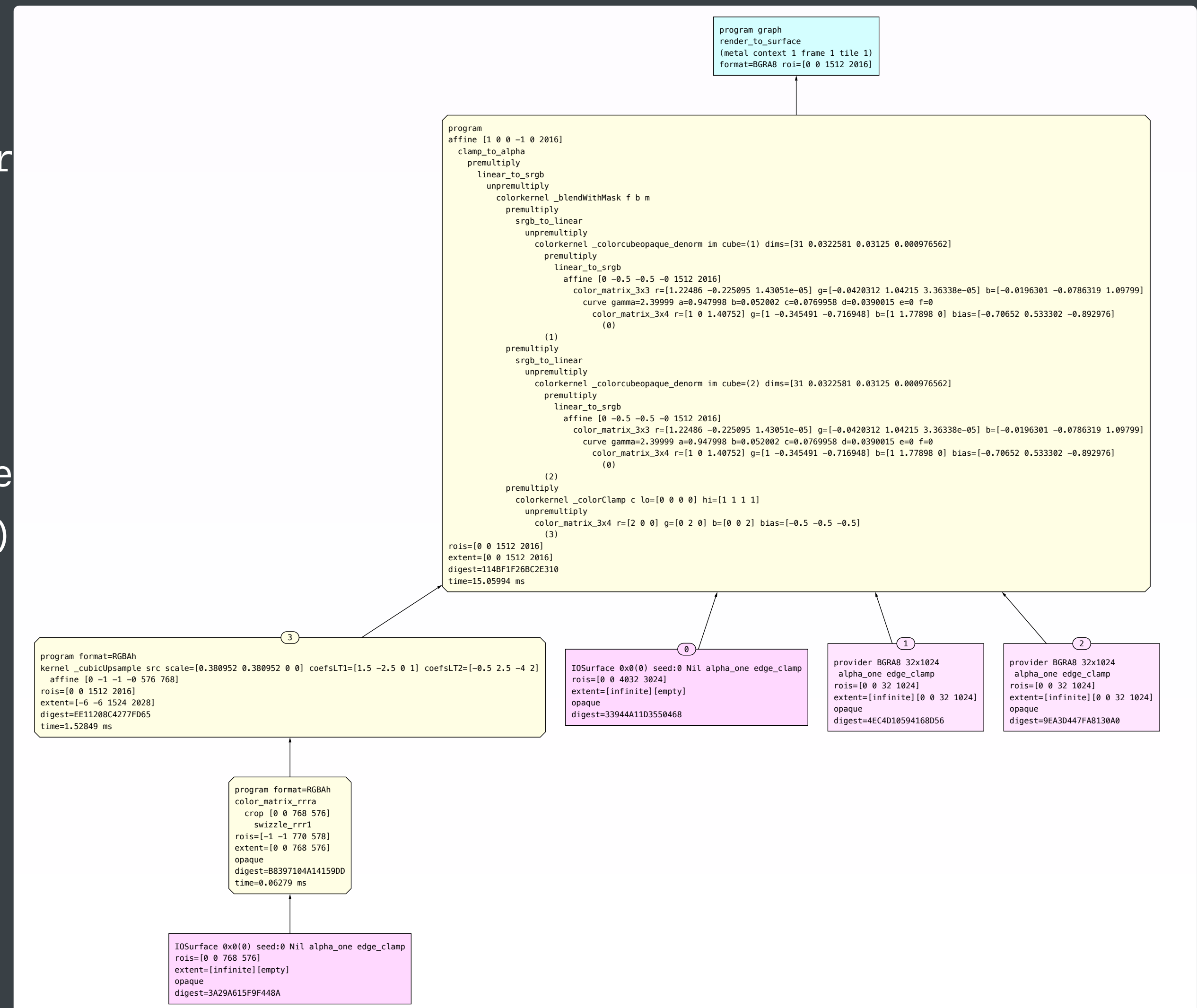
guard
    let task = try? context.startTask(toRender: output, to: dest),
    let info = try? task.waitUntilCompleted()
else { ▶ (CIRendererInfo) 0x00000000100e082b0 👁
    // handle render failure
}
```

CIRenderInfo Quick Look

```
// render output to an IOSurface
let dest = CIRenderDestination(ioSurface: sur

let context = self.context;

guard
    let task = try? context.startTask(toRende
    let info = try? task.waitUntilCompleted()
else {
    // (CIRenderInfo) 0x00000000100e082b0
    // handle render failure
}
```



```
srgb_to_linear
unpremultiply
colorkernel_colorcubeopaque_denorm im cube=(2) dims=[31 0.0322581 0.03125 0.000976562]
premultiply
linear_to_srgb
affine [0 -0.5 -0.5 -0 1512 2016]
color_matrix_3x3 r=[1.22486 -0.225095 1.43051e-05] g=[-0.0420312 1.04215 3.36338e-05] b=[-0.0196301 -0.0786319 1.09799]
curve gamma=2.39999 a=0.947998 b=0.052002 c=0.0769958 d=0.0390015 e=0 f=0
color_matrix_3x4 r=[1 0 1.40752] g=[1 -0.345491 -0.716948] b=[1 1.77898 0] bias=[-0.70652 0.533302 -0.892976]
(0)
(2)
premultiply
colorkernel_colorClamp c lo=[0 0 0 0] hi=[1 1 1 1]
unpremultiply
color_matrix_3x4 r=[2 0 0] g=[0 2 0] b=[0 0 2] bias=[-0.5 -0.5 -0.5]
(3)
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=114BF1F26BC2E310
time=15.05994 ms
```

3

```
program format=RGBA
kernel_cubicUpsample src scale=[0.380952 0.380952 0 0] coefsLT1=[1.5 -2.5 0 1] coefsLT2=[-0.5 2.5 -4 2]
affine [0 -1 -1 -0 576 768]
rois=[0 0 1512 2016]
extent=[-6 -6 1524 2028]
digest=EE11208C4277FD65
time=1.52849 ms
```

0

```
IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=33944A11D3550468
```

1

```
provider BGRA8 32x1024
alpha_one edge_clamp
rois=[0 0 32 1024]
extent=[infinite][0 0 32 1024]
opaque
digest=4EC4D10594168D56
```

2

```
provider BGRA8 32x1024
alpha_one edge_clamp
rois=[0 0 32 1024]
extent=[infinite][0 0 32 1024]
opaque
digest=9EA3D447FA8130A0
```

```
program format=RGBA
color_matrix_rrra
crop [0 0 768 576]
swizzle_rrr1
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=B8397104A14159DD
time=0.06279 ms
```

```
IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp
rois=[0 0 768 576]
extent=[infinite][empty]
opaque
digest=3A29A615F9F448A
```

```
srgb_to_linear
unpremultiply
colorkernel_colorcubeopaque_denorm im cube=(2) dims=[31 0.0322581 0.03125 0.000976562]
premultiply
linear_to_srgb
affine [0 -0.5 -0.5 -0 1512 2016]
color_matrix_3x3 r=[1.22486 -0.225095 1.43051e-05] g=[-0.0420312 1.04215 3.36338e-05] b=[-0.0196301 -0.0786319 1.09799]
curve gamma=2.39999 a=0.947998 b=0.052002 c=0.0769958 d=0.0390015 e=0 f=0
color_matrix_3x4 r=[1 0 1.40752] g=[1 -0.345491 -0.716948] b=[1 1.77898 0] bias=[-0.70652 0.533302 -0.892976]
(0)
(2)
premultiply
colorkernel_colorClamp c lo=[0 0 0 0] hi=[1 1 1 1]
unpremultiply
color_matrix_3x4 r=[2 0 0] g=[0 2 0] b=[0 0 2] bias=[-0.5 -0.5 -0.5]
(3)
rois=[0 0 1512 2016]
extent=[0 0 1512 2016]
digest=114BF1F26BC2E310
time=15.05994 ms
```

3

```
program format=RGBA
kernel_cubicUpsample src scale=[0.380952 0.380952 0 0] coefsLT1=[1.5 -2.5 0 1] coefsLT2=[-0.5 2.5 -4 2]
affine [0 -1 -1 -0 576 768]
rois=[0 0 1512 2016]
extent=[-6 -6 1524 2028]
digest=EE11208C4277FD65
time=1.52849 ms
```

0

```
IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp
rois=[0 0 4032 3024]
extent=[infinite][empty]
opaque
digest=33944A11D3550468
```

1

```
provider BGRA8 32x1024
alpha_one edge_clamp
rois=[0 0 32 1024]
extent=[infinite][0 0 32 1024]
opaque
digest=4EC4D10594168D56
```

2

```
provider BGRA8 32x1024
alpha_one edge_clamp
rois=[0 0 32 1024]
extent=[infinite][0 0 32 1024]
opaque
digest=9EA3D447FA8130A0
```

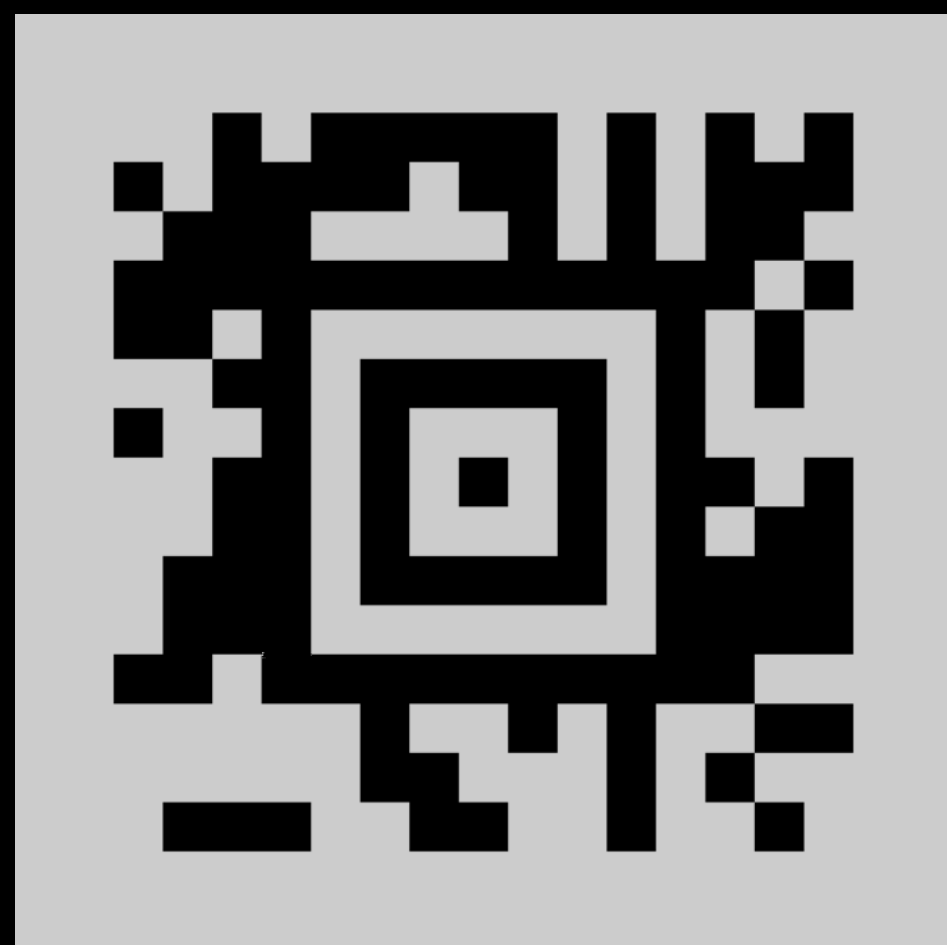
```
program format=RGBA
color_matrix_rrra
crop [0 0 768 576]
swizzle_rrr1
rois=[-1 -1 770 578]
extent=[0 0 768 576]
opaque
digest=B8397104A14159DD
time=0.06279 ms
```

```
IOSurface 0x0(0) seed:0 Nil alpha_one edge_clamp
rois=[0 0 768 576]
extent=[infinite][empty]
opaque
digest=3A29A615F9F448A
```


CIBarcodeDescriptor API

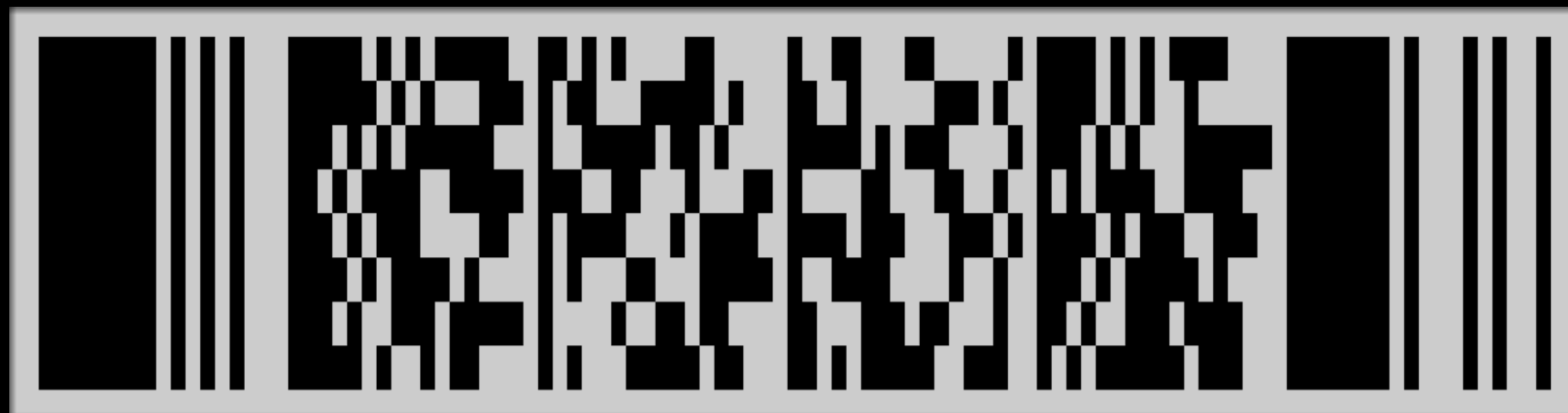
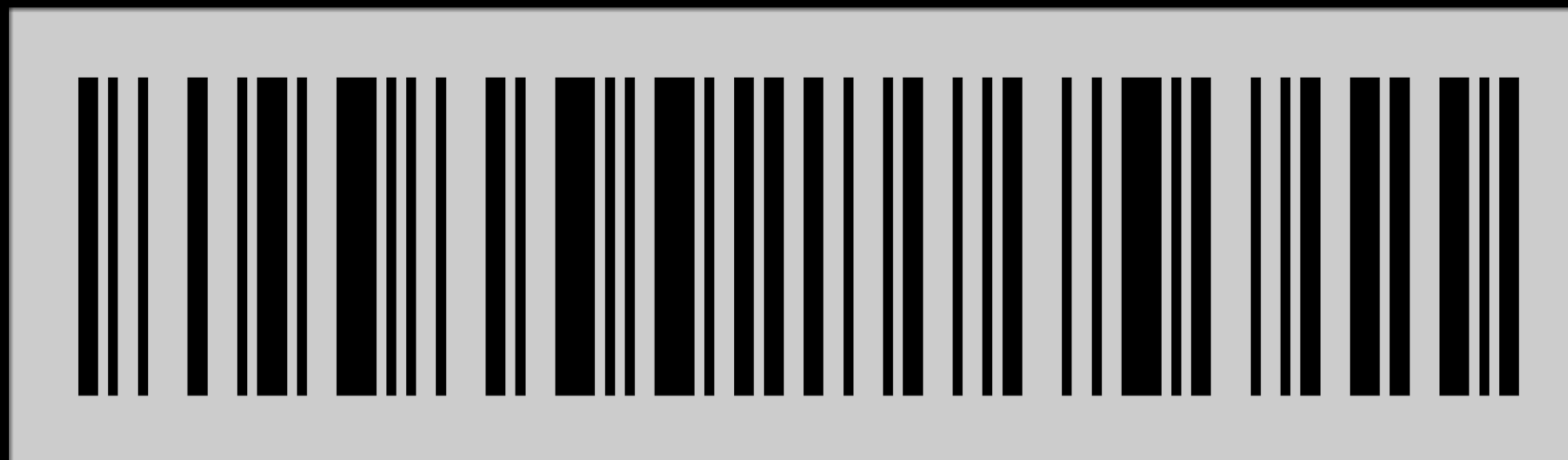
Framework Barcode Support

Several frameworks support various barcodes types



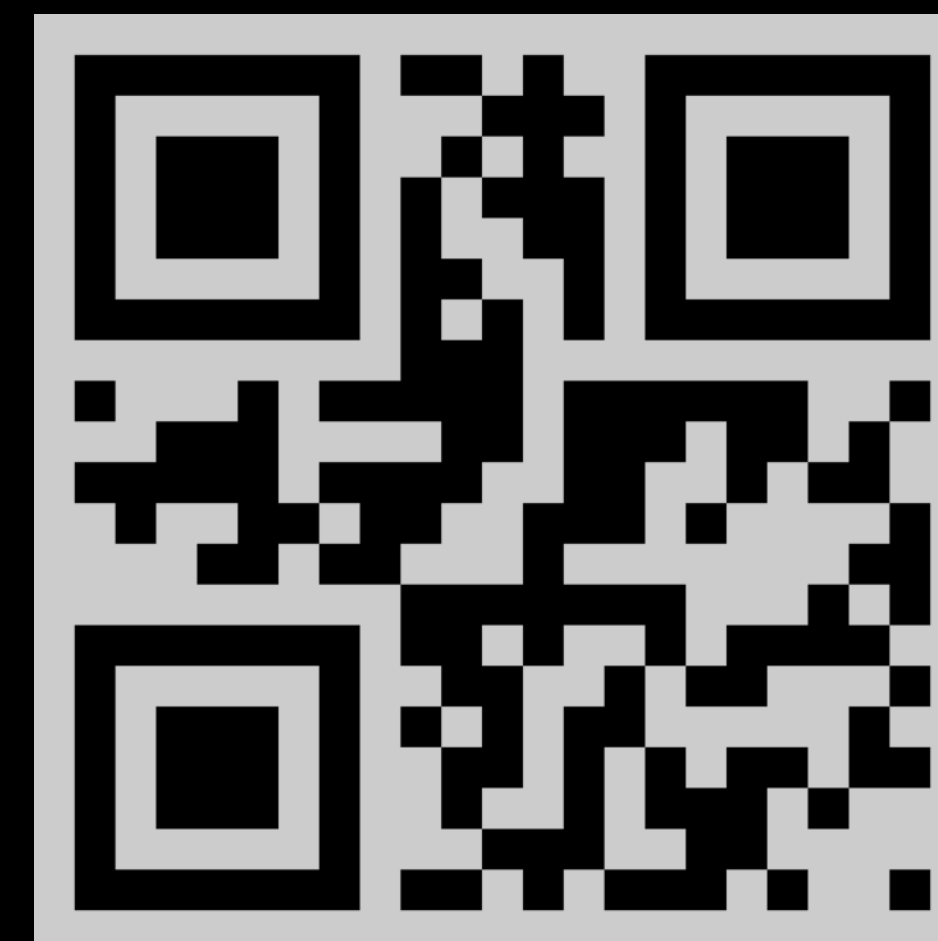
Aztec

Code128



PDF417

QRCode



Framework Barcode Support

Several frameworks support various barcodes types

AVFoundation.framework

Vision.framework

CoreImage.framework

Detection of barcodes
during capture

Detection of barcodes
after capture

Rendering barcodes

Framework Barcode Support

Several frameworks support various barcodes types

AVFoundation.framework

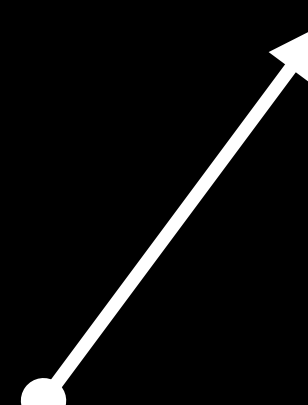
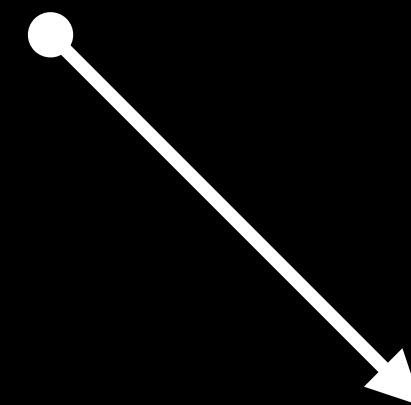
Vision.framework

CoreImage.framework

Detection of barcodes
during capture

Detection of barcodes
after capture

Rendering barcodes



CIBarcodeDescriptor

CIBarcodeDescriptor Objects Contain

The `errorCorrectedPayload` of the barcode

CIBarcodeDescriptor Objects Contain

The `errorCorrectedPayload` of the barcode

Additional code-specific properties such as

CIBarcodeDescriptor Objects Contain

The `errorCorrectedPayload` of the barcode

Additional code-specific properties such as

- Aztec's `layerCount`

CIBarcodeDescriptor Objects Contain

The `errorCorrectedPayload` of the barcode

Additional code-specific properties such as

- Aztec's `layerCount`
- QRCode's `maskPattern`


```
// Get a CIBarcodeDescriptor from AVFoundation.framework

class MyMetadataOutputObjectsDelegate: NSObject, AVCaptureMetadataOutputObjectsDelegate
{
    func metadataOutput(_ output: AVCaptureMetadataOutput,
                        didOutput metadataObjects: [AVMetadataObject],
                        from connection: AVCaptureConnection) {
        if let mrc = metadataObjects.first as? AVMetadataMachineReadableCodeObject,
            let descriptor = mrc.descriptor {
            print(descriptor)
        }
    }
}
}
```

```
// Get a CIBarcodeDescriptor from AVFoundation.framework

class MyMetadataOutputObjectsDelegate: NSObject, AVCaptureMetadataOutputObjectsDelegate
{
    func metadataOutput(_ output: AVCaptureMetadataOutput,
                        didOutput metadataObjects: [AVMetadataObject],
                        from connection: AVCaptureConnection) {
        if let mrc = metadataObjects.first as? AVMetadataMachineReadableCodeObject,
            let descriptor = mrc.descriptor {
            print(descriptor)
        }
    }
}
}
```

```
// Detect a CIBarcodeDescriptor using Vision.framework

func descriptorFromImage(_ image: CIImage) -> CIBarcodeDescriptor?
{
    // Create the request and request handler
    let requestHandler = VNImageRequestHandler(ciImage: image, options: [:])
    let request = VNDetectBarcodesRequest();

    // Send the request to the handler
    try? requestHandler.perform([request])

    // Get the observation
    let firstResult = request.results?.first
    return firstResult?.barcodeDescriptor
}
```

```
// Detect a CIBarcodeDescriptor using Vision.framework

func descriptorFromImage(_ image: CIImage) -> CIBarcodeDescriptor?
{
    // Create the request and request handler
    let requestHandler = VNImageRequestHandler(ciImage: image, options: [:])
    let request = VNDetectBarcodesRequest();

    // Send the request to the handler
    try? requestHandler.perform([request])

    // Get the observation
    let firstResult = request.results?.first
    return firstResult?.barcodeDescriptor
}
```

```
// Detect a CIBarcodeDescriptor using Vision.framework

func descriptorFromImage(_ image: CIImage) -> CIBarcodeDescriptor?
{
    // Create the request and request handler
    let requestHandler = VNImageRequestHandler(ciImage: image, options: [:])
    let request = VNDetectBarcodesRequest();

    // Send the request to the handler
    try? requestHandler.perform([request])

    // Get the observation
    let firstResult = request.results?.first
    return firstResult?.barcodeDescriptor
}
```

```
// Detect a CIBarcodeDescriptor using Vision.framework

func descriptorFromImage(_ image: CIImage) -> CIBarcodeDescriptor?
{
    // Create the request and request handler
    let requestHandler = VNImageRequestHandler(ciImage: image, options: [:])
    let request = VNDetectBarcodesRequest();

    // Send the request to the handler
    try? requestHandler.perform([request])

    // Get the observation
    let firstResult = request.results?.first
    return firstResult?.barcodeDescriptor
}
```

```
// Create an image for a CIBarcodeDescriptor using CoreImage.framework

func imageFromBarcodeCodeDescriptor(_ descriptor: CIBarcodeDescriptor) -> CIImage?
{
    return CIFilter(name: "CIBarcodeGenerator",
                    withInputParameters: ["inputBarcodeDescriptor" : descriptor])
        ?.outputImage
}
```

```
// Create an image for a CIBarcodeDescriptor using CoreImage.framework

func imageFromBarcodeCodeDescriptor(_ descriptor: CIBarcodeDescriptor) -> CIImage?
{
    return CIFilter(name: "CIBarcodeGenerator",
                    withInputParameters: ["inputBarcodeDescriptor" : descriptor])
        ?.outputImage
}
```



```
// Create an image for a CIBarcodeDescriptor using CoreImage.framework

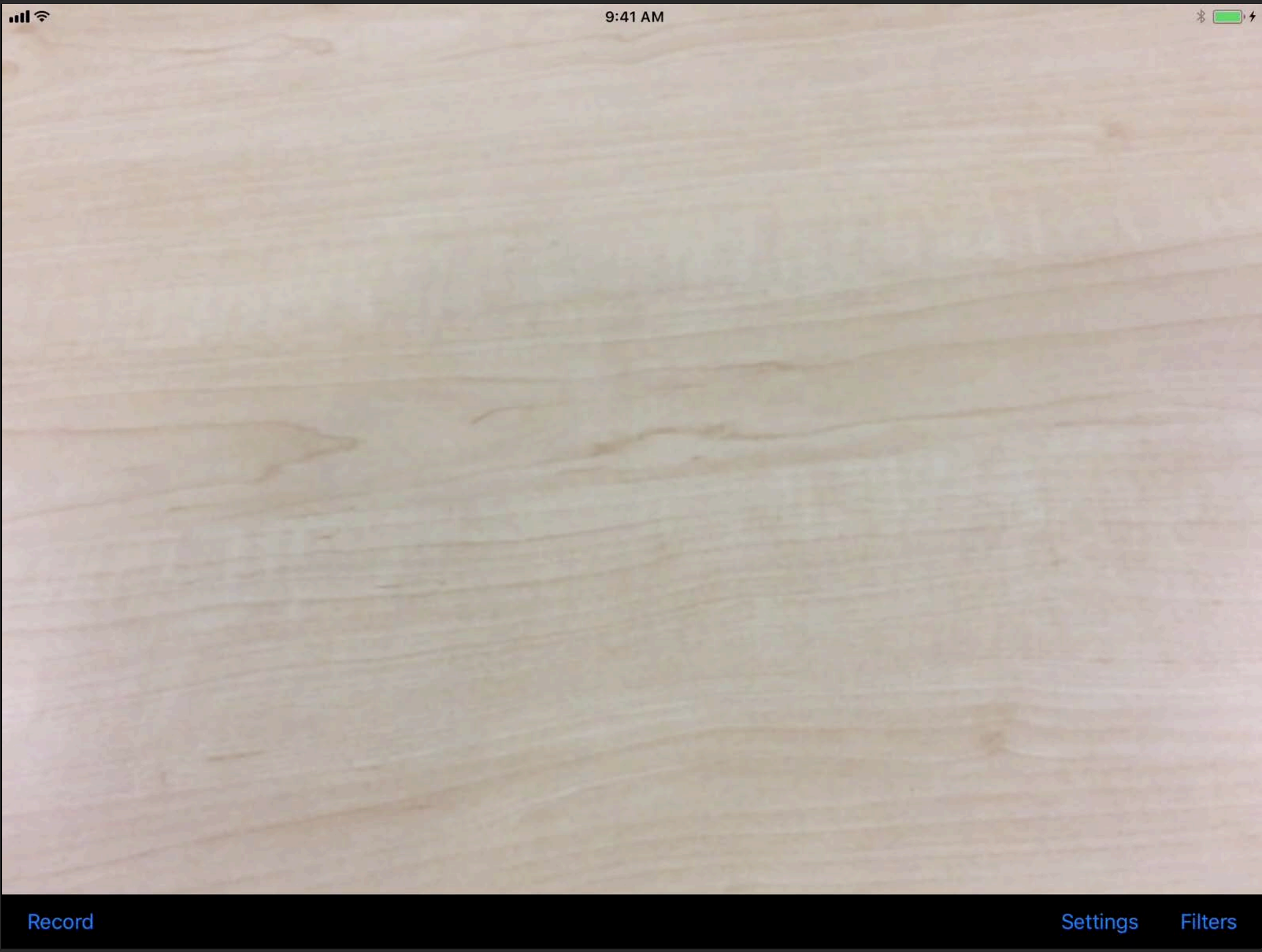
func imageFromBarcodeCodeDescriptor(_ descriptor: CIBarcodeDescriptor) -> CIImage?
{
    return CIFilter(name: "CIBarcodeGenerator",
                    withInputParameters: ["inputBarcodeDescriptor" : descriptor])
        ?.outputImage
}
```

```
// Create an image for a CIBarcodeDescriptor using CoreImage.framework

func imageFromBarcodeCodeDescriptor(_ descriptor: CIBarcodeDescriptor) -> CIImage?
{
    return CIFilter(name: "CIBarcodeGenerator",
                    withInputParameters: ["inputBarcodeDescriptor" : descriptor])
        ?.outputImage
}
```

Demo

Detection and Re-Generation of a QRCode



Using Core Image with Vision

Using Core Image with Vision

Using Core Image with Vision

Core Image can prepare images before being passed to Vision

- For example: Cropping, Orienting, Converting to grayscale

Using Core Image with Vision

Core Image can prepare images before being passed to Vision

- For example: Cropping, Orienting, Converting to grayscale

Vision can gather information before processing with Core Image

- For example: Feature detection can guide which CIFilters to apply

Photo from Video with Removal of Unwanted Objects

AVFoundation: Get the frames out of a short video

Photo from Video with Removal of Unwanted Objects

AVFoundation: Get the frames out of a short video



Photo from Video with Removal of Unwanted Objects

Vision: Determine homography matrices to align frames



Photo from Video with Removal of Unwanted Objects

Vision: Determine homography matrices to align frames

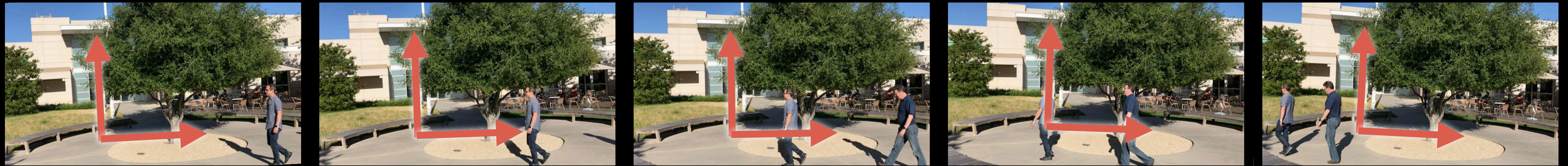


Photo from Video with Removal of Unwanted Objects

Core Image: Align each video frame

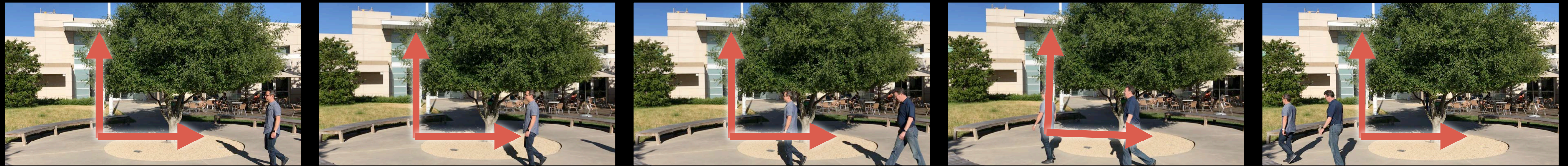


Photo from Video with Removal of Unwanted Objects

Core Image: Make a photo from the median of the aligned frames



Photo from Video with Removal of Unwanted Objects

Core Image: Make a photo from the median of the aligned frames



Photo from Video with Removal of Unwanted Objects

Core Image: Make a photo from the median of the aligned frames

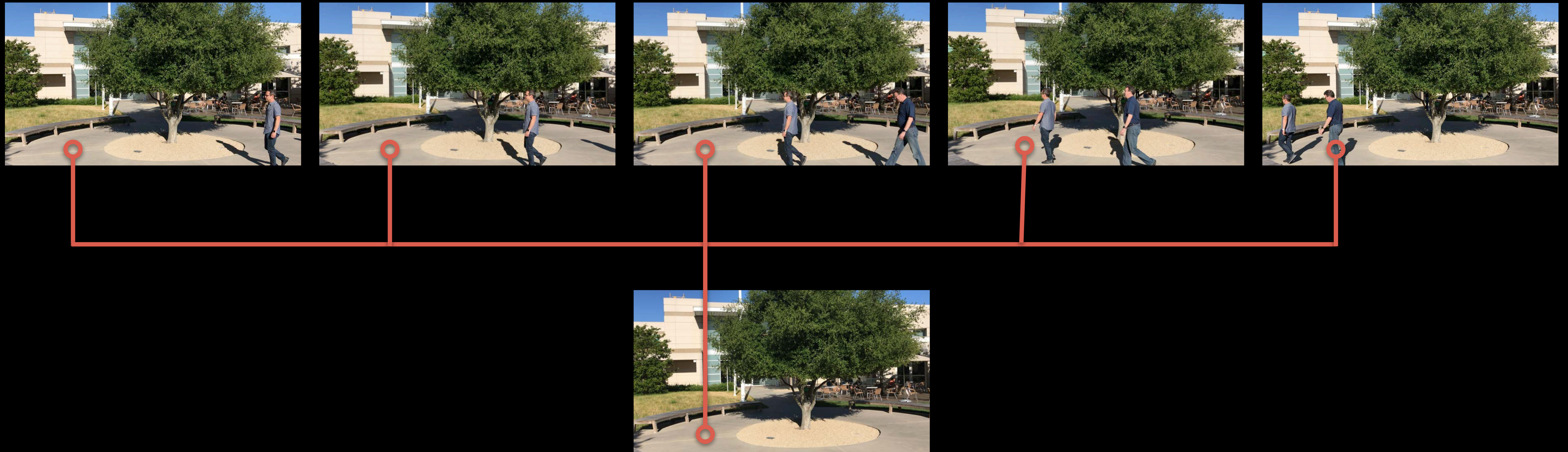
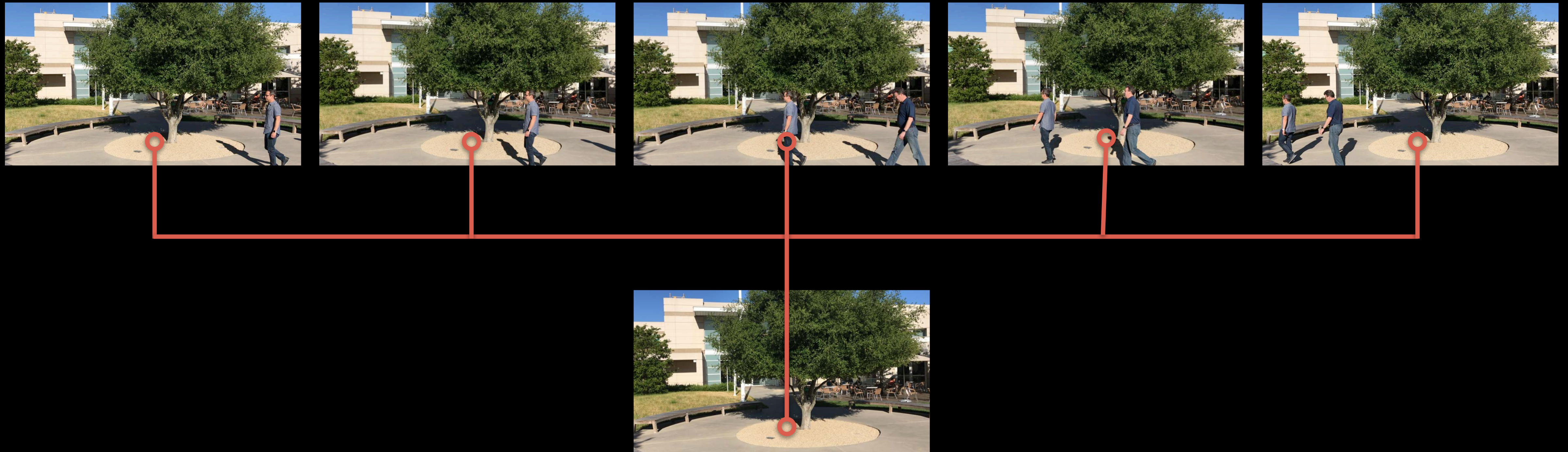


Photo from Video with Removal of Unwanted Objects

Core Image: Make a photo from the median of the aligned frames




```
// Use Vision to find homographic registration and pass it to CI

func homographicTransform(from image: CIImage, to reference: CIImage) -> matrix_float3x3? {
    // Create the request and request handler
    let request = VNHomographicImageRegistrationRequest(targetedCIImage: image);
    let requestHandler = VNImageRequestHandler(ciImage: reference, options: [:]);

    // Send the request to the handler
    try? requestHandler.perform([request]);

    // Get the observation
    guard let results = request.results,
          let observation = results.first as? VNImageHomographicAlignmentObservation
    else {
        return nil
    }
    return observation.warpTransform
}
```

```
// Use Vision to find homographic registration and pass it to CI

func homographicTransform(from image: CIImage, to reference: CIImage) -> matrix_float3x3? {
    // Create the request and request handler
    let request = VNHomographicImageRegistrationRequest(targetedCIImage: image);
    let requestHandler = VNImageRequestHandler(ciImage: reference, options: [:]);

    // Send the request to the handler
    try? requestHandler.perform([request]);

    // Get the observation
    guard let results = request.results,
          let observation = results.first as? VNImageHomographicAlignmentObservation
    else {
        return nil
    }
    return observation.warpTransform
}
```

```
// Use Vision to find homographic registration and pass it to CI

func homographicTransform(from image: CIImage, to reference: CIImage) -> matrix_float3x3? {
    // Create the request and request handler
    let request = VNHomographicImageRegistrationRequest(targetedCIImage: image);
    let requestHandler = VNImageRequestHandler(ciImage: reference, options: [:]);

    // Send the request to the handler
    try? requestHandler.perform([request]);

    // Get the observation
    guard let results = request.results,
          let observation = results.first as? VNImageHomographicAlignmentObservation
    else {
        return nil
    }
    return observation.warpTransform
}
```

```
// Use Vision to find homographic registration and pass it to CI

func homographicTransform(from image: CIImage, to reference: CIImage) -> matrix_float3x3? {
    // Create the request and request handler
    let request = VNHomographicImageRegistrationRequest(targetedCIImage: image);
    let requestHandler = VNImageRequestHandler(ciImage: reference, options: [:]);

    // Send the request to the handler
    try? requestHandler.perform([request]);

    // Get the observation
    guard let results = request.results,
          let observation = results.first as? VNImageHomographicAlignmentObservation
    else {
        return nil
    }
    return observation.warpTransform
}
```

```
// Use Vision to find homographic registration and pass it to CI

func homographicTransform(from image: CIImage, to reference: CIImage) -> matrix_float3x3? {
    // Create the request and request handler
    let request = VNHomographicImageRegistrationRequest(targetedCIImage: image);
    let requestHandler = VNImageRequestHandler(ciImage: reference, options: [:]);

    // Send the request to the handler
    try? requestHandler.perform([request]);

    // Get the observation
    guard let results = request.results,
          let observation = results.first as? VNImageHomographicAlignmentObservation
    else {
        return nil
    }

    return observation.warpTransform
}
```

```
// Core Image Metal kernel to apply a homography matrix
```

```
float2 warpHomography(float3x3 h, destination dest)
```

```
{
```

```
    float3 homogeneousDestCoord = float3(dest.coord(), 1.0);
```

```
    float3 homogeneousSrcCoord = h * homogeneousDestCoord;
```

```
    float2 srcCoord = homogeneousSrcCoord.xy / max(homogeneousSrcCoord.z, 0.000001);
```

```
    return srcCoord;
```

```
}
```

```
// Core Image Metal kernel to apply a homography matrix
```

```
float2 warpHomography(float3x3 h, destination dest)
```

```
{
```

```
    float3 homogeneousDestCoord = float3(dest.coord(), 1.0);
```

```
    float3 homogeneousSrcCoord = h * homogeneousDestCoord;
```

```
    float2 srcCoord = homogeneousSrcCoord.xy / max(homogeneousSrcCoord.z, 0.000001);
```

```
    return srcCoord;
```

```
}
```



```
// Core Image Metal kernel to apply a homography matrix
```

```
float2 warpHomography(float3x3 h, destination dest)
```

```
{
```

```
    float3 homogeneousDestCoord = float3(dest.coord(), 1.0);
```

```
    float3 homogeneousSrcCoord = h * homogeneousDestCoord;
```

```
    float2 srcCoord = homogeneousSrcCoord.xy / max(homogeneousSrcCoord.z, 0.000001);
```

```
    return srcCoord;
```

```
}
```

```
// Core Image Metal kernel to apply a homography matrix
```

```
float2 warpHomography(float3x3 h, destination dest)
```

```
{
```

```
    float3 homogeneousDestCoord = float3(dest.coord(), 1.0);
```

```
    float3 homogeneousSrcCoord = h * homogeneousDestCoord;
```

```
    float2 srcCoord = homogeneousSrcCoord.xy / max(homogeneousSrcCoord.z, 0.000001);
```

```
    return srcCoord;
```

```
}
```

```
// Core Image Metal kernel to return the median of 5 images

inline void swap(thread float4 &a, thread float4 &b) {
    float4 tmp = a; a = min(a,b); b = max(tmp, b); // swap sort of two elements
}

float4 medianReduction5(sample_t v0, sample_t v1, sample_t v2, sample_t v3, sample_t v4)
{
    // using a Bose-Nelson sorting network
    swap(v0, v1); swap(v3, v4); swap(v2, v4); swap(v2, v3); swap(v0, v3);
    swap(v0, v2); swap(v1, v4); swap(v1, v3); swap(v1, v2);
    return v2;
}
```

```
// Core Image Metal kernel to return the median of 5 images
```

```
inline void swap(thread float4 &a, thread float4 &b) {
```

```
    float4 tmp = a; a = min(a,b); b = max(tmp, b); // swap sort of two elements
```

```
}
```

```
float4 medianReduction5(sample_t v0, sample_t v1, sample_t v2, sample_t v3, sample_t v4)
```

```
{
```

```
    // using a Bose-Nelson sorting network
```

```
    swap(v0, v1); swap(v3, v4); swap(v2, v4); swap(v2, v3); swap(v0, v3);
```

```
    swap(v0, v2); swap(v1, v4); swap(v1, v3); swap(v1, v2);
```

```
    return v2;
```

```
}
```

```
// Core Image Metal kernel to return the median of 5 images
```

```
inline void swap(thread float4 &a, thread float4 &b) {
```

```
    float4 tmp = a; a = min(a,b); b = max(tmp, b); // swap sort of two elements
```

```
}
```

```
float4 medianReduction5(sample_t v0, sample_t v1, sample_t v2, sample_t v3, sample_t v4)
```

```
{
```

```
    // using a Bose-Nelson sorting network
```

```
    swap(v0, v1); swap(v3, v4); swap(v2, v4); swap(v2, v3); swap(v0, v3);
```

```
    swap(v0, v2); swap(v1, v4); swap(v1, v3); swap(v1, v2);
```

```
    return v2;
```

```
}
```

Demo

Photo from Video with Removal of Unwanted Objects

Sky Gao, Image Engineer

Summary

Performance

Write CIKernels in Metal
CIRenderDestination API

Information

CIRenderInfo API
Xcode Quick Looks

Functionality

New Filters
Barcode Support
Depth Support

More Information

<https://developer.apple.com/wwdc17/510>

Related Sessions

Image Editing with Depth

WWDC 2017

What's New in Photos APIs

WWDC 2017

Vision Framework: Building on Core ML

WWDC 2017

Capturing Depth in iPhone Photography

WWDC 2017

Labs

Photos Editing and Core Image Lab

Technology Lab F

Thu 3:10PM–6:00PM

Photos Depth and Capture Lab

Technology Lab A

Thu 3:10PM–6:00PM

Photos Depth and Capture Lab

Technology Lab F

Fri 1:50PM–4:00PM

