Class

CIImage

A representation of an image to be processed or produced by Core Image filters.

iOS 5.0+ | iPadOS 5.0+ | Mac Catalyst 13.1+ | macOS 10.4+ | tvOS | visionOS 1.0+

```
class CIImage
```

Mentioned in

- Processing an Image Using Built-in Filters
- Selectively Focusing on an Image
- Customizing Image Transitions

Overview

You use `CIImage` objects in conjunction with other Core Image classes—such as `CIFilter`, `CIContext`, `CIVector`, and `CIColor`—to take advantage of the built-in Core Image filters when processing images. You can create `CIImage` objects with data supplied from a variety of sources, including Quartz 2D images, Core Video image buffers (`CVImageBuffer`), URL-based objects, and `NSData` objects.

Although a `CIImage` object has image data associated with it, it is not an image. You can think of a `CIImage` object as an image "recipe." A `CIImage` object has all the information necessary to produce an image, but Core Image doesn't actually render an image until it is told to do so. This lazy evaluation allows Core Image to operate as efficiently as possible. To show a `CIImage` object as an on-screen image, you can display it as a `UIImage` in `UIImageView`:

Swift Objective-C

```
guard let imageURL = Bundle.main.url(forResource: "YourJPEGName", withExtension: "JP
    print("Could not find image")
    return
}
guard let let ciImage = CIImage(contentsOf: imageURL) else {
    print("Could not create CIImage")
    return
}
let uiImage = UIImage(ciImage: ciImage)
let imageView = UIImageView(image: uiImage)
self.view.addSubview(imageView)
```

`CIContext` and `CIImage` objects are immutable, which means each can be shared safely among threads. Multiple threads can use the same GPU or CPU `CIContext` object to render `CIImage` objects. However, this is not the case for `CIFilter` objects, which are mutable. A `CIFilter` object cannot be shared safely among threads. If you app is multithreaded, each thread must create its own `CIFilter` objects. Otherwise, your app could behave unexpectedly.

Core Image also provides auto-adjustment methods. These methods analyze an image for common deficiencies and return a set of filters to correct those deficiencies. The filters are preset with values for improving image quality by altering values for skin tones, saturation, contrast, and shadows and for removing red-eye or other artifacts caused by flash. (See Getting Autoadjustment Filters.)

For a discussion of all the methods you can use to create `CIImage` objects on iOS and macOS, see Core Image Programming Guide.

Topics

Creating an Image

`class func empty() -> CIImage`

Creates and returns an empty image object.

`init?(image: UIImage)`

Initializes an image object with the specified UIKit image object.

`init?(image: UIImage, options: [CIImageOption : Any]?)`

Initializes an image object with the specified UIKit image object, using the specified options.

`init?(contentsOf: URL)`

Initializes an image object by reading an image from a URL.

```
init?(contentsOf: URL, options: [CIImageOption : Any]?)
```

Initializes an image object by reading an image from a URL, using the specified options.

```
init(cgImage: CGImage)
```

Initializes an image object with a Quartz 2D image.

```
init(cgImage: CGImage, options: [CIImageOption : Any]?)
```

Initializes an image object with a Quartz 2D image, using the specified options.

```
init(cgImageSource: CGImageSource, index: Int, options: [CIImageOption
: Any]?)
```

```
init?(data: Data)
```

Initializes an image object with the supplied image data.

```
init?(data: Data, options: [CIImageOption : Any]?)
```

Initializes an image object with the supplied image data, using the specified options.

```
init(bitmapData: Data, bytesPerRow: Int, size: CGSize, format: CIFormat
, colorSpace: CGColorSpace?)
```

Initializes an image object with bitmap data.

```
init?(bitmapImageRep: NSBitmapImageRep)
```

Initializes an image object with the specified bitmap image representation.

```
init(imageProvider: Any, size: Int, Int, format: CIFormat, colorSpace:
CGColorSpace?, options: [CIImageOption : Any]?)
```

Initializes an image object based on pixels from an image provider object.

```
init?(depthData: AVDepthData)
```

```
init?(depthData: AVDepthData, options: [String : Any]?)
```

```
init?(portaitEffectsMatte: AVPortraitEffectsMatte)
```

```
init?(portaitEffectsMatte: AVPortraitEffectsMatte, options: [CIImage
Option : Any]?)
```

```
init?(semanticSegmentationMatte: AVSemanticSegmentationMatte)
```

```
init?(semanticSegmentationMatte: AVSemanticSegmentationMatte, options:
[CIImageOption : Any]?)
```

```
init(cvImageBuffer: CVImageBuffer)
```

Initializes an image object from the contents of a Core Video image buffer.

`init(cvImageBuffer: CVImageBuffer, options: [CIImageOption : Any]?)`

Initializes an image object from the contents of a Core Video image buffer, using the specified options.

`init(cvPixelBuffer: CVPixelBuffer)`

Initializes an image object from the contents of a Core Video pixel buffer.

`init(cvPixelBuffer: CVPixelBuffer, options: [CIImageOption : Any]?)`

Initializes an image object from the contents of a Core Video pixel buffer using the specified options.

`init?(mtlTexture: any MTLTexture, options: [CIImageOption : Any]?)`

Initializes an image object with data supplied by a Metal texture.

`init(ioSurface: IOSurfaceRef)`

Initializes an image with the contents of an IOSurface.

`init(ioSurface: IOSurfaceRef, options: [CIImageOption : Any]?)`

Initializes, using the specified options, an image with the contents of an IOSurface.

Creating an Image by Modifying an Existing Image

`func applyingFilter(String, parameters: [String : Any]) -> CIImage`

Returns a new image created by applying a filter to the original image with the specified name and parameters.

`func applyingFilter(String) -> CIImage`

Applies the filter to an image and returns the output.

`func transformed(by: CGAffineTransform) -> CIImage`

Returns a new image that represents the original image after applying an affine transform.

`func transformed(by: CGAffineTransform, highQualityDownsample: Bool) -> CIImage`

`func cropped(to: CGRect) -> CIImage`

Returns a new image with a cropped portion of the original image.

`func oriented(forExifOrientation: Int32) -> CIImage`

Returns a new image created by transforming the original image to the specified EXIF orientation.

```
func clampedToExtent() -> CIImage
```

Returns a new image created by making the pixel colors along its edges extend infinitely in all directions.

```
func clamped(to: CGRect) -> CIImage
```

Returns a new image created by cropping to a specified area, then making the pixel colors along the edges of the cropped image extend infinitely in all directions.

```
func composited(over: CIImage) -> CIImage
```

Returns a new image created by compositing the original image over the specified destination image.

```
func convertingWorkingSpaceToLab() -> CIImage
```

```
func convertingLabToWorkingSpace() -> CIImage
```

```
func matchedToWorkingSpace(from: CGColorSpace) -> CIImage?
```

Returns a new image created by color matching from the specified color space to the context's working color space.

```
func matchedFromWorkingSpace(to: CGColorSpace) -> CIImage?
```

Returns a new image created by color matching from the context's working color space to the specified color space.

```
func premultiplyingAlpha() -> CIImage
```

Returns a new image created by multiplying the image's RGB values by its alpha values.

```
func unpremultiplyingAlpha() -> CIImage
```

Returns a new image created by dividing the image's RGB values by its alpha values.

```
func settingAlphaOne(in: CGRect) -> CIImage
```

Returns a new image created by setting all alpha values to 1.0 within the specified rectangle and to 0.0 outside of that area.

```
func applyingGaussianBlur(sigma: Double) -> CIImage
```

Create an image by applying a gaussian blur to the receiver.

```
func settingProperties([AnyHashable : Any]) -> CIImage
```

Return a new image by changing the receiver's metadata properties.

```
func insertingIntermediate() -> CIImage
```

Create an image that inserts a intermediate that is cacheable

```
func insertingIntermediate(cache: Bool) -> CIImage
```

Create an image that inserts a intermediate that is cacheable.

Creating Solid Colors

```
init(color: CIColor)
```

Initializes an image of infinite extent whose entire content is the specified color.

```
class var black: CIImage
```

```
class var blue: CIImage
```

```
class var clear: CIImage
```

```
class var cyan: CIImage
```

```
class var gray: CIImage
```

```
class var green: CIImage
```

```
class var magenta: CIImage
```

```
class var red: CIImage
```

```
class var white: CIImage
```

```
class var yellow: CIImage
```

Getting Image Information

```
var definition: CIFilterShape
```

Returns a filter shape object that represents the domain of definition of the image.

```
var extent: CGRect
```

A rectangle that specifies the extent of the image.

```
var properties: [String : Any]
```

Returns the metadata properties dictionary of the image.

```
var url: URL?
```

The URL from which the image was loaded.

```
var colorSpace: CGColorSpace?
```

The color space of the image.

`func orientationTransform(forExifOrientation: Int32) -> CGAffine Transform`

Returns the transformation needed to reorient the image to the specified orientation.

Drawing Images

`func draw(at: NSPoint, from: NSRect, operation: NSCompositingOperation, fraction: CGFloat)`

Draws all or part of the image at the specified point in the current coordinate system.

`func draw(in: NSRect, from: NSRect, operation: NSCompositingOperation, fraction: CGFloat)`

Draws all or part of the image in the specified rectangle in the current coordinate system

Getting Autoadjustment Filters

`func autoAdjustmentFilters() -> [CIFilter]`

Returns all possible automatically selected and configured filters for adjusting the image.

`func autoAdjustmentFilters(options: [CIImageAutoAdjustmentOption : Any]?) -> [CIFilter]`

Returns a subset of automatically selected and configured filters for adjusting the image.

Autoadjustment Keys

Constants used as keys in the options dictionary for the `autoAdjustment Filters(options:)` method.

Working with Filter Regions of Interest

`func regionOfInterest(for: CIImage, in: CGRect) -> CGRect`

Returns the region of interest for the filter chain that generates the image.

Working with Orientation

`func oriented(CGImagePropertyOrientation) -> CIImage`

Transforms the original image by a given orientation.

`func orientationTransform(for: CGImagePropertyOrientation) -> CGAffine Transform`

The affine transform for changing the image to the given orientation.

Sampling the Image

`func samplingNearest() -> CIImage`

Create an image by changing the receiver's sample mode to nearest neighbor.

`func samplingLinear() -> CIImage`

Create an image by changing the receiver's sample mode to bilinear interpolation.

Accessing Original Image Content

`var cgImage: CGImage?`

The CoreGraphics image object this image was created from, if applicable.

`var pixelBuffer: CVPixelBuffer?`

The CoreVideo pixel buffer this image was created from, if applicable.

`var depthData: AVDepthData?`

Depth data associated with the image.

`var portraitEffectsMatte: AVPortraitEffectsMatte?`

The portrait effects matte associated with the image.

`var semanticSegmentationMatte: AVSemanticSegmentationMatte?`

Image Dictionary Keys

Constants used as keys in the options dictionary when initializing an image.

`struct CIImageOption`

AutoAdjustment Keys

Constants used as keys in the options dictionary for the `autoAdjustmentFilters(options:)` method.

`struct CIImageAutoAdjustmentOption`

Deprecated

~~`init(cgLayer: CGLayer)`~~

Initializes an image object from the contents supplied by a CGLayer object.

Deprecated

~~`init(cgLayer: CGLayer, options: [CIImageOption : Any]?)`~~

Initializes an image object from the contents supplied by a CGLayer object, using the specified options.

Deprecated

~~`init(texture: UInt32, size: CGSize, flipped: Bool, colorSpace: CGColorSpace?)`~~

Initializes an image object with data supplied by an OpenGL texture.

Deprecated

~~`init(texture: UInt32, size: CGSize, flipped: Bool, options: [CIImageOption : Any]?)`~~

Initializes an image object with data supplied by an OpenGL texture.

Deprecated

~~`init(ioSurface: IOSurfaceRef, plane: Int, format: CIFormat, options: [CIImageOption : Any]?)`~~

Initializes, using the specified format and options, an image with the contents of a specific data plane in an IOSurface.

Deprecated

~~`static let textureTarget: CIImageOption`~~

The key for an OpenGL texture target.

Deprecated

~~`static let textureFormat: CIImageOption`~~

The key for an OpenGL texture format.

Deprecated

Instance Properties

`var contentHeadroom: Float`

Returns the content headroom of the image.

`var isOpaque: Bool`

Returns YES if the image is known to have and alpha value of `1.0` over the entire image extent.

```
var metalTexture: (any MTLTexture)?
```

```
var contentAverageLightLevel: Float
```

Returns the content average light level of the image.

Instance Methods

```
func applyingGainMap(CIImage) -> CIImage
```

Create an image that applies a gain map Core Image image to the received Core Image image.

```
func applyingGainMap(CIImage, headroom: Float) -> CIImage
```

Create an image that applies a gain map Core Image image with a specified headroom to the received Core Image image.

```
func insertingTiledIntermediate() -> CIImage
```

Create an image that inserts a intermediate that is cached in tiles

```
func settingContentAverageLightLevel(Float) -> CIImage
```

Create an image by changing the receiver's contentAverageLightLevel property.

```
func settingContentHeadroom(Float) -> CIImage
```

Create an image by changing the receiver's contentHeadroom property.

Relationships

Inherits From

NSObject

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSCoding
NSCopying

`NSObjectProtocol`
`NSSecureCoding`
`Sendable`
`SendableMetatype`

See Also

Essentials

Processing an Image Using Built-in Filters

Apply effects such as sepia tint, highlight strengthening, and scaling to images.

`class` `CIContext`

The Core Image context class provides an evaluation context for Core Image processing with Metal, OpenGL, or OpenCL.