Core Image Reference Collection

Graphics & Animation: 2D Drawing

2006-12-05

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures

Chapter 16 **Core Image Filter Reference 153**

Core Image Reference Collection

Framework	Library/Frameworks/QuartzCore.framework
Header file directories	Library/Frameworks/QuartzCore.framework/Headers
Declared in	CACIFilterAdditions.h CIColor.h CIContext.h CIFilter.h CIFilterGenerator.h CIFilterShape.h CIImage.h CIImageAccumulator.h CIImageProvider.h CIKernel.h CIPlugIn.h CIPlugInInterface.h CIRAWFilter.h CISampler.h CIVector.h IKFilterUI.h

Introduction

The Core Image class hierarchy is rooted in the Foundation framework class `NSObject`. You can use Core Image classes to:

- Process images using existing image filters
- Create custom filters either for your own use or to package as image units
- Chain together filters and then archive them for later use

Core Image is designed to:

- Leverage programmable graphics hardware when possible
- Be easy to use and to extend. You can use the Core Image API even if you don't know how to use Open GL, access pixel buffers (pbuffers), or perform other low-level graphics-processing tasks
- Provide access to a rich set of plug-in filters and yet allow you to create custom filters that you can publish for use by others

PART I

Classes

CIColor Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIColor.h
Availability	Mac OS X v10.4 and later
Companion guides	Core Image Programming Guide Color Management Overview
Related sample code	CIAnnotation CIColorTracking CIHazeFilterSample CITransitionSelectorSample FunHouse

Overview

The `CIColor` class contains color values and the color space for which the color values are valid. You use `CIColor` objects in conjunction with other Core Image classes, such as `CIFilter`, `CIContext`,and `CIImage`, to take advantage of the built-in Core Image filters when processing images.

A color space defines a one-, two-, three-, or four-dimensional environment whose color components represent intensity values. A color component is also referred to as a color channel. An RGB color space, for example, is a three-dimensional color space whose stimuli are the red, green, and blue intensities that make up a given color. Regardless of the color space, in Core Image, color values range from `0.0` to `1.0`, with `0.0` representing an absence of that component (0 percent) and `1.0` representing 100 percent.

Colors also have an alpha component that represents the opacity of the color, with `0.0` meaning completely transparent and `1.0` meaning completely opaque. If a color does not have an explicit alpha component, Core Image paints the color as if the alpha component equals `1.0`. You always provide unpremultiplied color components to Core Image and Core Image provides unpremultiplied color components to you. Core Image premultiplies each color component with the alpha value in order to optimize calculations. For more information on premultiplied alpha values see *Core Image Programming Guide*.

Tasks

Initializing Color Objects

`– initWithCGColor:` (page 21)
Initializes a color object with a Quartz color.

Creating Color Objects

`+ colorWithCGColor:` (page 17)
Creates a color object from a Quartz color.

`+ colorWithRed:green:blue:` (page 17)
Creates a color object using the specified RGB color component values

`+ colorWithRed:green:blue:alpha:` (page 18)
Creates a color object using the specified RGBA color component values.

`+ colorWithString:` (page 19)
Creates a color object using the RGBA color component values specified by a string.

Getting Color Components

`– alpha` (page 19)
Returns the alpha value of the color.

`– blue` (page 20)
Returns the blue component of the color.

`– colorSpace` (page 20)
Returns the Quartz 2D color space associated with the color.

`– components` (page 20)
Returns the color components of the color.

`– green` (page 21)
Returns the green component of the color.

`– numberOfComponents` (page 22)
Returns the number of color components in the color.

`– red` (page 22)
Returns the red component of the color.

`– stringRepresentation` (page 22)
Returns a formatted string that specifies the components of the color.

Class Methods

colorWithCGColor:

Creates a color object from a Quartz color.

```
+ (CIColor *)colorWithCGColor:(CGColorRef)c
```

Parameters

c

A Quartz color (`CGColorRef` object) created using a Quartz color creation function such as `CGColorCreate`.

Return Value

A Core Image color object that represents a Quartz color.

Discussion

A `CGColorRef` object is the fundamental opaque data type used internally by Quartz to represent colors. For more information on Quartz 2D color and color spaces, see *Quartz 2D Programming Guide*.

You can pass a `CGColorRef` object that represents any color space, including CMYK, but Core Image converts all color spaces to the Core Image working color space before it passes the color space to the filter kernel. The Core Image working color space uses three color components plus alpha.

Availability

Mac OS X v10.4 and later.

See Also

`+ colorWithRed:green:blue:` (page 17)

`+ colorWithRed:green:blue:alpha:` (page 18)

`+ colorWithString:` (page 19)

Declared In

`CIColor.h`

colorWithRed:green:blue:

Creates a color object using the specified RGB color component values

```
+ (CIColor *)colorWithRed:(CGFloat)r green:(CGFloat)g blue:(CGFloat)b
```

Parameters

r

The value of the red component.

g

The value of the green component.

b

The value of the blue component.

Return Value

A Core Image color object that represents an RGB color in the color space specified by the Quartz 2D constant `kCGColorSpaceGenericRGB`.

Availability

Mac OS X v10.4 and later.

See Also

`+ colorWithCGColor:` (page 17)

`+ colorWithRed:green:blue:alpha:` (page 18)

`+ colorWithString:` (page 19)

Related Sample Code

CIColorTracking

CIHazeFilterSample

Declared In

`CIColor.h`

colorWithRed:green:blue:alpha:

Creates a color object using the specified RGBA color component values.

```
+ (CIColor *)colorWithRed:(CGFloat)r green:(CGFloat)g blue:(CGFloat)b
    alpha:(CGFloat)a
```

Parameters

r

The value of the red component.

g

The value of the green component.

b

The value of the blue component.

a

The value of the alpha component.

Return Value

A Core Image color object that represents an RGB color in the color space specified by the Quartz 2D constant `kCGColorSpaceGenericRGB` and an alpha value.

Availability

Mac OS X v10.4 and later.

See Also

`+ colorWithCGColor:` (page 17)

`+ colorWithRed:green:blue:` (page 17)

`+ colorWithString:` (page 19)

Related Sample Code

CIAnnotation

CIHazeFilterSample

CIMicroPaint

CITransitionSelectorSample
FunHouse

Declared In
`CIColor.h`

colorWithString:

Creates a color object using the RGBA color component values specified by a string.

```
+ (CIColor *)colorWithString:(NSString *)representation
```

Parameters

representation

A string that is in one of the formats returned by the `stringRepresentation` method. For example, the string:

```
@"0.5 0.7 0.3 1.0"
```

indicates an RGB color whose components are 50% red, 70% green, 30% blue, and 100% opaque (alpha value of 1.0). The string representation always has four components—red, green, blue, and alpha. The default value for the alpha component is `1.0`.

Return Value
A Core Image color object that represents an RGB color in the color space specified by the Quartz 2D constant `kCGColorSpaceGenericRGB`.

Availability
Mac OS X v10.4 and later.

See Also
`+ colorWithCGColor:` (page 17)
`+ colorWithRed:green:blue:` (page 17)
`+ colorWithRed:green:blue:alpha:` (page 18)

Related Sample Code
FunHouse

Declared In
`CIColor.h`

Instance Methods

alpha

Returns the alpha value of the color.

```
- (CGFloat)alpha
```

Return Value
The alpha value. A color created without an explicit alpha value has an alpha of 1.0 by default.

Availability

Mac OS X v10.4 and later.

See Also

`- components` (page 20)

Declared In

`CIColor.h`

blue

Returns the blue component of the color.

```
- (CGFloat)blue
```

Return Value

The unpremultiplied blue component of the color.

Availability

Mac OS X v10.4 and later.

See Also

`- components` (page 20)

Declared In

`CIColor.h`

colorSpace

Returns the Quartz 2D color space associated with the color.

```
- (CGColorSpaceRef)colorSpace
```

Return Value

The Quartz 2D color space (`CGColorSpaceRef` object). You are responsible for disposing of this color space by calling the Quartz 2D function `CGColorSpaceRelease`.

Availability

Mac OS X v10.4 and later.

See Also

`- components` (page 20)

Declared In

`CIColor.h`

components

Returns the color components of the color.

```
- (const CGFloat *)components
```

Return Value

An array of color components, specified as floating-point values in the range of 0.0 through 1.0. This array includes an alpha component if there is one.

Availability

Mac OS X v10.4 and later.

See Also

`- numberOfComponents` (page 22)
`- stringRepresentation` (page 22)

Declared In

`CIColor.h`

green

Returns the green component of the color.

```
- (CGFloat)green
```

Return Value

The unpremultiplied green component of the color.

Availability

Mac OS X v10.4 and later.

See Also

`- components` (page 20)

Declared In

`CIColor.h`

initWithCGColor:

Initializes a color object with a Quartz color.

```
- (id)initWithCGColor:(CGColorRef)c
```

Parameters

c

A Quartz color (`CGColorRef`) created using a Quartz color creation function such as `CGColorCreate`.

Discussion

A `CGColorRef` object is the fundamental opaque data type used internally by Quartz to represent colors. For more information on Quartz 2D color and color spaces, see *Quartz 2D Programming Guide*.

You can pass a `CGColorRef` object that represents any color space, including CMYK, but Core Image converts all color spaces to the Core Image working color space before it passes the color space to the filter kernel. The Core Image working color space uses three color components plus alpha.

Availability

Mac OS X v10.4 and later.

Declared In
`CIColor.h`

numberOfComponents

Returns the number of color components in the color.

```
- (size_t)numberOfComponents
```

Return Value
The number of color components, which includes an alpha component if there is one.

Availability
Mac OS X v10.4 and later.

See Also
`- components` (page 20)

Declared In
`CIColor.h`

red

Returns the red component of the color.

```
- (CGFloat)red
```

Return Value
The unpremultiplied red component of the color.

Availability
Mac OS X v10.4 and later.

See Also
`- components` (page 20)

Declared In
`CIColor.h`

stringRepresentation

Returns a formatted string that specifies the components of the color.

```
- (NSString *)stringRepresentation
```

Return Value
The formatted string.

Discussion
The string representation always has four components—red, green, blue, and alpha. The default value for the alpha component is `1.0`.F or example, this string:

```
@"0.5 0.7 0.3 1.0"
```

indicates an RGB color whose components are 50% red, 70% green, 30% blue, and 100% opaque (alpha value of 1.0).

Availability

Mac OS X v10.4 and later.

See Also

`- components` (page 20)

Declared In

`CIColor.h`

CIContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIContext.h
Availability	Mac OS X v10.4 and later
Companion guides	Core Image Programming Guide Image Unit Tutorial
Related sample code	CIAnnotation CIRAWFilterSample Denoise FunHouse Reducer

Overview

The CIContext class provides an evaluation context for rendering a CIImage object through Quartz 2D or OpenGL. You use CIContext objects in conjunction with other Core Image classes, such as CIFilter, CIImage, and CIColor, to take advantage of the built-in Core Image filters when processing images.

Tasks

Creating a Context

+ contextWithCGContext:options: (page 26)
  Creates a Core Image context from a Quartz context, using the specified options.

+ contextWithCGLContext:pixelFormat:options: (page 27)
  Creates a Core Image context from a CGL context, using the specified options and pixel format object.

Rendering Images

- `createCGImage:fromRect:` (page 29)
  Creates a Quartz 2D image from a region of a `CIImage` object.
- `createCGImage:fromRect:format:colorSpace:` (page 29)
  Creates a Quartz 2D image from a region of a `CIImage` object.
- `createCGLayerWithSize:info:` (page 30)
  Creates a CGLayer object from the provided parameters.
- `drawImage:atPoint:fromRect:` (page 31)
  Renders a region of an image to a point in the context destination.
- `drawImage:inRect:fromRect:` (page 31)
  Renders a region of an image to a rectangle in the context destination.
- `render:toBitmap:rowBytes:bounds:format:colorSpace:` (page 32)
  Renders to the given bitmap.

Managing Resources

- `clearCaches` (page 28)
  Frees any cached data, such as temporary images, associated with the context and runs the garbage collector.
- `reclaimResources` (page 32)
  Runs the garbage collector to reclaim any resources that the context no longer requires.

Class Methods

contextWithCGContext:options:

Creates a Core Image context from a Quartz context, using the specified options.

```
+ (CIContext *)contextWithCGContext:(CGContextRef)ctx options:(NSDictionary *)dict
```

Parameters

ctx

A Quartz graphics context (`CGContextRef` object) either obtained from the system or created using a Quartz function such as `CGBitmapContextCreate`. See *Quartz 2D Programming Guide* for information on creating Quartz graphics contexts.

dict

A dictionary that contains color space information. You can provide the keys `kCIContextOutputColorSpace` (page 33) or `kCIContextWorkingColorSpace` (page 33) along with a `CGColorSpaceRef`object for each color space.

Discussion

After calling this method, Core Image draws content to the specified Quartz graphics context.

When you create a `CIContext` object using a Quartz graphics context, any transformations that are already set on the Quartz graphics context affect drawing to that context.

Availability
Mac OS X v10.4 and later.

See Also
`+ contextWithCGLContext:pixelFormat:options:` (page 27)

Related Sample Code
CIAnnotation

CIRAWFilterSample

FunHouse

ImageApp

UnsharpMask

Declared In
`CIContext.h`

contextWithCGLContext:pixelFormat:options:

Creates a Core Image context from a CGL context, using the specified options and pixel format object. (Deprecated in Mac OS X v10.6.)

```
+ (CIContext *)contextWithCGLContext:(CGLContextObj)ctx
    pixelFormat:(CGLPixelFormatObj)pf options:(NSDictionary *)dict
```

Parameters

ctx

A CGL context (`CGLContextObj` object) obtain by calling the CGL function `CGLCreateContext`.

pf

A CGL pixel format object (`CGLPixelFormatObj` object) created by calling the CGL function `CGLChoosePixelFormat`. This argument must be the same pixel format object used to create the CGL context. The pixel format object must be valid for the lifetime of the Core Image context. Don't release the pixel format object until after you release the Core Image context.

options

A dictionary that contains color space information. You can provide the keys `kCIContextOutputColorSpace` (page 33) or `kCIContextWorkingColorSpace` (page 33) along with a `CGColorSpaceRef` object for each color space.

Discussion
After calling this method, Core Image draws content into the surface (drawable object) attached to the CGL context. A CGL context is an Mac OS X OpenGL context. For more information, see *OpenGL Programming Guide for Mac OS X*.

When you create a `CIContext` object using a CGL context, all OpenGL states set for the CGL context affect rendering to that context. That means that coordinate and viewport transformations set on the CGL context as well as the vertex color.

For best results, follow these guidelines when you use Core Image to render into an OpenGL context:

- Ensure that the a single unit in the coordinate space of the OpenGL context represents a single pixel in the output device.
- The Core Image coordinate space has the origin in the bottom left corner of the screen. You should configure the OpenGL context in the same way.

- The OpenGL context blending state is respected by Core Image. If the image you want to render contains translucent pixels, it's best to enable blending using a blend function with the parameters GL_ONE, GL_ONE_MINUS_SRC_ALPHA, as shown in the following code example.

Some typical initialization code for a view with width W and height H is:

```
glViewport (0, 0, W, H);
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
glOrtho (0, W, 0, H, -1, 1);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glBlendFunc (GL_ONE, GL_ONE_MINUS_SRC_ALPHA);
glEnable (GL_BLEND);
```

Availability
Mac OS X v10.4 and later.

Deprecated in Mac OS X v10.6.

See Also
`+ contextWithCGContext:options:` (page 26)

Related Sample Code
CIAnnotation

CIFilterGeneratorTest

CIVideoDemoGL

CoreImageGLTextureFBO

WhackedTV

Declared In
`CIContext.h`

Instance Methods

clearCaches

Frees any cached data, such as temporary images, associated with the context and runs the garbage collector.

```
- (void)clearCaches
```

Discussion
You can use this method to remove textures from the texture cache that reference deleted images.

Availability
Mac OS X v10.4 and later.

See Also
`- reclaimResources` (page 32)

Declared In
`CIContext.h`

createCGImage:fromRect:

Creates a Quartz 2D image from a region of a `CIImage` object.

```
- (CGImageRef)createCGImage:(CIImage *)im fromRect:(CGRect)r
```

Parameters

im

A `CIImage` object.

r

The region of the image to render.

Return Value

A Quartz 2D (`CGImageRef`) image. You are responsible for releasing the returned image when you no longer need it.

Discussion

Renders a region of an image into a temporary buffer using the context, then creates and returns a Quartz 2D image with the results.

Availability

Mac OS X v10.4 and later.

See Also

`- createCGImage:fromRect:format:colorSpace:` (page 29)

Related Sample Code

CIAnnotation

CIVideoDemoGL

DispatchFractal

FunHouse

Declared In

`CIContext.h`

createCGImage:fromRect:format:colorSpace:

Creates a Quartz 2D image from a region of a `CIImage` object.

```
- (CGImageRef)createCGImage:(CIImage *)im fromRect:(CGRect)r
        format:(CIFormat)f colorSpace:(CGColorSpaceRef)cs
```

Parameters

im

A `CIImage` object.

r

The region of the image to render.

f

The format of the image.

cs

The color space of the image.

Return Value

A Quartz 2D (`CGImageRef`) image. You are responsible for releasing the returned image when you no longer need it.

Discussion

Renders a region of an image into a temporary buffer using the context, then creates and returns a Quartz 2D image with the results.

Availability

Mac OS X v10.5 and later.

See Also

`– createCGImage:fromRect:` (page 29)

Related Sample Code

FunHouse

Declared In

`CIContext.h`

createCGLayerWithSize:info:

Creates a CGLayer object from the provided parameters.

```
- (CGLayerRef)createCGLayerWithSize:(CGSize)size info:(CFDictionaryRef)d
```

Parameters

`size`

The size, in default user space units, of the layer relative to the graphics context.

`d`

A dictionary, which is passed to `CGLayerCreateWithContext` as the `auxiliaryInfo` parameter. Pass `NULL` as this parameter is reserved for future use.

Return Value

A CGLayer (`CGLayerRef`) object.

Discussion

After calling this method, Core Image draws content into the CGLayer object. Core Image creates a CGLayer object by calling the Quartz 2D function `CGLayerCreateWithContext`, whose prototype is:

```
CGLayerRef CGLayerCreateWithContext (
   CGContextRef context,
   CGSize size,
   CFDictionaryRef auxiliaryInfo
);
```

Core Image passes the `CIContext` object as the `context` parameter, the size as the `size` parameter, and the dictionary as the `auxiliaryInfo` parameter. For more information on CGLayer objects, see *Quartz 2D Programming Guide* and *CGLayer Reference*.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithCGLayer:` (page 90)

`+ imageWithCGLayer:options:` (page 90)

Related Sample Code

CIAnnotation

CIBevelSample

FunHouse

Declared In

`CIContext.h`

drawImage:atPoint:fromRect:

Renders a region of an image to a point in the context destination.

```
- (void)drawImage:(CIImage *)im atPoint:(CGPoint)p fromRect:(CGRect)src
```

Parameters

im

A `CIImage` object.

p

The point in the context destination to draw to.

src

The region of the image to draw.

Discussion

You can call this method to force evaluation of the result after you apply a filter using one of the methods of the `CIFilter` class, such as `apply:` (page 43), `apply:arguments:options:` (page 44), and `apply:k, . . ..`

Availability

Mac OS X v10.4 and later.

See Also

`- drawImage:inRect:fromRect:` (page 31)

Related Sample Code

AnimatedTableView

CIHazeFilterSample

CITransitionSelectorSample

CIVideoDemoGL

Denoise

Declared In

`CIContext.h`

drawImage:inRect:fromRect:

Renders a region of an image to a rectangle in the context destination.

```
- (void)drawImage:(CIImage *)im inRect:(CGRect)dest fromRect:(CGRect)src
```

Parameters

im

A CIImage object.

dest

The rectangle in the context destination to draw into.

src

The subregion of the image that you want to draw into the context, with the origin and target size defined by the dest parameter.

Discussion

You can call this method to force evaluation of the result after you you apply a filter using one of the methods of the CIFilter class, such as apply: (page 43), apply:arguments:options: (page 44), and apply:k, ...

Availability

Mac OS X v10.4 and later.

See Also

- drawImage:atPoint:fromRect: (page 31)

Related Sample Code

CIColorTracking

CIRAWFilterSample

ImageApp

VideoViewer

Declared In

CIContext.h

reclaimResources

Runs the garbage collector to reclaim any resources that the context no longer requires.

```
- (void)reclaimResources
```

Discussion

The system calls this method automatically after every rendering operation. You can use this method to remove textures from the texture cache that reference deleted images.

Availability

Mac OS X v10.4 and later.

See Also

- clearCaches (page 28)

Declared In

CIContext.h

render:toBitmap:rowBytes:bounds:format:colorSpace:

Renders to the given bitmap.

```
- (void)render:(CIImage *)im toBitmap:(void *)data rowBytes:(ptrdiff_t)rb
    bounds:(CGRect)r format:(CIFormat)f colorSpace:(CGColorSpaceRef)cs
```

Parameters

im
: A `CIImage` object.

data
: Storage for the bitmap data.

rb
: The bytes per row.

r
: The bounds of the bitmap data.

f
: The format of the bitmap data.

cs
: The color space for the data. Pass `NULL` if you want to use the output color space of the context.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CIContext.h`

Constants

Context Options

Keys in the options dictionary for a `CIContext` object.

```
extern NSString *kCIContextOutputColorSpace;
extern NSString *kCIContextWorkingColorSpace;
extern NSString *kCIContextUseSoftwareRenderer;
```

Constants

`kCIContextOutputColorSpace`
: A key for the color space to use for images before they are rendered to the context. By default, Core Image uses the GenericRGB color space, which leaves color matching to the system. You can specify a different output color space by providing a Quartz 2D `CGColorSpace` object (`CGColorSpaceRef`). (See *Quartz 2D Programming Guide* for information on creating and using `CGColorSpace` objects.)

`kCIContextWorkingColorSpace`
: A key for the color space to use for image operations. By default, Core Image assumes that processing nodes are 128 bits-per-pixel, linear light, premultiplied RGBA floating-point values that use the GenericRGB color space. You can specify a different working color space by providing a Quartz 2D `CGColorSpace` object (`CGColorSpaceRef`). Note that the working color space must be RGB-based. If you have YUV data as input (or other data that is not RGB-based), you can use ColorSync functions to convert to the working color space. (See *Quartz 2D Programming Guide* for information on creating and using `CGColorSpace` objects.)

`kCIContextUseSoftwareRenderer`
A key for enabling software renderer use. If the associated `NSNumber` object is `YES`, then the software renderer is required.

Declared In
`CIContext.h`

CIFilter Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIFilter.h QuartzCore/CIRAWFilter.h
Availability	Mac OS X v10.4 and later
Companion guides	Core Image Programming Guide Image Unit Tutorial Core Image Filter Reference
Related sample code	CIAnnotation CIColorTracking CIRAWFilterSample FunHouse Reducer

Overview

The `CIFilter` class produces a `CIImage` object as output. Typically, a filter takes one or more images as input. Some filters, however, generate an image based on other types of input parameters. The parameters of a `CIFilter` object are set and retrieved through the use of key-value pairs.

You use the `CIFilter` object in conjunction with other Core Image classes, such as `CIImage`, `CIContext`, `CIImageAccumulator`, and `CIColor`, to take advantage of the built-in Core Image filters when processing images, creating filter generators, or writing custom filters.

Tasks

Creating a Filter

`+ filterWithName:` (page 39)
Creates a `CIFilter` object for a specific kind of filter.

+ filterWithName:keysAndValues: (page 40)
    Creates a CIFilter object for a specific kind of filter and initializes the input values.

Creating a Filter from a RAW Image

+ filterWithImageData:options: (page 38)
    Returns a CIFilter object initialized with RAW image data supplied to the method.

+ filterWithImageURL:options: (page 39)
    Returns a CIFilter object initialized with data from a RAW image file.

Accessing Registered Filters

+ filterNamesInCategories: (page 37)
    Returns an array of all published filter names that match all the specified categories.

+ filterNamesInCategory: (page 38)
    Returns an array of all published filter names in the specified category.

Registering a Filter

+ registerFilterName:constructor:classAttributes: (page 42)
    Publishes a custom filter that is not packaged as an image unit.

Getting Filter Parameters and Attributes

- attributes (page 45)
    Returns a dictionary of key-value pairs that describe the filter.

- inputKeys (page 46)
    Returns an array that contains the names of the input parameters to the filter.

- outputKeys (page 46)
    Returns an array that contains the names of the output parameters for the filter.

Setting Default Values

- setDefaults (page 46)
    Sets all input values for a filter to default values.

Applying a Filter

- apply:arguments:options: (page 44)
    Produces a *CIImage* object by applying arguments to a kernel function and using options to control how the kernel function is evaluated.

`- apply:` (page 43)
  Produces a `CIImage` object by applying a kernel function.

Getting Localized Information for Registered Filters

`+ localizedNameForFilterName:` (page 41)
  Returns the localized name for the specified filter name.

`+ localizedNameForCategory:` (page 41)
  Returns the localized name for the specified filter category.

`+ localizedDescriptionForFilterName:` (page 41)
  Returns the localized description of a filter for display in the user interface.

`+ localizedReferenceDocumentationForFilterName:` (page 42)
  Returns the location of the localized reference documentation that describes the filter.

Class Methods

filterNamesInCategories:

Returns an array of all published filter names that match all the specified categories.

```
+ (NSArray *)filterNamesInCategories:(NSArray *)categories
```

Parameters

`categories`
  One or more filter categories. Pass `nil` to get all filters in all categories.

Return Value
An array that contains all published filter names that match all the categories specified by the `categories` argument.

Discussion
When you pass more than one filter category, this method returns the intersection of the filters in the categories. For example, if you pass the categories `kCICategoryBuiltIn` (page 54) and `kCICategoryFilterGenerator` (page 55), you obtain all the filters that are members of both the built-in and generator categories. But if you pass in `kCICategoryGenerator` and `kCICategoryStylize` (page 54), you will not get any filters returned to you because there are no filters that are members of both the generator and stylize categories. If you want to obtain all stylize and generator filters, you must call the `filterNamesInCategories:` method for each category separately and then merge the results.

Availability
Mac OS X v10.4 and later.

See Also
`+ filterNamesInCategory:` (page 38)

Related Sample Code
CIAnnotation
CIColorTracking

CITransitionSelectorSample2

Declared In
`CIFilter.h`

filterNamesInCategory:

Returns an array of all published filter names in the specified category.

```
+ (NSArray *)filterNamesInCategory:(NSString *)category
```

Parameters

category
A string object that specifies a filter category.

Return Value
An array that contains all published names of the filter in a category.

Availability
Mac OS X v10.4 and later.

See Also
`+ filterNamesInCategories:` (page 37)

Related Sample Code
FunHouse

Declared In
`CIFilter.h`

filterWithImageData:options:

Returns a `CIFilter` object initialized with RAW image data supplied to the method.

```
+ (CIFilter *)filterWithImageData:(NSData *)data options:(NSDictionary *)options;
```

Parameters

data
The RAW image data to initialize the object with.

options
A options dictionary. You can pass any of the keys defined in "`RAW Image Options`" (page 60) along with the appropriate value. You should provide a source type identifier hint key (`kCGImageSourceTypeIdentifierHint`) and the appropriate source type value to help the decoder determine the file type. Otherwise it's possible to obtain incorrect results. See the Discussion for an example

Return Value
A `CIFilter` object.

Discussion
After calling this method, the `CIFilter` object returns a `CIImage` object that is properly processed similar to images retrieved using the `outputImage` key.

Here is an example of adding a source type identifier key-value pair to the options dictionary:

```
[opts setObject:(id)CGImageSourceGetTypeWithExtension ((CFStringRef)[[url path]
 pathExtension])
         forKey:(id)kCGImageSourceTypeIdentifierHint];
```

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterWithImageURL:options:` (page 39)

Declared In
`CIRAWFilter.h`

filterWithImageURL:options:

Returns a `CIFilter` object initialized with data from a RAW image file.

```
+ (CIFilter *)filterWithImageURL:(NSURL *)url options:(NSDictionary *)options;
```

Parameters

`url`

The location of a RAW image file.

`options`

An options dictionary. You can pass any of the keys defined in "`RAW Image Options`" (page 60).

Return Value
A `CIFilter` object.

Discussion
After calling this method, the `CIFilter` object returns a `CIImage` object that is properly processed similar to images retrieved using the `outputImage` key.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterWithImageData:options:` (page 38)

Related Sample Code
CIRAWFilterSample

Declared In
`CIRAWFilter.h`

filterWithName:

Creates a `CIFilter` object for a specific kind of filter.

```
+ (CIFilter *)filterWithName:(NSString *)name
```

Parameters

name

The name of the filter.

Return Value

A `CIFilter` object whose input values are undefined.

Discussion

You should call `setDefaults` (page 46) after you call this method or set values individually by calling `setValue:forKey`.

Availability

Mac OS X v10.4 and later.

See Also

`+ filterWithName:keysAndValues:` (page 40)

Related Sample Code

CIAnnotation

CIColorTracking

CocoaSlides

FunHouse

Reducer

Declared In

`CIFilter.h`

filterWithName:keysAndValues:

Creates a `CIFilter` object for a specific kind of filter and initializes the input values.

```
+ (CIFilter *)filterWithName:(NSString *)namekeysAndValues:key0, ...
```

Parameters

name

The name of the filter.

key0

A list of key-value pairs to set as input values to the filter. Each key is a constant that specifies the name of the input value to set, and must be followed by a value. You signal the end of the list by passing a `nil` value.

Return Value

A `CIFilter` object whose input values are initialized.

Availability

Mac OS X v10.4 and later.

See Also

`+ filterWithName:` (page 39)

Related Sample Code

CIAnnotation

CIBevelSample

CIColorTracking
CIMicroPaint
CITransitionSelectorSample

Declared In
`CIFilter.h`

localizedDescriptionForFilterName:

Returns the localized description of a filter for display in the user interface.

```
+ (NSString *)localizedDescriptionForFilterName:(NSString *)filterName
```

Parameters

filterName

The filter name.

Return Value
The localized description of the filter.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CIFilter.h`

localizedNameForCategory:

Returns the localized name for the specified filter category.

```
+ (NSString *)localizedNameForCategory:(NSString *)category
```

Parameters

category

A filter category.

Return Value
The localized name for the filter category.

Availability
Mac OS X v10.4 and later.

Related Sample Code
FunHouse

Declared In
`CIFilter.h`

localizedNameForFilterName:

Returns the localized name for the specified filter name.

```
+ (NSString *)localizedNameForFilterName:(NSString *)filterName
```

Parameters

filterName

A filter name.

Return Value

The localized name for the filter.

Availability

Mac OS X v10.4 and later.

Related Sample Code

AnimatedTableView

FunHouse

QTRecorder

Declared In

`CIFilter.h`

localizedReferenceDocumentationForFilterName:

Returns the location of the localized reference documentation that describes the filter.

```
+ (NSURL *)localizedReferenceDocumentationForFilterName:(NSString *)filterName
```

Parameters

filterName

The filter name.

Return Value

A URL that specifies the location of the localized documentation, or `nil` if the filter does not provide localized reference documentation.

Discussion

The URL can be a local file or a remote document on a web server. Because filters created prior to Mac OS X v10.5 could return `nil`, you should be make sure that your code handles this case gracefully.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CIFilter.h`

registerFilterName:constructor:classAttributes:

Publishes a custom filter that is not packaged as an image unit.

```
+ (void)registerFilterName:(NSString *)name constructor:(id)anObject
    classAttributes:(NSDictionary *)attributes
```

Parameters

name

A string object that specifies the name of the filter you want to publish.

anObject

A constructor object that implements the `filterWithName` method.

attributes

A dictionary that contains the class display name and filter categories attributes along with the appropriate value for each attributes. That is, the `kCIAttributeFilterDisplayName` (page 47) attribute and a string that specifies the display name, and the `kCIAttributeFilterCategories` (page 48) and an array that specifies the categories to which the filter belongs (such as `kCICategoryStillImage` (page 54) and `kCICategoryDistortionEffect` (page 52)). All other attributes for the filter should be returned by the custom `attributes` method implement by the filter.

Discussion

In most cases you don't need to use this method because the preferred way to register a custom filter that you write is to package it as an image unit. You do not need to use this method for a filter packaged as an image unit because you register your filter using the `CIPlugInRegistration` protocol. (See *Core Image Programming Guide* for additional details.)

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIHazeFilterSample

Declared In

`CIFilter.h`

Instance Methods

apply:

Produces a `CIImage` object by applying a kernel function.

```
- (CIImage *)apply:(CIKernel *)k, ...
```

Parameters

k

A `CIKernel` object that contains a kernel function.

A list of arguments to supply to the kernel function. The supplied arguments must be type-compatible with the function signature of the kernel function. The list of arguments must be terminated by the `nil` object.

Discussion

For example, if the kernel function has this signature:

```
kernel vec4 brightenEffect (sampler src, float k)
```

You would supply two arguments after the `k` argument to the `apply:k, ..` method. In this case, the first argument must be a sampler and the second a floating-point value. For more information on kernels, see *Core Image Kernel Language Reference*.

Availability
Mac OS X v10.4 and later.

See Also
`- apply:arguments:options:` (page 44)

Declared In
`CIFilter.h`

apply:arguments:options:

Produces a *`CIImage`* object by applying arguments to a kernel function and using options to control how the kernel function is evaluated.

```
- (CIImage *)apply:(CIKernel *)k arguments:(NSArray *)args options:(NSDictionary
    *)dict
```

Parameters

`k`

A `CIKernel` object that contains a kernel function.

`args`

The arguments that are type compatible with the function signature of the kernel function.

`dict`

A dictionary that contains options (key-value pairs) to control how the kernel function is evaluated.

Return Value
The `CIImage` object produced by a filter.

Discussion
You can pass any of the following keys in the dictionary:

- `kCIApplyOptionExtent` specifies the size of the produced image. The associated value is a four-element array (`NSArray`) that specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width, and height.
- `kCIApplyOptionDefinition` specifies the domain of definition (DOD) of the produces image. The associated value is either a Core Image filter shape or a four-element array (`NSArray`) that specifies a rectangle.
- `kCIApplyOptionUserInfo` specifies to retain the associated object and pass it to any callbacks invoked for that filter.

Availability
Mac OS X v10.4 and later.

See Also
`- apply:` (page 43)

Declared In
`CIFilter.h`

attributes

Returns a dictionary of key-value pairs that describe the filter.

```
- (NSDictionary *)attributes
```

Return Value

A dictionary that contains a key for each input and output parameter for the filter. Each key is a dictionary that contains all the attributes of an input or output parameter.

Discussion

For example, the attributes dictionary for the `CIColorControls` filter contains the following information:

```
CIColorControls:
{
    CIAttributeFilterCategories = (
        CICategoryColorAdjustment,
        CICategoryVideo,
        CICategoryStillImage,
        CICategoryInterlaced,
        CICategoryNonSquarePixels,
        CICategoryBuiltIn
    );
    CIAttributeFilterDisplayName = "Color Controls";
    CIAttributeFilterName = CIColorControls;
    inputBrightness = {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 0;
        CIAttributeIdentity = 0;
        CIAttributeMin = -1;
        CIAttributeSliderMax = 1;
        CIAttributeSliderMin = -1;
        CIAttributeType = CIAttributeTypeScalar;
    };
    inputContrast = {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 1;
        CIAttributeIdentity = 1;
        CIAttributeMin = 0.25;
        CIAttributeSliderMax = 4;
        CIAttributeSliderMin = 0.25;
        CIAttributeType = CIAttributeTypeScalar;
    };
    inputImage = {CIAttributeClass = CIImage; };
    inputSaturation = {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 1;
        CIAttributeIdentity = 1;
        CIAttributeMin = 0;
        CIAttributeSliderMax = 3;
        CIAttributeSliderMin = 0;
        CIAttributeType = CIAttributeTypeScalar;
    };
    outputImage = {CIAttributeClass = CIImage; };
}
```

Availability

Mac OS X v10.4 and later.

Related Sample Code
CIRAWFilterSample
CITransitionSelectorSample2
FunHouse
ImageApp

Declared In
`CIFilter.h`

inputKeys

Returns an array that contains the names of the input parameters to the filter.

```
- (NSArray *)inputKeys
```

Return Value
An array that contains the names of all input parameters to the filter.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIRAWFilterSample
CITransitionSelectorSample2
FunHouse

Declared In
`CIFilter.h`

outputKeys

Returns an array that contains the names of the output parameters for the filter.

```
- (NSArray *)outputKeys
```

Return Value
An array that contains the names of all output parameters from the filter.

Availability
Mac OS X v10.4 and later.

Declared In
`CIFilter.h`

setDefaults

Sets all input values for a filter to default values.

```
- (void)setDefaults
```

Discussion
Input values whose default values are not defined are left unchanged.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation
CIFilterGeneratorTest
CocoaSlides
CoreAnimationKioskStyleMenu
Reducer

Declared In
`CIFilter.h`

Constants

Filter Attribute Keys

Attributes for a filter and its parameters.

```
extern NSString *kCIAttributeFilterName;
extern NSString *kCIAttributeFilterDisplayName;
extern NSString *kCIAttributeDescription;
extern NSString *kCIAttributeReferenceDocumentation;
extern NSString *kCIAttributeFilterCategories;
extern NSString *kCIAttributeClass;
extern NSString *kCIAttributeType;
extern NSString *kCIAttributeMin;
extern NSString *kCIAttributeMax;
extern NSString *kCIAttributeSliderMin;
extern NSString *kCIAttributeSliderMax;
extern NSString *kCIAttributeDefault;
extern NSString *kCIAttributeIdentity;
extern NSString *kCIAttributeName;
extern NSString *kCIAttributeDisplayName;
```

Constants

`kCIAttributeFilterName`

The filter name, specified as an `NSString` object.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeFilterDisplayName`

The localized version of the filter name that is displayed in the user interface.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeDescription`
The localized description of the filter. This description should inform the end user what the filter does and be short enough to display in the user interface for the filter. It is not intended to be technically detailed.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIAttributeReferenceDocumentation`
The localized reference documentation for the filter. The reference should provide developers with technical details.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIAttributeFilterCategories`
An array of filter category keys that specifies all the categories in which the filter is a member.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeClass`
The class of the input parameter for a filter. If you are writing an image unit (see *Image Unit Tutorial*), Core Image supports only these classes for nonexecutable image units: `CIColor`, `CIVector`, `CIImage`, and `NSNumber` only. Executable image units may have input parameters of any class, but Core Image does not generate an automatic user interface for custom classes (see `CIFilter(IKFilterUIAddition)`).

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeType`
The attribute type.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeMin`
The minimum value for a filter parameter, specified as a floating-point value.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeMax`
The maximum value for a filter parameter, specified as a floating-point value.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeSliderMin`
The minimum value, specified as a floating-point value, to use for a slider that controls input values for a filter parameter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeSliderMax`
  The maximum value, specified as a floating-point value, to use for a slider that controls input values for a filter parameter.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCIAttributeDefault`
  The default value, specified as a floating-point value, for a filter parameter.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCIAttributeIdentity`
  If supplied as a value for a parameter, the parameter has no effect on the input image.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCIAttributeName`
  The name of the attribute.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

`kCIAttributeDisplayName`
  The localized display name of the attribute.

  Available in Mac OS X v10.4 and later.

  Declared in `CIFilter.h`.

Discussion

Attribute keys are used for the attribute dictionary of a filter. Most entries in the attribute dictionary are optional. The attribute `CIAttributeFilterName` is mandatory. For a parameter, the attribute `kCIAttributeClass` is mandatory.

A parameter of type `NSNumber` does not necessarily need the attributes `kCIAttributeMin` and `kCIAttributeMax`. These attributes are not present when the parameter has no upper or lower bounds. For example, the Gaussian blur filter has a radius parameter with a minimum of `0` but no maximum value to indicate that all nonnegative values are valid.

Declared In

`CIFilter.h`

Data Type Attributes

Numeric data types.

```
extern NSString *kCIAttributeTypeTime;
extern NSString *kCIAttributeTypeScalar;
extern NSString *kCIAttributeTypeDistance;
extern NSString *kCIAttributeTypeAngle;
extern NSString *kCIAttributeTypeBoolean;
extern NSString *kCIAttributeTypeInteger;
extern NSString *kCIAttributeTypeCount;
```

Constants

`kCIAttributeTypeTime`

A parametric time for transitions, specified as a floating-point value in the range of `0.0` to `1.0`.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeScalar`

A scalar value.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeDistance`

A distance.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeAngle`

An angle.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeBoolean`

A Boolean value.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeInteger`

An integer value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeCount`

A positive integer value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Declared In

`CIFilter.h`

Vector Quantity Attributes

Vector data types.

```
extern NSString *kCIAttributeTypePosition;
extern NSString *kCIAttributeTypeOffset;
extern NSString *kCIAttributeTypePosition3;
extern NSString *kCIAttributeTypeRectangle
```

Constants

`kCIAttributeTypePosition`

A two-dimensional location in the working coordinate space. (A 2-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeOffset`

An offset. (A 2-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypePosition3`

A three-dimensional location in the working coordinate space. (A 3-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeRectangle`

A Core Image vector that specifies the *x* and *y* values of the rectangle origin, and the width (*w*) and height (*h*) of the rectangle. The vector takes the form [*x*, *y*, *w*, *h*]. (A 4-element vector type.)

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Declared In

`CIFilter.h`

Color Attribute Keys

Color types.

```
extern NSString *kCIAttributeTypeOpaqueColor;
extern NSString *kCIAttributeTypeGradient;
```

Constants

`kCIAttributeTypeOpaqueColor`

A Core Image color (`CIColor` object) that specifies red, green, and blue component values. Use this key for colors with no alpha component. If the key is not present, Core Image assumes color with alpha.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIAttributeTypeGradient`

An n-by-1 gradient image used to describe a color ramp.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Declared In

`CIFilter.h`

Filter Category Keys

Categories of filters.

```
extern NSString *kCICategoryDistortionEffect;
extern NSString *kCICategoryGeometryAdjustment;
extern NSString *kCICategoryCompositeOperation;
extern NSString *kCICategoryHalftoneEffect;
extern NSString *kCICategoryColorAdjustment;
extern NSString *kCICategoryColorEffect;
extern NSString *kCICategoryTransition;
extern NSString *kCICategoryTileEffect;
extern NSString *kCICategoryGenerator;
extern NSString *kCICategoryReduction;
extern NSString *kCICategoryGradient;
extern NSString *kCICategoryStylize;
extern NSString *kCICategorySharpen;
extern NSString *kCICategoryBlur;
extern NSString *kCICategoryVideo;
extern NSString *kCICategoryStillImage;
extern NSString *kCICategoryInterlaced;
extern NSString *kCICategoryNonSquarePixels;
extern NSString *kCICategoryHighDynamicRange ;
extern NSString *kCICategoryBuiltIn;
extern NSString *kCICategoryFilterGenerator;
```

Constants

`kCICategoryDistortionEffect`

A filter that reshapes an image by altering its geometry to create a 3D effect. Using distortion filters, you can displace portions of an image, apply lens effects, make a bulge in an image, and perform other operation to achieve an artistic effect.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryGeometryAdjustment`

A filter that changes the geometry of an image. Some of these filters are used to warp an image to achieve an artistic effects, but these filters can also be used to correct problems in the source image. For example, you can apply an affine transform to straighten an image that is rotated with respect to the horizon.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryCompositeOperation`

A filter operates on two image sources, using the color values of one image to operate on the other. Composite filters perform computations such as computing maximum values, minimum values, and multiplying values between input images. You can use compositing filters to add effects to an image, crop an image, and achieve a variety of other effects.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryHalftoneEffect`

A filter that simulates a variety of halftone screens, to mimic the halftone process used in print media. The output of these filters has the familiar "newspaper" look of the various dot patterns. Filters are typically named after the pattern created by the virtual halftone screen, such as circular screen or hatched screen.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryColorAdjustment`

A filter that changes color values. Color adjustment filters are used to eliminate color casts, adjust hue, and correct brightness and contrast. Color adjustment filters do not perform color management; ColorSync performs color management. You can use Quartz 2D to specify the color space associated with an image. For more information, see *Color Management Overview* and *Quartz 2D Programming Guide*.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryColorEffect`

A filter that modifies the color of an image to achieve an artistic effect. Examples of color effect filters include filters that change a color image to a sepia image or a monochrome image or that produces such effects as posterizing.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryTransition`

A filter that provides a bridge between two or more images by applying a motion effect that defines how the pixels of a source image yield to that of the destination image.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryTileEffect`

A filter that typically applies an effect to an image and then create smaller versions of the image (tiles), which are then laid out to create a pattern that's infinite in extent.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryGenerator`

A filter that generates a pattern, such as a solid color, a checkerboard, or a star shine. The generated output is typically used as input to another filter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryReduction`

A filter that reduces image data. These filters are used to solve image analysis problems.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCICategoryGradient`

A filter that generates a fill whose color varies smoothly. Exactly how color varies depends on the type of gradient—linear, radial, or Gaussian.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryStylize`

A filter that makes a photographic image look as if it was painted or sketched. These filters are typically used alone or in combination with other filters to achieve artistic effects.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategorySharpen`

A filter that sharpens images, increasing the contrast between the edges in an image. Examples of sharpen filters are unsharp mask and sharpen luminance.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryBlur`

A filter that softens images, decreasing the contrast between the edges in an image. Examples of blur filters are Gaussian blur and zoom blur.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryVideo`

A filter that works on video images.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryStillImage`

A filter that works on still images.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryInterlaced`

A filter that works on interlaced images.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryNonSquarePixels`

A filter that works on non-square pixels.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryHighDynamicRange`

A filter that works on high dynamic range pixels.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryBuiltIn`

A filter provided by Core Image. This distinguishes built-in filters from plug-in filters.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCICategoryFilterGenerator`

A filter created by chaining several filters together and then packaged as a `CIFilterGenerator` object.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Declared In
`CIFilter.h`

Options for Applying a Filter

Options that control the application of a Core Image filter.

```
extern NSString *kCIApplyOptionExtent;
extern NSString *kCIApplyOptionDefinition;
extern NSString *kCIApplyOptionUserInfo;
```

Constants

`kCIApplyOptionExtent`

The size of the produced image. The associated value is a four-element array (`NSArray`) that specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width and height.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIApplyOptionDefinition`

The domain of definition (DOD) of the produced image. The associated value is either a Core Image filter shape or a four-element array (`NSArray`) that specifies a rectangle.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

`kCIApplyOptionUserInfo`

Information needed by a callback. The associated value is an object that Core Image will pass to any callbacks invoked for that filter.

Available in Mac OS X v10.4 and later.

Declared in `CIFilter.h`.

Declared In
`CIFilter.h`

User Interface Control Options

Sets of controls for various user scenarios.

```
extern NSString *kCIUIParameterSet;
extern NSString *kCIUISetBasic;
extern NSString *kCIUISetIntermediate;
extern NSString *kCIUISetAdvanced;
extern NSString *kCIUISetDevelopment;
```

Constants

`kCIUIParameterSet`

The set of input parameters to use. The associated value can be `kCIUISetBasic` (page 56), `kCIUISetIntermediate` (page 56), `kCIUISetAdvanced` (page 56), or `kCIUISetDevelopment` (page 56).

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetBasic`

Controls that are appropriate for a basic user scenario, that is, the minimum of settings to control the filter.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetIntermediate`

Controls that are appropriate for an intermediate user scenario.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetAdvanced`

Controls that are appropriate for an advanced user scenario.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIUISetDevelopment`

Controls that should be visible only for development purposes.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Discussion

You can use these constants to specify the controls that you want associated with each user scenario. For example, for a filter that has many input parameters you can choose a small set of input parameters that the typical consumer can control and set the other input parameters to default values. For the same filter, however, you can choose to allow professional customers to control all the input parameters.

Declared In

`CIFIlter.h`

Filter Parameter Keys

Keys for input parameters to filters.

```
extern NSString *kCIOutputImageKey;
extern NSString *kCIInputBackgroundImageKey;
extern NSString *kCIInputImageKey;
extern NSString *kCIInputTimeKey;
extern NSString *kCIInputTransformKey;
extern NSString *kCIInputScaleKey;
extern NSString *kCIInputAspectRatioKey;
extern NSString *kCIInputCenterKey;
extern NSString *kCIInputRadiusKey;
extern NSString *kCIInputAngleKey;
extern NSString *kCIInputRefractionKey;
extern NSString *kCIInputWidthKey;
extern NSString *kCIInputSharpnessKey;
extern NSString *kCIInputIntensityKey;
extern NSString *kCIInputEVKey;
extern NSString *kCIInputSaturationKey;
extern NSString *kCIInputColorKey;
extern NSString *kCIInputBrightnessKey;
extern NSString *kCIInputContrastKey;
extern NSString *kCIInputGradientImageKey;
extern NSString *kCIInputMaskImageKey;
extern NSString *kCIInputShadingImageKey;
extern NSString *kCIInputTargetImageKey;
extern NSString *kCIInputExtentKey;
```

Constants

`kCIOutputImageKey`

A key for the `CIImage` object produced by a filter.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputBackgroundImageKey`

A key for the `CIImage` object to use as a background image.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputImageKey`

A key for the `CIImage` object to use as an input image. For filters that also use a background image, this key refers to the foreground image.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputTimeKey`

A key for z scalar value (`NSNumber`) that specifies a time.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputTransformKey`

A key for an `NSAffineTransform` object that specifies a transformation to apply.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputScaleKey`

A key for a scalar value (`NSNumber`) that specifies the amount of the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputAspectRatioKey`

A key for a scalar value (`NSNumber`) that specifies a ratio.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputCenterKey`

A key for a `CIVector` object that specifies the center of the area, as *x* and *y*- coordinates, to be filtered.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputRadiusKey`

A key for a scalar value (`NSNumber`) that specifies that specifies the distance from the center of an effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputAngleKey`

A key for a scalar value (`NSNumber`) that specifies an angle.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputRefractionKey`

A key for a scalar value (`NSNumber`) that specifies the index of refraction of the material (such as glass) used in the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputWidthKey`

A key for a scalar value (`NSNumber`) that specifies the width of the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputSharpnessKey`

A key for a scalar value (`NSNumber`) that specifies the amount of sharpening to apply.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputIntensityKey`

A key for a scalar value (`NSNumber`) that specifies an intensity value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputEVKey`

A key for a scalar value (`NSNumber`) that specifies how many F-stops brighter or darker the image should be.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputSaturationKey`
A key for a scalar value (`NSNumber`) that specifies the amount to adjust the saturation.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputColorKey`
A key for a `CIColor` object that specifies a color value.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputBrightnessKey`
A key for a scalar value (`NSNumber`) that specifies a brightness level.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputContrastKey`
A key for a scalar value (`NSNumber`) that specifies a contrast level.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputGradientImageKey`
A key for a `CIImage` object that specifies an environment map with alpha. Typically, this image contains highlight and shadow.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputMaskImageKey`
A key for a `CIImage` object to use as a mask.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputShadingImageKey`
A key for a `CIImage` object that specifies an environment map with alpha values. Typically this image contains highlight and shadow.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputTargetImageKey`
A key for a `CIImage` object that is the target image for a transition.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

`kCIInputExtentKey`
A key for a `CIVector` object that specifies a rectangle that defines the extent of the effect.

Available in Mac OS X v10.5 and later.

Declared in `CIFilter.h`.

Discussion

These keys represent some of the most commonly used input parameters. A filter can use other kinds of input parameters.

Declared In

`CIFIlter.h`

RAW Image Options

Options for creating a `CIFilter` object from RAW image data.

```
extern NSString * const kCIInputDecoderVersionKey;
extern NSString * const kCISupportedDecoderVersionsKey;
extern NSString * const kCIInputBoostKey;
extern NSString * const kCIInputNeutralChromaticityXKey;
extern NSString * const kCIInputNeutralChromaticityYKey;
extern NSString * const kCIInputNeutralTemperatureKey;
extern NSString * const kCIInputNeutralTintKey;
extern NSString * const kCIInputNeutralLocation;
extern NSString * const kCIInputScaleFactorKey;
extern NSString * const kCIInputAllowDraftModeKey;
extern NSString * const kCIInputIgnoreImageOrientationKey;
extern NSString * const kCIInputImageOrientationKey;
extern NSString * const kCIInputEnableSharpeningKey;
extern NSString * const kCIInputEnableChromaticNoiseTrackingKey;
extern NSString * const kCIInputBoostShadowAmountKey;
extern NSString * const kCIInputBiasKey;
```

Constants

`kCIInputDecoderVersionKey`

A key for the version number of the method to be used for decoding. A newly initialized object defaults to the newest available decoder version for the given image type. You can request an alternative, older version to maintain compatibility with older releases. Must be one of `kCISupportedDecoderVersions`, otherwise a `nil` output image is generated. The associated value must be an `NSNumber` object that specifies an integer value in range of `0` to the current decoder version. When you request a specific version of the decoder, Core Image produces an image that is *visually* the same across different versions of the operating system. Core Image, however, does not guarantee that the same bits are produced across different versions of the operating system. That's because the rounding behavior of floating-point arithmetic can vary due to differences in compilers or hardware. Note that this option has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCISupportedDecoderVersionsKey`

A key for the supported decoder versions. The associated value is an `NSArray` object that contains all supported decoder versions for the given image type, sorted in increasingly newer order. Each entry is an `NSDictionary` object that contains key-value pairs. All entries represent a valid version identifier that can be passed as the `kCIDecoderVersion` value for the key `kCIDecoderMethodKey`. Version values are read-only; attempting to set this value raises an exception. Currently, the only defined key is `@"version"` which has as its value an `NSString` that uniquely describing a given decoder version. This string might not be suitable for user interface display..

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputBoostKey`

A key for the the amount of boost to apply to an image. The associated value is a floating-point value packaged as an `NSNumber` object. The value must be in the range of `0...1`. A value of `0` indicates no boost, that is, a linear response. The default value is `1`, which indicates full boost.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralChromaticityXKey`

The x value of the chromaticity. The associated value is a floating-point value packaged as an `NSNumber` object. You can query this value to get the current x value for neutral x, y.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralChromaticityYKey`

The y value of the chromaticity. The associated value is a floating-point value packaged as an `NSNumber` object. You can query this value to get the current y value for neutral x, y.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralTemperatureKey`

A key for neutral temperature. The associated value is a floating-point value packaged as an `NSNumber` object. You can query this value to get the current temperature value.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralTintKey`

A key for the neutral tint. The associated value is a floating-point value packaged as an `NSNumber` object. Use this key to set or fetch the temperature and tint values. You can query this value to get the current tint value.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputNeutralLocationKey`

A key for the neutral position. Use this key to set the location in geometric coordinates of the unrotated output image that should be used as neutral. You cannot query this value; it is undefined for reading. The associated value is a two-element `CIVector` object that specifies the location (`x`, `y`).

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputScaleFactorKey`

A key for the scale factor. The associated value is a floating-point value packaged as an `NSNumber` object that specifies the desired scale factor at which the image will be drawn. Setting this value can greatly improve the drawing performance. A value of `1` is the identity. In some cases, if you change the scale factor and enable draft mode, performance can decrease. See `kCIAllowDraftModeKey`.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

`kCIInputAllowDraftModeKey`

A key for allowing draft mode. The associated value is a Boolean value packaged as an `NSNumber` object. It’s best not to use draft mode if the image needs to be drawn without draft mode at a later time, because changing the value from `YES` to `NO` is an expensive operation. If the optional scale factor is smaller than a certain value, additionally setting draft mode can improve image decoding speed without any perceivable loss of quality. However, turning on draft mode does not have any effect if the scale factor is not below this threshold.

Available in Mac OS X v10.5 and later.

Declared in `CIRAWFilter.h`.

kCIInputIgnoreImageOrientationKey

A key for specifying whether to ignore the image orientation. The associated value is a Boolean value packaged as an NSNumber object. The default value is NO. An image is usually loaded in its proper orientation, as long as the associated metadata records its orientation. For special purposes you might want to load the image in its physical orientation. The exact meaning of "physical orientation" is dependent on the specific image.

Available in Mac OS X v10.5 and later.

Declared in CIRAWFilter.h.

kCIInputImageOrientationKey

A key for the image orientation. The associated value is an integer value packaged as an NSNumber object. Valid values are in range 1...8 and follow the EXIF specification. The value is disregarded when the kCIIgnoreImageOrientationKey flag is set. You can change the orientation of the image by overriding this value. By changing this value you can easily rotate an image in 90-degree increments.

Available in Mac OS X v10.5 and later.

Declared in CIRAWFilter.h.

kCIInputEnableSharpeningKey

A key for the sharpening state. The associated value must be an NSNumber object that specifies a BOOL value (YES or NO). The default is YES. This option has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in CIRAWFilter.h.

kCIInputEnableChromaticNoiseTrackingKey

A key for progressive chromatic noise tracking (based on ISO and exposure time). The associated value must be an NSNumber object that specifies a BOOL value (YES or NO). The default is YES. This option has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in CIRAWFilter.h.

kCIInputBoostShadowAmountKey

A key for the amount to boost the shadow areas of the image. The associated value must be an NSNumber object that specifies floating-point value. The value has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in CIRAWFilter.h.

kCIInputBiasKey

A key for the simple bias value to use along with the exposure adjustment (kCIInputEVKey). The associated value must be an NSNumber object that specifies floating-point value. The value has no effect if the image used for initialization is not RAW.

Available in Mac OS X v10.5 and later.

Declared in CIRAWFilter.h.

Discussion

You can also use the key kCIInputEVKey for RAW images.

Declared In

CIRAWFilter.h

CIFilter Core Animation Additions

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/QuartzCore.framework
Declared in	CACIFilterAdditions.h
Companion guides	Core Animation Programming Guide Core Animation Cookbook Core Image Programming Guide

Overview

Core Animation adds two additional properties to the `CIFilter` class. These properties are accessible through key-value coding as well as the properties declared below.

Tasks

Naming Filter Instances

`name` (page 64) *property*
The name of the receiver.

Enabling Filter Instances

`enabled` (page 64) *property*
Determines if the receiver is enabled. Animatable.

`– isEnabled` (page 64)
A synthesized accessor for the `enabled` (page 64) property.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

enabled

Determines if the receiver is enabled. Animatable.

```
@property BOOL enabled
```

Discussion
The receiver is applied to its input when this property is set to `YES`. Default is `YES`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CACIFilterAdditions.h`

name

The name of the receiver.

```
@property(copy) NSString *name
```

Discussion
Default is `nil`. Each `CIFilter` instance can have an assigned name. The name is used to construct key paths to the filter’s attributes. For example, if a `CIFilter` instance has the name “`myExposureFilter`”, you refer to attributes of the filter using a key path such as “`filters.myExposureFilter.inputEV`”. Layer animations may also access filter attributes via these key paths.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CACIFilterAdditions.h`

Instance Methods

isEnabled

A synthesized accessor for the `enabled` (page 64) property.

```
- (BOOL)isEnabled
```

See Also
`@property enabled` (page 64)

CIFilter Image Kit Additions

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	System/Library/Frameworks/Quartz.framework/ImageKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	IIKFilterUI.h
Companion guide	Core Image Programming Guide

Overview

This Image Kit addition to the `CIFilter` class, introduced in Mac OS X v10.5, consists of one method and a set of constants that generate a view with input parameter controls for a Core Image filter. Using this method, it is easier for applications to present a user interface for a filter than it was in Mac OS X v10.4. Then, applications could create a filter user interface only by analyzing the keys and key attributes of a filter and then writing the code to implement the user interface.

You use the `viewForUIConfiguration:excludedKeys:` method to request a view from Core Image. The view is a subclass of the `NSView` class so that you can insert it easily into any other view as a subview or into an `NSWindow` object as a content view. Core Image automatically generates the view for you unless you implement the `IKFilterCustomUIProvider` protocol, in which case calling `viewForUIConfiguration:excludedKeys:` causes Core Image to provide your custom view.

Tasks

Creating a View for a Filter

`- viewForUIConfiguration:excludedKeys:` (page 66)
Returns a filter view for the filter.

Instance Methods

viewForUIConfiguration:excludedKeys:

Returns a filter view for the filter.

```
-(IKFilterUIView*)viewForUIConfiguration:(NSDictionary*)inUIConfiguration
    excludedKeys:(NSArray*)inKeys;
```

Parameters

inUIConfiguration

A dictionary that contains values for the IKUISizeFlavor and kCIUIParameterSet keys. See "User Interface Options" (page 67) for the constants that you can provide as values for IKUISizeFlavor. For kCIUIParameterSet you can provide one of the following values: kCIUISetBasic (page 56), kCIUISetIntermediate (page 56), kCIUISetAdvanced (page 56), or kCIUISetDevelopment (page 56). When you request a user interface for a parameter set, all keys for that set and below are included. For example, the advanced set consists of all parameters in the basic, intermediate and advanced sets. The development set should contain parameters that are either experimental or for debugging purposes. You should use them only during the development of filters and client applications, and not in a shipping product.

inKeys

An array of the input keys for which you do *not* want to provide a user interface. Pass nil if you want all input keys to be represented in the user interface.

Return Value

An IKFilterUIView object. You should retain the view as long as you need it, but make sure to release it when you no longer need it as the view is retaining the filter.

Discussion

Calling this method to receive a view for a filter causes the CIFilter class to invoke the provideViewForUIConfiguration:excludedKeys: method. If you override provideViewForUIConfiguration:excludedKeys: the user interface is created by your filter subclass . Otherwise, Core Image automatically generates the user interface based on the filter keys and attributes.

The algorithm used to lay out the controls for a filter operates in a manner similar to the Core Image Fun House application (/Developer/Applications/Graphics Tools/). Applications can retrieve a view whose control sizes complement the size of user interface elements already used in the application. It is also possible to choose which filter input parameters appear in the view. Consumer applications, for example, may want to show a small, basic set of input parameters whereas professional applications may want to provide access to all input parameters.

The controls in the view use bindings to set the values of the filter. See *Cocoa Bindings Programming Topics* if you are unfamiliar with bindings.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

CIFilterGeneratorTest

CIRAWFilterSample

Declared In
IKFilterUI.h

Constants

User Interface Options

Keys or values for the size of the input parameter controls for a filter view.

```
NSString *IKUISizeFlavor;
NSString *IKUISizeMini;
NSString *IKUISizeSmall;
NSString *IKUISizeRegular;
NSString *IKUImaxSize;
NSString *IKUIFlavorAllowFallback;
```

Constants

`IKUISizeFlavor`

A key for the size of the controls in a filter view. The associated value can be `IKUISizeMini`, `IKUISizeSmall`, or `IKUISizeRegular`.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUISizeMini`

Controls whose size is mini, as defined by Interface Builder 2.5.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUISizeSmall`

Controls whose size is small, as defined by Interface Builder 2.5.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUISizeRegular`

Controls whose size is regular or normal, as defined by Interface Builder 2.5.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUImaxSize`

Controls whose dimensions are the maximum allowable for the filter view. A width or height of `0` indicates that that dimension of the view is not restricted. If the size requested is too small, the filter is expected to return a view as small as possible. It is up to the client to verify that the returned view fits into the context.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

`IKUIFlavorAllowFallback`

Substitute controls of another size. The associated value is a Boolean value. If the filter cannot provide a view for the requested size and a fallback is allowed, the filter can use controls of a different size.

Available in Mac OS X v10.5 and later.

Declared in `IKFilterUI.h`.

Declared In

`IKFilterUI.h`

CIFilterGenerator Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIFilterGenerator.h
Availability	Mac OS X v10.5 and later
Companion guides	Core Image Programming Guide Core Image Filter Reference
Related sample code	CIFilterGeneratorTest

Overview

The `CIFilterGenerator` class provides methods for creating a `CIFilter` object by chaining together existing `CIFilter` objects to create complex effects. (A **filter chain** refers to the `CIFilter` objects that are connected in the `CIFilterGenerator` object.) The complex effect can be encapsulated as a `CIFilterGenerator` object and saved as a file so that it can be used again. The **filter generator file** contains an archived instance of all the `CIFilter` objects that are chained together.

Any filter generator files that you copy to `/Library/Graphics/Image Units/` are loaded when any of the loading methods provided by the `CIPlugIn` class are invoked. A `CIFilterGenerator` object is registered by its filename or, if present, by a class attribute that you supply in its description.

You can create a `CIFilterGenerator` object programmatically, using the methods provided by the `CIFilterGenerator` class, or by using the editor view provided by Core Image (see *CIFilter Image Kit Additions*).

Tasks

Creating Filter Generator Objects

`+ filterGenerator` (page 71)
Creates and returns an empty filter generator object.

`+ filterGeneratorWithContentsOfURL:` (page 71)
Creates and returns a filter generator object and initializes it with the contents of a filter generator file.

Initializing a Filter Generator Object

`- initWithContentsOfURL:` (page 75)
Initializes a filter generator object with the contents of a filter generator file.

Connecting and Disconnecting Objects

`- connectObject:withKey:toObject:withKey:` (page 72)
Adds an object to the filter chain.

`- disconnectObject:withKey:toObject:withKey:` (page 73)
Removes the connection between two objects in the filter chain.

Managing Exported Keys

`- exportedKeys` (page 73)
Returns an array of the exported keys.

`- exportKey:fromObject:withName:` (page 74)
Exports an input or output key of an object in the filter chain.

`- removeExportedKey:` (page 76)
Removes a key that was previously exported.

`- setAttributes:forExportedKey:` (page 76)
Sets a dictionary of attributes for an exported key.

Setting and Getting Class Attributes

`- classAttributes` (page 72)
Retrieves the class attributes associated with a filter.

`- setClassAttributes:` (page 77)
Seta the class attributes for a filter.

Archiving a Filter Generator Object

`- writeToURL:atomically:` (page 77)
Archives a filter generator object to a filter generator file.

Registering a Filter Chain

`- registerFilterName:` (page 75)
: Registers the name associated with a filter chain.

Creating a Filter from a Filter Chain

`- filter` (page 75)
: Creates a filter object based on the filter chain.

Class Methods

filterGenerator

Creates and returns an empty filter generator object.

```
+ (CIFilterGenerator *)filterGenerator
```

Return Value
A `CIFilterGenerator` object.

Discussion
You use the returned object to connect two or more `CIFilter` objects and input images. It is also valid to have only one `CIFilter` object in a filter generator.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterGeneratorWithContentsOfURL:` (page 71)

Related Sample Code
CIFilterGeneratorTest

Declared In
`CIFilterGenerator.h`

filterGeneratorWithContentsOfURL:

Creates and returns a filter generator object and initializes it with the contents of a filter generator file.

```
+ (CIFilterGenerator *)filterGeneratorWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL
: The location of a filter generator file.

Return Value
A `CIFilterGenerator` object; returns `nil` if the file can't be read.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterGenerator` (page 71)

Related Sample Code
CIFilterGeneratorTest

Declared In
`CIFilterGenerator.h`

Instance Methods

classAttributes

Retrieves the class attributes associated with a filter.

```
- (NSDictionary *)classAttributes
```

Return Value
An `NSDictionary` object that contains the class attributes for a filter, or `nil` if attributes are not set for the filter.

Discussion
For more information about class attributes for a filter, see *Core Image Programming Guide* and the filter attributes key constants defined in *CIFilter Class Reference*.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setClassAttributes:` (page 77)

Declared In
`CIFilterGenerator.h`

connectObject:withKey:toObject:withKey:

Adds an object to the filter chain.

```
- (void)connectObject:(id)sourceObject withKey:(NSString *)sourceKey
    toObject:(id)targetObject withKey:(NSString *)targetKey
```

Parameters

sourceObject

A `CIFilter` object, a `CIImage` object, or a the path (an `NSString` or `NSURL` object) to an image.

sourceKey

The key that specifies the source object. For example, if the source is the output image of a filter, pass the `outputImage` key. Pass `nil` if the source object is used directly.

targetObject

The object that to link the source object to.

targetKey

The key that specifies the target for the source. For example, if you are connecting the source to the input image of a `CIFilter` object, you would pass the `inputImage` key.

Availability

Available in Mac OS X v10.5 and later.

See Also

`– disconnectObject:withKey:toObject:withKey:` (page 73)

Related Sample Code

CIFilterGeneratorTest

Declared In

`CIFilterGenerator.h`

disconnectObject:withKey:toObject:withKey:

Removes the connection between two objects in the filter chain.

```
- (void)disconnectObject:(id)sourceObject withKey:(NSString *)key
    toObject:(id)targetObject withKey:(NSString *)targetKey
```

Parameters

sourceObject

A `CIFilter` object, a `CIImage` object, or a the path (an `NSString` or `NSURL` object) to an image.

sourceKey

The key that specifies the source object. Pass `nil` if the source object is used directly.

targetObject

The object that you want to disconnect the source object from.

targetKey

The key that specifies the target that the source object is currently connected to.

Availability

Available in Mac OS X v10.5 and later.

See Also

`– connectObject:withKey:toObject:withKey:` (page 72)

Declared In

`CIFilterGenerator.h`

exportedKeys

Returns an array of the exported keys.

```
- (NSDictionary *)exportedKeys
```

Return Value

An array of dictionaries that describe the exported key and target object. See `kCIFilterGeneratorExportedKey` (page 78), `kCIFilterGeneratorExportedKeyTargetObject` (page 78), and `kCIFilterGeneratorExportedKey` (page 78) for keys used in the dictionary.

Discussion

This method returns the keys that you exported using the `exportKey:fromObject:withName:` (page 74) method or that were exported before being written to the file from which you read the filter chain.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- exportKey:fromObject:withName:` (page 74)

Declared In

`CIFilterGenerator.h`

exportKey:fromObject:withName:

Exports an input or output key of an object in the filter chain.

```
- (void)exportKey:(NSString *)key fromObject:(id)targetObject withName:(NSString
    *)exportedKeyName
```

Parameters

key

The key to export from the target object (for example, `inputImage`).

targetObject

The object associated with the key (for example, the filter).

exportedKeyName

A unique name to use for the exported key. Pass `nil` to use the original key name.

Discussion

When you create a `CIFilter` object from a `CIFilterGenerator` object, you might want the filter client to be able to set some of the parameters associated with the filter chain. You can make a parameter settable by exporting the key associated with the parameter. If the exported key represents an input parameter of the filter, the key is exported as an input key. If the key represents an output parameter, it is exported as an output key.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- exportedKeys` (page 73)

`- setAttributes:forExportedKey:` (page 76)

`- removeExportedKey:` (page 76)

Related Sample Code

CIFilterGeneratorTest

Declared In

`CIFilterGenerator.h`

filter

Creates a filter object based on the filter chain.

```
- (CIFilter *)filter
```

Return Value
A `CIFilter` object.

Discussion
The topology of the filter chain is immutable, meaning that any changes you make to the filter chain are not reflected in the filter. The returned filer has the input an output keys that are exported.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
CIFilterGeneratorTest

Declared In
`CIFilterGenerator.h`

initWithContentsOfURL:

Initializes a filter generator object with the contents of a filter generator file.

```
- (id)initWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

The location of a filter generator file.

Return Value
The initialized `CIFilterGenerator` object. Returns `nil` if the file can't be read.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ filterGenerator` (page 71)
`+ filterGeneratorWithContentsOfURL:` (page 71)

Declared In
`CIFilterGenerator.h`

registerFilterName:

Registers the name associated with a filter chain.

```
- (void)registerFilterName:(NSString *)name
```

Parameters

name

A unique name for the filter chain you want to register.

Discussion

This method allows you to register the filter chain as a named filter in the Core Image filter repository. You can then create a `CIFilter` object from it using the the `filterWithName:` (page 39) method of the `CIFilter` class.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CIFilterGenerator.h`

removeExportedKey:

Removes a key that was previously exported.

```
- (void)removeExportedKey:(NSString *)exportedKeyName
```

Parameters

exportedKeyName

The name of the key you want to remove.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- exportKey:fromObject:withName:` (page 74)

Declared In

`CIFilterGenerator.h`

setAttributes:forExportedKey:

Sets a dictionary of attributes for an exported key.

```
- (void)setAttributes:(NSDictionary *)attributes forExportedKey:(NSString *)key
```

Parameters

attributes

A dictionary that describes the attributes associated with the specified key.

key

The exported key whose attributes you want to set.

Discussion

By default, the exported key inherits the attributes from its original key and target object. You can use this method to change one or more of the existing attributes for the key, such as the default value or maximum value. For more information on attributes, see *CIFilter Class Reference* and *Core Image Programming Guide*.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- exportedKeys` (page 73)
`- exportKey:fromObject:withName:` (page 74)

Declared In
`CIFilterGenerator.h`

setClassAttributes:

Seta the class attributes for a filter.

```
- (void)setClassAttributes:(NSDictionary *)attributes
```

Parameters

attributes

An `NSDictionary` object that contains the class attributes for a filter For information on the required attributes, see *CIFilter Class Reference* and *Core Image Programming Guide*.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- classAttributes` (page 72)

Related Sample Code
CIFilterGeneratorTest

Declared In
`CIFilterGenerator.h`

writeToURL:atomically:

Archives a filter generator object to a filter generator file.

```
- (BOOL)writeToURL:(NSURL *)aURL atomically:(BOOL)flag
```

Parameters

aURL

A location for the file generator file.

flag

Pass `true` to specify that Core Image should create an interim file to avoid overwriting an existing file.

Return Value
Returns `true` if the the object is successfully archived to the file.

Discussion
Use this method to save your filter chain to a file for later use.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CIFilterGenerator.h`

Constants

Exported Keys

Keys for the exported parameters of a filter generator object.

```
extern NSString *const kCIFilterGeneratorExportedKey;
extern NSString *const kCIFilterGeneratorExportedKeyTargetObject;
extern NSString *const kCIFilterGeneratorExportedKeyName;
```

Constants

`kCIFilterGeneratorExportedKeyName`

The key (`CIFilterGeneratorExportedKeyName`) for the name used to export the `CIFilterGenerator` object. The associated value is a string that specifies a unique name for the filter generator object.

Available in Mac OS X v10.5 and later.

Declared in `CIFilterGenerator.h`.

`kCIFilterGeneratorExportedKey`

The key (`CIFilterGeneratorExportedKey`) for the exported parameter. The associated value is the key name of the parameter you are exporting, such as `inputRadius`.

Available in Mac OS X v10.5 and later.

Declared in `CIFilterGenerator.h`.

`kCIFilterGeneratorExportedKeyTargetObject`

The target object (`CIFilterGeneratorExportedKeyTargetObject`) for the exported key. The associated value is the name of the object, such as `CIMotionBlur`.

Available in Mac OS X v10.5 and later.

Declared in `CIFilterGenerator.h`.

Declared In

`CIFilterGenerator.h`

CIFilterShape Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIFilterShape.h
Availability	Mac OS X v10.4 and later
Companion guide	Core Image Programming Guide
Related sample code	CIAnnotation CIColorTracking

Overview

The `CIFilterShape` class describes the bounding shape of a filter and the domain of definition (DOD) of a filter operation. You use `CIFilterShape` objects in conjunction with Core Image classes, such as `CIFilter`, `CIKernel`, and `CISampler`, to create custom filters.

Tasks

Creating a Filter Shape

`+ shapeWithRect:` (page 80)
Creates a filter shape object and initializes it with a rectangle.

Initializing a Filter Shape

`- initWithRect:` (page 81)
Initializes a filter shape object with a rectangle.

Modifying a Filter Shape

- `insetByX:Y:` (page 81)
  Modifies a filter shape object so that it is inset by the specified x and y values.
- `intersectWith:` (page 81)
  Creates a filter shape object that represents the intersection of the current filter shape and the specified filter shape object.
- `intersectWithRect:` (page 82)
  Creates a filter shape that represents the intersection of the current filter shape and a rectangle.
- `transformBy:interior:` (page 82)
  Creates a filter shape that results from applying a transform to the current filter shape.
- `unionWith:` (page 83)
  Creates a filter shape that results from the union of the current filter shape and another filter shape object.
- `unionWithRect:` (page 83)
  Creates a filter shape that results from the union of the current filter shape and a rectangle.

Class Methods

shapeWithRect:

Creates a filter shape object and initializes it with a rectangle.

```
+ (id)shapeWithRect:(CGRect)r
```

Parameters

r

A rectangle. The filter shape object will contain the smallest integral rectangle specified by this argument.

Availability
Mac OS X v10.4 and later.

See Also
- `initWithRect:` (page 81)

Related Sample Code
CIAnnotation

Declared In
`CIFilterShape.h`

Instance Methods

initWithRect:

Initializes a filter shape object with a rectangle.

```
- (id)initWithRect:(CGRect)r
```

Parameters

r

A rectangle. Core Image uses the rectangle specified by integer parts of the values in the `CGRect` data structure.

Return Value

An initialized CIFilterShape object, or `nil` if the method fails.

Availability

Mac OS X v10.4 and later.

See Also

`+ shapeWithRect:` (page 80)

Declared In

`CIFilterShape.h`

insetByX:Y:

Modifies a filter shape object so that it is inset by the specified x and y values.

```
- (CIFilterShape *)insetByX:(int)dx Y:(int)dy
```

Parameters

dx

A value that specifies an inset in the x direction.

dy

A value that specifies an inset in the y direction.

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIDemoImageUnit

Declared In

`CIFilterShape.h`

intersectWith:

Creates a filter shape object that represents the intersection of the current filter shape and the specified filter shape object.

```
- (CIFilterShape *)intersectWith:(CIFilterShape *)s2
```

Parameters

s2

A filter shape object.

Return Value
The filter shape object that results from the intersection.

Availability
Mac OS X v10.4 and later.

See Also
`- intersectWithRect:` (page 82)

Declared In
`CIFilterShape.h`

intersectWithRect:

Creates a filter shape that represents the intersection of the current filter shape and a rectangle.

```
- (CIFilterShape *)intersectWithRect:(CGRect)r
```

Parameters

rect

A rectangle. Core Image uses the rectangle specified by integer parts of the width and height.

Return Value
The filter shape that results from the intersection

Availability
Mac OS X v10.4 and later.

See Also
`- intersectWith:` (page 81)

Declared In
`CIFilterShape.h`

transformBy:interior:

Creates a filter shape that results from applying a transform to the current filter shape.

```
- (CIFilterShape *)transformBy:(CGAffineTransform)m interior:(BOOL)flag
```

Parameters

m

A transform.

flag

NO specifies that the new filter shape object can contain all the pixels in the transformed shape (and possibly some that are outside the transformed shape). YES specifies that the new filter shape object can contain a subset of the pixels in the transformed shape (but none of those outside the transformed shape).

Return Value
The transformed filter shape object.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIColorTracking

Declared In
`CIFilterShape.h`

unionWith:

Creates a filter shape that results from the union of the current filter shape and another filter shape object.

```
- (CIFilterShape *)unionWith:(CIFilterShape *)s2
```

Parameters

s2

A filter shape object.

Return Value
The filter shape object that results from the union.

Availability
Mac OS X v10.4 and later.

See Also
`– unionWithRect:` (page 83)

Declared In
`CIFilterShape.h`

unionWithRect:

Creates a filter shape that results from the union of the current filter shape and a rectangle.

```
- (CIFilterShape *)unionWithRect:(CGRect)r
```

Parameters

rect

A rectangle. Core Image uses the rectangle specified by integer parts of the width and height.

Availability
Mac OS X v10.4 and later.

See Also

`- unionWith:` (page 83)

Related Sample Code

CIAnnotation

Declared In

`CIFilterShape.h`

CIImage Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIImage.h QuartzCore/CIImageProvider.h
Availability	Mac OS X v10.4 and later
Companion guide	Core Image Programming Guide
Related sample code	CIAnnotation CIColorTracking CITransitionSelectorSample CITransitionSelectorSample2 FunHouse

Overview

The `CIImage` class represents an image. Core Image images are immutable. 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 (`CVImageBufferRef`), 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" method allows Core Image to operate as efficiently as possible.

Core Image defines methods for creating and initializing images. Additional methods that support drawing and initializing an image with an `NSBitmapImageRep` object are defined in *CIImage Additions Reference*.

Tasks

Creating an Image

+ `emptyImage` (page 88)
  Creates and returns an empty image object.

+ `imageWithColor:` (page 91)
  Creates and returns an image of infinite extent that is initialized the specified color.

+ `imageWithBitmapData:bytesPerRow:size:format:colorSpace:` (page 88)
  Creates and returns an image object from bitmap data.

+ `imageWithCGImage:` (page 89)
  Creates and returns an image object from a Quartz 2D image.

+ `imageWithCGImage:options:` (page 89)
  Creates and returns an image object from a Quartz 2D image using the specified color space.

+ `imageWithCGLayer:` (page 90)
  Creates and returns an image object from the contents supplied by a `CGLayer` object.

+ `imageWithCGLayer:options:` (page 90)
  Creates and returns an image object from the contents supplied by a `CGLayer` object, using the specified options.

+ `imageWithContentsOfURL:` (page 91)
  Creates and returns an image object from the contents of a file.

+ `imageWithContentsOfURL:options:` (page 92)
  Creates and returns an image object from the contents of a file, using the specified options.

+ `imageWithCVImageBuffer:` (page 92)
  Creates and returns an image object from the contents of `CVImageBuffer` object.

+ `imageWithCVImageBuffer:options:` (page 93)
  Creates and returns an image object from the contents of `CVImageBuffer` object, using the specified options.

+ `imageWithData:` (page 93)
  Creates and returns an image object initialized with the supplied image data.

+ `imageWithData:options:` (page 94)
  Creates and returns an image object initialized with the supplied image data, using the specified options.

+ `imageWithImageProvider:size:format:colorSpace:options:` (page 94)
  Creates and returns an image object initialized with data provided by an image provider.

+ `imageWithTexture:size:flipped:colorSpace:` (page 95)
  Creates and returns an image object initialized with data supplied by an OpenGL texture.

Creating an Image by Modifying an Existing Image

– `imageByApplyingTransform:` (page 97)
  Returns a new image that represents the original image after applying an affine transform.

`– imageByCroppingToRect:` (page 97)
Returns a new image that represents the original image after cropping to a rectangle.

Initializing an Image

`– initWithColor:` (page 101)
Initializes an image with the specified color.

`– initWithBitmapData:bytesPerRow:size:format:colorSpace:` (page 98)
Initializes an image object with bitmap data.

`– initWithCGImage:` (page 99)
Initializes an image object with a Quartz 2D image.

`– initWithCGImage:options:` (page 99)
Initializes an image object with a Quartz 2D image, using the specified options.

`– initWithCGLayer:` (page 100)
Initializes an image object from the contents supplied by a CGLayer object.

`– initWithCGLayer:options:` (page 100)
Initializes an image object from the contents supplied by a CGLayer object, using the specified options.

`– initWithContentsOfURL:` (page 101)
Initializes an image object from the contents of a file.

`– initWithContentsOfURL:options:` (page 101)
Initializes an image object from the contents of a file, using the specified options.

`– initWithCVImageBuffer:` (page 102)
Initializes an image object from the contents of CVImageBuffer object.

`– initWithCVImageBuffer:options:` (page 102)
Initializes an image object from the contents of CVImageBuffer object, using the specified options.

`– initWithData:` (page 103)
Initializes an image object with the supplied image data.

`– initWithData:options:` (page 103)
Initializes an image object with the supplied image data, using the specified options.

`– initWithImageProvider:size:format:colorSpace:options:` (page 104)
Initializes an image object with data provided by an image provider, using the specified options.

`– initWithTexture:size:flipped:colorSpace:` (page 105)
Initializes an image object with data supplied by an OpenGL texture.

Getting Image Information

`– definition` (page 96)
Returns a filter shape object that represents the domain of definition of the image.

`– extent` (page 97)
Returns a rectangle that specifies the extent of the image.

Class Methods

emptyImage

Creates and returns an empty image object.

```
+ (CIImage *)emptyImage
```

Return Value
An image object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CIImage.h`

imageWithBitmapData:bytesPerRow:size:format:colorSpace:

Creates and returns an image object from bitmap data.

```
+ (CIImage *)imageWithBitmapData:(NSData *)d bytesPerRow:(size_t)bpr
    size:(CGSize)size format:(CIFormat)f colorSpace:(CGColorSpaceRef)cs
```

Parameters

d

The bitmap data for the image. This data must be premultiplied.

bpr

The number of bytes per row.

size

The dimensions of the image.

f

The format and size of each pixel. You must supply a pixel format constant. See "`Pixel Formats`" (page 106).

cs

The color space that the image is defined in. If this value is `nil`, the image is not color matched. Pass `nil` for images that don't contain color data (such as elevation maps, normal vector maps, and sampled function tables).

Return Value
An image object.

Availability
Mac OS X v10.4 and later.

See Also
`– initWithBitmapData:bytesPerRow:size:format:colorSpace:` (page 98)

Related Sample Code
FunHouse

Declared In

`CIImage.h`

imageWithCGImage:

Creates and returns an image object from a Quartz 2D image.

```
+ (CIImage *)imageWithCGImage:(CGImageRef)image
```

Parameters

image

A Quartz 2D image (`CGImageRef`) object. For more information, see *Quartz 2D Programming Guide* and *CGImage Reference*.

Return Value

An image object initialized with the contents of the Quartz 2D image.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithCGImage:options:` (page 89)

`- initWithCGImage:` (page 99)

Related Sample Code

AnimatedTableView

CIVideoDemoGL

ImageApp

Declared In

`CIImage.h`

imageWithCGImage:options:

Creates and returns an image object from a Quartz 2D image using the specified color space.

```
+ (CIImage *)imageWithCGImage:(CGImageRef)image options:(NSDictionary *)d
```

Parameters

image

A Quartz 2D image (`CGImageRef`) object. For more information, see *Quartz 2D Programming Guide* and *CGImage Reference*.

d

A dictionary that contains a color space key (`kCIImageColorSpace` (page 107)) whose value is a `CGColorSpace`object. (See `CGColorSpaceRef`.)

Return Value

An image object initialized with the contents of the Quartz 2D image and the specified color space.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithCGImage:` (page 89)

`- initWithCGImage:options:` (page 99)

Declared In

`CIImage.h`

imageWithCGLayer:

Creates and returns an image object from the contents supplied by a `CGLayer` object.

```
+ (CIImage *)imageWithCGLayer:(CGLayerRef)layer
```

Parameters

`layer`

A `CGLayer` object. For more information see *Quartz 2D Programming Guide* and *CGLayer Reference*.

Return Value

An image object initialized with the contents of the layer object.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithCGLayer:options:` (page 90)

`- initWithCGLayer:` (page 100)

Declared In

`CIImage.h`

imageWithCGLayer:options:

Creates and returns an image object from the contents supplied by a `CGLayer` object, using the specified options.

```
+ (CIImage *)imageWithCGLayer:(CGLayerRef)layer options:(NSDictionary *)d
```

Parameters

`layer`

A `CGLayer` object. For more information see *Quartz 2D Programming Guide* and *CGLayer Reference*.

`d`

A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "Pixel Formats" (page 106).

Return Value

An image object initialized with the contents of the layer object and set up with the specified options.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithCGLayer:` (page 90)

`- initWithCGLayer:options:` (page 100)

Declared In

`CIImage.h`

imageWithColor:

Creates and returns an image of infinite extent that is initialized the specified color.

```
+ (CIImage *)imageWithColor:(CIColor *)color
```

Parameters

color

A color object.

Return Value

The image object initialized with the color represented by the `CIColor` object.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- initWithColor:` (page 101)

Declared In

`CIImage.h`

imageWithContentsOfURL:

Creates and returns an image object from the contents of a file.

```
+ (CIImage *)imageWithContentsOfURL:(NSURL *)url
```

Parameters

url

The location of the file.

Return Value

An image object initialized with the contents of the file.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithContentsOfURL:options:` (page 92)

`- initWithContentsOfURL:` (page 101)

Related Sample Code

CIAnnotation

CITransitionSelectorSample

CITransitionSelectorSample2

CoreImageGLTextureFBO

FunHouse

Declared In
`CIImage.h`

imageWithContentsOfURL:options:

Creates and returns an image object from the contents of a file, using the specified options.

```
+ (CIImage *)imageWithContentsOfURL:(NSURL *)url options:(NSDictionary *)d
```

Parameters

`url`
The location of the file.

`d`
A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "Pixel Formats" (page 106).

Return Value
An image object initialized with the contents of the file and set up with the specified options.

Availability
Mac OS X v10.4 and later.

See Also
`+ imageWithContentsOfURL:` (page 91)
`- initWithContentsOfURL:options:` (page 101)

Declared In
`CIImage.h`

imageWithCVImageBuffer:

Creates and returns an image object from the contents of `CVImageBuffer` object.

```
+ (CIImage *)imageWithCVImageBuffer:(CVImageBufferRef)imageBuffer
```

Parameters

`imageBuffer`
A `CVImageBuffer` object. For more information, see *Core Video Programming Guide* and *Core Video Reference*.

Return Value
An image object initialized with the contents of the image buffer object.

Availability
Mac OS X v10.4 and later.

See Also
`+ imageWithCVImageBuffer:options:` (page 93)
`- initWithCVImageBuffer:` (page 102)

Related Sample Code
CIColorTracking
CIVideoDemoGL

QTCoreImage101
StillMotion
WhackedTV

Declared In
`CIImage.h`

imageWithCVImageBuffer:options:

Creates and returns an image object from the contents of `CVImageBuffer` object, using the specified options.

```
+ (CIImage *)imageWithCVImageBuffer:(CVImageBufferRef)imageBuffer
    options:(NSDictionary *)dict
```

Parameters

`imageBuffer`

A `CVImageBuffer` object. For more information, see *Core Video Programming Guide* and *Core Video Reference*.

`dict`

A dictionary that contains options for creating an image object. You can supply such options as a color space. (The pixel format is supplied by the `CVImageBuffer` object.)

Return Value
An image object initialized with the contents of the image buffer object and set up with the specified options.

Availability
Mac OS X v10.4 and later.

See Also
`+ imageWithCVImageBuffer:` (page 92)
`- initWithCVImageBuffer:options:` (page 102)

Declared In
`CIImage.h`

imageWithData:

Creates and returns an image object initialized with the supplied image data.

```
+ (CIImage *)imageWithData:(NSData *)data
```

Parameters

`data`

The data object that holds the contents of an image file (such as TIFF, GIF, JPG, or whatever else the system supports). The image data must be premultiplied.

Return Value
An image object initialized with the supplied data, or `nil` if the method cannot create an image representation from the contents of the supplied data object.

Availability
Mac OS X v10.4 and later.

See Also

`+ imageWithData:options:` (page 94)

`- initWithData:` (page 103)

Related Sample Code

CoreImageGLTextureFBO

Denoise

LayerBackedOpenGLView

WebKitCIPlugIn

Declared In

`CIImage.h`

imageWithData:options:

Creates and returns an image object initialized with the supplied image data, using the specified options.

```
+ (CIImage *)imageWithData:(NSData *)data options:(NSDictionary *)d
```

Parameters

data

A pointer to the image data. The data must be premultiplied

d

A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "`Pixel Formats`" (page 106).

Return Value

An image object initialized with the supplied data and set up with the specified options.

Availability

Mac OS X v10.4 and later.

See Also

`+ imageWithData:` (page 93)

`- initWithData:options:` (page 103)

Declared In

`CIImage.h`

imageWithImageProvider:size:format:colorSpace:options:

Creates and returns an image object initialized with data provided by an image provider.

```
+ (CIImage *)imageWithImageProvider:(id)p size:(size_t)width :(size_t)height
    format(CIFormat)f colorSpace:(CGColorSpaceRef)cs options:(NSDictionary *)dict
```

Parameters

p

A data provider that implements the `CIImageProvider` informal protocol. Core Image retains this data until the image is deallocated.

width
The width of the image.

height
The height of the image.

f
A pixel format constant. See "Pixel Formats" (page 106).

cs
The color space that the image is defined in. If the this value is nil, the image is not color matched. Pass nil for images that don't contain color data (such as elevation maps, normal vector maps, and sampled function tables).

dict
A dictionary that specifies image-creation options, which can be kCIImageProviderTileSize or kCIImageProviderUserInfo. See *CIImageProvider Protocol Reference* for more information on these options.

Return Value
An image object initialized with the data from the data provider. Core Image does not populate the image object until the object needs the data.

Availability
Mac OS X v10.4 and later.

Declared In
CIImageProvider.h

See Also
- initWithImageProvider:size::format:colorSpace:options: (page 104)

imageWithTexture:size:flipped:colorSpace:

Creates and returns an image object initialized with data supplied by an OpenGL texture.

```
+ (CIImage *)imageWithTexture:(unsigned int)name size:(CGSize)size flipped:(BOOL)flag
      colorSpace:(CGColorSpaceRef)cs
```

Parameters

name
An OpenGL texture. Because CIImage objects are immutable, the texture must remain unchanged for the life of the image object. See the discussion for more information.

size
The dimensions of the texture.

flag
YES to have Core Image flip the contents of the texture vertically.

cs
The color space that the image is defined in. If the colorSpace value is nil, the image is not color matched. Pass nil for images that don't contain color data (such as elevation maps, normal vector maps, and sampled function tables).

Return Value
An image object initialized with the texture data.

Discussion
When using a texture to create a `CIImage` object, the texture must be valid in the Core Image context (`CIContext`) that you draw the `CIImage` object into. This means that one of the following must be true:

- The texture must be created using the `CGLContext` object that the `CIContext` is based on.
- The context that the texture was created in must be shared with the `CGLContext` that the `CIContext` is based on.

Note that textures do not have a retain and release mechanism. This means that your application must make sure that the texture exists for the life cycle of the image. When you no longer need the image, you can delete the texture.

Core Image ignores the texture filtering and wrap modes (`GL_TEXTURE_FILTER` and `GL_TEXTURE_WRAP`) that you set through OpenGL. The filter and wrap modes are overridden by what the CISampler object specifies when you apply a filter to the `CIImage` object.

Availability
Mac OS X v10.4 and later.

See Also
`– initWithTexture:size:flipped:colorSpace:` (page 105)

Related Sample Code
DispatchFractal

Declared In
`CIImage.h`

Instance Methods

definition

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

```
- (CIFilterShape *)definition
```

Return Value
A filter shape object.

Availability
Mac OS X v10.4 and later.

See Also
`– extent` (page 97)

Declared In
`CIImage.h`

extent

Returns a rectangle that specifies the extent of the image.

```
- (CGRect)extent
```

Return Value
A rectangle that specifies the extent of the image in working space coordinates.

Availability
Mac OS X v10.4 and later.

See Also
`- definition` (page 96)

Related Sample Code
CIAnnotation

CIRAWFilterSample

FunHouse

ImageApp

Reducer

Declared In
`CIImage.h`

imageByApplyingTransform:

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

```
- (CIImage *)imageByApplyingTransform:(CGAffineTransform)matrix
```

Parameters

matrix

An affine transform.

Return Value
The transformed image object.

Availability
Mac OS X v10.4 and later.

See Also
`- imageByCroppingToRect:` (page 97)

Related Sample Code
ImageApp

Declared In
`CIImage.h`

imageByCroppingToRect:

Returns a new image that represents the original image after cropping to a rectangle.

```
- (CIImage *)imageByCroppingToRect:(CGRect)r
```

Return Value
An image object cropped to the specified rectangle.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- imageByApplyingTransform:` (page 97)

Declared In
`CIImage.h`

initWithBitmapData:bytesPerRow:size:format:colorSpace:

Initializes an image object with bitmap data.

```
- (id)initWithBitmapData:(NSData *)d bytesPerRow:(size_t)bpr size:(CGSize)size
    format:(CIFormat)f colorSpace:(CGColorSpaceRef)c
```

Parameters

d

The bitmap data to use for the image. The data you supply must be premultiplied.

bpr

The number of bytes per row.

size

The size of the image data.

f

A pixel format constant. See "`Pixel Formats`" (page 106).

c

The color space that the image is defined in and must be a Quartz 2D color space (`CGColorSpaceRef`). Pass `nil` for images that don't contain color data (such as elevation maps, normal vector maps, and sampled function tables).

Return Value
The initialized image object or `nil` if the object could not be initialized.

Availability
Mac OS X v10.4 and later.

See Also
`+ imageWithBitmapData:bytesPerRow:size:format:colorSpace:` (page 88)

Related Sample Code
SonogramViewDemo

Declared In
`CIImage.h`

initWithCGImage:

Initializes an image object with a Quartz 2D image.

```
- (id)initWithCGImage:(CGImageRef)image
```

Parameters

image

A Quartz 2D image (`CGImageRef`) object. For more information, see *Quartz 2D Programming Guide* and *CGImage Reference*.

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithCGImage:options:` (page 99)

`+ imageWithCGImage:` (page 89)

Declared In

`CIImage.h`

initWithCGImage:options:

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

```
- (id)initWithCGImage:(CGImageRef)image options:(NSDictionary *)d
```

Parameters

image

A Quartz 2D image (`CGImageRef`) object. For more information, see *Quartz 2D Programming Guide* and *CGImage Reference*.

d

A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "`Pixel Formats`" (page 106).

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithCGImage:` (page 99)

`+ imageWithCGImage:options:` (page 89)

Declared In

`CIImage.h`

initWithCGLayer:

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

```
- (id)initWithCGLayer:(CGLayerRef)layer
```

Parameters

layer

A CGLayer object. For more information see *Quartz 2D Programming Guide* and *CGLayer Reference*.

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithCGLayer:options:` (page 100)

`+ imageWithCGLayer:` (page 90)

Related Sample Code

CIAnnotation

CIBevelSample

FunHouse

Declared In

`CIImage.h`

initWithCGLayer:options:

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

```
- (id)initWithCGLayer:(CGLayerRef)layer options:(NSDictionary *)d
```

Parameters

layer

A CGLayer object. For more information see *Quartz 2D Programming Guide* and *CGLayer Reference*.

d

A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "`Pixel Formats`" (page 106).

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithCGLayer:` (page 100)

`+ imageWithCGLayer:options:` (page 90)

Declared In

`CIImage.h`

initWithColor:

Initializes an image with the specified color.

```
- (id)initWithColor:(CIColor *)color
```

Parameters

color

A color object.

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Available in Mac OS X v10.5 and later.

See Also

`+ imageWithColor:` (page 91)

Declared In

`CIImage.h`

initWithContentsOfURL:

Initializes an image object from the contents of a file.

```
- (id)initWithContentsOfURL:(NSURL *)url
```

Parameters

url

The location of the file.

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithContentsOfURL:options:` (page 101)

`+ imageWithContentsOfURL:` (page 91)

Declared In

`CIImage.h`

initWithContentsOfURL:options:

Initializes an image object from the contents of a file, using the specified options.

```
- (id)initWithContentsOfURL:(NSURL *)url options:(NSDictionary *)d
```

Parameters

url

The location of the file.

d

A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "`Pixel Formats`" (page 106).

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithContentsOfURL:` (page 101)

`+ imageWithContentsOfURL:options:` (page 92)

Declared In

`CIImage.h`

initWithCVImageBuffer:

Initializes an image object from the contents of CVImageBuffer object.

```
- (id)initWithCVImageBuffer:(CVImageBufferRef)imageBuffer
```

Parameters

imageBuffer

A CVImageBuffer object. For more information, see *Core Video Programming Guide* and *Core Video Reference*.

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithCVImageBuffer:options:` (page 102)

`+ imageWithCVImageBuffer:` (page 92)

Related Sample Code

VideoViewer

Declared In

`CIImage.h`

initWithCVImageBuffer:options:

Initializes an image object from the contents of CVImageBuffer object, using the specified options.

```
- (id)initWithCVImageBuffer:(CVImageBufferRef)imageBuffer options:(NSDictionary
    *)dict
```

Parameters

imageBuffer

A CVImageBuffer object. For more information, see *Core Video Programming Guide* and *Core Video Reference*.

dict

A dictionary that contains options for creating an image object. You can supply such options as a color space. (The pixel format is supplied by the `CVImageBuffer` object.)

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithCVImageBuffer:` (page 102)

`+ imageWithCVImageBuffer:options:` (page 93)

Declared In

`CIImage.h`

initWithData:

Initializes an image object with the supplied image data.

```
- (id)initWithData:(NSData *)data
```

Parameters

data

The image data. The data you supply must be premultiplied.

Return Value

The initialized image object or `nil` if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithData:options:` (page 103)

`+ imageWithData:` (page 93)

Declared In

`CIImage.h`

initWithData:options:

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

```
- (id)initWithData:(NSData *)data options:(NSDictionary *)d
```

Parameters

data

The image data. The data you supply must be premultiplied.

d

A dictionary that contains options for creating an image object. You can supply such options as a pixel format and a color space. See "Pixel Formats" (page 106).

Return Value

The initialized image object or nil if the object could not be initialized.

Availability

Mac OS X v10.4 and later.

See Also

- initWithData: (page 103)

+ imageWithData:options: (page 94)

Declared In

CIImage.h

initWithImageProvider:size:format:colorSpace:options:

Initializes an image object with data provided by an image provider, using the specified options.

```
- (id)initWithImageProvider:(id)p size:(size_t)width:(size_t)height
    format:(CIFormat)f colorSpace:(CGColorSpaceRef)cs options:(NSDictionary *)dict
```

Parameters

p

A data provider that implements the CIImageProvider informal protocol. Core Image retains this data until the image is deallocated.

width

The width of the image data.

height

The height of the image data.

f

A pixel format constant. See "Pixel Formats" (page 106).

cs

The color space of the image. If this value is nil, the image is not color matched. Pass nil for images that don't contain color data (such as elevation maps, normal vector maps, and sampled function tables).

dict

A dictionary that specifies image-creation options, which can be kCIImageProviderTileSize or kCIImageProviderUserInfo. See *CIImageProvider Protocol Reference* for more information on these options.

Return Value

The initialized image object or nil if the object could not be initialized.

Discussion

Core Image does not populate the image until it actually needs the data.

Availability
Mac OS X v10.4 and later.

Declared In
`CIImageProvider.h`

See Also
`+ imageWithImageProvider:size::format:colorSpace:options:` (page 94)

initWithTexture:size:flipped:colorSpace:

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

```
- (id)initWithTexture:(unsigned int)name size:(CGSize)size flipped:(BOOL)flag
    colorSpace:(CGColorSpaceRef)cs
```

Parameters

name

An OpenGL texture. Because `CIImage` objects are immutable, the texture must remain unchanged for the life of the image object. See the discussion for more information.

size

The dimensions of the texture.

flag

`YES` to have Core Image flip the contents of the texture vertically.

cs

The color space that the image is defined in. This must be a Quartz color space (`CGColorSpaceRef`). If the `colorSpace` value is `nil`, the image is not color matched. Pass `nil` for images that don't contain color data (such as elevation maps, normal vector maps, and sampled function tables).

Return Value
The initialized image object or `nil` if the object could not be initialized.

Discussion
When using a texture to create a `CIImage` object, the texture must be valid in the Core Image context (`CIContext`) that you draw the `CIImage` object into. This means that one of the following must be true:

- The texture must be created using the `CGLContext` object that the `CIContext` is based on.
- The context that the texture was created in must be shared with the `CGLContext` that the `CIContext`is based on.

Note that textures do not have a retain and release mechanism. This means that your application must make sure that the texture exists for the life cycle of the image. When you no longer need the image, you can delete the texture.

Core Image ignores the texture filtering and wrap modes (`GL_TEXTURE_FILTER` and `GL_TEXTURE_WRAP`) that you set through OpenGL. The filter and wrap modes are overridden by what the CISampler object specifies when you apply a filter to the `CIImage` object.

Availability
Mac OS X v10.4 and later.

See Also
`+ imageWithTexture:size:flipped:colorSpace:` (page 95)

Declared In
`CIImage.h`

Constants

Pixel Formats

Image data pixel formats.

```
extern CIFormat kCIFormatARGB8;
extern CIFormat kCIFormatRGBA16;
extern CIFormat kCIFormatRGBAf;
```

Constants

`CIFormat`
The data type for a pixel format.

`kCIFormatARGB8`
A 32 bit-per-pixel, fixed-point pixel format.

Available in Mac OS X v10.6 and later.

Declared in `CIImage.h`.

`kCIFormatRGBA16`
A 64 bit-per-pixel, fixed-point pixel format.

Available in Mac OS X v10.6 and later.

Declared in `CIImage.h`.

`kCIFormatRGBAf`
A 128 bit-per-pixel, floating-point pixel format.

Available in Mac OS X v10.6 and later.

Declared in `CIImage.h`.

Declared In
`CIImage.h`

Color Space Key

A key for the color space of an image.

```
extern NSString *kCIImageColorSpace;
```

Constants

`kCIImageColorSpace`

The key for a color space. The value you supply for this dictionary key must be a `CGColorSpaceRef` data type. For more information on this data type see *CGColorSpace Reference*. Typically you use this option when you need to load an elevation, mask, normal vector, or RAW sensor data directly from a file without color correcting it. This constant specifies to override Core Image, which, by default, assumes that data is in GenericRGB.

Available in Mac OS X v10.6 and later.

Declared in `CIImage.h`.

Declared In

`CIImage.h`

CIImageAccumulator Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIImageAccumulator.h
Availability	Mac OS X v10.4 and later
Companion guide	Core Image Programming Guide
Related sample code	CIAnnotation CIMicroPaint

Overview

The `CIImageAccumulator` class enables feedback-based image processing for such things as iterative painting operations or fluid dynamics simulations. You use `CIImageAccumulator` objects in conjunction with other Core Image classes, such as `CIFilter`, `CIImage`, `CIVector`, and `CIContext`, to take advantage of the built-in Core Image filters when processing images.

Tasks

Creating an Image Accumulator

`+ imageAccumulatorWithExtent:format:` (page 110)
Creates an image accumulator with the specified extent and pixel format.

Initializing an Image Accumulator

`- initWithExtent:format:` (page 112)
Initializes an image accumulator with the specified extent and pixel format.

Setting an Image

- setImage: (page 112)

  Sets the contents of the image accumulator to the contents of the specified image object.

- setImage:dirtyRect: (page 113)

  Updates an image accumulator with a subregion of an image object.

Obtaining Data From an Image Accumulator

- extent (page 111)

  Returns the extent of the image associated with the image accumulator.

- format (page 111)

  Returns the pixel format of the image accumulator.

- image (page 112)

  Returns the current contents of the image accumulator.

Resetting an Accumulator

- clear (page 111)

  Resets the accumulator, discarding any pending updates and the current content.

Class Methods

imageAccumulatorWithExtent:format:

Creates an image accumulator with the specified extent and pixel format.

```
+ (CIImageAccumulator *)imageAccumulatorWithExtent:(CGRect)r format:(CIFormat)f
```

Parameters

r

A rectangle that specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width and height.

f

The format and size of each pixel. You must supply a pixel format constant, such as `kCIFormatARGB8` (32 bit-per-pixel, fixed-point pixel format) or `kCIFormatRGBAf` (128 bit-per-pixel, floating-point pixel format). See *CIImage Class Reference* for more information about pixel format constants.

Return Value

The image accumulator object.

Availability

Mac OS X v10.4 and later.

See Also

- initWithExtent:format: (page 112)

Declared In
`CIImageAccumulator.h`

Instance Methods

clear

Resets the accumulator, discarding any pending updates and the current content.

```
- (void)clear
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CIImageAccumulator.h`

extent

Returns the extent of the image associated with the image accumulator.

```
- (CGRect)extent
```

Return Value
The rectangle that specifies the size of the image associated with the image accumulator. This rectangle is the size of the complete region of the working coordinate space, and is a fixed area. It specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width and height.

Availability
Mac OS X v10.4 and later.

Declared In
`CIImageAccumulator.h`

format

Returns the pixel format of the image accumulator.

```
- (CIFormat)format
```

Return Value
The pixel format of the image accumulator.

Availability
Mac OS X v10.4 and later.

Declared In
`CIImageAccumulator.h`

image

Returns the current contents of the image accumulator.

```
- (CIImage *)image
```

Return Value
The image object that represents the current contents of the image accumulator.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIMicroPaint

Declared In
`CIImageAccumulator.h`

initWithExtent:format:

Initializes an image accumulator with the specified extent and pixel format.

```
- (id)initWithExtent:(CGRect)r format:(CIFormat)f
```

Parameters

r

A rectangle that specifies the x-value of the rectangle origin, the y-value of the rectangle origin, and the width and height.

f

The format and size of each pixel. You must supply a pixel format constant, such as`kCIFormatARGB8` (32 bit-per-pixel, fixed-point pixel format) or `kCIFormatRGBAf` (128 bit-per-pixel, floating-point pixel format). See *CIImage Class Reference* for more information about pixel format constants.

Return Value
The initialized image accumulator object.

Availability
Mac OS X v10.4 and later.

See Also
`+ imageAccumulatorWithExtent:format:` (page 110)

Related Sample Code
CIAnnotation

CIMicroPaint

Declared In
`CIImageAccumulator.h`

setImage:

Sets the contents of the image accumulator to the contents of the specified image object.

```
- (void)setImage:(CIImage *)im
```

Parameters

im

The image object whose contents you want to assign to the image accumulator.

Availability

Mac OS X v10.4 and later.

See Also

`- setImage:dirtyRect:` (page 113)

Related Sample Code

CIAnnotation

CIMicroPaint

Declared In

`CIImageAccumulator.h`

setImage:dirtyRect:

Updates an image accumulator with a subregion of an image object.

```
- (void)setImage:(CIImage *)im dirtyRect:(CGRect)r
```

Parameters

im

The image object whose contents you want to assign to the image accumulator.

r

A rectangle that defines the subregion of the image object that's changed since the last time you updated the image accumulator. You must guarantee that the new contents differ from the old only within the region specified by the this argument.

Discussion

For additional details on using this method, see "Imaging Dynamical Systems" in *Core Image Programming Guide*.

Availability

Mac OS X v10.4 and later.

See Also

`- setImage:` (page 112)

Declared In

`CIImageAccumulator.h`

CIKernel Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIKernel.h
Availability	Mac OS X v10.4 and later
Companion guides	Core Image Programming Guide Core Image Kernel Language Reference
Related sample code	CIAnnotation CIColorTracking CIDemoImageUnit CIHazeFilterSample

Overview

The `CIKernel` class maintains kernel routines that process individual pixels. The kernel routines in a `CIKernel` object use a subset of the OpenGL Shading Language and Core Image extensions to this language. You use a `CIKernel` object in conjunction with other Core Image classes, such as `CIFilter`, `CIFilterShape`, and `CISampler`, to create custom filters.

Tasks

Creating a Kernel

`+ kernelsWithString:` (page 116)
Creates and returns and array of `CIKernel` objects.

Getting a Kernel Name

`– name` (page 116)
Returns the name of a kernel routine.

Setting a Selector

`- setROISelector:` (page 117)
Sets the selector used to query the region of interest of the kernel.

Class Methods

kernelsWithString:

Creates and returns and array of `CIKernel` objects.

```
+ (NSArray *)kernelsWithString:(NSString *)s
```

Parameters

s

A program in the Core Image dialect of the OpenGL Shading Language that contains one or more routines, each of which is marked using the `kernel` keyword.

Return Value
An array of `CIKernel` objects. The array contains one `CIKernel` objects for each kernel routine in the supplied string.

Discussion
See *Core Image Kernel Language Reference* for more details.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation
CIColorTracking
CIDemoImageUnit
CIHazeFilterSample

Declared In
`CIKernel.h`

Instance Methods

name

Returns the name of a kernel routine.

```
- (NSString *)name
```

Return Value
The name of the kernel routine.

Availability
Mac OS X v10.4 and later.

Declared In
`CIKernel.h`

setROISelector:

Sets the selector used to query the region of interest of the kernel.

```
- (void)setROISelector:(SEL)aMethod
```

Parameters

aMethod

A selector name.

Discussion
The `aMethod` argument must use the signature that is defined for the `regionOf:destRect:userInfo:` method, which is as follows:

```
- (CGRect) regionOf:(int)samplerIndex destRect:(CGRect)r userInfo:obj;
```

where:

- `samplerIndex` defines the sampler to query
- `destRect` is the extent of the region, in working space coordinates, to render.
- `userInfo` is the object associated with the `kCIApplyOptionUserInfo` option when the kernel is applied to its arguments. The `userInfo` is important because instance variables can't be used by the defining class. Instance variables must be passed through the `userInfo` argument.

The `regionOf:destRect:userInfo:` method of the CIFilter object is called by the framework. This method returns the rectangle that contains the region of the sampler that the kernel needs to render the specified destination rectangle.

A sample `regionOf:destRect:userInfo:` method might look as follows:

```
- (CGRect)regionOf:(int)sampler destRect:(CGRect)r userInfo:params
{
  float scale = fabs ([params X]);
  return CGRectInset (r, scale * -1.3333, scale * -1.3333);
}
```

In the filter code, you set the selector using the following:

```
kernel setROISelector:@selector(regionOf:destRect:userInfo:)]
```

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation

Declared In
`CIKernel.h`

CIPlugIn Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIPlugIn.h
Availability	Mac OS X v10.4 and later
Companion guides	Image Unit Tutorial Core Image Programming Guide
Related sample code	CIAnnotation CIColorTracking CIFilterGeneratorTest CIVideoDemoGL FunHouse

Overview

The `CIPlugIn` class loads image units. An image unit is an image processing bundle that contains one or more Core Image filters. The `.plugin` extension indicates one or more filters that are packaged as an image unit.

Tasks

Loading Plug-ins

`+ loadAllPlugIns` (page 120)
Scans directories for files that have the `.plugin` extension and then loads the image units.

`+ loadNonExecutablePlugIns` (page 120)
Scans directories for files that have the `.plugin` extension and then loads only those filters that are marked by the image unit as non-executable filters.

`+ loadPlugIn:allowNonExecutable:` (page 121)
Loads filters from an image unit that have the appropriate executable status.

Class Methods

loadAllPlugIns

Scans directories for files that have the `.plugin` extension and then loads the image units.

```
+ (void)loadAllPlugIns
```

Discussion

This method scans the following directories:

- `/Library/Graphics/Image Units`
- `~/Library/Graphics/Image Units`

Call this method once. If you call this method more than once, Core Image loads newly added image units, but image units (and the filters they contain) that are already loaded are not removed.

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIAnnotation

CIColorTracking

CIFilterGeneratorTest

CIVideoDemoGL

FunHouse

Declared In

`CIPlugIn.h`

loadNonExecutablePlugIns

Scans directories for files that have the `.plugin` extension and then loads only those filters that are marked by the image unit as non-executable filters.

```
+ (void)loadNonExecutablePlugIns
```

Discussion

This call does not execute any of the code in the image unit, it simply loads the code. You need to call this method only once to load a specific image unit. The behavior of this method is not defined for multiple calls for the same image unit.

Availability

Mac OS X v10.4 and later.

Declared In

`CIPlugIn.h`

loadPlugIn:allowNonExecutable:

Loads filters from an image unit that have the appropriate executable status.

```
+ (void)loadPlugIn:(NSURL *)url allowNonExecutable:(BOOL)allowNonExecutable
```

Parameters

url

The location of the image unit to load.

allowNonExecutable

`TRUE` to load only those filters that are marked by the image unit as non-executable filters.

Discussion

You need to call this method only once to load a specific image unit. The behavior of this method is not defined for multiple calls for the same image unit.

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIAnnotation

CIColorTracking

Declared In

`CIPlugIn.h`

CISampler Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CISampler.h
Availability	Mac OS X v10.4 and later
Companion guide	Core Image Programming Guide
Related sample code	CIAnnotation CIColorTracking CIDemoImageUnit CIHazeFilterSample

Overview

The `CISampler` class retrieves samples of images for processing by a `CIKernel` object. A `CISampler` object defines a coordinate transform, and modes for interpolation and wrapping. You use `CISampler` objects in conjunction with other Core Image classes, such as `CIFilter`, `CIKernel`, and `CIFilterShape`, to create custom filters.

Tasks

Creating a Sampler

`+ samplerWithImage:` (page 124)
Creates and returns a sampler that references an image.

`+ samplerWithImage:keysAndValues:` (page 125)
Creates and returns a sampler that references an image using options specified as key-value pairs.

`+ samplerWithImage:options:` (page 125)
Creates and returns a sampler that references an image using options specified in a dictionary.

Initializing a Sampler

`- initWithImage:` (page 127)
    Initializes a sampler with an image object.

`- initWithImage:keysAndValues:` (page 127)
    Initializes the sampler with an image object using options specified as key-value pairs.

`- initWithImage:options:` (page 127)
    Initializes the sampler with an image object using options specified in a dictionary.

Getting Information About the Sampler Object

`- definition` (page 126)
    Gets the domain of definition (DOD) of the sampler.

`- extent` (page 126)
    Gets the rectangle that specifies the extent of the sampler.

Class Methods

samplerWithImage:

Creates and returns a sampler that references an image.

```
+ (CISampler *)samplerWithImage:(CIImage *)im
```

Parameters

im

The image that you want the sampler to reference.

Return Value

A sampler object that references the image specified by the `im` argument.

Availability

Mac OS X v10.4 and later.

See Also

`+ samplerWithImage:keysAndValues:` (page 125)

`+ samplerWithImage:options:` (page 125)

Related Sample Code

CIAnnotation

CIColorTracking

CIDemoImageUnit

CIHazeFilterSample

Declared In

`CISampler.h`

samplerWithImage:keysAndValues:

Creates and returns a sampler that references an image using options specified as key-value pairs.

```
+ (CISampler *)samplerWithImage:(CIImage *)im keysAndValues:key0, ...
```

Parameters

im

The image that you want the sampler to reference.

key0

A list of key-value pairs that represent options. Each key needs to be followed by that appropriate value. You can supply one or more key-value pairs. Use `nil` to specify the end of the key-value options. See “`Sampler Option Keys`” (page 128).

Return Value

A sampler that references the image specified by the `im` argument and uses the specified options.

Availability

Mac OS X v10.4 and later.

See Also

`+ samplerWithImage:` (page 124)
`+ samplerWithImage:options:` (page 125)

Related Sample Code

CIColorTracking

Declared In

`CISampler.h`

samplerWithImage:options:

Creates and returns a sampler that references an image using options specified in a dictionary.

```
+ (CISampler *)samplerWithImage:(CIImage *)im options:(NSDictionary *)dict
```

Parameters

im

The image that you want the sampler to reference.

dict

A dictionary that contains options specified as key-value pairs. See “`Sampler Option Keys`” (page 128).

Return Value

A sampler that references the image specified by the `im` argument and uses the options specified in the dictionary.

Availability

Mac OS X v10.4 and later.

See Also

`+ samplerWithImage:` (page 124)
`+ samplerWithImage:keysAndValues:` (page 125)

Declared In

`CISampler.h`

Instance Methods

definition

Gets the domain of definition (DOD) of the sampler.

```
- (CIFilterShape *)definition
```

Return Value

The filter shape object that contains the DOD.

Discussion

The DOD contains all nontransparent pixels produced by referencing the sampler.

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIAnnotation

CIColorTracking

CIDemoImageUnit

CIHazeFilterSample

Declared In

`CISampler.h`

extent

Gets the rectangle that specifies the extent of the sampler.

```
- (CGRect)extent
```

Return Value

The rectangle that specifies the area outside which the wrap mode set for the sampler is invoked.

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIAnnotation

Declared In

`CISampler.h`

initWithImage:

Initializes a sampler with an image object.

```
- (id)initWithImage:(CIImage *)im
```

Parameters

im

The image object to initialize the sampler with.

Availability

Mac OS X v10.4 and later.

See Also

`- initWithImage:keysAndValues:` (page 127)

`- initWithImage:options:` (page 127)

Declared In

`CISampler.h`

initWithImage:keysAndValues:

Initializes the sampler with an image object using options specified as key-value pairs.

```
- (id)initWithImage:(CIImage *)im keysAndValues:key0, ...
```

Parameters

im

The image object to initialize the sampler with.

key0

A list of key-value pairs that represent options. Each key needs to be followed by that appropriate value. You can supply one or more key-value pairs. Use `nil` to specify the end of the key-value options. See "`Sampler Option Keys`" (page 128).

Availability

Mac OS X v10.4 and later.

See Also

`- initWithImage:` (page 127)

`- initWithImage:options:` (page 127)

Declared In

`CISampler.h`

initWithImage:options:

Initializes the sampler with an image object using options specified in a dictionary.

```
- (id)initWithImage:(CIImage *)im options:(NSDictionary *)dict
```

Parameters

im

The image to initialize the sampler with.

`dict`

A dictionary that contains options specified as key-value pairs. See "`Sampler Option Keys`" (page 128).

Availability

Mac OS X v10.4 and later.

See Also

`- initWithImage:` (page 127)

`- initWithImage:keysAndValues:` (page 127)

Declared In

`CISampler.h`

Constants

Sampler Option Keys

Keys for creating a sampler.

```
extern NSString *kCISamplerAffineMatrix;
extern NSString *kCISamplerWrapMode;
extern NSString *kCISamplerFilterMode
```

Constants

`kCISamplerAffineMatrix`

The key for an affine matrix. The associated value is an `NSArray` object ([*a b c d tx ty*]) that defines the transformation to apply to the sampler.

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

`kCISamplerWrapMode`

The key for the sampler wrap mode. The wrap mode specifies how Core Image produces pixels that are outside the extent of the sample. Possible values are `kCISamplerWrapBlack` (page 129) and `kCISamplerWrapClamp` (page 129).

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

`kCISamplerFilterMode`

The key for the filtering to use when sampling the image. Possible values are `kCISamplerFilterNearest` (page 129) and `kCISamplerFilterLinear` (page 129).

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

Declared In

`CISampler.h`

Sampler Option Values

Values for sampler option keys.

```
extern NSString *kCISamplerWrapBlack;
extern NSString *kCISamplerWrapClamp;
extern NSString *kCISamplerFilterNearest;
extern NSString *kCISamplerFilterLinear;
```

Constants

`kCISamplerWrapBlack`

Pixels are transparent black.

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

`kCISamplerWrapClamp`

Coordinates are clamped to the extent.

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

`kCISamplerFilterNearest`

Nearest neighbor sampling.

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

`kCISamplerFilterLinear`

Bilinear interpolation.

Available in Mac OS X v10.4 and later.

Declared in `CISampler.h`.

Declared In

`CISampler.h`

CIVector Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIVector.h
Availability	Mac OS X v10.4 and later
Companion guide	Core Image Programming Guide
Related sample code	CIAnnotation CIColorTracking CITransitionSelectorSample CocoaSlides FunHouse

Overview

The `CIVector` class is used for coordinate values and direction vectors. You typically use a `CIVector` object to pass parameter values to Core Image filters. `CIVector` objects work in conjunction with other Core Image classes, such as `CIFilter`, `CIContext`, `CIImage`, and `CIColor`, to process images using the Core Image framework.

Tasks

Creating a Vector

`+ vectorWithValues:count:` (page 133)
Creates and returns a vector that is initialized with the specified values.

`+ vectorWithX:Y:` (page 134)
Creates and returns a vector that is initialized with two values.

`+ vectorWithX:Y:Z:` (page 135)
Creates and returns a vector that is initialized with three values.

`+ vectorWithX:Y:Z:W:` (page 135)
    Creates and returns a vector that is initialized with four values.

`+ vectorWithString:` (page 133)
    Creates and returns a vector that is initialized with values provided in a string representation.

`+ vectorWithX:` (page 134) Deprecated in Mac OS X v10.6
    Creates and returns a vector that is initialized with one value.

Initializing a Vector

`– initWithValues:count:` (page 136)
    Initializes a vector with the provided values.

`– initWithX:` (page 137)
    Initializes the first position of a vector with the provided values.

`– initWithX:Y:` (page 137)
    Initializes the first two positions of a vector with the provided values.

`– initWithX:Y:Z:` (page 137)
    Initializes the first three positions of a vector with the provided values.

`– initWithX:Y:Z:W:` (page 138)
    Initializes four positions of a vector with the provided values.

`– initWithString:` (page 136)
    Initializes a vector with values provided in a string representation.

Getting Values From a Vector

`– valueAtIndex:` (page 139)
    Returns a value from a specific position in a vector.

`– count` (page 136)
    Returns the number of items in a vector.

`– X` (page 140)
    Returns the value located in the first position in a vector.

`– Y` (page 140)
    Returns the value located in the second position in a vector.

`– Z` (page 140)
    Returns the value located in the third position in a vector.

`– W` (page 139)
    Returns the value located in the fourth position in a vector.

`– stringRepresentation` (page 138)
    Returns a string representation for a vector.

Class Methods

vectorWithString:

Creates and returns a vector that is initialized with values provided in a string representation.

```
+ (CIVector *)vectorWithString:(NSString *)representation
```

Parameters

representation

A string that is in one of the formats returned by the `stringRepresentation` method.

Discussion

Some typical string representations for vectors are:

```
@"[1.0 0.5 0.3]"
```

which specifies a `vec3` vector whose components are `X = 1.0`, `Y = 0.5`, and `Z = 0.3`

```
@"[10.0 23.0]
```

which specifies a `vec2` vector show components are `X = 10.0` and `Y = 23.0`

Availability

Mac OS X v10.4 and later.

See Also

`- stringRepresentation` (page 138)

Related Sample Code

FunHouse

Declared In

`CIVector.h`

vectorWithValues:count:

Creates and returns a vector that is initialized with the specified values.

```
+ (CIVector *)vectorWithValues:(const CGFloat *)values count:(size_t)count
```

Parameters

values

The values to initialize the vector with.

count

The number of values in the vector.

Return Value

A vector initialized with the provided values.

Availability

Mac OS X v10.4 and later.

Related Sample Code
FunHouse

Declared In
`CIVector.h`

vectorWithX:

Creates and returns a vector that is initialized with one value.

```
+ (CIVector *)vectorWithX:(CGFloat)x
```

Parameters

x

The value to initialize the vector with.

Return Value
A vector initialized with the specified value.

Availability
Mac OS X v10.4 and later.

Declared In
`CIVector.h`

vectorWithX:Y:

Creates and returns a vector that is initialized with two values.

```
+ (CIVector *)vectorWithX:(CGFloat)x Y:(CGFloat)y
```

Parameters

x

The value for the first position in the vector.

y

The value for the second position in the vector.

Return Value
A vector initialized with the specified values.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation
CIColorTracking
CITransitionSelectorSample
CocoaSlides
Reducer

Declared In
`CIVector.h`

vectorWithX:Y:Z:

Creates and returns a vector that is initialized with three values.

```
+ (CIVector *)vectorWithX:(CGFloat)x Y:(CGFloat)y Z:(CGFloat)z
```

Parameters

x

The value for the first position in the vector.

y

The value for the second position in the vector.

z

The value for the third position in the vector.

Return Value

A vector initialized with the specified values.

Availability

Mac OS X v10.4 and later.

Related Sample Code

FunHouse

Declared In

`CIVector.h`

vectorWithX:Y:Z:W:

Creates and returns a vector that is initialized with four values.

```
+ (CIVector *)vectorWithX:(CGFloat)x Y:(CGFloat)y Z:(CGFloat)z W:(CGFloat)w
```

Parameters

x

The value for the first position in the vector.

y

The value for the second position in the vector.

z

The value for the third position in the vector.

w

The value for the fourth position in the vector.

Return Value

A vector initialized with the specified values.

Availability

Mac OS X v10.4 and later.

Related Sample Code

CIColorTracking

CITransitionSelectorSample

CocoaSlides

FunHouse
Reducer

Declared In
`CIVector.h`

Instance Methods

count

Returns the number of items in a vector.

```
- (size_t)count
```

Return Value
The number of items in the vector.

Availability
Mac OS X v10.4 and later.

Declared In
`CIVector.h`

initWithString:

Initializes a vector with values provided in a string representation.

```
- (id)initWithString:(NSString *)representation;
```

Parameters

`representation`
A string that is in one of the formats returned by the `stringRepresentation` method.

Availability
Mac OS X v10.4 and later.

See Also
`- stringRepresentation` (page 138)

Declared In
`CIVector.h`

initWithValues:count:

Initializes a vector with the provided values.

```
- (id)initWithValues:(const CGFloat *)values count:(size_t)count
```

Parameters

values

The values to initialize the vector with.

count

The number of values specified by the `values` argument.

Availability
Mac OS X v10.4 and later.

Declared In
`CIVector.h`

initWithX:

Initializes the first position of a vector with the provided values.

```
- (id)initWithX:(CGFloat)x
```

Parameters

x

The initialization value.

Availability
Mac OS X v10.4 and later.

Declared In
`CIVector.h`

initWithX:Y:

Initializes the first two positions of a vector with the provided values.

```
- (id)initWithX:(CGFloat)x Y:(CGFloat)y
```

Parameters

x

The initialization value for the first position.

y

The initialization value for the second position.

Availability
Mac OS X v10.4 and later.

Declared In
`CIVector.h`

initWithX:Y:Z:

Initializes the first three positions of a vector with the provided values.

```
- (id)initWithX:(CGFloat)x Y:(CGFloat)y Z:(CGFloat)z
```

Parameters

x

The initialization value for the first position.

y

The initialization value for the second position.

z

The initialization value for the third position.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`CIVector.h`

initWithX:Y:Z:W:

Initializes four positions of a vector with the provided values.

```
- (id)initWithX:(CGFloat)x Y:(CGFloat)y Z:(CGFloat)z W:(CGFloat)w
```

Parameters

x

The initialization value for the first position.

y

The initialization value for the second position.

z

The initialization value for the third position.

w

The initialization value for the fourth position.

Availability

Mac OS X v10.4 and later.

Declared In

`CIVector.h`

stringRepresentation

Returns a string representation for a vector.

```
- (NSString *)stringRepresentation
```

Return Value

A string object.

Discussion

You convert the string representation returned by this method to a vector by supplying it as a parameter to the `vectorWithString:` method.

Some typical string representations for vectors are:

```
@"[1.0 0.5 0.3]"
```

which specifies a `vec3` vector whose components are `X = 1.0`, `Y = 0.5`, and `Z = 0.3`

```
@"[10.0 23.0]
```

which specifies a `vec2` vector show components are `X = 10.0` and `Y = 23.0`

Availability
Mac OS X v10.4 and later.

See Also
`+ vectorWithString:` (page 133)

Declared In
`CIVector.h`

valueAtIndex:

Returns a value from a specific position in a vector.

```
- (CGFloat)valueAtIndex:(size_t)index
```

Parameters

index
The position in the vector of the value that you want to retrieve.

Return Value
The value retrieved from the vector or `0` if the position is undefined.

Discussion
The numbering of elements in a vector begins with zero.

Availability
Mac OS X v10.4 and later.

Related Sample Code
FunHouse

Declared In
`CIVector.h`

W

Returns the value located in the fourth position in a vector.

```
- (CGFloat)W
```

Return Value
The value retrieved from the vector.

Availability
Mac OS X v10.4 and later.

Related Sample Code
FunHouse

Declared In
`CIVector.h`

X

Returns the value located in the first position in a vector.

```
- (CGFloat)X
```

Return Value
The value retrieved from the vector.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation

CIColorTracking

FunHouse

Declared In
`CIVector.h`

Y

Returns the value located in the second position in a vector.

```
- (CGFloat)Y
```

Return Value
The value retrieved from the vector.

Availability
Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation

CIColorTracking

FunHouse

Declared In
`CIVector.h`

Z

Returns the value located in the third position in a vector.

```
- (CGFloat)Z
```

Return Value

The value retrieved from the vector.

Availability

Mac OS X v10.4 and later.

Related Sample Code

FunHouse

Declared In

`CIVector.h`

PART II

Protocols

CIImageProvider Protocol Reference

(informal protocol)

Adopted by	NSObject
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIImageProvider.h
Availability	Available in Mac OS X v10.4 and later
Companion guide	Core Image Programming Guide

Overview

The `CIImageProvider` informal protocol defines methods for supplying bitmap data to create or initialize a `CIImage` object.

Tasks

Providing Image Data

`– provideImageData:bytesPerRow:origin:size:userInfo:` (page 145)
Supplies data to a `CIImage` object.

Instance Methods

provideImageData:bytesPerRow:origin:size:userInfo:

Supplies data to a `CIImage` object.

```
- (void)provideImageData:(void *)data bytesPerRow:(size_t)rowbytes  origin:(size_t)
    x:(size_t)y size:(size_t)width:(size_t)height userInfo:(id)info
```

Parameters

data

A pointer to image data. Note that `data[0]` refers to the first byte of the requested subimage, not the larger image buffer.

rowbytes
The number of bytes per row.

x
The x origin of the image data.

y
The y origin of the image data.

width
The width of the image data.

height
The height of the image data.

info
User supplied data, which is optional.

Discussion
You can supply the image provider to these methods of the CIImage class:

- imageWithImageProvider:size::format:colorSpace:options: to create a CIImage object from image data
- initWithImageProvider:size::format:colorSpace:options: to initialize an existing CIImage with data

You initialize the given bitmap with the subregion specified by the arguments x, y, width, and height. The subregion uses the local coordinate space of the image, with the origin at the upper-left corner of the image. If you change the virtual memory mapping of the buffer specified by the data argument (such as by using vm_copy to modify it), the behavior is undefined.

That this callback always requests the full image data regardless of what is actually visible. All of the image is loaded or none of it is. The exception is when you create a tiled image by specifying the kCIImageProviderTileSize option. In this case, only the needed tiles are requested.

Constants

Image Provider Options

Keys for the options dictionary of an image provider.

```
NSString *kCIImageProviderTileSize;
NSString *kCIImageProviderUserInfo;
```

Constants
kCIImageProviderTileSize
A key for the image tiles size. The associated value is an NSArray that containsNSNumber objects for the dimensions of the image tiles requested from the image provider.

Available in Mac OS X v10.4 and later.

Declared in CIImageProvider.h.

`kCIImageProviderUserInfo`

A key for data needed by the image provider. The associated value is an object that contains the needed data.

Available in Mac OS X v10.4 and later.

Declared in `CIImageProvider.h`.

Discussion

You can use these options when you create or initialize an image provider with such methods as `imageWithImageProvider:size::format:colorSpace:options:` or `initWithImageProvider:size::format:colorSpace:options:`.

CIPlugInRegistration Protocol Reference

Adopted by	CIPlugIn
Framework	Library/Frameworks/QuartzCore.framework
Declared in	QuartzCore/CIPlugInInterface.h
Availability	Mac OS X v10.4 and later
Companion guides	Image Unit Tutorial Core Image Programming Guide
Related sample code	CIAnnotation CIColorTracking CIDemoImageUnit

Overview

The `CIPlugInRegistration` protocol defines a method for loading Core Image image units. The principal class of an image unit bundle must support this protocol.

Tasks

Initializing Plug-ins

`– load:` (page 149) *required method*
Loads and initializes an image unit, performing custom tasks as needed. (required)

Instance Methods

load:

Loads and initializes an image unit, performing custom tasks as needed. (required)

```
- (BOOL)load:(void *)host
```

Parameters

`host`

Reserved for future use.

Return Value

Returns `true` if the image unit is successfully initialized

Discussion

The `load` method is called once by the host to initialize the image unit when the first filter in the image unit is instantiated. The method provides the image unit with an opportunity to perform custom initialization, such as a registration check.

Availability

Mac OS X v10.4 and later.

Declared In

`CIPlugInInterface.h`

PART III

Other References

Core Image Filter Reference

Framework: QuartzCore

Companion guide Core Image Programming Guide

Overview

This reference describes the built-in filters available through the Core Image API. You can inspect filter parameters and see an image produced by a filter by using the CIFilterBrowser widget, available for ADC members from ADC Developer Connection Member Site.

Filters by Task

CICategoryBlur

`CIBoxBlur` (page 169)
Blurs an image using a box-shaped convolution kernel.

`CIDiscBlur` (page 192)
Blurs an image using a disc-shaped convolution kernel.

`CIGaussianBlur` (page 207)
Spreads source pixels by an amount specified by a Gaussian distribution.

`CIMedianFilter` (page 229)
Computes the median value for a group of neighboring pixels and replaces each pixel value with the median.

`CIMotionBlur` (page 232)
Blurs an image to simulate the effect of using a camera that moves a specified angle and distance while capturing the image.

`CINoiseReduction` (page 235)
Reduces noise using a threshold value to define what is considered noise.

`CIZoomBlur` (page 273)
Simulates the effect of zooming the camera while capturing the image.

CICategoryColorAdjustment

`CIColorControls` (page 178)
Adjusts saturation, brightness, and contrast values.

`CIColorMatrix` (page 182)
: Multiplies source color values and adds a bias factor to each color component.

`CIExposureAdjust` (page 200)
: Adjusts the exposure setting for an image similar to the way you control exposure for a camera when you change the F-stop.

`CIGammaAdjust` (page 206)
: Adjusts midtone brightness.

`CIHueAdjust` (page 217)
: Changes the overall hue, or tint, of the source pixels.

`CIWhitePointAdjust` (page 272)
: Adjusts the reference white point for an image and maps all colors in the source using the new reference.

CICategoryColorEffect

`CIColorCube` (page 179)
: Uses a three-dimensional color table to transform the source image pixels.

`CIColorInvert` (page 181)
: Inverts the colors in an image.

`CIColorMap` (page 182)
: Performs a nonlinear transformation of source color values using mapping values provided in a table.

`CIColorMonochrome` (page 183)
: Remaps colors so they fall within shades of a single color.

`CIColorPosterize` (page 184)
: Remaps red, green, and blue color components to the number of brightness values you specify for each color component.

`CIFalseColor` (page 200)
: Maps luminance to a color ramp of two colors.

`CIMaskToAlpha` (page 227)
: Converts a grayscale image to a white image that is masked by alpha.

`CIMaximumComponent` (page 228)
: Returns a grayscale image from max(r,g,b).

`CIMinimumComponent` (page 230)
: Returns a grayscale image from min(r,g,b).

`CISepiaTone` (page 250)
: Maps the colors of an image to various shades of brown.

CICategoryCompositeOperation

`CIAdditionCompositing` (page 160)
: Adds color components to achieve a brightening effect.

`CIColorBlendMode` (page 177)
: Uses the luminance values of the background with the hue and saturation values of the source image.

`CIColorBurnBlendMode` (page 177)
: Darkens the background image samples to reflect the source image samples.

`CIColorDodgeBlendMode` (page 180)
: Brightens the background image samples to reflect the source image samples.

`CIDarkenBlendMode` (page 190)
: Creates composite image samples by choosing the darker samples (from either the source image or the background).

`CIDifferenceBlendMode` (page 191)
: Subtracts either the source image sample color from the background image sample color, or the reverse, depending on which sample has the greater brightness value.

`CIExclusionBlendMode` (page 199)
: Produces an effect similar to that produced by the CIDifferenceBlendMode filter but with lower contrast.

`CIHardLightBlendMode` (page 212)
: Either multiplies or screens colors, depending on the source image sample color.

`CIHueBlendMode` (page 218)
: Uses the luminance and saturation values of the background with the hue of the source image.

`CILightenBlendMode` (page 222)
: Creates composite image samples by choosing the lighter samples (either from the source image or the background).

`CILuminosityBlendMode` (page 226)
: Uses the hue and saturation of the background with the luminance of the source image.

`CIMaximumCompositing` (page 228)
: Computes the maximum value, by color component, of two input images and creates an output image using the maximum values.

`CIMinimumCompositing` (page 230)
: Computes the minimum value, by color component, of two input images and creates an output image using the minimum values.

`CIMultiplyBlendMode` (page 233)
: Multiplies the source image samples with the background image samples.

`CIMultiplyCompositing` (page 234)
: Multiplies the color component of two input images and creates an output image using the multiplied values.

`CIOverlayBlendMode` (page 237)
: Either multiplies or screens the source image samples with the background image samples, depending on the background color.

`CISaturationBlendMode` (page 248)
: Uses the luminance and hue values of the background with the saturation of the source image.

`CIScreenBlendMode` (page 249)
: Multiplies the inverse of the source image samples with the inverse of the background image samples.

`CISoftLightBlendMode` (page 254)
: Either darkens or lightens colors, depending on the source image sample color.

`CISourceAtopCompositing` (page 255)
: Places the source image over the background image, then uses the luminance of the background image to determine what to show.

`CISourceInCompositing` (page 256)
Uses the second image to define what to leave in the source image, effectively cropping the image.

`CISourceOutCompositing` (page 257)
Uses the second image to define what to take out of the first image.

`CISourceOverCompositing` (page 257)
Places the second image over the first.

CICategoryDistortionEffect

`CIBumpDistortion` (page 170)
Creates a bump that originates at a specified point in the image.

`CIBumpDistortionLinear` (page 171)
Creates a concave or convex distortion that originates from a line in the image.

`CICircleSplashDistortion` (page 173)
Distorts the pixels starting at the circumference of a circle and emanating outward.

`CICircularWrap` (page 174)
Wraps an image around a transparent circle.

`CIDisplacementDistortion` (page 193)
Applies the grayscale values of the second image to the first image.

`CIGlassDistortion` (page 208)
Distorts an image by applying a glass-like texture.

`CIGlassLozenge` (page 209)
Creates a lozenge-shaped lens and distorts the portion of the image over which the lens is placed.

`CIHoleDistortion` (page 216)
Creates a circular area that pushes the image pixels outward, distorting those pixels closest to the circle the most.

`CIPinchDistortion` (page 242)
Creates a rectangular-shaped area that pinches source pixels inward, distorting those pixels closest to the rectangle the most.

`CITorusLensDistortion` (page 266)
Creates a torus-shaped lens and distorts the portion of the image over which the lens is placed.

`CITwirlDistortion` (page 269)
Rotates pixels around a point to give a twirling effect.

`CIVortexDistortion` (page 271)
Rotates pixels around a point to simulate a vortex.

CICategoryGenerator

`CICheckerboardGenerator` (page 172)
Generates a checkerboard pattern.

`CIConstantColorGenerator` (page 186)
Generates a solid color.

`CILenticularHaloGenerator` (page 220)
Simulates a halo that is generated by the diffraction associated with the spread of a lens.

CIRandomGenerator (page 246)
Generates an image of infinite extent whose pixel values are made up of four independent, uniformly-distributed random numbers in the 0 to 1 range.

CIStarShineGenerator (page 261)
Generates a starburst pattern.

CIStripesGenerator (page 262)
Generates a stripe pattern.

CISunbeamsGenerator (page 263)
Generates a sun effect.

CICategoryGeometryAdjustment

CIAffineTransform (page 163)
Applies an affine transform to an image.

CICrop (page 188)
Applies a crop to an image.

CILanczosScaleTransform (page 220)
Produces a high-quality, scaled version of a source image.

CIPerspectiveTransform (page 241)
Alters the geometry of an image to simulate the observer changing viewing position.

CICategoryGradient

CIGaussianGradient (page 208)
Generates a gradient that varies from one color to another using a Gaussian distribution.

CILinearGradient (page 223)
Generates a gradient that varies along a linear axis between two defined endpoints.

CIRadialGradient (page 245)
Generates a gradient that varies radially between two circles having the same center.

CICategoryHalftoneEffect

CICircularScreen (page 173)
Simulates a circular-shaped halftone screen.

CICMYKHalftone (page 175)
Creates a color, halftoned rendition of the source image, using cyan, magenta, yellow, and black inks over a white page.

CIDotScreen (page 195)
Simulates the dot patterns of a halftone screen.

CIHatchedScreen (page 213)
Simulates the hatched pattern of a halftone screen.

CILineScreen (page 225)
Simulates the line pattern of a halftone screen.

CICategoryReduction

`CIAreaAverage` (page 164)
Returns a single-pixel image that contains the average color for the region of interest.

`CIAreaHistogram` (page 164)
Returns a 1D image (`inputCount` wide by one pixel high) that contains the component-wise histogram computed for the specified rectangular area.

`CIRowAverage` (page 248)
Returns a 1-pixel high image that contains the average color for each scan row.

`CIColumnAverage` (page 185)
Returns a 1-pixel high image that contains the average color for each scan column.

`CIAreaMaximum` (page 164)
Returns a single-pixel image that contains the maximum color components for the region of interest.

`CIAreaMinimum` (page 165)
Returns a single-pixel image that contains the minimum color components for the region of interest.

`CIAreaMaximumAlpha` (page 165)
Returns a single-pixel image that contains the color vector with the maximum alpha value for the region of interest.

`CIAreaMinimumAlpha` (page 166)
Returns a single-pixel image that contains the color vector with the minimum alpha value for the region of interest.

CICategorySharpen

`CISharpenLuminance` (page 252)
Increases image detail by sharpening.

`CIUnsharpMask` (page 270)
Increases the contrast of the edges between pixels of different colors in an image.

CICategoryStylize

`CIBlendWithMask` (page 167)
Uses values from a grayscale mask to interpolate between an image and the background.

`CIBloom` (page 168)
Softens edges and applies a pleasant glow to an image.

`CIComicEffect` (page 185)
Simulates a comic book drawing by outlining edges and applying a color halftone effect.

`CICrystallize` (page 189)
Creates polygon-shaped color blocks by aggregating source pixel-color values.

`CIEdges` (page 196)
Finds all edges in an image and displays them in color.

`CIEdgeWork` (page 197)
Produces a stylized black-and-white rendition of an image that looks similar to a woodblock cutout.

CIGloom (page 211)
: Dulls the highlights of an image.

CIHeightFieldFromMask (page 214)
: Produces a continuous three-dimensional, loft-shaped height field from a grayscale mask.

CIHexagonalPixellate (page 215)
: Maps an image to colored hexagons whose color is defined by the replaced pixels.

CILineOverlay (page 224)
: Creates a sketch that outlines the edges of an image in black.

CIPixellate (page 243)
: Makes an image blocky by mapping the image to colored squares whose color is defined by the replaced pixels.

CIPointillize (page 244)
: Renders the source image in a pointillistic style.

CIShadedMaterial (page 251)
: Produces a shaded image from a height field.

CISpotColor (page 258)
: Replaces one or more color ranges with spot colors.

CISpotLight (page 260)
: Applies a directional spotlight effect to an image.

CICategoryTileEffect

CIAffineClamp (page 161)
: Performs an affine transform on a source image and then clamps the pixels at the edge of the transformed image, extending them outwards.

CIAffineTile (page 162)
: Applies an affine transform to an image and then tiles the transformed image.

CIEightfoldReflectedTile (page 198)
: Produces a tiled image from a source image by applying an 8-way reflected symmetry.

CIFourfoldReflectedTile (page 203)
: Produces a tiled image from a source image by applying a 4-way reflected symmetry.

CIFourfoldRotatedTile (page 204)
: Produces a tiled image from a source image by rotating the source image at increments of 90 degrees.

CIFourfoldTranslatedTile (page 205)
: Produces a tiled image from a source image by applying 4 translation operations.

CIGlideReflectedTile (page 210)
: Produces a tiled image from a source image by translating and smearing the image.

CIKaleidoscope (page 219)
: Produces a kaleidoscopic image from a source image by applying 12-way symmetry.

CIOpTile (page 236)
: Segments an image, applying any specified scaling and rotation, and then assembles the image again to give an op art appearance.

CIParallelogramTile (page 239)
: Warps an image by reflecting it in a parallelogram, and then tiles the result.

`CIPerspectiveTile` (page 240)
    Applies a perspective transform to an image and then tiles the result.

`CISixfoldReflectedTile` (page 253)
    Produces a tiled image from a source image by applying a 6-way reflected symmetry.

`CISixfoldRotatedTile` (page 253)
    Produces a tiled image from a source image by rotating the source image at increments of 60 degrees.

`CITriangleTile` (page 267)
    Maps a triangular portion of image to a triangular area and then tiles the result.

`CITwelvefoldReflectedTile` (page 268)
    Produces a tiled image from a source image by rotating the source image at increments of 30 degrees.

CICategoryTransition

`CIBarsSwipeTransition` (page 166)
    Transitions from one image to another by passing a bar over the source image.

`CICopyMachineTransition` (page 187)
    Transitions from one image to another by simulating the effect of a copy machine.

`CIDisintegrateWithMaskTransition` (page 192)
    Transitions from one image to another using the shape defined by a mask.

`CIDissolveTransition` (page 194)
    Uses a dissolve to transition from one image to another.

`CIFlashTransition` (page 201)
    Transitions from one image to another by creating a flash.

`CIModTransition` (page 231)
    Transitions from one image to another by revealing the target image through irregularly shaped holes.

`CIPageCurlTransition` (page 238)
    Transitions from one image to another by simulating a curling page, revealing the new image as the page curls.

`CIRippleTransition` (page 247)
    Transitions from one image to another by creating a circular wave that expands from the center point, revealing the new image in the wake of the wave.

`CISwipeTransition` (page 265)
    Transitions from one image to another by simulating a swiping action.

Filters

CIAdditionCompositing

Adds color components to achieve a brightening effect.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

A `CIImage` class whose display name is Background Image.

Discussion

This filter is typically used to add highlights and lens flare effects. The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

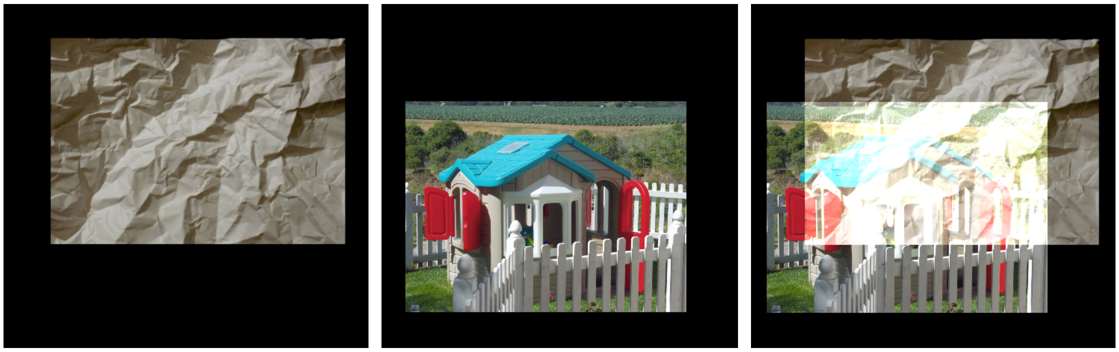
Member of

`CICategoryBuiltIn`, `CICategoryHighDynamicRange`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name

Addition

Figure 16-1 The result of using the CIAdditionCompositing filter



Availability

Available in Mac OS X v10.4 and later.

CIAffineClamp

Performs an affine transform on a source image and then clamps the pixels at the edge of the transformed image, extending them outwards.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputTransform

An `NSAffineTransform` class whose display name is Transform.

Discussion

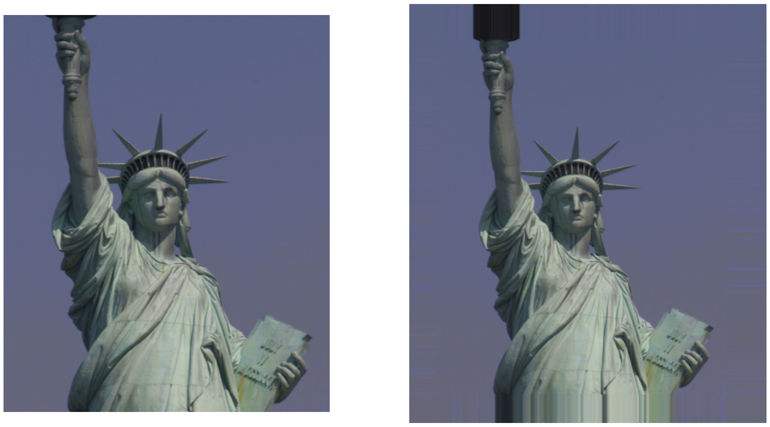
This filter performs similarly to the CIAffineTransform filter except that it produces an image with infinite extent. You can use this filter when you need to blur an image but you want to avoid a soft, black fringe along the edges.

Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
Affine Clamp

Figure 16-2 The result of using the CIAffineClamp filter



Availability
Available in Mac OS X v10.4 and later.

CIAffineTile

Applies an affine transform to an image and then tiles the transformed image.

Parameters

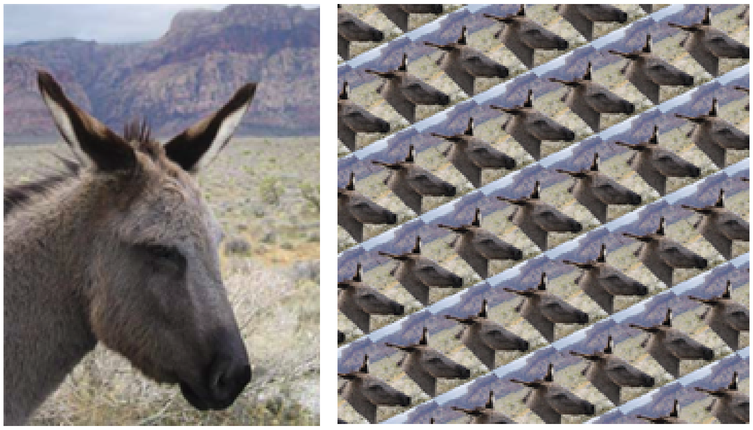
`inputImage`
A `CIImage` class whose display name is Image.

`inputTransform`
An `NSAffineTransform` class whose display name is Transform.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
Affine Tile

Figure 16-3 The result of using the CIAffineTile filter



Availability
Available in Mac OS X v10.4 and later.

CIAffineTransform

Applies an affine transform to an image.

Parameters

`inputImage`
A `CIImage` class whose display name is Image.

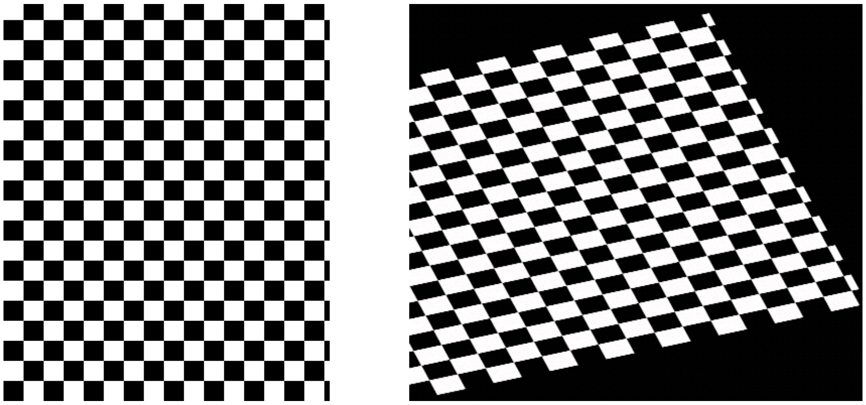
`inputTransform`
An `NSAffineTransform` class whose display name is Transform.

Discussion
You can scale, translate, or rotate the input image. You can also apply a combination of these operations.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGeometryAdjustment`

Localized Display Name
Affine Transform

Figure 16-4 The result of using the CIAffineTransform filter



Availability
Available in Mac OS X v10.4 and later.

CIAreaAverage

Returns a single-pixel image that contains the average color for the region of interest.

Parameters

`inputImage`
A `CIImage` class whose display name is Image. This is the image data you want to process.

`inputExtent`
The rectangular region of interest.

Member of
`CICategoryReduction`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBuiltIn`

Localized Display Name
Area Average

Availability
Available in Mac OS X v10.5 and later.

CIAreaHistogram

Returns a 1D image (`inputCount` wide by one pixel high) that contains the component-wise histogram computed for the specified rectangular area.

Parameters

`inputImage`
A `CIImage` class whose display name is Image. This is the image data you want to process.

`inputExtent`
The rectangular region of interest.

`inputCount`
The number of "buckets" for the histogram.

`inputScale`
A scaling factor. Core Image scales the histogram by dividing the scale by the area of the `inputExtent` rectangle.

Member of
`CICategoryReduction`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBuiltIn`

Localized Display Name
Area Histogram

Availability
Available in Mac OS X v10.5 and later.

CIAreaMaximum

Returns a single-pixel image that contains the maximum color components for the region of interest.

Parameters

inputImage

A CIImage class whose display name is Image. This is the image data you want to process.

inputExtent

The rectangular region of interest.

Member of

CICategoryReduction, CICategoryStillImage, CICategoryVideo, CICategoryBuiltIn

Localized Display Name

Area Maximum

Availability

Available in Mac OS X v10.5 and later.

CIAreaMaximumAlpha

Returns a single-pixel image that contains the color vector with the maximum alpha value for the region of interest.

Parameters

inputImage

A CIImage class whose display name is Image. This is the image data you want to process.

inputExtent

The rectangular region of interest.

Discussion

If more than one pixel exists with the maximum alpha value, Core Image returns the vector that has the lowest *x* and y coordinate.

Member of

CICategoryReduction, CICategoryStillImage, CICategoryVideo, CICategoryBuiltIn

Localized Display Name

Area Maximum Alpha

Availability

Available in Mac OS X v10.5 and later.

CIAreaMinimum

Returns a single-pixel image that contains the minimum color components for the region of interest.

Parameters

inputImage

A CIImage class whose display name is Image. This is the image data you want to process.

inputExtent

The rectangular region of interest.

Member of

CICategoryReduction, CICategoryStillImage, CICategoryVideo, CICategoryBuiltIn

Localized Display Name

Area Minimum

Availability
Available in Mac OS X v10.5 and later.

CIAreaMinimumAlpha

Returns a single-pixel image that contains the color vector with the minimum alpha value for the region of interest.

Parameters

inputImage

A CIImage class whose display name is Image. This is the image data you want to process.

inputExtent

The rectangular region of interest.

Discussion
If more than one pixel exists with the minimum alpha value, Core Image returns the vector that has the lowest *x* and y coordinate.

Member of
CICategoryReduction, CICategoryStillImage, CICategoryVideo, CICategoryBuiltIn

Localized Display Name
Area Minimum Alpha

Availability
Available in Mac OS X v10.5 and later.

CIBarsSwipeTransition

Transitions from one image to another by passing a bar over the source image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputTargetImage

A CIImage class whose display name is Target Image.

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 3.14 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 6.28 Identity: 0.00

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 30.00 Minimum: 2.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 300.00 Identity: 30.00

inputBarOffset

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Bar Offset.

Default value: 10.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 100.00 Identity: 10.00

inputTime

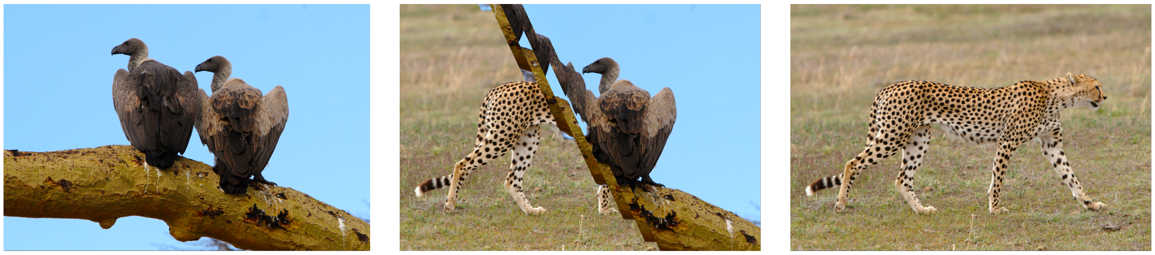
An `NSNumber` class whose attribute type is `CIAttributeTypeTime` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name
CIBarsSwipeTransition

Figure 16-5 The result of using the CIBarsSwipeTransition filter



Availability
Available in Mac OS X v10.5 and later.

CIBlendWithMask

Uses values from a grayscale mask to interpolate between an image and the background.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

A `CIImage` class whose display name is Background Image.

inputMaskImage

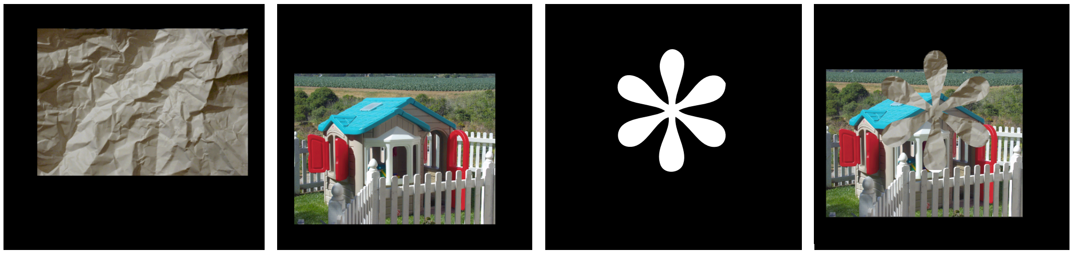
A `CIImage` class whose display name is Mask Image.

Discussion
When a mask value is 0.0, the result is the background. When the mask value is 1.0, the result is the image.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Blend With Mask

Figure 16-6 The result of using the CIBlendWithMask filter



Availability
Available in Mac OS X v10.4 and later.

CIBloom

Softens edges and applies a pleasant glow to an image.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputRadius`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 0.00

`inputIntensity`

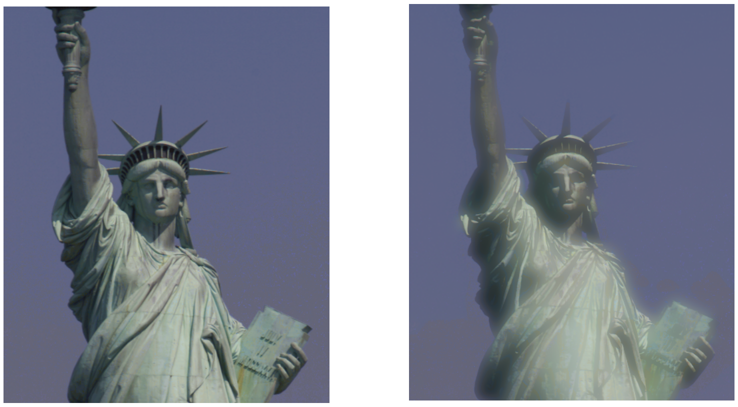
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Bloom

Figure 16-7 The result of using the CIBloom filter



Availability
Available in Mac OS X v10.4 and later.

CIBoxBlur

Blurs an image using a box-shaped convolution kernel.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputRadius

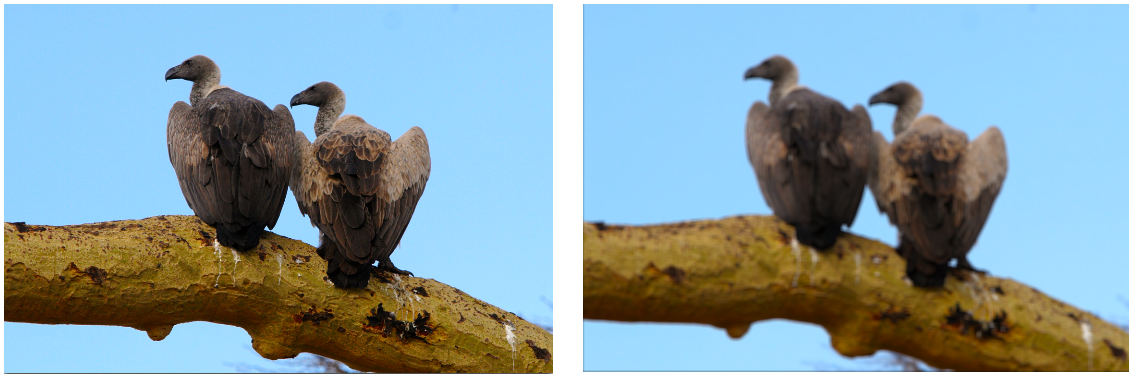
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 100.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name
CIBoxBlur

Figure 16-8 The result of using the CIBoxBlur filter



Availability
Available in Mac OS X v10.5 and later.

CIBumpDistortion

Creates a bump that originates at a specified point in the image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 300.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 600.00 Identity: 300.00

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Scale.

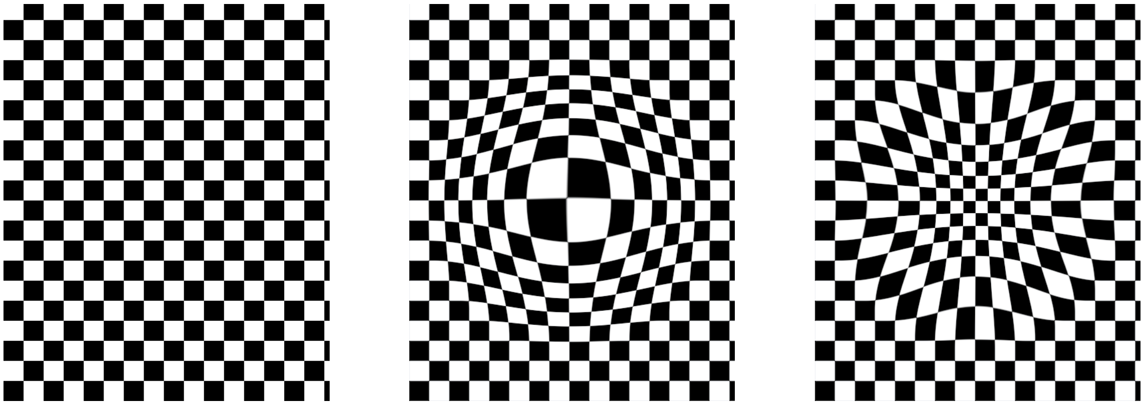
Default value: 0.50 Minimum: -1.00 Maximum: 0.00 Slider minimum: -1.00 Slider maximum: 1.00 Identity: 0.00

Discussion
The bump can be concave or convex.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name
Bump Distortion

Figure 16-9 The result of using the CIBumpDistortion filter



Availability
Available in Mac OS X v10.4 and later.

CIBumpDistortionLinear

Creates a concave or convex distortion that originates from a line in the image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [300 300] Identity: (null)

inputRadius

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Radius.

Default value: 300.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 600.00 Identity: 300.00

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 6.28 Identity: 0.00

inputScale

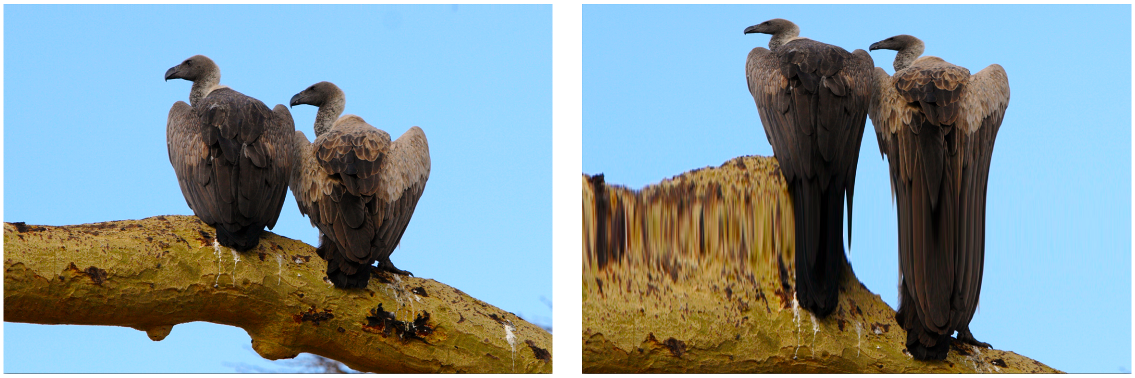
An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Scale.

Default value: 0.50 Minimum: -1.00 Maximum: 0.00 Slider minimum: -1.00 Slider maximum: 1.00 Identity: 1.00

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryDistortionEffect

Localized Display Name
CIBumpDistortionLinear

Figure 16-10 The result of using the CIBumpDistortionLinear filter



Availability
Available in Mac OS X v10.5 and later.

CICheckerboardGenerator

Generates a checkerboard pattern.

Parameters

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputColor0

A `CIColor` class whose display name is Color 1.

inputColor1

A `CIColor` class whose display name is Color 2.

inputWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 80.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 80.00

inputSharpness

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Sharpness.

Default value: 1.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

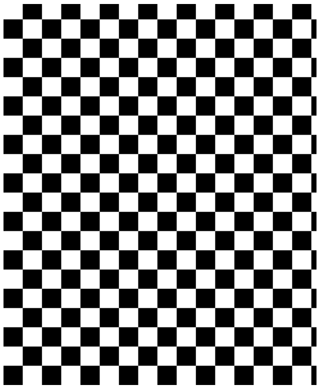
Discussion

You can specify the checkerboard size and colors, and the sharpness of the pattern.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGenerator`

Localized Display Name
Checkerboard

Figure 16-11 The result of using the CICheckerboardGenerator filter



Availability

Available in Mac OS X v10.4 and later.

CICircleSplashDistortion

Distorts the pixels starting at the circumference of a circle and emanating outward.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 150.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1000.00 Identity: 0.10

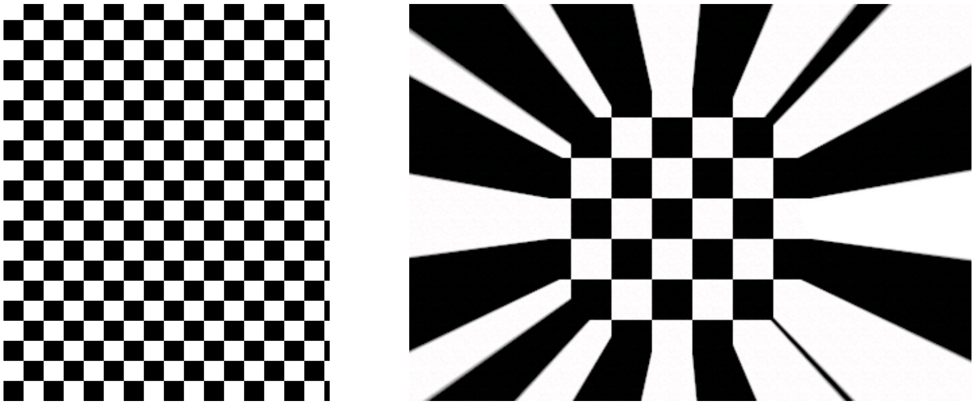
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name

Circle Splash Distortion

Figure 16-12 The result of using the CICircleSplashDistortion filter



Availability

Available in Mac OS X v10.4 and later.

CICircularScreen

Simulates a circular-shaped halftone screen.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 6.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 50.00 Identity: 0.00

inputSharpness

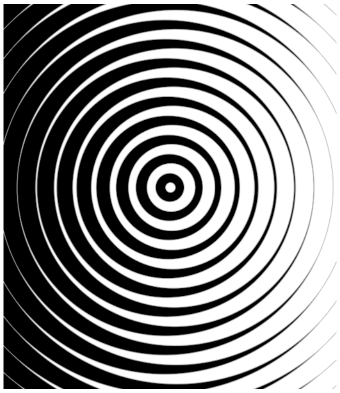
An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Sharpness.

Default value: 0.70 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryHalftoneEffect

Localized Display Name
Circular Screen

Figure 16-13 The result of using the CICircularScreen filter



Availability
Available in Mac OS X v10.4 and later.

CICircularWrap

Wraps an image around a transparent circle.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Radius.

Default value: 150.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 600.00 Identity: 0.00

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

Discussion

The distortion of the image increases with the distance from the center of the circle.

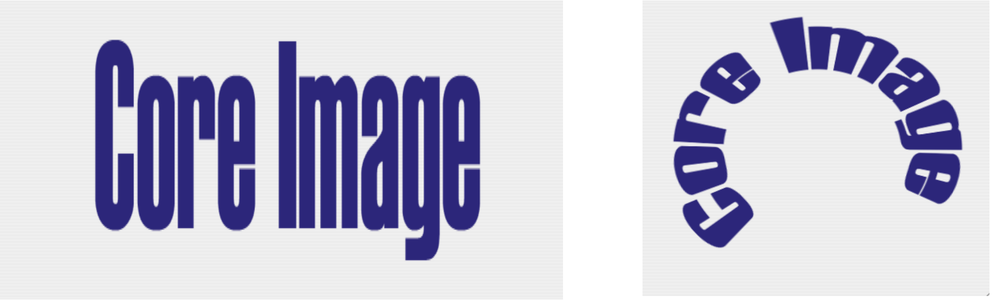
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryDistortionEffect

Localized Display Name

Circular Wrap Distortion

Figure 16-14 The result of using the CICircularWrap filter



Availability

Available in Mac OS X v10.4 and later.

CICMYKHalftone

Creates a color, halftoned rendition of the source image, using cyan, magenta, yellow, and black inks over a white page.

Parameters

inputImage

A CIImage class whose display name is Image.

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputWidth`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 6.00 Minimum: 2.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 100.00 Identity: 6.00

`inputAngle`

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

`inputSharpness`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Sharpness.

Default value: 0.70 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

`inputGCR`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Gray Component Replacement.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 1.00

`inputUCR`

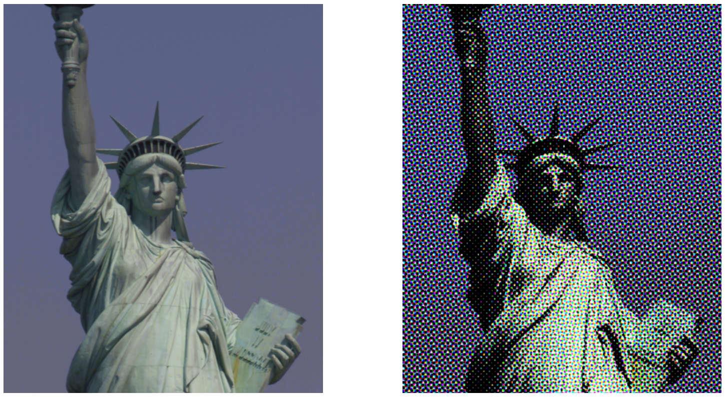
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Under Color Removal.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.50

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryHalftoneEffect`

Localized Display Name
CMYK Halftone

Figure 16-15 The result of using the CICMYKHalftone filter



Availability

Available in Mac OS X v10.4 and later.

CIColorBlendMode

Uses the luminance values of the background with the hue and saturation values of the source image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputBackgroundImage

A CIImage class whose display name is Background Image.

Discussion

This mode preserves the gray levels in the image. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

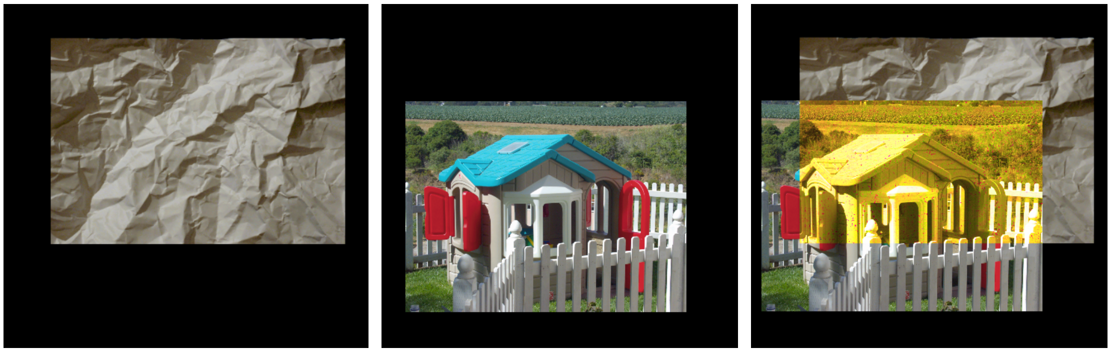
Member of

CICategoryBuiltIn, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryStillImage, CICategoryVideo, CICategoryCompositeOperation

Localized Display Name

Color Blend Mode

Figure 16-16 The result of using the CIColorBlendMode filter



Availability

Available in Mac OS X v10.4 and later.

CIColorBurnBlendMode

Darkens the background image samples to reflect the source image samples.

Parameters

inputImage

A CIImage class whose display name is Image.

inputBackgroundImage

A CIImage class whose display name is Background Image.

Discussion

Source image sample values that specify white do not produce a change. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

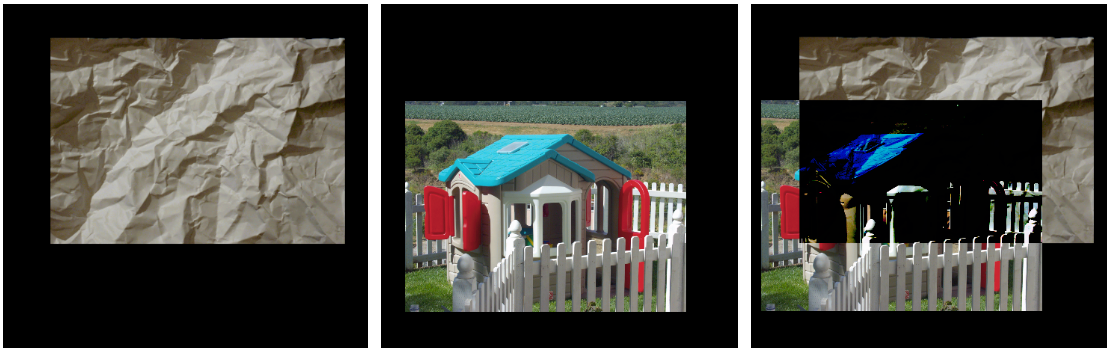
Member of

`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name

Color Burn Blend Mode

Figure 16-17 The result of using the CIColorBurnBlendMode filter



Availability

Available in Mac OS X v10.4 and later.

CIColorControls

Adjusts saturation, brightness, and contrast values.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputSaturation`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Saturation.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 2.00 Identity: 1.00

`inputBrightness`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Brightness.

Default value: 0.00 Minimum: -1.00 Maximum: 0.00 Slider minimum: -1.00 Slider maximum: 1.00 Identity: 0.00

inputContrast

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Contrast.

Default value: 1.00 Minimum: 0.25 Maximum: 0.00 Slider minimum: 0.25 Slider maximum: 4.00 Identity: 1.00

Discussion

To calculate saturation, this filter linearly interpolates between a grayscale image (saturation = `0.0`) and the original image (saturation = `1.0`). The filter supports extrapolation: For values large than `1.0`, it increases saturation.

To calculate contrast, this filter uses the following formula:

```
(color.rgb - vec3(0.5)) * contrast + vec3(0.5)
```

This filter calculates brightness by adding a bias value:

```
color.rgb + vec3(brightness)
```

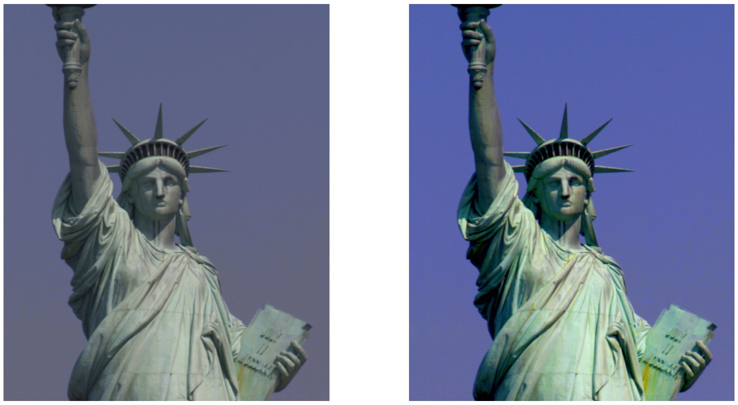
Member of

`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryColorAdjustment`

Localized Display Name

Color Controls

Figure 16-18 The result of using the CIColorControls filter



Availability

Available in Mac OS X v10.4 and later.

CIColorCube

Uses a three-dimensional color table to transform the source image pixels.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCubeDimension

An NSNumber class whose attribute type is CIAttributeTypeCount and whose display name is Cube Dimension.

Default value: 2.00 Minimum: 2.00 Maximum: 128.00 Slider minimum: 0.00 Slider maximum: 0.00 Identity: 2.00 The value must be a power of two.

inputCubeData

An NSData class whose display name is Cube Data.

Discussion

The color table must be composed of floating-point RGBA cells that use premultiplied alpha. The cells are organized in a standard ordering. The columns and rows of the data are indexed by red and green, respectively. Each data plane is followed by the next higher plane in the data, with planes indexed by blue.

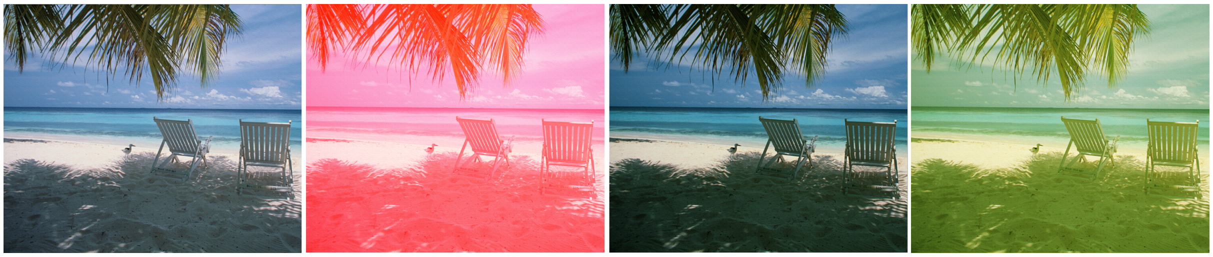
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryVideo, CICategoryColorEffect

Localized Display Name

Color Cube

Figure 16-19 The result of using the CIColorCube filter



Availability

Available in Mac OS X v10.4 and later.

CIColorDodgeBlendMode

Brightens the background image samples to reflect the source image samples.

Parameters

inputImage

A CIImage class whose display name is Image.

inputBackgroundImage

A CIImage class whose display name is Background Image.

Discussion

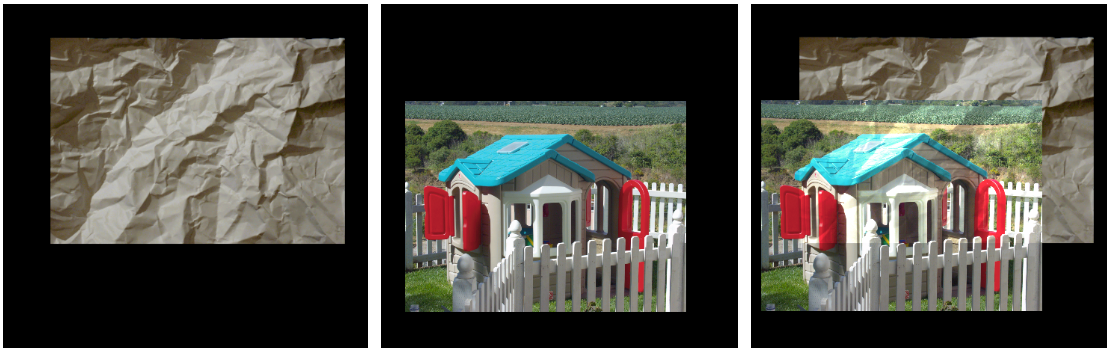
Source image sample values that specify black do not produce a change. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of

CICategoryBuiltIn, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryStillImage, CICategoryVideo, CICategoryCompositeOperation

Localized Display Name
Color Dodge Blend Mode

Figure 16-20 The result of using the CIColorDodgeBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIColorInvert

Inverts the colors in an image.

Parameters

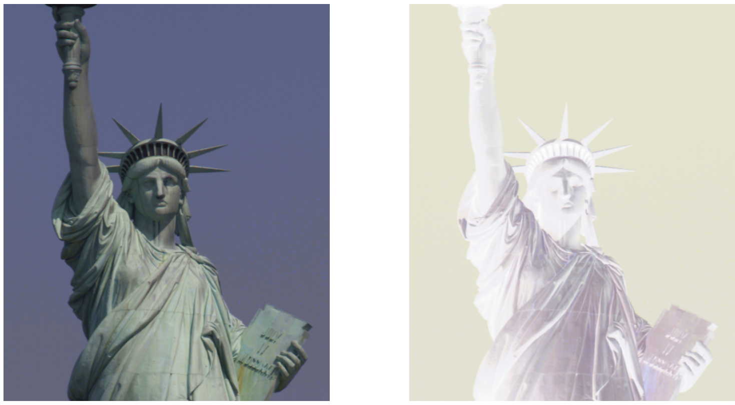
`inputImage`

A `CIImage` class whose display name is Image.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryVideo`, `CICategoryColorEffect`

Localized Display Name
Color Invert

Figure 16-21 The result of using the CIColorInvert filter



Availability
Available in Mac OS X v10.4 and later.

CIColorMap

Performs a nonlinear transformation of source color values using mapping values provided in a table.

Parameters

inputImage

A CIImage class whose display name is Image.

inputGradientImage

A CIImage class whose attribute type is CIAttributeTypeGradient and whose display name is Gradient Image.

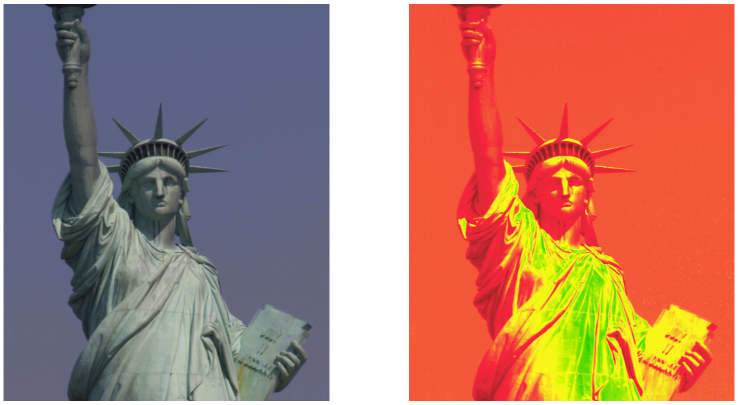
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryVideo, CICategoryColorEffect

Localized Display Name

Color Map

Figure 16-22 The result of using the CIColorMap filter



Availability

Available in Mac OS X v10.4 and later.

CIColorMatrix

Multiplies source color values and adds a bias factor to each color component.

Parameters

inputImage

A CIImage class whose display name is Image.

inputRVector

A CIVector class whose display name is Red Vector.

Default value: [1 0 0 0] Identity: [1 0 0 0]

inputGVector

A CIVector class whose display name is Green Vector.

Default value: [0 1 0 0] Identity: [0 1 0 0]

inputBVector

A CIVector class whose display name is Blue Vector.

Default value: [0 0 1 0] Identity: [0 0 1 0]

inputAVector

A CIVector class whose display name is Alpha Vector.

Default value: [0 0 0 1] Identity: [0 0 0 1]

inputBiasVector

A CIVector class whose display name is Bias Vector.

Default value: [0 0 0 0] Identity: [0 0 0 0]

Discussion

This filter performs a matrix multiplication, as follows, to transform the color vector:

```
s.r = dot(s, redVector)
s.g = dot(s, greenVector)
s.b = dot(s, blueVector)
s.a = dot(s, alphaVector)
s = s + bias
```

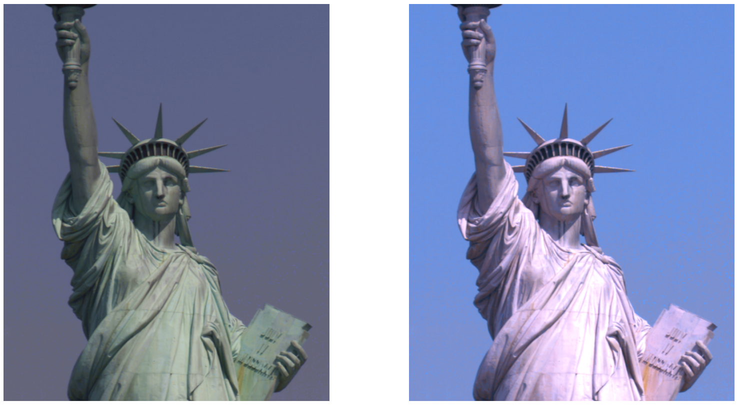
Member of

CICategoryBuiltIn, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryStillImage, CICategoryVideo, CICategoryColorAdjustment

Localized Display Name

Color Matrix

Figure 16-23 The result of using the CIColorMatrix filter



Availability

Available in Mac OS X v10.4 and later.

CIColorMonochrome

Remaps colors so they fall within shades of a single color.

Parameters

inputImage

A CIImage class whose display name is Image.

inputColor

A CIColor class whose attribute type is CIAttributeTypeOpaqueColor and whose display name is Color.

inputIntensity

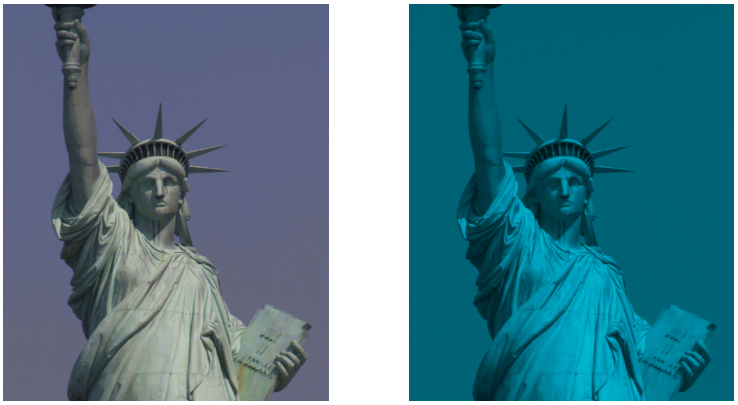
An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Intensity.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryVideo, CICategoryColorEffect

Localized Display Name
Color Monochrome

Figure 16-24 The result of using the CIColorMonochrome filter



Availability
Available in Mac OS X v10.4 and later.

CIColorPosterize

Remaps red, green, and blue color components to the number of brightness values you specify for each color component.

Parameters

inputImage

A CIImage class whose display name is Image.

inputLevels

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Levels.

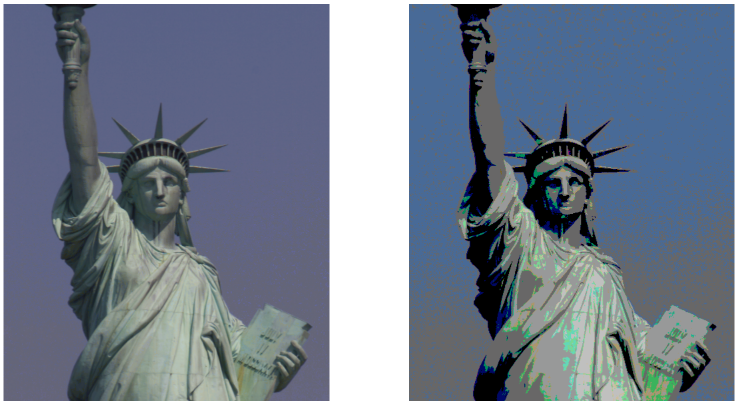
Default value: 6.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 30.00 Identity: 300.00

Discussion
This filter flattens colors to achieve a look similar to that of a silk-screened poster.

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryVideo, CICategoryColorEffect

Localized Display Name
Color Posterize

Figure 16-25 The result of using the CIColorPosterize filter



Availability
Available in Mac OS X v10.4 and later.

CIColumnAverage

Returns a 1-pixel high image that contains the average color for each scan column.

Parameters

inputImage

A CIImage class whose display name is Image. This is the image data you want to process.

inputExtent

The rectangular region of interest.

Member of
CICategoryReduction, CICategoryStillImage, CICategoryVideo, CICategoryBuiltIn

Localized Display Name
Column Average

Availability
Available in Mac OS X v10.5 and later.

CIComicEffect

Simulates a comic book drawing by outlining edges and applying a color halftone effect.

Parameters

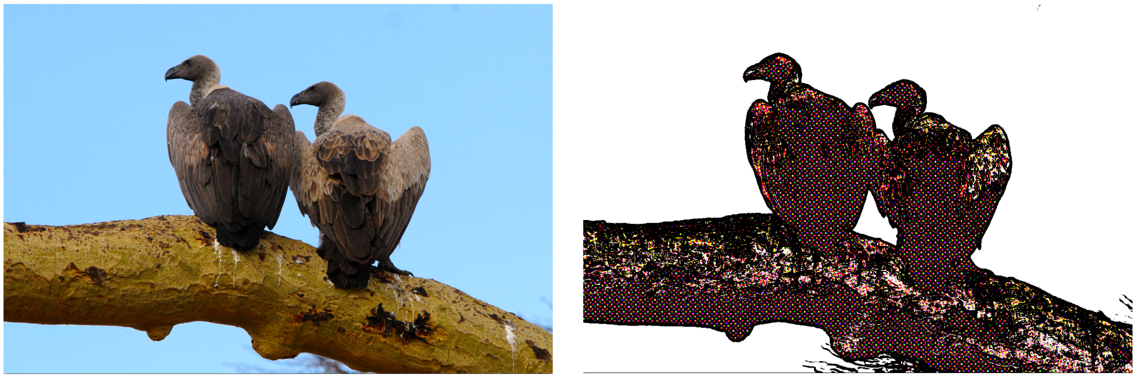
inputImage

A CIImage class whose display name is Image.

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryStylize

Localized Display Name
Comic Effect

Figure 16-26 The result of using the CIComicEffect filter



Availability
Available in Mac OS X v10.5 and later.

CIConstantColorGenerator

Generates a solid color.

Parameters

inputColor
A CIColor class whose display name is Color.

Discussion
You typically use the output of this filter as the input to another filter.

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryGenerator

Localized Display Name
Constant Color

Figure 16-27 The result of using the CIConstantColorGenerator filter

Availability
Available in Mac OS X v10.4 and later.

CICopyMachineTransition

Transitions from one image to another by simulating the effect of a copy machine.

Parameters

inputImage

A CIImage class whose display name is Image.

inputTargetImage

A CIImage class whose display name is Target Image.

inputExtent

A CIVector class whose attribute type is CIAttributeTypeRectangle and whose display name is Extent.

Default value: [0 0 300 300] Identity: (null)

inputColor

A CIColor class whose attribute type is CIAttributeTypeOpaqueColor and whose display name is Color.

inputTime

An NSNumber class whose attribute type is CIAttributeTypeTime and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 6.28 Identity: 0.00

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 200.00 Minimum: 0.10 Maximum: 0.00 Slider minimum: 0.10 Slider maximum: 500.00 Identity: 200.00

inputOpacity

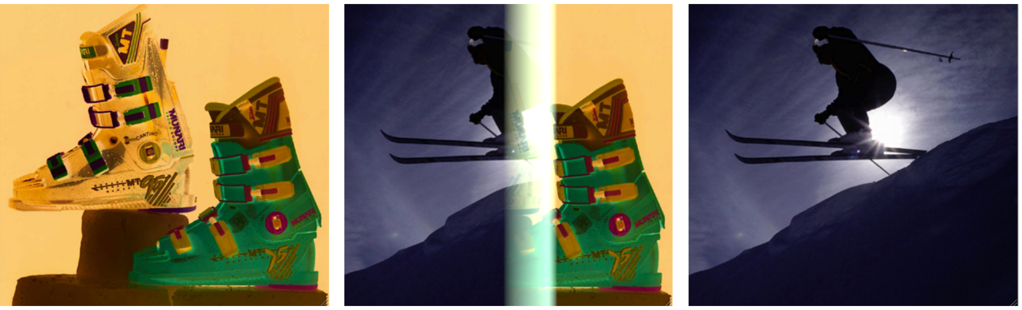
An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Opacity.

Default value: 1.30 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 3.00 Identity: 1.30

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryTransition

Localized Display Name
Copy Machine

Figure 16-28 The result of using the CICopyMachineTransition filter



Availability
Available in Mac OS X v10.4 and later.

CICrop

Applies a crop to an image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputRectangle

A CIVector class whose attribute type is CIAttributeTypeRectangle and whose display name is Rectangle.

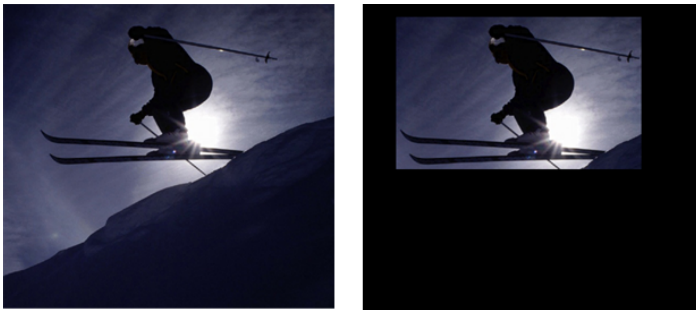
Default value: [0 0 300 300] Identity: (null)

Discussion
The size and shape of the cropped image depend on the rectangle you specify.

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryGeometryAdjustment

Localized Display Name
Crop

Figure 16-29 The result of using the CICrop filter



Availability
Available in Mac OS X v10.4 and later.

CICrystallize

Creates polygon-shaped color blocks by aggregating source pixel-color values.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputRadius`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 20.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 100.00 Identity: 1.00

`inputCenter`

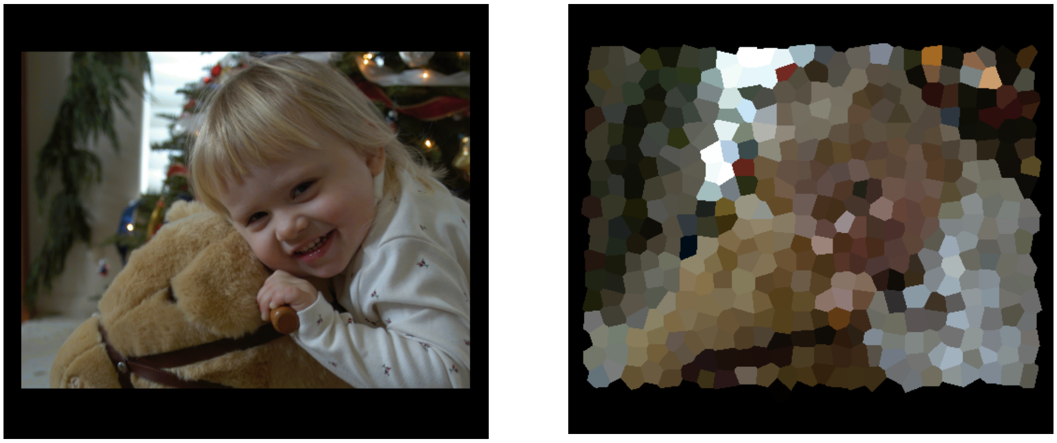
A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Crystallize

Figure 16-30 The result of using the CICrystallize filter



Availability
Available in Mac OS X v10.4 and later.

CIDarkenBlendMode

Creates composite image samples by choosing the darker samples (from either the source image or the background).

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

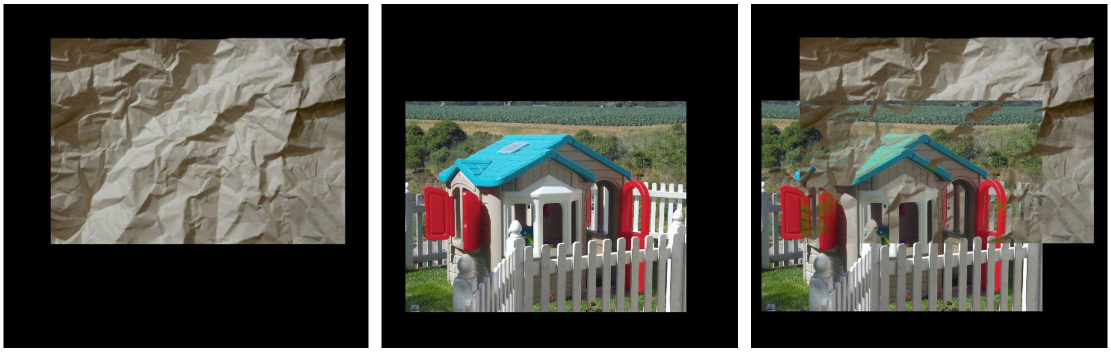
A `CIImage` class whose display name is Background Image.

Discussion
The result is that the background image samples are replaced by any source image samples that are darker. Otherwise, the background image samples are left unchanged. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Darken Blend Mode

Figure 16-31 The result of using the CIDarkenBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIDifferenceBlendMode

Subtracts either the source image sample color from the background image sample color, or the reverse, depending on which sample has the greater brightness value.

Parameters

inputImage
A `CIImage` class whose display name is Image.

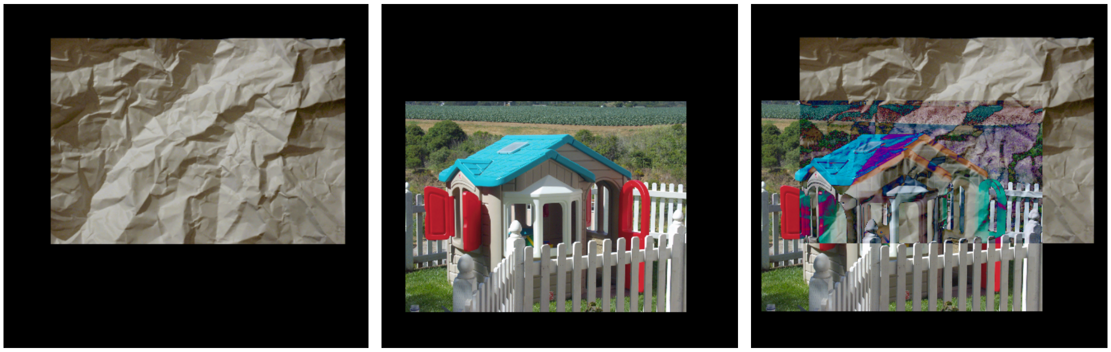
inputBackgroundImage
A `CIImage` class whose display name is Background Image.

Discussion
Source image sample values that are black produce no change; white inverts the background color values. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Difference Blend Mode

Figure 16-32 The result of using the CIDifferenceBlendMode filter



Availability

Available in Mac OS X v10.4 and later.

CIDiscBlur

Blurs an image using a disc-shaped convolution kernel.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 8.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 0.00

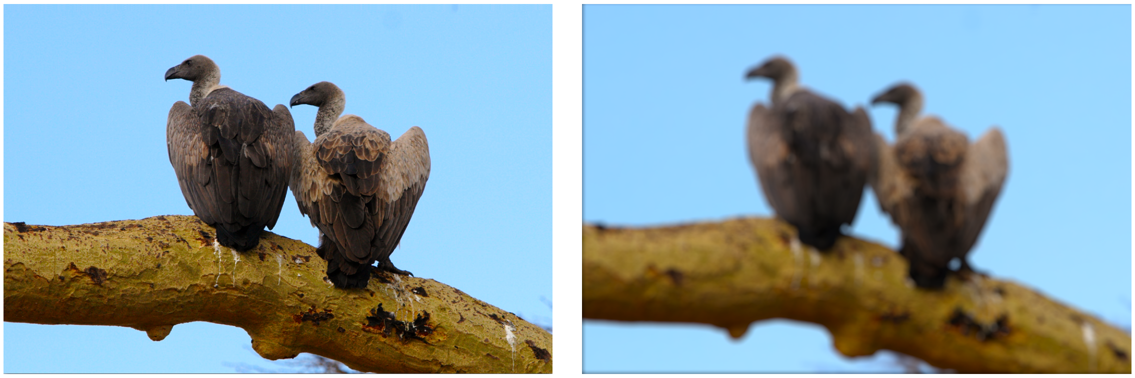
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name

CIDiscBlur

Figure 16-33 The result of using the CIDiscBlur filter



Availability

Available in Mac OS X v10.5 and later.

CIDisintegrateWithMaskTransition

Transitions from one image to another using the shape defined by a mask.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputTargetImage

A `CIImage` class whose display name is Target Image.

inputMaskImage

A CIImage class whose display name is Mask Image.

inputTime

An NSNumber class whose attribute type is CIAttributeTypeTime and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputShadowRadius

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Shadow Radius.

Default value: 8.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 50.00 Identity: 0.00

inputShadowDensity

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Shadow Density.

Default value: 0.65 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputShadowOffset

A CIVector class whose attribute type is CIAttributeTypeOffset and whose display name is Shadow Offset.

Default value: [0 -10] Identity: [0 0]

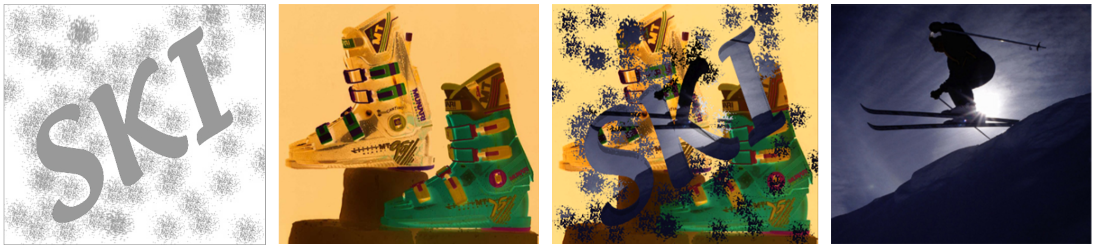
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryTransition

Localized Display Name

Disintegrate with Mask

Figure 16-34 The result of using the CIDisintegrateWithMaskTransition filter



Availability

Available in Mac OS X v10.4 and later.

CIDisplacementDistortion

Applies the grayscale values of the second image to the first image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputDisplacementImage

A `CIImage` class whose display name is Displacement Image.

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Scale.

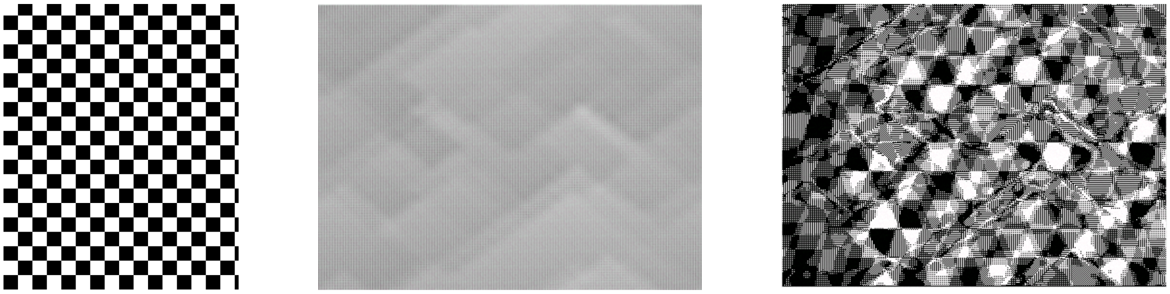
Default value: 50.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 200.00 Identity: 0.00

Discussion
The output image has a texture defined by the grayscale values.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name
Displacement Distortion

Figure 16-35 The result of using the CIDisplacementDistortion filter



Availability
Available in Mac OS X v10.4 and later.

CIDissolveTransition

Uses a dissolve to transition from one image to another.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputTargetImage

A `CIImage` class whose display name is Target Image.

inputTime

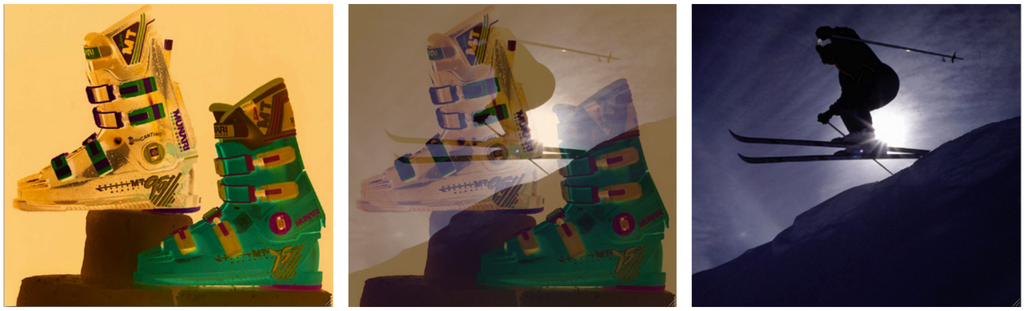
An `NSNumber` class whose attribute type is `CIAttributeTypeTime` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name
Dissolve

Figure 16-36 The result of using the CIDissolveTransition filter



Availability
Available in Mac OS X v10.4 and later.

CIDotScreen

Simulates the dot patterns of a halftone screen.

Parameters

inputImage
A CIImage class whose display name is Image.

inputCenter
A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle
An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth
An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 6.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 50.00 Identity: 0.00

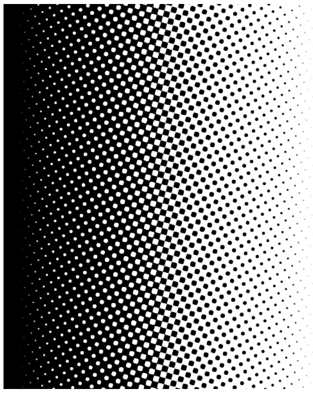
inputSharpness
An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Sharpness.

Default value: 0.70 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryHalftoneEffect

Localized Display Name
Dot Screen

Figure 16-37 The result of using the CIDotScreen filter



Availability
Available in Mac OS X v10.4 and later.

CIEdges

Finds all edges in an image and displays them in color.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputIntensity`

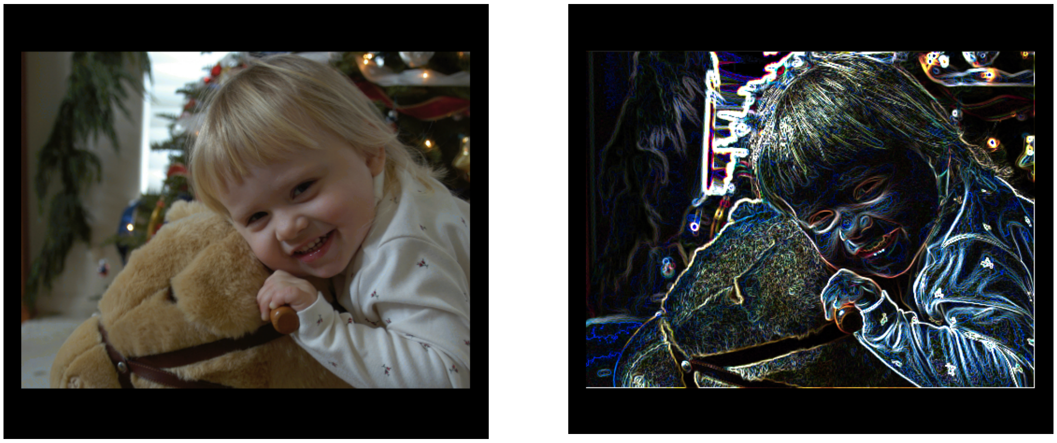
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 10.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Edges

Figure 16-38 The result of using the CIEdges filter



Availability
Available in Mac OS X v10.4 and later.

CIEdgeWork

Produces a stylized black-and-white rendition of an image that looks similar to a woodblock cutout.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputRadius`

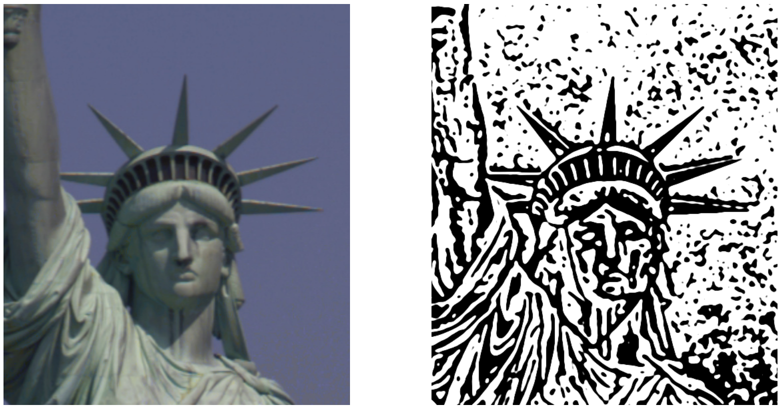
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 3.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 20.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Edge Work

Figure 16-39 The result of using the CIEdgeWork filter



Availability
Available in Mac OS X v10.4 and later.

CIEightfoldReflectedTile

Produces a tiled image from a source image by applying an 8-way reflected symmetry.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputAngle`

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

`inputWidth`

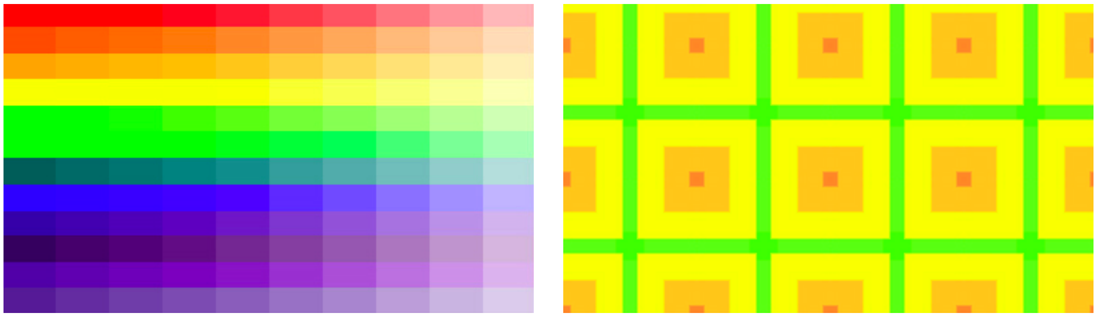
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
CIEightfoldReflectedTile

Figure 16-40 The result of using the CIEightfoldReflectedTile filter



Availability
Available in Mac OS X v10.5 and later.

CIExclusionBlendMode

Produces an effect similar to that produced by the CIDifferenceBlendMode filter but with lower contrast.

Parameters

`inputImage`
A `CIImage` class whose display name is Image.

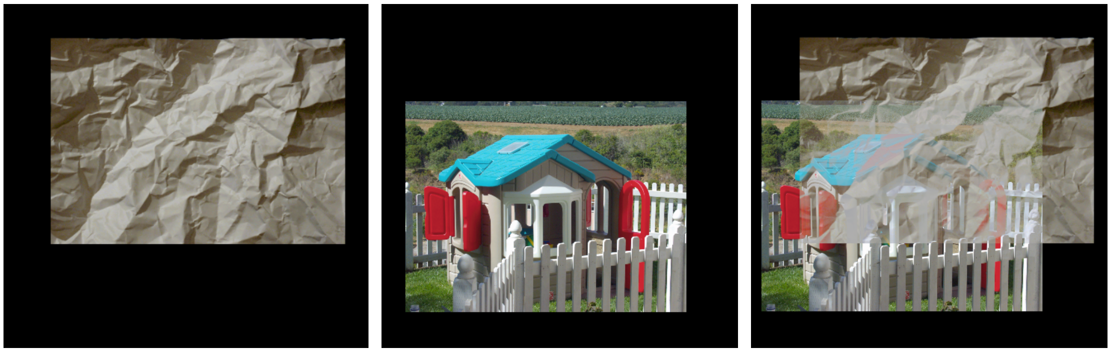
`inputBackgroundImage`
A `CIImage` class whose display name is Background Image.

Discussion
Source image sample values that are black do not produce a change; white inverts the background color values. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Exclusion Blend Mode

Figure 16-41 The result of using the CIExclusionBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIExposureAdjust

Adjusts the exposure setting for an image similar to the way you control exposure for a camera when you change the F-stop.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputEV

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is EV.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: -10.00 Slider maximum: 10.00 Identity: 0.00

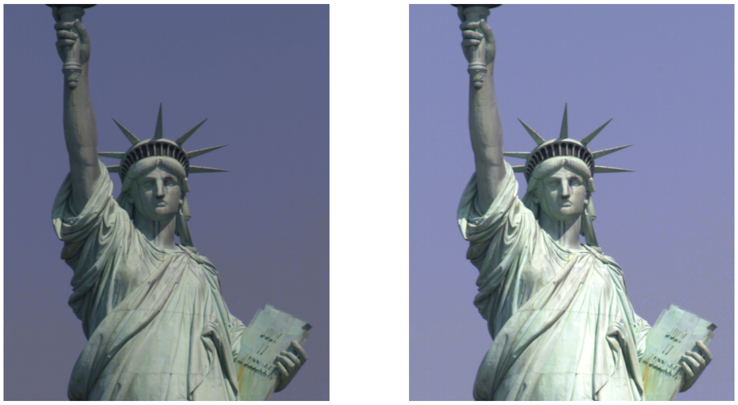
Discussion
This filter multiplies the color values, as follows, to simulate exposure change by the specified F-stops:

```
s.rgb * pow(2.0, ev)
```

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryColorAdjustment`

Localized Display Name
Exposure Adjust

Figure 16-42 The result of using the CIExposureAdjust filter



Availability
Available in Mac OS X v10.4 and later.

CIFalseColor

Maps luminance to a color ramp of two colors.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputColor0

A `CIColor` class whose display name is Color 1.

inputColor1

A `CIColor` class whose display name is Color 2.

Discussion

False color is often used to process astronomical and other scientific data, such as ultraviolet and x-ray images.

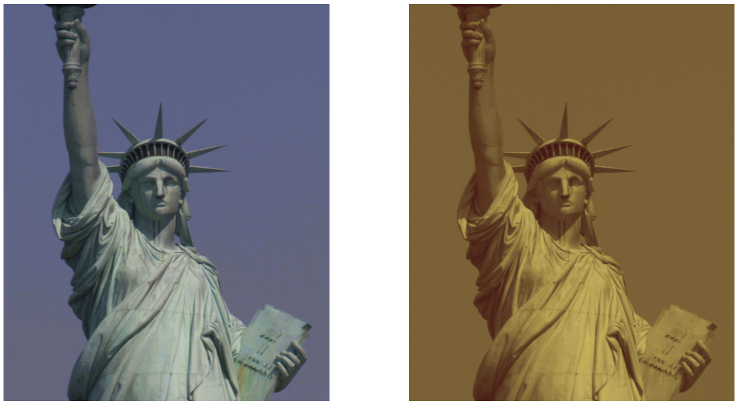
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryVideo`, `CICategoryColorEffect`

Localized Display Name

False Color

Figure 16-43 The result of using the CIFalseColor filter



Availability

Available in Mac OS X v10.4 and later.

CIFlashTransition

Transitions from one image to another by creating a flash.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputTargetImage

A `CIImage` class whose display name is Target Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputExtent

A `CIVector` class whose attribute type is `CIAttributeTypeRectangle` and whose display name is Extent.

Default value: [0 0 300 300] Identity: (null)

inputColor

A `CIColor` class whose attribute type is `CIAttributeTypeOpaqueColor` and whose display name is Color.

inputTime

An `NSNumber` class whose attribute type is `CIAttributeTypeTime` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputMaxStriationRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Maximum Striation Radius.

Default value: 2.58 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 10.00 Identity: 2.58

inputStriationStrength

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Striation Strength.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 3.00 Identity: 0.50

inputStriationContrast

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Striation Contrast.

Default value: 1.38 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 5.00 Identity: 1.38

inputFadeThreshold

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Fade Threshold.

Default value: 0.85 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.85

Discussion

The flash originates from a point you specify. Small at first, it rapidly expands until the image frame is completely filled with the flash color. As the color fades, the target image begins to appear.

Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name

Flash

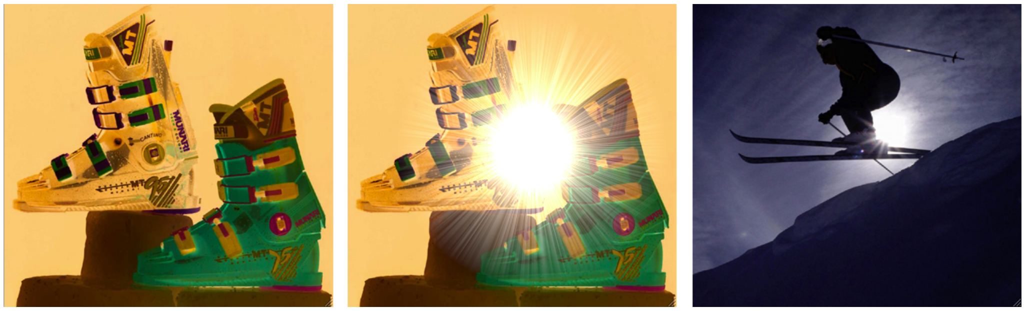


Figure 16-44 The result of using the CIFlashTransition filter

Availability
Available in Mac OS X v10.4 and later.

CIFourfoldReflectedTile

Produces a tiled image from a source image by applying a 4-way reflected symmetry.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputAcuteAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Acute Angle.

Default value: 1.57 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 1.57

inputWidth

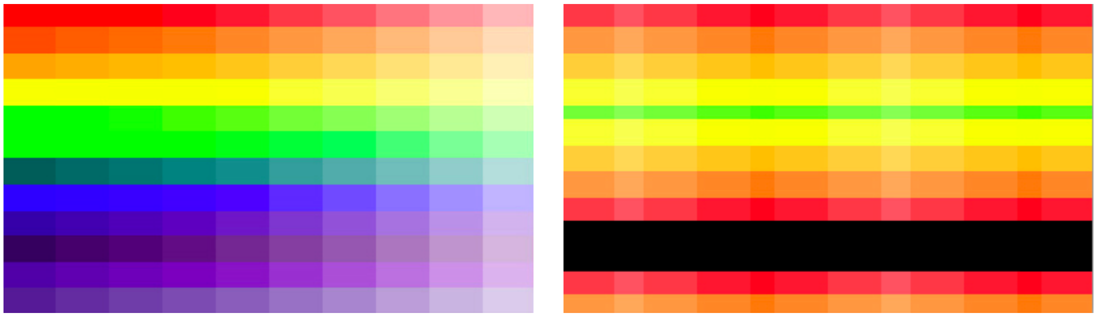
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
CIFourfoldReflectedTile

Figure 16-45 The result of using the CIFourfoldReflectedTile filter



Availability
Available in Mac OS X v10.5 and later.

CIFourfoldRotatedTile

Produces a tiled image from a source image by rotating the source image at increments of 90 degrees.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

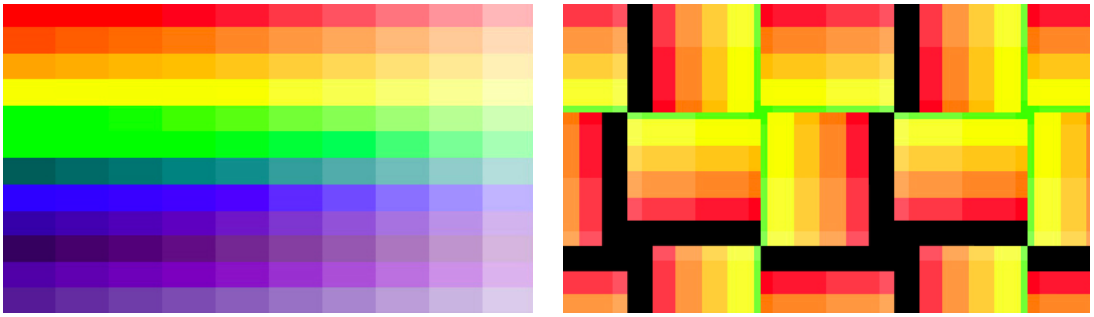
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
CIFourfoldRotatedTile

Figure 16-46 The result of using the CIFourfoldRotatedTile filter



Availability

Available in Mac OS X v10.5 and later.

CIFourfoldTranslatedTile

Produces a tiled image from a source image by applying 4 translation operations.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputAcuteAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Acute Angle.

Default value: 1.57 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 1.57

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

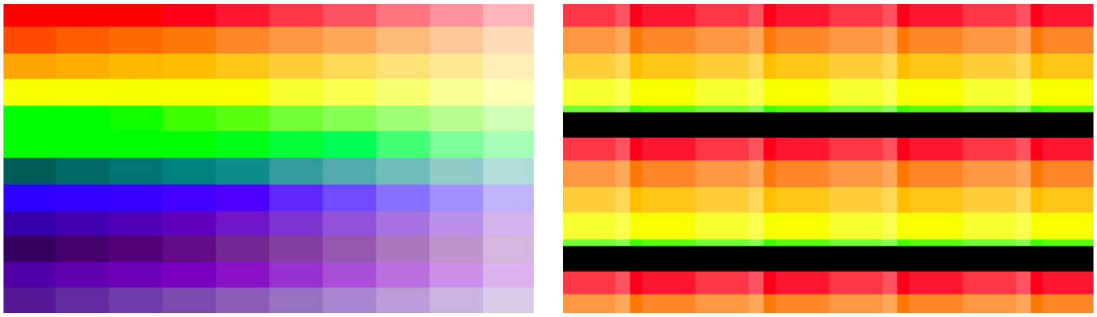
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryTileEffect

Localized Display Name

CIFourfoldTranslatedTile

Figure 16-47 The result of using the CIFourfoldTranslatedTile filter



Availability

Available in Mac OS X v10.5 and later.

CIGammaAdjust

Adjusts midtone brightness.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputPower

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Power.

Default value: 0.75 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.10 Slider maximum: 3.00 Identity: 1.00

Discussion

This filter is typically used to compensate for nonlinear effects of displays. Adjusting the gamma effectively changes the slope of the transition between black and white. It uses the following formula:

```
pow(s.rgb, vec3(power))
```

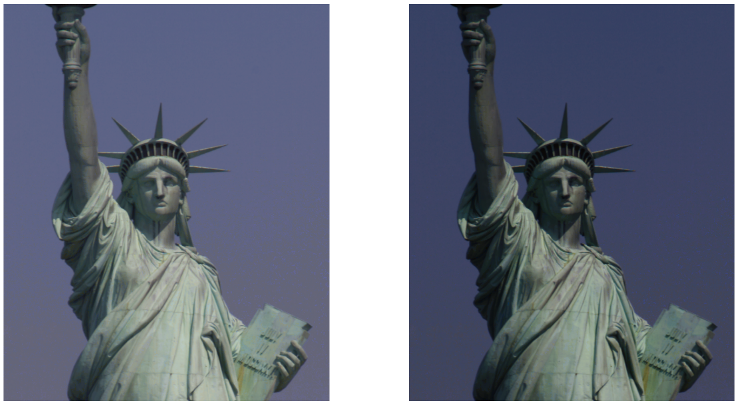
Member of

`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryColorAdjustment`

Localized Display Name

Gamma Adjust

Figure 16-48 The result of using the CIGammaAdjust filter



Availability
Available in Mac OS X v10.4 and later.

CIGaussianBlur

Spreads source pixels by an amount specified by a Gaussian distribution.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputRadius

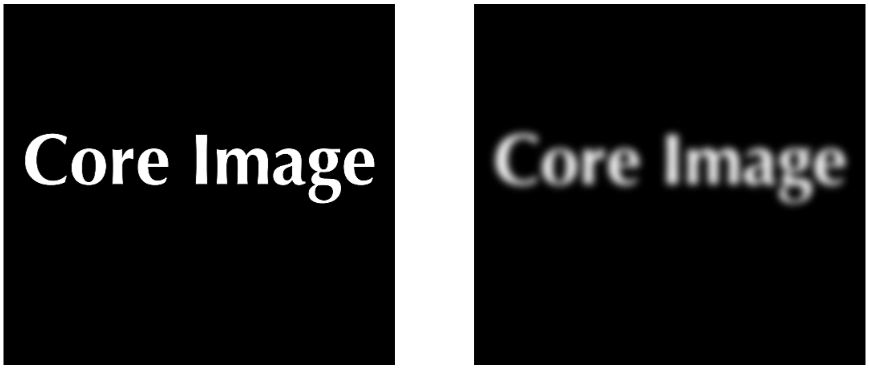
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name
Gaussian Blur

Figure 16-49 The result of using the CIGaussianBlur filter



Availability
Available in Mac OS X v10.4 and later.

CIGaussianGradient

Generates a gradient that varies from one color to another using a Gaussian distribution.

Parameters

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputColor0

A `CIColor` class whose display name is Color 1.

inputColor1

A `CIColor` class whose display name is Color 2.

inputRadius

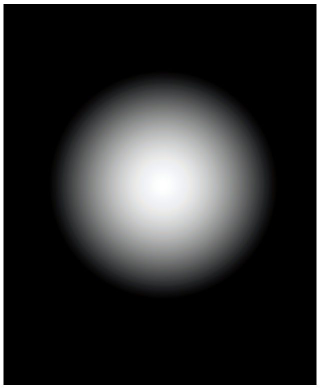
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 300.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 300.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGradient`

Localized Display Name
Gaussian Gradient

Figure 16-50 The result of using the CIGaussianGradient filter



Availability
Available in Mac OS X v10.4 and later.

CIGlassDistortion

Distorts an image by applying a glass-like texture.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputTexture`

A `CIImage` class whose display name is Texture.

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputScale`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Scale.

Default value: 200.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.01 Slider maximum: 500.00 Identity: 0.00

Discussion

The raised portions of the output image are the result of applying a texture map.

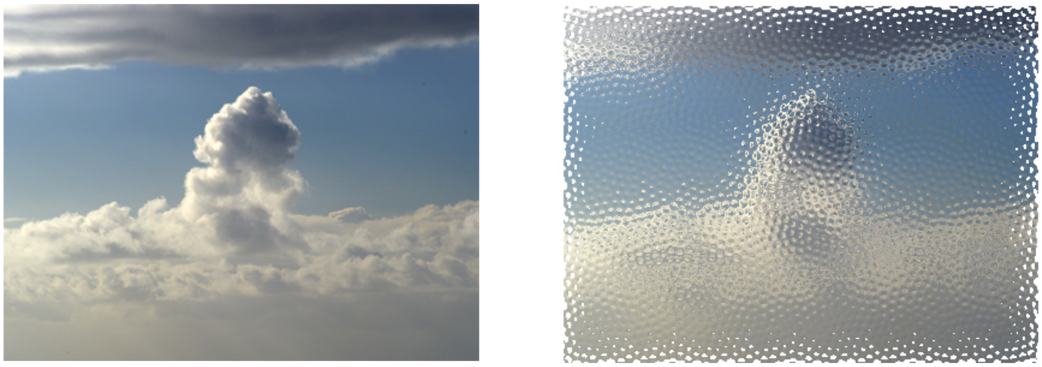
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name

Glass Distortion

Figure 16-51 The result of using the CIGlassDistortion filter



Availability

Available in Mac OS X v10.4 and later.

CIGlassLozenge

Creates a lozenge-shaped lens and distorts the portion of the image over which the lens is placed.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

inputPoint0

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Point 1.

Default value: [150 150] Identity: (null)

inputPoint1

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Point 2.

Default value: [350 150] Identity: (null)

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1000.00 Identity: 100.00

inputRefraction

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Refraction.

Default value: 1.70 Minimum: -5.00 Maximum: 0.00 Slider minimum: -5.00 Slider maximum: 5.00 Identity: 0.00

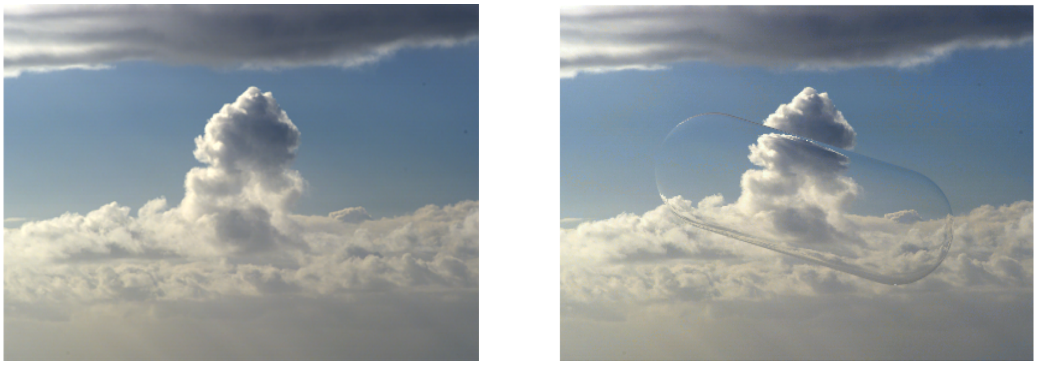
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name

Glass Lozenge

Figure 16-52 The result of using the CIGlassLozenge filter



Availability

Available in Mac OS X v10.4 and later.

CIGlideReflectedTile

Produces a tiled image from a source image by translating and smearing the image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputAngle`

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

`inputWidth`

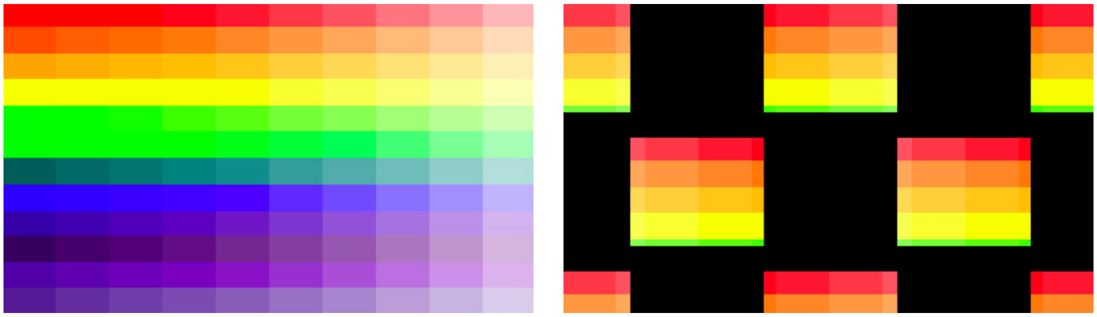
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
CIGlideReflectedTile

Figure 16-53 The result of using the CIGlideReflectedTile filter



Availability
Available in Mac OS X v10.5 and later.

CIGloom

Dulls the highlights of an image.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputRadius`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 0.00

inputIntensity

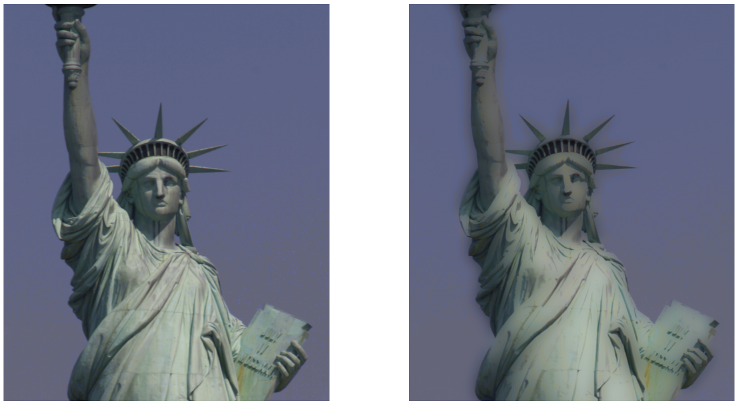
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Gloom

Figure 16-54 The result of using the CIGloom filter



Availability
Available in Mac OS X v10.4 and later.

CIHardLightBlendMode

Either multiplies or screens colors, depending on the source image sample color.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

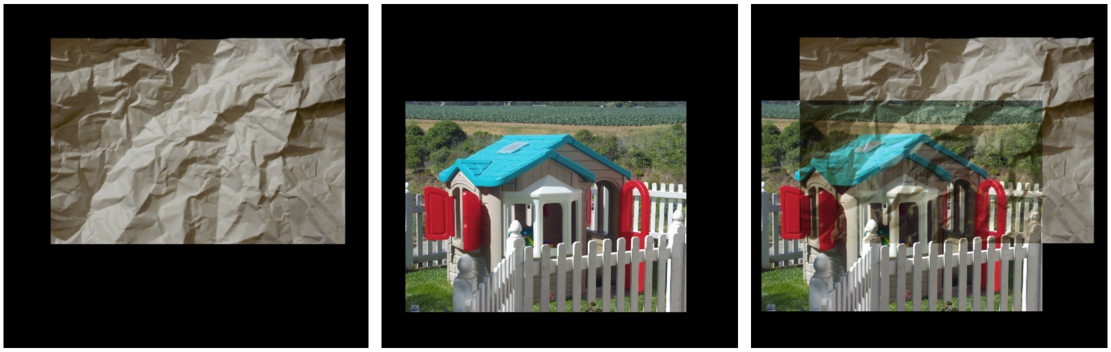
A `CIImage` class whose display name is Background Image.

Discussion
If the source image sample color is lighter than 50% gray, the background is lightened, similar to screening. If the source image sample color is darker than 50% gray, the background is darkened, similar to multiplying. If the source image sample color is equal to 50% gray, the source image is not changed. Image samples that are equal to pure black or pure white result in pure black or white. The overall effect is similar to what you would achieve by shining a harsh spotlight on the source image. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Hard Light Blend Mode

Figure 16-55 The result of using the CIHardLightBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIHatchedScreen

Simulates the hatched pattern of a halftone screen.

Parameters

inputImage
A `CIImage` class whose display name is Image.

inputCenter
A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle
An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 6.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 50.00 Identity: 0.00

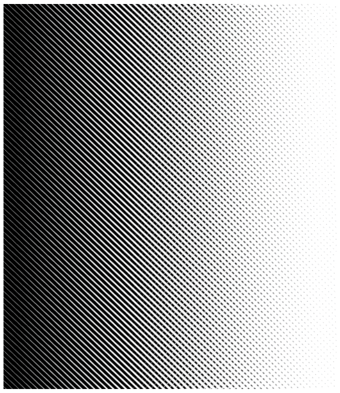
inputSharpness
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Sharpness.

Default value: 0.70 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryHalftoneEffect`

Localized Display Name
Hatched Screen

Figure 16-56 The result of using the CIHatchedScreen filter



Availability
Available in Mac OS X v10.4 and later.

CIHeightFieldFromMask

Produces a continuous three-dimensional, loft-shaped height field from a grayscale mask.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 300.00 Identity: 10.00

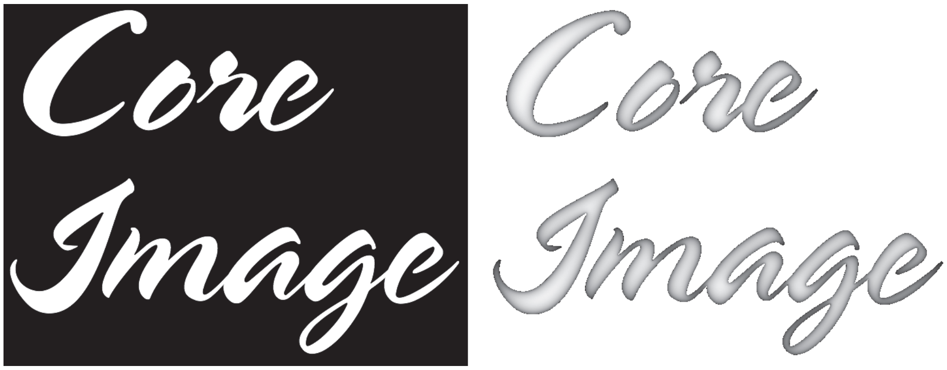
Discussion

The white values of the mask define those pixels that are inside the height field while the black values define those pixels that are outside. The field varies smoothly and continuously inside the mask, reaching the value 0 at the edge of the mask. You can use this filter with the CIShadedMaterial filter to produce extremely realistic shaded objects.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Height Field From Mask

Figure 16-57 The result of using the CIHeightFieldFromMask filter



Availability
Available in Mac OS X v10.4 and later.

CIHexagonalPixellate

Maps an image to colored hexagons whose color is defined by the replaced pixels.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputScale

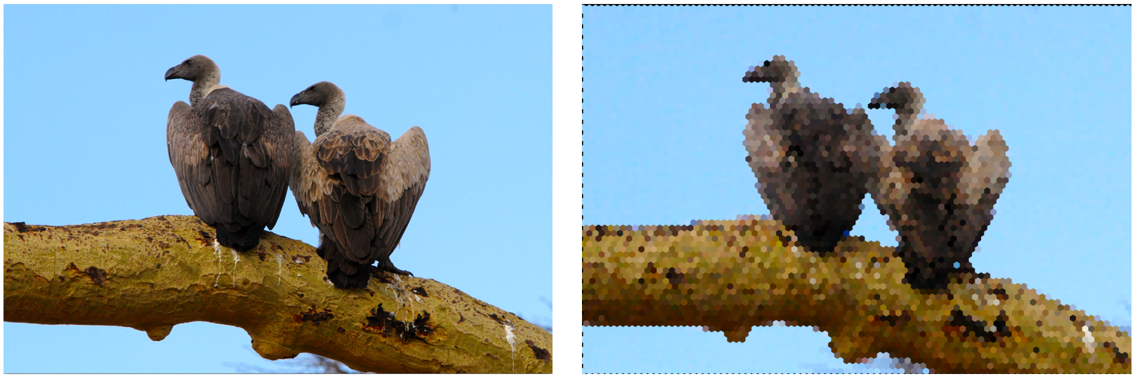
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Scale.

Default value: 8.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 100.00 Identity: 1.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
CIHexagonalPixellate

Figure 16-58 The result of using the CIHexagonalPixellate filter



Availability
Available in Mac OS X v10.5 and later.

CIHoleDistortion

Creates a circular area that pushes the image pixels outward, distorting those pixels closest to the circle the most.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

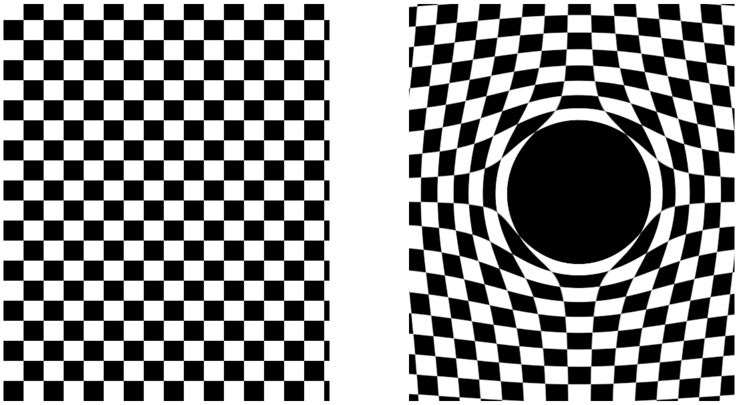
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 150.00 Minimum: 0.01 Maximum: 0.00 Slider minimum: 0.01 Slider maximum: 1000.00 Identity: 0.10

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name
Hole Distortion

Figure 16-59 The result of using the CIHoleDistortion filter



Availability

Available in Mac OS X v10.4 and later.

CIHueAdjust

Changes the overall hue, or tint, of the source pixels.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

Discussion

This filter essentially rotates the color cube around the neutral axis.

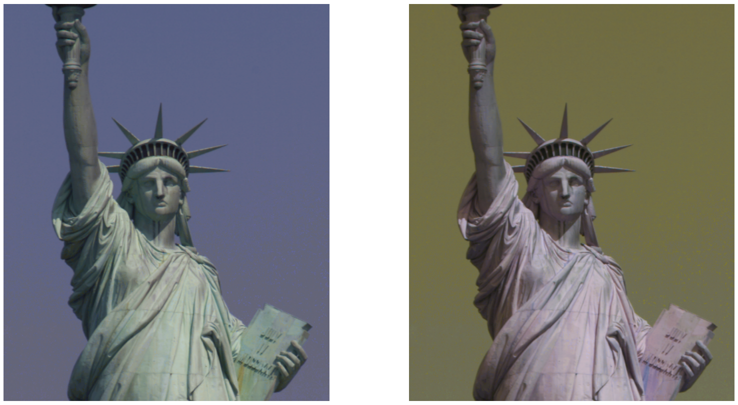
Member of

`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryColorAdjustment`

Localized Display Name

Hue Adjust

Figure 16-60 The result of using the CIHueAdjust filter



Availability
Available in Mac OS X v10.4 and later.

CIHueBlendMode

Uses the luminance and saturation values of the background with the hue of the source image.

Parameters

inputImage
A `CIImage` class whose display name is Image.

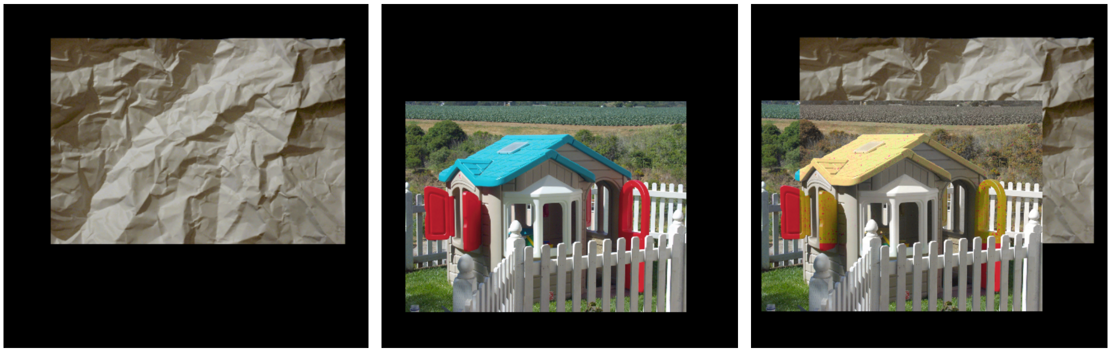
inputBackgroundImage
A `CIImage` class whose display name is Background Image.

Discussion
The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Hue Blend Mode

Figure 16-61 The result of using the CIHueBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIKaleidoscope

Produces a kaleidoscopic image from a source image by applying 12-way symmetry.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCount

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Count.

Default value: 6.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 64.00 Identity: 0.00

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

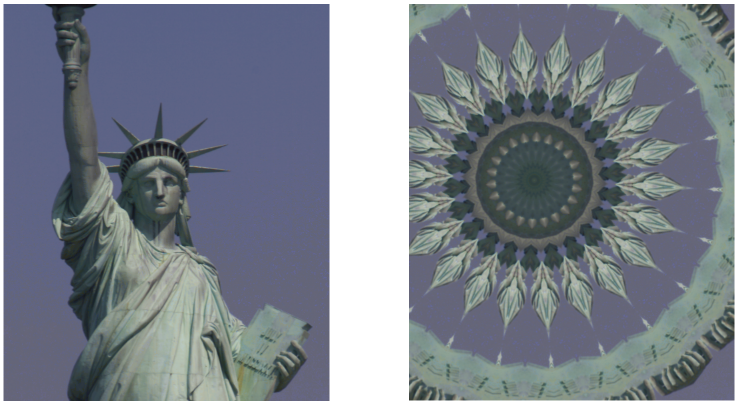
An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
Kaleidoscope

Figure 16-62 The result of using the CIKaleidoscope filter



Availability
Available in Mac OS X v10.4 and later.

CILanczosScaleTransform

Produces a high-quality, scaled version of a source image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Scale.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.05 Slider maximum: 1.50 Identity: 1.00

inputAspectRatio

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Aspect Ratio.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.50 Slider maximum: 2.00 Identity: 1.00

Discussion

You typically use this filter to scale down an image.

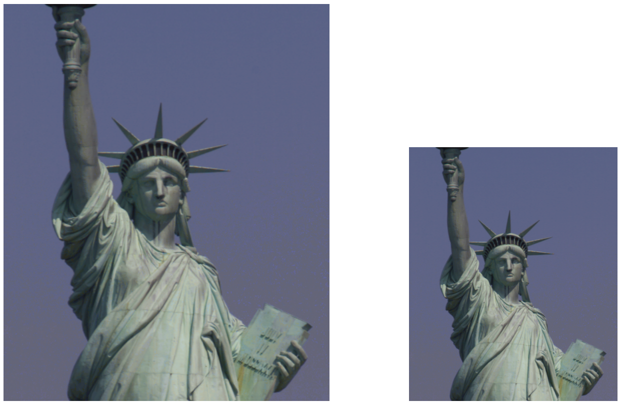
Member of

`CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryGeometryAdjustment`

Localized Display Name

Lanczos Scale Transform

Figure 16-63 The result of using the CILanczosScaleTransform filter



Availability

Available in Mac OS X v10.4 and later.

CILenticularHaloGenerator

Simulates a halo that is generated by the diffraction associated with the spread of a lens.

Parameters

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputColor

A `CIColor` class whose display name is Color.

inputHaloRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Halo Radius.

Default value: 70.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1000.00 Identity: 0.00

inputHaloWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Halo Width.

Default value: 87.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 300.00 Identity: 0.00

inputHaloOverlap

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Halo Overlap.

Default value: 0.77 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputStriationStrength

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Striation Strength.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 3.00 Identity: 0.00

inputStriationContrast

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Striation Contrast.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 5.00 Identity: 0.00

inputTime

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Discussion

This filter is typically applied to another image to simulate lens flares and similar effects.

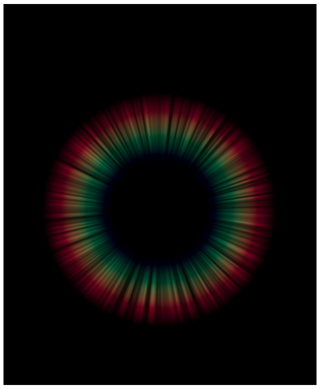
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGenerator`

Localized Display Name

Lenticular Halo

Figure 16-64 The result of using the CILenticularHaloGenerator filter



Availability
Available in Mac OS X v10.4 and later.

CILightenBlendMode

Creates composite image samples by choosing the lighter samples (either from the source image or the background).

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputBackgroundImage`

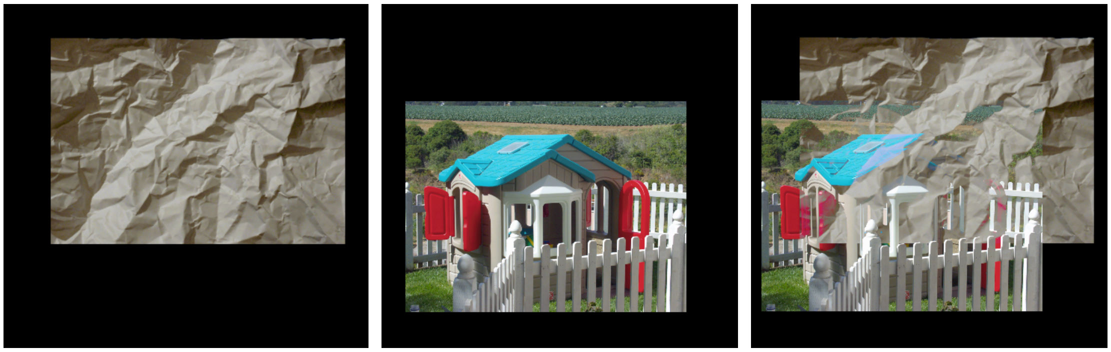
A `CIImage` class whose display name is Background Image.

Discussion
The result is that the background image samples are replaced by any source image samples that are lighter. Otherwise, the background image samples are left unchanged. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Lighten Blend Mode

Figure 16-65 The result of using the CILightenBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CILinearGradient

Generates a gradient that varies along a linear axis between two defined endpoints.

Parameters

`inputPoint0`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Point 1.

Default value: [0 0] Identity: (null)

`inputPoint1`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Point 2.

Default value: [200 200] Identity: (null)

`inputColor0`

A `CIColor` class whose display name is Color 1.

`inputColor1`

A `CIColor` class whose display name is Color 2.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGradient`

Localized Display Name
Linear Gradient

Figure 16-66 The result of using the CILinearGradient filter

Availability
Available in Mac OS X v10.4 and later.

CILineOverlay

Creates a sketch that outlines the edges of an image in black.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputNRNoiseLevel

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is NR Noise Level.

Default value: 0.07 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 0.10 Identity: 0.00

inputNRSharpness

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is NR Sharpness.

Default value: 0.71 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 2.00 Identity: 0.00

inputEdgeIntensity

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Edge Intensity.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 200.00 Identity: 0.00

inputThreshold

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Threshold.

Default value: 0.10 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputContrast

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Contrast.

Default value: 50.00 Minimum: 0.25 Maximum: 0.00 Slider minimum: 0.25 Slider maximum: 200.00 Identity: 1.00

Discussion

The portions of the image that are not outlined are transparent.

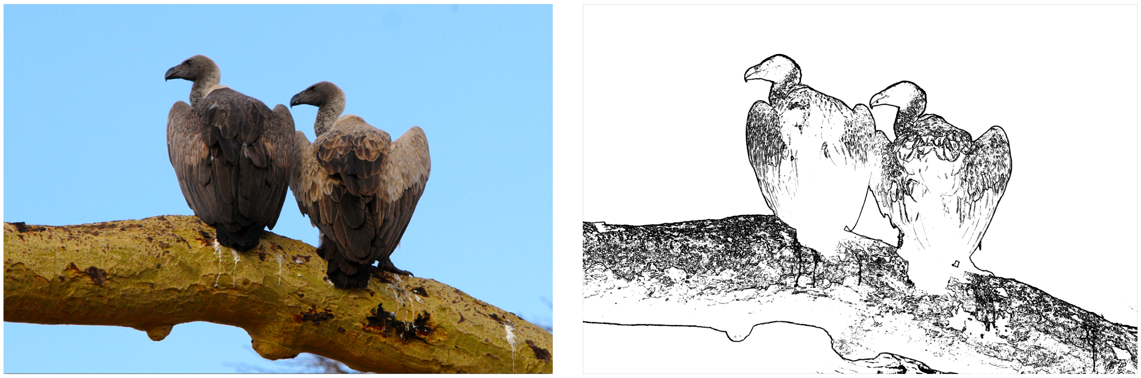
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name

Line Overlay

Figure 16-67 The result of using the CILineOverlay filter



Availability

Available in Mac OS X v10.5 and later.

CILineScreen

Simulates the line pattern of a halftone screen.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 6.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 2.00 Slider maximum: 50.00 Identity: 0.00

inputSharpness

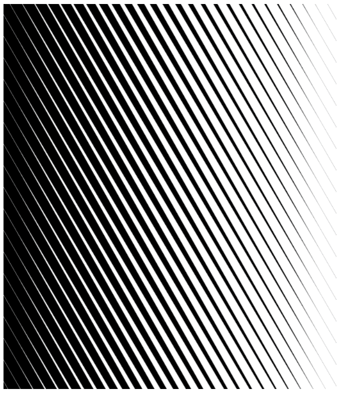
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Sharpness.

Default value: 0.70 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryHalftoneEffect`

Localized Display Name
Line Screen

Figure 16-68 The result of using the CILineScreen filter



Availability
Available in Mac OS X v10.4 and later.

CILuminosityBlendMode

Uses the hue and saturation of the background with the luminance of the source image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

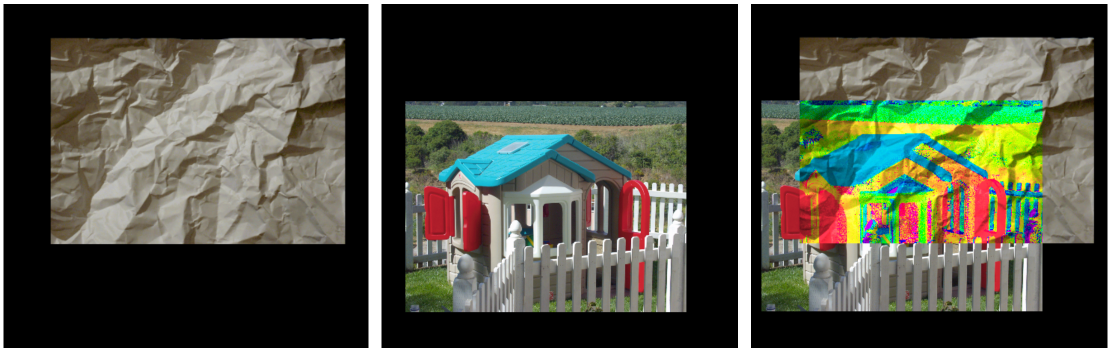
A `CIImage` class whose display name is Background Image.

Discussion
This mode creates an effect that is inverse to the effect created by the CIColorBlendMode filter. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
CICategoryBuiltIn, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryStillImage, CICategoryVideo, CICategoryCompositeOperation

Localized Display Name
Luminosity Blend Mode

Figure 16-69 The result of using the CILuminosityBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIMaskToAlpha

Converts a grayscale image to a white image that is masked by alpha.

Parameters

inputImage

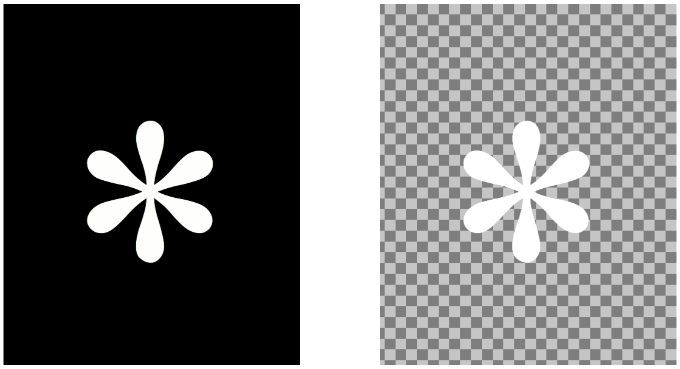
A CIImage class whose display name is Image.

Discussion
The white values from the source image produce the inside of the mask; the black values become completely transparent.

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryVideo, CICategoryColorEffect

Localized Display Name
Mask To Alpha

Figure 16-70 The result of using the CIMaskToAlpha filter



Availability
Available in Mac OS X v10.4 and later.

CIMaximumComponent

Returns a grayscale image from max(r,g,b).

Parameters

inputImage

A `CIImage` class whose display name is Image. This is the image data you want to process.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryVideo`, `CICategoryColorEffect`

Localized Display Name
Maximum Component

Availability
Available in Mac OS X v10.5 and later.

CIMaximumCompositing

Computes the maximum value, by color component, of two input images and creates an output image using the maximum values.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

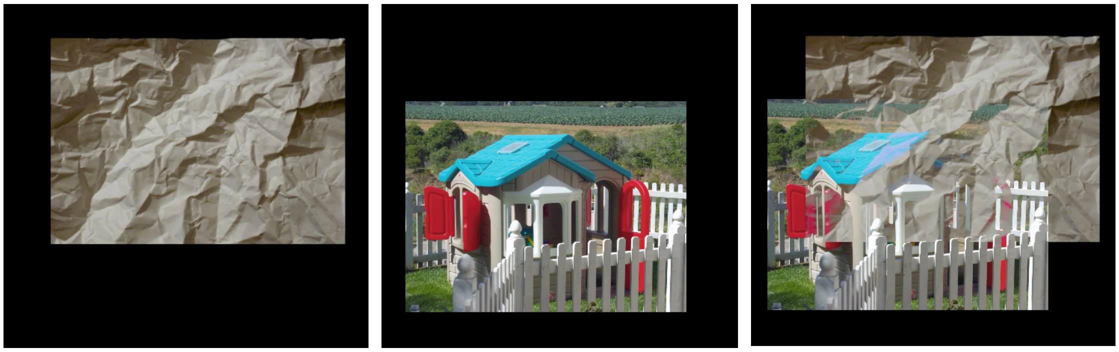
A `CIImage` class whose display name is Background Image.

Discussion
This is similar to dodging. The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

Member of
`CICategoryBuiltIn`, `CICategoryHighDynamicRange`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Maximum

Figure 16-71 The result of using the CIMaximumCompositing filter



Availability
Available in Mac OS X v10.4 and later.

CIMedianFilter

Computes the median value for a group of neighboring pixels and replaces each pixel value with the median.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

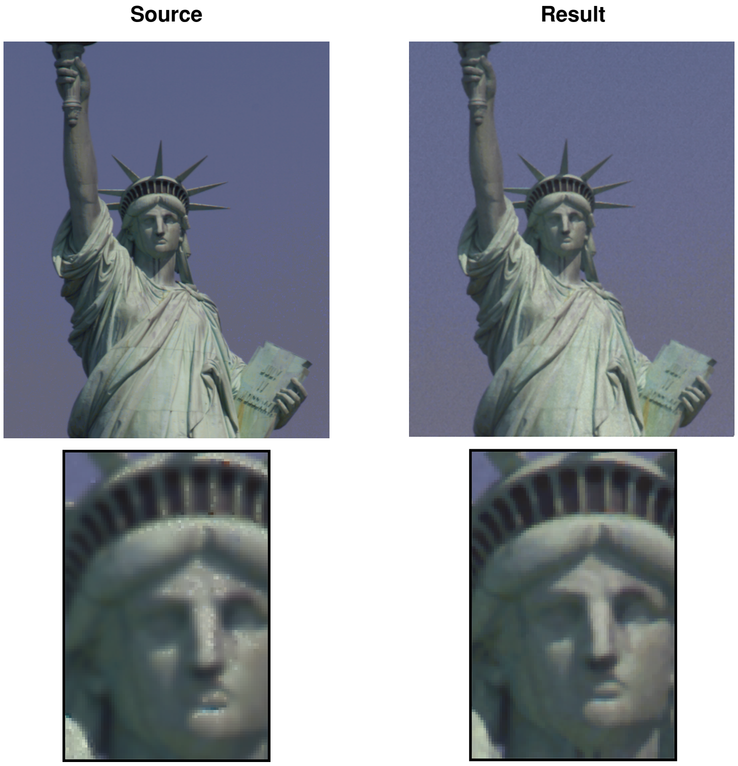
Discussion

The effect is to reduce noise.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name
Median

Figure 16-72 The result of using the CIMedianFilter filter



Availability
Available in Mac OS X v10.4 and later.

CIMinimumComponent

Returns a grayscale image from min(r,g,b).

Parameters

inputImage

A `CIImage` class whose display name is Image. This is the image data you want to process.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryVideo`, `CICategoryColorEffect`

Localized Display Name
Minimum Component

Availability
Available in Mac OS X v10.5 and later.

CIMinimumCompositing

Computes the minimum value, by color component, of two input images and creates an output image using the minimum values.

Parameters

inputImage

A CIImage class whose display name is Image.

inputBackgroundImage

A CIImage class whose display name is Background Image.

Discussion

This is similar to burning. The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

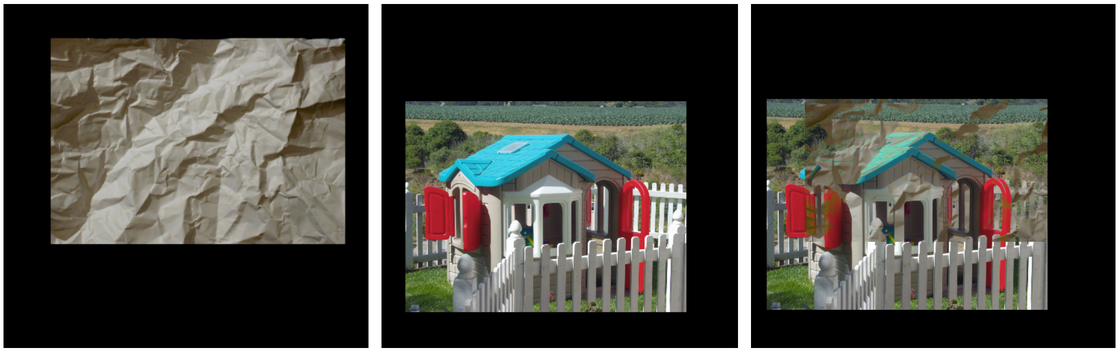
Member of

CICategoryBuiltIn, CICategoryHighDynamicRange, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryStillImage, CICategoryVideo, CICategoryCompositeOperation

Localized Display Name

Minimum

Figure 16-73 The result of using the CIMinimumCompositing filter



Availability

Available in Mac OS X v10.4 and later.

CIModTransition

Transitions from one image to another by revealing the target image through irregularly shaped holes.

Parameters

inputImage

A CIImage class whose display name is Image.

inputTargetImage

A CIImage class whose display name is Target Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputTime

An `NSNumber` class whose attribute type is `CIAttributeTypeTime` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 2.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -6.28 Slider maximum: 6.28 Identity: 2.00

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 150.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 0.00

inputCompression

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Compression.

Default value: 300.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 100.00 Slider maximum: 800.00 Identity: 0.00

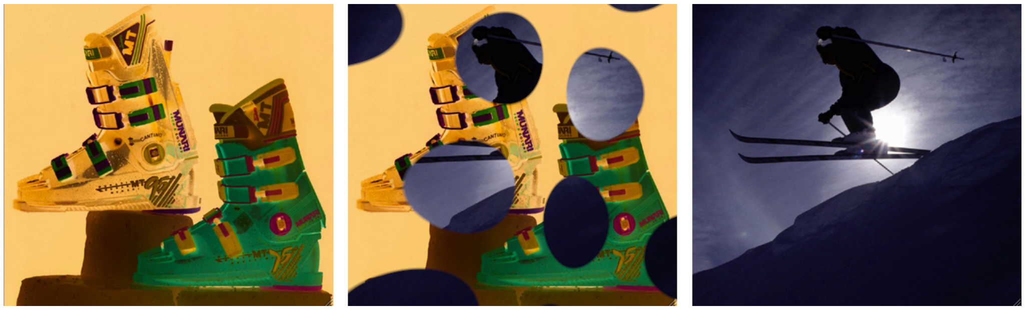
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name

Mod

Figure 16-74 The result of using the CIModTransition filter



Availability

Available in Mac OS X v10.4 and later.

CIMotionBlur

Blurs an image to simulate the effect of using a camera that moves a specified angle and distance while capturing the image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 20.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 0.00

inputAngle

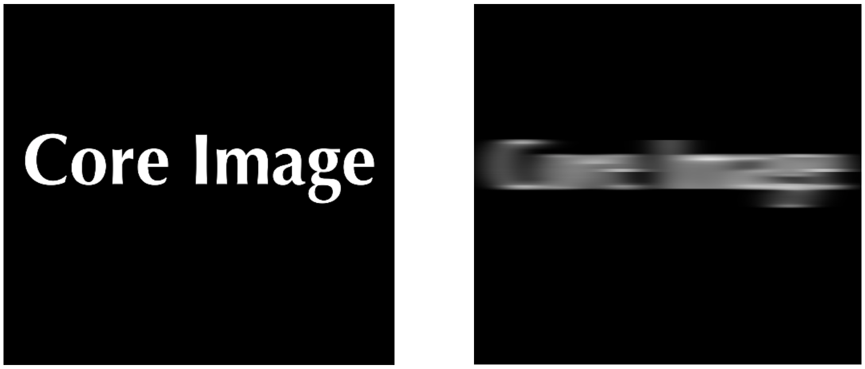
An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name
Motion Blur

Figure 16-75 The result of using the CIMotionBlur filter



Availability
Available in Mac OS X v10.4 and later.

CIMultiplyBlendMode

Multiplies the source image samples with the background image samples.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

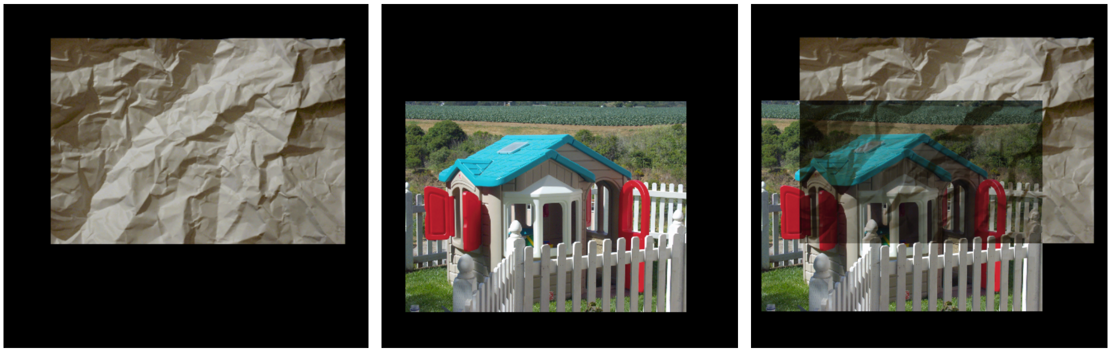
A `CIImage` class whose display name is Background Image.

Discussion
This results in colors that are at least as dark as either of the two contributing sample colors. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Multiply Blend Mode

Figure 16-76 The result of using the CIMultiplyBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIMultiplyCompositing

Multiplies the color component of two input images and creates an output image using the multiplied values.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputBackgroundImage`

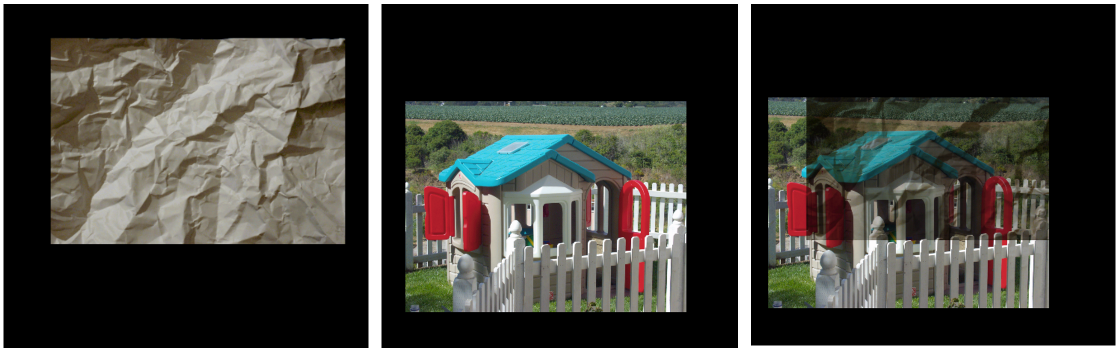
A `CIImage` class whose display name is Background Image.

Discussion
This filter is typically used to add a spotlight or similar lighting effect to an image. The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

Member of
`CICategoryBuiltIn`, `CICategoryHighDynamicRange`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Multiply

Figure 16-77 The result of using the CIMultiplyCompositing filter



Availability
Available in Mac OS X v10.4 and later.

CINoiseReduction

Reduces noise using a threshold value to define what is considered noise.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputNoiseLevel

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Noise Level.

Default value: 0.02 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 0.10 Identity: 0.00

inputSharpness

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Sharpness.

Default value: 0.40 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 2.00 Identity: 0.00

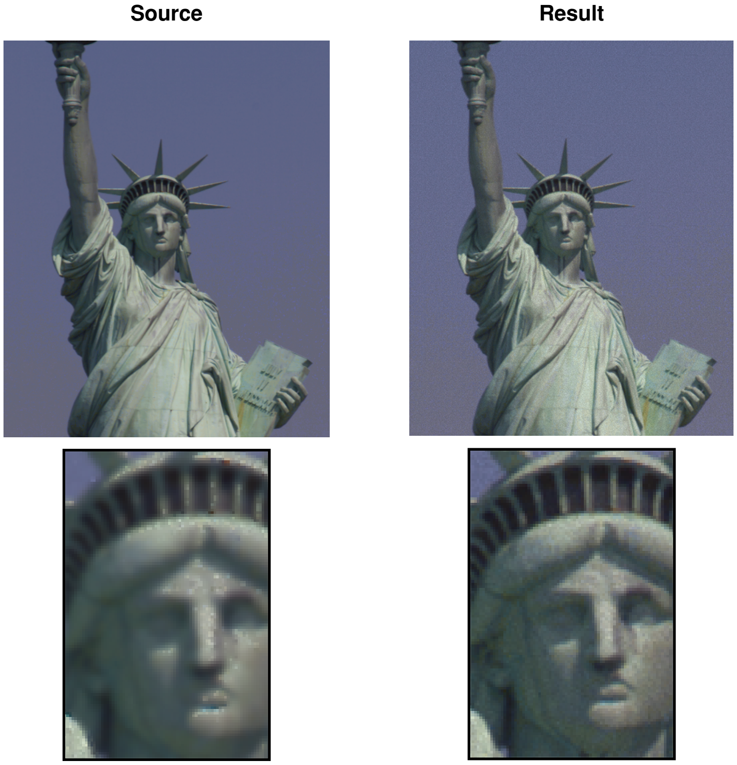
Discussion

Small changes in luminance below that value are considered noise and get a noise reduction treatment, which is a local blur. Changes above the threshold value are considered edges, so they are sharpened.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name
Noise Reduction

Figure 16-78 The result of using the CINoiseReduction filter



Availability
Available in Mac OS X v10.4 and later.

CIOpTile

Segments an image, applying any specified scaling and rotation, and then assembles the image again to give an op art appearance.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Scale.

Default value: 2.80 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.10 Slider maximum: 10.00 Identity: 1.00

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

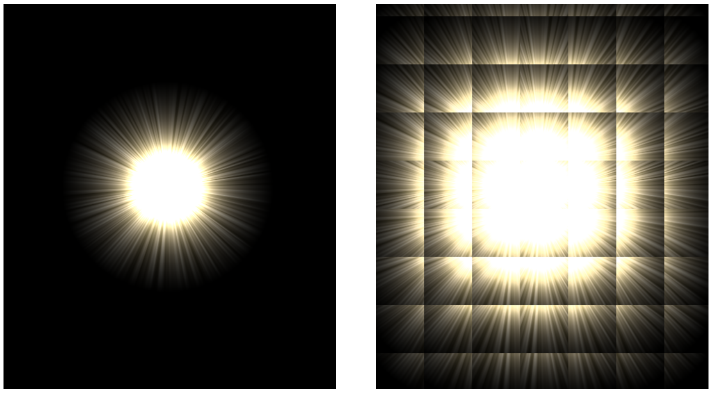
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 65.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 1000.00 Identity: 65.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
Op Tile

Figure 16-79 The result of using the CIOpTile filter



Availability
Available in Mac OS X v10.4 and later.

CIOverlayBlendMode

Either multiplies or screens the source image samples with the background image samples, depending on the background color.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

A `CIImage` class whose display name is Background Image.

Discussion
The result is to overlay the existing image samples while preserving the highlights and shadows of the background. The background color mixes with the source image to reflect the lightness or darkness of the background. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

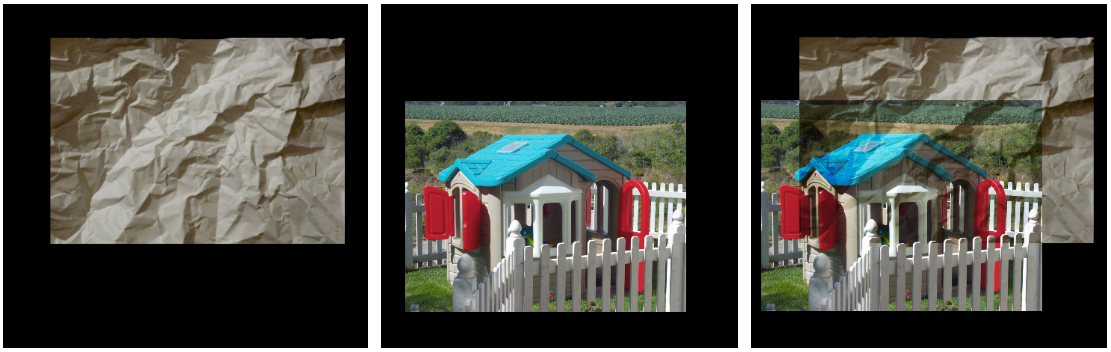
Member of

`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name

Overlay Blend Mode

Figure 16-80 The result of using the CIOverlayBlendMode filter



Availability

Available in Mac OS X v10.4 and later.

CIPageCurlTransition

Transitions from one image to another by simulating a curling page, revealing the new image as the page curls.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputTargetImage`

A `CIImage` class whose display name is Target Image.

`inputBacksideImage`

A `CIImage` class whose display name is Backside Image.

`inputShadingImage`

A `CIImage` class whose display name is Shading Image.

`inputExtent`

A `CIVector` class whose attribute type is `CIAttributeTypeRectangle` and whose display name is Extent.

Default value: [0 0 300 300] Identity: (null)

`inputTime`

An `NSNumber` class whose attribute type is `CIAttributeTypeTime` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 100.00 Minimum: 0.01 Maximum: 0.00 Slider minimum: 0.01 Slider maximum: 200.00 Identity: 0.00

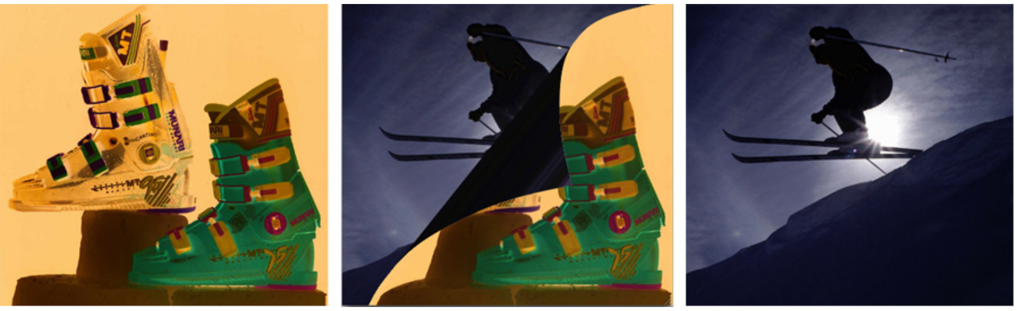
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name

Page Curl

Figure 16-81 The result of using the CIPageCurlTransition filter



Availability

Available in Mac OS X v10.4 and later.

CIParallelogramTile

Warps an image by reflecting it in a parallelogram, and then tiles the result.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputAcuteAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Acute Angle.

Default value: 1.57 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 1.57

inputWidth

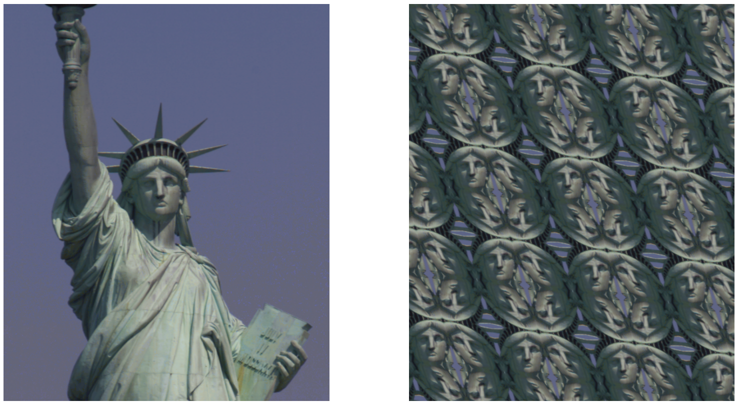
An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryTileEffect

Localized Display Name
Parallelogram Tile

Figure 16-82 The result of using the CIParallelogramTile filter



Availability
Available in Mac OS X v10.4 and later.

CIPerspectiveTile

Applies a perspective transform to an image and then tiles the result.

Parameters

inputImage

A CIImage class whose display name is Image.

inputTopLeft

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Top Left.

Default value: [118 484] Identity: (null)

inputTopRight

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Top Right.

Default value: [646 507] Identity: (null)

inputBottomRight

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Bottom Right.

Default value: [548 140] Identity: (null)

inputBottomLeft

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Bottom Left.

Default value: [155 153] Identity: (null)

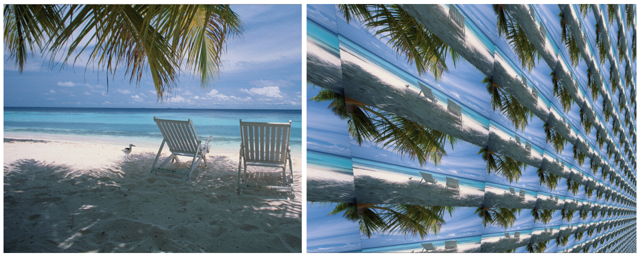
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name

Perspective Tile

Figure 16-83 The result of using the CIPerspectiveTile filter



Availability

Available in Mac OS X v10.4 and later.

CIPerspectiveTransform

Alters the geometry of an image to simulate the observer changing viewing position.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputTopLeft

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Top Left.

Default value: [118 484] Identity: (null)

inputTopRight

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Top Right.

Default value: [646 507] Identity: (null)

inputBottomRight

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Bottom Right.

Default value: [548 140] Identity: (null)

inputBottomLeft

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Bottom Left.

Default value: [155 153] Identity: (null)

Discussion

You can use the perspective filter to skew an image.

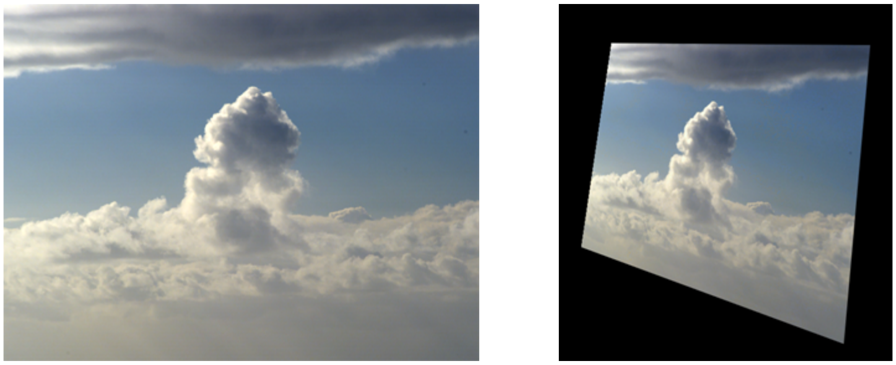
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGeometryAdjustment`

Localized Display Name

Perspective Transform

Figure 16-84 The result of using the CIPerspectiveTransform filter



Availability

Available in Mac OS X v10.4 and later.

CIPinchDistortion

Creates a rectangular-shaped area that pinches source pixels inward, distorting those pixels closest to the rectangle the most.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 300.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1000.00 Identity: 0.00

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Scale.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

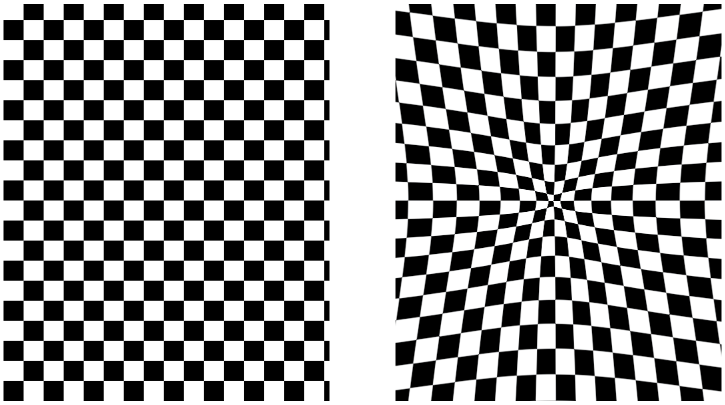
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name

Pinch Distortion

Figure 16-85 The result of using the CIPinchDistortion filter



Availability

Available in Mac OS X v10.4 and later.

CIPixellate

Makes an image blocky by mapping the image to colored squares whose color is defined by the replaced pixels.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputScale

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Scale.

Default value: 8.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 100.00 Identity: 1.00

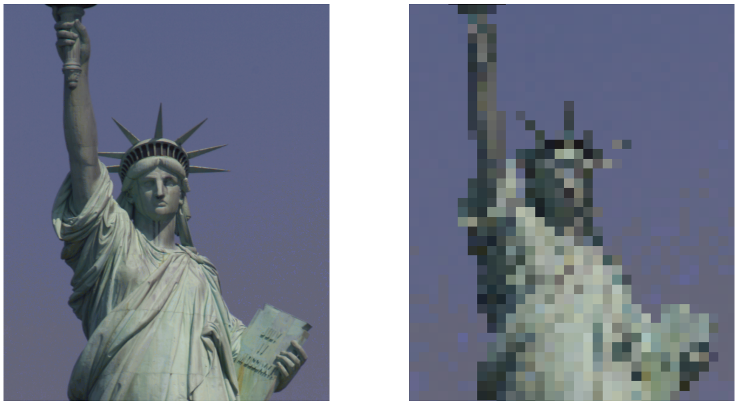
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryStylize

Localized Display Name

Pixellate

Figure 16-86 The result of using the CIPixellate filter



Availability

Available in Mac OS X v10.4 and later.

CIPointillize

Renders the source image in a pointillistic style.

Parameters

inputImage

A CIImage class whose display name is Image.

inputRadius

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Radius.

Default value: 20.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 100.00 Identity: 1.00

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

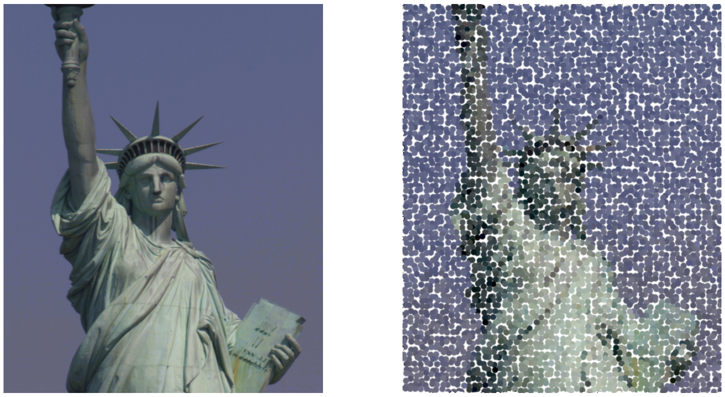
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryStylize

Localized Display Name

Pointillize

Figure 16-87 The result of using the CIPointillize filter



Availability
Available in Mac OS X v10.4 and later.

CIRadialGradient

Generates a gradient that varies radially between two circles having the same center.

Parameters

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputRadius0`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius 1.

Default value: 5.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 5.00

`inputRadius1`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius 2.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 100.00

`inputColor0`

A `CIColor` class whose display name is Color 1.

`inputColor1`

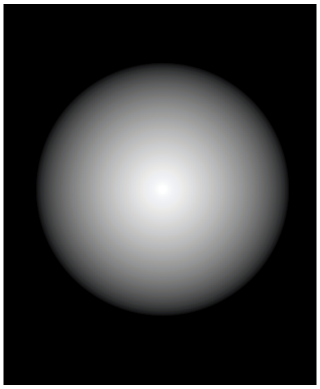
A `CIColor` class whose display name is Color 2.

Discussion
It is valid for one of the two circles to have a radius of 0.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGradient`

Localized Display Name
Radial Gradient

Figure 16-88 The result of using the CIRadialGradient filter



Availability
Available in Mac OS X v10.4 and later.

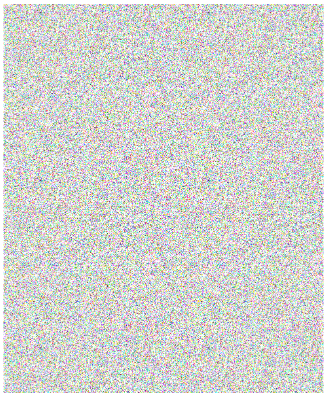
CIRandomGenerator

Generates an image of infinite extent whose pixel values are made up of four independent, uniformly-distributed random numbers in the 0 to 1 range.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryGenerator`

Localized Display Name
Random Generator

Figure 16-89 The result of using the CIRandomGenerator filter



Availability
Available in Mac OS X v10.4 and later.

CIRippleTransition

Transitions from one image to another by creating a circular wave that expands from the center point, revealing the new image in the wake of the wave.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputTargetImage

A `CIImage` class whose display name is Target Image.

inputShadingImage

A `CIImage` class whose display name is Shading Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputExtent

A `CIVector` class whose attribute type is `CIAttributeTypeRectangle` and whose display name is Extent.

Default value: [0 0 300 300] Identity: (null)

inputTime

An `NSNumber` class whose attribute type is `CIAttributeTypeTime` and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 1.00 Maximum: 0.00 Slider minimum: 10.00 Slider maximum: 300.00 Identity: 0.00

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Scale.

Default value: 50.00 Minimum: -50.00 Maximum: 0.00 Slider minimum: -50.00 Slider maximum: 50.00 Identity: 0.00

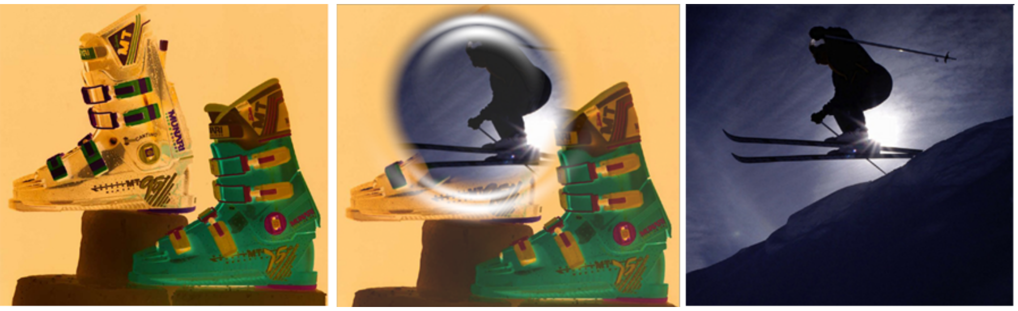
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name

Ripple

Figure 16-90 The result of using the CIRippleTransition filter



Availability
Available in Mac OS X v10.4 and later.

CIRowAverage

Returns a 1-pixel high image that contains the average color for each scan row.

Parameters

`inputImage`

A `CIImage` class whose display name is Image. This is the image data you want to process.

`inputExtent`

The rectangular region of interest.

Member of
`CICategoryReduction`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBuiltIn`

Localized Display Name
Row Average

Availability
Available in Mac OS X v10.5 and later.

CISaturationBlendMode

Uses the luminance and hue values of the background with the saturation of the source image.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputBackgroundImage`

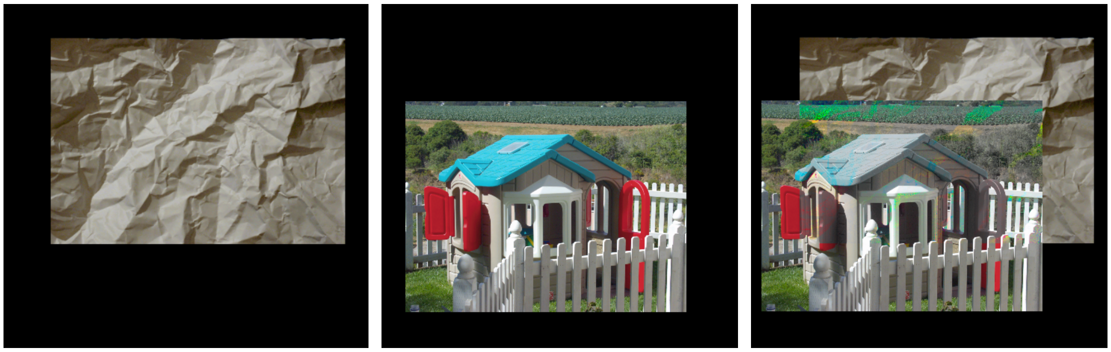
A `CIImage` class whose display name is Background Image.

Discussion
Areas of the background that have no saturation (that is, pure gray areas) do not produce a change. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Saturation Blend Mode

Figure 16-91 The result of using the CISaturationBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CIScreenBlendMode

Multiplies the inverse of the source image samples with the inverse of the background image samples.

Parameters

`inputImage`
A `CIImage` class whose display name is Image.

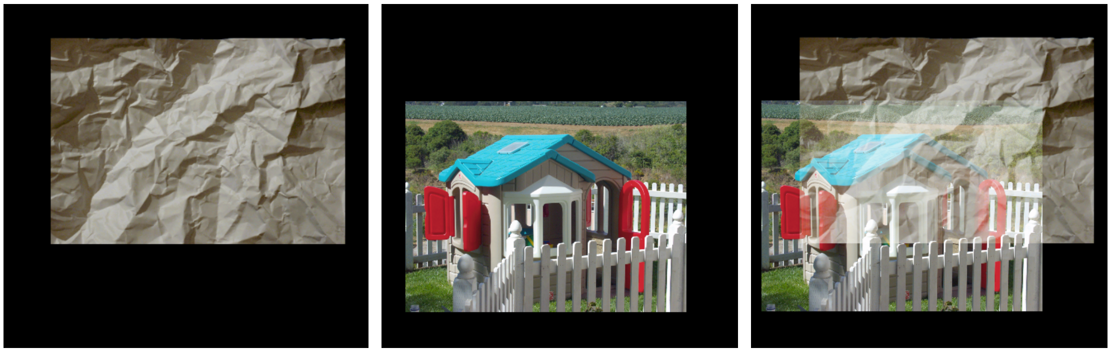
`inputBackgroundImage`
A `CIImage` class whose display name is Background Image.

Discussion
This results in colors that are at least as light as either of the two contributing sample colors. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Screen Blend Mode

Figure 16-92 The result of using the CIScreenBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CISepiaTone

Maps the colors of an image to various shades of brown.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputIntensity

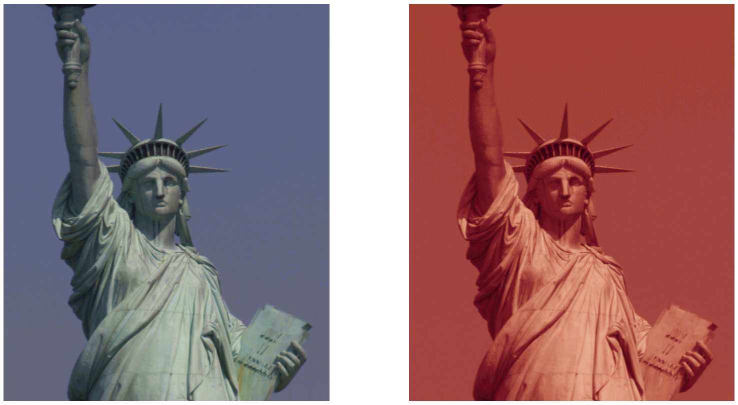
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryVideo`, `CICategoryColorEffect`

Localized Display Name
Sepia Tone

Figure 16-93 The result of using the CISepiaTone filter



Availability
Available in Mac OS X v10.4 and later.

CIShadedMaterial

Produces a shaded image from a height field.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputShadingImage

A `CIImage` class whose display name is Shading Image.

inputScale

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Scale.

Default value: 10.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.50 Slider maximum: 200.00 Identity: 0.00

Discussion
The height field is defined to have greater heights with lighter shades, and lesser heights (lower areas) with darker shades. You can combine this filter with the CIHeightFieldFromMask filter to produce quick shadings of masks, such as text.

This filter sets the input image as a height-field (multiplied by the scale parameter), and computes a normal vector for each pixel. It then uses that normal vector to look up the reflected color for that direction in the input shading image.

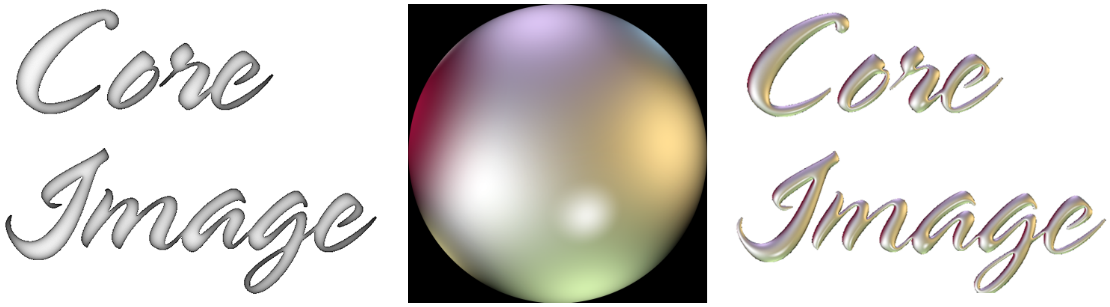
The input shading image contains the picture of a hemisphere, which defines the way the surface is shaded. The look-up coordinate for a normal vector is:

```
(normal.xy + 1.0) * 0.5 * vec2(shadingImageWidth, shadingImageHeight)
```

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name
Shaded Material

Figure 16-94 The result of using the CIShadedMaterial filter



Availability
Available in Mac OS X v10.4 and later.

CISharpenLuminance

Increases image detail by sharpening.

Parameters

inputImage

A CIImage class whose display name is Image.

inputSharpness

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Sharpness.

Default value: 0.40 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 2.00 Identity: 0.00

Discussion

It operates on the luminance of the image; the chrominance of the pixels remains unaffected.

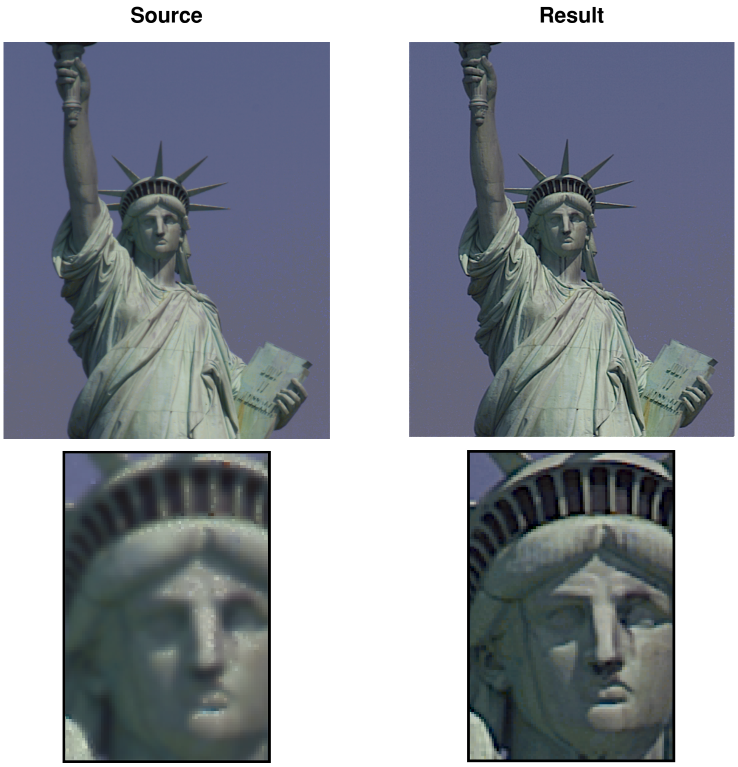
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategorySharpen

Localized Display Name

Sharpen Luminance

Figure 16-95 The result of using the CISharpenLuminance filter



Availability

Available in Mac OS X v10.4 and later.

CISixfoldReflectedTile

Produces a tiled image from a source image by applying a 6-way reflected symmetry.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

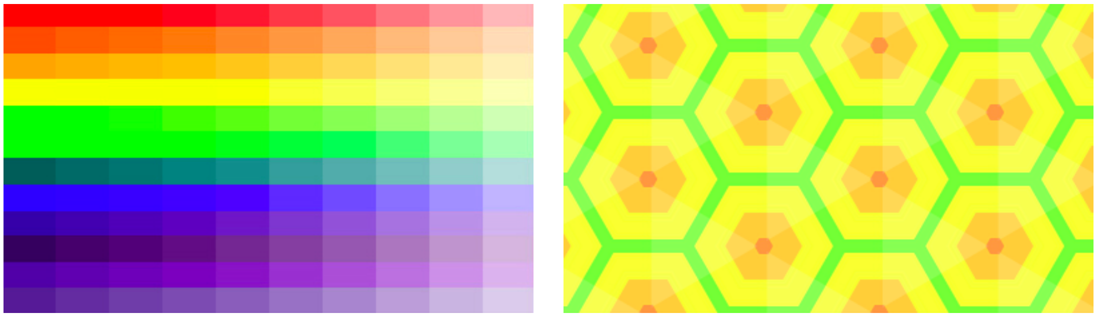
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryTileEffect

Localized Display Name

CISixfoldReflectedTile

Figure 16-96 The result of using the CISixfoldReflectedTile filter



Availability

Available in Mac OS X v10.5 and later.

CISixfoldRotatedTile

Produces a tiled image from a source image by rotating the source image at increments of 60 degrees.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

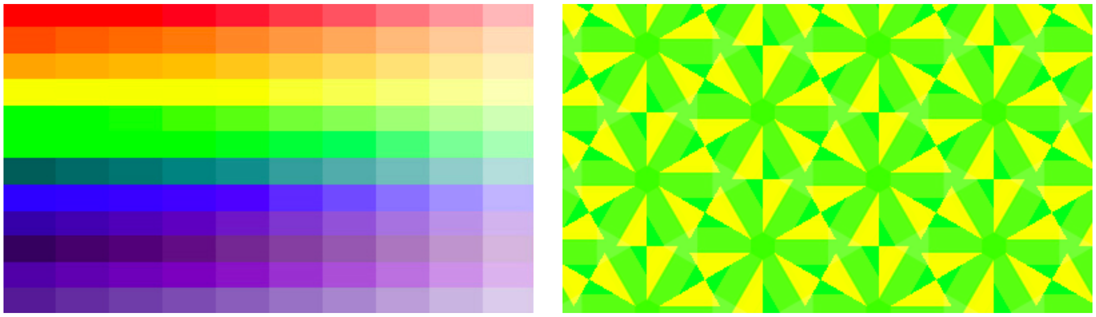
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name
CISixfoldRotatedTile

Figure 16-97 The result of using the CISixfoldRotatedTile filter



Availability
Available in Mac OS X v10.5 and later.

CISoftLightBlendMode

Either darkens or lightens colors, depending on the source image sample color.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

A `CIImage` class whose display name is Background Image.

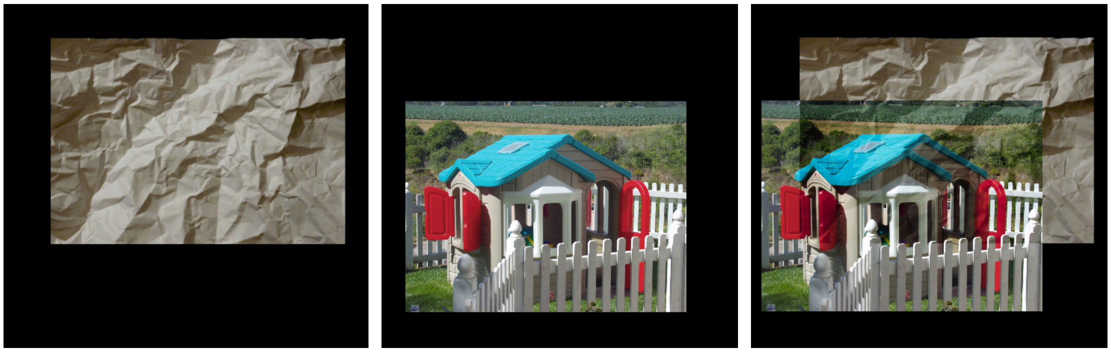
Discussion
If the source image sample color is lighter than 50% gray, the background is lightened, similar to dodging. If the source image sample color is darker than 50% gray, the background is darkened, similar to burning. If the source image sample color is equal to 50% gray, the background is not changed. Image samples that are

equal to pure black or pure white produce darker or lighter areas, but do not result in pure black or white. The overall effect is similar to what you would achieve by shining a diffuse spotlight on the source image. The formula used to create this filter is described in the PDF specification, which is available online from the Adobe Developer Center. See PDF Blend Modes: Addendum.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Soft Light Blend Mode

Figure 16-98 The result of using the CISoftLightBlendMode filter



Availability
Available in Mac OS X v10.4 and later.

CISourceAtopCompositing

Places the source image over the background image, then uses the luminance of the background image to determine what to show.

Parameters

`inputImage`
A `CIImage` class whose display name is Image.

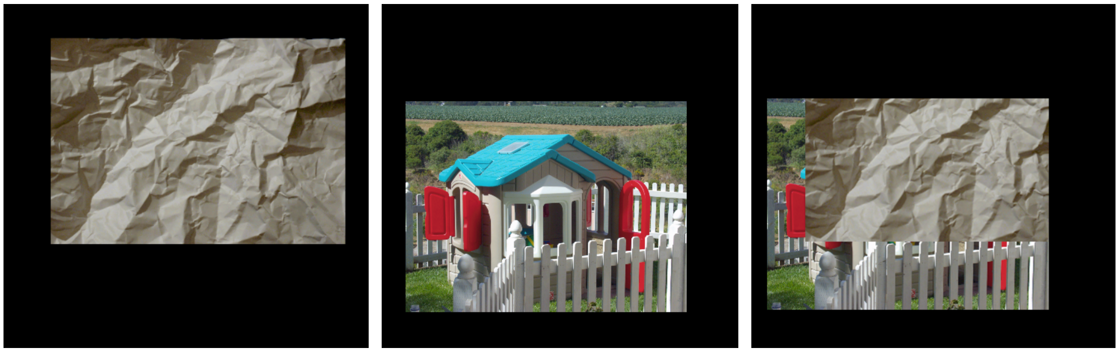
`inputBackgroundImage`
A `CIImage` class whose display name is Background Image.

Discussion
The composite shows the background image and only those portions of the source image that are over visible parts of the background. The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

Member of
`CICategoryBuiltIn`, `CICategoryHighDynamicRange`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name
Source Atop

Figure 16-99 The result of using the CISourceAtopCompositing filter



Availability

Available in Mac OS X v10.4 and later.

CISourceInCompositing

Uses the second image to define what to leave in the source image, effectively cropping the image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputBackgroundImage

A `CIImage` class whose display name is Background Image.

Discussion

The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

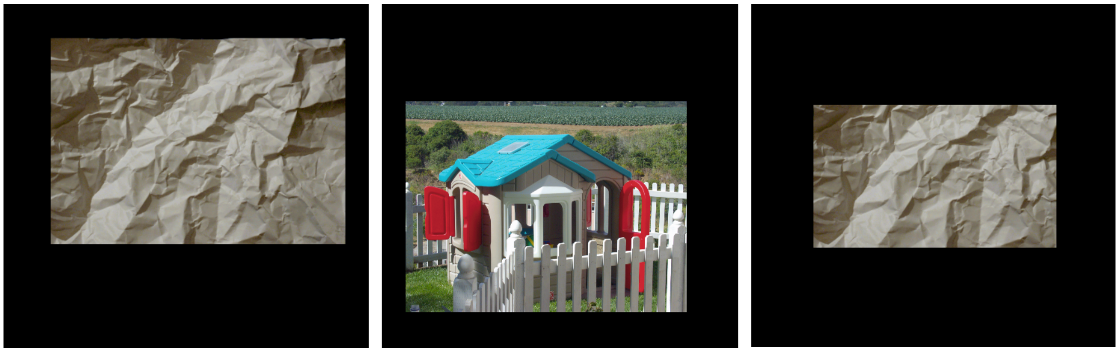
Member of

`CICategoryBuiltIn`, `CICategoryHighDynamicRange`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name

Source In

Figure 16-100 The result of using the CISourceInCompositing filter



Availability
Available in Mac OS X v10.4 and later.

CISourceOutCompositing

Uses the second image to define what to take out of the first image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputBackgroundImage

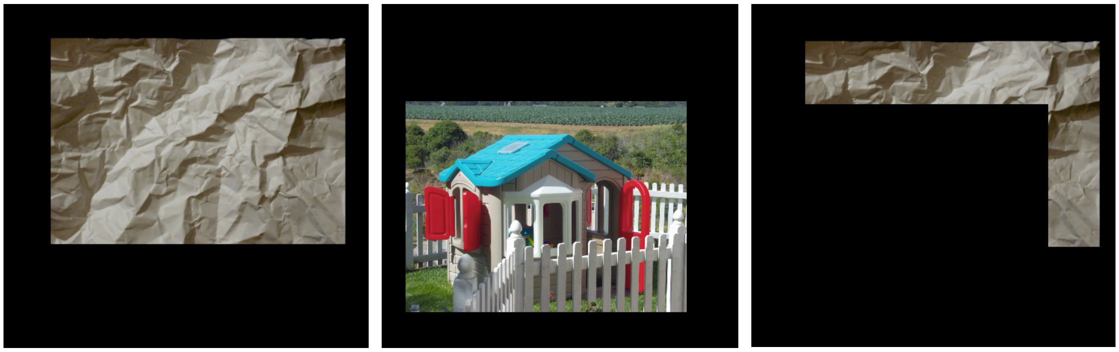
A CIImage class whose display name is Background Image.

Discussion
The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

Member of
CICategoryBuiltIn, CICategoryHighDynamicRange, CICategoryNonSquarePixels, CICategoryInterlaced, CICategoryStillImage, CICategoryVideo, CICategoryCompositeOperation

Localized Display Name
Source Out

Figure 16-101 The result of using the CISourceOutCompositing filter



Availability
Available in Mac OS X v10.4 and later.

CISourceOverCompositing

Places the second image over the first.

Parameters

inputImage

A CIImage class whose display name is Image.

inputBackgroundImage

A CIImage class whose display name is Background Image.

Discussion

The formula used to create this filter is described in Thomas Porter and Tom Duff. 1984. Compositing Digital Images. *Computer Graphics*, 18 (3): 253-259.

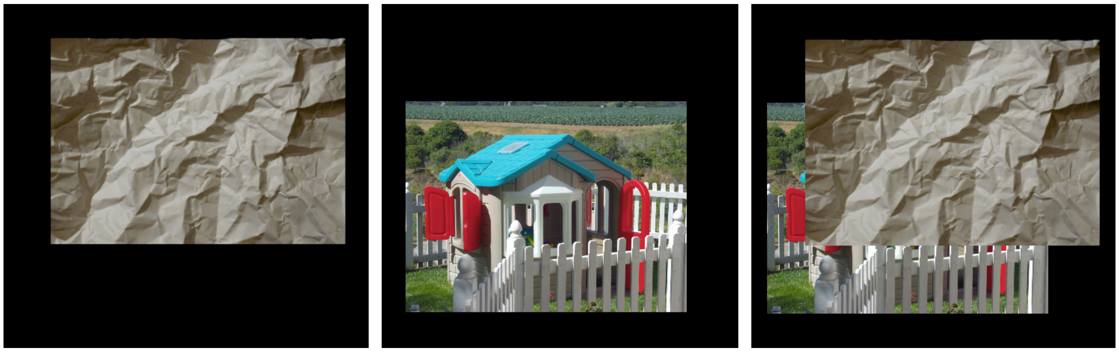
Member of

`CICategoryBuiltIn`, `CICategoryHighDynamicRange`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryCompositeOperation`

Localized Display Name

Source Over

Figure 16-102 The result of using the CISourceOverCompositing filter



Availability

Available in Mac OS X v10.4 and later.

CISpotColor

Replaces one or more color ranges with spot colors.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputCenterColor1`

A `CIColor` class whose display name is Center Color 1.

`inputReplacementColor1`

A `CIColor` class whose display name is Replacement Color 1.

`inputCloseness1`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Closeness1.

Default value: 0.22 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 0.50 Identity: 0.00

`inputContrast1`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Contrast 1.

Default value: 0.98 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputCenterColor2

A CIColor class whose display name is Center Color 2.

inputReplacementColor2

A CIColor class whose display name is Replacement Color 2.

inputCloseness2

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Closeness 2.

Default value: 0.15 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 0.50 Identity: 0.00

inputContrast2

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Contrast 2.

Default value: 0.98 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputCenterColor3

A CIColor class whose display name is Center Color 3.

inputReplacementColor3

A CIColor class whose display name is Replacement Color 3.

inputCloseness3

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Closeness 3.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 0.50 Identity: 0.00

inputContrast3

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Contrast 3.

Default value: 0.99 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

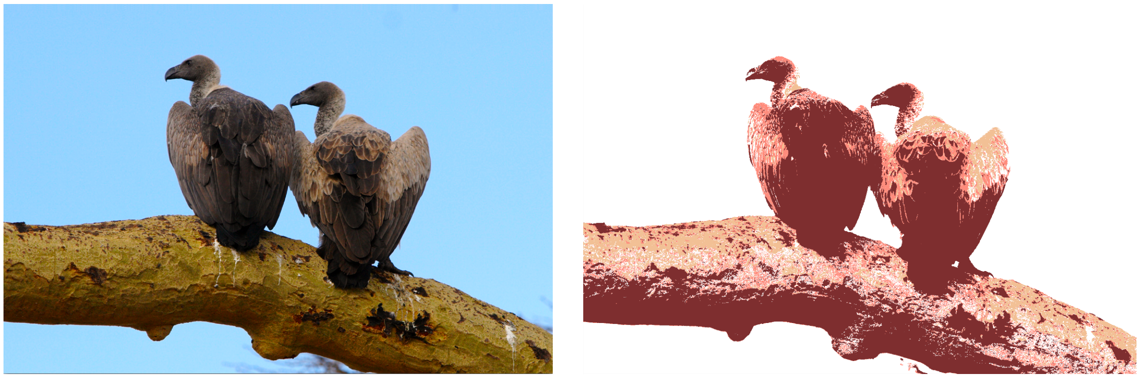
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryStylize

Localized Display Name

Spot Color

Figure 16-103 The result of using the CISpotColor filter



Availability
Available in Mac OS X v10.5 and later.

CISpotLight

Applies a directional spotlight effect to an image.

Parameters

inputImage

A CIImage class whose display name is Image.

inputLightPosition

A CIVector class whose attribute type is CIAttributeTypePosition3 and whose display name is Light Position.

Default value: [400 600 150] Identity: (null)

inputLightPointsAt

A CIVector class whose attribute type is CIAttributeTypePosition3 and whose display name is Light Points At.

Default value: [200 200 0] Identity: (null)

inputBrightness

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Brightness.

Default value: 3.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 10.00 Identity: 1.00

inputConcentration

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Concentration.

Default value: 0.10 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.50 Identity: 20.00

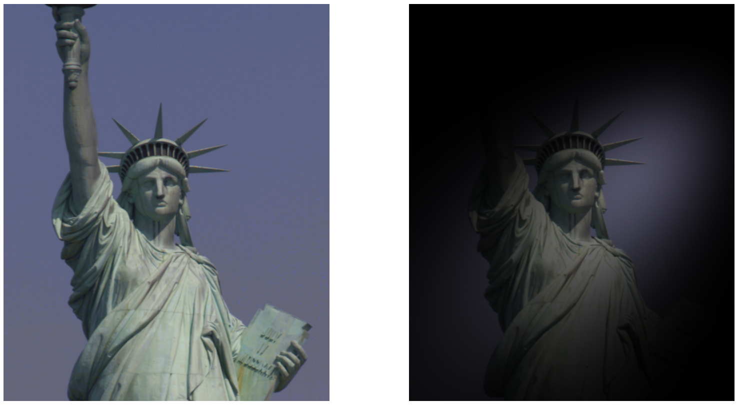
inputColor

A CIColor class whose attribute type is CIAttributeTypeOpaqueColor and whose display name is Color.

Member of
CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryStylize

Localized Display Name
Spot Light

Figure 16-104 The result of using the CISpotLight filter



Availability
Available in Mac OS X v10.4 and later.

CIStarShineGenerator

Generates a starburst pattern.

Parameters

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputColor`

A `CIColor` class whose display name is Color.

`inputRadius`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 50.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 300.00 Identity: 50.00

`inputCrossScale`

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Cross Scale.

Default value: 15.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 15.00

`inputCrossAngle`

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Cross Angle.

Default value: 0.60 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.60

inputCrossOpacity

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Cross Opacity.

Default value: -2.00 Minimum: -8.00 Maximum: 0.00 Slider minimum: -8.00 Slider maximum: 0.00 Identity: -2.00

inputCrossWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Cross Width.

Default value: 2.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.50 Slider maximum: 10.00 Identity: 2.50

inputEpsilon

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Epsilon.

Default value: -2.00 Minimum: -8.00 Maximum: 0.00 Slider minimum: -8.00 Slider maximum: 0.00 Identity: -2.00

Discussion

The output image is typically used as input to another filter

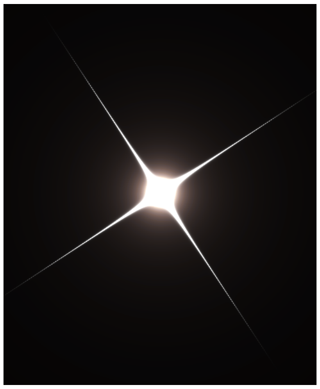
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryGenerator

Localized Display Name

Star Shine

Figure 16-105 The result of using the CIStarShineGenerator filter



Availability

Available in Mac OS X v10.4 and later.

CIStripesGenerator

Generates a stripe pattern.

Parameters

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputColor0

A CIColor class whose display name is Color 1.

inputColor1

A CIColor class whose display name is Color 2.

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 80.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 80.00

inputSharpness

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Sharpness.

Default value: 1.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 1.00

Discussion

You can control the color of the stripes, the spacing, and the contrast.

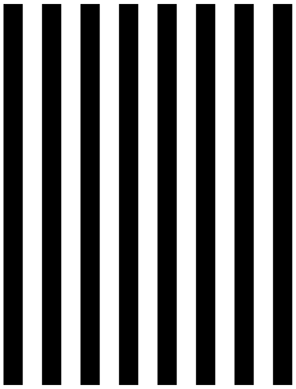
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryGenerator

Localized Display Name

Stripes

Figure 16-106 The result of using the CIStripesGenerator filter



Availability

Available in Mac OS X v10.4 and later.

CISunbeamsGenerator

Generates a sun effect.

Parameters

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputColor

A CIColor class whose display name is Color.

inputSunRadius

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Sun Radius.

Default value: 40.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 40.00

inputMaxStriationRadius

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Maximum Striation Radius.

Default value: 2.58 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 10.00 Identity: 2.58

inputStriationStrength

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Striation Strength.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 3.00 Identity: 0.50

inputStriationContrast

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Striation Contrast.

Default value: 1.38 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 5.00 Identity: 1.38

inputTime

An NSNumber class whose attribute type is CIAttributeTypeScalar and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Discussion

You typically use the output of the sunbeams filter as input to a composite filter.

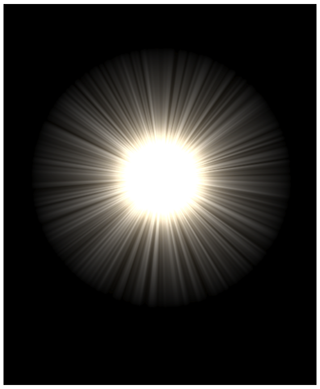
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryGenerator

Localized Display Name

Sunbeams

Figure 16-107 The result of using the CISunbeamsGenerator filter



Availability
Available in Mac OS X v10.4 and later.

CISwipeTransition

Transitions from one image to another by simulating a swiping action.

Parameters

inputImage

A CIImage class whose display name is Image.

inputTargetImage

A CIImage class whose display name is Target Image.

inputExtent

A CIVector class whose attribute type is CIAttributeTypeRectangle and whose display name is Extent.

Default value: [0 0 300 300] Identity: (null)

inputColor

A CIColor class whose attribute type is CIAttributeTypeOpaqueColor and whose display name is Color.

inputTime

An NSNumber class whose attribute type is CIAttributeTypeTime and whose display name is Time.

Default value: 0.00 Minimum: 0.00 Maximum: 1.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 300.00 Minimum: 0.10 Maximum: 0.00 Slider minimum: 0.10 Slider maximum: 800.00 Identity: 0.00

inputOpacity

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Opacity.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

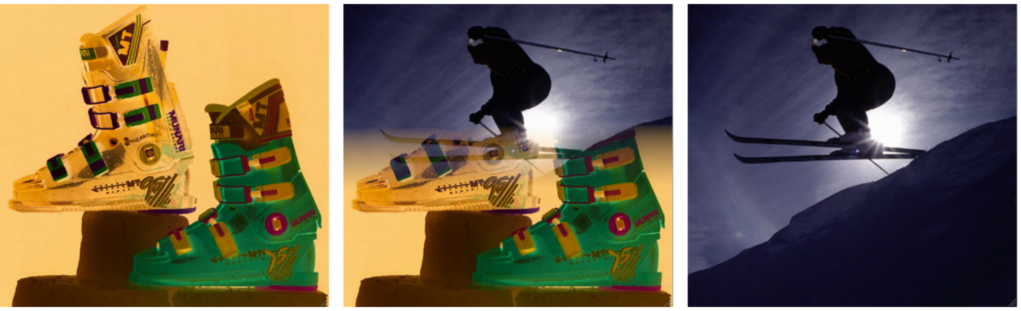
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTransition`

Localized Display Name

Swipe

Figure 16-108 The result of using the CISwipeTransition filter



Availability

Available in Mac OS X v10.4 and later.

CITorusLensDistortion

Creates a torus-shaped lens and distorts the portion of the image over which the lens is placed.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 160.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 500.00 Identity: 160.00

inputWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 80.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 200.00 Identity: 80.00

inputRefraction

An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Refraction.

Default value: 1.70 Minimum: -5.00 Maximum: 0.00 Slider minimum: -5.00 Slider maximum: 5.00 Identity: 0.00

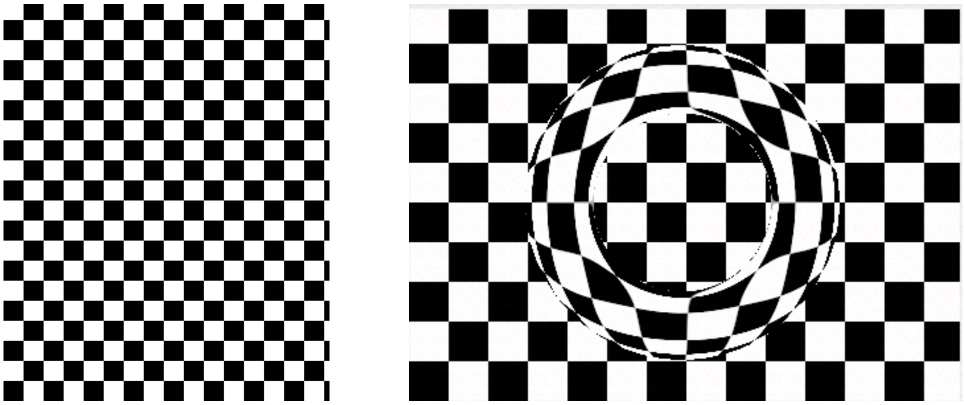
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name

Torus Lens Distortion

Figure 16-109 The result of using the CITorusLensDistortion filter



Availability

Available in Mac OS X v10.4 and later.

CITriangleTile

Maps a triangular portion of image to a triangular area and then tiles the result.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

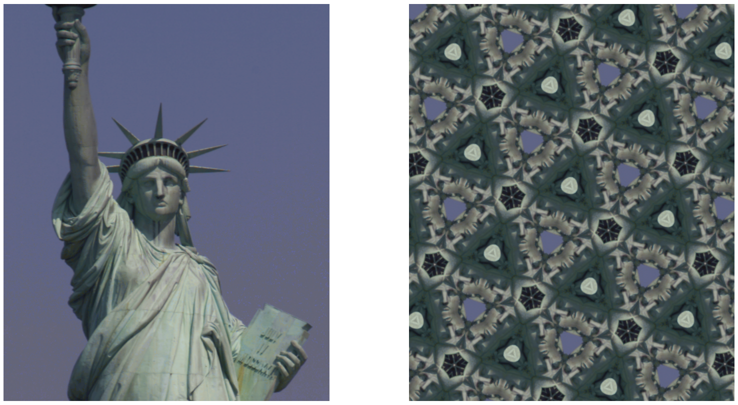
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryTileEffect`

Localized Display Name

Triangle Tile

Figure 16-110 The result of using the CITriangleTile filter



Availability

Available in Mac OS X v10.4 and later.

CITwelvefoldReflectedTile

Produces a tiled image from a source image by rotating the source image at increments of 30 degrees.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAngle

An NSNumber class whose attribute type is CIAttributeTypeAngle and whose display name is Angle.

Default value: 0.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: -3.14 Slider maximum: 3.14 Identity: 0.00

inputWidth

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Width.

Default value: 100.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 1.00 Slider maximum: 200.00 Identity: 100.00

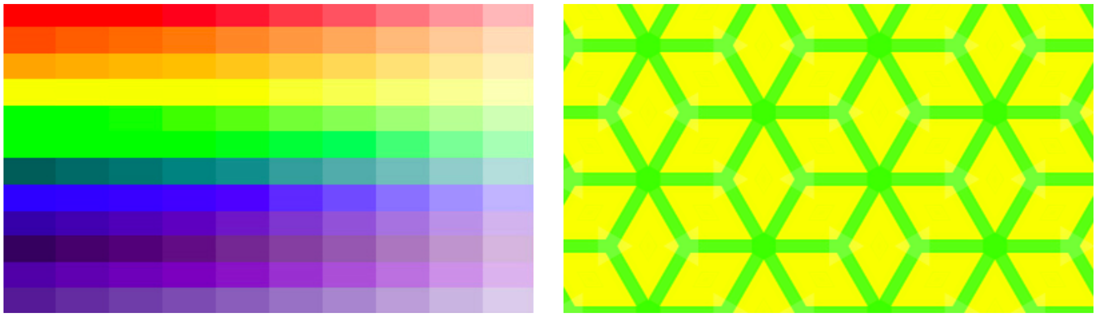
Member of

CICategoryBuiltIn, CICategoryStillImage, CICategoryVideo, CICategoryTileEffect

Localized Display Name

CITwelvefoldReflectedTile

Figure 16-111 The result of using the CITwelvefoldReflectedTile filter



Availability

Available in Mac OS X v10.5 and later.

CITwirlDistortion

Rotates pixels around a point to give a twirling effect.

Parameters

inputImage

A CIImage class whose display name is Image.

inputCenter

A CIVector class whose attribute type is CIAttributeTypePosition and whose display name is Center.

Default value: [150 150] Identity: (null)

inputRadius

An NSNumber class whose attribute type is CIAttributeTypeDistance and whose display name is Radius.

Default value: 300.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 500.00 Identity: 300.00

inputAngle

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

Default value: 3.14 Minimum: 0.00 Maximum: 0.00 Slider minimum: -12.57 Slider maximum: 12.57 Identity: 0.00

Discussion

You can specify the number of rotations as well as the center and radius of the effect.

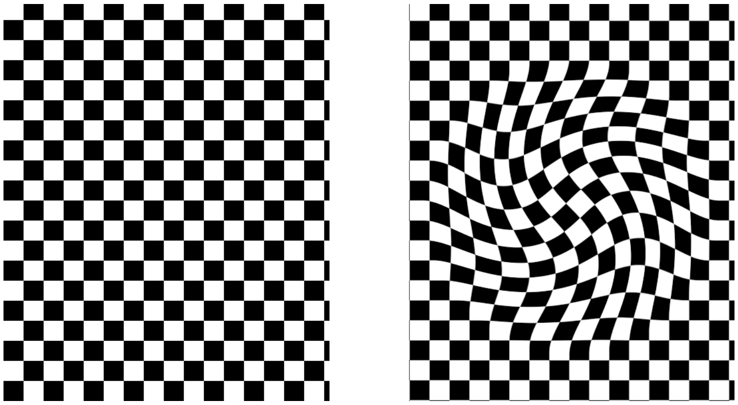
Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name

Twirl Distortion

Figure 16-112 The result of using the CITwirlDistortion filter



Availability

Available in Mac OS X v10.4 and later.

CIUnsharpMask

Increases the contrast of the edges between pixels of different colors in an image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputRadius

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 2.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 100.00 Identity: 0.00

inputIntensity

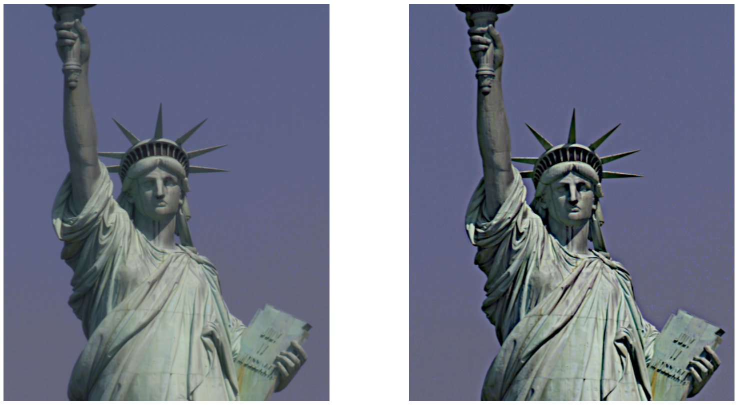
An `NSNumber` class whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 0.50 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 1.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategorySharpen`

Localized Display Name
Unsharp Mask

Figure 16-113 The result of using the CIUnsharpMask filter



Availability
Available in Mac OS X v10.4 and later.

CIVortexDistortion

Rotates pixels around a point to simulate a vortex.

Parameters

`inputImage`

A `CIImage` class whose display name is Image.

`inputCenter`

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

`inputRadius`

An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 300.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 800.00 Identity: 300.00

`inputAngle`

An `NSNumber` class whose attribute type is `CIAttributeTypeAngle` and whose display name is Angle.

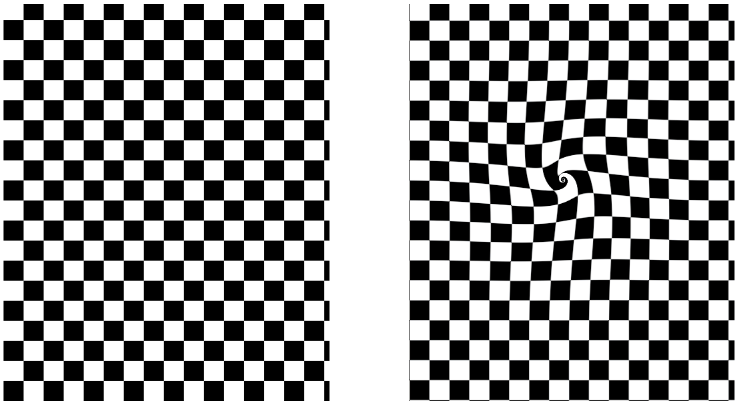
Default value: 56.55 Minimum: 0.00 Maximum: 0.00 Slider minimum: -94.25 Slider maximum: 94.25 Identity: 0.00

Discussion
You can specify the number of rotations as well the center and radius of the effect.

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryDistortionEffect`

Localized Display Name
Vortex Distortion

Figure 16-114 The result of using the CIVortexDistortion filter



Availability
Available in Mac OS X v10.4 and later.

CIWhitePointAdjust

Adjusts the reference white point for an image and maps all colors in the source using the new reference.

Parameters

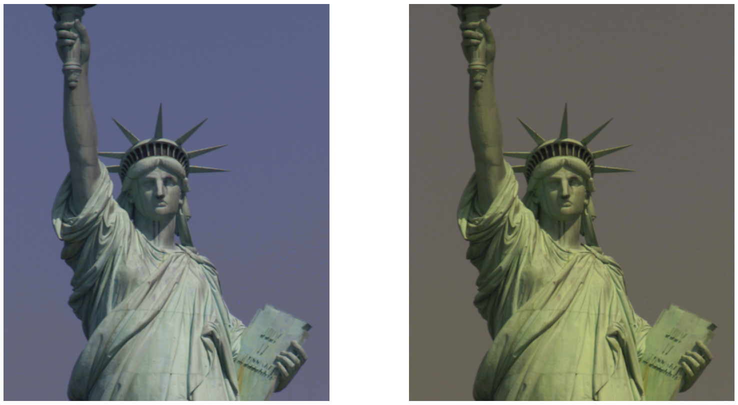
`inputImage`
A `CIImage` class whose display name is Image.

`inputColor`
A `CIColor` class whose display name is Color.

Member of
`CICategoryBuiltIn`, `CICategoryNonSquarePixels`, `CICategoryInterlaced`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryColorAdjustment`

Localized Display Name
White Point Adjust

Figure 16-115 The result of using the CIWhitePointAdjust filter



Availability
Available in Mac OS X v10.4 and later.

CIZoomBlur

Simulates the effect of zooming the camera while capturing the image.

Parameters

inputImage

A `CIImage` class whose display name is Image.

inputCenter

A `CIVector` class whose attribute type is `CIAttributeTypePosition` and whose display name is Center.

Default value: [150 150] Identity: (null)

inputAmount

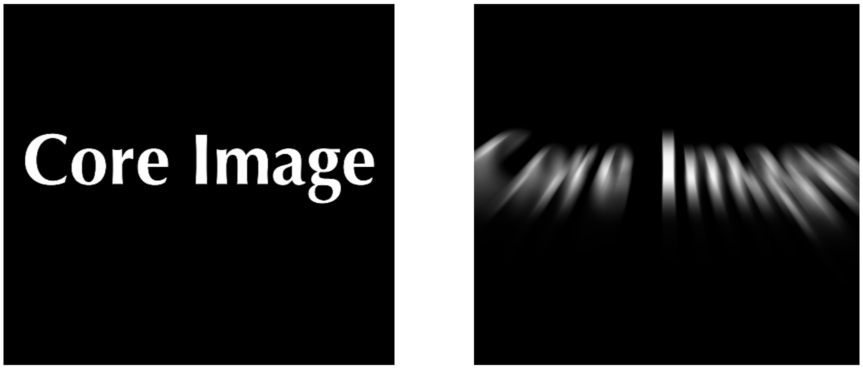
An `NSNumber` class whose attribute type is `CIAttributeTypeDistance` and whose display name is Amount.

Default value: 20.00 Minimum: 0.00 Maximum: 0.00 Slider minimum: 0.00 Slider maximum: 200.00 Identity: 0.00

Member of
`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryBlur`

Localized Display Name
Zoom Blur

Figure 16-116 The result of using the CIZoomBlur filter



Availability
Available in Mac OS X v10.4 and later.

Document Revision History

This table describes the changes to *Core Image Reference Collection*.

Date	Notes
2006-12-05	Updated for Mac OS X v10.5. Added several documents.
2006-06-28	Added two documents to the collection.
2006-05-23	First publication of this content as a collection of separate documents.