imlib2 egg

Chicken bindings for the imlib2 image library.
Extension for Chicken Scheme
Version 0.3

Peter Bex

Table of Contents

1 About this egg **1**
- 1.1 Version history 1
- 1.2 Requirements 1
- 1.3 Usage 1

2 Documentation **2**
- 2.1 Image creation, destruction and friends 2
- 2.2 Image properties 2
- 2.3 Image manipulation operations 3
  - 2.3.1 Orientation 3
    - 2.3.1.1 Initiation 4
  - 2.3.2 Texture/retouching functions 4
- 2.4 Pixel query functions 5
- 2.5 Color specifiers 6
- 2.6 Drawing functions 6

3 License **8**

Index **9**

1 About this egg

1.1 Version history

`0.3`	Set GC finalizers on all functions that create new imlib objects and allow linking against imlib2 that was compiled without X and converted documentation to eggdoc.
`0.2`	`imlib:alpha-set!` wasn't exported
`0.1`	beta release

1.2 Requirements

This egg requires the following extensions:

```
syntax-case
```

1.3 Usage

Load this egg like so:

```
(require-extension imlib2)
```

2 Documentation

Note: Not all imlib functionality is provided by this egg yet!

2.1 Image creation, destruction and friends

`imlib:create` [procedure]

`(imlib:create width height)`

Returns a new `imlib:image` object which describes a transparent image of the given size.

`imlib:image?` [procedure]

`(imlib:image? img)`

Determine if the given `img` object is an imlib image.

`imlib:destroy` [procedure]

`(imlib:destroy img)`

Destroy the given image.

`imlib:clone` [procedure]

`(imlib:clone img)`

Create a fresh copy of the image object `img`

`imlib:load` [procedure]

`(imlib:load filename)`

Returns a new `imlib:image` object which describes the image stored in the file `filename`. This automatically uses the correct loader, as determined by Imlib2 on the basis of the file's extension.

`imlib:save` [procedure]

`(imlib:save img filename)`

Store the imlib image object described by `img` in the file `filename`. The right loader is automatically selected by Imlib2 if you haven't set it explicitly with `imlib:format-set!`.

2.2 Image properties

`imlib:format` [procedure]

`(imlib:format img)`

Request the currently set format for the image, or `#f` if no format has been associated with it yet.

`imlib:format-set!` [procedure]

`(imlib:format-set! img format)`

Explicitly set the file format on the image for subsequent calls to `imlib:save`. The `format` argument is passed to the loaders in the same way a file extension would be.

`imlib:width` [procedure]

`(imlib:width img)`

Returns the width of the supplied image, in pixels.

`imlib:height` [procedure]

`(imlib:height img)`

Returns the height of the supplied image, in pixels.

`imlib:filename` [procedure]

`(imlib:filename img)`

Returns the original filename for the image, if it was loaded from a file. Otherwise it returns `#f`.

`imlib:alpha?` [procedure]

`(imlib:alpha? img)`

Does the image have an alpha layer?

`imlib:alpha-set!` [procedure]

`(imlib:alpha-set! img value)`

Enable or disable alpha layer support for the image.

`imlib:track-changes-on-disk` [procedure]

`(imlib:track-changes-on-disk img)`

From now on, track changes on disk to the file that is associated with `img`. By default, all images are cached by imlib2 in such a way that closing and reopening it just pulls it from cache instead of really loading it. Unfortunately, there's no way to request the status of this option or disable it.

2.3 Image manipulation operations

test paragraph

2.3.1 Orientation

`imlib:flip-horizontal` [procedure]

`(imlib:flip-horizontal img)`

Create a new, horizontally flipped, copy of `img`.

`imlib:flip-horizontal!` [procedure]

`(imlib:flip-horizontal! img)`

Destructively flip `img` horizontally.

`imlib:flip-vertical` [procedure]

`(imlib:flip-vertical img)`

Create a new, vertically flipped, copy of `img`.

2.3.1.1 Initiation

`imlib:flip-vertical!` [procedure]

`(imlib:flip-vertical! img)`

Destructively flip `img` vertically.

`imlib:flip-diagonal` [procedure]

`(imlib:flip-diagonal img)`

Create a new, diagonally flipped, copy of `img`. This works like transposing a matrix.

`imlib:flip-diagonal!` [procedure]

`(imlib:flip-diagonal! img)`

Destructively flip `img` diagonally.

`imlib:orientate` [procedure]

`(imlib:orientate img orientation)`

Create a new, orientated copy of `img`. According to imlib2 documentation, this function rotates the image by 90 degrees `orientation` times. However, the function accepts values between 0 and 7, inclusive. What values 4-7 do, I'm not really sure of. They appear to rotate the image `(mod orientation 3)` times 90 degrees, but flip it as well.

2.3.2 Texture/retouching functions

`imlib:sharpen` [procedure]

`(imlib:sharpen img radius)`

Create a new, sharpened copy of `img`. The `radius` argument is an integer number indicating the degree of sharpening that has to take place. I am not sure what a negative value means, but it is allowed.

`imlib:sharpen!` [procedure]

`(imlib:sharpen! img radius)`

Destructively sharpen an image.

`imlib:blur` [procedure]

`(imlib:blur img radius)`

Create a new, blurred copy of `img`. The `radius` argument is a positive integer indicating the blur matrix radius, so 0 has no effect.

`imlib:blur!` [procedure]

`(imlib:blur! img radius)`

Destructively blur an image.

`imlib:tile` [procedure]

`(imlib:tile img)`

Create a new copy of `img` adjusted in such a way around the edges, such that it is suitable for use in repeating ("tiled") patterns on all sides.

`imlib:tile!` [procedure]

```
(imlib:tile! img)
```

Destructively tile an image.

`imlib:tile-horizontal` [procedure]

```
(imlib:tile-horizontal img)
```

Create a new copy of `img` adjusted on the left and right edges so it can be used for horizontally repeating patterns.

`imlib:tile-horizontal!` [procedure]

```
(imlib:tile-horizontal! img)
```

Destructively tile an image horizontally.

`imlib:tile-vertical` [procedure]

```
(imlib:tile-vertical img)
```

Create a new copy of `img` adjusted on the top and bottom edges so it can be used for vertically repeating patterns.

`imlib:tile-vertical!` [procedure]

```
(imlib:tile-vertical! img)
```

Destructively tile an image vertically.

`imlib:crop` [procedure]

```
(imlib:crop img x y width height)
```

Create a new, cropped copy of `img`. The `x` and `y` parameters indicate the upper left pixel of the new image. The `width` and `height` parameters indicate the size of the new image. Returns `#f` on failure. **Note: This function will return an image of the requested size, but with undefined contents if you pass it arguments that lie outside the image! I am still unsure if this is a bug or feature.**

`imlib:scale` [procedure]

```
(imlib:scale img width height)
```

Create a new, scaled copy of the image.

`imlib:crop&scale` [procedure]

```
(imlib:crop&scale img src-x src-y src-width src-height dest-width dest-height)
```

Create a new, cropped *and* scaled copy of `img`. The arguments `src-x`, `src-y`, `src-width` and `src-height` indicate cropping dimensions as per `imlib:crop`, in the original image. The `dest-width` and `dest-height` arguments indicate the new image's width and height.

2.4 Pixel query functions

`imlib:pixel/rgba` [procedure]

```
(imlib:pixel/rgba img x y)
```

Returns the RGBA (red, green, blue, alpha) color values for the image at the specified pixel coordinate as 4 values.

`imlib:pixel/hsva` [procedure]

`(imlib:pixel/hsva img x y)`

Returns the HSVA (hue, saturation, value, alpha) color values for the image at the specified pixel coordinate as 4 values.

`imlib:pixel/hlsa` [procedure]

`(imlib:pixel/hlsa img x y)`

Returns the HLSA (hue, lightness, saturation, alpha) color values for the image at the specified pixel coordinate as 4 values.

`imlib:pixel/cmya` [procedure]

`(imlib:pixel/cmya img x y)`

Returns the CMYA (cyan, magenta, yellow, alpha) color values for the image at the specified pixel coordinate as 4 values.

2.5 Color specifiers

Note: This could use some more work. Perhaps the functions fromthe previous section should return values of these types instead.

`imlib:color?` [procedure]

`(imlib:color? color)`

Is the specified object an imlib color specifier?

`imlib:color/rgba` [procedure]

`(imlib:color/rgba r g b a)`

Create a color specifier for the given RGBA values.

`imlib:color/hsva` [procedure]

`(imlib:color/hsva h s v a)`

Create a color specifier for the given HSVA values.

`imlib:color/hlsa` [procedure]

`(imlib:color/hlsa h l s a)`

Create a color specifier for the given HLSA values.

`imlib:color/cmya` [procedure]

`(imlib:color/cmya c m y a)`

Create a color specifier for the given CMYA values.

2.6 Drawing functions

`imlib:draw-pixel` [procedure]

`(imlib:draw-pixel img color x y)`

Draw a pixel in the given image on the given coordinates with the given color specifier.

`imlib:draw-line` [procedure]

`(imlib:draw-line img color x1 y1 x2 y2)`

Draw a line in the image from the coordinates (`x1`,`y1`) to (`x2`,`y2`.

`imlib:draw-rectangle` [procedure]

`(imlib:draw-rectangle img color x y width height)`

Draw a one-pixel wide outline of a rectangle with the given color. The `x` and `y` coordinates are of the upper left corner of the rectangle.

`imlib:fill-rectangle` [procedure]

`(imlib:fill-rectangle img color x y width height)`

Same as `imlib:draw-rectangle`, but filled in.

3 License

Index

imlib:alpha-set! . . . 3
imlib:alpha? . . . 3
imlib:blur . . . 4
imlib:blur! . . . 4
imlib:clone . . . 2
imlib:color/cmya . . . 6
imlib:color/hlsa . . . 6
imlib:color/hsva . . . 6
imlib:color/rgba . . . 6
imlib:color? . . . 6
imlib:create . . . 2
imlib:crop . . . 5
imlib:crop&scale . . . 5
imlib:destroy . . . 2
imlib:draw-line . . . 6
imlib:draw-pixel . . . 6
imlib:draw-rectangle . . . 7
imlib:filename . . . 3
imlib:fill-rectangle . . . 7
imlib:flip-diagonal . . . 4
imlib:flip-diagonal! . . . 4
imlib:flip-horizontal . . . 3
imlib:flip-horizontal! . . . 3
imlib:flip-vertical . . . 3
imlib:flip-vertical! . . . 4
imlib:format . . . 2
imlib:format-set! . . . 2
imlib:height . . . 3
imlib:image? . . . 2
imlib:load . . . 2
imlib:orientate . . . 4
imlib:pixel/cmya . . . 6
imlib:pixel/hlsa . . . 6
imlib:pixel/hsva . . . 6
imlib:pixel/rgba . . . 5
imlib:save . . . 2
imlib:scale . . . 5
imlib:sharpen . . . 4
imlib:sharpen! . . . 4
imlib:tile . . . 4
imlib:tile! . . . 5
imlib:tile-horizontal . . . 5
imlib:tile-horizontal! . . . 5
imlib:tile-vertical . . . 5
imlib:tile-vertical! . . . 5
imlib:track-changes-on-disk . . . 3
imlib:width . . . 3