LUA Image Processor

Paolo Medici
©2005-2006

8 marzo 2007

1 Introduction

For information about LUA programming, visit `www.lua.org`. This book is not intended to explain how develop LUA application, but only how to concern LIP software native function extensions.

LIP source code are under GPL license.

LIP uses FreeImage (`http://freeimage.sourceforge.net/`) and FreeType (`http://sourceforge.net/projects/freetype`).

Due to particular reference frame provided by FreeImage the coordinate (0,0) of image are placed in bottom-left corner.

2 API

2.1 System Call

`number argc()`

Return the number of argument passed to script

```
string argv(number n)
```

return the n-th param.

2.2 Image Library

Image is an object (user data with metatables), image is image library.

2.3 Image library methods

```
number image.getfiletype(filename)
```

Orders FreeImage to analyze the bitmap signature. The function then returns one of the predefined FREE_IMAGE_FORMAT constants or a bitmap identification number registered by a plugin. Because not all formats can be identified by their header, filename extension are used to determinate the file format.

```
number image.getfiletypebyext(filename)
```

Return the predefined FREE_IMAGE_FORMAT constants looking only the filename extension.

```
Image image.load(filename, FormatID, flag = 0)
```

This function decodes a bitmap, allocates memory for it and then returns it as a Image (Lua UserData/Object). A parameter defines the type of bitmap to be loaded. For example, when FIF_BMP is passed, a BMP file is loaded into memory (an overview of possible FREE_IMAGE_FORMAT constants is available in Constants section). Optional Flag (default is 0) can be used to change some load specific behaviour.

```
Image image.load(filename)
```

Try to load image and use `getfiletype` to determinate the image type.

```
Image image.new(width, height, bpp)
```

Create new image width specified width,height and bpp. Only bpp 8,24,32 are currently supported. Return a LUA userdata Image Object.

2.3.1 Images Arithmetic Operations

`Image image.add(a,b)`

Sum (clamping) two images

`Image image.sub(a,b)`

Subtract (clamping) two images

`Image image.lightest(a,b)`

Return lightest pixels of two images

`Image image.darkest(a,b)`

Return darkest pixels of two images

`Image image.avg(a,b)`

Return mix of the images

`Image image.blend(a,b,perc)`

Mix image of the images using *perc* factor (0.0 .. 1.0)

2.4 Image Methods

`Image:save(filename, FormatID, flag = 0)`

Save image onto filename with FormatID file format (see Costants). Flag (0 used as default) can used to specify image quality.

`Image:save(filename)`

Save image onto filename using format returned by `getfiletypebyext`

```
number Image:width()
number Image:height()
```

Returns the width,height of the bitmap in pixel units.

`number Image:bpp()`

Returns the size of one pixel in the bitmap in bits. For example when each pixel takes 32-bits of space in the bitmap, this function returns 32. Possible bit depths are 1, 4, 8, 16, 24, 32 for standard bitmaps and 16-, 32-, 48-, 64-, 96- and 128-bit for non standard bitmaps.

`Image Image:rescale(width, height, FilterMethod)`

Rescale object to width and height, using FilterMethod (see Costants).

`Image Image:convert(bpp)`

Change bpp of object to 8,24,32.

`Image image:clone()`

Makes an exact reproduction of an existing bitmap, including metadata and attached profile if any.

`Image Image:crop(x0,y0,x1,y1)`

`Image Image:paste(x0,y0)`

`Image Image:rotateclassic(angle)`

`Image Image:blur(var)`

Apply a gaussian blur on the image

`Image:flip()`

Flip vertically the image

`Image:mirror()`

Flip horizzontally the image

`Image:invert()`

Invert colors in image

`Image:binarize(th)`

Binarize image (0 under the threshold, 255 over the threshold)

`Image:quantize(levels)`

Quantize the channel to n levels

`Image:brightness(factor)`

Multiply any pixel value by $factor$.

`Image:wnoise(factor)`

Add white noise to image

`Image:adjustgamma(gamma)`

Change image gamma correction

`Image:merge(b,m)`

Mix current image with image b using mask image m

2.4.1 Drawing API

```
Image:color(grey)
Image:color(grey,alpha)
Image:color(r,g,b)
Image:color(r,g,b,alpha)
```

Set the current drawing color

`Image:box(x0,y0,x1,y1)`

Draw a filled box using drawing color

`Image:clear()`

Reset entire image using drawing color

`Image:rectangle(x0,y0,x1,y1)`

Draw a rectangle (frame)

`Image:line(x0,y0,x1,y1)`

Draw a line

`Image:circle(x,y,r,filled)`

Draw a circle

`Image:ellipse(x0,y0,x1,y1,filled)`

Draw an ellipse

`Image:pixel(x,y)`

Draw a point using current color.

```
Image:pixel(x,y, grey)
Image:pixel(x,y, grey,alpha)
Image:pixel(x,y, r,g,b)
Image:pixel(x,y, r,g,b,alpha)
```

Draw a point using specified color.

`(r,g,b,a) = Image:value(x,y)`

Return the pixel value

2.5 Font Library

2.6 GUI Library

`Window gui.window()`

Create new window

`Window gui.window(title)`

Create new window with specified title

`gui.flush()`

Flush al pending messages

2.6.1 Window Object

`window:title(title)`

Set Window Title

`window:image(Image)`

Set displayed image

2.7 Costants

FreeImage File Format Identifier:

```
FIF_UNKNOWN = -1
FIF_BMP = 0
FIF_ICO = 1
FIF_JPEG = 2
FIF_JNG = 3
FIF_KOALA = 4
FIF_LBM = 5
FIF_IFF = FIF_LBM
FIF_MNG = 6
FIF_PBM = 7
FIF_PBMRAW = 8
FIF_PCD = 9
FIF_PCX = 10
FIF_PGM = 11
FIF_PGMRAW = 12
FIF_PNG = 13
FIF_PPM = 14
FIF_PPMRAW = 15
FIF_RAS = 16
FIF_TARGA = 17
FIF_TIFF = 18
FIF_WBMP = 19
FIF_PSD = 20
FIF_CUT = 21
FIF_XBM = 22
FIF_XPM = 23
FIF_DDS = 24
FIF_GIF     = 25
FIF_HDR = 26
FIF_FAXG3 = 27
FIF_SGI = 28
```

FreeImage Resample Filtering:

```
FILTER_BOX   = 0
FILTER_BICUBIC   = 1
FILTER_BILINEAR   = 2
FILTER_BSPLINE   = 3
```

```
FILTER_CATMULLROM = 4
FILTER_LANCZOS3   = 5
```