Imlib2 Reference Manual

Generated by Doxygen 1.5.1

Wed Mar 28 00:04:39 2007

Contents

Chapter 1

Imlib2 Library Documentation

Version:

1.1.1

Author:

Carsten Haitzler <raster@rasterman.com>

Date:

1999-2004

1.1 What is Imlib2 ?

Imlib 2 is the successor to Imlib. It is NOT a newer version - it is a completely new library. Imlib 2 can be installed alongside Imlib 1.x without any problems since they are effectively different libraries - BUT they Have very similar functionality.

Imlib 2 does the following:

- Load image files from disk in one of many formats
- Save images to disk in one of many formats
- Render image data onto other images
- Render images to an X-Windows drawable
- Produce pixmaps and pixmap masks of Images
- Apply filters to images
- Rotate images
- Accept RGBA Data for images
- Scale images

- Alpha blend Images on other images or drawables
- Apply color correction and modification tables and factors to images
- Render images onto images with color correction and modification tables
- Render truetype anti-aliased text
- Render truetype anti-aliased text at any angle
- Render anti-aliased lines
- Render rectangles
- Render linear multi-colored gradients
- Cache data intelligently for maximum performance
- Allocate colors automatically
- Allow full control over caching and color allocation
- Provide highly optimized MMX assembly for core routines
- Provide plug-in filter interface
- Provide on-the-fly runtime plug-in image loading and saving interface
- Fastest image compositing, rendering and manipulation library for X

If what you want isn't in the list above somewhere then likely Imlib 2 does not do it. If it does it it likely does it faster than any other library you can find (this includes gdk-pixbuf, gdkrgb, etc.) primarily because of highly optimized code and a smart subsystem that does the dirty work for you and picks up the pieces for you so you can be lazy and let all the optimizations for FOR you.

Imlib 2 can run without a display, so it can be easily used for background image processing for web sites or servers - it only requires the X libraries to be installed - that is all - it does not require an XServer to run unless you wish to display images.

The interface is simple - once you get used to it, the functions do exactly what they say they do.

1.2 A Simple Example

The best way to start is to show a simple example of an Imlib2 program. This one will load an image of any format you have a loader installed for (all loaders are dynamic code objects that Imlib2 will use and update automatically runtime - anyone is free to write a loader. All that has to be done is for the object to be dropped into the loaders directory with the others and all Imlib2 programs will automatically be able to use it - without a restart).

```
/* main program */
int main(int argc, char **argv)
{
  /* an image handle */
  Imlib_Image image;

  /* if we provided < 2 arguments after the command - exit */
  if (argc != 3) exit(1);
  /* load the image */
  image = imlib_load_image(argv[1]);
  /* if the load was successful */
```

```
  if (image)
    {
      char *tmp;
      /* set the image we loaded as the current context image to work on */
      imlib_context_set_image(image);
      /* set the image format to be the format of the extension of our last */
      /* argument - i.e. .png = png, .tif = tiff etc. */
      tmp = strrchr(argv[2], '.');
      if(tmp)
         imlib_image_set_format(tmp + 1);
      /* save the image */
      imlib_save_image(argv[2]);
    }
}
```

Now to compile this

```
cc imlib2_convert.c -o imlib2_convert `imlib2-config --cflags` `imlib2-config --libs`
```

You now have a program that if used as follows:

```
cc imlib2_con
./imlib2_convert image1.jpg image2.png
```

will convert image1.jpg into a png called image2.png. It is that simple.

1.3 How Image Loading Works

It is probably a good idea to discuss how Imlib2 actually loads an Image so the programmer knows what is going on, how to take advantage of the optimizations already there and to explain why things work as they do.

1.3.1 Loading using imlib_load_image();

This is likely to be by far the most common way to load an image - when you don't really care about the details of the loading process or why it failed - all you care about is if you got a valid image handle.

When you call this function Imlib2 attempts to find the file specified as the parameter. This will involve Imlib2 first checking to see if that file path already has been loaded and is in Imlib2's cache (a cache of already decoded images in memory to save having to load and decode from disk all the time). If there already is a copy in the cache (either already active or speculatively cached from a previous load & free) this copy will have its handle returned instead of Imlib2 checking on disk (in some circumstances this is not true - see later in this section to find out). This means if your program blindly loads an Image, renders it, then frees it - then soon afterwards loads the same image again, it will not be loaded from disk at all, instead it will simply be re-referenced from the cache - meaning the load will be almost instant. A great way to take full advantage of this is to set the cache to some size you are happy with for the image data being used by your application and then all rendering o an image follows the pseudo code:

```
set cache to some amount (e.g. 4 Mb)
...
rendering loop ...
    load image
    render image
    free image
... continue loop
```

This may normally sound silly - load image, render then free - EVERY time we want to use it, BUT - it is actually the smartest way to use Imlib2 - since the caching will find the image for you in the cache - you do not need to manage your own cache, or worry about filling up memory with image data - only as much memory as you have set for the cache size will actually ever be used to store image data - if you have lots of image data to work with then increase the cache size for better performance, but this is the only thing you need to worry about. you won't have problems of accidentally forgetting to free images later since you free them immediately after use.

Now what happens if the file changes on disk while it's in cache? By default nothing. The file is ignored. This is an optimization (to avoid hitting the disk to check if the file changed for every load if it's cached). You can inform Imlib2 that you care about this by using the imlib_image_set_changes_on_disk(); call. Do this whenever you load an Image that you expect will change on disk, and the fact that it changes really matters. Remember this will marginally reduce the caching performance.

Now what actually happens when we try and load an image using a filename? First the filename is broken down into 2 parts. the filename before a colon (:) and the key after the colon. This means when we have a filename like:

```
/path/to/file.jpg
```

the filename is:

```
/path/to/file.jpg
```

and the key is blank. If we have:

```
/path/to/file.db:key.value/blah
```

the filename is:

```
/path/to/file.db
```

and the key is:

```
key.value/blah
```

You may ask what is this thing with keys and filenames? Well Imlib2 has loaders that are able to load data that is WITHIN a file (the loader capable of this right now is the database loader that is able to load image data stored with a key in a Berkeley-db database file). The colon is used to delimit where the filename ends and the key begins. Fro the majority of files you load you won't have to worry, but there is a limit in this case that filenames cannot contain a color character.

First Imlib2 checks to see if the file exists and that you have permission to read it. If this fails it will abort the load. Now that it has checked that this is the case it evaluates that it's list of dynamically loaded loader modules it up to date then it runs through the loader modules until one of them claims it can load this file. If this is the case that loader is now used to decode the image and return an Image handle to the calling program. If the loader is written correctly and the file format sanely supports this, the loader will NOT decode any image data at this point. It will ONLY read the header of the image to figure out its size, if it has an alpha channel, format and any other header information. The loader is remembered and it will be re-used to load the image data itself later if and ONLY if the actual image data itself is needed. This means you can scan vast directories of files figuring their format and size and other such information just by loading and freeing - and it will be fast because no image data is decoded. You can take advantage of this

by loading the image and checking its size to calculate the size of an output area before you ever load the data. This means geometry calculations can be done fast ahead of time.

If you desire more detailed information about why a load failed you can use imlib_load_image_with_error_return(); and it will return a detailed error return code.

If you do not wish to have the image data loaded later using the optimized "deferred" method of loading, you can force the data to be decoded immediately with imlib_load_image_immediately();

If you wish to bypass the cache when loading images you can using imlib_load_image_without_cache(); and imlib_load_image_immediately_without_cache();.

Sometimes loading images can take a while. Often it is a good idea to provide feedback to the user whilst this is happening. This is when you set the progress function callback. Setting this to NULL will mean no progress function is called during load - this is the default. When it is set you set it to a function that will get called every so often (depending on the progress granularity) during load. Use imlib_context_set_progress_function(); and imlib_context_set_progress_granularity(); to set this up.

1.4 A more advanced Example

This is a more comprehensive example that should show off a fair number of features of imlib2. The code this was based off can be found in Imlib2's test directory. This covers a lot of the core of Imlib2's API so you should have a pretty good idea on how it works if you understand this code snippet.

```
/* include X11 stuff */
#include <X11/Xlib.h>
/* include Imlib2 stuff */
#include <Imlib2.h>
/* sprintf include */
#include <stdio.h>

/* some globals for our window & X display */
Display *disp;
Window   win;
Visual  *vis;
Colormap cm;
int      depth;

/* the program... */
int main(int argc, char **argv)
{
   /* events we get from X */
   XEvent ev;
   /* areas to update */
   Imlib_Updates updates, current_update;
   /* our virtual framebuffer image we draw into */
   Imlib_Image buffer;
   /* a font */
   Imlib_Font font;
   /* our color range */
   Imlib_Color_Range range;
   /* our mouse x, y coordinates */
   int mouse_x = 0, mouse_y = 0;

   /* connect to X */
   disp  = XOpenDisplay(NULL);
   /* get default visual , colormap etc. you could ask imlib2 for what it */
   /* thinks is the best, but this example is intended to be simple */
```

```
vis   = DefaultVisual(disp, DefaultScreen(disp));
depth = DefaultDepth(disp, DefaultScreen(disp));
cm    = DefaultColormap(disp, DefaultScreen(disp));
/* create a window 640x480 */
win = XCreateSimpleWindow(disp, DefaultRootWindow(disp),
                          0, 0, 640, 480, 0, 0, 0);
/* tell X what events we are interested in */
XSelectInput(disp, win, ButtonPressMask | ButtonReleaseMask |
             PointerMotionMask | ExposureMask);
/* show the window */
XMapWindow(disp, win);
/* set our cache to 2 Mb so it doesn't have to go hit the disk as long as */
/* the images we use use less than 2Mb of RAM (that is uncompressed) */
imlib_set_cache_size(2048 * 1024);
/* set the font cache to 512Kb - again to avoid re-loading */
imlib_set_font_cache_size(512 * 1024);
/* add the ./ttfonts dir to our font path - you'll want a notepad.ttf */
/* in that dir for the text to display */
imlib_add_path_to_font_path("./ttfonts");
/* set the maximum number of colors to allocate for 8bpp and less to 128 */
imlib_set_color_usage(128);
/* dither for depths < 24bpp */
imlib_context_set_dither(1);
/* set the display , visual, colormap and drawable we are using */
imlib_context_set_display(disp);
imlib_context_set_visual(vis);
imlib_context_set_colormap(cm);
imlib_context_set_drawable(win);
/* infinite event loop */
for (;;)
  {
     /* image variable */
     Imlib_Image image;
     /* width and height values */
     int w, h, text_w, text_h;

     /* init our updates to empty */
     updates = imlib_updates_init();
     /* while there are events form X - handle them */
     do
       {
          XNextEvent(disp, &ev);
          switch (ev.type)
            {
            case Expose:
               /* window rectangle was exposed - add it to the list of */
               /* rectangles we need to re-render */
               updates = imlib_update_append_rect(updates,
                                                  ev.xexpose.x, ev.xexpose.y,
                                                  ev.xexpose.width, ev.xexpose.height);
               break;
            case ButtonPress:
               /* if we click anywhere in the window, exit */
               exit(0);
               break;
            case MotionNotify:
               /* if the mouse moves - note it */
               /* add a rectangle update for the new mouse position */
               image = imlib_load_image("./test_images/mush.png");
               imlib_context_set_image(image);
               w = imlib_image_get_width();
               h = imlib_image_get_height();
               imlib_context_set_image(image);
               imlib_free_image();
               /* the old position - so we wipe over where it used to be */
               updates = imlib_update_append_rect(updates,
                                                  mouse_x - (w / 2), mouse_y - (h / 2),
```

```
                                        w, h);
            font = imlib_load_font("notepad/30");
            if (font)
              {
                 char text[4096];

                 imlib_context_set_font(font);
                 sprintf(text, "Mouse is at %i, %i", mouse_x, mouse_y);
                 imlib_get_text_size(text, &text_w, &text_h);
                 imlib_free_font();
                 updates = imlib_update_append_rect(updates,
                                                    320 - (text_w / 2), 240 - (text_h / 2),
                                                    text_w, text_h);
              }

            mouse_x = ev.xmotion.x;
            mouse_y = ev.xmotion.y;
            /* the new one */
            updates = imlib_update_append_rect(updates,
                                        mouse_x - (w / 2), mouse_y - (h / 2),
                                        w, h);
            font = imlib_load_font("notepad/30");
            if (font)
              {
                 char text[4096];

                 imlib_context_set_font(font);
                 sprintf(text, "Mouse is at %i, %i", mouse_x, mouse_y);
                 imlib_get_text_size(text, &text_w, &text_h);
                 imlib_free_font();
                 updates = imlib_update_append_rect(updates,
                                                    320 - (text_w / 2), 240 - (text_h / 2),
                                                    text_w, text_h);
              }
          default:
             /* any other events - do nothing */
             break;
          }
      }
    while (XPending(disp));

    /* no more events for now ? ok - idle time so lets draw stuff */

    /* take all the little rectangles to redraw and merge them into */
    /* something sane for rendering */
    updates = imlib_updates_merge_for_rendering(updates, 640, 480);
    for (current_update = updates;
         current_update;
         current_update = imlib_updates_get_next(current_update))
      {
         int up_x, up_y, up_w, up_h;

         /* find out where the first update is */
         imlib_updates_get_coordinates(current_update,
                                       &up_x, &up_y, &up_w, &up_h);

         /* create our buffer image for rendering this update */
         buffer = imlib_create_image(up_w, up_h);

         /* we can blend stuff now */
         imlib_context_set_blend(1);

         /* fill the window background */
         /* load the background image - you'll need to have some images */
         /* in ./test_images lying around for this to actually work */
         image = imlib_load_image("./test_images/bg.png");
         /* we're working with this image now */
```

```
imlib_context_set_image(image);
/* get its size */
w = imlib_image_get_width();
h = imlib_image_get_height();
/* now we want to work with the buffer */
imlib_context_set_image(buffer);
/* if the iimage loaded */
if (image)
  {
     /* blend image onto the buffer and scale it to 640x480 */
     imlib_blend_image_onto_image(image, 0,
                                  0, 0, w, h,
                                  - up_x, - up_y, 640, 480);
     /* working with the loaded image */
     imlib_context_set_image(image);
     /* free it */
     imlib_free_image();
  }

/* draw an icon centered around the mouse position */
image = imlib_load_image("./test_images/mush.png");
imlib_context_set_image(image);
w = imlib_image_get_width();
h = imlib_image_get_height();
imlib_context_set_image(buffer);
if (image)
  {
     imlib_blend_image_onto_image(image, 0,
                                  0, 0, w, h,
                                  mouse_x - (w / 2) - up_x, mouse_y - (h / 2) - up_y, w, h);
     imlib_context_set_image(image);
     imlib_free_image();
  }

/* draw a gradient on top of things at the top left of the window */
/* create a range */
range = imlib_create_color_range();
imlib_context_set_color_range(range);
/* add white opaque as the first color */
imlib_context_set_color(255, 255, 255, 255);
imlib_add_color_to_color_range(0);
/* add an orange color, semi-transparent 10 units from the first */
imlib_context_set_color(255, 200, 10, 100);
imlib_add_color_to_color_range(10);
/* add black, fully transparent at the end 20 units away */
imlib_context_set_color(0, 0, 0, 0);
imlib_add_color_to_color_range(20);
/* draw the range */
imlib_context_set_image(buffer);
imlib_image_fill_color_range_rectangle(- up_x, - up_y, 128, 128, -45.0);
/* free it */
imlib_free_color_range();

/* draw text - centered with the current mouse x, y */
font = imlib_load_font("notepad/30");
if (font)
  {
     char text[4096];

     /* set the current font */
     imlib_context_set_font(font);
     /* set the image */
     imlib_context_set_image(buffer);
     /* set the color (black) */
     imlib_context_set_color(0, 0, 0, 255);
     /* print text to display in the buffer */
     sprintf(text, "Mouse is at %i, %i", mouse_x, mouse_y);
```

```
                /* query the size it will be */
                imlib_get_text_size(text, &text_w, &text_h);
                /* draw it */
                imlib_text_draw(320 - (text_w / 2) - up_x, 240 - (text_h / 2) - up_y, text);
                /* free the font */
                imlib_free_font();
              }

            /* don't blend the image onto the drawable - slower */
            imlib_context_set_blend(0);
            /* set the buffer image as our current image */
            imlib_context_set_image(buffer);
            /* render the image at 0, 0 */
            imlib_render_image_on_drawable(up_x, up_y);
            /* don't need that temporary buffer image anymore */
            imlib_free_image();
          }
        /* if we had updates - free them */
        if (updates)
           imlib_updates_free(updates);
        /* loop again waiting for events */
     }
   return 0;
}
```

1.5 Imlib2 filters

1.5.1 Imlib 2 Dynamic Filters

Imlib2 has built in features allowing filters and effects to be applied at run time through a very small scripting language, this is similar to that of script-fu found in the GIMP (http://www.gimp.org). There are two parts to the system, the client library call "imlib_apply_filter" and the library side filters. The library side filters are synonymous with image loaders.

To run a script on an image you need to set the context image then call:

```
imlib_apply_filter( script_string, ... );
```

The script_string variable is made up of the script language, which is very simple and made up of only function calls. Functions calls look like this:

```
filter name( key=value [, ...] );
```

Where,

- **filter name** is the name of the filter you wish to apply
- **key** is an expected value
- **value** is a "string", a number, or an actual variable in

the program, or the result of another filter.

eg.

```
bump_map( map=tint(red=50,tint=200), blue=10 );
```

This example would bump map using a a map generated from the tint filter.

It is also possible to pass application information to the filters via the usage of the [] operator. When the script is being compiled the script engine looks on the parameters passed to it and picks up a pointer for every [] found.

eg2.

```
imlib_apply_filter( "tint( x=[], y=[], red=255, alpha=55 );", &myxint, &myyint );
```

This will cause a tint to the current image at (myxint,myyint) to be done. This is very useful for when you want the filters to dynamically change according to program variables. The system is very quick as the code is pseudo-compiled and then run. The advantage of having the scripting system allows customization of the image manipulations, this is particularly useful in applications that allow modifications to be done (eg. image viewers).

1.5.2 Filter Library

There are three functions that must be in every filter library

```
void init( struct imlib_filter_info *info ); - Called once on loading of the filter
```

info - a structure passed to the filter to be filled in with information about the filter info->name - Name of the filter library info->author - Name of the library author info->description - Description of the filter library info->num_filters - Number of filters the library exports info->filters - An array of "char *" with each filter name in it.

```
void deinit(); - Called when the filter is closed

/* Called every time a filter the library exports is called */
void *exec( char *filter, void *im, pIFunctionParam params );
```

filter - The name of the filter being asked for im - The image that the filter should be applied against params - A linked list of parameters.

The best way to get all the values is such:

Declare all parameters and initialize them to there default values.

```
for( ptr = params; ptr != NULL; ptr = ptr->next )
  {
    ..MACRO TO GET VALUE..
  }
```

Current Macros are:

```
 ASSIGN_INT( keyname, local variable )
 ASSIGN_DATA8( keyname, local variable )
 ASSIGN_IMAGE( keyname, local variable )
```

eg.

```
int r = 50;
IFunctionParam *ptr;

for( ptr = params; ptr != NULL; ptr = ptr->next )
  {
    ASSIGN_INT( "red", r );
  }
```

If the value "red" is not passed to the filter then it will remain at 50, but it a value is passed, it will be assign to r.

return type - Imlib_Image, this is the result of filter.

Todo

line code doesnt draw nice liens when clipping - fix

Todo

filled polygons can break fill bounds on corner cases - fix

Todo

go thru TODOs and FIXMEs

Chapter 2

Imlib2 File Index

2.1 Imlib2 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Imlib2 Page Index

3.1 Imlib2 Related Pages

Here is a list of all related documentation pages:

Chapter 4

Imlib2 File Documentation

4.1 imlib2.c File Reference

Imlib2 library.

Functions

- EAPI void imlib_context_set_cliprect (int x, int y, int w, int h)

  Parameters:

  x *The top left x coordinate of the rectangle.*

- EAPI void imlib_context_set_dither_mask (char dither_mask)

  Parameters:

  dither_mask *The dither mask flag.*

- EAPI char imlib_context_get_dither_mask (void)

  Returns:

  The current dither mask flag.

- EAPI void imlib_context_set_mask_alpha_threshold (int mask_alpha_threshold)

  Parameters:

  mask_alpha_threshold *The mask alpha threshold.*

- EAPI int imlib_context_get_mask_alpha_threshold (void)

  Returns:

  The current mask mask alpha threshold.

- EAPI void imlib_context_set_anti_alias (char anti_alias)

  Parameters:

  anti_alias *The anti alias flag.*

- EAPI char imlib_context_get_anti_alias (void)
  Returns:
  The current anti alias flag.

- EAPI void imlib_context_set_dither (char dither)
  Parameters:
  dither *The dithering flag.*

- EAPI char imlib_context_get_dither (void)
  Returns:
  The current dithering flag.

- EAPI void imlib_context_set_blend (char blend)
  Parameters:
  blend *The blending flag.*

- EAPI char imlib_context_get_blend (void)
  Returns:
  The current blending flag.

- EAPI void imlib_context_set_color_modifier (Imlib_Color_Modifier color_modifier)
  Parameters:
  color_modifier *Current color modifier.*

- EAPI Imlib_Color_Modifier imlib_context_get_color_modifier (void)
  Returns:
  The current color modifier.

- EAPI void imlib_context_set_operation (Imlib_Operation operation)
  Parameters:
  operation

- EAPI Imlib_Operation imlib_context_get_operation (void)
  Returns:
  The current operation mode.

- EAPI void imlib_context_set_font (Imlib_Font font)
  Parameters:
  font *Current font.*

- EAPI Imlib_Font imlib_context_get_font (void)
  Returns:
  The current font.

- EAPI void imlib_context_set_direction (Imlib_Text_Direction direction)
  Parameters:
  direction *Text direction.*

- EAPI void imlib_context_set_angle (double angle)

  Parameters:

  angle *Angle of the text strings.*

- EAPI double imlib_context_get_angle (void)

  Returns:

  The current angle of the text strings.

- EAPI Imlib_Text_Direction imlib_context_get_direction (void)

  Returns:

  The current direction of the text.

- EAPI void imlib_context_set_color (int red, int green, int blue, int alpha)

  Parameters:

  red *Red channel of the current color.*

- EAPI void imlib_context_get_color (int *red, int *green, int *blue, int *alpha)

  Parameters:

  red *Red channel of the current color.*

- EAPI Imlib_Color * imlib_context_get_imlib_color (void)

  Returns:

  The current color.

- EAPI void imlib_context_set_color_hsva (float hue, float saturation, float value, int alpha)

  Parameters:

  hue *Hue channel of the current color.*

- EAPI void imlib_context_get_color_hsva (float *hue, float *saturation, float *value, int *alpha)

  Parameters:

  hue *Hue channel of the current color.*

- EAPI void imlib_context_set_color_hlsa (float hue, float lightness, float saturation, int alpha)

  Parameters:

  hue *Hue channel of the current color.*

- EAPI void imlib_context_get_color_hlsa (float *hue, float *lightness, float *saturation, int *alpha)

  Parameters:

  hue *Hue channel of the current color.*

- EAPI void imlib_context_set_color_cmya (int cyan, int magenta, int yellow, int alpha)

  Parameters:

  cyan *Cyan channel of the current color.*

- EAPI void imlib_context_get_color_cmya (int *cyan, int *magenta, int *yellow, int *alpha)

  Parameters:

  cyan *Cyan channel of the current color.*

- EAPI void imlib_context_set_color_range (Imlib_Color_Range color_range)

  Parameters:

  color_range *Color range.*

- EAPI Imlib_Color_Range imlib_context_get_color_range (void)

  Returns:

  The current color range.

- EAPI void imlib_context_set_progress_function (Imlib_Progress_Function progress_function)

  Parameters:

  progress_function *A progress function.*

- EAPI Imlib_Progress_Function imlib_context_get_progress_function (void)

  Returns:

  The current progress function.

- EAPI void imlib_context_set_progress_granularity (char progress_granularity)

  Parameters:

  progress_granularity *A char.*

- EAPI char imlib_context_get_progress_granularity (void)

  Returns:

  The current progress granularity

- EAPI void imlib_context_set_image (Imlib_Image image)

  Parameters:

  image *Current image.*

- EAPI Imlib_Image imlib_context_get_image (void)

  Returns:

  The current image.

- EAPI int imlib_get_cache_size (void)

  Returns:

  The current image size.

- EAPI void imlib_set_cache_size (int bytes)

  Parameters:

  bytes *Cache size.*

- EAPI int imlib_get_color_usage (void)

Returns:

The current number of colors.

- EAPI void imlib_set_color_usage (int max)

  Parameters:

  max *Maximum number of colors.*

- EAPI void imlib_flush_loaders (void)

  If you want Imlib2 to forcibly flush any cached loaders it has and re-load them from disk (this is useful if the program just installed a new loader and does not want to wait till Imlib2 deems it an optimal time to rescan the loaders).

- EAPI Imlib_Image imlib_load_image (const char *file)

  Parameters:

  file *Image file.*

- EAPI Imlib_Image imlib_load_image_immediately (const char *file)

  Parameters:

  file *Image file.*

- EAPI Imlib_Image imlib_load_image_without_cache (const char *file)

  Parameters:

  file *Image file.*

- EAPI Imlib_Image imlib_load_image_immediately_without_cache (const char *file)

  Parameters:

  file *Image file.*

- EAPI Imlib_Image imlib_load_image_with_error_return (const char *file, Imlib_Load_Error *error_return)

  Parameters:

  file *Image file.*

- EAPI void imlib_free_image (void)

  Frees the image that is set as the current image in Imlib2's context.

- EAPI void imlib_free_image_and_decache (void)

  Frees the current image in Imlib2's context AND removes it from the cache.

- EAPI int imlib_image_get_width (void)

  Returns the width in pixels of the current image in Imlib2's context.

- EAPI int imlib_image_get_height (void)

  Returns the height in pixels of the current image in Imlib2's context.

- EAPI const char * imlib_image_get_filename (void)

  Returns:

  The current filename.

- EAPI DATA32 * imlib_image_get_data (void)

  Returns:

  A pointer to the image data.

- EAPI DATA32 * imlib_image_get_data_for_reading_only (void)

  Returns:

  A pointer to the image data.

- EAPI void imlib_image_put_back_data (DATA32 *data)

  Parameters:

  data *The pointer to the image data.*

- EAPI char imlib_image_has_alpha (void)

  Returns:

  Current alpha channel flag.

- EAPI void imlib_image_set_changes_on_disk (void)

  By default Imlib2 will not check the timestamp of an image on disk and compare it with the image in its cache - this is to minimize disk activity when using the cache.

- EAPI void imlib_image_get_border (Imlib_Border *border)

  Parameters:

  border *The border of the image.*

- EAPI void imlib_image_set_border (Imlib_Border *border)

  Parameters:

  border *The border of the image.*

- EAPI void imlib_image_set_format (const char *format)

  Parameters:

  format *Format of the image.*

- EAPI void imlib_image_set_irrelevant_format (char irrelevant)

  Parameters:

  irrelevant *Irrelevant format flag.*

- EAPI void imlib_image_set_irrelevant_border (char irrelevant)

  Parameters:

  irrelevant *Irrelevant border flag.*

- EAPI void imlib_image_set_irrelevant_alpha (char irrelevant)

  Parameters:

  irrelevant *Irrelevant alpha flag.*

- EAPI char * imlib_image_format (void)

  Returns:

  Current image format.

- EAPI void imlib_image_set_has_alpha (char has_alpha)

  Parameters:

  has_alpha *Alpha flag.*

- EAPI void imlib_blend_image_onto_image (Imlib_Image source_image, char merge_alpha, int source_x, int source_y, int source_width, int source_height, int destination_x, int destination_y, int destination_width, int destination_height)

  Parameters:

  source_image *The source image.*

- EAPI Imlib_Image imlib_create_image (int width, int height)

  Parameters:

  width *The width of the image.*

- EAPI Imlib_Image imlib_create_image_using_data (int width, int height, DATA32 *data)

  Parameters:

  width *The width of the image.*

- EAPI Imlib_Image imlib_create_image_using_copied_data (int width, int height, DATA32 *data)

  Parameters:

  width *The width of the image.*

- EAPI Imlib_Image imlib_clone_image (void)

  Returns:

  A valid image, otherwise NULL.

- EAPI Imlib_Image imlib_create_cropped_image (int x, int y, int width, int height)

  Parameters:

  x *The top left x coordinate of the rectangle.*

- EAPI Imlib_Image imlib_create_cropped_scaled_image (int source_x, int source_y, int source_width, int source_height, int destination_width, int destination_height)

  Parameters:

  source_x *The top left x coordinate of the source rectangle.*

- EAPI Imlib_Updates imlib_updates_clone (Imlib_Updates updates)

  Parameters:

  updates *An updates list.*

- EAPI Imlib_Updates imlib_update_append_rect (Imlib_Updates updates, int x, int y, int w, int h)

  Parameters:

  updates *An updates list.*

- EAPI Imlib_Updates imlib_updates_merge (Imlib_Updates updates, int w, int h)

  Parameters:

  updates *An updates list.*

- EAPI Imlib_Updates imlib_updates_merge_for_rendering (Imlib_Updates updates, int w, int h)

  Parameters:

  updates *An updates list.*

- EAPI void imlib_updates_free (Imlib_Updates updates)

  Parameters:

  updates *An updates list.*

- EAPI Imlib_Updates imlib_updates_get_next (Imlib_Updates updates)

  Parameters:

  updates *An updates list.*

- EAPI void imlib_updates_get_coordinates (Imlib_Updates updates, int *x_return, int *y_return, int *width_return, int *height_return)

  Parameters:

  updates *An updates list.*

- EAPI void imlib_updates_set_coordinates (Imlib_Updates updates, int x, int y, int width, int height)

  Parameters:

  updates *An updates list.*

- EAPI Imlib_Updates imlib_updates_init (void)

  Returns:

  The initialized updates list.

- EAPI Imlib_Updates imlib_updates_append_updates (Imlib_Updates updates, Imlib_Updates appended_updates)

  Parameters:

  updates *An updates list.*

- EAPI void imlib_image_flip_horizontal (void)

  Flips/mirrors the current image horizontally.

- EAPI void imlib_image_flip_vertical (void)

  Flips/mirrors the current image vertically.

- EAPI void imlib_image_flip_diagonal (void)

  Flips/mirrors the current image diagonally (good for quick and dirty 90 degree rotations if used before to after a horizontal or vertical flip).

- EAPI void imlib_image_orientate (int orientation)

  Parameters:

  orientation *The orientation.*

- EAPI void imlib_image_blur (int radius)

Parameters:

radius *The radius.*

- EAPI void imlib_image_sharpen (int radius)

  Parameters:

  radius *The radius.*

- EAPI void imlib_image_tile_horizontal (void)

  Modifies an image so it will tile seamlessly horizontally if used as a tile (i.e.

- EAPI void imlib_image_tile_vertical (void)

  Modifies an image so it will tile seamlessly vertically if used as a tile (i.e.

- EAPI void imlib_image_tile (void)

  Modifies an image so it will tile seamlessly horizontally and vertically if used as a tile (i.e.

- EAPI Imlib_Font imlib_load_font (const char *font_name)

  Parameters:

  font_name *The font name with the size.*

- EAPI void imlib_free_font (void)

  Frees the current font.

- EAPI void imlib_text_draw (int x, int y, const char *text)

  Parameters:

  x *The x coordinate of the top left corner.*

- EAPI void imlib_text_draw_with_return_metrics (int x, int y, const char *text, int *width_return, int *height_return, int *horizontal_advance_return, int *vertical_advance_return)

  Parameters:

  x *The x coordinate of the top left corner.*

- EAPI void imlib_get_text_size (const char *text, int *width_return, int *height_return)

  Parameters:

  text *A string.*

- EAPI void imlib_get_text_advance (const char *text, int *horizontal_advance_return, int *vertical_advance_return)

  Parameters:

  text *A string.*

- EAPI int imlib_get_text_inset (const char *text)

  Parameters:

  text *A string.*

- EAPI void imlib_add_path_to_font_path (const char *path)

  Parameters:

  path *A directory path.*

- EAPI void imlib_remove_path_from_font_path (const char *path)

  Parameters:

  path *A directory path.*

- EAPI char ** imlib_list_font_path (int *number_return)

  Parameters:

  number_return *Number of paths in the list.*

- EAPI int imlib_text_get_index_and_location (const char *text, int x, int y, int *char_x_return, int *char_y_return, int *char_width_return, int *char_height_return)

  Parameters:

  text *A string.*

- EAPI void imlib_text_get_location_at_index (const char *text, int index, int *char_x_return, int *char_y_return, int *char_width_return, int *char_height_return)

  Parameters:

  text *A string.*

- EAPI char ** imlib_list_fonts (int *number_return)

  Parameters:

  number_return *Number of fonts in the list.*

- EAPI void imlib_free_font_list (char **font_list, int number)

  Parameters:

  font_list *The font list.*

- EAPI int imlib_get_font_cache_size (void)

  Returns:

  The font cache size.

- EAPI void imlib_set_font_cache_size (int bytes)

  Parameters:

  bytes *The font cache size.*

- EAPI void imlib_flush_font_cache (void)

  Causes a flush of all speculatively cached fonts from the font cache.

- EAPI int imlib_get_font_ascent (void)

  Returns:

  The font's ascent.

- EAPI int imlib_get_font_descent (void)

  Returns:

  The font's descent.

- EAPI int imlib_get_maximum_font_ascent (void)

Returns:

The font's maximum ascent.

- EAPI int imlib_get_maximum_font_descent (void)

  Returns:

  The font's maximum descent.

- EAPI Imlib_Color_Modifier imlib_create_color_modifier (void)

  Returns:

  Valid handle.

- EAPI void imlib_free_color_modifier (void)

  Frees the current color modifier.

- EAPI void imlib_modify_color_modifier_gamma (double gamma_value)

  Parameters:

  gamma_value *Value of gamma.*

- EAPI void imlib_modify_color_modifier_brightness (double brightness_value)

  Parameters:

  brightness_value *Value of brightness.*

- EAPI void imlib_modify_color_modifier_contrast (double contrast_value)

  Parameters:

  contrast_value *Value of contrast.*

- EAPI void imlib_set_color_modifier_tables (DATA8 *red_table, DATA8 *green_table, DATA8 *blue_table, DATA8 *alpha_table)

  Parameters:

  red_table *An array of DATA8.*

- EAPI void imlib_get_color_modifier_tables (DATA8 *red_table, DATA8 *green_table, DATA8 *blue_table, DATA8 *alpha_table)

  Parameters:

  red_table,: *an array of DATA8.*

- EAPI void imlib_reset_color_modifier (void)

  Resets the current color modifier to have linear mapping tables.

- EAPI void imlib_apply_color_modifier (void)

  Uses the current color modifier and modifies the current image using the mapping tables in the current color modifier.

- EAPI void imlib_apply_color_modifier_to_rectangle (int x, int y, int width, int height)

  Parameters:

  x *The x coordinate of the left edge of the rectangle.*

- EAPI Imlib_Updates imlib_image_draw_line (int x1, int y1, int x2, int y2, char make_updates)

Parameters:

x1 *The x coordinate of the first point.*

- EAPI void imlib_image_draw_rectangle (int x, int y, int width, int height)

  Parameters:

  x *The top left x coordinate of the rectangle.*

- EAPI void imlib_image_fill_rectangle (int x, int y, int width, int height)

  Parameters:

  x *The top left x coordinate of the rectangle.*

- EAPI void imlib_image_copy_alpha_to_image (Imlib_Image image_source, int x, int y)

  Parameters:

  image_source *An image.*

- EAPI void imlib_image_copy_alpha_rectangle_to_image (Imlib_Image image_source, int x, int y, int width, int height, int destination_x, int destination_y)

  Parameters:

  image_source *An image.*

- EAPI void imlib_image_scroll_rect (int x, int y, int width, int height, int delta_x, int delta_y)

  Parameters:

  x *The top left x coordinate of the rectangle.*

- EAPI void imlib_image_copy_rect (int x, int y, int width, int height, int new_x, int new_y)

  Parameters:

  x *The top left x coordinate of the rectangle.*

- EAPI Imlib_Color_Range imlib_create_color_range (void)

  Returns:

  valid handle.

- EAPI void imlib_free_color_range (void)

  Frees the current color range.

- EAPI void imlib_add_color_to_color_range (int distance_away)

  Parameters:

  distance_away *Distance from the previous color.*

- EAPI void imlib_image_fill_color_range_rectangle (int x, int y, int width, int height, double angle)

  Parameters:

  x *The x coordinate of the left edge of the rectangle.*

- EAPI void imlib_image_fill_hsva_color_range_rectangle (int x, int y, int width, int height, double angle)

Parameters:

x *The x coordinate of the left edge of the rectangle.*

- EAPI void imlib_image_query_pixel (int x, int y, Imlib_Color *color_return)

  Parameters:

  x *The x coordinate of the pixel.*

- EAPI void imlib_image_query_pixel_hsva (int x, int y, float *hue, float *saturation, float *value, int *alpha)

  Parameters:

  x *The x coordinate of the pixel.*

- EAPI void imlib_image_query_pixel_hlsa (int x, int y, float *hue, float *lightness, float *saturation, int *alpha)

  Parameters:

  x *The x coordinate of the pixel.*

- EAPI void imlib_image_query_pixel_cmya (int x, int y, int *cyan, int *magenta, int *yellow, int *alpha)

  Parameters:

  x *Tthe x coordinate of the pixel.*

- EAPI void imlib_image_attach_data_value (const char *key, void *data, int value, Imlib_Internal_Data_Destructor_Function destructor_function)

  Parameters:

  key *A string.*

- EAPI void * imlib_image_get_attached_data (const char *key)

  Parameters:

  key *A string.*

- EAPI int imlib_image_get_attached_value (const char *key)

  Parameters:

  key *A string.*

- EAPI void imlib_image_remove_attached_data_value (const char *key)

  Parameters:

  key *A string.*

- EAPI void imlib_image_remove_and_free_attached_data_value (const char *key)

  Parameters:

  key *A string.*

- EAPI void imlib_save_image (const char *filename)

  Parameters:

  filename *The file name.*

- EAPI void imlib_save_image_with_error_return (const char *filename, Imlib_Load_Error *error_return)

Parameters:

filename *The file name.*

- EAPI Imlib_Image imlib_create_rotated_image (double angle)

  Parameters:

  angle *An angle in radians.*

- EAPI void imlib_blend_image_onto_image_at_angle (Imlib_Image source_image, char merge_alpha, int source_x, int source_y, int source_width, int source_height, int destination_x, int destination_y, int angle_x, int angle_y)

  Parameters:

  source_image *The image source.*

- EAPI void imlib_blend_image_onto_image_skewed (Imlib_Image source_image, char merge_alpha, int source_x, int source_y, int source_width, int source_height, int destination_x, int destination_y, int h_angle_x, int h_angle_y, int v_angle_x, int v_angle_y)

  Parameters:

  source_image *The source image.*

- EAPI void imlib_context_set_filter (Imlib_Filter filter)

  Parameters:

  filter *Current filter.*

- EAPI Imlib_Filter imlib_context_get_filter (void)

  Returns:

- EAPI ImlibPolygon imlib_polygon_new (void)

  Returns a new polygon object with no points set.

- EAPI void imlib_polygon_add_point (ImlibPolygon poly, int x, int y)

  Parameters:

  poly *A polygon*

- EAPI void imlib_polygon_free (ImlibPolygon poly)

  Parameters:

  poly *A polygon.*

- EAPI void imlib_image_draw_polygon (ImlibPolygon poly, unsigned char closed)

  Parameters:

  poly *A polygon.*

- EAPI void imlib_image_fill_polygon (ImlibPolygon poly)

  Parameters:

  poly *A polygon.*

- EAPI void imlib_polygon_get_bounds (ImlibPolygon poly, int *px1, int *py1, int *px2, int *py2)

Parameters:

poly *A polygon.*

- EAPI void imlib_image_draw_ellipse (int xc, int yc, int a, int b)

  Parameters:

  xc *X coordinate of the center of the ellipse.*

- EAPI void imlib_image_fill_ellipse (int xc, int yc, int a, int b)

  Parameters:

  xc *X coordinate of the center of the ellipse.*

- EAPI unsigned char imlib_polygon_contains_point (ImlibPolygon poly, int x, int y)

  Parameters:

  poly *A polygon*

4.1.1 Detailed Description

Imlib2 library.

Brief of imlib2 library

4.1.2 Function Documentation

4.1.2.1 EAPI void imlib_add_color_to_color_range (int *distance_away*)

Parameters:

distance_away Distance from the previous color.

Adds the current color to the current color range at a `distance_away` distance from the previous color in the range (if it's the first color in the range this is irrelevant).

4.1.2.2 EAPI void imlib_add_path_to_font_path (const char * *path*)

Parameters:

path A directory path.

Adds the directory `path` to the end of the current list of directories to scan for fonts.

4.1.2.3 EAPI void imlib_apply_color_modifier_to_rectangle (int *x*, int *y*, int *width*, int *height*)

Parameters:

x The x coordinate of the left edge of the rectangle.

Parameters:

y The y coordinate of the top edge of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

Works the same way as imlib_apply_color_modifier() but only modifies a selected rectangle in the current image.

4.1.2.4 EAPI void imlib_blend_image_onto_image (Imlib_Image *source_image*, char *merge_alpha*, int *source_x*, int *source_y*, int *source_width*, int *source_height*, int *destination_x*, int *destination_y*, int *destination_width*, int *destination_height*)

Parameters:

source_image The source image.

Parameters:

merge_alpha Alpha flag.

source_x X coordinate of the source image.

source_y Y coordinate of the source image.

source_width Width of the source image.

source_height Height of the source image.

destination_x X coordinate of the destination image.

destination_y Y coordinate of the destination image.

destination_width Width of the destination image.

destination_height Height of the destination image.

Blends the source rectangle (`source_x`, `source_y`, `source_width`, `source_height`) from `source_image` onto the current image at the destination (`destination_x`, `destination_y`) location scaled to the width `destination_width` and height `destination_height`. If `merge_alpha` is set to 1 it will also modify the destination image alpha channel, otherwise the destination alpha channel is left untouched.

4.1.2.5 EAPI void imlib_blend_image_onto_image_at_angle (Imlib_Image *source_image*, char *merge_alpha*, int *source_x*, int *source_y*, int *source_width*, int *source_height*, int *destination_x*, int *destination_y*, int *angle_x*, int *angle_y*)

Parameters:

source_image The image source.

Parameters:

merge_alpha A char.

source_x The source x coordinate.

source_y The source y coordinate.

source_width The source width.

source_height The source height.

destination_x The destination x coordinate.

destination_y The destination y coordinate.

angle_x An angle.

angle_y An angle.

Works just like imlib_blend_image_onto_image_skewed() except you cannot skew an image (`v_angle_x` and `v_angle_y` are 0).

4.1.2.6 EAPI void imlib_blend_image_onto_image_skewed (Imlib_Image *source_image*, char *merge_alpha*, int *source_x*, int *source_y*, int *source_width*, int *source_height*, int *destination_x*, int *destination_y*, int *h_angle_x*, int *h_angle_y*, int *v_angle_x*, int *v_angle_y*)

Parameters:

source_image The source image.

Parameters:

merge_alpha A char

source_x The source x coordinate.

source_y The source y coordinate.

source_width The source width.

source_height The source height.

destination_x The destination x coordinate.

destination_y The destination y coordinate.

h_angle_x An angle.

h_angle_y An angle.

v_angle_x An angle.

v_angle_y An angle.

Blends the source rectangle (`source_x`, `source_y`, `source_width`, `source_height`) from the `source_image` onto the current image at the destination (`destination_x`, `destination_y`) location. It will be rotated and scaled so that the upper right corner will be positioned `h_angle_x` pixels to the right (or left, if negative) and `h_angle_y` pixels down (from (`destination_x`, `destination_y`). If `v_angle_x` and `v_angle_y` are not 0, the image will also be skewed so that the lower left corner will be positioned `v_angle_x` pixels to the right and `v_angle_y` pixels down. The at_angle versions simply have the `v_angle_x` and `v_angle_y` set to 0 so the rotation doesn't get skewed, and the render_..._on_drawable ones seem obvious enough; they do the same on a drawable.

Example:

```
imlib_blend_image_onto_image_skewed(..., 0, 0, 100, 0, 0, 100);
```

will simply scale the image to be 100x100.

```
imlib_blend_image_onto_image_skewed(..., 0, 0, 0, 100, 100, 0);
```

will scale the image to be 100x100, and flip it diagonally.

```
imlib_blend_image_onto_image_skewed(..., 100, 0, 0, 100, -100, 0);
```

will scale the image and rotate it 90 degrees clockwise.

```
imlib_blend_image_onto_image_skewed(..., 50, 0, 50, 50, -50, 50);
```

will rotate the image 45 degrees clockwise, and will scale it so its corners are at (50,0)-(100,50)-(50,100)-(0,50) i.e. it fits into the 100x100 square, so it's scaled down to 70.7% (sqrt(2)/2).

```
imlib_blend_image_onto_image_skewed(..., 50, 50, 100 * cos(a), 100 * sin(a), 0);
```

will rotate the image 'a' degrees, with its upper left corner at (50,50).

4.1.2.7 EAPI Imlib_Image imlib_clone_image (void)

Returns:

A valid image, otherwise NULL.

Creates an exact duplicate of the current image and returns a valid image handle on success, or NULL on failure.

4.1.2.8 EAPI double imlib_context_get_angle (void)

Returns:

The current angle of the text strings.

Returns the current angle used to render text at if the direction is IMLIB_TEXT_TO_ANGLE.

4.1.2.9 EAPI char imlib_context_get_anti_alias (void)

Returns:

The current anti alias flag.

Returns if Imlib2 currently will smoothly scale images. 1 means it will and 0 means it will not.

4.1.2.10 EAPI char imlib_context_get_blend (void)

Returns:

The current blending flag.

Returns if Imlib2 will blend images onto a drawable whilst rendering to that drawable. 1 means yes and 0 means no.

4.1.2.11 EAPI void imlib_context_get_color (int * *red*, int * *green*, int * *blue*, int * *alpha*)

Parameters:

red Red channel of the current color.

Parameters:

green Green channel of the current color.

blue Blue channel of the current color.

alpha Alpha channel of the current color.

Returns the current color for rendering text, rectangles and lines.

4.1.2.12 EAPI void imlib_context_get_color_cmya (int * *cyan*, int * *magenta*, int * *yellow*, int * *alpha*)

Parameters:

cyan Cyan channel of the current color.

Parameters:

magenta Magenta channel of the current color.

yellow Yellow channel of the current color.

alpha Alpha channel of the current color.

Returns the current color for rendering text, rectangles and lines in CMYA space.

4.1.2.13 EAPI void imlib_context_get_color_hlsa (float * *hue*, float * *lightness*, float * *saturation*, int * *alpha*)

Parameters:

hue Hue channel of the current color.

Parameters:

lightness Lightness channel of the current color.

saturation Saturation channel of the current color.

alpha Alpha channel of the current color.

Returns the current color for rendering text, rectangles and lines in HLSA space.

4.1.2.14 EAPI void imlib_context_get_color_hsva (float * *hue*, float * *saturation*, float * *value*, int * *alpha*)

Parameters:

hue Hue channel of the current color.

Parameters:

saturation Saturation channel of the current color.

value Value channel of the current color.

alpha Alpha channel of the current color.

Returns the current color for rendering text, rectangles and lines in HSVA space.

4.1.2.15 EAPI Imlib_Color_Modifier imlib_context_get_color_modifier (void)

Returns:

The current color modifier.

Returns the current color modifier being used.

4.1.2.16 EAPI Imlib_Color_Range imlib_context_get_color_range (void)

Returns:

The current color range.

Returns the current color range being used for gradients.

4.1.2.17 EAPI Imlib_Text_Direction imlib_context_get_direction (void)

Returns:

The current direction of the text.

Returns the current direction to render text in.

4.1.2.18 EAPI char imlib_context_get_dither (void)

Returns:

The current dithering flag.

Returns if image data is rendered with dithering currently. 1 means yes and 0 means no.

4.1.2.19 EAPI char imlib_context_get_dither_mask (void)

Returns:

The current dither mask flag.

Returns the current mode for dithering pixmap masks. 1 means dithering is enabled and 0 means it is not.

4.1.2.20 EAPI Imlib_Filter imlib_context_get_filter (void)

Returns:

Gets the current context image filter.

4.1.2.21 EAPI Imlib_Font imlib_context_get_font (void)

Returns:

The current font.

Returns the current font.

4.1.2.22 EAPI Imlib_Image imlib_context_get_image (void)

Returns:

The current image.

Returns the current context image.

4.1.2.23 EAPI Imlib_Color* imlib_context_get_imlib_color (void)

Returns:

The current color.

Returns the current color as a color struct. Do NOT free this pointer.

4.1.2.24 EAPI int imlib_context_get_mask_alpha_threshold (void)

Returns:

The current mask mask alpha threshold.

Returns the current mask alpha threshold.

4.1.2.25 EAPI Imlib_Operation imlib_context_get_operation (void)

Returns:

The current operation mode.

Returns the current operation mode.

4.1.2.26 EAPI Imlib_Progress_Function imlib_context_get_progress_function (void)

Returns:

The current progress function.

Returns the current progress function being used.

4.1.2.27 EAPI char imlib_context_get_progress_granularity (void)

Returns:

The current progress granularity

Returns the current progress granularity being used.

4.1.2.28 EAPI void imlib_context_set_angle (double *angle*)

Parameters:

angle Angle of the text strings.

Sets the angle at which text strings will be drawn if the text direction has been set to IMLIB_-TEXT_TO_ANGLE with imlib_context_set_direction().

4.1.2.29 EAPI void imlib_context_set_anti_alias (char *anti_alias*)

Parameters:

anti_alias The anti alias flag.

Toggles "anti-aliased" scaling of images. This isn't quite correct since it's actually super and sub pixel sampling that it turns on and off, but anti-aliasing is used for having "smooth" edges to lines and shapes and this means when images are scaled they will keep their smooth appearance. Passing in 1 turns this on and 0 turns it off.

4.1.2.30 EAPI void imlib_context_set_blend (char *blend*)

Parameters:

blend The blending flag.

When rendering an image to a drawable, Imlib2 is able to blend the image directly onto the drawable during rendering. Setting this to 1 will enable this. If the image has no alpha channel this has no effect. Setting it to 0 will disable this.

4.1.2.31 EAPI void imlib_context_set_cliprect (int *x*, int *y*, int *w*, int *h*)

Parameters:

x The top left x coordinate of the rectangle.

Parameters:

y The top left y coordinate of the rectangle.

w The width of the rectangle.

h The height of the rectangle.

Sets the rectangle of the current context.

4.1.2.32 EAPI void imlib_context_set_color (int *red*, int *green*, int *blue*, int *alpha*)

Parameters:

red Red channel of the current color.

Parameters:

green Green channel of the current color.

blue Blue channel of the current color.

alpha Alpha channel of the current color.

Sets the color with which text, lines and rectangles are drawn when being rendered onto an image. Values for `red`, `green`, `blue` and `alpha` are between 0 and 255 - any other values have undefined results.

4.1.2.33 EAPI void imlib_context_set_color_cmya (int *cyan*, int *magenta*, int *yellow*, int *alpha*)

Parameters:

cyan Cyan channel of the current color.

Parameters:

magenta Magenta channel of the current color.

yellow Yellow channel of the current color.

alpha Alpha channel of the current color.

Sets the color in CMYA space. Values for `cyan`, `magenta`, `yellow` and `alpha` are between 0 and 255 - any other values have undefined results.

4.1.2.34 EAPI void imlib_context_set_color_hlsa (float *hue*, float *lightness*, float *saturation*, int *alpha*)

Parameters:

hue Hue channel of the current color.

Parameters:

lightness Lightness channel of the current color.

saturation Saturation channel of the current color.

alpha Alpha channel of the current color.

Sets the color in HLSA space. Values for `hue` are between 0 and 360, values for `lightness` and `saturation` between 0 and 1, and values for `alpha` are between 0 and 255 - any other values have undefined results.

4.1.2.35 EAPI void imlib_context_set_color_hsva (float *hue*, float *saturation*, float *value*, int *alpha*)

Parameters:

hue Hue channel of the current color.

Parameters:

saturation Saturation channel of the current color.

value Value channel of the current color.

alpha Alpha channel of the current color.

Sets the color in HSVA space. Values for `hue` are between 0 and 360, values for `saturation` and `value` between 0 and 1, and values for `alpha` are between 0 and 255 - any other values have undefined results.

4.1.2.36 EAPI void imlib_context_set_color_modifier (Imlib_Color_Modifier *color_modifier*)

Parameters:

color_modifier Current color modifier.

Sets the current color modifier used for rendering pixmaps or images to a drawable or images onto other images. Color modifiers are lookup tables that map the values in the red, green, blue and alpha channels to other values in the same channel when rendering, allowing for fades, color correction etc. to be done whilst rendering. pass in NULL as the color_modifier to disable the color modifier for rendering.

4.1.2.37 EAPI void imlib_context_set_color_range (Imlib_Color_Range *color_range*)

Parameters:

color_range Color range.

Sets the current color range to use for rendering gradients.

4.1.2.38 EAPI void imlib_context_set_direction (Imlib_Text_Direction *direction*)

Parameters:

direction Text direction.

Sets the direction in which to draw text in terms of simple 90 degree orientations or an arbitrary angle. The direction can be one of IMLIB_TEXT_TO_RIGHT, IMLIB_TEXT_TO_LEFT, IMLIB_TEXT_TO_DOWN, IMLIB_TEXT_TO_UP or IMLIB_TEXT_TO_ANGLE. The default is IMLIB_TEXT_TO_RIGHT. If you use IMLIB_TEXT_TO_ANGLE, you will also have to set the angle with imlib_context_set_angle().

4.1.2.39 EAPI void imlib_context_set_dither (char *dither*)

Parameters:

dither The dithering flag.

Sets the dithering flag for rendering to a drawable or when pixmaps are produced. This affects the color image appearance by enabling dithering. Dithering slows down rendering but produces

considerably better results. this option has no effect foe rendering in 24 bit and up, but in 16 bit and lower it will dither, producing smooth gradients and much better quality images. setting dither to 1 enables it and 0 disables it.

4.1.2.40 EAPI void imlib_context_set_dither_mask (char *dither_mask*)

Parameters:

dither_mask The dither mask flag.

Selects if, you are rendering to a mask, or producing pixmap masks from images, if the mask is to be dithered or not. passing in 1 for dither_mask means the mask pixmap will be dithered, 0 means it will not be dithered.

4.1.2.41 EAPI void imlib_context_set_filter (Imlib_Filter *filter*)

Parameters:

filter Current filter.

Sets the current filter to be used when applying filters to images. Set this to NULL to disable filters.

4.1.2.42 EAPI void imlib_context_set_font (Imlib_Font *font*)

Parameters:

font Current font.

Sets the current font to use when rendering text. you should load the font first with imlib_load_font().

4.1.2.43 EAPI void imlib_context_set_image (Imlib_Image *image*)

Parameters:

image Current image.

Sets the current image Imlib2 will be using with its function calls.

4.1.2.44 EAPI void imlib_context_set_mask_alpha_threshold (int *mask_alpha_threshold*)

Parameters:

mask_alpha_threshold The mask alpha threshold.

Selects, if you are rendering to a mask, the alpha threshold above which mask bits are set. The default mask alpha threshold is 128, meaning that a mask bit will be set if the pixel alpha is >= 128.

4.1.2.45 EAPI void imlib_context_set_operation (Imlib_Operation *operation*)

Parameters:

operation

When Imlib2 draws an image onto another or an image onto a drawable it is able to do more than just blend the result on using the given alpha channel of the image. It is also able to do saturating additive, subtractive and a combination of the both (called reshade) rendering. The default mode is IMLIB_OP_COPY. you can also set it to IMLIB_OP_ADD, IMLIB_OP_SUBTRACT or IMLIB_OP_RESHADE. Use this function to set the rendering operation. IMLIB_OP_COPY performs basic alpha blending: DST = (SRC * A) + (DST * (1 - A)). IMLIB_OP_ADD does DST = DST + (SRC * A). IMLIB_OP_SUBTRACT does DST = DST - (SRC * A) and IMLIB_OP_RESHADE does DST = DST + (((SRC - 0.5) / 2) * A).

4.1.2.46 EAPI void imlib_context_set_progress_function (Imlib_Progress_Function *progress_function*)

Parameters:

progress_function A progress function.

Sets the progress function to be called back whilst loading images. Set this to the function to be called, or set it to NULL to disable progress callbacks whilst loading.

4.1.2.47 EAPI void imlib_context_set_progress_granularity (char *progress_granularity*)

Parameters:

progress_granularity A char.

This hints as to how often to call the progress callback. 0 means as often as possible. 1 means whenever 15 more of the image has been decoded, 10 means every 10% of the image decoding, 50 means every 50% and 100 means only call at the end. Values outside of the range 0-100 are undefined.

4.1.2.48 EAPI Imlib_Color_Modifier imlib_create_color_modifier (void)

Returns:

Valid handle.

Creates a new empty color modifier and returns a valid handle on success. NULL is returned on failure.

4.1.2.49 EAPI Imlib_Color_Range imlib_create_color_range (void)

Returns:

valid handle.

Creates a new empty color range and returns a valid handle to that color range.

4.1.2.50 EAPI Imlib_Image imlib_create_cropped_image (int *x*, int *y*, int *width*, int *height*)

Parameters:

x The top left x coordinate of the rectangle.

Parameters:

y The top left y coordinate of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

Returns:

A valid image, otherwise NULL.

Creates a duplicate of a (`x`, `y`, `width`, `height`) rectangle in the current image and returns a valid image handle on success, or NULL on failure.

4.1.2.51 EAPI Imlib_Image imlib_create_cropped_scaled_image (int *source_x*, int *source_y*, int *source_width*, int *source_height*, int *destination_width*, int *destination_height*)

Parameters:

source_x The top left x coordinate of the source rectangle.

Parameters:

source_y The top left y coordinate of the source rectangle.

source_width The width of the source rectangle.

source_height The height of the source rectangle.

destination_width The width of the destination image.

destination_height The height of the destination image.

Returns:

A valid image, otherwise NULL.

Works the same as imlib_create_cropped_image() but will scale the new image to the new destination `destination_width` and `destination_height` whilst cropping.

4.1.2.52 EAPI Imlib_Image imlib_create_image (int *width*, int *height*)

Parameters:

width The width of the image.

Parameters:

height The height of the image.

Returns:

A new blank image.

Creates a new blank image of size `width` and `height`. The contents of this image at creation time are undefined (they could be garbage memory). You are free to do whatever you like with this image. It is not cached. On success an image handle is returned - on failure NULL is returned.

4.1.2.53 EAPI Imlib_Image imlib_create_image_using_copied_data (int *width*, int *height*, DATA32 * *data*)

Parameters:

width The width of the image.

Parameters:

height The height of the image.

data The data.

Returns:

A valid image, otherwise NULL.

Works the same way as imlib_create_image_using_data() but Imlib2 copies the image data to the image structure. You may now do whatever you wish with the original data as it will not be needed anymore. Imlib2 returns a valid image handle on success or NULL on failure.

4.1.2.54 EAPI Imlib_Image imlib_create_image_using_data (int *width*, int *height*, DATA32 * *data*)

Parameters:

width The width of the image.

Parameters:

height The height of the image.

data The data.

Returns:

A valid image, otherwise NULL.

Creates an image from the image data specified with the width `width` and the height `height` specified. The image data `data` must be in the same format as imlib_image_get_data() would return. You are responsible for freeing this image data once the image is freed - Imlib2 will not do that for you. This is useful for when you already have static buffers of the same format Imlib2 uses (many video grabbing devices use such a format) and wish to use Imlib2 to render the results onto another image, or X drawable. You should free the image when you are done with it. Imlib2 returns a valid image handle on success or NULL on failure

4.1.2.55 EAPI Imlib_Image imlib_create_rotated_image (double *angle*)

Parameters:

angle An angle in radians.

Returns:

A new image, or NULL.

Creates an new copy of the current image, but rotated by `angle` radians. On success it returns a valid image handle, otherwise NULL.

4.1.2.56 EAPI void imlib_free_font_list (char ** *font_list*, int *number*)

Parameters:

font_list The font list.

Parameters:

number Number of fonts in the list.

Frees the font list returned by imlib_list_fonts().

4.1.2.57 EAPI int imlib_get_cache_size (void)

Returns:

The current image size.

Returns the current size of the image cache in bytes. The cache is a unified cache used for image data AND pixmaps.

4.1.2.58 EAPI void imlib_get_color_modifier_tables (DATA8 * *red_table*, DATA8 * *green_table*, DATA8 * *blue_table*, DATA8 * *alpha_table*)

Parameters:

red_table,: an array of DATA8.

Parameters:

green_table,: an array of DATA8.

blue_table,: an array of DATA8.

alpha_table,: an array of DATA8.

Copies the table values from the current color modifier into the pointers to mapping tables specified. They must have 256 entries and be DATA8 format.

4.1.2.59 EAPI int imlib_get_color_usage (void)

Returns:

The current number of colors.

Gets the number of colors Imlib2 currently at a maximum is allowed to allocate for rendering. The default is 256.

4.1.2.60 EAPI int imlib_get_font_ascent (void)

Returns:

The font's ascent.

Returns the current font's ascent value in pixels.

4.1.2.61 EAPI int imlib_get_font_cache_size (void)

Returns:

The font cache size.

Returns the font cache size in bytes.

4.1.2.62 EAPI int imlib_get_font_descent (void)

Returns:

The font's descent.

Returns the current font's descent value in pixels.

4.1.2.63 EAPI int imlib_get_maximum_font_ascent (void)

Returns:

The font's maximum ascent.

Returns the current font's maximum ascent extent.

4.1.2.64 EAPI int imlib_get_maximum_font_descent (void)

Returns:

The font's maximum descent.

Returns the current font's maximum descent extent.

4.1.2.65 EAPI void imlib_get_text_advance (const char * *text*, int * *horizontal_advance_return*, int * *vertical_advance_return*)

Parameters:

text A string.

Parameters:

horizontal_advance_return Horizontal offset.

vertical_advance_return Vertical offset.

Gets the advance horizontally and vertically in pixels the next text string would need to be placed at for the current font. The advances are not adjusted for rotation so you will have to translate the advances (which are calculated as if the text was drawn horizontally from left to right) depending on the text orientation.

4.1.2.66 EAPI int imlib_get_text_inset (const char * *text*)

Parameters:

text A string.

Returns:

The inset value of .

Returns the inset of the first character of `text` in using the current font and returns that value in pixels.

4.1.2.67 EAPI void imlib_get_text_size (const char * *text*, int * *width_return*, int * *height_return*)

Parameters:

text A string.

Parameters:

width_return The width of the text.

height_return The height of the text.

Gets the width and height in pixels the `text` string would use up if drawn with the current font.

4.1.2.68 EAPI void imlib_image_attach_data_value (const char * *key*, void * *data*, int *value*, Imlib_Internal_Data_Destructor_Function *destructor_function*)

Parameters:

key A string.

Parameters:

data A pointer.

value A value.

destructor_function An Imlib internal function.

Attaches data to the current image with the string key of `key`, and the data of `data` and an integer of `value`. The `destructor_function` function, if not NULL is called when this image is freed so the destructor can free `data`, if this is needed.

4.1.2.69 EAPI void imlib_image_blur (int *radius*)

Parameters:

radius The radius.

Blurs the current image. A `radius` value of 0 has no effect, 1 and above determine the blur matrix radius that determine how much to blur the image.

4.1.2.70 EAPI void imlib_image_copy_alpha_rectangle_to_image (Imlib_Image *image_source*, int *x*, int *y*, int *width*, int *height*, int *destination_x*, int *destination_y*)

Parameters:

image_source An image.

Parameters:

x The top left x coordinate of the rectangle.

y The top left y coordinate of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

destination_x The top left x coordinate of the destination rectangle.

destination_y The top left x coordinate of the destination rectangle.

Copies the source (`x`, `y`, `width`, `height`) rectangle alpha channel from the source image `image_source` and replaces the alpha channel on the destination image at the (`destination_x`, `destination_y`) coordinates.

4.1.2.71 EAPI void imlib_image_copy_alpha_to_image (Imlib_Image *image_source*, int *x*, int *y*)

Parameters:

image_source An image.

Parameters:

x The x coordinate.

y The y coordinate.

Copies the alpha channel of the source image `image_source` to the (`x`, `y`) coordinates of the current image, replacing the alpha channel there.

4.1.2.72 EAPI void imlib_image_copy_rect (int *x*, int *y*, int *width*, int *height*, int *new_x*, int *new_y*)

Parameters:

x The top left x coordinate of the rectangle.

Parameters:

y The top left y coordinate of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

new_x The top left x coordinate of the new location.

new_y The top left y coordinate of the new location.

Copies a rectangle of size `width`, `height` at the (`x`, `y`) location specified in the current image to a new location (`new_x`, `new_y`) in the same image.

4.1.2.73 EAPI void imlib_image_draw_ellipse (int *xc*, int *yc*, int *a*, int *b*)

Parameters:

xc X coordinate of the center of the ellipse.

Parameters:

yc Y coordinate of the center of the ellipse.

a The horizontal amplitude of the ellipse.

b The vertical amplitude of the ellipse.

Draws an ellipse on the current context image. The ellipse is defined as `(x-xc)^2/a^2 + (y-yc)^2/b^2 = 1`. This means that the point (`xc`, `yc`) marks the center of the ellipse, `a` defines the horizontal amplitude of the ellipse, and `b` defines the vertical amplitude.

4.1.2.74 EAPI Imlib_Updates imlib_image_draw_line (int *x1*, int *y1*, int *x2*, int *y2*, char *make_updates*)

Parameters:

x1 The x coordinate of the first point.

Parameters:

y1 The y coordinate of the first point.

x2 The x coordinate of the second point.

y2 The y coordinate of the second point.

make_updates,: a char.

Returns:

An updates list.

Draws a line using the current color on the current image from coordinates (`x1`, `y1`) to (`x2`, `y2`). If `make_updates` is 1 it will also return an update you can use for an updates list, otherwise it returns NULL.

4.1.2.75 EAPI void imlib_image_draw_polygon (ImlibPolygon *poly*, unsigned char *closed*)

Parameters:

poly A polygon.

Parameters:

closed Closed polygon flag.

Draws the polygon `poly` onto the current context image. Points which have been added to the polygon are drawn in sequence, first to last. The final point will be joined with the first point if `closed` is non-zero.

4.1.2.76 EAPI void imlib_image_draw_rectangle (int *x*, int *y*, int *width*, int *height*)

Parameters:

x The top left x coordinate of the rectangle.

Parameters:

y The top left y coordinate of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

Draws the outline of a rectangle on the current image at the (`x`, `y`) coordinates with a size of `width` and `height` pixels, using the current color.

4.1.2.77 EAPI void imlib_image_fill_color_range_rectangle (int *x*, int *y*, int *width*, int *height*, double *angle*)

Parameters:

x The x coordinate of the left edge of the rectangle.

Parameters:

y The y coordinate of the top edge of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

angle Angle of gradient.

Fills a rectangle of width `width` and height `height` at the (`x`, `y`) location specified in the current image with a linear gradient of the current color range at an angle of `angle` degrees with 0 degrees being vertical from top to bottom going clockwise from there.

4.1.2.78 EAPI void imlib_image_fill_ellipse (int *xc*, int *yc*, int *a*, int *b*)

Parameters:

xc X coordinate of the center of the ellipse.

Parameters:

yc Y coordinate of the center of the ellipse.

a The horizontal amplitude of the ellipse.

b The vertical amplitude of the ellipse.

Fills an ellipse on the current context image using the current context color. The ellipse is defined as `(x-xc)^2/a^2 + (y-yc)^2/b^2 = 1`. This means that the point (`xc`, `yc`) marks the center of the ellipse, `a` defines the horizontal amplitude of the ellipse, and `b` defines the vertical amplitude.

4.1.2.79 EAPI void imlib_image_fill_hsva_color_range_rectangle (int *x*, int *y*, int *width*, int *height*, double *angle*)

Parameters:

x The x coordinate of the left edge of the rectangle.

Parameters:

y The y coordinate of the top edge of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

angle Angle of gradient.

Fills a rectangle of width `width` and height `height` at the (`x`, `y`) location specified in the current image with a linear gradient in HSVA color space of the current color range at an angle of `angle` degrees with 0 degrees being vertical from top to bottom going clockwise from there.

4.1.2.80 EAPI void imlib_image_fill_polygon (ImlibPolygon *poly*)

Parameters:

poly A polygon.

Fills the area defined by the polygon `polyon` the current context image with the current context color.

4.1.2.81 EAPI void imlib_image_fill_rectangle (int *x*, int *y*, int *width*, int *height*)

Parameters:

x The top left x coordinate of the rectangle.

Parameters:

y The top left y coordinate of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

Draws a filled rectangle on the current image at the (`x`, `y`) coordinates with a size of `width` and `height` pixels, using the current color.

4.1.2.82 EAPI char* imlib_image_format (void)

Returns:

Current image format.

Returns the current image's format. Do not free this string. Duplicate it if you need it for later use.

4.1.2.83 EAPI void* imlib_image_get_attached_data (const char * *key*)

Parameters:

key A string.

Returns:

The attached data as a pointer, or NULL if none.

Gets the data attached to the current image with the key `key` specified. NULL is returned if no data could be found with that key on the current image.

4.1.2.84 EAPI int imlib_image_get_attached_value (const char * *key*)

Parameters:

key A string.

Returns:

The attached value as an integer, or 0 if none.

Returns the value attached to the current image with the specified key `key`. If none could be found 0 is returned.

4.1.2.85 EAPI void imlib_image_get_border (Imlib_Border * *border*)

Parameters:

border The border of the image.

Fills the Imlib_Border structure to which `border` points to with the values of the border of the current context image. The border is the area at the edge of the image that does not scale with the rest of the image when resized - the borders remain constant in size. This is useful for scaling bevels at the edge of images differently to the image center.

4.1.2.86 EAPI DATA32* imlib_image_get_data (void)

Returns:

A pointer to the image data.

Returns a pointer to the image data in the image set as the image for the current context. When you get this pointer it is assumed you are planning on writing to the data, thus once you do this

the image can no longer be used for caching - in fact all images cached from this one will also be affected when you put the data back. If this matters it is suggested you clone the image first before playing with the image data. The image data is returned in the format of a DATA32 (32 bits) per pixel in a linear array ordered from the top left of the image to the bottom right going from left to right each line. Each pixel has the upper 8 bits as the alpha channel and the lower 8 bits are the blue channel - so a pixel's bits are ARGB (from most to least significant, 8 bits per channel). You must put the data back at some point.

4.1.2.87 EAPI DATA32* imlib_image_get_data_for_reading_only (void)

Returns:

A pointer to the image data.

Functions the same way as imlib_image_get_data(), but returns a pointer expecting the program to NOT write to the data returned (it is for inspection purposes only). Writing to this data has undefined results. The data does not need to be put back.

4.1.2.88 EAPI const char* imlib_image_get_filename (void)

Returns:

The current filename.

Returns the filename for the file that is set as the current context. The pointer returned is only valid as long as no operations cause the filename of the image to change. Saving the file with a different name would cause this. It is suggested you duplicate the string if you wish to continue to use the string for later processing. Do not free the string pointer returned by this function.

4.1.2.89 EAPI char imlib_image_has_alpha (void)

Returns:

Current alpha channel flag.

Returns 1 if the current context image has an alpha channel, or 0 if it does not (the alpha data space is still there and available - just "unused").

4.1.2.90 EAPI void imlib_image_orientate (int *orientation*)

Parameters:

orientation The orientation.

Performs 90 degree rotations on the current image. Passing in `orientation` does not rotate, 1 rotates clockwise by 90 degree, 2, rotates clockwise by 180 degrees, 3 rotates clockwise by 270 degrees.

4.1.2.91 EAPI void imlib_image_put_back_data (DATA32 * *data*)

Parameters:

data The pointer to the image data.

Puts back `data` when it was obtained by imlib_image_get_data(). `data` must be the same pointer returned by imlib_image_get_data(). This operated on the current context image.

4.1.2.92 EAPI void imlib_image_query_pixel (int *x*, int *y*, Imlib_Color * *color_return*)

Parameters:

x The x coordinate of the pixel.

Parameters:

y The y coordinate of the pixel.

color_return The returned color.

Fills the `color_return` color structure with the color of the pixel in the current image that is at the (`x`, `y`) location specified.

4.1.2.93 EAPI void imlib_image_query_pixel_cmya (int *x*, int *y*, int * *cyan*, int * *magenta*, int * *yellow*, int * *alpha*)

Parameters:

x Tthe x coordinate of the pixel.

Parameters:

y The y coordinate of the pixel.

cyan The returned cyan channel.

magenta The returned magenta channel.

yellow The returned yellow channel.

alpha The returned alpha channel.

Gets the CMYA color of the pixel from the current image that is at the (`x`, `y`) location specified.

4.1.2.94 EAPI void imlib_image_query_pixel_hlsa (int *x*, int *y*, float * *hue*, float * *lightness*, float * *saturation*, int * *alpha*)

Parameters:

x The x coordinate of the pixel.

Parameters:

y The y coordinate of the pixel.

hue The returned hue channel.

lightness The returned lightness channel.

saturation The returned saturation channel.

alpha The returned alpha channel.

Gets the HLSA color of the pixel from the current image that is at the (`x`, `y`) location specified.

4.1.2.95 EAPI void imlib_image_query_pixel_hsva (int *x*, int *y*, float * *hue*, float * *saturation*, float * *value*, int * *alpha*)

Parameters:

x The x coordinate of the pixel.

Parameters:

y The y coordinate of the pixel.

hue The returned hue channel.

saturation The returned saturation channel.

value The returned value channel.

alpha The returned alpha channel.

Gets the HSVA color of the pixel from the current image that is at the (`x`, `y`) location specified.

4.1.2.96 EAPI void imlib_image_remove_and_free_attached_data_value (const char * *key*)

Parameters:

key A string.

Removes the data and value attached to the current image with the specified key `key` and also calls the destructor function that was supplied when attaching it (see imlib_image_attach_data_value()).

4.1.2.97 EAPI void imlib_image_remove_attached_data_value (const char * *key*)

Parameters:

key A string.

Detaches the data & value attached with the specified key `key` from the current image.

4.1.2.98 EAPI void imlib_image_scroll_rect (int *x*, int *y*, int *width*, int *height*, int *delta_x*, int *delta_y*)

Parameters:

x The top left x coordinate of the rectangle.

Parameters:

y The top left y coordinate of the rectangle.

width The width of the rectangle.

height The height of the rectangle.

delta_x Distance along the x coordinates.

delta_y Distance along the y coordinates.

Scrolls a rectangle of size `width`, `height` at the (`x`, `y`) location within the current image by the `delta_x`, `delta_y` distance (in pixels).

4.1.2.99 EAPI void imlib_image_set_border (Imlib_Border * *border*)

Parameters:

border The border of the image.

Sets the border of the current context image to the values contained in the Imlib_Border structure `border` points to.

4.1.2.100 EAPI void imlib_image_set_changes_on_disk (void)

By default Imlib2 will not check the timestamp of an image on disk and compare it with the image in its cache - this is to minimize disk activity when using the cache.

Call this function and it will flag the current context image as being liable to change on disk and Imlib2 will check the timestamp of the image file on disk and compare it with the cached image when it next needs to use this image in the cache.

4.1.2.101 EAPI void imlib_image_set_format (const char * *format*)

Parameters:

format Format of the image.

Sets the format of the current image. This is used for when you wish to save an image in a different format that it was loaded in, or if the image currently has no file format associated with it.

4.1.2.102 EAPI void imlib_image_set_has_alpha (char *has_alpha*)

Parameters:

has_alpha Alpha flag.

Sets the alpha flag for the current image. Set `has_alpha` to 1 to enable the alpha channel in the current image, or 0 to disable it.

4.1.2.103 EAPI void imlib_image_set_irrelevant_alpha (char *irrelevant*)

Parameters:

irrelevant Irrelevant alpha flag.

Sets if the alpha channel status of the current image (i.e. if there is or is not one) is important for caching purposes. By default it is not. Set irrelevant to 1 to make it irrelevant and 0 to make it relevant.

4.1.2.104 EAPI void imlib_image_set_irrelevant_border (char *irrelevant*)

Parameters:

irrelevant Irrelevant border flag.

Sets if the border of the current image is irrelevant for caching purposes. By default it is. Set irrelevant to 1 to make it irrelevant, and 0 to make it relevant.

4.1.2.105 EAPI void imlib_image_set_irrelevant_format (char *irrelevant*)

Parameters:

irrelevant Irrelevant format flag.

Sets if the format value of the current image is irrelevant for caching purposes - by default it is. pass irrelevant as 1 to make it irrelevant and 0 to make it relevant for caching.

4.1.2.106 EAPI void imlib_image_sharpen (int *radius*)

Parameters:

radius The radius.

Sharpens the current image. The `radius` value affects how much to sharpen by.

4.1.2.107 EAPI void imlib_image_tile (void)

Modifies an image so it will tile seamlessly horizontally and vertically if used as a tile (i.e.

drawn multiple times horizontally and vertically).

4.1.2.108 EAPI void imlib_image_tile_horizontal (void)

Modifies an image so it will tile seamlessly horizontally if used as a tile (i.e.

drawn multiple times horizontally).

4.1.2.109 EAPI void imlib_image_tile_vertical (void)

Modifies an image so it will tile seamlessly vertically if used as a tile (i.e.

drawn multiple times vertically).

4.1.2.110 EAPI char** imlib_list_font_path (int * *number_return*)

Parameters:

number_return Number of paths in the list.

Returns:

A list of strings.

Returns a list of strings that are the directories in the font path. Do not free this list or change it in any way. If you add or delete members of the font path this list will be invalid. If you intend to use this list later duplicate it for your own use. The number of elements in the array of strings is put into `number_return`.

4.1.2.111 EAPI char** imlib_list_fonts (int * *number_return*)

Parameters:

number_return Number of fonts in the list.

Returns:

A list of fonts.

Returns a list of fonts imlib2 can find in its font path.

4.1.2.112 EAPI Imlib_Font imlib_load_font (const char * *font_name*)

Parameters:

font_name The font name with the size.

Returns:

NULL if no font found.

Loads a truetype font from the first directory in the font path that contains that font. The font name `font_name` format is "font_name/size". For example. If there is a font file called blum.ttf somewhere in the font path you might use "blum/20" to load a 20 pixel sized font of blum. If the font cannot be found NULL is returned.

4.1.2.113 EAPI Imlib_Image imlib_load_image (const char * *file*)

Parameters:

file Image file.

Returns:

An image handle.

Loads an image from disk located at the path specified by `file`. Please see the section How Image Loading Works for more detail. Returns an image handle on success or NULL on failure.

4.1.2.114 EAPI Imlib_Image imlib_load_image_immediately (const char * *file*)

Parameters:

file Image file.

Returns:

An image handle.

Loads an image from disk located at the path specified by `file`. This forces the image data to be decoded at load time too, instead of decoding being deferred until it is needed. Returns an image handle on success or NULL on failure.

4.1.2.115 EAPI Imlib_Image imlib_load_image_immediately_without_cache (const char * *file*)

Parameters:

file Image file.

Returns:

An image handle.

Loads the image without deferred image data decoding (i.e. it is decoded straight away) and without looking in the cache. Returns an image handle on success or NULL on failure.

4.1.2.116 EAPI Imlib_Image imlib_load_image_with_error_return (const char * *file*, Imlib_Load_Error * *error_return*)

Parameters:

file Image file.

Parameters:

error_return The returned error.

Returns:

An image handle.

Loads an image at the path `file` on disk. If it succeeds it returns a valid image handle, if not NULL is returned and `error_return` is set to the detail of the error.

4.1.2.117 EAPI Imlib_Image imlib_load_image_without_cache (const char * *file*)

Parameters:

file Image file.

Returns:

An image handle.

Loads the image without looking in the cache first. Returns an image handle on success or NULL on failure.

4.1.2.118 EAPI void imlib_modify_color_modifier_brightness (double *brightness_value*)

Parameters:

brightness_value Value of brightness.

Modifies the current color modifier by adjusting the brightness by the value `brightness_value`. The color modifier is modified not set, so calling this repeatedly has cumulative effects. brightness values of 0 do not affect anything. -1.0 will make things completely black and 1.0 will make things all white. Values in-between vary brightness linearly.

4.1.2.119 EAPI void imlib_modify_color_modifier_contrast (double *contrast_value*)

Parameters:

contrast_value Value of contrast.

Modifies the current color modifier by adjusting the contrast by the value `contrast_value`. The color modifier is modified not set, so calling this repeatedly has cumulative effects. Contrast of 1.0 does nothing. 0.0 will merge to gray, 2.0 will double contrast etc.

4.1.2.120 EAPI void imlib_modify_color_modifier_gamma (double *gamma_value*)

Parameters:

gamma_value Value of gamma.

Modifies the current color modifier by adjusting the gamma by the value specified `gamma_value`. The color modifier is modified not set, so calling this repeatedly has cumulative effects. A gamma of 1.0 is normal linear, 2.0 brightens and 0.5 darkens etc. Negative values are not allowed.

4.1.2.121 EAPI void imlib_polygon_add_point (ImlibPolygon *poly*, int *x*, int *y*)

Parameters:

poly A polygon

Parameters:

x The X coordinate.

y The Y coordinate.

Adds the point (`x`, `y`) to the polygon `poly`. The point will be added to the end of the polygon's internal point list. The points are drawn in order, from the first to the last.

4.1.2.122 EAPI unsigned char imlib_polygon_contains_point (ImlibPolygon *poly*, int *x*, int *y*)

Parameters:

poly A polygon

Parameters:

x The X coordinate.

y The Y coordinate.

Returns non-zero if the point (`x`, `y`) is within the area defined by the polygon `poly`.

4.1.2.123 EAPI void imlib_polygon_free (ImlibPolygon *poly*)

Parameters:

poly A polygon.

Frees a polygon object.

4.1.2.124 EAPI void imlib_polygon_get_bounds (ImlibPolygon *poly*, int * *px1*, int * *py1*, int * *px2*, int * *py2*)

Parameters:

poly A polygon.

Parameters:

px1 X coordinate of the upper left corner.

py1 Y coordinate of the upper left corner.

px2 X coordinate of the lower right corner.

py2 Y coordinate of the lower right corner.

Calculates the bounding area of the polygon `poly`. (`px1`, `py1`) defines the upper left corner of the bounding box and (`px2`, `py2`) defines it's lower right corner.

4.1.2.125 EAPI void imlib_remove_path_from_font_path (const char * *path*)

Parameters:

path A directory path.

Removes all directories in the font path that match `path`.

4.1.2.126 EAPI void imlib_save_image (const char * *filename*)

Parameters:

filename The file name.

Saves the current image in the format specified by the current image's format settings to the filename `filename`.

4.1.2.127 EAPI void imlib_save_image_with_error_return (const char * *filename*, Imlib_Load_Error * *error_return*)

Parameters:

filename The file name.

Parameters:

error_return The returned error.

Works the same way imlib_save_image() works, but will set the `error_return` to an error value if the save fails.

4.1.2.128 EAPI void imlib_set_cache_size (int *bytes*)

Parameters:

bytes Cache size.

Sets the cache size. The size is in bytes. Setting the cache size to 0 effectively flushes the cache and keeps the cache size at 0 until set to another value. Whenever you set the cache size Imlib2 will flush as many old images and pixmap from the cache as needed until the current cache usage is less than or equal to the cache size.

4.1.2.129 EAPI void imlib_set_color_modifier_tables (DATA8 * *red_table*, DATA8 * *green_table*, DATA8 * *blue_table*, DATA8 * *alpha_table*)

Parameters:

red_table An array of DATA8.

Parameters:

green_table An array of DATA8.

blue_table An array of DATA8.

alpha_table An array of DATA8.

Explicitly copies the mapping tables from the table pointers passed into this function into those of the current color modifier. Tables are 256 entry arrays of DATA8 which are a mapping of that channel value to a new channel value. A normal mapping would be linear (v[0] = 0, v[10] = 10, v[50] = 50, v[200] = 200, v[255] = 255).

4.1.2.130 EAPI void imlib_set_color_usage (int *max*)

Parameters:

max Maximum number of colors.

Sets the maximum number of colors you would like Imlib2 to allocate for you when rendering. The default is 256. This has no effect in depths greater than 8 bit.

4.1.2.131 EAPI void imlib_set_font_cache_size (int *bytes*)

Parameters:

bytes The font cache size.

Sets the font cache in bytes. Whenever you set the font cache size Imlib2 will flush fonts from the cache until the memory used by fonts is less than or equal to the font cache size. Setting the size to 0 effectively frees all speculatively cached fonts.

4.1.2.132 EAPI void imlib_text_draw (int *x*, int *y*, const char * *text*)

Parameters:

x The x coordinate of the top left corner.

Parameters:

y The y coordinate of the top left corner.

text A null-byte terminated string.

Draws the null-byte terminated string `text` using the current font on the current image at the (`x`, `y`) location (`x`, `y` denoting the top left corner of the font string)

4.1.2.133 EAPI void imlib_text_draw_with_return_metrics (int *x*, int *y*, const char * *text*, int * *width_return*, int * *height_return*, int * *horizontal_advance_return*, int * *vertical_advance_return*)

Parameters:

x The x coordinate of the top left corner.

Parameters:

y The y coordinate of the top left corner.

text A null-byte terminated string.

width_return The width of the string.

height_return The height of the string.

horizontal_advance_return Horizontal offset.

vertical_advance_return Vertical offset.

Works just like imlib_text_draw() but also returns the width and height of the string drawn, and `horizontal_advance_return` returns the number of pixels you should advance horizontally to draw another string (useful if you are drawing a line of text word by word) and `vertical_advance_return` does the same for the vertical direction (i.e. drawing text line by line).

4.1.2.134 EAPI int imlib_text_get_index_and_location (const char * *text*, int *x*, int *y*, int * *char_x_return*, int * *char_y_return*, int * *char_width_return*, int * *char_height_return*)

Parameters:

text A string.

Parameters:

x The x offset.

y The y offset.

char_x_return The x coordinate of the character.

char_y_return The x coordinate of the character.

char_width_return The width of the character.

char_height_return The height of the character.

Returns:

-1 if no character found.

Returns the character number in the string `text` using the current font at the (`x`, `y`) pixel location which is an offset relative to the top left of that string. -1 is returned if there is no character there. If there is a character, `char_x_return`, `char_y_return`, `char_width_return` and `char_height_return` (respectively the character x, y, width and height) are also filled in.

4.1.2.135 EAPI void imlib_text_get_location_at_index (const char * *text*, int *index*, int * *char_x_return*, int * *char_y_return*, int * *char_width_return*, int * *char_height_return*)

Parameters:

text A string.

Parameters:

index The index of .

char_x_return The x coordinate of the character.

char_y_return The y coordinate of the character.

char_width_return The width of the character.

char_height_return The height of the character.

Gets the geometry of the character at index `index` in the `text` string using the current font.

4.1.2.136 EAPI Imlib_Updates imlib_update_append_rect (Imlib_Updates *updates*, int *x*, int *y*, int *w*, int *h*)

Parameters:

updates An updates list.

Parameters:

x The top left x coordinate of the rectangle.

y The top left y coordinate of the rectangle.

w The width of the rectangle.

h The height of the rectangle.

Returns:

The updates handle.

Appends an update rectangle to the updates list passed in (if the updates is NULL it will create a new updates list) and returns a handle to the modified updates list (the handle may be modified so only use the new updates handle returned).

4.1.2.137 EAPI Imlib_Updates imlib_updates_append_updates (Imlib_Updates *updates*, Imlib_Updates *appended_updates*)

Parameters:

updates An updates list.

Parameters:

appended_updates The updates list to append.

Returns:

The new updates list.

Appends `appended_updates` to the updates list `updates` and returns the new list.

4.1.2.138 EAPI Imlib_Updates imlib_updates_clone (Imlib_Updates *updates*)

Parameters:

updates An updates list.

Returns:

Duplicate of `updates`.

Creates a duplicate of the updates list passed into the function.

4.1.2.139 EAPI void imlib_updates_free (Imlib_Updates *updates*)

Parameters:

updates An updates list.

Frees an updates list.

4.1.2.140 EAPI void imlib_updates_get_coordinates (Imlib_Updates *updates*, int * *x_return*, int * *y_return*, int * *width_return*, int * *height_return*)

Parameters:

updates An updates list.

Parameters:

x_return The top left x coordinate of the update.
y_return The top left y coordinate of the update.
width_return The width of the update.
height_return The height of the update.

Returns the coordinates of an update.

4.1.2.141 EAPI Imlib_Updates imlib_updates_get_next (Imlib_Updates *updates*)

Parameters:

updates An updates list.

Returns:

The next updates.

Gets the next update in the updates list relative to the one passed in.

4.1.2.142 EAPI Imlib_Updates imlib_updates_init (void)

Returns:

The initialized updates list.

Initializes an updates list before you add any updates to it or merge it for rendering etc.

4.1.2.143 EAPI Imlib_Updates imlib_updates_merge (Imlib_Updates *updates*, int *w*, int *h*)

Parameters:

updates An updates list.

Parameters:

w The width of the rectangle.

h The height of the rectangle.

Returns:

The updates handle.

Takes an updates list, and modifies it by merging overlapped rectangles and lots of tiny rectangles into larger rectangles to minimize the number of rectangles in the list for optimized redrawing. The new updates handle is now valid and the old one passed in is not.

4.1.2.144 EAPI Imlib_Updates imlib_updates_merge_for_rendering (Imlib_Updates *updates*, int *w*, int *h*)

Parameters:

updates An updates list.

Parameters:

w The width of the rectangle.

h The height of the rectangle.

Returns:

The updates handle.

Works almost exactly as imlib_updates_merge() but is more lenient on the spacing between update rectangles - if they are very close it amalgamates 2 smaller rectangles into 1 larger one.

4.1.2.145 EAPI void imlib_updates_set_coordinates (Imlib_Updates *updates*, int *x*, int *y*, int *width*, int *height*)

Parameters:

updates An updates list.

Parameters:

x The top left x coordinate of the update.

y The top left y coordinate of the update.

width The width of the update.

height The height of the update.

Modifies the coordinates of an update in `update`.

Chapter 5

Imlib2 Page Documentation

5.1 Todo List

page Imlib2 Library Documentation line code doesnt draw nice liens when clipping - fix

page Imlib2 Library Documentation filled polygons can break fill bounds on corner cases - fix

page Imlib2 Library Documentation go thru TODOs and FIXMEs

Index