ProjetTechnoL3TP1

0.0.1

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 ei_app_event_t Struct Reference

The event parameter for application defined event types.

`#include <ei_event.h>`

Public Attributes

- void * **user_param**

3.1.1 Detailed Description

The event parameter for application defined event types.

The documentation for this struct was generated from the following file:

- include/ei_event.h

3.2 ei_color_t Struct Reference

A color with transparency.

`#include <ei_types.h>`

Public Attributes

- unsigned char red
  The red component of the color.
- unsigned char green
  The green component of the color.
- unsigned char blue
  The blue component of the color.
- unsigned char alpha

3.2.1 Detailed Description

A color with transparency.

Each channel is represented as an 8 bits unsigned interger, hence channel's minimum value is 0, maximum is 255.

3.2.2 Member Data Documentation

3.2.2.1 unsigned char ei_color_t::alpha

The transparency of the color. 0 is invisible,

The documentation for this struct was generated from the following file:

- include/ei_types.h

3.3 ei_display_event_t Struct Reference

The event parameter for display-related event types.

`#include <ei_event.h>`

Public Attributes

- ei_bool_t resized
  The window has been resized.
- ei_bool_t closed
  The close button of the window has been pressed.
- ei_bool_t switched_out
  The window is no longer active.
- ei_bool_t switched_in
  The window is active once again.

3.3.1 Detailed Description

The event parameter for display-related event types.

The documentation for this struct was generated from the following file:

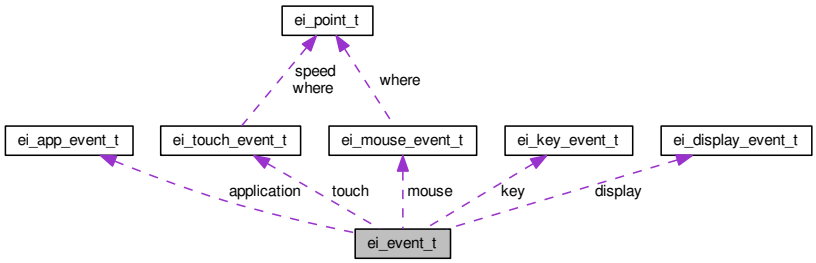
- include/ei_event.h

3.4 ei_event_t Struct Reference

Describes an event.

`#include <ei_event.h>`

Collaboration diagram for ei_event_t:



Public Attributes

- ei_eventtype_t type
  The type of the event.
- union {
  ei_display_event_t display
  Event parameter for display-realted events (see ei_display_event_t).
  ei_key_event_t key
  Event parameters for keyboard-related events (see ei_key_event_t).
  ei_mouse_event_t mouse
  Event parameters for mouse-related events (see ei_mouse_event_t).
  ei_touch_event_t touch
  Event parameters for touch-related events (see ei_touch_event_t).
  ei_app_event_t application
  Event parameters for application-related events (see ei_app_event_t).
  } **param**

3.4.1 Detailed Description

Describes an event.

The documentation for this struct was generated from the following file:

- include/ei_event.h

3.5 ei_key_event_t Struct Reference

The event parameter for keyboard-related event types.

`#include <ei_event.h>`

Public Attributes

- int key_sym

  The keyboard key symbol (see allegro5/keycodes.h).
- int unichar

  For ei_ev_keychar, a Unicode code point (character).
- ei_modifier_mask_t modifier_mask

  The state of the modifier keys at the time of the event.

3.5.1 Detailed Description

The event parameter for keyboard-related event types.

The documentation for this struct was generated from the following file:

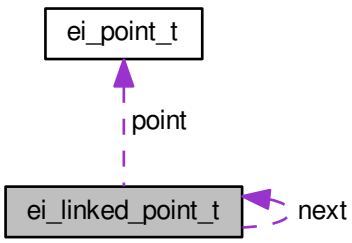
- include/ei_event.h

3.6 ei_linked_point_t Struct Reference

A point plus a pointer to create a linked list.

```
#include <ei_types.h>
```

Collaboration diagram for ei_linked_point_t:



Public Attributes

- ei_point_t point

  The point.
- struct ei_linked_point_t * next

  The pointer to the next element in the linked list.

3.6.1 Detailed Description

A point plus a pointer to create a linked list.

The documentation for this struct was generated from the following file:

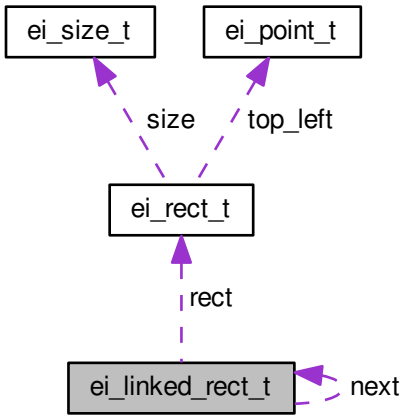
- include/ei_types.h

3.7 ei_linked_rect_t Struct Reference

A rectangle plus a pointer to create a linked list.

`#include <ei_types.h>`

Collaboration diagram for ei_linked_rect_t:



Public Attributes

- ei_rect_t rect

  The rectangle.
- struct ei_linked_rect_t * next

  The pointer to the next element in the linked list.

3.7.1 Detailed Description

A rectangle plus a pointer to create a linked list.

The documentation for this struct was generated from the following file:

- include/ei_types.h

3.8 ei_linked_tag_t Struct Reference

A tag and a pointer to create a linked list.

`#include <ei_event.h>`

Collaboration diagram for ei_linked_tag_t:

Public Attributes

- ei_tag_t **tag**
- struct ei_linked_tag_t * **next**

3.8.1 Detailed Description

A tag and a pointer to create a linked list.

The documentation for this struct was generated from the following file:

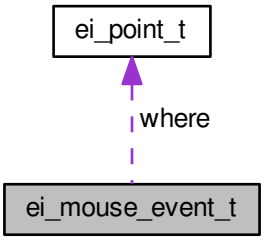
- include/ei_event.h

3.9 ei_mouse_event_t Struct Reference

The event parameter for mouse-related event types.

`#include <ei_event.h>`

Collaboration diagram for ei_mouse_event_t:



Public Attributes

- ei_point_t where
  Where the mouse pointer was at the time of the event.
- int button_number

3.9.1 Detailed Description

The event parameter for mouse-related event types.

3.9.2 Member Data Documentation

3.9.2.1 int ei_mouse_event_t::button_number

The number of the button that was pressed or released.

The documentation for this struct was generated from the following file:

- include/ei_event.h

3.10 ei_point_t Struct Reference

A 2-D point with integer coordinates.

```
#include <ei_types.h>
```

Public Attributes

- int x
  The abscissa of the point. The origin is on the left side of the image.
- int y

3.10.1 Detailed Description

A 2-D point with integer coordinates.

3.10.2 Member Data Documentation

3.10.2.1 int ei_point_t::y

The ordinate of the point, the origin is at the top of the image,

The documentation for this struct was generated from the following file:

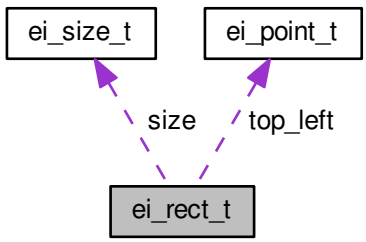
- include/ei_types.h

3.11 ei_rect_t Struct Reference

A rectangle defined by its top-left corner, and its size.

`#include <ei_types.h>`

Collaboration diagram for ei_rect_t:



Public Attributes

- ei_point_t top_left
  Coordinates of the top-left corner of the rectangle.
- ei_size_t size
  Size of the rectangle.

3.11.1 Detailed Description

A rectangle defined by its top-left corner, and its size.

The documentation for this struct was generated from the following file:

- include/ei_types.h

3.12 ei_size_t Struct Reference

A 2-D size with integer dimensions.

`#include <ei_types.h>`

Public Attributes

- int **width**
- int **height**

3.12.1 Detailed Description

A 2-D size with integer dimensions.

The documentation for this struct was generated from the following file:

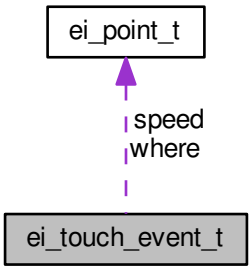
- include/ei_types.h

3.13 ei_touch_event_t Struct Reference

The event parameter for mouse-related event types.

```
#include <ei_event.h>
```

Collaboration diagram for ei_touch_event_t:



Public Attributes

- ei_point_t where

  Where the touch was at the time of the event.
- ei_point_t speed

  Movement speed in pixels.
- int touch_id
- ei_bool_t primary

  it will stay the same for events from the same finger until the touch ends.

3.13.1 Detailed Description

The event parameter for mouse-related event types.

3.13.2 Member Data Documentation

3.13.2.1 ei_bool_t ei_touch_event_t::primary

it will stay the same for events from the same finger until the touch ends.

Whether this is the only/first touch or an additional touch.

3.13.2.2 int ei_touch_event_t::touch_id

An identifier for this touch. If supported by the device

The documentation for this struct was generated from the following file:

- include/ei_event.h

Chapter 4

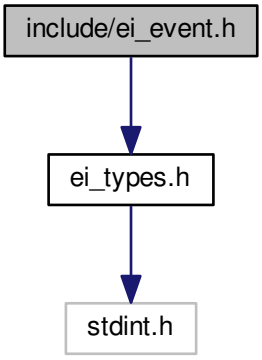
File Documentation

4.1 include/ei_event.h File Reference

Allows the binding and unbinding of callbacks to events.

```
#include "ei_types.h"
```

Include dependency graph for ei_event.h:



Classes

- struct ei_linked_tag_t

  A tag and a pointer to create a linked list.
- struct ei_display_event_t

  The event parameter for display-related event types.
- struct ei_key_event_t

  The event parameter for keyboard-related event types.
- struct ei_mouse_event_t

  The event parameter for mouse-related event types.

- struct ei_touch_event_t

  The event parameter for mouse-related event types.
- struct ei_app_event_t

  The event parameter for application defined event types.
- struct ei_event_t

  Describes an event.

Typedefs

- typedef char * ei_tag_t

  A string that can be attached to a widget. All widget have the tag of the name of their widget class, and the tag "all".
- typedef struct ei_linked_tag_t ei_linked_tag_t

  A tag and a pointer to create a linked list.
- typedef uint32_t ei_modifier_mask_t

  A bitfield indicating which of the modifier keys are currently pressed.
- typedef struct ei_event_t ei_event_t

  Describes an event.

Enumerations

- enum ei_eventtype_t {
  ei_ev_none = 0, ei_ev_app, ei_ev_display, ei_ev_keydown,
  ei_ev_keyup, ei_ev_keychar, ei_ev_mouse_buttondown, ei_ev_mouse_buttonup,
  ei_ev_mouse_move, ei_ev_touch_begin, ei_ev_touch_end, ei_ev_touch_move,
  ei_ev_last }

  The types of events.
- enum ei_modifier_key_t {
  ei_mod_shift = 0x00001, ei_mod_ctrl = 0x00002, ei_mod_alt = 0x00004, ei_mod_meta_left = 0x00008,
  ei_mod_meta_right = 0x00010, ei_mod_alt_grad = 0x00040 }

  The modifier keys (shift, alt, etc.)

4.1.1 Detailed Description

Allows the binding and unbinding of callbacks to events.

Author

Created by François Bérard on 30.12.11. Copyright 2011 Ensimag. All rights reserved.

4.1.2 Enumeration Type Documentation

4.1.2.1 enum ei_eventtype_t

The types of events.

Enumerator

ei_ev_none No event, used on an un-initialized structure.

ei_ev_app An application event, created by hw_event_post_app.
ei_ev_display A display / window event.
ei_ev_keydown A keyboard key has been pressed.
ei_ev_keyup A keyboard key has been released.
ei_ev_keychar A character was typed on the keyboard.
ei_ev_mouse_buttondown A mouse button has been pressed.
ei_ev_mouse_buttonup A mouse button has been released.
ei_ev_mouse_move The mouse has moved.
ei_ev_touch_begin The touch input device registered a new touch.
ei_ev_touch_end A touch ended.
ei_ev_touch_move The position of a touch changed.
ei_ev_last Last event type, its value is the number of event types.

4.1.2.2 enum ei_modifier_key_t

The modifier keys (shift, alt, etc.)

Enumerator

ei_mod_shift The "shift" key.
ei_mod_ctrl The "control" key.
ei_mod_alt The "alternate" key at the left of the space bar.
ei_mod_meta_left The "meta" (command) key at the left of the space bar.
ei_mod_meta_right The "meta" (command) key at the right of the space bar.
ei_mod_alt_grad The "alternate" key at the right of the space bar.

4.2 include/ei_main.h File Reference

Declares the "ei_main" function: the main function of programs built with the libei.

Functions

- int ei_main (int argc, char ∗argv[ ])
  The main function of the program.

4.2.1 Detailed Description

Declares the "ei_main" function: the main function of programs built with the libei.

Author

Created by François Bérard on 30.12.11. Copyright 2011 Ensimag. All rights reserved.

4.2.2 Function Documentation

4.2.2.1 int ei_main (int *argc*, char ∗ *argv[ ]*)

The main function of the program.

Programmers must not define their main function in a function called "main", because the "main" function is defined by SDL and linked with in the libeibase library.

Parameters

argc,argv	The parameters that were passed the the "main" function.

Returns

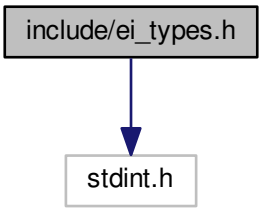
An error code: 0 means ok, 1 means error.

4.3 include/ei_types.h File Reference

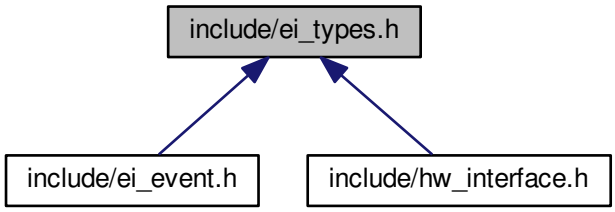
Type, constant, and global definitions for the ei library.

```
#include <stdint.h>
```

Include dependency graph for ei_types.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct ei_point_t

  A 2-D point with integer coordinates.
- struct ei_size_t

A 2-D size with integer dimensions.

- struct ei_rect_t

  A rectangle defined by its top-left corner, and its size.

- struct ei_linked_rect_t

  A rectangle plus a pointer to create a linked list.

- struct ei_linked_point_t

  A point plus a pointer to create a linked list.

- struct ei_color_t

  A color with transparency.

Typedefs

- typedef struct ei_linked_rect_t ei_linked_rect_t

  A rectangle plus a pointer to create a linked list.

- typedef struct ei_linked_point_t ei_linked_point_t

  A point plus a pointer to create a linked list.

- typedef void * ei_font_t

  An opaque type for handling fonts.

Enumerations

- enum ei_bool_t { **EI_FALSE** = 0, **EI_TRUE** = 1 }

  The boolean type used in the library.

- enum ei_anchor_t {
  ei_anc_none = 0, ei_anc_center, ei_anc_north, ei_anc_northeast,
  ei_anc_east, ei_anc_southeast, ei_anc_south, ei_anc_southwest,
  ei_anc_west, ei_anc_northwest }

  Identifies one particular point of a rectangle.

- enum ei_relief_t { ei_relief_none = 0, ei_relief_raised, ei_relief_sunken }

  Type of relief.

- enum ei_axis_set_t { ei_axis_none = 0, ei_axis_x, ei_axis_y, ei_axis_both }

  Set of axis.

Variables

- ei_font_t ei_default_font

  The default font used in widgets.

4.3.1 Detailed Description

Type, constant, and global definitions for the ei library.

4.3.2 Typedef Documentation

4.3.2.1 typedef void* ei_font_t

An opaque type for handling fonts.

Fonts are created by calling hw_text_font_create and released by calling hw_text_font_free.

4.3.3 Enumeration Type Documentation

4.3.3.1 enum ei_anchor_t

Identifies one particular point of a rectangle.

Enumerator

ei_anc_none No anchor defined.

ei_anc_center Anchor in the center.

ei_anc_north Anchor on the top side, centered horizontally.

ei_anc_northeast Anchor on the top-right corner.

ei_anc_east Anchor on the right side, centered vertically.

ei_anc_southeast Anchor on the bottom-right corner.

ei_anc_south Anchor on the bottom side, centered horizontally.

ei_anc_southwest Anchor on the bottom-left corner.

ei_anc_west Anchor on the left side, centered vertically.

ei_anc_northwest Anchor on the top-left corner.

4.3.3.2 enum ei_axis_set_t

Set of axis.

Enumerator

ei_axis_none No axis.

ei_axis_x Horizontal axis.

ei_axis_y Vertical axis.

ei_axis_both Both horizontal and vertical axis.

4.3.3.3 enum ei_relief_t

Type of relief.

Enumerator

ei_relief_none No relief (i.e. flat).

ei_relief_raised Above the screen.

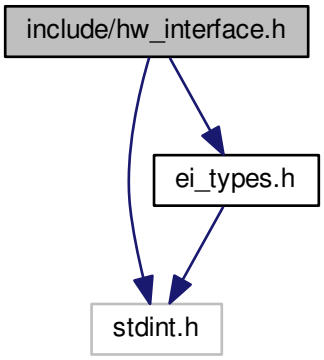
ei_relief_sunken Inside the screen.

4.4 include/hw_interface.h File Reference

Low level interface with the graphic hadware. This interface is based on the SDL library.

```
#include <stdint.h>
#include "ei_types.h"
```

Include dependency graph for hw_interface.h:



Typedefs

- typedef void * ei_surface_t

  Surface hidden type. A surface represents a 2 dimentional array of pixels where drawing can be done. The displayed screen itself is represented by a surface, it is accessed by hw_create_window. Other "offscreen" surfaces can be created by hw_surface_create.

Functions

- void hw_init ()

  Initialises access to the low-level operating system services.
- void hw_quit ()

  Closes the access to the low-level operating system services.
- ei_surface_t hw_create_window (ei_size_t *size, const ei_bool_t fullScreen)

  Opens the main graphical window of the application.
- ei_surface_t hw_surface_create (const ei_surface_t root, const ei_size_t *size)

  Allocates an off-screen drawing surface.
- void hw_surface_free (ei_surface_t surface)

  Frees a surface allocated by hw_surface_create. This must be called on an unlocked surface (see hw_surface_unlock).
- void hw_surface_lock (ei_surface_t surface)

  Gains exclusive access to a surface. Every call to this function must be matched by a call to hw_surface_unlock. The address of the pixel buffer may change while the surface is unlocked. Thus, hw_surface_get_buffer must called after each call to this function.
- void hw_surface_unlock (ei_surface_t surface)

Releases the exclusive access to a surface that was locked by hw_surface_lock.

- ei_size_t hw_surface_get_size (const ei_surface_t surface)

  Returns the size of a surface.
- void hw_surface_update_rects (const ei_linked_rect_t *rects)

  Requests that a list of rectangular regions of the root surface be updated on screen.
- ei_rect_t hw_surface_get_rect (const ei_surface_t surface)

  Returns the rectangle of a surface (origin and size).
- ei_color_t hw_get_pixel (const ei_surface_t surface, const ei_point_t pos)

  Returns the color of the pixel at pos.
- void hw_put_pixel (const ei_surface_t surface, const ei_point_t pos, const ei_color_t color)

  Draw the pixel of the surface at pos.
- ei_surface_t hw_image_load (const char *filename)

  Creates a surface and loads into it an image read from a file. The caller is responsible to release this surface (hw_surface_free) when it is no more needed.
- ei_font_t hw_text_font_create (const char *filename, int size)

  Creates a font that can be used to render text. The font must be freed by calling hw_text_font_free.
- void hw_text_font_free (ei_font_t font)

  Frees a font created by hw_text_font_create.
- void hw_text_compute_size (const char *text, const ei_font_t font, int *width, int *height)

  Computes the size of a text surface givent the font and the text.
- ei_surface_t hw_text_create_surface (const char *text, const ei_font_t font, const ei_color_t *color)

  Creates a surface containing a text. The size of the created surface is just big enough to contain the text. The caller is responsible to release this surface (hw_surface_free) when it is no more needed.
- void hw_wait (int s_delay)

  Waits for the specified number of seconds. This tells the system to pause the current thread for the given amount of time.
- void hw_event_wait_next (struct ei_event_t *event)

  Lets this processus sleep until a new event is available.
- ei_bool_t hw_event_post_app (void *user_param)

  Put an application-generated event on the event queue. This will cause hw_event_wait_next to wake.
- double hw_now ()

  Returns the number of seconds since the library was initialised by hw_init. Can be used to measure elpased time between to calls.

Variables

- const int **EI_MOUSEBUTTON_LEFT**
- const int **EI_MOUSEBUTTON_MIDDLE**
- const int **EI_MOUSEBUTTON_RIGHT**

4.4.1 Detailed Description

Low level interface with the graphic hadware. This interface is based on the SDL library.

Created by François Bérard on 30.12.11. Copyright 2011 Ensimag. All rights reserved.

4.4.2 Function Documentation

4.4.2.1 ei_surface_t hw_create_window (ei_size_t * *size*, const ei_bool_t *fullScreen*)

Opens the main graphical window of the application.

Parameters

size	Number of horizontal and vertical pixels.
fullScreen	If true, opens the window in full screen. Otherwise opens a floating window.

Returns

The unlocked drawing surface (see hw_surface_lock). This surface should not be freed by calling hw_surface_free, it is freed when releasing access to the low-level services by calling hw_quit.

4.4.2.2 ei_bool_t hw_event_post_app (void * *user_param*)

Put an application-generated event on the event queue. This will cause hw_event_wait_next to wake.

Parameters

user_param	The user parameter that will be retrievable in the event.

4.4.2.3 void hw_event_wait_next (struct ei_event_t * *event*)

Lets this processus sleep until a new event is available.

Parameters

event	Where to store the new event. The structure must be allocated by the caller. On return, the structure is filled with informations about the new event.

4.4.2.4 ei_color_t hw_get_pixel (const ei_surface_t *surface*, const ei_point_t *pos*)

Returns the color of the pixel at pos.

Parameters

surface	The surface on which to draw.
pos	The position at which to draw.
color	The color to draw.

Returns

The color of teh pixel.

4.4.2.5 ei_surface_t hw_image_load (const char * *filename*)

Creates a surface and loads into it an image read from a file. The caller is responsible to release this surface (hw_surface_free) when it is no more needed.

Parameters

filename	The name of the file containing the image. The file can be .bmp, .png, .jpg, .tga

Returns

A new unlocked surface containing the image.

4.4.2.6 double hw_now ( )

Returns the number of seconds since the library was initialised by hw_init. Can be used to measure elpased time between to calls.

Returns

The current time, in seconds.

4.4.2.7 void hw_put_pixel (const ei_surface_t *surface*, const ei_point_t *pos*, const ei_color_t *color*)

Draw the pixel of the surface at pos.

Parameters

surface	The surface on which to draw.
pos	The position at which to draw.
color	The color to draw.

4.4.2.8 ei_surface_t hw_surface_create (const ei_surface_t *root*, const ei_size_t * *size*)

Allocates an off-screen drawing surface.

Parameters

root	The root window which channel indices will be used. This insures that the offscreen uses the same channel indices (Red, Green, Blue, Alpha) as the root surface.
size	Number of horizontal and vertical pixels.
force_alpha	If true, then the returned surface will use an alpha channel regardless of root having an alpha channel or not.

Returns

The unlocked drawing surface (see hw_surface_lock). The surface should be freed by calling hw_surface_↩ free.

4.4.2.9 void hw_surface_free (ei_surface_t *surface*)

Frees a surface allocated by hw_surface_create. This must be called on an unlocked surface (see hw_surface_unlock).

Parameters

surface	The surface to be freed.

4.4.2.10 ei_rect_t hw_surface_get_rect (const ei_surface_t *surface*)

Returns the rectangle of a surface (origin and size).

Parameters

surface	The surface which rectangle is requested.

Returns

The rectangle of the surface.

4.4.2.11 ei_size_t hw_surface_get_size (const ei_surface_t *surface*)

Returns the size of a surface.

Parameters

surface	The surface which size is requested.

Returns

The size of the surface.

4.4.2.12 void hw_surface_lock (ei_surface_t *surface*)

Gains exclusive access to a surface. Every call to this function must be matched by a call to hw_surface_unlock. The address of the pixel buffer may change while the surface is unlocked. Thus, hw_surface_get_buffer must called after each call to this function.

Parameters

surface	The surface to lock.

4.4.2.13 void hw_surface_unlock (ei_surface_t *surface*)

Releases the exclusive access to a surface that was locked by hw_surface_lock.

Parameters

surface	The surface to unlock.

4.4.2.14 void hw_surface_update_rects (const ei_linked_rect_t * *rects*)

Requests that a list of rectangular regions of the root surface be updated on screen.

Parameters

rects	The list of rectangle to be updated on screen. If NULL, then the entire surface is updated.

4.4.2.15 void hw_text_compute_size (const char * *text,* const ei_font_t *font,* int * *width,* int * *height*)

Computes the size of a text surface givent the font and the text.

Parameters

text	The string of the message.
font	The font used to render the text.
width,height	Addresses where to store the computed width and height of the text surface.

4.4.2.16 ei_surface_t hw_text_create_surface (const char * *text,* const ei_font_t *font,* const ei_color_t * *color*)

Creates a surface containing a text. The size of the created surface is just big enough to contain the text. The caller is responsible to release this surface (hw_surface_free) when it is no more needed.

Parameters

text	The string of the message.
font	The font used to render the text.
color	The text color. The alpha parameter is not used. However, the text is rendered with alpha blending to smooth the curves of the letters (anti-aliasing).

Returns

A newly created unlocked surface containing an anti-aliased rendering of the text. The anti-aliasing is implemented with the alpha channel of the surface: pixels on the text's boundaries have some transparency.

4.4.2.17 ei_font_t hw_text_font_create (const char * *filename*, int *size*)

Creates a font that can be used to render text. The font must be freed by calling hw_text_font_free.

Parameters

filename	The path to the file containing the ttf font definition. Can be relative.
style	The style of the font (normal, bold, ...).
size	The size of the characters in pixels.

Returns

The font.

4.4.2.18 void hw_text_font_free (ei_font_t *font*)

Frees a font created by hw_text_font_create.

Parameters

font	The font to be freed.

4.4.2.19 void hw_wait (int *s_delay*)

Waits for the specified number of seconds. This tells the system to pause the current thread for the given amount of time.

Parameters

s_delay	The amount of time, in seconds, to wait.

Index