Projet C - Interaction Graphique

Generated by Doxygen 1.6.1

Thu May 26 16:34:57 2016

Contents

Index

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.

6.11.3.22 ei_font_t hw_text_font_create (const char * *filename*, ei_fontstyle_t *style*, 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.

6.11.3.23 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.

6.11.4 Variable Documentation

6.11.4.1 const int EI_MOUSEBUTTON_LEFT

6.11.4.2 const int EI_MOUSEBUTTON_MIDDLE

6.11.4.3 const int EI_MOUSEBUTTON_RIGHT

Chapter 1

Directory Hierarchy

1.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

6.11.3.18 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.

6.11.3.19 void hw_surface_update_rects (ei_surface_t *surface*, const ei_linked_rect_t * *rects*)

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

Parameters:

surface The surface returned by hw_create_window. This function can only be called on *unlocked surfaces* (hw_surface_unlock).

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

6.11.3.20 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.

6.11.3.21 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).

6.11.3.14 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.

6.11.3.15 ei_bool_t hw_surface_has_alpha (ei_surface_t *surface*)

Tells if a surface manages transparency, i.e. if the surface has an alpha channel.

Returns:

A boolean: EI_TRUE means that the surface has an alpha (transparency) channel, EI_FALSE means it does not.

6.11.3.16 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.

6.11.3.17 void hw_surface_set_origin (ei_surface_t *surface*, const ei_point_t *origin*)

Sets the coordinates of the first pixel of the surface's memory. By default, the coordinates of the first pixel are (0, 0). This can be changed by a call to this function. After a call to this function, the function hw_surface_get_buffer returns a different address than before.

Parameters:

surface The surface which origin must be changed.

origin The new coordinates of the first pixel of the surface.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Returns:

The unlocked drawing surface (see hw_surface_lock). The surface should be freed by calling hw_surface_free.

6.11.3.10 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.

6.11.3.11 uint8_t* hw_surface_get_buffer (const ei_surface_t *surface*)

Returns a pointer to the address of the pixel at coordinated (0, 0) of a surface. This is usually the first pixel of the surface's memory. But after a call to hw_surface_set_origin, the (0, 0) pixel may point within the surface memory or not. Pixels are ordered by horizontal lines, from top to bottom, and from left to right whitin lines. The pixel buffer of a surface may be moved when the surface is unlocked (hw_surface_unlock), you must thus call this function after each call to hw_surface_lock.

Parameters:

surface The surface from which the pixel address is returned.

6.11.3.12 void hw_surface_get_channel_indices (ei_surface_t *surface*, int * *ir*, int * *ig*, int * *ib*, int * *ia*)

Returns the R, G, B, Alpha channel indices of a surface.

Parameters:

surface The surface.

ir,ig,ib,ia Where to store the resulting indices.

6.11.3.13 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.

6.11.3.5 ei_surface_t hw_image_load (const char * *filename*, ei_surface_t *channels*)

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 .png, .tiff, .jpg, etc.

channels A surface to define channel ordering: the newly created surface that is returned by this function will have the same channel order as this surface.

Returns:

A new unlocked surface containing the image.

6.11.3.6 void hw_init ()

Initialises access to the low-level operating system services.

6.11.3.7 double hw_now ()

Returns the current time, in seconds, from some arbitraty origin. Can be used to measure elpased time between to calls.

Returns:

The current time, in seconds.

6.11.3.8 void hw_quit ()

Closes the access to the low-level operating system services.

6.11.3.9 ei_surface_t hw_surface_create (const ei_surface_t *root*, const ei_size_t * *size*, ei_bool_t *force_alpha*)

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.

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

6.11.3 Function Documentation

6.11.3.1 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.

6.11.3.2 int 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.

6.11.3.3 void hw_event_schedule_app (int *ms_delay*, void * *user_param*)

Schedule an application-generated event to be posted after some amount of time. This will cause hw_event_wait_next to wake after this amount of time.

Parameters:

ms_delay The amount of time, in milliseconds, to wait before the event is posted in the event queue.

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

6.11.3.4 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.

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.

- void hw_event_wait_next (struct ei_event_t *event)

  Lets this processus sleep until a new event is available.

- int 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.

- void hw_event_schedule_app (int ms_delay, void *user_param)

  Schedule an application-generated event to be posted after some amount of time. This will cause hw_event_wait_next to wake after this amount of time.

- double hw_now ()

  Returns the current time, in seconds, from some arbitraty origin. 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

6.11.1 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.

Definition in file hw_interface.h.

6.11.2 Typedef Documentation

6.11.2.1 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.

Definition at line 30 of file hw_interface.h.

Chapter 4

Directory Documentation

4.1 include/ Directory Reference

Files

- file ei_application.h

  Manages the main steps of a graphical application: initialization, main window, main loop, quitting, resource freeing.

- file ei_draw.h

  Graphical primitives to draw lines, polygons, text, and operation of drawing surfaces.

- file ei_event.h

  Allows the binding and unbinding of callbacks to events.

- file ei_geometrymanager.h

  Manages the positionning and sizing of widgets on the screen.

- file ei_main.h

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

- file ei_parser.h
- file ei_types.h

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

- file ei_utils.h

  General purpose utility functions: creation of points and sizes, and arithmetics on them.

- file ei_widget.h

  API for widgets management: creation, configuration, hierarchy, redisplay.

- file ei_widgetclass.h
- file hw_interface.h

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

- void hw_surface_get_channel_indices (ei_surface_t surface, int *ir, int *ig, int *ib, int *ia)

  Returns the R, G, B, Alpha channel indices of a surface.

- void hw_surface_set_origin (ei_surface_t surface, const ei_point_t origin)

  Sets the coordinates of the first pixel of the surface's memory. By default, the coordinates of the first pixel are (0, 0). This can be changed by a call to this function. After a call to this function, the function hw_surface_get_buffer returns a different address than before.

- uint8_t * hw_surface_get_buffer (const ei_surface_t surface)

  Returns a pointer to the address of the pixel at coordinated (0, 0) of a surface. This is usually the first pixel of the surface's memory. But after a call to hw_surface_set_origin, the (0, 0) pixel may point within the surface memory or not. Pixels are ordered by horizontal lines, from top to bottom, and from left to right whitin lines. The pixel buffer of a surface may be moved when the surface is unlocked (hw_surface_unlock), you must thus call this function after each call to hw_surface_lock.

- ei_size_t hw_surface_get_size (const ei_surface_t surface)

  Returns the size of a surface.

- ei_rect_t hw_surface_get_rect (const ei_surface_t surface)

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

- ei_bool_t hw_surface_has_alpha (ei_surface_t surface)

  Tells if a surface manages transparency, i.e. if the surface has an alpha channel.

- ei_font_t hw_text_font_create (const char *filename, ei_fontstyle_t style, 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.

- ei_surface_t hw_image_load (const char *filename, ei_surface_t channels)

6.11 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"`

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, ei_bool_t force_alpha)

  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.

- void hw_surface_update_rects (ei_surface_t surface, const ei_linked_rect_t *rects)

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

Chapter 5

Data Structure Documentation

5.1 ei_app_event_t Struct Reference

The event parameter for application defined event types.

`#include <ei_event.h>`

Data Fields

- void * user_param

5.1.1 Detailed Description

The event parameter for application defined event types.

Definition at line 101 of file ei_event.h.

5.1.2 Field Documentation

5.1.2.1 void* ei_app_event_t::user_param

Definition at line 102 of file ei_event.h.

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

- ei_event.h

5.2 ei_color_t Struct Reference

A color with transparency.

`#include <ei_types.h>`

Data Fields

- 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
  The transparency of the color. 0 is invisible, 255 is totally opaque.

5.2.1 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.

Definition at line 78 of file ei_types.h.

5.2.2 Field Documentation

5.2.2.1 unsigned char ei_color_t::alpha

The transparency of the color. 0 is invisible, 255 is totally opaque.

Definition at line 82 of file ei_types.h.

5.2.2.2 unsigned char ei_color_t::blue

The blue component of the color.

Definition at line 81 of file ei_types.h.

5.2.2.3 unsigned char ei_color_t::green

The green component of the color.

Definition at line 80 of file ei_types.h.

configure.

6.10.2.2 void ei_frame_register_class ()

Registers the "frame" widget class in the program. This must be called only once before widgets of the class "frame" can be created and configured with ei_frame_configure.

6.10.2.3 void ei_toplevel_register_class ()

Registers the "toplevel" widget class in the program. This must be called only once before widgets of the class "toplevel" can be created and configured with ei_toplevel_configure.

6.10.2.4 ei_widgetclass_t* ei_widgetclass_from_name (ei_widgetclass_name_t *name*)

Returns the structure describing a class, from its name.

Parameters:

name The name of the class of widget.

Returns:

The structure describing the class.

6.10.2.5 void ei_widgetclass_register (ei_widgetclass_t * *widgetclass*)

Registers a class to the program so that widgets of this class can be created. This must be done only once per widged class in the application.

Parameters:

widgetclass The structure describing the class.

6.10.2.6 static char * ei_widgetclass_stringname (ei_widgetclass_name_t *name*) [inline, static]

Returns the string of the name of a class.

Parameters:

name The class name.

Returns:

The string representing the name of the class.

Definition at line 146 of file ei_widgetclass.h.

6.10.1.3 typedef void(* ei_widgetclass_geomnotifyfunc_t)(struct ei_widget_t *widget, ei_rect_t rect)

A function that is called to notify the widget that its geometry has been modified by its geometry manager. Can set to NULL in ei_widgetclass_t.

Parameters:

widget The widget instance to notify of a geometry change.

rect The new rectangular screen location of the widget (i.e. = widget->screen_location).

Definition at line 75 of file ei_widgetclass.h.

6.10.1.4 typedef char ei_widgetclass_name_t[20]

A name of a class of widget.

Definition at line 23 of file ei_widgetclass.h.

6.10.1.5 typedef void(* ei_widgetclass_releasefunc_t)(struct ei_widget_t *widget)

A function that releases the memory used by a widget before it is destroyed. The ei_widget_t structure itself, passed as parameter, must *not* by freed by these functions. Can be set to NULL in ei_widgetclass_t if no memory is used by a class of widget.

Parameters:

widget The widget which resources are to be freed.

Definition at line 44 of file ei_widgetclass.h.

6.10.1.6 typedef void(* ei_widgetclass_setdefaultsfunc_t)(struct ei_widget_t *widget)

A function that sets the default values for a class.

Parameters:

widget A pointer to the widget instance to intialize.

Definition at line 65 of file ei_widgetclass.h.

6.10.2 Function Documentation

6.10.2.1 void ei_button_register_class ()

Registers the "button" widget class in the program. This must be called only once before widgets of the class "button" can be created and configured with ei_button_-

5.2.2.4 unsigned char ei_color_t::red

The red component of the color.

Definition at line 79 of file ei_types.h.

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

- ei_types.h

5.3 ei_event_t Struct Reference

Describes an event.

`#include <ei_event.h>`

Data Fields

- ei_eventtype_t type
  The type of the event.
- union {
  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_app_event_t application
  Event parameters for application-related events (see ei_app_event_t).
  } param

5.3.1 Detailed Description

Describes an event.

Definition at line 108 of file ei_event.h.

5.3.2 Field Documentation

5.3.2.1 ei_app_event_t ei_event_t::application

Event parameters for application-related events (see ei_app_event_t).

Definition at line 113 of file ei_event.h.

5.3.2.2 ei_key_event_t ei_event_t::key

Event parameters for keyboard-related events (see ei_key_event_t).

Definition at line 111 of file ei_event.h.

5.3.2.3 ei_mouse_event_t ei_event_t::mouse

Event parameters for mouse-related events (see ei_mouse_event_t).

Definition at line 112 of file ei_event.h.

Registers a class to the program so that widgets of this class can be created. This must be done only once per widged class in the application.

- ei_widgetclass_t * ei_widgetclass_from_name (ei_widgetclass_name_t name)
  Returns the structure describing a class, from its name.
- void ei_frame_register_class ()
  Registers the "frame" widget class in the program. This must be called only once before widgets of the class "frame" can be created and configured with ei_frame_configure.
- void ei_button_register_class ()
  Registers the "button" widget class in the program. This must be called only once before widgets of the class "button" can be created and configured with ei_button_configure.
- void ei_toplevel_register_class ()
  Registers the "toplevel" widget class in the program. This must be called only once before widgets of the class "toplevel" can be created and configured with ei_toplevel_configure.

6.10.1 Typedef Documentation

6.10.1.1 typedef void*(* ei_widgetclass_allocfunc_t)()

A function that allocates a block of memory that is big enough to store the attributes of a widget of a class. After allocation, the function *must* initialize the memory to 0.

Returns:

A block of memory with all bytes set to 0.

Definition at line 34 of file ei_widgetclass.h.

6.10.1.2 typedef void(* ei_widgetclass_drawfunc_t)(struct ei_widget_t *widget, ei_surface_t surface, ei_surface_t pick_surface, ei_rect_t *clipper)

A function that draws widgets of a class.

Parameters:

widget A pointer to the widget instance to draw.

surface Where to draw the widget. The actual location of the widget in the surface is stored in its "screen_location" field.

clipper If not NULL, the drawing is restricted within this rectangle (expressed in the surface reference frame).

Definition at line 55 of file ei_widgetclass.h.

6.10 ei_widgetclass.h File Reference

```
#include "hw_interface.h"
#include "ei_draw.h"
```

Data Structures

- struct ei_widgetclass_t

  A structure that stores information about a class of widgets.

Typedefs

- typedef char ei_widgetclass_name_t [20]

  A name of a class of widget.

- typedef void *(* ei_widgetclass_allocfunc_t)()

  *A function that allocates a block of memory that is big enough to store the attributes of a widget of a class. After allocation, the function *must* initialize the memory to 0.*

- typedef void(* ei_widgetclass_releasefunc_t)(struct ei_widget_t *widget)

  *A function that releases the memory used by a widget before it is destroyed. The ei_widget_t structure itself, passed as parameter, must *not* by freed by these functions. Can be set to NULL in ei_widgetclass_t if no memory is used by a class of widget.*

- typedef void(* ei_widgetclass_drawfunc_t)(struct ei_widget_t *widget, ei_surface_t surface, ei_surface_t pick_surface, ei_rect_t *clipper)

  A function that draws widgets of a class.

- typedef void(* ei_widgetclass_setdefaultsfunc_t)(struct ei_widget_t *widget)

  A function that sets the default values for a class.

- typedef void(* ei_widgetclass_geomnotifyfunc_t)(struct ei_widget_t *widget, ei_rect_t rect)

  A function that is called to notify the widget that its geometry has been modified by its geometry manager. Can set to NULL in ei_widgetclass_t.

Functions

- static char * ei_widgetclass_stringname (ei_widgetclass_name_t name)

  Returns the string of the name of a class.

- void ei_widgetclass_register (ei_widgetclass_t *widgetclass)

5.3.2.4 union { ... } ei_event_t::param

5.3.2.5 ei_eventtype_t ei_event_t::type

The type of the event.

Definition at line 109 of file ei_event.h.

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

- ei_event.h

5.4 ei_geometry_param_t Struct Reference

A structure that stores information about the geometry manager managing a widget, and the widget's geometry management parameters. This a the generic type. Each geometry manager adds field after "manager".

`#include <ei_geometrymanager.h>`

Data Fields

- ei_geometrymanager_t * manager

5.4.1 Detailed Description

A structure that stores information about the geometry manager managing a widget, and the widget's geometry management parameters. This a the generic type. Each geometry manager adds field after "manager".

Definition at line 56 of file ei_geometrymanager.h.

5.4.2 Field Documentation

5.4.2.1 ei_geometrymanager_t* ei_geometry_param_t::manager

Points to the geometry manager's structure

Definition at line 57 of file ei_geometrymanager.h.

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

- ei_geometrymanager.h

Parameters:

class_name The name of the class of the widget that is to be created.

parent A pointer to the parent widget. Can not be NULL.

Returns:

The newly created widget, or NULL if there was an error.

6.9.3.5 void ei_widget_destroy (ei_widget_t * *widget*)

Destroys a widget. Removes it from screen if it is managed by a geometry manager. Destroys all its descendants.

Parameters:

widget The widget that is to be destroyed.

6.9.3.6 ei_widget_t* ei_widget_pick (ei_point_t * *where*)

Returns the widget that is at a given location on screen.

Parameters:

where The location on screen, expressed in the root window coordinates.

Returns:

The top-most widget at this location, or NULL if there is no widget at this location (except for the root widget).

6.9.4 Variable Documentation

6.9.4.1 const int k_default_button_border_width = 4 [static]

The default border width of button widgets.

Definition at line 158 of file ei_widget.h.

6.9.4.2 const int k_default_button_corner_radius = 10 [static]

The default corner radius of button widgets.

Definition at line 159 of file ei_widget.h.

text The text to display in the widget, or NULL. Only one of the parameter "text" and "img" should be used (i.e. non-NULL). Defaults to NULL.

text_font The font used to display the text. Defaults to ei_default_font.

text_color The color used to display the text. Defaults to ei_font_default_color.

text_anchor The anchor of the text, i.e. where it is placed whithin the widget when the size of the widget is bigger than the size of the text. Defaults to ei_anc_center.

img The image to display in the widget, or NULL. Any surface can be used, but usually a surface returned by hw_image_load. Only one of the parameter "text" and "img" should be used (i.e. non-NULL). Defaults to NULL.

img_rect If not NULL, this rectangle defines a subpart of "img" to use as the image displayed in the widget. Defaults to NULL.

img_anchor The anchor of the image, i.e. where it is placed whithin the widget when the size of the widget is bigger than the size of the image. Defaults to ei_anc_center.

6.9.3.3 void ei_toplevel_configure (ei_widget_t * *widget*, ei_size_t * *requested_size*, ei_color_t * *color*, int * *border_width*, char ** *title*, ei_bool_t * *closable*, ei_axis_set_t * *resizable*, ei_size_t ** *min_size*)

Configures the attributes of widgets of the class "toplevel".

Parameters:

widget The widget to configure.

requested_size The content size requested for this widget, this does not include the decorations (border, title bar). The geometry manager may override this size due to other constraints. Defaults to (320x240).

color The color of the background of the content of the widget. Defaults to ei_default_background_color.

border_width The width in pixel of the border of the widget. Defaults to 4.

title The string title diplayed in the title bar. Defaults to "Toplevel".

closable If true, the toplevel is closable by the user, the toplevel must show a close button in its title bar. Defaults to EI_TRUE.

resizable Defines if the widget can be resized horizontally and/or vertically by the user. Defaults to ei_axis_both.

min_size For resizable widgets, defines the minimum size. Defaults to (160, 120).

6.9.3.4 ei_widget_t* ei_widget_create (ei_widgetclass_name_t *class_name*, ei_widget_t * *parent*)

Creates a new instance of a widget of some particular class, as a descendant of an existing widget. The widget is not displayed on screen until it is managed by a geometry manager. The widget should be released by calling ei_widget_destroy when no more needed.

5.5 ei_geometrymanager_t Struct Reference

The structure that stores information about a geometry manager.

`#include <ei_geometrymanager.h>`

Data Fields

- ei_geometrymanager_name_t name
- ei_geometrymanager_runfunc_t runfunc
- ei_geometrymanager_releasefunc_t releasefunc
- struct ei_geometrymanager_t * next

5.5.1 Detailed Description

The structure that stores information about a geometry manager.

Definition at line 44 of file ei_geometrymanager.h.

5.5.2 Field Documentation

5.5.2.1 ei_geometrymanager_name_t ei_geometrymanager_t::name

Definition at line 45 of file ei_geometrymanager.h.

5.5.2.2 struct ei_geometrymanager_t* ei_geometrymanager_t::next [read]

Definition at line 48 of file ei_geometrymanager.h.

5.5.2.3 ei_geometrymanager_releasefunc_t ei_geometrymanager_t::releasefunc

Definition at line 47 of file ei_geometrymanager.h.

5.5.2.4 ei_geometrymanager_runfunc_t ei_geometrymanager_t::runfunc

Definition at line 46 of file ei_geometrymanager.h.

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

- ei_geometrymanager.h

5.6 ei_key_event_t Struct Reference

The event parameter for keyboard-related event types.

`#include <ei_event.h>`

Data Fields

- SDLKey key_sym

  The keyboard key symbol (see SDL_keysym::h).

- ei_modifier_mask_t modifier_mask

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

5.6.1 Detailed Description

The event parameter for keyboard-related event types.

Definition at line 85 of file ei_event.h.

5.6.2 Field Documentation

5.6.2.1 SDLKey ei_key_event_t::key_sym

The keyboard key symbol (see SDL_keysym::h).

Definition at line 86 of file ei_event.h.

5.6.2.2 ei_modifier_mask_t ei_key_event_t::modifier_mask

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

Definition at line 87 of file ei_event.h.

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

- ei_event.h

6.9.3 Function Documentation

6.9.3.1 void ei_button_configure (ei_widget_t * *widget*, ei_size_t * *requested_size*, const ei_color_t * *color*, int * *border_width*, int * *corner_radius*, ei_relief_t * *relief*, char ** *text*, ei_font_t * *text_font*, ei_color_t * *text_color*, ei_anchor_t * *text_anchor*, ei_surface_t * *img*, ei_rect_t ** *img_rect*, ei_anchor_t * *img_anchor*, ei_callback_t * *callback*, void ** *user_param*)

Configures the attributes of widgets of the class "button".

Parameters:

widget,requested_size,color,border_width,relief,text,text_font,text_color,text_anchor,img,img_rect,img_anchor See the parameter definition of ei_frame_configure. The only difference is that relief defaults to ei_relief_raised and border_width defaults to k_default_button_border_width.

corner_radius The radius (in pixels) of the rounded corners of the button. 0 means straight corners. Defaults to k_default_button_corner_radius.

callback The callback function to call when the user clicks on the button. Defaults to NULL (no callback).

user_param A programmer supplied parameter that will be passed to the callback when called. Defaults to NULL.

6.9.3.2 void ei_frame_configure (ei_widget_t * *widget*, ei_size_t * *requested_size*, const ei_color_t * *color*, int * *border_width*, ei_relief_t * *relief*, char ** *text*, ei_font_t * *text_font*, ei_color_t * *text_color*, ei_anchor_t * *text_anchor*, ei_surface_t * *img*, ei_rect_t ** *img_rect*, ei_anchor_t * *img_anchor*)

Configures the attributes of widgets of the class "frame". Parameters obey the "default" protocol: if a parameter is "NULL" and it has never been defined before, then a default value should be used (default values are specified for each parameter). If the parameter is "NULL" but was defined on a previous call, then its value must not be changed.

Parameters:

widget The widget to configure.

requested_size The size requested for this widget. The geometry manager may override this size due to other constraints. Defaults to the "natural size" of the widget, ie. big enough to display the text or the image, or (0, 0) if the widget has no text and no image.

color The color of the background of the widget. Defaults to ei_default_background_color.

border_width The width in pixel of the border decoration of the widget. The final appearance depends on the "relief" parameter. Defaults to 0.

relief Appearance of the border of the widget. Defaults to ei_relief_none.

Configures the attributes of widgets of the class "button".

- void ei_toplevel_configure (ei_widget_t *widget, ei_size_t *requested_size, ei_color_t *color, int *border_width, char **title, ei_bool_t *closable, ei_axis_set_t *resizable, ei_size_t **min_size)

  Configures the attributes of widgets of the class "toplevel".

Variables

- static const int k_default_button_border_width = 4

  The default border width of button widgets.

- static const int k_default_button_corner_radius = 10

  The default corner radius of button widgets.

6.9.1 Detailed Description

API for widgets management: creation, configuration, hierarchy, redisplay. Created by François Bérard on 30.12.11. Copyright 2011 Ensimag. All rights reserved.

Definition in file ei_widget.h.

6.9.2 Typedef Documentation

6.9.2.1 typedef ei_bool_t(* ei_callback_t)(ei_widget_t *widget, struct ei_event_t *event, void *user_param)

A function that is called in response to a user event. Usually passed as a parameter to ei_bind.

Parameters:

widget The widget for which the event was generated.

event The event containing all its parameters (type, etc.)

user_param The user parameters that was provided by the caller when registering this callback.

Returns:

A boolean telling if the event was consumed by the callback or not. If TRUE, the library does not try to call other callbacks for this event. If FALSE, the library will call the next callback registered for this event, if any. Note: The callback may execute many operations and still return FALSE, or return TRUE without having done anything.

Definition at line 61 of file ei_widget.h.

5.7 ei_linked_point_t Struct Reference

A point plus a pointer to create a linked list.

`#include <ei_types.h>`

Data Fields

- ei_point_t point

  The point.

- struct ei_linked_point_t * next

  The pointer to the next element in the linked list.

5.7.1 Detailed Description

A point plus a pointer to create a linked list.

Definition at line 67 of file ei_types.h.

5.7.2 Field Documentation

5.7.2.1 struct ei_linked_point_t* ei_linked_point_t::next [read]

The pointer to the next element in the linked list.

Definition at line 69 of file ei_types.h.

5.7.2.2 ei_point_t ei_linked_point_t::point

The point.

Definition at line 68 of file ei_types.h.

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

- ei_types.h

5.8 ei_linked_rect_t Struct Reference

A rectangle plus a pointer to create a linked list.

`#include <ei_types.h>`

Data Fields

- ei_rect_t rect
  The rectangle.
- struct ei_linked_rect_t * next
  The pointer to the next element in the linked list.

5.8.1 Detailed Description

A rectangle plus a pointer to create a linked list.

Definition at line 59 of file ei_types.h.

5.8.2 Field Documentation

5.8.2.1 struct ei_linked_rect_t* ei_linked_rect_t::next [read]

The pointer to the next element in the linked list.

Definition at line 61 of file ei_types.h.

5.8.2.2 ei_rect_t ei_linked_rect_t::rect

The rectangle.

Definition at line 60 of file ei_types.h.

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

- ei_types.h

6.9 ei_widget.h File Reference

API for widgets management: creation, configuration, hierarchy, redisplay.
`#include "ei_draw.h"`

`#include "ei_widgetclass.h"`

Data Structures

- struct ei_widget_t
  Fields common to all types of widget. Every widget classes specializes this base class by adding its own fields.

Typedefs

- typedef ei_bool_t(* ei_callback_t)(ei_widget_t *widget, struct ei_event_t *event, void *user_param)
  A function that is called in response to a user event. Usually passed as a parameter to ei_bind.

Functions

- ei_widget_t * ei_widget_create (ei_widgetclass_name_t class_name, ei_widget_t *parent)
  Creates a new instance of a widget of some particular class, as a descendant of an existing widget.
- void ei_widget_destroy (ei_widget_t *widget)
  Destroys a widget. Removes it from screen if it is managed by a geometry manager. Destroys all its descendants.
- ei_widget_t * ei_widget_pick (ei_point_t *where)
  Returns the widget that is at a given location on screen.
- void ei_frame_configure (ei_widget_t *widget, ei_size_t *requested_size, const ei_color_t *color, int *border_width, ei_relief_t *relief, char **text, ei_font_t *text_font, ei_color_t *text_color, ei_anchor_t *text_anchor, ei_surface_t *img, ei_rect_t **img_rect, ei_anchor_t *img_anchor)
  Configures the attributes of widgets of the class "frame".
- void ei_button_configure (ei_widget_t *widget, ei_size_t *requested_size, const ei_color_t *color, int *border_width, int *corner_radius, ei_relief_t *relief, char **text, ei_font_t *text_font, ei_color_t *text_color, ei_anchor_t *text_anchor, ei_surface_t *img, ei_rect_t **img_rect, ei_anchor_t *img_anchor, ei_callback_t *callback, void **user_param)

6.8.2.7 static ei_rect_t ei_rect_zero () [inline, static]

Returns a ei_rect_t located in (0, 0) and of size (0, 0).

Definition at line 113 of file ei_utils.h.

6.8.2.8 static ei_size_t ei_size (int *width*, int *height*) [inline, static]

Returns a ei_size_t initialized with the width and height passed as parameters.

Definition at line 27 of file ei_utils.h.

6.8.2.9 static ei_size_t ei_size_add (ei_size_t *s1*, ei_size_t *s2*) [inline, static]

Returns a ei_size_t which components are the sum of the components of the two sizes passed as parameters.

Definition at line 92 of file ei_utils.h.

6.8.2.10 static ei_size_t ei_size_sub (ei_size_t *s1*, ei_size_t *s2*) [inline, static]

Returns a ei_size_t which components are the components of the first size parameter minus the components of the second size parameter.

Definition at line 103 of file ei_utils.h.

6.8.2.11 static ei_size_t ei_size_zero () [inline, static]

Returns a ei_size_t with width = 0 and height = 0;.

Definition at line 18 of file ei_utils.h.

5.9 ei_linked_tag_t Struct Reference

A tag and a pointer to create a linked list.

`#include <ei_event.h>`

Data Fields

- ei_tag_t tag
- struct ei_linked_tag_t * next

5.9.1 Detailed Description

A tag and a pointer to create a linked list.

Definition at line 26 of file ei_event.h.

5.9.2 Field Documentation

5.9.2.1 struct ei_linked_tag_t* ei_linked_tag_t::next [read]

Definition at line 28 of file ei_event.h.

5.9.2.2 ei_tag_t ei_linked_tag_t::tag

Definition at line 27 of file ei_event.h.

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

- ei_event.h

5.10 ei_mouse_event_t Struct Reference

The event parameter for mouse-related event types.

`#include <ei_event.h>`

Data Fields

- ei_point_t where

  Where the mouse pointer was at the time of the event.

- int button_number

  The number of the button that was pressed or released. Only valid for ei_ev_mouse_buttondown or ei_ev_mouse_buttonup event types.

5.10.1 Detailed Description

The event parameter for mouse-related event types.

Definition at line 93 of file ei_event.h.

5.10.2 Field Documentation

5.10.2.1 int ei_mouse_event_t::button_number

The number of the button that was pressed or released. Only valid for ei_ev_mouse_buttondown or ei_ev_mouse_buttonup event types.

Definition at line 95 of file ei_event.h.

5.10.2.2 ei_point_t ei_mouse_event_t::where

Where the mouse pointer was at the time of the event.

Definition at line 94 of file ei_event.h.

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

- ei_event.h

6.8.1 Detailed Description

General purpose utility functions: creation of points and sizes, and arithmetics on them.

Definition in file ei_utils.h.

6.8.2 Function Documentation

6.8.2.1 static ei_point_t ei_point (int *x*, int *y*) [inline, static]

Returns a ei_point_t initialized with the x and y passed as parameters.

Definition at line 47 of file ei_utils.h.

6.8.2.2 static ei_point_t ei_point_add (ei_point_t *p1*, ei_point_t *p2*) [inline, static]

Returns a ei_point_t which coordinates are the sum of the coordinates of the two points passed as parameters.

Definition at line 70 of file ei_utils.h.

6.8.2.3 static ei_point_t ei_point_neg (ei_point_t *point*) [inline, static]

Returns a ei_point_t which coordinates are opposite from the coordinate of the point passed as a parameter.

Definition at line 59 of file ei_utils.h.

6.8.2.4 static ei_point_t ei_point_sub (ei_point_t *p1*, ei_point_t *p2*) [inline, static]

Returns a ei_point_t which coordinates are the coordinates of the first point parameter minus the coordinates of the second point parameter.

Definition at line 81 of file ei_utils.h.

6.8.2.5 static ei_point_t ei_point_zero () [inline, static]

Returns a ei_point_t with x = 0 and y = 0;.

Definition at line 38 of file ei_utils.h.

6.8.2.6 static ei_rect_t ei_rect (ei_point_t *top_left*, ei_size_t *size*) [inline, static]

Returns a ei_rect_t which position and size are passed as parameters.

Definition at line 122 of file ei_utils.h.

6.8 ei_utils.h File Reference

General purpose utility functions: creation of points and sizes, and arithmetics on them.
`#include "ei_types.h"`

Functions

- static ei_size_t ei_size_zero ()

  Returns a ei_size_t with width = 0 and height = 0;.

- static ei_size_t ei_size (int width, int height)

  Returns a ei_size_t initialized with the width and height passed as parameters.

- static ei_point_t ei_point_zero ()

  Returns a ei_point_t with x = 0 and y = 0;.

- static ei_point_t ei_point (int x, int y)

  Returns a ei_point_t initialized with the x and y passed as parameters.

- static ei_point_t ei_point_neg (ei_point_t point)

  Returns a ei_point_t which coordinates are opposite from the coordinate of the point passed as a parameter.

- static ei_point_t ei_point_add (ei_point_t p1, ei_point_t p2)

  Returns a ei_point_t which coordinates are the sum of the coordinates of the two points passed as parameters.

- static ei_point_t ei_point_sub (ei_point_t p1, ei_point_t p2)

  Returns a ei_point_t which coordinates are the coordinates of the first point parameter minus the coordinates of the second point parameter.

- static ei_size_t ei_size_add (ei_size_t s1, ei_size_t s2)

  Returns a ei_size_t which components are the sum of the components of the two sizes passed as parameters.

- static ei_size_t ei_size_sub (ei_size_t s1, ei_size_t s2)

  Returns a ei_size_t which components are the components of the first size parameter minus the components of the second size parameter.

- static ei_rect_t ei_rect_zero ()

  Returns a ei_rect_t located in (0, 0) and of size (0, 0).

- static ei_rect_t ei_rect (ei_point_t top_left, ei_size_t size)

  Returns a ei_rect_t which position and size are passed as parameters.

5.11 ei_point_t Struct Reference

A 2-D point with integer coordinates.

`#include <ei_types.h>`

Data Fields

- int x

  The abscissa of the point. The origin is on the left side of the image.

- int y

  The ordinate of the point, the origin is at the top of the image, ordinates increase towards the bottom.

5.11.1 Detailed Description

A 2-D point with integer coordinates.

Definition at line 35 of file ei_types.h.

5.11.2 Field Documentation

5.11.2.1 int ei_point_t::x

The abscissa of the point. The origin is on the left side of the image.

Definition at line 36 of file ei_types.h.

5.11.2.2 int ei_point_t::y

The ordinate of the point, the origin is at the top of the image, ordinates increase towards the bottom.

Definition at line 37 of file ei_types.h.

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

- ei_types.h

5.12 ei_rect_t Struct Reference

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

`#include <ei_types.h>`

Data Fields

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

5.12.1 Detailed Description

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

Definition at line 51 of file ei_types.h.

5.12.2 Field Documentation

5.12.2.1 ei_size_t ei_rect_t::size

Size of the rectangle.

Definition at line 53 of file ei_types.h.

5.12.2.2 ei_point_t ei_rect_t::top_left

Coordinates of the top-left corner of the rectangle.

Definition at line 52 of file ei_types.h.

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

- ei_types.h

6.7.4.5 const int ei_font_default_size = 22 `[static]`

Default font color.

Definition at line 161 of file ei_types.h.

6.7.3.4 enum ei_fontstyle_t

Font style.

Enumerator:

ei_style_normal
ei_style_bold
ei_style_italic
ei_style_underline
ei_style_strikethrough

Definition at line 140 of file ei_types.h.

6.7.3.5 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.

Definition at line 111 of file ei_types.h.

6.7.4 Variable Documentation

6.7.4.1 const ei_color_t ei_default_background_color = { 0xA0, 0xA0, 0xA0, 0xff } [static]

The default background color of widgets.

Definition at line 88 of file ei_types.h.

6.7.4.2 ei_font_t ei_default_font

The default font used in widgets.

6.7.4.3 const char ei_default_font_filename[ ] = "misc/font.ttf" [static]

Definition at line 163 of file ei_types.h.

6.7.4.4 const ei_color_t ei_font_default_color = { 0x00, 0x00, 0x00, 0xff } [static]

Definition at line 162 of file ei_types.h.

5.13 ei_size_t Struct Reference

A 2-D size with integer dimensions.

`#include <ei_types.h>`

Data Fields

- int width
- int height

5.13.1 Detailed Description

A 2-D size with integer dimensions.

Definition at line 43 of file ei_types.h.

5.13.2 Field Documentation

5.13.2.1 int ei_size_t::height

Definition at line 45 of file ei_types.h.

5.13.2.2 int ei_size_t::width

Definition at line 44 of file ei_types.h.

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

- ei_types.h

5.14 ei_widget_t Struct Reference

Fields common to all types of widget. Every widget classes specializes this base class by adding its own fields.

`#include <ei_widget.h>`

Data Fields

- ei_widgetclass_t * wclass

  The class of widget of this widget. Avoid the field name "class" which is a keyword in C++.

- uint32_t pick_id

  Id of this widget in the picking offscreen.

- ei_color_t * pick_color

  pick_id encoded as a color.

- struct ei_widget_t * parent

  Pointer to the parent of this widget.

- struct ei_widget_t * children_head

  Pointer to the first child of this widget. Children are chained with the "next_sibling" field.

- struct ei_widget_t * children_tail

  Pointer to the last child of this widget.

- struct ei_widget_t * next_sibling

  Pointer to the next child of this widget's parent widget.

- struct ei_geometry_param_t * geom_params

  Pointer to the geometry management parameters for this widget. If NULL, the widget is not currently managed and thus, is not mapped on the screen.

- ei_size_t requested_size

  Size requested by the widget (big enough for its label, for example), or by the programmer. This can be different than its screen size defined by the placer.

- ei_rect_t screen_location

  Position and size of the widget expressed in the root window reference.

- ei_rect_t * content_rect

  Where to place children, when this widget is used as a container. By defaults, points to the screen_location.

6.7.3 Enumeration Type Documentation

6.7.3.1 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.

Definition at line 95 of file ei_types.h.

6.7.3.2 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.

Definition at line 120 of file ei_types.h.

6.7.3.3 enum ei_bool_t

The boolean type used in the library.

Enumerator:

EI_FALSE

EI_TRUE

Definition at line 24 of file ei_types.h.

- 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.

- enum ei_fontstyle_t {

  ei_style_normal = 0, ei_style_bold = 1, ei_style_italic = 2, ei_style_underline = 4,

  ei_style_strikethrough = 8 }

  Font style.

Variables

- static const ei_color_t ei_default_background_color = { 0xA0, 0xA0, 0xA0, 0xff }

  The default background color of widgets.

- ei_font_t ei_default_font

  The default font used in widgets.

- static const int ei_font_default_size = 22

  Default font color.

- static const ei_color_t ei_font_default_color = { 0x00, 0x00, 0x00, 0xff }
- static const char ei_default_font_filename [ ] = "misc/font.ttf"

6.7.1 Detailed Description

Type, constant, and global definitions for the ei library. Created by François Bérard on 18.12.11. Copyright 2011 Ensimag. All rights reserved.

Definition in file ei_types.h.

6.7.2 Typedef Documentation

6.7.2.1 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.

Definition at line 154 of file ei_types.h.

5.14.1 Detailed Description

Fields common to all types of widget. Every widget classes specializes this base class by adding its own fields.

Definition at line 24 of file ei_widget.h.

5.14.2 Field Documentation

5.14.2.1 struct ei_widget_t* ei_widget_t::children_head [read]

Pointer to the first child of this widget. Children are chained with the "next_sibling" field.

Definition at line 31 of file ei_widget.h.

5.14.2.2 struct ei_widget_t* ei_widget_t::children_tail [read]

Pointer to the last child of this widget.

Definition at line 32 of file ei_widget.h.

5.14.2.3 ei_rect_t* ei_widget_t::content_rect

Where to place children, when this widget is used as a container. By defaults, points to the screen_location.

Definition at line 40 of file ei_widget.h.

5.14.2.4 struct ei_geometry_param_t* ei_widget_t::geom_params [read]

Pointer to the geometry management parameters for this widget. If NULL, the widget is not currently managed and thus, is not mapped on the screen.

Definition at line 36 of file ei_widget.h.

5.14.2.5 struct ei_widget_t* ei_widget_t::next_sibling [read]

Pointer to the next child of this widget's parent widget.

Definition at line 33 of file ei_widget.h.

5.14.2.6 struct ei_widget_t* ei_widget_t::parent [read]

Pointer to the parent of this widget.

Definition at line 30 of file ei_widget.h.

5.14.2.7 ei_color_t* ei_widget_t::pick_color

pick_id encoded as a color.

Definition at line 27 of file ei_widget.h.

5.14.2.8 uint32_t ei_widget_t::pick_id

Id of this widget in the picking offscreen.

Definition at line 26 of file ei_widget.h.

5.14.2.9 ei_size_t ei_widget_t::requested_size

Size requested by the widget (big enough for its label, for example), or by the programmer. This can be different than its screen size defined by the placer.

Definition at line 38 of file ei_widget.h.

5.14.2.10 ei_rect_t ei_widget_t::screen_location

Position and size of the widget expressed in the root window reference.

Definition at line 39 of file ei_widget.h.

5.14.2.11 ei_widgetclass_t* ei_widget_t::wclass

The class of widget of this widget. Avoid the field name "class" which is a keyword in C++.

Definition at line 25 of file ei_widget.h.

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

- ei_widget.h

6.7 ei_types.h File Reference

Type, constant, and global definitions for the ei library. `#include "SDL_keysym.h"`

`#include <stdint.h>`

Data Structures

- 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 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.

6.6 ei_parser.h File Reference

```
#include "ei_widget.h"
```

Functions

- int ei_parse_file (char *file_path)
- ei_widget_t * ei_parse_widget_from_name (char *name)
- void free_name_to_widget_list ()

6.6.1 Function Documentation

6.6.1.1 int ei_parse_file (char * *file_path*)

6.6.1.2 ei_widget_t* ei_parse_widget_from_name (char * *name*)

6.6.1.3 void free_name_to_widget_list ()

5.15 ei_widgetclass_t Struct Reference

A structure that stores information about a class of widgets.

```
#include <ei_widgetclass.h>
```

Data Fields

- ei_widgetclass_name_t name

  The string name of this class of widget.

- ei_widgetclass_allocfunc_t allocfunc

  The function that allocated instances of this class of widget.

- ei_widgetclass_releasefunc_t releasefunc

  The function that releases all the resources used by an instance of this class of widget.

- ei_widgetclass_drawfunc_t drawfunc

  The function that draws on screen an instance of this class of widget.

- ei_widgetclass_setdefaultsfunc_t setdefaultsfunc

  The function that sets the default values to all the parameters of an instance of this class of widget.

- ei_widgetclass_geomnotifyfunc_t geomnotifyfunc

  The function that is called to notify an instance of widget of this class that its geometry has changed.

- struct ei_widgetclass_t * next

  A pointer to the next instance of ei_widget_class_t, allows widget class descriptions to be chained.

5.15.1 Detailed Description

A structure that stores information about a class of widgets.

Definition at line 81 of file ei_widgetclass.h.

5.15.2 Field Documentation

5.15.2.1 ei_widgetclass_allocfunc_t ei_widgetclass_t::allocfunc

The function that allocated instances of this class of widget.

Definition at line 83 of file ei_widgetclass.h.

5.15.2.2 ei_widgetclass_drawfunc_t ei_widgetclass_t::drawfunc

The function that draws on screen an instance of this class of widget.

Definition at line 85 of file ei_widgetclass.h.

5.15.2.3 ei_widgetclass_geomnotifyfunc_t ei_widgetclass_t::geomnotifyfunc

The function that is called to notify an instance of widget of this class that its geometry has changed.

Definition at line 87 of file ei_widgetclass.h.

5.15.2.4 ei_widgetclass_name_t ei_widgetclass_t::name

The string name of this class of widget.

Definition at line 82 of file ei_widgetclass.h.

5.15.2.5 struct ei_widgetclass_t* ei_widgetclass_t::next [read]

A pointer to the next instance of ei_widget_class_t, allows widget class descriptions to be chained.

Definition at line 88 of file ei_widgetclass.h.

5.15.2.6 ei_widgetclass_releasefunc_t ei_widgetclass_t::releasefunc

The function that releases all the resources used by an instance of this class of widget.

Definition at line 84 of file ei_widgetclass.h.

5.15.2.7 ei_widgetclass_setdefaultsfunc_t ei_widgetclass_t::setdefaultsfunc

The function that sets the default values to all the parameters of an instance of this class of widget.

Definition at line 86 of file ei_widgetclass.h.

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

- ei_widgetclass.h

6.5 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.

6.5.1 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.

Definition in file ei_main.h.

6.5.2 Function Documentation

6.5.2.1 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.

6.4.3.5 void ei_register_placer_manager ()

Registers the "placer" geometry manager in the program. This must be called only once before the ei_place function can be called.

Chapter 6

File Documentation

6.1 ei_application.h File Reference

Manages the main steps of a graphical application: initialization, main window, main loop, quitting, resource freeing. `#include "ei_types.h"`

`#include "ei_widget.h"`

Functions

- void ei_app_create (ei_size_t *main_window_size, ei_bool_t fullscreen)

  Creates an application.

- void ei_app_free ()

  Releases all the resources of the application, and releases the hardware (ie. calls hw_quit).

- void ei_app_run ()

  Runs the application: enters the main event loop. Exits when ei_app_quit_request is called.

- void ei_app_invalidate_rect (ei_rect_t *rect)

  Adds a rectangle to the list of rectangles that must be updated on screen. The real update on the screen will be done at the right moment in the main loop.

- void ei_app_quit_request ()

  Tells the application to quite. Is usually called by an event handler (for example when pressing the "Escape" key).

- ei_widget_t * ei_app_root_widget ()

  Returns the "root widget" of the application: a "frame" widget that encapsulate the root window.

- ei_surface_t ei_app_root_surface ()

  Returns the surface of the root window. Used to create surfaces with similar r, g, b channels.

6.1.1 Detailed Description

Manages the main steps of a graphical application: initialization, main window, main loop, quitting, resource freeing.

Author:

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

Definition in file ei_application.h.

6.1.2 Function Documentation

6.1.2.1 void ei_app_create (ei_size_t * *main_window_size*, ei_bool_t *fullscreen*)

Creates an application.

- initializes the hardware (calls hw_init),
- registers all classes of widget and all geometry managers,
- creates the root window (either in a system window, or the entire screen),
- creates the root widget to accress the root window.

Parameters:

main_window_size If fullscreen is false, the size of the root window of the application. If "fullscreen" is true, the current monitor resolution is used as the size of the root window, this size is returned in this parameter.

fullScreen If true, the root window is the entire screen. Otherwise, it is a system window.

6.1.2.2 void ei_app_free ()

Releases all the resources of the application, and releases the hardware (ie. calls hw_quit).

- the ei_geometrymanager_releasefunc_t of the geometry manager in charge of this widget is called,
- the geom_param field of the widget is freed,
- the current screen_location of the widget is invalided (which will trigger a redraw),
- the screen_location of the widget is reset to 0.

Parameters:

widget The widget to unmap from the screen.

6.4.3.4 void ei_place (ei_widget_t * *widget*, ei_anchor_t * *anchor*, int * *x*, int * *y*, int * *width*, int * *height*, float * *rel_x*, float * *rel_y*, float * *rel_width*, float * *rel_height*)

Configures the geometry of a widget using the "placer" geometry manager. If the widget was already managed by another geometry manager, then it is first removed from the previous geometry manager. If the widget was already managed by the "placer", then this calls simply updates the placer parameters: arguments that are not NULL replace previous values. When the arguments are passed as NULL, the placer uses default values (detailed in the argument descriptions below). If no size is provided (either absolute or relative), then the requested size of the widget is used, i.e. the minimal size required to display its content.

Parameters:

widget The widget to place.

anchor How to anchor the widget to the position defined by the placer (defaults to ei_anc_northwest).

x The absolute x position of the widget (defaults to 0).

y The absolute y position of the widget (defaults to 0).

width The absolute width for the widget (defaults to the requested width of the widget).

height The absolute height for the widget (defaults to the requested height of the widget).

rel_x The relative x position of the widget: 0.0 corresponds to the left side of the master, 1.0 to the right side (defaults to 0.0).

rel_y The relative y position of the widget: 0.0 corresponds to the top side of the master, 1.0 to the bottom side (defaults to 0.0).

rel_width The relative width of the widget: 0.0 corresponds to a width of 0, 1.0 to the width of the master (defaults to 0.0).

rel_height The relative height of the widget: 0.0 corresponds to a height of 0, 1.0 to the height of the master (defaults to 0.0).

Parameters:

widget The widget instance that must be forgotten by the geometry manager.

Definition at line 39 of file ei_geometrymanager.h.

6.4.2.3 typedef void(∗ ei_geometrymanager_runfunc_t)(struct ei_widget_t ∗widget)

A function that runs the geometry computation for this widget. This may trigger geometry computation for this widget's master and the other slaves of the master.

Parameters:

widget The widget instance for which to compute geometry.

Definition at line 29 of file ei_geometrymanager.h.

6.4.3 Function Documentation

6.4.3.1 ei_geometrymanager_t∗ ei_geometrymanager_from_name (ei_geometrymanager_name_t *name*)

Returns a geometry manager structure from its name.

Parameters:

name The name of the geometry manager.

Returns:

The structure describing the geometry manager.

6.4.3.2 void ei_geometrymanager_register (ei_geometrymanager_t ∗ *geometrymanager*)

Registers a geometry manager to the program so that it can be called to manager widgets. This must be done only once in the application.

Parameters:

geometrymanager The structure describing the geometry manager.

6.4.3.3 void ei_geometrymanager_unmap (ei_widget_t ∗ *widget*)

Tell the geometry manager in charge of a widget to forget it. This removes the widget from the screen. If the widget is not currently managed, this function returns silently. Side effects:.

6.1.2.3 void ei_app_invalidate_rect (ei_rect_t ∗ *rect*)

Adds a rectangle to the list of rectangles that must be updated on screen. The real update on the screen will be done at the right moment in the main loop.

Parameters:

rect The rectangle to add, expressed in the root window coordinates. A copy is made, so it is safe to release the rectangle on return.

6.1.2.4 void ei_app_quit_request ()

Tells the application to quite. Is usually called by an event handler (for example when pressing the "Escape" key).

6.1.2.5 ei_surface_t ei_app_root_surface ()

Returns the surface of the root window. Used to create surfaces with similar r, g, b channels.

Returns:

The surface of the root window.

6.1.2.6 ei_widget_t∗ ei_app_root_widget ()

Returns the "root widget" of the application: a "frame" widget that encapsulate the root window.

Returns:

The root widget.

6.1.2.7 void ei_app_run ()

Runs the application: enters the main event loop. Exits when ei_app_quit_request is called.

6.2 ei_draw.h File Reference

Graphical primitives to draw lines, polygons, text, and operation of drawing surfaces.

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

Functions

- uint32_t ei_map_rgba (ei_surface_t surface, const ei_color_t *color)

  Converts the three red, green and blue component of a color in a 32 bits integer using the order of the channels of the surface. This integer can be stored directly in the pixels memory of the surface (ie. hw_surface_get_buffer).

- void ei_draw_polyline (ei_surface_t surface, const ei_linked_point_t *first_point, const ei_color_t color, const ei_rect_t *clipper)

  Draws a line made of many line segments.

- void ei_draw_polygon (ei_surface_t surface, const ei_linked_point_t *first_point, const ei_color_t color, const ei_rect_t *clipper)

  Draws a filled polygon.

- void ei_draw_text (ei_surface_t surface, const ei_point_t *where, const char *text, const ei_font_t font, const ei_color_t *color, const ei_rect_t *clipper)

  Draws text by calling hw_text_create_surface.

- void ei_fill (ei_surface_t surface, const ei_color_t *color, const ei_rect_t *clipper)

  Fills the surface with the specified color.

- int ei_copy_surface (ei_surface_t destination, const ei_rect_t *dst_rect, const ei_surface_t source, const ei_rect_t *src_rect, const ei_bool_t alpha)

  *Copies a surface, or a subpart, to another one. The source and destination area of the copy (either the entire surfaces, or subparts) must have the same size (before clipping). Both the source and destination surfaces must be *locked* by hw_surface_lock.*

6.2.1 Detailed Description

Graphical primitives to draw lines, polygons, text, and operation of drawing surfaces.

Author:

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

Definition in file ei_draw.h.

Tell the geometry manager in charge of a widget to forget it. This removes the widget from the screen. If the widget is not currently managed, this function returns silently. Side effects:.

- void ei_register_placer_manager ()

  Registers the "placer" geometry manager in the program. This must be called only once before the ei_place function can be called.

- void ei_place (ei_widget_t *widget, ei_anchor_t *anchor, int *x, int *y, int *width, int *height, float *rel_x, float *rel_y, float *rel_width, float *rel_height)

  Configures the geometry of a widget using the "placer" geometry manager. If the widget was already managed by another geometry manager, then it is first removed from the previous geometry manager. If the widget was already managed by the "placer", then this calls simply updates the placer parameters: arguments that are not NULL replace previous values. When the arguments are passed as NULL, the placer uses default values (detailed in the argument descriptions below). If no size is provided (either absolute or relative), then the requested size of the widget is used, i.e. the minimal size required to display its content.

6.4.1 Detailed Description

Manages the positionning and sizing of widgets on the screen.

Author:

Created by François Bérard on 18.12.11. Copyright 2011 Ensimag. All rights reserved.

Definition in file ei_geometrymanager.h.

6.4.2 Typedef Documentation

6.4.2.1 typedef char ei_geometrymanager_name_t[20]

A name of a geometry manager.

Definition at line 21 of file ei_geometrymanager.h.

6.4.2.2 typedef void(* ei_geometrymanager_releasefunc_t)(struct ei_widget_t *widget)

A function called when a widget cease to be managed by its geometry manager. Most of the work is done in ei_geometrymanager_unmap. This function hook is only provided to trigger recomputation when the disappearance of a widget has an effect on the geometry of other widgets.

6.4 ei_geometrymanager.h File Reference

Manages the positionning and sizing of widgets on the screen. `#include "ei_types.h"`

`#include "ei_widget.h"`

Data Structures

- struct ei_geometrymanager_t

  The structure that stores information about a geometry manager.

- struct ei_geometry_param_t

  A structure that stores information about the geometry manager managing a widget, and the widget's geometry management parameters. This a the generic type. Each geometry manager adds field after "manager".

Typedefs

- typedef char ei_geometrymanager_name_t [20]

  A name of a geometry manager.

- typedef void(∗ ei_geometrymanager_runfunc_t)(struct ei_widget_t ∗widget)

  A function that runs the geometry computation for this widget. This may trigger geometry computation for this widget's master and the other slaves of the master.

- typedef void(∗ ei_geometrymanager_releasefunc_t)(struct ei_widget_t ∗widget)

  A function called when a widget cease to be managed by its geometry manager. Most of the work is done in ei_geometrymanager_unmap. This function hook is only provided to trigger recomputation when the disappearance of a widget has an effect on the geometry of other widgets.

Functions

- void ei_geometrymanager_register (ei_geometrymanager_t ∗geometrymanager)

  Registers a geometry manager to the program so that it can be called to manager widgets. This must be done only once in the application.

- ei_geometrymanager_t ∗ ei_geometrymanager_from_name (ei_geometrymanager_name_t name)

  Returns a geometry manager structure from its name.

- void ei_geometrymanager_unmap (ei_widget_t ∗widget)

6.2.2 Function Documentation

6.2.2.1 int ei_copy_surface (ei_surface_t *destination*, const ei_rect_t ∗ *dst_rect*, const ei_surface_t *source*, const ei_rect_t ∗ *src_rect*, const ei_bool_t *alpha*)

Copies a surface, or a subpart, to another one. The source and destination area of the copy (either the entire surfaces, or subparts) must have the same size (before clipping). Both the source and destination surfaces must be ∗locked∗ by hw_surface_lock.

Parameters:

destination The surface on which to copy pixels from the source surface.

dst_rect If NULL, the entire destination surface is used. If not NULL, defines the rectangle on the destination surface where to copy the pixels.

source The surface from which to copy pixels.

src_rect If NULL, the entire source surface is used. If not NULL, defines the rectangle on the source surface from which to copy the pixels.

alpha If true, the final pixels are a combination of source and destination pixels weighted by the source alpha channel. The transparency of the final pixels is set to opaque. If false, the final pixels are an exact copy of the source pixels, including the alpha channel.

Returns:

Returns 0 on success, 1 on failure (different ROI size).

6.2.2.2 void ei_draw_polygon (ei_surface_t *surface*, const ei_linked_point_t ∗ *first_point*, const ei_color_t *color*, const ei_rect_t ∗ *clipper*)

Draws a filled polygon.

Parameters:

surface Where to draw the polygon. The surface must be ∗locked∗ by hw_surface_lock.

first_point The head of a linked list of the points of the line. It is either NULL (i.e. draws nothing), or has more than 2 points.

color The color used to draw the polygon, alpha channel is managed.

clipper If not NULL, the drawing is restricted within this rectangle.

6.2.2.3 void ei_draw_polyline (ei_surface_t *surface*, const ei_linked_point_t ∗ *first_point*, const ei_color_t *color*, const ei_rect_t ∗ *clipper*)

Draws a line made of many line segments.

Parameters:

surface Where to draw the line. The surface must be ∗locked∗ by hw_surface_lock.

first_point The head of a linked list of the points of the line. It can be NULL (i.e. draws nothing), can have a single point, or more. If the last point is the same as the first point, then this pixel is drawn only once.

color The color used to draw the line, alpha channel is managed.

clipper If not NULL, the drawing is restricted within this rectangle.

6.2.2.4 void ei_draw_text (ei_surface_t *surface*, const ei_point_t ∗ *where*, const char ∗ *text*, const ei_font_t *font*, const ei_color_t ∗ *color*, const ei_rect_t ∗ *clipper*)

Draws text by calling hw_text_create_surface.

Parameters:

surface Where to draw the text. The surface must be ∗locked∗ by hw_surface_lock.

where Coordinates, in the surface, where to anchor the top-left corner of the rendered text.

text The string of the text. Can't be NULL.

font The font used to render the text. If NULL, the ei_default_font is used.

color The text color. Can't be NULL. The alpha parameter is not used.

clipper If not NULL, the drawing is restricted within this rectangle.

6.2.2.5 void ei_fill (ei_surface_t *surface*, const ei_color_t ∗ *color*, const ei_rect_t ∗ *clipper*)

Fills the surface with the specified color.

Parameters:

surface The surface to be filled. The surface must be ∗locked∗ by hw_surface_lock.

color The color used to fill the surface. If NULL, it means that the caller want it painted black (opaque).

clipper If not NULL, the drawing is restricted within this rectangle.

6.2.2.6 uint32_t ei_map_rgba (ei_surface_t *surface*, const ei_color_t ∗ *color*)

Converts the three red, green and blue component of a color in a 32 bits integer using the order of the channels of the surface. This integer can be stored directly in the pixels memory of the surface (ie. hw_surface_get_buffer).

widget The callback is only called if the event is related to this widget. This parameter must be NULL if the "tag" parameter is not NULL.

tag The callback is only called if the event is related to a widget that has this tag. A tag can be a widget class name, or the tag "all". This parameter must be NULL is the "widget" parameter is not NULL.

callback The callback (i.e. the function to call).

user_param A user parameter that will be passed to the callback when it is called.

6.3.4.2 static ei_bool_t ei_has_modifier (ei_modifier_mask_t *mask*, ei_modifier_key_t *modifier*) `[inline, static]`

Tests is a modifier key is currently pressed, according to a bitfield.

Parameters:

mask The bitfield.

modifier The modifier key that is tested.

Returns:

EI_TRUE is this modifier key is currently pressed, EI_FALSE otherwise.

Definition at line 78 of file ei_event.h.

6.3.4.3 void ei_unbind (ei_eventtype_t *eventtype*, ei_widget_t ∗ *widget*, ei_tag_t *tag*, ei_callback_t *callback*, void ∗ *user_param*)

Unbinds a callback from an event type and widget or tag.

Parameters:

eventtype,widget,tag,callback,user_param All parameters must have the same value as when ei_bind was called to create the binding.

6.3.3 Enumeration Type Documentation

6.3.3.1 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_keydown A keyboard key has been pressed.

ei_ev_keyup A keyboard key has been released.

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_last Last event type, its value is the number of event types.

Definition at line 34 of file ei_event.h.

6.3.3.2 enum ei_modifier_key_t

The modifier keys (shift, alt, etc.).

Enumerator:

ei_mod_shift_right The "shift" key at the right of the space bar.

ei_mod_shift_left The "shift" key at the left of the space bar.

ei_mod_ctrl_right The "control" key at the right of the space bar.

ei_mod_ctrl_left The "control" key at the left of the space bar.

ei_mod_alt_right The "alternate" key at the right of the space bar.

ei_mod_alt_left The "alternate" 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_meta_left The "meta" (command) key at the left of the space bar.

Definition at line 51 of file ei_event.h.

6.3.4 Function Documentation

6.3.4.1 void ei_bind (ei_eventtype_t *eventtype*, ei_widget_t * *widget*, ei_tag_t *tag*, ei_callback_t *callback*, void * *user_param*)

Binds a callback to an event type and a widget or a tag.

Parameters:

eventtype The type of the event.

Parameters:

surface The surface where to store this pixel.

color The color to convert, can't be NULL.

Returns:

The 32 bit integer corresponding to the color. The alpha component of the color is ignored in the case of surfaces the don't have an alpha channel.

6.3 ei_event.h File Reference

Allows the binding and unbinding of callbacks to events. `#include "ei_types.h"`

`#include "ei_widget.h"`

Data Structures

- struct ei_linked_tag_t
  A tag and a pointer to create a linked list.
- 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_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 uint32_t ei_modifier_mask_t
  A bitfield indicating which of the modifier keys are currently pressed.

Enumerations

- enum ei_eventtype_t {
  ei_ev_none = 0, ei_ev_app, ei_ev_keydown, ei_ev_keyup,
  ei_ev_mouse_buttondown, ei_ev_mouse_buttonup, ei_ev_mouse_move, ei_ev_last }
  The types of events.
- enum ei_modifier_key_t {
  ei_mod_shift_right = 0, ei_mod_shift_left, ei_mod_ctrl_right, ei_mod_ctrl_left,
  ei_mod_alt_right, ei_mod_alt_left, ei_mod_meta_right, ei_mod_meta_left }

The modifier keys (shift, alt, etc.).

Functions

- static ei_bool_t ei_has_modifier (ei_modifier_mask_t mask, ei_modifier_key_t modifier)
  Tests is a modifier key is currently pressed, according to a bitfield.
- void ei_bind (ei_eventtype_t eventtype, ei_widget_t *widget, ei_tag_t tag, ei_callback_t callback, void *user_param)
  Binds a callback to an event type and a widget or a tag.
- void ei_unbind (ei_eventtype_t eventtype, ei_widget_t *widget, ei_tag_t tag, ei_callback_t callback, void *user_param)
  Unbinds a callback from an event type and widget or tag.

6.3.1 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.

Definition in file ei_event.h.

6.3.2 Typedef Documentation

6.3.2.1 typedef uint32_t ei_modifier_mask_t

A bitfield indicating which of the modifier keys are currently pressed.

Definition at line 66 of file ei_event.h.

6.3.2.2 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".

Definition at line 21 of file ei_event.h.