

Projet C - Interaction Graphique

Generated by Doxygen 1.6.1

Thu May 26 16:34:57 2016



# Contents

<b>1</b>	<b>Directory Hierarchy</b>	<b>1</b>
	1.1 Directories	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
	2.1 Data Structures	3
<b>3</b>	<b>File Index</b>	<b>5</b>
	3.1 File List	5
<b>4</b>	<b>Directory Documentation</b>	<b>7</b>
	4.1 include/Directory Reference	7
<b>5</b>	<b>Data Structure Documentation</b>	<b>9</b>
	5.1 et_app_event_t Struct Reference	9
	5.1.1 Detailed Description	9
	5.1.2 Field Documentation	9
	5.1.2.1 user_param	9
	5.2 et_color_t Struct Reference	10
	5.2.1 Detailed Description	10
	5.2.2 Field Documentation	10
	5.2.2.1 alpha	10
	5.2.2.2 blue	10
	5.2.2.3 green	10
	5.2.2.4 red	11
	5.3 et_event_t Struct Reference	12
	5.3.1 Detailed Description	12
	5.3.2 Field Documentation	12

5.3.2.1	application	12
5.3.2.2	key	12
5.3.2.3	mouse	12
5.3.2.4	param	13
5.3.2.5	type	13
5.4	ei_geometry_param_t Struct Reference	14
5.4.1	Detailed Description	14
5.4.2	Field Documentation	14
5.4.2.1	manager	14
5.5	ei_geometrymanager_t Struct Reference	15
5.5.1	Detailed Description	15
5.5.2	Field Documentation	15
5.5.2.1	name	15
5.5.2.2	next	15
5.5.2.3	releasefunc	15
5.5.2.4	runfunc	15
5.6	ei_key_event_t Struct Reference	16
5.6.1	Detailed Description	16
5.6.2	Field Documentation	16
5.6.2.1	key_sym	16
5.6.2.2	modifier_mask	16
5.7	ei_linked_point_t Struct Reference	17
5.7.1	Detailed Description	17
5.7.2	Field Documentation	17
5.7.2.1	next	17
5.7.2.2	point	17
5.8	ei_linked_rect_t Struct Reference	18
5.8.1	Detailed Description	18
5.8.2	Field Documentation	18
5.8.2.1	next	18
5.8.2.2	rect	18
5.9	ei_linked_tag_t Struct Reference	19
5.9.1	Detailed Description	19
5.9.2	Field Documentation	19

ei_widgetclass_t	28
requested_size	
ei_widget_t	26
runfunc	
ei_geometrymanager_t	15
screen_location	
ei_widget_t	26
setdefaultsfunc	
ei_widgetclass_t	28
size	
ei_rect_t	22
tag	
ei_linked_tag_t	19
top_left	
ei_rect_t	22
type	
ei_event_t	13
user_param	
ei_app_event_t	9
wclass	
ei_widget_t	26
where	
ei_mouse_event_t	20
width	
ei_size_t	23
x	
ei_point_t	21
y	
ei_point_t	21



5.14.2.10 screen_location	26
5.14.2.11 wclass	26
5.15 ei_widgetclass_t Struct Reference	27
5.15.1 Detailed Description	27
5.15.2 Field Documentation	27
5.15.2.1 allocfunc	27
5.15.2.2 drawfunc	28
5.15.2.3 geomnotifyfunc	28
5.15.2.4 name	28
5.15.2.5 next	28
5.15.2.6 releasefunc	28
5.15.2.7 setdefaultsfunc	28
<b>6 File Documentation</b>	<b>29</b>
6.1 ei_application.h File Reference	29
6.1.1 Detailed Description	30
6.1.2 Function Documentation	30
6.1.2.1 ei_app_create	30
6.1.2.2 ei_app_free	30
6.1.2.3 ei_app_invalidate_rect	31
6.1.2.4 ei_app_quit_request	31
6.1.2.5 ei_app_root_surface	31
6.1.2.6 ei_app_root_widget	31
6.1.2.7 ei_app_run	31
6.2 ei_draw.h File Reference	32
6.2.1 Detailed Description	32
6.2.2 Function Documentation	33
6.2.2.1 ei_copy_surface	33
6.2.2.2 ei_draw_polygon	33
6.2.2.3 ei_draw_polyline	33
6.2.2.4 ei_draw_text	34
6.2.2.5 ei_fill	34
6.2.2.6 ei_map_rgba	34
6.3 ei_event.h File Reference	36

ei_frame_configure, 57	ei_widgetclass.h, 63
ei_toplevel_configure, 58	ei_widgetclass_releasefunc_t
ei_widget_create, 58	ei_widgetclass.h, 62
ei_widget_destroy, 59	ei_widgetclass_setdefaultsfunc_t
ei_widget_pick, 59	ei_widgetclass.h, 62
k_default_button_border_width, 59	ei_widgetclass_stringname
k_default_button_corner_radius, 59	ei_widgetclass.h, 63
ei_widget_create	ei_widgetclass_t, 27
ei_widget.h, 58	allocfunc, 27
ei_widget_destroy	drawfunc, 27
ei_widget.h, 59	geomnotifyfunc, 28
ei_widget_pick	name, 28
ei_widget.h, 59	next, 28
ei_widget_t, 24	releasefunc, 28
children_head, 25	setdefaultsfunc, 28
children_tail, 25	
content_rect, 25	free_name_to_widget_list
geom_params, 25	ei_parser.h, 46
next_sibling, 25	
parent, 25	geom_params
pick_color, 25	ei_widget_t, 25
pick_id, 26	geomnotifyfunc
requested_size, 26	ei_widgetclass_t, 28
screen_location, 26	green
wclass, 26	ei_color_t, 10
ei_widgetclass.h, 60	
ei_button_register_class, 62	height
ei_frame_register_class, 63	ei_size_t, 23
ei_toplevel_register_class, 63	hw_create_window
ei_widgetclass_allocfunc_t, 61	hw_interface.h, 67
ei_widgetclass_drawfunc_t, 61	hw_event_post_app
ei_widgetclass_from_name, 63	hw_interface.h, 67
ei_widgetclass_geomnotifyfunc_t, 61	hw_event_schedule_app
ei_widgetclass_name_t, 62	hw_interface.h, 67
ei_widgetclass_register, 63	hw_event_wait_next
ei_widgetclass_releasefunc_t, 62	hw_interface.h, 67
ei_widgetclass_setdefaultsfunc_t, 62	hw_image_load
ei_widgetclass_stringname, 63	hw_interface.h, 67
ei_widgetclass_allocfunc_t	hw_init
ei_widgetclass.h, 61	hw_interface.h, 68
ei_widgetclass_drawfunc_t	hw_interface.h, 64
ei_widgetclass.h, 61	EI_MOUSEBUTTON_LEFT, 72
ei_widgetclass_from_name	EI_MOUSEBUTTON_MIDDLE,
ei_widgetclass.h, 63	72
ei_widgetclass_geomnotifyfunc_t	EI_MOUSEBUTTON_RIGHT, 72
ei_widgetclass.h, 61	ei_surface_t, 66
ei_widgetclass_name_t	hw_create_window, 67
ei_widgetclass.h, 62	hw_event_post_app, 67
ei_widgetclass_register	hw_event_schedule_app, 67
	hw_event_wait_next, 67

6.3.1	Detailed Description	37	ei_main.h, 45	ei_main.h, 45
6.3.2	TypeDef Documentation	37	ei_map_rgb	ei_main, 45
6.3.3	Enumeration Type Documentation	38	ei_modifier_key_t	ei_modifier_mask_t
6.3.4	Function Documentation	38	ei_event.h, 38	ei_event.h, 38
6.3.1	ei_event_t	38	ei_modifier_mask_t	ei_modifier_mask_t
6.3.2	ei_modifier_key_t	38	ei_size_t, 23	ei_size_t, 23
6.3.4.1	ei_bind	38	ei_size_zero	ei_size_zero
6.3.4.2	ei_has_modifier	39	ei_surface_t	ei_surface_t
6.3.4.3	ei_unbind	39	ei_tag_t	ei_tag_t
6.4	ei_geometrymanager.h File Reference	40	ei_event.h, 37	ei_event.h, 37
6.4.1	Detailed Description	41	ei_toplevel_configure	ei_toplevel_configure
6.4.2	TypeDef Documentation	41	ei_widget.h, 58	ei_widget.h, 58
6.4.2.1	ei_geometrymanager_name_t	41	ei_widgetclass.h, 63	ei_widgetclass.h, 63
6.4.2.2	ei_geometrymanager_relcasfunc_t	41	ei_types.h, 47	ei_types.h, 47
6.4.2.3	ei_geometrymanager_runfunc_t	42	ei_anchor_t, 49	ei_anchor_t, 49
6.4.3	Function Documentation	42	ei_axis_set_t, 49	ei_axis_set_t, 49
6.4.3.1	ei_geometrymanager_from_name	42	ei_bool_t, 49	ei_bool_t, 49
6.4.3.2	ei_geometrymanager_register	42	ei_default_font_filename, 50	ei_default_font_filename, 50
6.4.3.3	ei_geometrymanager_unmap	42	ei_font_t, 48	ei_font_t, 48
6.4.3.4	ei_place	43	ei_fontstyle_t, 49	ei_fontstyle_t, 49
6.4.3.5	ei_register_placer_manager	44	ei_relief_t, 50	ei_relief_t, 50
6.5	ei_main.h File Reference	45	ei_unbind	ei_unbind
6.5.1	Detailed Description	45	ei_event.h, 39	ei_event.h, 39
6.5.2	Function Documentation	45	ei_utils.h, 52	ei_utils.h, 52
6.5.2.1	ei_main	45	ei_point_t, 21	ei_point_t, 21
6.6	ei_parser.h File Reference	46	ei_point_add, 53	ei_point_add, 53
6.6.1	Function Documentation	46	ei_point_neg, 53	ei_point_neg, 53
6.6.1.1	ei_parse_file	46	ei_point_sub, 53	ei_point_sub, 53
6.6.1.2	ei_parse_widget_from_name	46	ei_point_zero, 53	ei_point_zero, 53
6.6.1.3	free_name_to_widget_list	46	ei_rect, 53	ei_rect, 53
6.7	ei_types.h File Reference	47	ei_rect_zero, 53	ei_rect_zero, 53
6.7.1	Detailed Description	48	ei_size, 54	ei_size, 54
			ei_size_add, 54	ei_size_add, 54
			ei_size_sub, 54	ei_size_sub, 54
			ei_size_zero, 54	ei_size_zero, 54
			ei_widget.h, 55	ei_widget.h, 55
			ei_button_configure, 57	ei_button_configure, 57
			ei_callback_t, 56	ei_callback_t, 56
			ei_geometrymanager.h, 43	ei_geometrymanager.h, 43
			ei_register_placer_manager	ei_register_placer_manager
			ei_utils.h, 53	ei_utils.h, 53
			ei_rect, 22	ei_rect, 22
			top_left, 22	top_left, 22
			size, 22	size, 22
			ei_utils.h, 53	ei_utils.h, 53
			ei_rect, 22	ei_rect, 22
			ei_size, 54	ei_size, 54
			ei_rect_zero, 53	ei_rect_zero, 53
			ei_rect, 53	ei_rect, 53
			ei_point_zero, 53	ei_point_zero, 53
			ei_point_sub, 53	ei_point_sub, 53
			ei_point_neg, 53	ei_point_neg, 53
			ei_point_add, 53	ei_point_add, 53
			ei_point, 53	ei_point, 53
			ei_utils.h, 53	ei_utils.h, 53
			ei_event.h, 39	ei_event.h, 39
			ei_unbind	ei_unbind
			ei_point_sub	ei_point_sub
			ei_utils.h, 53	ei_utils.h, 53
			ei_point_neg	ei_point_neg
			ei_utils.h, 53	ei_utils.h, 53
			ei_point_add	ei_point_add
			ei_utils.h, 53	ei_utils.h, 53
			ei_point	ei_point
			ei_geometrymanager.h, 43	ei_geometrymanager.h, 43
			ei_place	ei_place
			free_name_to_widget_list, 46	free_name_to_widget_list, 46
			ei_parse_widget_from_name, 46	ei_parse_widget_from_name, 46
			ei_parse_file, 46	ei_parse_file, 46
			ei_parser.h, 46	ei_parser.h, 46
			ei_parser.h, 46	ei_parser.h, 46
			ei_parser_widget_from_name	ei_parser_widget_from_name
			ei_parser.h, 46	ei_parser.h, 46
			ei_parse_file	ei_parse_file
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_RIGHT	EI_MOUSEBUTTON_RIGHT
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_MIDDLE	EI_MOUSEBUTTON_MIDDLE
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_LEFT	EI_MOUSEBUTTON_LEFT
			hw_interface.h, 72	hw_interface.h, 72
			where, 20	where, 20
			button_number, 20	button_number, 20
			ei_mouse_event_t, 20	ei_mouse_event_t, 20
			ei_event.h, 37	ei_event.h, 37
			ei_modifier_mask_t	ei_modifier_mask_t
			ei_event.h, 38	ei_event.h, 38
			ei_modifier_key_t	ei_modifier_key_t
			ei_draw.h, 34	ei_draw.h, 34
			ei_map_rgb	ei_map_rgb
			ei_main, 45	ei_main, 45
			ei_types.h, 50	ei_types.h, 50
			ei_size	ei_size
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_add	ei_size_add
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_sub	ei_size_sub
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_t, 23	ei_size_t, 23
			height, 23	height, 23
			width, 23	width, 23
			ei_size_zero	ei_size_zero
			ei_utils.h, 54	ei_utils.h, 54
			ei_surface_t	ei_surface_t
			hw_interface.h, 66	hw_interface.h, 66
			ei_tag_t	ei_tag_t
			ei_event.h, 37	ei_event.h, 37
			ei_toplevel_configure	ei_toplevel_configure
			ei_widget.h, 58	ei_widget.h, 58
			ei_widgetclass.h, 63	ei_widgetclass.h, 63
			ei_types.h, 47	ei_types.h, 47
			ei_anchor_t, 49	ei_anchor_t, 49
			ei_axis_set_t, 49	ei_axis_set_t, 49
			ei_bool_t, 49	ei_bool_t, 49
			ei_default_font_filename, 50	ei_default_font_filename, 50
			ei_font_t, 48	ei_font_t, 48
			ei_fontstyle_t, 49	ei_fontstyle_t, 49
			ei_relief_t, 50	ei_relief_t, 50
			ei_unbind	ei_unbind
			ei_event.h, 39	ei_event.h, 39
			ei_utils.h, 52	ei_utils.h, 52
			ei_point_t, 21	ei_point_t, 21
			ei_point_add, 53	ei_point_add, 53
			ei_point_neg, 53	ei_point_neg, 53
			ei_point_sub, 53	ei_point_sub, 53
			ei_point_zero, 53	ei_point_zero, 53
			ei_rect, 53	ei_rect, 53
			ei_rect_zero, 53	ei_rect_zero, 53
			ei_size, 54	ei_size, 54
			ei_size_add, 54	ei_size_add, 54
			ei_size_sub, 54	ei_size_sub, 54
			ei_size_zero, 54	ei_size_zero, 54
			ei_widget.h, 55	ei_widget.h, 55
			ei_button_configure, 57	ei_button_configure, 57
			ei_callback_t, 56	ei_callback_t, 56
			ei_geometrymanager.h, 43	ei_geometrymanager.h, 43
			ei_register_placer_manager	ei_register_placer_manager
			ei_utils.h, 53	ei_utils.h, 53
			ei_rect, 22	ei_rect, 22
			top_left, 22	top_left, 22
			size, 22	size, 22
			ei_utils.h, 53	ei_utils.h, 53
			ei_rect, 22	ei_rect, 22
			ei_size, 54	ei_size, 54
			ei_rect_zero, 53	ei_rect_zero, 53
			ei_rect, 53	ei_rect, 53
			ei_point_zero, 53	ei_point_zero, 53
			ei_point_sub, 53	ei_point_sub, 53
			ei_point_neg, 53	ei_point_neg, 53
			ei_point_add, 53	ei_point_add, 53
			ei_point, 53	ei_point, 53
			ei_utils.h, 53	ei_utils.h, 53
			ei_event.h, 39	ei_event.h, 39
			ei_unbind	ei_unbind
			ei_point_sub	ei_point_sub
			ei_utils.h, 53	ei_utils.h, 53
			ei_point_neg	ei_point_neg
			ei_utils.h, 53	ei_utils.h, 53
			ei_point_add	ei_point_add
			ei_utils.h, 53	ei_utils.h, 53
			ei_point	ei_point
			ei_geometrymanager.h, 43	ei_geometrymanager.h, 43
			ei_place	ei_place
			free_name_to_widget_list, 46	free_name_to_widget_list, 46
			ei_parse_widget_from_name, 46	ei_parse_widget_from_name, 46
			ei_parse_file, 46	ei_parse_file, 46
			ei_parser.h, 46	ei_parser.h, 46
			ei_parser.h, 46	ei_parser.h, 46
			ei_parser_widget_from_name	ei_parser_widget_from_name
			ei_parser.h, 46	ei_parser.h, 46
			ei_parse_file	ei_parse_file
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_RIGHT	EI_MOUSEBUTTON_RIGHT
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_MIDDLE	EI_MOUSEBUTTON_MIDDLE
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_LEFT	EI_MOUSEBUTTON_LEFT
			hw_interface.h, 72	hw_interface.h, 72
			where, 20	where, 20
			button_number, 20	button_number, 20
			ei_mouse_event_t, 20	ei_mouse_event_t, 20
			ei_event.h, 37	ei_event.h, 37
			ei_modifier_mask_t	ei_modifier_mask_t
			ei_event.h, 38	ei_event.h, 38
			ei_modifier_key_t	ei_modifier_key_t
			ei_draw.h, 34	ei_draw.h, 34
			ei_map_rgb	ei_map_rgb
			ei_main, 45	ei_main, 45
			ei_types.h, 50	ei_types.h, 50
			ei_size	ei_size
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_add	ei_size_add
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_sub	ei_size_sub
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_t, 23	ei_size_t, 23
			height, 23	height, 23
			width, 23	width, 23
			ei_size_zero	ei_size_zero
			ei_utils.h, 54	ei_utils.h, 54
			ei_surface_t	ei_surface_t
			hw_interface.h, 66	hw_interface.h, 66
			ei_tag_t	ei_tag_t
			ei_event.h, 37	ei_event.h, 37
			ei_toplevel_configure	ei_toplevel_configure
			ei_widget.h, 58	ei_widget.h, 58
			ei_widgetclass.h, 63	ei_widgetclass.h, 63
			ei_types.h, 47	ei_types.h, 47
			ei_anchor_t, 49	ei_anchor_t, 49
			ei_axis_set_t, 49	ei_axis_set_t, 49
			ei_bool_t, 49	ei_bool_t, 49
			ei_default_font_filename, 50	ei_default_font_filename, 50
			ei_font_t, 48	ei_font_t, 48
			ei_fontstyle_t, 49	ei_fontstyle_t, 49
			ei_relief_t, 50	ei_relief_t, 50
			ei_unbind	ei_unbind
			ei_event.h, 39	ei_event.h, 39
			ei_utils.h, 52	ei_utils.h, 52
			ei_point_t, 21	ei_point_t, 21
			ei_point_add, 53	ei_point_add, 53
			ei_point_neg, 53	ei_point_neg, 53
			ei_point_sub, 53	ei_point_sub, 53
			ei_point_zero, 53	ei_point_zero, 53
			ei_rect, 53	ei_rect, 53
			ei_rect_zero, 53	ei_rect_zero, 53
			ei_size, 54	ei_size, 54
			ei_size_add, 54	ei_size_add, 54
			ei_size_sub, 54	ei_size_sub, 54
			ei_size_zero, 54	ei_size_zero, 54
			ei_widget.h, 55	ei_widget.h, 55
			ei_button_configure, 57	ei_button_configure, 57
			ei_callback_t, 56	ei_callback_t, 56
			ei_geometrymanager.h, 43	ei_geometrymanager.h, 43
			ei_register_placer_manager	ei_register_placer_manager
			ei_utils.h, 53	ei_utils.h, 53
			ei_rect, 22	ei_rect, 22
			top_left, 22	top_left, 22
			size, 22	size, 22
			ei_utils.h, 53	ei_utils.h, 53
			ei_rect, 22	ei_rect, 22
			ei_size, 54	ei_size, 54
			ei_rect_zero, 53	ei_rect_zero, 53
			ei_rect, 53	ei_rect, 53
			ei_point_zero, 53	ei_point_zero, 53
			ei_point_sub, 53	ei_point_sub, 53
			ei_point_neg, 53	ei_point_neg, 53
			ei_point_add, 53	ei_point_add, 53
			ei_point, 53	ei_point, 53
			ei_utils.h, 53	ei_utils.h, 53
			ei_event.h, 39	ei_event.h, 39
			ei_unbind	ei_unbind
			ei_point_sub	ei_point_sub
			ei_utils.h, 53	ei_utils.h, 53
			ei_point_neg	ei_point_neg
			ei_utils.h, 53	ei_utils.h, 53
			ei_point_add	ei_point_add
			ei_utils.h, 53	ei_utils.h, 53
			ei_point	ei_point
			ei_geometrymanager.h, 43	ei_geometrymanager.h, 43
			ei_place	ei_place
			free_name_to_widget_list, 46	free_name_to_widget_list, 46
			ei_parse_widget_from_name, 46	ei_parse_widget_from_name, 46
			ei_parse_file, 46	ei_parse_file, 46
			ei_parser.h, 46	ei_parser.h, 46
			ei_parser.h, 46	ei_parser.h, 46
			ei_parser_widget_from_name	ei_parser_widget_from_name
			ei_parser.h, 46	ei_parser.h, 46
			ei_parse_file	ei_parse_file
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_RIGHT	EI_MOUSEBUTTON_RIGHT
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_MIDDLE	EI_MOUSEBUTTON_MIDDLE
			hw_interface.h, 72	hw_interface.h, 72
			EI_MOUSEBUTTON_LEFT	EI_MOUSEBUTTON_LEFT
			hw_interface.h, 72	hw_interface.h, 72
			where, 20	where, 20
			button_number, 20	button_number, 20
			ei_mouse_event_t, 20	ei_mouse_event_t, 20
			ei_event.h, 37	ei_event.h, 37
			ei_modifier_mask_t	ei_modifier_mask_t
			ei_event.h, 38	ei_event.h, 38
			ei_modifier_key_t	ei_modifier_key_t
			ei_draw.h, 34	ei_draw.h, 34
			ei_map_rgb	ei_map_rgb
			ei_main, 45	ei_main, 45
			ei_types.h, 50	ei_types.h, 50
			ei_size	ei_size
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_add	ei_size_add
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_sub	ei_size_sub
			ei_utils.h, 54	ei_utils.h, 54
			ei_size_t, 23	ei_size_t, 23
			height, 23	height, 23
			width, 23	width, 23
			ei_size_zero	ei_size_zero
			ei_utils.h, 54	ei_utils.h, 54
			ei_surface_t	ei_surface_t
			hw_interface.h, 66	hw_interface.h, 66
			ei_tag_t	ei_tag_t
			ei_event.h, 37	ei_event.h, 37
			ei_toplevel_configure	ei_toplevel_configure
			ei_widget.h, 58	ei_widget.h, 58
			ei_widgetclass.h, 63	ei_widgetclass.h, 63
			ei_types.h, 47	ei_types.h, 47
			ei_anchor_t, 49	ei_anchor_t, 49
			ei_axis_set_t, 49	ei_axis_set_t, 49
			ei_bool_t, 49	ei_bool_t, 49
			ei_default_font_filename, 5	

6.7.2	Typedef Documentation	48
6.7.2.1	ei_font_t	48
6.7.3	Enumeration Type Documentation	49
6.7.3.1	ei_anchor_t	49
6.7.3.2	ei_axis_set_t	49
6.7.3.3	ei_bool_t	49
6.7.3.4	ei_fontstyle_t	50
6.7.3.5	ei_relief_t	50
6.7.4	Variable Documentation	50
6.7.4.1	ei_default_background_color	50
6.7.4.2	ei_default_font	50
6.7.4.3	ei_default_font_filename	50
6.7.4.4	ei_font_default_color	50
6.7.4.5	ei_font_default_size	51
6.8	ei_utils.h File Reference	52
6.8.1	Detailed Description	53
6.8.2	Function Documentation	53
6.8.2.1	ei_point	53
6.8.2.2	ei_point_add	53
6.8.2.3	ei_point_neg	53
6.8.2.4	ei_point_sub	53
6.8.2.5	ei_point_zero	53
6.8.2.6	ei_rect	53
6.8.2.7	ei_rect_zero	54
6.8.2.8	ei_size	54
6.8.2.9	ei_size_add	54
6.8.2.10	ei_size_sub	54
6.8.2.11	ei_size_zero	54
6.9	ei_widget.h File Reference	55
6.9.1	Detailed Description	56
6.9.2	Typedef Documentation	56
6.9.2.1	ei_callback_t	56
6.9.3	Function Documentation	57
6.9.3.1	ei_button_configure	57

alpha,	10
blue,	10
green,	10
red,	10
ei_copy_surface	
ei_draw.h,	33
ei_default_background_color	
ei_types.h,	50
ei_default_font	
ei_types.h,	50
ei_default_font_filename	
ei_types.h,	50
ei_draw.h,	32
ei_copy_surface,	33
ei_draw_polygon,	33
ei_draw_polyline,	33
ei_draw_text,	34
ei_fill,	34
ei_map_rgba,	34
ei_draw_polygon	
ei_draw.h,	33
ei_draw_polyline	
ei_draw.h,	33
ei_draw_text	
ei_draw.h,	34
ei_event.h,	36
ei_bind,	38
ei_eventtype_t,	38
ei_has_modifier,	39
ei_modifier_key_t,	38
ei_modifier_mask_t,	37
ei_tag_t,	37
ei_unbind,	39
ei_event_t,	12
application,	12
key,	12
mouse,	12
param,	12
type,	13
ei_eventtype_t	
ei_event.h,	38
ei_fill	
ei_draw.h,	34
ei_font_default_color	
ei_types.h,	50
ei_font_default_size	
ei_types.h,	50
ei_font_t	
ei_types.h,	48
ei_fontstyle_t	
ei_types.h,	49
ei_frame_configure	
ei_widget.h,	57
ei_frame_register_class	
ei_widgetclass.h,	63
ei_geometry_param_t,	14
manager,	14
ei_geometrymanager.h,	40
ei_geometrymanager_from_name,	42
ei_geometrymanager_name_t,	41
ei_geometrymanager_register,	42
ei_geometrymanager_releasefunc_t,	41
ei_geometrymanager_runfunc_t,	42
ei_geometrymanager_unmap,	42
ei_place,	43
ei_register_placer_manager,	43
ei_geometrymanager_from_name	
ei_geometrymanager.h,	42
ei_geometrymanager_name_t	
ei_geometrymanager.h,	41
ei_geometrymanager_register	
ei_geometrymanager.h,	42
ei_geometrymanager_releasefunc_t	
ei_geometrymanager.h,	41
ei_geometrymanager_runfunc_t	
ei_geometrymanager.h,	42
ei_geometrymanager_t,	15
name,	15
next,	15
releasefunc,	15
runfunc,	15
ei_geometrymanager_unmap	
ei_geometrymanager.h,	42
ei_has_modifier	
ei_event.h,	39
ei_key_event_t,	16
key_sym,	16
modifier_mask,	16
ei_linked_point_t,	17
next,	17
point,	17
ei_linked_rect_t,	18
next,	18
rect,	18
ei_linked_tag_t,	19
next,	19
tag,	19
ei_main	





6.11.3.7	<a href="#">hw_now</a>	68
6.11.3.8	<a href="#">hw_quit</a>	68
6.11.3.9	<a href="#">hw_surface_create</a>	68
6.11.3.10	<a href="#">hw_surface_free</a>	69
6.11.3.11	<a href="#">hw_surface_get_buffer</a>	69
6.11.3.12	<a href="#">hw_surface_get_channel_indices</a>	69
6.11.3.13	<a href="#">hw_surface_get_rect</a>	69
6.11.3.14	<a href="#">hw_surface_get_size</a>	70
6.11.3.15	<a href="#">hw_surface_has_alpha</a>	70
6.11.3.16	<a href="#">hw_surface_lock</a>	70
6.11.3.17	<a href="#">hw_surface_set_origin</a>	70
6.11.3.18	<a href="#">hw_surface_unlock</a>	71
6.11.3.19	<a href="#">hw_surface_update_rects</a>	71
6.11.3.20	<a href="#">hw_text_compute_size</a>	71
6.11.3.21	<a href="#">hw_text_create_surface</a>	71
6.11.3.22	<a href="#">hw_text_font_create</a>	72
6.11.3.23	<a href="#">hw_text_font_free</a>	72
6.11.4	<a href="#">Variable Documentation</a>	72
6.11.4.1	<a href="#">EI_MOUSEBUTTON_LEFT</a>	72
6.11.4.2	<a href="#">EI_MOUSEBUTTON_MIDDLE</a>	72
6.11.4.3	<a href="#">EI_MOUSEBUTTON_RIGHT</a>	72

## Index

allocfunc	
ei_widgetclass_t, 27	
alpha	
ei_color_t, 10	
application	
ei_event_t, 12	
blue	
ei_color_t, 10	
button_number	
ei_mouse_event_t, 20	
children_head	
ei_widget_t, 25	
children_tail	
ei_widget_t, 25	
content_rect	
ei_widget_t, 25	
drawfunc	
ei_widgetclass_t, 27	
ei_anc_center	
ei_types.h, 49	
ei_anc_east	
ei_types.h, 49	
ei_anc_none	
ei_types.h, 49	
ei_anc_north	
ei_types.h, 49	
ei_anc_northeast	
ei_types.h, 49	
ei_anc_northwest	
ei_types.h, 49	
ei_anc_south	
ei_types.h, 49	
ei_anc_southeast	
ei_types.h, 49	
ei_anc_southwest	
ei_types.h, 49	
ei_anc_west	
ei_types.h, 49	
ei_axis_both	
ei_types.h, 49	
ei_axis_none	
ei_types.h, 49	
ei_axis_x	
ei_types.h, 49	
ei_axis_y	
ei_types.h, 49	
ei_ev_app	
ei_event.h, 38	
ei_ev_keydown	
ei_event.h, 38	
ei_ev_keyup	
ei_event.h, 38	
ei_ev_last	
ei_event.h, 38	
ei_ev_mouse_buttondown	
ei_event.h, 38	
ei_ev_mouse_buttonup	
ei_event.h, 38	
ei_ev_mouse_move	
ei_event.h, 38	
ei_ev_none	
ei_event.h, 38	
ei_event.h	
ei_ev_app, 38	
ei_ev_keydown, 38	
ei_ev_keyup, 38	
ei_ev_last, 38	
ei_ev_mouse_buttondown, 38	
ei_ev_mouse_buttonup, 38	
ei_ev_mouse_move, 38	
ei_ev_none, 38	
ei_mod_alt_left, 38	
ei_mod_alt_right, 38	
ei_mod_ctrl_left, 38	
ei_mod_ctrl_right, 38	
ei_mod_meta_left, 38	
ei_mod_meta_right, 38	
ei_mod_shift_left, 38	
ei_mod_shift_right, 38	

**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 `el_font_t hw_text_font_create (const char * filename, el_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 (el_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_MOUSEBUTTONDOWN_LEFT`

6.11.4.2 `const int EI_MOUSEBUTTONDOWN_MIDDLE`

6.11.4.3 `const int EI_MOUSEBUTTONDOWN_RIGHT`

### 1.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:  
include . . . . . 7

## Directory Hierarchy

## Chapter 1

**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, then 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 given 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 2Data Structure Index

## 2.1 Data Structures

Here are the data structures with brief descriptions:

9 `ei_app_event_t` (The event parameter for application defined event types) . . .  
10 `ei_color_t` (A color with transparency) . . .  
11 `ei_event_t` (Describes an event) . . .  
12 `ei_geometry_param_t` (A structure that stores information about the geometry manager managing a widget, and the widget's geometry management parameters. This is a generic type. Each geometry manager adds field after "manager") . . .  
13 `ei_geometry_manager_t` (The structure that stores information about a geometry manager)  
14 `ei_linked_event_t` (The event parameter for keyboard-related event types) . . .  
15 `ei_linked_point_t` (A point plus a pointer to create a linked list) . . .  
16 `ei_linked_rect_t` (A rectangle plus a pointer to create a linked list) . . .  
17 `ei_linked_tag_t` (A tag and a pointer to create a linked list) . . .  
18 `ei_mouse_event_t` (The event parameter for mouse-related event types) . . .  
19 `ei_point_t` (A 2-D point with integer coordinates) . . .  
20 `ei_rect_t` (A rectangle defined by its top-left corner, and its size) . . .  
21 `ei_size_t` (A 2-D size with integer dimensions) . . .  
22 `ei_widget_t` (Fields common to all types of widget. Every widget classes specializes this base class by adding its own fields) . . .  
23 `ei_widgetclass_t` (A structure that stores information about a class of widgets) . . .  
24  
25  
26  
27

**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 within 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 arbitrary origin. Can be used to measure elapsed 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 3File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

- [ei\\_application.h](#) (Manages the main steps of a graphical application: initial-ization, main window, main loop, quitting, resource freeing) . . . . . 29
- [ei\\_draw.h](#) (Graphical primitives to draw lines, polygons, text, and operation of drawing surfaces) . . . . . 32
- [ei\\_event.h](#) (Allows the binding and unbinding of callbacks to events) . . . . . 36
- [ei\\_geometrymanager.h](#) (Manages the positioning and sizing of widgets on the screen) . . . . . 40
- [ei\\_main.h](#) (Declares the "ei\_main" function: the main function of programs built with the libei) . . . . . 45
- [ei\\_parser.h](#) . . . . . 46
- [ei\\_types.h](#) (Type, constant, and global definitions for the ei library) . . . . . 47
- [ei\\_utils.h](#) (General purpose utility functions: creation of points and sizes, and arithmetics on them) . . . . . 52
- [ei\\_widgectl.h](#) (API for widgets management: creation, configuration, hierar-chy, redisplay) . . . . . 55
- [ei\\_widgectl.h](#) . . . . . 60
- [hw\\_interface.h](#) (Low level interface with the graphic hardware. This interface is based on the SDL library) . . . . . 64

### 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\\_apd](#) (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\\_apd](#) (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 arbitray origin. Can be used to measure elapsed time between to calls.*

## Variables

- const int [EL\\_MOUSEBUTTON\\_LEFT](#)
- const int [EL\\_MOUSEBUTTON\\_MIDDLE](#)
- const int [EL\\_MOUSEBUTTON\\_RIGHT](#)

### 6.11.1 Detailed Description

Low level interface with the graphic hardware. 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 dimensional 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 4Directory 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 positioning 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\\_widgect.h](#)  
*API for widgets management: creation, configuration, hierarchy, redisplay.*

- file [ei\\_widgetclass.h](#)
- file [hw\\_interface.h](#)

*Low level interface with the graphic hardware. 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 within 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 given 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 hardware. 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 widget 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_widgetclass_t` structure itself, passed as parameter, must `*not*` be 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_setdefaultfunc_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 initialize.

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 widgeted 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\* be 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\_event\_type\_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 is a 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 is a 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 `el_default_font`.

*text\_color* The color used to display the text. Defaults to `el_font_default_color`.

*text\_anchor* The anchor of the text, i.e. where it is placed within the widget when the size of the widget is bigger than the size of the text. Defaults to `el_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 within the widget when the size of the widget is bigger than the size of the image. Defaults to `el_anc_center`.

6.9.3.3 void `el_toplevel_configure` (`el_widget_t * widget`, `el_size_t * requested_size`, `el_color_t * color`, `int * border_width`, `char ** title`, `el_bool_t * closable`, `el_axis_sel_t * resizable`, `el_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 `el_default_background_color`.

*border\_width* The width in pixel of the border of the widget. Defaults to 4.

*title* The string title displayed 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 `EL_TRUE`.

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

*min\_size* For resizable widgets, defines the minimum size. Defaults to (160, 120).

6.9.3.4 `el_widget_t * el_widget_create` (`el_widgetclass_name_t class_name`, `el_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 `el_widget_destroy` when no more needed.

5.5 `el_geometrymanager_t` Struct Reference

The structure that stores information about a geometry manager.

```
#include <el_geometrymanager.h>
```

Data Fields

- `el_geometrymanager_name_t name`
- `el_geometrymanager_runfunc_t runfunc`
- `el_geometrymanager_releasefunc_t releasefunc`
- `struct el_geometrymanager_t * next`

5.5.1 Detailed Description

The structure that stores information about a geometry manager.

Definition at line 44 of file `el_geometrymanager.h`.

5.5.2 Field Documentation

5.5.2.1 `el_geometrymanager_name_t name`

Definition at line 45 of file `el_geometrymanager.h`.

5.5.2.2 `struct el_geometrymanager_t * el_geometrymanager_t::next` [ `read` ]

Definition at line 48 of file `el_geometrymanager.h`.

5.5.2.3 `el_geometrymanager_releasefunc_t el_geometrymanager_t::releasefunc`

Definition at line 47 of file `el_geometrymanager.h`.

5.5.2.4 `el_geometrymanager_runfunc_t el_geometrymanager_t::runfunc`

Definition at line 46 of file `el_geometrymanager.h`.

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

- `el_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 attribues of widgets of the class "button".

- void `et_toplevel_configure` (`et_widget_t` \*`widget`, `et_size_t` \*`requested_size`, `et_color_t` \*`color`, `int` \*`border_width`, `char` \*\*`title`, `et_bool_t` \*`closable`, `et_axis_set_t` \*`resizable`, `et_size_t` \*`min_size`)  
Configures the attribues 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 `et_widget.h`.

6.9.2 Typedef Documentation

**6.9.2.1** `typedef et_bool_t (*et_callback_t)(et_widget_t *widget, struct et_event_t *event, void *user_param)`

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

`et_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 `et_widget.h`.

5.7 et\_linked\_point\_t Struct Reference

A point plus a pointer to create a linked list.

```
#include <et_types.h>
```

Data Fields

- `et_point_t` `point`  
The point.
- struct `et_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 `et_types.h`.

5.7.2 Field Documentation

**5.7.2.1** `struct et_linked_point_t * et_linked_point_t::next` [read]

The pointer to the next element in the linked list.

Definition at line 69 of file `et_types.h`.

5.7.2.2 et\_point\_t et\_linked\_point\_t::point

The point.

Definition at line 68 of file `et_types.h`.

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

- `et_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 el\_rect\_t el\_rect\_zero () [ inline, static]

Returns a [el\\_rect\\_t](#) located in (0, 0) and of size (0, 0).  
Definition at line 113 of file [el\\_utils.h](#).

6.8.2.8 static el\_size\_t el\_size (int width, int height) [ inline, static]

Returns a [el\\_size\\_t](#) initialized with the width and height passed as parameters.  
Definition at line 27 of file [el\\_utils.h](#).

6.8.2.9 static el\_size\_t el\_size\_add (el\_size\_t s1, el\_size\_t s2) [ inline, static]

Returns a [el\\_size\\_t](#) which components are the sum of the components of the two sizes passed as parameters.  
Definition at line 92 of file [el\\_utils.h](#).

6.8.2.10 static el\_size\_t el\_size\_sub (el\_size\_t s1, el\_size\_t s2) [ inline, static]

Returns [el\\_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 [el\\_utils.h](#).

6.8.2.11 static el\_size\_t el\_size\_zero () [ inline, static]

Returns a [el\\_size\\_t](#) with width = 0 and height = 0.  
Definition at line 18 of file [el\\_utils.h](#).

5.9 el\_linked\_tag\_t Struct Reference

A tag and a pointer to create a linked list.  
#include <el\_event.h>

Data Fields

- [el\\_tag\\_t tag](#)
- struct [el\\_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 [el\\_event.h](#).

5.9.2 Field Documentation

5.9.2.1 struct [el\\_linked\\_tag\\_t \\* el\\_linked\\_tag\\_t::next](#) [ read]

Definition at line 28 of file [el\\_event.h](#).

5.9.2.2 [el\\_tag\\_t el\\_linked\\_tag\\_t::tag](#)

Definition at line 27 of file [el\\_event.h](#).

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

- [el\\_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::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::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 el\_fontstyle\_t

Font style.

Enumerator:

*el\_style\_normal*

*el\_style\_bold*

*el\_style\_italic*

*el\_style\_underline*

*el\_style\_strikethrough*

Definition at line 140 of file el\_types.h.

6.7.3.5 enum el\_relief\_t

Type of relief.

Enumerator:

*el\_relief\_none* No relief (i.e. flat).

*el\_relief\_raised* Above the screen.

*el\_relief\_sunken* Inside the screen.

Definition at line 111 of file el\_types.h.

## 6.7.4 Variable Documentation

6.7.4.1 const el\_color\_t el\_default\_background\_color = { 0xA0, 0xA0, 0xA0, 0xFF } [static]

The default background color of widgets.

Definition at line 88 of file el\_types.h.

6.7.4.2 el\_font\_t el\_default\_font

The default font used in widgets.

6.7.4.3 const char el\_default\_font\_filename[] = "misc/font.ttf" [static]

Definition at line 163 of file el\_types.h.

6.7.4.4 const el\_color\_t el\_font\_default\_color = { 0x00, 0x00, 0x00, 0xFF } [static]

Definition at line 162 of file el\_types.h.

## 5.13 el\_size\_t Struct Reference

A 2-D size with integer dimensions.

#include <el\_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 el\_types.h.

### 5.13.2 Field Documentation

5.13.2.1 int el\_size\_t::height

Definition at line 45 of file el\_types.h.

5.13.2.2 int el\_size\_t::width

Definition at line 44 of file el\_types.h.

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

- el\_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\_font\_default\_color = { 0xA0, 0xA0, 0xA0, 0xFF }

static const ei\_font\_default\_size = 22

Default font color.

static const int ei\_font\_default\_size = 22

The default font used in widgets.

ei\_font\_t ei\_default\_font

The default background color of widgets.

static const ei\_color\_t ei\_default\_background\_color = { 0xA0, 0xA0, 0xA0, 0xFF }

static const char ei\_default\_font\_filename [] = "misc/font.ttf"

Type, constant, and global definitions for the ei library. Created by François Bérard on 18.12.11. Copyright 2011 Ensmimag. 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 default, 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 }
  - 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 widget.

```
#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\\_widgetclass\\_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\_widgetclass\_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 access 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 invalidated (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:** *widgel* 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\_widgel\_t \*widgel)**

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:** *widgel* 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\_widgel\_t \*widgel)**

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\_widgel\_t\* ei\_app\_root\_widgel()**

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 positioning 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 positioning and sizing of widgets on the screen. #include "ei\_geometrymanager.h" #include "ei\_widget.h"

Data Structures

- struct ei\_geometrymanager\_t  
The structure that stores information about a geometry manager.  
A structure that stores information about the geometry manager managing a widget, and the widget's geometry management parameters. This is a the generic type. Each geometry manager adds field after "manager".
- struct ei\_geometry\_param\_t  
The structure that stores information about the geometry 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 junction 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 \* ei\_geometrymanager\_from\_name (ei\_geometrymanager\_name\_t name)  
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\_unmap(ei\_widget\_t \*widget)  
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, 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 weighed 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\_poline(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 if 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\_event\_type\_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\_buttonup* A mouse button has been pressed.

*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\_event\_type\_t *event\_type*, 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:

*event\_type* 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\\_buttonmousedown](#), [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 if 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](#).