

GaeGebra

v1.0

Generated by Doxygen 1.10.0

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 AngleBisector Struct Reference	5
3.1.1 Member Data Documentation	5
3.1.1.1 base	5
3.1.1.2 line1	5
3.1.1.3 line2	5
3.2 AppData Struct Reference	5
3.2.1 Member Data Documentation	6
3.2.1.1 delta_time	6
3.2.1.2 frame_start	6
3.2.1.3 last_frame_start	6
3.2.1.4 target_frame_time	6
3.2.1.5 windows	6
3.3 Circle Struct Reference	6
3.3.1 Member Data Documentation	6
3.3.1.1 base	6
3.3.1.2 center	6
3.3.1.3 perimeter_point	7
3.4 CoordinateSystem Struct Reference	7
3.4.1 Member Data Documentation	7
3.4.1.1 intersection_points	7
3.4.1.2 origin	7
3.4.1.3 position	7
3.4.1.4 shapes	7
3.4.1.5 size	7
3.4.1.6 zoom	7
3.5 Font Struct Reference	8
3.5.1 Member Data Documentation	8
3.5.1.1 font	8
3.5.1.2 size	8
3.6 InputData Struct Reference	8
3.6.1 Member Data Documentation	8
3.6.1.1 current_keyboard_state	8
3.6.1.2 current_mouse_button_state	8
3.6.1.3 current_mouse_position	8
3.6.1.4 key_count	8
3.6.1.5 mouse_wheel_delta	9

3.6.1.6 old_keyboard_state	9
3.6.1.7 old_mouse_button_state	9
3.6.1.8 old_mouse_position	9
3.7 Line Struct Reference	9
3.7.1 Member Data Documentation	9
3.7.1.1 base	9
3.7.1.2 p1	9
3.7.1.3 p2	9
3.8 Parallel Struct Reference	10
3.8.1 Member Data Documentation	10
3.8.1.1 base	10
3.8.1.2 line	10
3.8.1.3 point	10
3.9 Perpendicular Struct Reference	10
3.9.1 Member Data Documentation	10
3.9.1.1 base	10
3.9.1.2 line	10
3.9.1.3 point	10
3.10 Point Struct Reference	11
3.10.1 Member Data Documentation	11
3.10.1.1 base	11
3.10.1.2 coordinates	11
3.11 Shape Struct Reference	11
3.11.1 Member Data Documentation	11
3.11.1.1 dragged	11
3.11.1.2 selected	11
3.11.1.3 type	11
3.12 Tangent Struct Reference	12
3.12.1 Member Data Documentation	12
3.12.1.1 base	12
3.12.1.2 circle	12
3.12.1.3 point	12
3.13 Texture Struct Reference	12
3.13.1 Member Data Documentation	12
3.13.1.1 height	12
3.13.1.2 texture	12
3.13.1.3 width	12
3.14 UIButton Struct Reference	13
3.14.1 Member Data Documentation	13
3.14.1.1 base	13
3.14.1.2 color	13
3.14.1.3 corner_radius	13

3.14.1.4 mouse_state	13
3.14.1.5 on_click	13
3.14.1.6 text	13
3.14.1.7 text_color	13
3.14.1.8 text_position	13
3.15 UICheckbox Struct Reference	14
3.15.1 Member Data Documentation	14
3.15.1.1 base	14
3.15.1.2 checked	14
3.15.1.3 checked_color	14
3.15.1.4 corner_radius	14
3.15.1.5 mouse_state	14
3.15.1.6 on_checked_changed	14
3.15.1.7 unchecked_color	14
3.16 UIConstraint Struct Reference	14
3.16.1 Member Data Documentation	15
3.16.1.1 constraint_type	15
3.16.1.2 value	15
3.17 UIConstraints Struct Reference	15
3.17.1 Member Data Documentation	15
3.17.1.1 height	15
3.17.1.2 width	15
3.17.1.3 x	15
3.17.1.4 y	15
3.18 UIContainer Struct Reference	15
3.18.1 Member Data Documentation	16
3.18.1.1 base	16
3.18.1.2 children	16
3.18.1.3 on_size_changed	16
3.19 UIData Struct Reference	16
3.19.1 Member Data Documentation	16
3.19.1.1 backspace_pressed	16
3.19.1.2 expanded_splitbutton	16
3.19.1.3 main_container	16
3.19.1.4 mouse_captured	16
3.19.1.5 text_input	17
3.20 UIDropdownList Struct Reference	17
3.20.1 Member Data Documentation	17
3.20.1.1 base	17
3.20.1.2 color	17
3.20.1.3 corner_radius	17
3.20.1.4 expanded	17

3.20.1.5 items	17
3.20.1.6 on_selection_changed	17
3.20.1.7 selected_item	17
3.20.1.8 text_color	18
3.21 UIElement Struct Reference	18
3.21.1 Member Data Documentation	18
3.21.1.1 constraints	18
3.21.1.2 destroy	18
3.21.1.3 parent	18
3.21.1.4 position	18
3.21.1.5 recalculate	18
3.21.1.6 render	18
3.21.1.7 shown	18
3.21.1.8 size	19
3.21.1.9 update	19
3.22 UIButton Struct Reference	19
3.22.1 Member Data Documentation	19
3.22.1.1 base	19
3.22.1.2 mouse_state	19
3.22.1.3 on_click	19
3.22.1.4 texture	19
3.23 UILabel Struct Reference	19
3.23.1 Member Data Documentation	20
3.23.1.1 base	20
3.23.1.2 color	20
3.23.1.3 text	20
3.24 UIPanel Struct Reference	20
3.24.1 Member Data Documentation	20
3.24.1.1 base	20
3.24.1.2 border_color	20
3.24.1.3 border_width	20
3.24.1.4 color	20
3.24.1.5 corner_radius	21
3.25 UISlider Struct Reference	21
3.25.1 Member Data Documentation	21
3.25.1.1 base	21
3.25.1.2 color	21
3.25.1.3 corner_radius	21
3.25.1.4 mouse_state	21
3.25.1.5 on_value_changed	21
3.25.1.6 slider_color	21
3.25.1.7 thickness	21

3.25.1.8 value	22
3.26 UISplitButton Struct Reference	22
3.26.1 Member Data Documentation	22
3.26.1.1 auto_dropdown	22
3.26.1.2 base	22
3.26.1.3 color	22
3.26.1.4 corner_radius	22
3.26.1.5 expanded	22
3.26.1.6 items	22
3.26.1.7 on_item_clicked	22
3.26.1.8 text_color	23
3.27 UITextbox Struct Reference	23
3.27.1 Member Data Documentation	23
3.27.1.1 base	23
3.27.1.2 color	23
3.27.1.3 corner_radius	23
3.27.1.4 focused	23
3.27.1.5 mouse_state	23
3.27.1.6 on_text_changed	23
3.27.1.7 text	23
3.27.1.8 text_color	24
3.28 Vector Struct Reference	24
3.28.1 Member Data Documentation	24
3.28.1.1 capacity	24
3.28.1.2 data	24
3.28.1.3 size	24
3.29 Vector2 Struct Reference	24
3.29.1 Member Data Documentation	24
3.29.1.1 x	24
3.29.1.2 y	25
3.30 Window Struct Reference	25
3.30.1 Member Data Documentation	25
3.30.1.1 close_requested	25
3.30.1.2 input_data	25
3.30.1.3 renderer	25
3.30.1.4 ui_data	25
3.30.1.5 window	25
4 File Documentation	27
4.1 src/app/app.c File Reference	27
4.1.1 Function Documentation	27
4.1.1.1 _app_add_window()	27

4.1.1.2 app_close()	27
4.1.1.3 app_get_delta_time()	28
4.1.1.4 app_get_target()	28
4.1.1.5 app_get_time()	28
4.1.1.6 app_get_windows()	28
4.1.1.7 app_init()	28
4.1.1.8 app_render()	29
4.1.1.9 app_request_close()	29
4.1.1.10 app_set_target()	29
4.1.1.11 app_set_target_fps()	29
4.1.1.12 app_update()	29
4.2 src/app/app.h File Reference	29
4.2.1 Typedef Documentation	30
4.2.1.1 AppData	30
4.2.2 Function Documentation	30
4.2.2.1 _app_add_window()	30
4.2.2.2 app_close()	30
4.2.2.3 app_get_delta_time()	30
4.2.2.4 app_get_target()	31
4.2.2.5 app_get_time()	31
4.2.2.6 app_get_windows()	31
4.2.2.7 app_init()	31
4.2.2.8 app_render()	31
4.2.2.9 app_request_close()	31
4.2.2.10 app_set_target()	31
4.2.2.11 app_set_target_fps()	32
4.2.2.12 app_update()	32
4.3 app.h	32
4.4 src/color/color.c File Reference	33
4.4.1 Function Documentation	33
4.4.1.1 color_clever_shift()	33
4.4.1.2 color_fade()	33
4.4.1.3 color_from_grayscale()	34
4.4.1.4 color_from_hex()	34
4.4.1.5 color_from_hsv()	34
4.4.1.6 color_from_rgb()	35
4.4.1.7 color_from_rgba()	35
4.4.1.8 color_shift()	35
4.5 src/color/color.h File Reference	36
4.5.1 Macro Definition Documentation	36
4.5.1.1 BLACK	36
4.5.1.2 BLUE	36

4.5.1.3 CYAN	36
4.5.1.4 DARK_GRAY	36
4.5.1.5 GRAY	36
4.5.1.6 GREEN	37
4.5.1.7 MAGENTA	37
4.5.1.8 RED	37
4.5.1.9 TRANSPARENT	37
4.5.1.10 WHITE	37
4.5.1.11 YELLOW	37
4.5.2 Typedef Documentation	37
4.5.2.1 Color	37
4.5.3 Function Documentation	37
4.5.3.1 color_clever_shift()	37
4.5.3.2 color_fade()	38
4.5.3.3 color_from_grayscale()	38
4.5.3.4 color_from_hex()	38
4.5.3.5 color_from_hsv()	39
4.5.3.6 color_from_rgb()	39
4.5.3.7 color_from_rgba()	39
4.5.3.8 color_shift()	40
4.6 color.h	40
4.7 src/font/font.c File Reference	41
4.7.1 Function Documentation	41
4.7.1.1 _font_close()	41
4.7.1.2 _font_init()	41
4.7.1.3 font_load()	41
4.8 src/font/font.h File Reference	41
4.8.1 Typedef Documentation	42
4.8.1.1 Font	42
4.8.2 Function Documentation	42
4.8.2.1 _font_close()	42
4.8.2.2 _font_init()	42
4.8.2.3 font_load()	42
4.9 font.h	42
4.10 src/geometry/coordinate_system/coordinate_system.c File Reference	43
4.10.1 Function Documentation	43
4.10.1.1 coordinate_system_clear()	43
4.10.1.2 coordinate_system_create()	44
4.10.1.3 coordinate_system_delete_selected_shapes()	44
4.10.1.4 coordinate_system_deselect_shape()	44
4.10.1.5 coordinate_system_deselect_shapes()	44
4.10.1.6 coordinate_system_destroy()	45

4.10.1.7	<code>coordinate_system_destroy_shape()</code>	45
4.10.1.8	<code>coordinate_system_drag_selected_shapes()</code>	45
4.10.1.9	<code>coordinate_system_draw()</code>	45
4.10.1.10	<code>coordinate_system_get_hovered_shape()</code>	46
4.10.1.11	<code>coordinate_system_get_selected_shapes()</code>	46
4.10.1.12	<code>coordinate_system_is_hovered()</code>	46
4.10.1.13	<code>coordinate_system_load()</code>	47
4.10.1.14	<code>coordinate_system_save()</code>	47
4.10.1.15	<code>coordinate_system_select_all_shapes()</code>	47
4.10.1.16	<code>coordinate_system_select_shape()</code>	48
4.10.1.17	<code>coordinate_system_translate()</code>	48
4.10.1.18	<code>coordinate_system_update()</code>	48
4.10.1.19	<code>coordinate_system_update_dimensions()</code>	48
4.10.1.20	<code>coordinate_system_zoom()</code>	49
4.10.1.21	<code>coordinates_to_screen()</code>	49
4.10.1.22	<code>screen_to_coordinates()</code>	49
4.11	<code>src/geometry/coordinate_system/coordinate_system.h</code> File Reference	50
4.11.1	Macro Definition Documentation	50
4.11.1.1	<code>INITIAL_ZOOM</code>	50
4.11.2	Typedef Documentation	50
4.11.2.1	<code>CoordinateSystem</code>	50
4.11.3	Function Documentation	50
4.11.3.1	<code>coordinate_system_clear()</code>	50
4.11.3.2	<code>coordinate_system_create()</code>	51
4.11.3.3	<code>coordinate_system_delete_selected_shapes()</code>	51
4.11.3.4	<code>coordinate_system_deselect_shape()</code>	51
4.11.3.5	<code>coordinate_system_deselect_shapes()</code>	52
4.11.3.6	<code>coordinate_system_destroy()</code>	52
4.11.3.7	<code>coordinate_system_destroy_shape()</code>	52
4.11.3.8	<code>coordinate_system_drag_selected_shapes()</code>	52
4.11.3.9	<code>coordinate_system_draw()</code>	53
4.11.3.10	<code>coordinate_system_get_hovered_shape()</code>	53
4.11.3.11	<code>coordinate_system_get_selected_shapes()</code>	53
4.11.3.12	<code>coordinate_system_is_hovered()</code>	53
4.11.3.13	<code>coordinate_system_load()</code>	54
4.11.3.14	<code>coordinate_system_save()</code>	54
4.11.3.15	<code>coordinate_system_select_all_shapes()</code>	54
4.11.3.16	<code>coordinate_system_select_shape()</code>	55
4.11.3.17	<code>coordinate_system_translate()</code>	55
4.11.3.18	<code>coordinate_system_update()</code>	55
4.11.3.19	<code>coordinate_system_update_dimensions()</code>	55
4.11.3.20	<code>coordinate_system_zoom()</code>	56

4.11.3.21 <code>coordinates_to_screen()</code>	56
4.11.3.22 <code>screen_to_coordinates()</code>	56
4.12 <code>coordinate_system.h</code>	57
4.13 <code>src/geometry/intersection/intersection.c</code> File Reference	57
4.13.1 Macro Definition Documentation	57
4.13.1.1 EPSILON	57
4.13.2 Function Documentation	58
4.13.2.1 <code>intersection_get()</code>	58
4.14 <code>src/geometry/intersection/intersection.h</code> File Reference	58
4.14.1 Function Documentation	58
4.14.1.1 <code>intersection_get()</code>	58
4.15 <code>intersection.h</code>	58
4.16 <code>src/geometry/shape/shape.c</code> File Reference	59
4.16.1 Macro Definition Documentation	59
4.16.1.1 EPSILON	59
4.16.2 Function Documentation	59
4.16.2.1 <code>angle_bisector_create()</code>	59
4.16.2.2 <code>circle_create()</code>	60
4.16.2.3 <code>line_create()</code>	60
4.16.2.4 <code>parallel_create()</code>	60
4.16.2.5 <code>perpendicular_create()</code>	61
4.16.2.6 <code>point_create()</code>	61
4.16.2.7 <code>shape_destroy()</code>	62
4.16.2.8 <code>shape_draw()</code>	62
4.16.2.9 <code>shape_is_defined_by()</code>	62
4.16.2.10 <code>shape_overlap_point()</code>	62
4.16.2.11 <code>shape_translate()</code>	63
4.16.2.12 <code>shape_update()</code>	63
4.16.2.13 <code>tangent_create()</code>	63
4.16.3 Variable Documentation	64
4.16.3.1 <code>shape_destroy_funcs</code>	64
4.16.3.2 <code>shape_draw_funcs</code>	64
4.16.3.3 <code>shape_is_defined_by_funcs</code>	64
4.16.3.4 <code>shape_overlap_point_funcs</code>	64
4.16.3.5 <code>shape_translate_funcs</code>	64
4.17 <code>src/geometry/shape/shape.h</code> File Reference	65
4.17.1 Macro Definition Documentation	65
4.17.1.1 OVERLAP_DISTANCE	65
4.17.2 Typedef Documentation	65
4.17.2.1 AngleBisector	65
4.17.2.2 Circle	65
4.17.2.3 CoordinateSystem	66

4.17.2.4 Line	66
4.17.2.5 Parallel	66
4.17.2.6 Perpendicular	66
4.17.2.7 Point	66
4.17.2.8 Shape	66
4.17.2.9 ShapeDestroy	66
4.17.2.10 ShapeDraw	66
4.17.2.11 ShapesDefinedBy	67
4.17.2.12 ShapeOverlapPoint	67
4.17.2.13 ShapeTranslate	67
4.17.2.14 ShapeType	67
4.17.2.15 Tangent	67
4.17.3 Enumeration Type Documentation	67
4.17.3.1 ShapeType	67
4.17.4 Function Documentation	68
4.17.4.1 angle_bisector_create()	68
4.17.4.2 circle_create()	68
4.17.4.3 line_create()	68
4.17.4.4 parallel_create()	69
4.17.4.5 perpendicular_create()	69
4.17.4.6 point_create()	70
4.17.4.7 shape_destroy()	70
4.17.4.8 shape_draw()	70
4.17.4.9 shape_is_defined_by()	70
4.17.4.10 shape_overlap_point()	71
4.17.4.11 shape_translate()	71
4.17.4.12 shape_update()	72
4.17.4.13 tangent_create()	72
4.18 shape.h	72
4.19 src/geometry/vector2/vector2.c File Reference	73
4.19.1 Function Documentation	74
4.19.1.1 vector2_add()	74
4.19.1.2 vector2_angle()	74
4.19.1.3 vector2_create()	75
4.19.1.4 vector2_cross()	75
4.19.1.5 vector2_distance()	75
4.19.1.6 vector2_divide()	76
4.19.1.7 vector2_dot()	76
4.19.1.8 vector2_down()	76
4.19.1.9 vector2_from_point()	77
4.19.1.10 vector2_from_polar()	77
4.19.1.11 vector2_left()	77

4.19.1.12 vector2_length()	78
4.19.1.13 vector2_multiply()	79
4.19.1.14 vector2_negate()	79
4.19.1.15 vector2_normalize()	79
4.19.1.16 vector2_one()	80
4.19.1.17 vector2_reflect()	80
4.19.1.18 vector2_right()	80
4.19.1.19 vector2_rotate()	81
4.19.1.20 vector2_rotate90()	82
4.19.1.21 vector2_scale()	82
4.19.1.22 vector2_subtract()	82
4.19.1.23 vector2_up()	83
4.19.1.24 vector2_zero()	83
4.20 src/geometry/vector2/vector2.h File Reference	83
4.20.1 Typedef Documentation	84
4.20.1.1 Vector2	84
4.20.2 Function Documentation	84
4.20.2.1 vector2_add()	84
4.20.2.2 vector2_angle()	85
4.20.2.3 vector2_create()	85
4.20.2.4 vector2_cross()	85
4.20.2.5 vector2_distance()	86
4.20.2.6 vector2_divide()	86
4.20.2.7 vector2_dot()	86
4.20.2.8 vector2_down()	87
4.20.2.9 vector2_from_point()	87
4.20.2.10 vector2_from_polar()	87
4.20.2.11 vector2_left()	88
4.20.2.12 vector2_length()	88
4.20.2.13 vector2_multiply()	88
4.20.2.14 vector2_negate()	88
4.20.2.15 vector2_normalize()	89
4.20.2.16 vector2_one()	89
4.20.2.17 vector2_reflect()	89
4.20.2.18 vector2_right()	90
4.20.2.19 vector2_rotate()	90
4.20.2.20 vector2_rotate90()	90
4.20.2.21 vector2_scale()	91
4.20.2.22 vector2_subtract()	91
4.20.2.23 vector2_up()	91
4.20.2.24 vector2_zero()	91
4.21 vector2.h	92

4.22 src/input/input.c File Reference	92
4.22.1 Function Documentation	93
4.22.1.1 _input_close()	93
4.22.1.2 _input_handle_event()	94
4.22.1.3 _input_init()	94
4.22.1.4 _input_reset()	94
4.22.1.5 _input_set_target()	94
4.22.1.6 input_get_mouse_motion()	95
4.22.1.7 input_get_mouse_position()	95
4.22.1.8 input_get_mouse_wheel_delta()	95
4.22.1.9 input_is_key_down()	95
4.22.1.10 input_is_key_pressed()	96
4.22.1.11 input_is_key_released()	96
4.22.1.12 input_is_mouse_button_down()	96
4.22.1.13 input_is_mouse_button_pressed()	97
4.22.1.14 input_is_mouse_button_released()	97
4.23 src/input/input.h File Reference	97
4.23.1 Typedef Documentation	98
4.23.1.1 InputData	98
4.23.2 Function Documentation	98
4.23.2.1 _input_close()	98
4.23.2.2 _input_handle_event()	98
4.23.2.3 _input_init()	99
4.23.2.4 _input_reset()	99
4.23.2.5 _input_set_target()	99
4.23.2.6 input_get_mouse_motion()	99
4.23.2.7 input_get_mouse_position()	100
4.23.2.8 input_get_mouse_wheel_delta()	100
4.23.2.9 input_is_key_down()	100
4.23.2.10 input_is_key_pressed()	100
4.23.2.11 input_is_key_released()	101
4.23.2.12 input_is_mouse_button_down()	101
4.23.2.13 input_is_mouse_button_pressed()	101
4.23.2.14 input_is_mouse_button_released()	102
4.24 input.h	102
4.25 src/main.c File Reference	103
4.25.1 Detailed Description	103
4.25.2 Macro Definition Documentation	103
4.25.2.1 FPS	103
4.25.2.2 MOUSE_WHEEL_SENSITIVITY	103
4.25.3 Typedef Documentation	103
4.25.3.1 State	103

4.25.4 Enumeration Type Documentation	104
4.25.4.1 State	104
4.25.5 Function Documentation	104
4.25.5.1 main()	104
4.25.5.2 on_angle_bisector_clicked() [1/2]	104
4.25.5.3 on_angle_bisector_clicked() [2/2]	104
4.25.5.4 on_cancel_button_clicked()	105
4.25.5.5 on_canvas_size_changed()	105
4.25.5.6 on_circle_clicked() [1/2]	105
4.25.5.7 on_circle_clicked() [2/2]	105
4.25.5.8 on_editmenu_clicked() [1/2]	105
4.25.5.9 on_editmenu_clicked() [2/2]	105
4.25.5.10 on_filemenu_clicked() [1/2]	105
4.25.5.11 on_filemenu_clicked() [2/2]	105
4.25.5.12 on_line_clicked() [1/2]	106
4.25.5.13 on_line_clicked() [2/2]	106
4.25.5.14 on_open_button_clicked()	106
4.25.5.15 on_parallel_clicked() [1/2]	106
4.25.5.16 on_parallel_clicked() [2/2]	106
4.25.5.17 on_perpendicular_clicked() [1/2]	106
4.25.5.18 on_perpendicular_clicked() [2/2]	106
4.25.5.19 on_point_clicked() [1/2]	106
4.25.5.20 on_point_clicked() [2/2]	106
4.25.5.21 on_pointer_clicked() [1/2]	107
4.25.5.22 on_pointer_clicked() [2/2]	107
4.25.5.23 on_save_button_clicked()	107
4.25.5.24 on_tangent_clicked() [1/2]	107
4.25.5.25 on_tangent_clicked() [2/2]	107
4.25.6 Variable Documentation	107
4.25.6.1 cs	107
4.25.6.2 state	107
4.26 src/renderer/renderer.c File Reference	108
4.26.1 Function Documentation	108
4.26.1.1 _renderer_set_target()	108
4.26.1.2 renderer_bind_framebuffer()	108
4.26.1.3 renderer_clear()	109
4.26.1.4 renderer_create_framebuffer()	109
4.26.1.5 renderer_draw_arc()	109
4.26.1.6 renderer_draw_bezier()	110
4.26.1.7 renderer_draw_circle()	110
4.26.1.8 renderer_draw_ellipse()	110
4.26.1.9 renderer_draw_filled_circle()	111

4.26.1.10	renderer_draw_filled_ellipse()	111
4.26.1.11	renderer_draw_filled_polygon()	112
4.26.1.12	renderer_draw_filled_rect()	112
4.26.1.13	renderer_draw_filled_rounded_rect()	112
4.26.1.14	renderer_draw_filled_triangle()	113
4.26.1.15	renderer_draw_line()	113
4.26.1.16	renderer_draw_pie()	114
4.26.1.17	renderer_draw_pixel()	114
4.26.1.18	renderer_draw_polygon()	114
4.26.1.19	renderer_draw_rect()	115
4.26.1.20	renderer_draw_rounded_rect()	115
4.26.1.21	renderer_draw_text()	116
4.26.1.22	renderer_draw_texture()	116
4.26.1.23	renderer_draw_triangle()	116
4.26.1.24	renderer_query_text_size()	117
4.26.1.25	renderer_reset_clip_rect()	117
4.26.1.26	renderer_resize_framebuffer()	117
4.26.1.27	renderer_set_clip_rect()	118
4.26.1.28	renderer_set_default_font()	118
4.27	src/renderer/renderer.h File Reference	118
4.27.1	Function Documentation	119
4.27.1.1	_renderer_set_target()	119
4.27.1.2	renderer_bind_framebuffer()	119
4.27.1.3	renderer_clear()	119
4.27.1.4	renderer_create_framebuffer()	120
4.27.1.5	renderer_draw_arc()	120
4.27.1.6	renderer_draw_bezier()	120
4.27.1.7	renderer_draw_circle()	121
4.27.1.8	renderer_draw_ellipse()	121
4.27.1.9	renderer_draw_filled_circle()	121
4.27.1.10	renderer_draw_filled_ellipse()	122
4.27.1.11	renderer_draw_filled_polygon()	122
4.27.1.12	renderer_draw_filled_rect()	123
4.27.1.13	renderer_draw_filled_rounded_rect()	123
4.27.1.14	renderer_draw_filled_triangle()	123
4.27.1.15	renderer_draw_line()	124
4.27.1.16	renderer_draw_pie()	124
4.27.1.17	renderer_draw_pixel()	125
4.27.1.18	renderer_draw_polygon()	125
4.27.1.19	renderer_draw_rect()	125
4.27.1.20	renderer_draw_rounded_rect()	126
4.27.1.21	renderer_draw_text()	126

4.27.1.22	renderer_draw_texture()	127
4.27.1.23	renderer_draw_triangle()	127
4.27.1.24	renderer_query_text_size()	127
4.27.1.25	renderer_reset_clip_rect()	128
4.27.1.26	renderer_resize_framebuffer()	128
4.27.1.27	renderer_set_clip_rect()	128
4.27.1.28	renderer_set_default_font()	129
4.28	renderer.h	129
4.29	src/texture/texture.c File Reference	129
4.29.1	Function Documentation	130
4.29.1.1	_texture_add()	130
4.29.1.2	_texture_close()	130
4.29.1.3	_texture_init()	130
4.29.1.4	texture_load()	130
4.30	src/texture/texture.h File Reference	130
4.30.1	Typedef Documentation	131
4.30.1.1	Texture	131
4.30.2	Function Documentation	131
4.30.2.1	_texture_add()	131
4.30.2.2	_texture_close()	131
4.30.2.3	_texture_init()	131
4.30.2.4	texture_load()	131
4.31	texture.h	132
4.32	src/ui/ui.c File Reference	132
4.32.1	Function Documentation	132
4.32.1.1	_ui_close()	132
4.32.1.2	_ui_get_target()	133
4.32.1.3	_ui_handle_event()	133
4.32.1.4	_ui_init()	133
4.32.1.5	_ui_render()	133
4.32.1.6	_ui_set_target()	134
4.32.1.7	_ui_update()	134
4.32.2	Variable Documentation	134
4.32.2.1	target_ui_data	134
4.33	src/ui/ui.h File Reference	134
4.33.1	Typedef Documentation	135
4.33.1.1	UIData	135
4.33.2	Function Documentation	135
4.33.2.1	_ui_close()	135
4.33.2.2	_ui_get_target()	135
4.33.2.3	_ui_handle_event()	135
4.33.2.4	_ui_init()	136

4.33.2.5 <code>_ui_render()</code>	136
4.33.2.6 <code>_ui_set_target()</code>	136
4.33.2.7 <code>_ui_update()</code>	136
4.34 <code>ui.h</code>	137
4.35 <code>src/ui/ui_constraint/ui_constraint.c</code> File Reference	137
4.35.1 Function Documentation	137
4.35.1.1 <code>constraints_from_string()</code>	137
4.35.1.2 <code>new_aspect_constraint()</code>	138
4.35.1.3 <code>new_center_constraint()</code>	138
4.35.1.4 <code>new_offset_constraint()</code>	138
4.35.1.5 <code>new_pixel_constraint()</code>	138
4.35.1.6 <code>new_relative_constraint()</code>	139
4.36 <code>src/ui/ui_constraint/ui_constraint.h</code> File Reference	139
4.36.1 Typedef Documentation	139
4.36.1.1 <code>ConstraintType</code>	139
4.36.1.2 <code>UIConstraint</code>	140
4.36.1.3 <code>UIConstraints</code>	140
4.36.2 Enumeration Type Documentation	140
4.36.2.1 <code>ConstraintType</code>	140
4.36.3 Function Documentation	140
4.36.3.1 <code>constraints_from_string()</code>	140
4.36.3.2 <code>new_aspect_constraint()</code>	141
4.36.3.3 <code>new_center_constraint()</code>	141
4.36.3.4 <code>new_offset_constraint()</code>	141
4.36.3.5 <code>new_pixel_constraint()</code>	141
4.36.3.6 <code>new_relative_constraint()</code>	142
4.37 <code>ui_constraint.h</code>	142
4.38 <code>src/ui/ui_element/ui_element.c</code> File Reference	143
4.38.1 Typedef Documentation	143
4.38.1.1 <code>_UIDropdownItem</code>	143
4.38.1.2 <code>_UISplitButtonItem</code>	143
4.38.2 Function Documentation	143
4.38.2.1 <code>_ui_container_destroy()</code>	143
4.38.2.2 <code>_ui_container_recalculate()</code>	144
4.38.2.3 <code>_ui_container_render()</code>	144
4.38.2.4 <code>_ui_container_update()</code>	144
4.38.2.5 <code>ui_create_button()</code>	144
4.38.2.6 <code>ui_create_checkbox()</code>	145
4.38.2.7 <code>ui_create_container()</code>	145
4.38.2.8 <code>ui_create_dropdown()</code>	146
4.38.2.9 <code>ui_create_imagebutton()</code>	146
4.38.2.10 <code>ui_create_label()</code>	147

4.38.2.11 ui_create_panel()	147
4.38.2.12 ui_create_slider()	148
4.38.2.13 ui_create_splitbutton()	148
4.38.2.14 ui_create_textbox()	149
4.38.2.15 ui_hide_element()	149
4.38.2.16 ui_show_element()	149
4.39 src/ui/ui_element/ui_element.h File Reference	150
4.39.1 Macro Definition Documentation	150
4.39.1.1 UITEXT_MAX_LENGTH	150
4.39.2 Typedef Documentation	151
4.39.2.1 MouseState	151
4.39.2.2 UIButton	151
4.39.2.3 UIButtonClick	151
4.39.2.4 UICheckbox	151
4.39.2.5 UICheckboxCheckedChanged	151
4.39.2.6 UIContainer	151
4.39.2.7 UIContainerSizeChanged	151
4.39.2.8 UIDropdownList	151
4.39.2.9 UIDropdownListSelectionChanged	152
4.39.2.10 UIElement	152
4.39.2.11 UIElementDestroy	152
4.39.2.12 UIElementRecalculate	152
4.39.2.13 UIElementRender	152
4.39.2.14 UIElementUpdate	152
4.39.2.15 UIImageButton	152
4.39.2.16 UIImageButtonClick	152
4.39.2.17 UILabel	152
4.39.2.18 UIPanel	153
4.39.2.19 UISlider	153
4.39.2.20 UISliderValueChanged	153
4.39.2.21 UISplitButton	153
4.39.2.22 UISplitButtonClicked	153
4.39.2.23 UITextbox	153
4.39.2.24 UITextboxTextChanged	153
4.39.3 Enumeration Type Documentation	153
4.39.3.1 MouseState	153
4.39.4 Function Documentation	154
4.39.4.1 _ui_container_destroy()	154
4.39.4.2 _ui_container_recalculate()	154
4.39.4.3 _ui_container_render()	154
4.39.4.4 _ui_container_update()	155
4.39.4.5 ui_create_button()	155

4.39.4.6 ui_create_checkbox()	155
4.39.4.7 ui_create_container()	156
4.39.4.8 ui_create_dropdown()	156
4.39.4.9 ui_create_imagebutton()	157
4.39.4.10 ui_create_label()	157
4.39.4.11 ui_create_panel()	157
4.39.4.12 ui_create_slider()	158
4.39.4.13 ui_create_splitbutton()	158
4.39.4.14 ui_create_textbox()	159
4.39.4.15 ui_hide_element()	159
4.39.4.16 ui_show_element()	160
4.40 ui_element.h	160
4.41 src/utils/math/math.c File Reference	162
4.41.1 Function Documentation	163
4.41.1.1 check_collision_point_rect()	163
4.41.1.2 clamp()	163
4.41.1.3 deg_to_rad()	163
4.41.1.4 lerp()	164
4.41.1.5 map()	164
4.41.1.6 rad_to_deg()	165
4.42 src/utils/math/math.h File Reference	165
4.42.1 Macro Definition Documentation	165
4.42.1.1 HALF_PI	165
4.42.1.2 PI	165
4.42.1.3 TWO_PI	165
4.42.2 Function Documentation	166
4.42.2.1 check_collision_point_rect()	166
4.42.2.2 clamp()	166
4.42.2.3 deg_to_rad()	166
4.42.2.4 lerp()	167
4.42.2.5 map()	167
4.42.2.6 rad_to_deg()	168
4.43 math.h	168
4.44 src/utils/vector/vector.c File Reference	168
4.44.1 Function Documentation	169
4.44.1.1 vector_clear()	169
4.44.1.2 vector_contains()	169
4.44.1.3 vector_create()	169
4.44.1.4 vector_destroy()	170
4.44.1.5 vector_get()	170
4.44.1.6 vector_index_of()	170
4.44.1.7 vector_insert()	170

4.44.1.8	vector_pop_back()	171
4.44.1.9	vector_push_back()	171
4.44.1.10	vector_remove()	171
4.44.1.11	vector_remove_at()	172
4.44.1.12	vector_reserve()	172
4.44.1.13	vector_set()	172
4.44.1.14	vector_size()	173
4.45	src/utlis/vector/vector.h File Reference	173
4.45.1	Typedef Documentation	173
4.45.1.1	Vector	173
4.45.2	Function Documentation	173
4.45.2.1	vector_clear()	173
4.45.2.2	vector_contains()	174
4.45.2.3	vector_create()	174
4.45.2.4	vector_destroy()	174
4.45.2.5	vector_get()	175
4.45.2.6	vector_index_of()	175
4.45.2.7	vector_insert()	175
4.45.2.8	vector_pop_back()	176
4.45.2.9	vector_push_back()	176
4.45.2.10	vector_remove()	176
4.45.2.11	vector_remove_at()	176
4.45.2.12	vector_reserve()	177
4.45.2.13	vector_set()	177
4.45.2.14	vector_size()	177
4.46	vector.h	178
4.47	src/window/window.c File Reference	178
4.47.1	Function Documentation	178
4.47.1.1	_window_close()	178
4.47.1.2	_window_handle_event()	179
4.47.1.3	_window_render()	179
4.47.1.4	_window_reset()	179
4.47.1.5	_window_update()	179
4.47.1.6	window_create()	180
4.47.1.7	window_focus()	180
4.47.1.8	window_get_main_container()	180
4.47.1.9	window_hide()	181
4.47.1.10	window_show()	181
4.48	src/window/window.h File Reference	181
4.48.1	Typedef Documentation	181
4.48.1.1	Window	181
4.48.2	Function Documentation	182

4.48.2.1 _window_close()	182
4.48.2.2 _window_handle_event()	182
4.48.2.3 _window_render()	182
4.48.2.4 _window_reset()	182
4.48.2.5 _window_update()	183
4.48.2.6 window_create()	183
4.48.2.7 window_focus()	183
4.48.2.8 window_get_main_container()	183
4.48.2.9 window_hide()	185
4.48.2.10 window_show()	185
4.49 window.h	185

Index	187
--------------	------------

Chapter 1

Class Index

1.1 Class List

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

AngleBisector	The angle bisector lines struct	5
AppData	Contains the application data, like windows and target fps. There is only one instance of this struct, and should not be modified directly	5
Circle	The circle struct	6
CoordinateSystem	7
Font	Holds a TTF_Font and its size	8
InputData	Holds the input data of a window (needed for press, hold, release events and mouse motion)	8
Line	The line struct	9
Parallel	The parallel line struct	10
Perpendicular	The perpendicular line struct	10
Point	The point struct	11
Shape	The base shape struct (needed for polymorphism)	11
Tangent	The tangent lines struct (circle tangents)	12
Texture	The texture struct that holds the texture data (SDL_Texture*, width, height)	12
UIButton	The UI button structure	13
UICheckbox	The UI checkbox structure	14
UIConstraint	A constraint for a UIElement . A constraint is a value that can be used to calculate the position or size of a UIElement . A pixel constraint is a fixed value in pixels (can be negative to measure it from the right). A center constraint represents the center of the parent element. A relative constraint is a value between 0 and 1 that represents the percentage of the parent element. An offset constraint represents the offset from the parent element in pixels. An aspect constraint is a value that represents the aspect ratio of a UIElement	14

UIConstraints	
A set of constraints for a UIElement	15
UIContainer	
The UI container structure (provides a container for other UI elements)	15
UIData	
Holds the ui data of a window (contains the main container, the text input of the current frame, whether the backspace was pressed, whether the mouse is captured by a ui element and the expanded splitbutton)	16
UIDropdownList	
The UI dropdown list structure	17
UIElement	
The base UI element structure (needed for polymorphism)	18
UIImageButton	
The UI image button structure	19
UILabel	
The UI label structure	19
UIPanel	
The UI panel structure (colored panel with border)	20
UISlider	
The UI slider structure	21
UISplitButton	
The UI split button structure	22
UITextbox	
The UI textbox structure (has a fixed length)	23
Vector	
A generic vector type (dynamic array for void pointers)	24
Vector2	
A 2D vector, used for coordinate geometry	24
Window	
The Window struct, contains an <code>SDL_Window</code> , an <code>SDL_Renderer</code> and other window specific data, like the input data and the UI data, and a flag to check if the window is requested to be closed	25

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/main.c	
This is the entry point of the application	103
src/app/app.c	27
src/app/app.h	29
src/color/color.c	33
src/color/color.h	36
src/font/font.c	41
src/font/font.h	41
src/geometry/coordinate_system/coordinate_system.c	43
src/geometry/coordinate_system/coordinate_system.h	50
src/geometry/intersection/intersection.c	57
src/geometry/intersection/intersection.h	58
src/geometry/shape/shape.c	59
src/geometry/shape/shape.h	65
src/geometry/vector2/vector2.c	73
src/geometry/vector2/vector2.h	83
src/input/input.c	92
src/input/input.h	97
src/renderer/renderer.c	108
src/renderer/renderer.h	118
src/texture/texture.c	129
src/texture/texture.h	130
src/ui/ui.c	132
src/ui/ui.h	134
src/ui/ui_constraint/ui_constraint.c	137
src/ui/ui_constraint/ui_constraint.h	139
src/ui/ui_element/ui_element.c	143
src/ui/ui_element/ui_element.h	150
src/utils/math/math.c	162
src/utils/math/math.h	165
src/utils/vector/vector.c	168
src/utils/vector/vector.h	173
src/window/window.c	178
src/window/window.h	181

Chapter 3

Class Documentation

3.1 AngleBisector Struct Reference

```
#include <shape.h>
```

3.1.1 Member Data Documentation

3.1.1.1 base

```
Shape AngleBisector::base
```

3.1.1.2 line1

```
Line* AngleBisector::line1
```

3.1.1.3 line2

```
Line* AngleBisector::line2
```

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

- [src/geometry/shape/shape.h](#)

3.2 AppData Struct Reference

```
#include <app.h>
```

3.2.1 Member Data Documentation

3.2.1.1 `delta_time`

```
double AppData::delta_time
```

3.2.1.2 `frame_start`

```
UInt32 AppData::frame_start
```

3.2.1.3 `last_frame_start`

```
UInt32 AppData::last_frame_start
```

3.2.1.4 `target_frame_time`

```
UInt32 AppData::target_frame_time
```

3.2.1.5 `windows`

```
Vector* AppData::windows
```

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

- [src/app/app.h](#)

3.3 Circle Struct Reference

```
#include <shape.h>
```

3.3.1 Member Data Documentation

3.3.1.1 `base`

```
Shape Circle::base
```

3.3.1.2 `center`

```
Point* Circle::center
```

3.3.1.3 perimeter_point

```
Point* Circle::perimeter_point
```

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

- [src/geometry/shape/shape.h](#)

3.4 CoordinateSystem Struct Reference

```
#include <coordinate_system.h>
```

3.4.1 Member Data Documentation

3.4.1.1 intersection_points

```
Vector* CoordinateSystem::intersection_points
```

3.4.1.2 origin

```
Vector2 CoordinateSystem::origin
```

3.4.1.3 position

```
Vector2 CoordinateSystem::position
```

3.4.1.4 shapes

```
Vector* CoordinateSystem::shapes
```

3.4.1.5 size

```
Vector2 CoordinateSystem::size
```

3.4.1.6 zoom

```
double CoordinateSystem::zoom
```

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

- [src/geometry/coordinate_system/coordinate_system.h](#)

3.5 Font Struct Reference

```
#include <font.h>
```

3.5.1 Member Data Documentation

3.5.1.1 font

```
TTF_Font* Font::font
```

3.5.1.2 size

```
int Font::size
```

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

- [src/font/font.h](#)

3.6 InputData Struct Reference

```
#include <input.h>
```

3.6.1 Member Data Documentation

3.6.1.1 current_keyboard_state

```
Uint8* InputData::current_keyboard_state
```

3.6.1.2 current_mouse_button_state

```
bool InputData::current_mouse_button_state[5]
```

3.6.1.3 current_mouse_position

```
SDL_Point InputData::current_mouse_position
```

3.6.1.4 key_count

```
int InputData::key_count
```

3.6.1.5 mouse_wheel_delta

```
int InputData::mouse_wheel_delta
```

3.6.1.6 old_keyboard_state

```
Uint8* InputData::old_keyboard_state
```

3.6.1.7 old_mouse_button_state

```
bool InputData::old_mouse_button_state[5]
```

3.6.1.8 old_mouse_position

```
SDL_Point InputData::old_mouse_position
```

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

- [src/input/input.h](#)

3.7 Line Struct Reference

```
#include <shape.h>
```

3.7.1 Member Data Documentation

3.7.1.1 base

```
Shape Line::base
```

3.7.1.2 p1

```
Point* Line::p1
```

3.7.1.3 p2

```
Point * Line::p2
```

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

- [src/geometry/shape/shape.h](#)

3.8 Parallel Struct Reference

```
#include <shape.h>
```

3.8.1 Member Data Documentation

3.8.1.1 base

```
Shape Parallel::base
```

3.8.1.2 line

```
Line* Parallel::line
```

3.8.1.3 point

```
Point* Parallel::point
```

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

- src/geometry/shape/[shape.h](#)

3.9 Perpendicular Struct Reference

```
#include <shape.h>
```

3.9.1 Member Data Documentation

3.9.1.1 base

```
Shape Perpendicular::base
```

3.9.1.2 line

```
Line* Perpendicular::line
```

3.9.1.3 point

```
Point* Perpendicular::point
```

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

- src/geometry/shape/[shape.h](#)

3.10 Point Struct Reference

```
#include <shape.h>
```

3.10.1 Member Data Documentation

3.10.1.1 base

```
Shape Point::base
```

3.10.1.2 coordinates

```
Vector2 Point::coordinates
```

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

- src/geometry/shape/[shape.h](#)

3.11 Shape Struct Reference

```
#include <shape.h>
```

3.11.1 Member Data Documentation

3.11.1.1 dragged

```
bool Shape::dragged
```

3.11.1.2 selected

```
bool Shape::selected
```

3.11.1.3 type

```
ShapeType Shape::type
```

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

- src/geometry/shape/[shape.h](#)

3.12 Tangent Struct Reference

```
#include <shape.h>
```

3.12.1 Member Data Documentation

3.12.1.1 base

```
Shape Tangent::base
```

3.12.1.2 circle

```
Circle* Tangent::circle
```

3.12.1.3 point

```
Point* Tangent::point
```

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

- [src/geometry/shape/shape.h](#)

3.13 Texture Struct Reference

```
#include <texture.h>
```

3.13.1 Member Data Documentation

3.13.1.1 height

```
int Texture::height
```

3.13.1.2 texture

```
SDL_Texture* Texture::texture
```

3.13.1.3 width

```
int Texture::width
```

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

- [src/texture/texture.h](#)

3.14 UIButton Struct Reference

```
#include <ui_element.h>
```

3.14.1 Member Data Documentation

3.14.1.1 base

[UIElement](#) UIButton::base

3.14.1.2 color

[Color](#) UIButton::color

3.14.1.3 corner_radius

[Uint32](#) UIButton::corner_radius

3.14.1.4 mouse_state

[MouseState](#) UIButton::mouse_state

3.14.1.5 on_click

[UIButtonClick](#) UIButton::on_click

3.14.1.6 text

[char](#) UIButton::text[[UITEXT_MAX_LENGTH](#)+1]

3.14.1.7 text_color

[Color](#) UIButton::text_color

3.14.1.8 text_position

[SDL_Point](#) UIButton::text_position

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

- [src/ui/ui_element/ui_element.h](#)

3.15 UICheckbox Struct Reference

```
#include <ui_element.h>
```

3.15.1 Member Data Documentation

3.15.1.1 base

[UIElement](#) UICheckbox::base

3.15.1.2 checked

bool UICheckbox::checked

3.15.1.3 checked_color

[Color](#) UICheckbox::checked_color

3.15.1.4 corner_radius

UInt32 UICheckbox::corner_radius

3.15.1.5 mouse_state

[MouseState](#) UICheckbox::mouse_state

3.15.1.6 on_checked_changed

[UICheckboxCheckedChanged](#) UICheckbox::on_checked_changed

3.15.1.7 unchecked_color

[Color](#) UICheckbox::unchecked_color

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

- [src/ui/ui_element/ui_element.h](#)

3.16 UIConstraint Struct Reference

```
#include <ui_constraint.h>
```

3.16.1 Member Data Documentation

3.16.1.1 `constraint_type`

`ConstraintType` `UIConstraint::constraint_type`

3.16.1.2 `value`

`double` `UIConstraint::value`

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

- `src/ui/ui_constraint/ui_constraint.h`

3.17 UIConstraints Struct Reference

```
#include <ui_constraint.h>
```

3.17.1 Member Data Documentation

3.17.1.1 `height`

`UIConstraint` `UIConstraints::height`

3.17.1.2 `width`

`UIConstraint` `UIConstraints::width`

3.17.1.3 `x`

`UIConstraint` `UIConstraints::x`

3.17.1.4 `y`

`UIConstraint` `UIConstraints::y`

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

- `src/ui/ui_constraint/ui_constraint.h`

3.18 UIContainer Struct Reference

```
#include <ui_element.h>
```

3.18.1 Member Data Documentation

3.18.1.1 base

[UIElement](#) `UIContainer::base`

3.18.1.2 children

[Vector*](#) `UIContainer::children`

3.18.1.3 on_size_changed

[UIContainerSizeChanged](#) `UIContainer::on_size_changed`

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

- [src/ui/ui_element/ui_element.h](#)

3.19 UIData Struct Reference

```
#include <ui.h>
```

3.19.1 Member Data Documentation

3.19.1.1 backspace_pressed

`bool` `UIData::backspace_pressed`

3.19.1.2 expanded_splitbutton

[UISplitButton*](#) `UIData::expanded_splitbutton`

3.19.1.3 main_container

[UIContainer*](#) `UIData::main_container`

3.19.1.4 mouse_captured

`bool` `UIData::mouse_captured`

3.19.1.5 text_input

```
char UIData::text_input[SDL_TEXTINPUTEVENT_TEXT_SIZE]
```

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

- [src/ui/ui.h](#)

3.20 UIDropdownList Struct Reference

```
#include <ui_element.h>
```

3.20.1 Member Data Documentation

3.20.1.1 base

```
UIElement UIDropdownList::base
```

3.20.1.2 color

```
Color UIDropdownList::color
```

3.20.1.3 corner_radius

```
Uint32 UIDropdownList::corner_radius
```

3.20.1.4 expanded

```
bool UIDropdownList::expanded
```

3.20.1.5 items

```
Vector* UIDropdownList::items
```

3.20.1.6 on_selection_changed

```
UIDropdownListSelectionChanged UIDropdownList::on_selection_changed
```

3.20.1.7 selected_item

```
Uint32 UIDropdownList::selected_item
```

3.20.1.8 text_color

`Color UIDropdownList::text_color`

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

- `src/ui/ui_element/ui_element.h`

3.21 UIElement Struct Reference

```
#include <ui_element.h>
```

3.21.1 Member Data Documentation

3.21.1.1 constraints

`UIConstraints UIElement::constraints`

3.21.1.2 destroy

`UIElementDestroy UIElement::destroy`

3.21.1.3 parent

`UIElement* UIElement::parent`

3.21.1.4 position

`SDL_Point UIElement::position`

3.21.1.5 recalculate

`UIElementRecalculate UIElement::recalculate`

3.21.1.6 render

`UIElementRender UIElement::render`

3.21.1.7 shown

`bool UIElement::shown`

3.21.1.8 size

```
SDL_Point UElement::size
```

3.21.1.9 update

```
UElementUpdate UElement::update
```

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

- [src/ui/ui_element/ui_element.h](#)

3.22 UIButton Struct Reference

```
#include <ui_element.h>
```

3.22.1 Member Data Documentation

3.22.1.1 base

```
UElement UIButton::base
```

3.22.1.2 mouse_state

```
MouseState UIButton::mouse_state
```

3.22.1.3 on_click

```
UIButtonOnClick UIButton::on_click
```

3.22.1.4 texture

```
Texture* UIButton::texture
```

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

- [src/ui/ui_element/ui_element.h](#)

3.23 UILabel Struct Reference

```
#include <ui_element.h>
```

3.23.1 Member Data Documentation

3.23.1.1 base

`UIElement UILabel::base`

3.23.1.2 color

`Color UILabel::color`

3.23.1.3 text

`char UILabel::text[UITEXT_MAX_LENGTH+1]`

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

- `src/ui/ui_element/ui_element.h`

3.24 UIPanel Struct Reference

```
#include <ui_element.h>
```

3.24.1 Member Data Documentation

3.24.1.1 base

`UIElement UIPanel::base`

3.24.1.2 border_color

`Color UIPanel::border_color`

3.24.1.3 border_width

`UInt32 UIPanel::border_width`

3.24.1.4 color

`Color UIPanel::color`

3.24.1.5 corner_radius

```
UInt32 UIPanel::corner_radius
```

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

- [src/ui/ui_element/ui_element.h](#)

3.25 UISlider Struct Reference

```
#include <ui_element.h>
```

3.25.1 Member Data Documentation

3.25.1.1 base

```
UIElement UISlider::base
```

3.25.1.2 color

```
Color UISlider::color
```

3.25.1.3 corner_radius

```
UInt32 UISlider::corner_radius
```

3.25.1.4 mouse_state

```
MouseState UISlider::mouse_state
```

3.25.1.5 on_value_changed

```
UISliderValueChanged UISlider::on_value_changed
```

3.25.1.6 slider_color

```
Color UISlider::slider_color
```

3.25.1.7 thickness

```
UInt32 UISlider::thickness
```

3.25.1.8 value

```
double UISlider::value
```

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

- [src/ui/ui_element/ui_element.h](#)

3.26 UISplitButton Struct Reference

```
#include <ui_element.h>
```

3.26.1 Member Data Documentation

3.26.1.1 auto_dropdown

```
bool UISplitButton::auto_dropdown
```

3.26.1.2 base

```
UIElement UISplitButton::base
```

3.26.1.3 color

```
Color UISplitButton::color
```

3.26.1.4 corner_radius

```
UInt32 UISplitButton::corner_radius
```

3.26.1.5 expanded

```
bool UISplitButton::expanded
```

3.26.1.6 items

```
Vector* UISplitButton::items
```

3.26.1.7 on_item_clicked

```
UISplitButtonClicked UISplitButton::on_item_clicked
```

3.26.1.8 text_color

`Color UISplitButton::text_color`

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

- `src/ui/ui_element/ui_element.h`

3.27 UITextbox Struct Reference

```
#include <ui_element.h>
```

3.27.1 Member Data Documentation

3.27.1.1 base

`UIElement UITextbox::base`

3.27.1.2 color

`Color UITextbox::color`

3.27.1.3 corner_radius

`UInt32 UITextbox::corner_radius`

3.27.1.4 focused

`bool UITextbox::focused`

3.27.1.5 mouse_state

`MouseState UITextbox::mouse_state`

3.27.1.6 on_text_changed

`UITextboxTextChanged UITextbox::on_text_changed`

3.27.1.7 text

`char UITextbox::text[UITEXT_MAX_LENGTH+1]`

3.27.1.8 text_color

```
Color UITextbox::text_color
```

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

- [src/ui/ui_element/ui_element.h](#)

3.28 Vector Struct Reference

```
#include <vector.h>
```

3.28.1 Member Data Documentation

3.28.1.1 capacity

```
size_t Vector::capacity
```

3.28.1.2 data

```
void** Vector::data
```

3.28.1.3 size

```
size_t Vector::size
```

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

- [src/utls/vector/vector.h](#)

3.29 Vector2 Struct Reference

```
#include <vector2.h>
```

3.29.1 Member Data Documentation

3.29.1.1 x

```
double Vector2::x
```

3.29.1.2 y

```
double Vector2::y
```

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

- [src/geometry/vector2/vector2.h](#)

3.30 Window Struct Reference

```
#include <window.h>
```

3.30.1 Member Data Documentation

3.30.1.1 close_requested

```
bool Window::close_requested
```

3.30.1.2 input_data

```
InputData Window::input_data
```

3.30.1.3 renderer

```
SDL_Renderer* Window::renderer
```

3.30.1.4 ui_data

```
UIData Window::ui_data
```

3.30.1.5 window

```
SDL_Window* Window::window
```

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

- [src/window/window.h](#)

Chapter 4

File Documentation

4.1 src/app/app.c File Reference

Functions

- void `app_set_target_fps` (Uint32 fps)
- void `app_set_target` (Window *window)
- Window * `app_get_target` ()
- Vector * `app_get_windows` ()
- double `app_get_time` ()
- double `app_get_delta_time` ()
- void `_app_add_window` (Window *window)

4.1.1 Function Documentation

4.1.1.1 _app_add_window()

```
void _app_add_window (  
    Window * window )
```

Adds a window to the application (this is an internal function, should not be called directly)

Parameters

<i>window</i>	The window to add
---------------	-------------------

4.1.1.2 app_close()

```
void app_close ( )
```

Closes the application cleans up resources (SDL stuff)

4.1.1.3 app_get_delta_time()

```
double app_get_delta_time ( )
```

Returns the time since the last frame.

Returns

double The delta time

4.1.1.4 app_get_target()

```
Window * app_get_target ( )
```

Returns the target window.

Returns

Window* The target window

4.1.1.5 app_get_time()

```
double app_get_time ( )
```

Returns the time since the application started.

Returns

double The elapsed time

4.1.1.6 app_get_windows()

```
Vector * app_get_windows ( )
```

Returns the added windows.

Returns

Vector* The windows (should not be modified or freed)

4.1.1.7 app_init()

```
void app_init ( )
```

Initializes the application and SDL.

4.1.1.8 app_render()

```
void app_render ( )
```

Renders the windows and waits for the target frame time.

4.1.1.9 app_request_close()

```
void app_request_close ( )
```

This should be called to safely close the application.

4.1.1.10 app_set_target()

```
void app_set_target (
    Window * window )
```

Sets the target window for rendering, input handling and UI. A target should be set before doing any of these things.

Parameters

<i>window</i>	The target window
---------------	-------------------

4.1.1.11 app_set_target_fps()

```
void app_set_target_fps (
    Uint32 fps )
```

Sets the target fps.

Parameters

<i>fps</i>	The target fps
------------	----------------

4.1.1.12 app_update()

```
void app_update ( )
```

Updates the windows and handles events.

4.2 src/app/app.h File Reference

Classes

- struct [AppData](#)

Functions

- void `app_set_target_fps` (Uint32 fps)
- void `app_set_target` (Window *window)
- Window * `app_get_target` ()
- Vector * `app_get_windows` ()
- double `app_get_time` ()
- double `app_get_delta_time` ()
- void `_app_add_window` (Window *window)

4.2.1 Typedef Documentation

4.2.1.1 AppData

```
typedef struct AppData AppData
```

Contains the application data, like windows and target fps. There is only one instance of this struct, and should not be modified directly.

4.2.2 Function Documentation

4.2.2.1 _app_add_window()

```
void _app_add_window (
    Window * window )
```

Adds a window to the application (this is an internal function, should not be called directly)

Parameters

<i>window</i>	The window to add
---------------	-------------------

4.2.2.2 app_close()

```
void app_close ( )
```

Closes the application cleans up resources (SDL stuff)

4.2.2.3 app_get_delta_time()

```
double app_get_delta_time ( )
```

Returns the time since the last frame.

Returns

double The delta time

4.2.2.4 app_get_target()

```
Window * app_get_target ( )
```

Returns the target window.

Returns

Window* The target window

4.2.2.5 app_get_time()

```
double app_get_time ( )
```

Returns the time since the application started.

Returns

double The elapsed time

4.2.2.6 app_get_windows()

```
Vector * app_get_windows ( )
```

Returns the added windows.

Returns

Vector* The windows (should not be modified or freed)

4.2.2.7 app_init()

```
void app_init ( )
```

Initializes the application and SDL.

4.2.2.8 app_render()

```
void app_render ( )
```

Renders the windows and waits for the target frame time.

4.2.2.9 app_request_close()

```
void app_request_close ( )
```

This should be called to safely close the application.

4.2.2.10 app_set_target()

```
void app_set_target (
    Window * window )
```

Sets the target window for rendering, input handling and UI. A target should be set before doing any of these things.

Parameters

<i>window</i>	The target window
---------------	-------------------

4.2.2.11 app_set_target_fps()

```
void app_set_target_fps (
    Uint32 fps )
```

Sets the target fps.

Parameters

<i>fps</i>	The target fps
------------	----------------

4.2.2.12 app_update()

```
void app_update ( )
```

Updates the windows and handles events.

4.3 app.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #ifndef _WIN32
00004     #include <SDL.h>
00005 #elif defined(__unix__) || defined(__linux__)
00006     #include <SDL2/SDL.h>
00007 #endif
00008
00009 #include "../window/window.h"
00010 #include "../utils/vector/vector.h"
00011
00015 typedef struct AppData
00016 {
00017     Vector* windows;
00018     Uint32 target_frame_time;
00019     Uint32 last_frame_start;
00020     Uint32 frame_start;
00021     double delta_time;
00022 } AppData;
00023
00027 void app_init();
00031 void app_update();
00035 void app_render();
00039 void app_request_close();
00043 void app_close();
00049 void app_set_target_fps(Uint32 fps);
00050
00056 void app_set_target(Window* window);
00062 Window* app_get_target();
00068 Vector* app_get_windows();
00074 double app_get_time();
00080 double app_get_delta_time();
00081
00087 void _app_add_window(Window* window);
```

4.4 src/color/color.c File Reference

Functions

- [Color color_from_hex](#) (int hex)
- [Color color_from_rgb](#) (int r, int g, int b)
- [Color color_from_rgba](#) (int r, int g, int b, int a)
- [Color color_from_hsv](#) (double h, double s, double v)
- [Color color_from_grayscale](#) (int value)
- [Color color_fade](#) (Color color, double fade)
- [Color color_shift](#) (Color color, int shift)
- [Color color_clever_shift](#) (Color color, int shift)

4.4.1 Function Documentation

4.4.1.1 color_clever_shift()

```
Color color_clever_shift (
    Color color,
    int shift )
```

Shifts a color by a certain amount with a different algorithm (only used by ui)

Parameters

<i>color</i>	The color to shift
<i>shift</i>	The amount to shift by (-255 - 255)

Returns

Color The shifted color

4.4.1.2 color_fade()

```
Color color_fade (
    Color color,
    double fade )
```

Fades a color by a certain amount.

Parameters

<i>color</i>	The color to fade
<i>fade</i>	The amount to fade by (0.0 - 1.0)

Returns

Color The faded color

4.4.1.3 color_from_grayscale()

```
Color color_from_grayscale (
    int value )
```

Creates a color from a grayscale value.

Parameters

<i>value</i>	The grayscale value
--------------	---------------------

Returns

Color The color

4.4.1.4 color_from_hex()

```
Color color_from_hex (
    int hex )
```

Converts a hex color to a Color struct.

Parameters

<i>hex</i>	Hex color
------------	-----------

Returns

Color The color

4.4.1.5 color_from_hsv()

```
Color color_from_hsv (
    double h,
    double s,
    double v )
```

Creates a color from HSV values.

Parameters

<i>h</i>	The hue value
<i>s</i>	The saturation value
<i>v</i>	The value value

Returns

Color The color

4.4.1.6 color_from_rgb()

```
Color color_from_rgb (
    int r,
    int g,
    int b )
```

Creates a color from RGB values.

Parameters

<i>r</i>	The red value
<i>g</i>	The green value
<i>b</i>	The blue value

Returns

Color The color

4.4.1.7 color_from_rgba()

```
Color color_from_rgba (
    int r,
    int g,
    int b,
    int a )
```

Creates a color from RGBA values.

Parameters

<i>r</i>	The red value
<i>g</i>	The green value
<i>b</i>	The blue value
<i>a</i>	The alpha value

Returns

Color The color

4.4.1.8 color_shift()

```
Color color_shift (
    Color color,
    int shift )
```

Shifts a color by a certain amount.

Parameters

<i>color</i>	The color to shift
<i>shift</i>	The amount to shift by (-255 - 255)

Returns

Color The shifted color

4.5 src/color/color.h File Reference

Functions

- [Color color_from_hex](#) (int hex)
- [Color color_from_rgb](#) (int r, int g, int b)
- [Color color_from_rgba](#) (int r, int g, int b, int a)
- [Color color_from_hsv](#) (double h, double s, double v)
- [Color color_from_grayscale](#) (int value)
- [Color color_fade](#) (Color color, double fade)
- [Color color_shift](#) (Color color, int shift)
- [Color color_clever_shift](#) (Color color, int shift)

4.5.1 Macro Definition Documentation

4.5.1.1 BLACK

```
#define BLACK (Color) { 0, 0, 0, 255 }
```

4.5.1.2 BLUE

```
#define BLUE (Color) { 0, 0, 255, 255 }
```

4.5.1.3 CYAN

```
#define CYAN (Color) { 0, 255, 255, 255 }
```

4.5.1.4 DARK_GRAY

```
#define DARK_GRAY (Color) { 40, 40, 40, 255 }
```

4.5.1.5 GRAY

```
#define GRAY (Color) { 128, 128, 128, 255 }
```

4.5.1.6 GREEN

```
#define GREEN (Color) { 0, 255, 0, 255 }
```

4.5.1.7 MAGENTA

```
#define MAGENTA (Color) { 255, 0, 255, 255 }
```

4.5.1.8 RED

```
#define RED (Color) { 255, 0, 0, 255 }
```

4.5.1.9 TRANSPARENT

```
#define TRANSPARENT (Color) { 0, 0, 0, 0 }
```

4.5.1.10 WHITE

```
#define WHITE (Color) { 255, 255, 255, 255 }
```

4.5.1.11 YELLOW

```
#define YELLOW (Color) { 255, 255, 0, 255 }
```

4.5.2 Typedef Documentation

4.5.2.1 Color

```
typedef SDL_Color Color
```

4.5.3 Function Documentation

4.5.3.1 color_clever_shift()

```
Color color_clever_shift (  
    Color color,  
    int shift )
```

Shifts a color by a certain amount with a different algorithm (only used by ui)

Parameters

<i>color</i>	The color to shift
<i>shift</i>	The amount to shift by (-255 - 255)

Returns

Color The shifted color

4.5.3.2 color_fade()

```
Color color_fade (
    Color color,
    double fade )
```

Fades a color by a certain amount.

Parameters

<i>color</i>	The color to fade
<i>fade</i>	The amount to fade by (0.0 - 1.0)

Returns

Color The faded color

4.5.3.3 color_from_grayscale()

```
Color color_from_grayscale (
    int value )
```

Creates a color from a grayscale value.

Parameters

<i>value</i>	The grayscale value
--------------	---------------------

Returns

Color The color

4.5.3.4 color_from_hex()

```
Color color_from_hex (
    int hex )
```

Converts a hex color to a Color struct.

Parameters

<i>hex</i>	Hex color
------------	-----------

Returns

Color The color

4.5.3.5 color_from_hsv()

```
Color color_from_hsv (
    double h,
    double s,
    double v )
```

Creates a color from HSV values.

Parameters

<i>h</i>	The hue value
<i>s</i>	The saturation value
<i>v</i>	The value value

Returns

Color The color

4.5.3.6 color_from_rgb()

```
Color color_from_rgb (
    int r,
    int g,
    int b )
```

Creates a color from RGB values.

Parameters

<i>r</i>	The red value
<i>g</i>	The green value
<i>b</i>	The blue value

Returns

Color The color

4.5.3.7 color_from_rgba()

```
Color color_from_rgba (
    int r,
    int g,
    int b,
    int a )
```

Creates a color from RGBA values.

Parameters

<i>r</i>	The red value
<i>g</i>	The green value
<i>b</i>	The blue value
<i>a</i>	The alpha value

Returns

Color The color

4.5.3.8 color_shift()

```
Color color_shift (
    Color color,
    int shift )
```

Shifts a color by a certain amount.

Parameters

<i>color</i>	The color to shift
<i>shift</i>	The amount to shift by (-255 - 255)

Returns

Color The shifted color

4.6 color.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #ifndef _WIN32
00004     #include <SDL.h>
00005 #elif defined(__unix__) || defined(__linux__)
00006     #include <SDL2/SDL.h>
00007 #endif
00008
00009 typedef SDL_Color Color;
00010
00011 #define WHITE (Color) { 255, 255, 255, 255 }
00012 #define BLACK (Color) { 0, 0, 0, 255 }
00013 #define GRAY (Color) { 128, 128, 128, 255 }
00014 #define DARK_GRAY (Color) { 40, 40, 40, 255 }
00015 #define RED (Color) { 255, 0, 0, 255 }
00016 #define GREEN (Color) { 0, 255, 0, 255 }
00017 #define BLUE (Color) { 0, 0, 255, 255 }
00018 #define YELLOW (Color) { 255, 255, 0, 255 }
00019 #define MAGENTA (Color) { 255, 0, 255, 255 }
00020 #define CYAN (Color) { 0, 255, 255, 255 }
00021 #define TRANSPARENT (Color) { 0, 0, 0, 0 }
00022
00029 Color color_from_hex(int hex);
00038 Color color_from_rgb(int r, int g, int b);
00048 Color color_from_rgba(int r, int g, int b, int a);
00057 Color color_from_hsv(double h, double s, double v);
00064 Color color_from_grayscale(int value);
00072 Color color_fade(Color color, double fade);
00080 Color color_shift(Color color, int shift);
00088 Color color_clever_shift(Color color, int shift);
```

4.7 src/font/font.c File Reference

Functions

- [Font](#) * [font_load](#) (const char *path, int size)

4.7.1 Function Documentation

4.7.1.1 _font_close()

```
void _font_close ( )
```

Destroys the fonts and the font vector (should not be called directly)

4.7.1.2 _font_init()

```
void _font_init ( )
```

Creates the font vector that contains all the loaded fonts (should not be called directly, it is needed for the `_font_close` function)

4.7.1.3 font_load()

```
Font * font_load (
    const char * path,
    int size )
```

Loads a font from a file (freed automatically when the program closes)

Parameters

<i>path</i>	The path to the font file
<i>size</i>	The size of the font

Returns

Font* The font

4.8 src/font/font.h File Reference

Classes

- struct [Font](#)

Functions

- [Font](#) * [font_load](#) (const char *path, int size)

4.8.1 Typedef Documentation

4.8.1.1 Font

```
typedef struct Font Font
```

Holds a TTF_Font and its size.

4.8.2 Function Documentation

4.8.2.1 _font_close()

```
void _font_close ( )
```

Destroys the fonts and the font vector (should not be called directly)

4.8.2.2 _font_init()

```
void _font_init ( )
```

Creates the font vector that contains all the loaded fonts (should not be called directly, it is needed for the _font_close function)

4.8.2.3 font_load()

```
Font * font_load (
    const char * path,
    int size )
```

Loads a font from a file (freed automatically when the program closes)

Parameters

<i>path</i>	The path to the font file
<i>size</i>	The size of the font

Returns

Font* The font

4.9 font.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #ifdef _WIN32
00004     #include <SDL_ttf.h>
```



```

00005 #elif defined(__unix__) || defined(__linux__)
00006     #include <SDL2/SDL_ttf.h>
00007 #endif
00008
00012 typedef struct Font
00013 {
00014     TTF_Font* font;
00015     int size;
00016 } Font;
00017
00025 Font* font_load(const char* path, int size);
00026
00030 void _font_init();
00034 void _font_close();

```

4.10 src/geometry/coordinate_system/coordinate_system.c File Reference

Functions

- [CoordinateSystem * coordinate_system_create](#) ([Vector2](#) position, [Vector2](#) size, [Vector2](#) origin)
- void [coordinate_system_clear](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_destroy](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_save](#) ([CoordinateSystem](#) *cs, const char *path)
- [CoordinateSystem](#) * [coordinate_system_load](#) (const char *path)
- [Vector2](#) [screen_to_coordinates](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- [Vector2](#) [coordinates_to_screen](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- bool [coordinate_system_is_hovered](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- void [coordinate_system_select_shape](#) ([CoordinateSystem](#) *cs, [Shape](#) *shape)
- void [coordinate_system_deselect_shape](#) ([CoordinateSystem](#) *cs, [Shape](#) *shape)
- void [coordinate_system_select_all_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_drag_selected_shapes](#) ([CoordinateSystem](#) *cs, bool drag)
- [Shape](#) * [coordinate_system_get_hovered_shape](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- [Vector](#) * [coordinate_system_get_selected_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_deselect_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_delete_selected_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_translate](#) ([CoordinateSystem](#) *cs, [Vector2](#) translation)
- void [coordinate_system_zoom](#) ([CoordinateSystem](#) *cs, double zoom)
- void [coordinate_system_update](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_draw](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_update_dimensions](#) ([CoordinateSystem](#) *cs, [Vector2](#) position, [Vector2](#) size)
- void [coordinate_system_destroy_shape](#) ([CoordinateSystem](#) *cs, [Shape](#) *shape)

4.10.1 Function Documentation

4.10.1.1 coordinate_system_clear()

```

void coordinate_system_clear (
    CoordinateSystem * cs )

```

Clears a coordinate system (removes all the shapes)

Parameters

<i>CS</i>	
-----------	--

4.10.1.2 coordinate_system_create()

```
CoordinateSystem * coordinate_system_create (
    Vector2 position,
    Vector2 size,
    Vector2 origin )
```

Creates a coordinate system.

Parameters

<i>position</i>	The position of the coordinate system in the screen
<i>size</i>	The size of the coordinate system (in pixels)
<i>origin</i>	The origin of the coordinate system (relative to the coordinate system (normalized))

Returns

CoordinateSystem* The created coordinate system

4.10.1.3 coordinate_system_delete_selected_shapes()

```
void coordinate_system_delete_selected_shapes (
    CoordinateSystem * cs )
```

Deletes all the selected shapes.

Parameters

<i>cs</i>	The coordinate system to delete the shapes in
-----------	---

4.10.1.4 coordinate_system_deselect_shape()

```
void coordinate_system_deselect_shape (
    CoordinateSystem * cs,
    Shape * shape )
```

Deselects a shape.

Parameters

<i>cs</i>	The coordinate system to deselect the shape in
<i>shape</i>	The shape to deselect

4.10.1.5 coordinate_system_deselect_shapes()

```
void coordinate_system_deselect_shapes (
    CoordinateSystem * cs )
```

Deselects all the selected shapes.

Parameters

<i>cs</i>	The coordinate system to deselect the shapes in
-----------	---

4.10.1.6 coordinate_system_destroy()

```
void coordinate_system_destroy (  
    CoordinateSystem * cs )
```

Destroys a coordinate system.

Parameters

<i>cs</i>	The coordinate system to destroy
-----------	----------------------------------

4.10.1.7 coordinate_system_destroy_shape()

```
void coordinate_system_destroy_shape (  
    CoordinateSystem * cs,  
    Shape * shape )
```

Destroys a shape and removes it from the coordinate system (as well as the shapes it defined)

Parameters

<i>cs</i>	The coordinate system to remove the shape from
<i>shape</i>	The shape to remove

4.10.1.8 coordinate_system_drag_selected_shapes()

```
void coordinate_system_drag_selected_shapes (  
    CoordinateSystem * cs,  
    bool drag )
```

Sets the dragged shape.

Parameters

<i>cs</i>	The coordinate system to set the dragged shape in
<i>shape</i>	The shape to set as dragged

4.10.1.9 coordinate_system_draw()

```
void coordinate_system_draw (  

```

```
CoordinateSystem * cs )
```

Draws the coordinate system.

Parameters

<i>cs</i>	The coordinate system to draw
-----------	-------------------------------

4.10.1.10 coordinate_system_get_hovered_shape()

```
Shape * coordinate_system_get_hovered_shape (
    CoordinateSystem * cs,
    Vector2 point )
```

Returns the shape hovered by the point.

Parameters

<i>cs</i>	The coordinate system to check
<i>point</i>	The point to check

Returns

Shape* The hovered shape (NULL if none)

4.10.1.11 coordinate_system_get_selected_shapes()

```
Vector * coordinate_system_get_selected_shapes (
    CoordinateSystem * cs )
```

Returns the selected shapes.

Parameters

<i>cs</i>	The coordinate system to retrieve to selected shapes from
-----------	---

Returns

Vector* A vector of the selected shapes

4.10.1.12 coordinate_system_is_hovered()

```
bool coordinate_system_is_hovered (
    CoordinateSystem * cs,
    Vector2 point )
```

Returns whether the coordinate system is hovered by the point.

Parameters

<i>cs</i>	The coordinate system to check
<i>point</i>	The point to check

4.10.1.13 coordinate_system_load()

```
CoordinateSystem * coordinate_system_load (
    const char * path )
```

Loads a coordinate system from a file (loads the shapes from a .gae file)

Parameters

<i>path</i>	The path to load the coordinate system from
-------------	---

Returns

CoordinateSystem* The loaded coordinate system

4.10.1.14 coordinate_system_save()

```
void coordinate_system_save (
    CoordinateSystem * cs,
    const char * path )
```

Saves a coordinate system to a file (saves the shapes into a .gae file)

Parameters

<i>cs</i>	The coordinate system to save
<i>path</i>	The path to save the coordinate system to

4.10.1.15 coordinate_system_select_all_shapes()

```
void coordinate_system_select_all_shapes (
    CoordinateSystem * cs )
```

Selects all the shapes.

Parameters

<i>cs</i>	The coordinate system to select the shapes in
-----------	---

4.10.1.16 coordinate_system_select_shape()

```
void coordinate_system_select_shape (
    CoordinateSystem * cs,
    Shape * shape )
```

Selects a shape.

Parameters

<i>cs</i>	The coordinate system to select the shape in
<i>shape</i>	The shape to select

4.10.1.17 coordinate_system_translate()

```
void coordinate_system_translate (
    CoordinateSystem * cs,
    Vector2 translation )
```

Translates the coordinate system.

Parameters

<i>cs</i>	The coordinate system to translate
<i>translation</i>	The translation vector (in pixels)

4.10.1.18 coordinate_system_update()

```
void coordinate_system_update (
    CoordinateSystem * cs )
```

Updates the coordinate system and calculates the intersections.

Parameters

<i>cs</i>	The coordinate system to update
-----------	---------------------------------

4.10.1.19 coordinate_system_update_dimensions()

```
void coordinate_system_update_dimensions (
    CoordinateSystem * cs,
    Vector2 position,
    Vector2 size )
```

Updates the dimensions of the coordinate system.

Parameters

<i>cs</i>	The coordinate system to update
<i>position</i>	The new position
<i>size</i>	The new size

4.10.1.20 coordinate_system_zoom()

```
void coordinate_system_zoom (  
    CoordinateSystem * cs,  
    double zoom )
```

Zooms into the coordinate system.

Parameters

<i>cs</i>	The coordinate system to zoom into
<i>zoom</i>	The zoom factor

4.10.1.21 coordinates_to_screen()

```
Vector2 coordinates_to_screen (  
    CoordinateSystem * cs,  
    Vector2 point )
```

Translates a point from the coordinate system to the screen.

Parameters

<i>cs</i>	The coordinate system to translate the point from
<i>point</i>	The point to translate

Returns

[Vector2](#) The translated point

4.10.1.22 screen_to_coordinates()

```
Vector2 screen_to_coordinates (  
    CoordinateSystem * cs,  
    Vector2 point )
```

Translates a point from the screen to the coordinate system.

Parameters

<i>cs</i>	The coordinate system to translate the point to
<i>point</i>	The point to translate

Returns

[Vector2](#) The translated point

4.11 src/geometry/coordinate_system/coordinate_system.h File Reference

Classes

- struct [CoordinateSystem](#)

Functions

- [CoordinateSystem](#) * [coordinate_system_create](#) ([Vector2](#) position, [Vector2](#) size, [Vector2](#) origin)
- void [coordinate_system_clear](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_destroy](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_save](#) ([CoordinateSystem](#) *cs, const char *path)
- [CoordinateSystem](#) * [coordinate_system_load](#) (const char *path)
- [Vector2](#) [screen_to_coordinates](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- [Vector2](#) [coordinates_to_screen](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- bool [coordinate_system_is_hovered](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- void [coordinate_system_select_shape](#) ([CoordinateSystem](#) *cs, [Shape](#) *shape)
- void [coordinate_system_deselect_shape](#) ([CoordinateSystem](#) *cs, [Shape](#) *shape)
- void [coordinate_system_select_all_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_drag_selected_shapes](#) ([CoordinateSystem](#) *cs, bool drag)
- [Shape](#) * [coordinate_system_get_hovered_shape](#) ([CoordinateSystem](#) *cs, [Vector2](#) point)
- [Vector](#) * [coordinate_system_get_selected_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_deselect_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_delete_selected_shapes](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_translate](#) ([CoordinateSystem](#) *cs, [Vector2](#) translation)
- void [coordinate_system_zoom](#) ([CoordinateSystem](#) *cs, double zoom)
- void [coordinate_system_update](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_draw](#) ([CoordinateSystem](#) *cs)
- void [coordinate_system_update_dimensions](#) ([CoordinateSystem](#) *cs, [Vector2](#) position, [Vector2](#) size)
- void [coordinate_system_destroy_shape](#) ([CoordinateSystem](#) *cs, [Shape](#) *shape)

4.11.1 Macro Definition Documentation

4.11.1.1 INITIAL_ZOOM

```
#define INITIAL_ZOOM 20
```

4.11.2 Typedef Documentation

4.11.2.1 CoordinateSystem

```
typedef struct CoordinateSystem CoordinateSystem
```

4.11.3 Function Documentation

4.11.3.1 coordinate_system_clear()

```
void coordinate\_system\_clear (
    CoordinateSystem * cs )
```

Clears a coordinate system (removes all the shapes)

Parameters

<i>cs</i>	
-----------	--

4.11.3.2 coordinate_system_create()

```
CoordinateSystem * coordinate_system_create (
    Vector2 position,
    Vector2 size,
    Vector2 origin )
```

Creates a coordinate system.

Parameters

<i>position</i>	The position of the coordinate system in the screen
<i>size</i>	The size of the coordinate system (in pixels)
<i>origin</i>	The origin of the coordinate system (relative to the coordinate system (normalized))

Returns

CoordinateSystem* The created coordinate system

4.11.3.3 coordinate_system_delete_selected_shapes()

```
void coordinate_system_delete_selected_shapes (
    CoordinateSystem * cs )
```

Deletes all the selected shapes.

Parameters

<i>cs</i>	The coordinate system to delete the shapes in
-----------	---

4.11.3.4 coordinate_system_deselect_shape()

```
void coordinate_system_deselect_shape (
    CoordinateSystem * cs,
    Shape * shape )
```

Deselects a shape.

Parameters

<i>cs</i>	The coordinate system to deselect the shape in
<i>shape</i>	The shape to deselect

4.11.3.5 coordinate_system_deselect_shapes()

```
void coordinate_system_deselect_shapes (
    CoordinateSystem * cs )
```

Deselects all the selected shapes.

Parameters

<i>cs</i>	The coordinate system to deselect the shapes in
-----------	---

4.11.3.6 coordinate_system_destroy()

```
void coordinate_system_destroy (
    CoordinateSystem * cs )
```

Destroys a coordinate system.

Parameters

<i>cs</i>	The coordinate system to destroy
-----------	----------------------------------

4.11.3.7 coordinate_system_destroy_shape()

```
void coordinate_system_destroy_shape (
    CoordinateSystem * cs,
    Shape * shape )
```

Destroys a shape and removes it from the coordinate system (as well as the shapes it defined)

Parameters

<i>cs</i>	The coordinate system to remove the shape from
<i>shape</i>	The shape to remove

4.11.3.8 coordinate_system_drag_selected_shapes()

```
void coordinate_system_drag_selected_shapes (
    CoordinateSystem * cs,
    bool drag )
```

Sets the dragged shape.

Parameters

<i>cs</i>	The coordinate system to set the dragged shape in
<i>shape</i>	The shape to set as dragged

4.11.3.9 coordinate_system_draw()

```
void coordinate_system_draw (
    CoordinateSystem * cs )
```

Draws the coordinate system.

Parameters

<i>cs</i>	The coordinate system to draw
-----------	-------------------------------

4.11.3.10 coordinate_system_get_hovered_shape()

```
Shape * coordinate_system_get_hovered_shape (
    CoordinateSystem * cs,
    Vector2 point )
```

Returns the shape hovered by the point.

Parameters

<i>cs</i>	The coordinate system to check
<i>point</i>	The point to check

Returns

Shape* The hovered shape (NULL if none)

4.11.3.11 coordinate_system_get_selected_shapes()

```
Vector * coordinate_system_get_selected_shapes (
    CoordinateSystem * cs )
```

Returns the selected shapes.

Parameters

<i>cs</i>	The coordinate system to retrieve to selected shapes from
-----------	---

Returns

Vector* A vector of the selected shapes

4.11.3.12 coordinate_system_is_hovered()

```
bool coordinate_system_is_hovered (
    CoordinateSystem * cs,
    Vector2 point )
```

Returns whether the coordinate system is hovered by the point.

Parameters

<i>cs</i>	The coordinate system to check
<i>point</i>	The point to check

4.11.3.13 coordinate_system_load()

```
CoordinateSystem * coordinate_system_load (
    const char * path )
```

Loads a coordinate system from a file (loads the shapes from a .gae file)

Parameters

<i>path</i>	The path to load the coordinate system from
-------------	---

Returns

CoordinateSystem* The loaded coordinate system

4.11.3.14 coordinate_system_save()

```
void coordinate_system_save (
    CoordinateSystem * cs,
    const char * path )
```

Saves a coordinate system to a file (saves the shapes into a .gae file)

Parameters

<i>cs</i>	The coordinate system to save
<i>path</i>	The path to save the coordinate system to

4.11.3.15 coordinate_system_select_all_shapes()

```
void coordinate_system_select_all_shapes (
    CoordinateSystem * cs )
```

Selects all the shapes.

Parameters

<i>cs</i>	The coordinate system to select the shapes in
-----------	---

4.11.3.16 coordinate_system_select_shape()

```
void coordinate_system_select_shape (
    CoordinateSystem * cs,
    Shape * shape )
```

Selects a shape.

Parameters

<i>cs</i>	The coordinate system to select the shape in
<i>shape</i>	The shape to select

4.11.3.17 coordinate_system_translate()

```
void coordinate_system_translate (
    CoordinateSystem * cs,
    Vector2 translation )
```

Translates the coordinate system.

Parameters

<i>cs</i>	The coordinate system to translate
<i>translation</i>	The translation vector (in pixels)

4.11.3.18 coordinate_system_update()

```
void coordinate_system_update (
    CoordinateSystem * cs )
```

Updates the coordinate system and calculates the intersections.

Parameters

<i>cs</i>	The coordinate system to update
-----------	---------------------------------

4.11.3.19 coordinate_system_update_dimensions()

```
void coordinate_system_update_dimensions (
    CoordinateSystem * cs,
    Vector2 position,
    Vector2 size )
```

Updates the dimensions of the coordinate system.

Parameters

<i>cs</i>	The coordinate system to update
<i>position</i>	The new position
<i>size</i>	The new size

4.11.3.20 coordinate_system_zoom()

```
void coordinate_system_zoom (
    CoordinateSystem * cs,
    double zoom )
```

Zooms into the coordinate system.

Parameters

<i>cs</i>	The coordinate system to zoom into
<i>zoom</i>	The zoom factor

4.11.3.21 coordinates_to_screen()

```
Vector2 coordinates_to_screen (
    CoordinateSystem * cs,
    Vector2 point )
```

Translates a point from the coordinate system to the screen.

Parameters

<i>cs</i>	The coordinate system to translate the point from
<i>point</i>	The point to translate

Returns

[Vector2](#) The translated point

4.11.3.22 screen_to_coordinates()

```
Vector2 screen_to_coordinates (
    CoordinateSystem * cs,
    Vector2 point )
```

Translates a point from the screen to the coordinate system.

Parameters

<i>cs</i>	The coordinate system to translate the point to
<i>point</i>	The point to translate

Returns

[Vector2](#) The translated point

4.12 coordinate_system.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "../shape/shape.h"
00004 #include "../vector2/vector2.h"
00005 #include "../../texture/texture.h"
00006 #include "../../utils/vector/vector.h"
00007
00008 #define INITIAL_ZOOM 20
00009
00010 typedef struct CoordinateSystem
00011 {
00012     Vector2 position;
00013     Vector2 size;
00014     Vector2 origin;
00015     double zoom;
00016
00017     Vector* shapes;
00018     Vector* intersection_points;
00019 } CoordinateSystem;
00020
00029 CoordinateSystem* coordinate_system_create(Vector2 position, Vector2 size, Vector2 origin);
00035 void coordinate_system_clear(CoordinateSystem* cs);
00041 void coordinate_system_destroy(CoordinateSystem* cs);
00042
00049 void coordinate_system_save(CoordinateSystem* cs, const char* path);
00056 CoordinateSystem* coordinate_system_load(const char* path);
00057
00065 Vector2 screen_to_coordinates(CoordinateSystem* cs, Vector2 point);
00073 Vector2 coordinates_to_screen(CoordinateSystem* cs, Vector2 point);
00074
00081 bool coordinate_system_is_hovered(CoordinateSystem* cs, Vector2 point);
00088 void coordinate_system_select_shape(CoordinateSystem* cs, Shape* shape);
00095 void coordinate_system_deselect_shape(CoordinateSystem* cs, Shape* shape);
00101 void coordinate_system_select_all_shapes(CoordinateSystem* cs);
00108 void coordinate_system_drag_selected_shapes(CoordinateSystem* cs, bool drag);
00116 Shape* coordinate_system_get_hovered_shape(CoordinateSystem* cs, Vector2 point);
00123 Vector* coordinate_system_get_selected_shapes(CoordinateSystem* cs);
00129 void coordinate_system_deselect_shapes(CoordinateSystem* cs);
00135 void coordinate_system_delete_selected_shapes(CoordinateSystem* cs);
00142 void coordinate_system_translate(CoordinateSystem* cs, Vector2 translation);
00149 void coordinate_system_zoom(CoordinateSystem* cs, double zoom);
00155 void coordinate_system_update(CoordinateSystem* cs);
00161 void coordinate_system_draw(CoordinateSystem* cs);
00169 void coordinate_system_update_dimensions(CoordinateSystem* cs, Vector2 position, Vector2 size);
00176 void coordinate_system_destroy_shape(CoordinateSystem* cs, Shape* shape);
```

4.13 src/geometry/intersection/intersection.c File Reference

Functions

- [Vector](#) * [intersection_get](#) (Shape *shape1, Shape *shape2)

4.13.1 Macro Definition Documentation

4.13.1.1 EPSILON

```
#define EPSILON 0.0001
```

4.13.2 Function Documentation

4.13.2.1 intersection_get()

```
Vector * intersection_get (
    Shape * shape1,
    Shape * shape2 )
```

Returns the intersection(s) of two shapes.

Parameters

<i>shape1</i>	The first shape
<i>shape2</i>	The second shape

Returns

Vector* The intersection vector, containing the intersection point(s) (should be freed!)

4.14 src/geometry/intersection/intersection.h File Reference

Functions

- [Vector * intersection_get](#) ([Shape *shape1](#), [Shape *shape2](#))

4.14.1 Function Documentation

4.14.1.1 intersection_get()

```
Vector * intersection_get (
    Shape * shape1,
    Shape * shape2 )
```

Returns the intersection(s) of two shapes.

Parameters

<i>shape1</i>	The first shape
<i>shape2</i>	The second shape

Returns

Vector* The intersection vector, containing the intersection point(s) (should be freed!)

4.15 intersection.h

[Go to the documentation of this file.](#)


```

00001 #pragma once
00002
00003 #include "../shape/shape.h"
00004 #include "../../utils/vector/vector.h"
00005
00013 Vector* intersection_get(Shape* shape1, Shape* shape2);

```

4.16 src/geometry/shape/shape.c File Reference

Functions

- [Point](#) * [point_create](#) ([CoordinateSystem](#) *cs, [Vector2](#) coordinates)
- [Line](#) * [line_create](#) ([CoordinateSystem](#) *cs, [Point](#) *p1, [Point](#) *p2)
- [Circle](#) * [circle_create](#) ([CoordinateSystem](#) *cs, [Point](#) *center, [Point](#) *perimeter_point)
- [Parallel](#) * [parallel_create](#) ([CoordinateSystem](#) *cs, [Line](#) *line, [Point](#) *point)
- [Perpendicular](#) * [perpendicular_create](#) ([CoordinateSystem](#) *cs, [Line](#) *line, [Point](#) *point)
- [AngleBisector](#) * [angle_bisector_create](#) ([CoordinateSystem](#) *cs, [Line](#) *line1, [Line](#) *line2)
- [Tangent](#) * [tangent_create](#) ([CoordinateSystem](#) *cs, [Circle](#) *circle, [Point](#) *point)
- void [shape_draw](#) ([CoordinateSystem](#) *cs, [Shape](#) *self)
- void [shape_update](#) ([CoordinateSystem](#) *cs, [Shape](#) *self)
- void [shape_translate](#) ([CoordinateSystem](#) *cs, [Shape](#) *self, [Vector2](#) translation)
- void [shape_destroy](#) ([CoordinateSystem](#) *cs, [Shape](#) *self)
- bool [shape_overlap_point](#) ([CoordinateSystem](#) *cs, [Shape](#) *self, [Vector2](#) point)
- bool [shape_is_defined_by](#) ([Shape](#) *self, [Shape](#) *shape)

4.16.1 Macro Definition Documentation

4.16.1.1 EPSILON

```
#define EPSILON 0.0001
```

4.16.2 Function Documentation

4.16.2.1 angle_bisector_create()

```

AngleBisector * angle_bisector_create (
    CoordinateSystem * cs,
    Line * line1,
    Line * line2 )

```

Creates an angle bisector in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the angle bisector in
<i>line1</i>	The first line to create the angle bisector to
<i>line2</i>	The second line to create the angle bisector to

Returns

AngleBisector* The created angle bisector

4.16.2.2 circle_create()

```
Circle * circle_create (
    CoordinateSystem * cs,
    Point * center,
    Point * perimeter_point )
```

Creates a circle in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the circle in
<i>center</i>	The center of the circle
<i>perimeter_point</i>	A point on the perimeter of the circle (has to be different from center)

Returns

Circle* The created circle

4.16.2.3 line_create()

```
Line * line_create (
    CoordinateSystem * cs,
    Point * p1,
    Point * p2 )
```

Creates a line in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the line in
<i>p1</i>	A point of the line
<i>p2</i>	Another point of the line (has to be different from p1)

Returns

Line* The created line

4.16.2.4 parallel_create()

```
Parallel * parallel_create (
    CoordinateSystem * cs,
    Line * line,
    Point * point )
```

Creates a parallel line in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the parallel line in
<i>line</i>	The line to create the parallel line to
<i>point</i>	The point the parallel line goes through

Returns

Parallel* The created parallel line

4.16.2.5 perpendicular_create()

```
Perpendicular * perpendicular_create (
    CoordinateSystem * cs,
    Line * line,
    Point * point )
```

Creates a perpendicular line in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the perpendicular line in
<i>line</i>	The line to create the perpendicular line to
<i>point</i>	The point the perpendicular line goes through

Returns

Perpendicular* The created perpendicular line

4.16.2.6 point_create()

```
Point * point_create (
    CoordinateSystem * cs,
    Vector2 coordinates )
```

Creates a point in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the point in
<i>coordinates</i>	The coordinates of the point

Returns

Point* The created point

4.16.2.7 shape_destroy()

```
void shape_destroy (
    CoordinateSystem * cs,
    Shape * self )
```

Destroys a shape, but does not remove it from the coordinate systems shapes! (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to destroy the shape in
<i>self</i>	The shape to destroy

4.16.2.8 shape_draw()

```
void shape_draw (
    CoordinateSystem * cs,
    Shape * self )
```

Draws a shape (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to draw the point in
<i>self</i>	The shape to draw

4.16.2.9 shape_is_defined_by()

```
bool shape_is_defined_by (
    Shape * self,
    Shape * shape )
```

Checks if a shape is defined by another shape (can be called on any shape)

Parameters

<i>self</i>	The shape to check if it is defined by the other shape
<i>shape</i>	The shape to check if it defines the other shape

Returns

true If the shape is defined by the other shape
false If the shape is not defined by the other shape

4.16.2.10 shape_overlap_point()

```
bool shape_overlap_point (
    CoordinateSystem * cs,
```

```
Shape * self,
Vector2 point )
```

Checks if a point overlaps the shape (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to check the overlap in
<i>self</i>	The shape to check the overlap with
<i>point</i>	The point to check the overlap with

Returns

true If the point overlaps with the shape

false If the point does not overlap with the shape

4.16.2.11 shape_translate()

```
void shape_translate (
    CoordinateSystem * cs,
    Shape * self,
    Vector2 translation )
```

Translates a shape if it is allowed (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to translate the shape in
<i>self</i>	The shape to translate
<i>translation</i>	The translation vector

4.16.2.12 shape_update()

```
void shape_update (
    CoordinateSystem * cs,
    Shape * self )
```

Updates a shape (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to update the shape in
<i>self</i>	The shape to update

4.16.2.13 tangent_create()

```
Tangent * tangent_create (
    CoordinateSystem * cs,
```

```
Circle * circle,
Point * point )
```

Creates a tangent to a circle in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the tangent in
<i>circle</i>	The circle to create the tangent to
<i>point</i>	The point the tangent goes through

Returns

Tangent* The created tangent

4.16.3 Variable Documentation

4.16.3.1 shape_destroy_funcs

```
ShapeDestroy shape_destroy_funcs[ST_COUNT] = {_point_destroy, _line_destroy, _circle_destroy,
_parallel_destroy, _perpendicular_destroy, _angle_bisector_destroy, _tangent_destroy}
```

4.16.3.2 shape_draw_funcs

```
ShapeDraw shape_draw_funcs[ST_COUNT] = {_point_draw, _line_draw, _circle_draw, _parallel_draw,
_perpendicular_draw, _angle_bisector_draw, _tangent_draw}
```

4.16.3.3 shape_is_defined_by_funcs

```
ShapeIsDefinedBy shape_is_defined_by_funcs[ST_COUNT] = {_point_is_defined_by, _line_is_↵
defined_by, _circle_is_defined_by, _parallel_is_defined_by, _perpendicular_is_defined_by, _↵
angle_bisector_is_defined_by, _tangent_is_defined_by}
```

4.16.3.4 shape_overlap_point_funcs

```
ShapeOverlapPoint shape_overlap_point_funcs[ST_COUNT] = {_point_overlap, _line_overlap, _↵
circle_overlap, _parallel_overlap, _perpendicular_overlap, _angle_bisector_overlap, _tangent↵
_overlap}
```

4.16.3.5 shape_translate_funcs

```
ShapeTranslate shape_translate_funcs[ST_COUNT] = {_point_translate, _line_translate, _circle↵
_translate, _parallel_translate, _perpendicular_translate, _angle_bisector_translate, _↵
tangent_translate}
```

4.17 src/geometry/shape/shape.h File Reference

Classes

- struct [Shape](#)
- struct [Point](#)
- struct [Line](#)
- struct [Circle](#)
- struct [Parallel](#)
- struct [Perpendicular](#)
- struct [AngleBisector](#)
- struct [Tangent](#)

Functions

- [Point](#) * [point_create](#) ([CoordinateSystem](#) *cs, [Vector2](#) coordinates)
- [Line](#) * [line_create](#) ([CoordinateSystem](#) *cs, [Point](#) *p1, [Point](#) *p2)
- [Circle](#) * [circle_create](#) ([CoordinateSystem](#) *cs, [Point](#) *center, [Point](#) *perimeter_point)
- [Parallel](#) * [parallel_create](#) ([CoordinateSystem](#) *cs, [Line](#) *line, [Point](#) *point)
- [Perpendicular](#) * [perpendicular_create](#) ([CoordinateSystem](#) *cs, [Line](#) *line, [Point](#) *point)
- [AngleBisector](#) * [angle_bisector_create](#) ([CoordinateSystem](#) *cs, [Line](#) *line1, [Line](#) *line2)
- [Tangent](#) * [tangent_create](#) ([CoordinateSystem](#) *cs, [Circle](#) *circle, [Point](#) *point)
- void [shape_draw](#) ([CoordinateSystem](#) *cs, [Shape](#) *self)
- void [shape_update](#) ([CoordinateSystem](#) *cs, [Shape](#) *self)
- void [shape_translate](#) ([CoordinateSystem](#) *cs, [Shape](#) *self, [Vector2](#) translation)
- void [shape_destroy](#) ([CoordinateSystem](#) *cs, [Shape](#) *self)
- bool [shape_overlap_point](#) ([CoordinateSystem](#) *cs, [Shape](#) *self, [Vector2](#) point)
- bool [shape_is_defined_by](#) ([Shape](#) *self, [Shape](#) *shape)

4.17.1 Macro Definition Documentation

4.17.1.1 OVERLAP_DISTANCE

```
#define OVERLAP_DISTANCE 5
```

4.17.2 Typedef Documentation

4.17.2.1 AngleBisector

```
typedef struct AngleBisector AngleBisector
```

The angle bisector lines struct.

4.17.2.2 Circle

```
typedef struct Circle Circle
```

The circle struct.

4.17.2.3 CoordinateSystem

```
typedef struct CoordinateSystem CoordinateSystem
```

4.17.2.4 Line

```
typedef struct Line Line
```

The line struct.

4.17.2.5 Parallel

```
typedef struct Parallel Parallel
```

The parallel line struct.

4.17.2.6 Perpendicular

```
typedef struct Perpendicular Perpendicular
```

The perpendicular line struct.

4.17.2.7 Point

```
typedef struct Point Point
```

The point struct.

4.17.2.8 Shape

```
typedef struct Shape Shape
```

The base shape struct (needed for polymorphism)

4.17.2.9 ShapeDestroy

```
typedef void(* ShapeDestroy) (struct CoordinateSystem *cs, struct Shape *self)
```

4.17.2.10 ShapeDraw

```
typedef void(* ShapeDraw) (struct CoordinateSystem *cs, struct Shape *self)
```


4.17.2.11 ShapelsDefinedBy

```
typedef bool(* ShapeIsDefinedBy) (struct Shape *self, struct Shape *shape)
```

4.17.2.12 ShapeOverlapPoint

```
typedef bool(* ShapeOverlapPoint) (struct CoordinateSystem *cs, struct Shape *self, Vector2
point)
```

4.17.2.13 ShapeTranslate

```
typedef void(* ShapeTranslate) (struct CoordinateSystem *cs, struct Shape *self, Vector2 translation)
```

4.17.2.14 ShapeType

```
typedef enum ShapeType ShapeType
```

The types of shapes that can be created.

4.17.2.15 Tangent

```
typedef struct Tangent Tangent
```

The tangent lines struct (circle tangents)

4.17.3 Enumeration Type Documentation

4.17.3.1 ShapeType

```
enum ShapeType
```

The types of shapes that can be created.

Enumerator

ST_POINT	
ST_LINE	
ST_CIRCLE	
ST_PARALLEL	
ST_PERPENDICULAR	
ST_ANGLE_BISECTOR	
ST_TANGENT	
ST_COUNT	

4.17.4 Function Documentation

4.17.4.1 angle_bisector_create()

```
AngleBisector * angle_bisector_create (
    CoordinateSystem * cs,
    Line * line1,
    Line * line2 )
```

Creates an angle bisector in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the angle bisector in
<i>line1</i>	The first line to create the angle bisector to
<i>line2</i>	The second line to create the angle bisector to

Returns

AngleBisector* The created angle bisector

4.17.4.2 circle_create()

```
Circle * circle_create (
    CoordinateSystem * cs,
    Point * center,
    Point * perimeter_point )
```

Creates a circle in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the circle in
<i>center</i>	The center of the circle
<i>perimeter_point</i>	A point on the perimeter of the circle (has to be different from center)

Returns

Circle* The created circle

4.17.4.3 line_create()

```
Line * line_create (
    CoordinateSystem * cs,
    Point * p1,
    Point * p2 )
```

Creates a line in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the line in
<i>p1</i>	A point of the line
<i>p2</i>	Another point of the line (has to be different from p1)

Returns

Line* The created line

4.17.4.4 parallel_create()

```
Parallel * parallel_create (
    CoordinateSystem * cs,
    Line * line,
    Point * point )
```

Creates a parallel line in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the parallel line in
<i>line</i>	The line to create the parallel line to
<i>point</i>	The point the parallel line goes through

Returns

Parallel* The created parallel line

4.17.4.5 perpendicular_create()

```
Perpendicular * perpendicular_create (
    CoordinateSystem * cs,
    Line * line,
    Point * point )
```

Creates a perpendicular line in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the perpendicular line in
<i>line</i>	The line to create the perpendicular line to
<i>point</i>	The point the perpendicular line goes through

Returns

Perpendicular* The created perpendicular line

4.17.4.6 point_create()

```
Point * point_create (
    CoordinateSystem * cs,
    Vector2 coordinates )
```

Creates a point in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the point in
<i>coordinates</i>	The coordinates of the point

Returns

Point* The created point

4.17.4.7 shape_destroy()

```
void shape_destroy (
    CoordinateSystem * cs,
    Shape * self )
```

Destroys a shape, but does not remove it from the coordinate systems shapes! (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to destroy the shape in
<i>self</i>	The shape to destroy

4.17.4.8 shape_draw()

```
void shape_draw (
    CoordinateSystem * cs,
    Shape * self )
```

Draws a shape (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to draw the point in
<i>self</i>	The shape to draw

4.17.4.9 shape_is_defined_by()

```
bool shape_is_defined_by (
    Shape * self,
    Shape * shape )
```

Checks if a shape is defined by another shape (can be called on any shape)

Parameters

<i>self</i>	The shape to check if it is defined by the other shape
<i>shape</i>	The shape to check if it defines the other shape

Returns

true If the shape is defined by the other shape
false If the shape is not defined by the other shape

4.17.4.10 shape_overlap_point()

```
bool shape_overlap_point (
    CoordinateSystem * cs,
    Shape * self,
    Vector2 point )
```

Checks if a point overlaps the shape (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to check the overlap in
<i>self</i>	The shape to check the overlap with
<i>point</i>	The point to check the overlap with

Returns

true If the point overlaps with the shape
false If the point does not overlap with the shape

4.17.4.11 shape_translate()

```
void shape_translate (
    CoordinateSystem * cs,
    Shape * self,
    Vector2 translation )
```

Translates a shape if it is allowed (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to translate the shape in
<i>self</i>	The shape to translate
<i>translation</i>	The translation vector

4.17.4.12 shape_update()

```
void shape_update (
    CoordinateSystem * cs,
    Shape * self )
```

Updates a shape (can be called on any shape)

Parameters

<i>cs</i>	The coordinate system to update the shape in
<i>self</i>	The shape to update

4.17.4.13 tangent_create()

```
Tangent * tangent_create (
    CoordinateSystem * cs,
    Circle * circle,
    Point * point )
```

Creates a tangent to a circle in the coordinate system.

Parameters

<i>cs</i>	The coordinate system to create the tangent in
<i>circle</i>	The circle to create the tangent to
<i>point</i>	The point the tangent goes through

Returns

Tangent* The created tangent

4.18 shape.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdbool.h>
00004
00005 #include "../vector2/vector2.h"
00006
00007 #define OVERLAP_DISTANCE 5
00008
00009 typedef struct CoordinateSystem CoordinateSystem;
00010 typedef struct Shape Shape;
00011
00012 typedef void (*ShapeDraw)(struct CoordinateSystem* cs, struct Shape* self);
00013 typedef void (*ShapeTranslate)(struct CoordinateSystem* cs, struct Shape* self, Vector2 translation);
00014 typedef void (*ShapeDestroy)(struct CoordinateSystem* cs, struct Shape* self);
00015 typedef bool (*ShapeOverlapPoint)(struct CoordinateSystem* cs, struct Shape* self, Vector2 point);
00016 typedef bool (*ShapeIsDefinedBy)(struct Shape* self, struct Shape* shape);
00017
00021 typedef enum ShapeType
00022 {
00023     ST_POINT,
00024     ST_LINE,
00025     ST_CIRCLE,
```

```

00026
00027     ST_PARALLEL,
00028     ST_PERPENDICULAR,
00029     ST_ANGLE_BISECTOR,
00030     ST_TANGENT,
00031
00032     ST_COUNT
00033 } ShapeType;
00034
00038 typedef struct Shape
00039 {
00040     ShapeType type;
00041     bool selected;
00042     bool dragged;
00043 } Shape;
00044
00048 typedef struct Point
00049 {
00050     Shape base;
00051     Vector2 coordinates;
00052 } Point;
00053
00057 typedef struct Line
00058 {
00059     Shape base;
00060     Point *p1, *p2;
00061 } Line;
00062
00066 typedef struct Circle
00067 {
00068     Shape base;
00069     Point* center;
00070     Point* perimeter_point;
00071 } Circle;
00072
00076 typedef struct Parallel
00077 {
00078     Shape base;
00079     Line* line;
00080     Point* point;
00081 } Parallel;
00082
00086 typedef struct Perpendicular
00087 {
00088     Shape base;
00089     Line* line;
00090     Point* point;
00091 } Perpendicular;
00092
00096 typedef struct AngleBisector
00097 {
00098     Shape base;
00099     Line* line1;
00100     Line* line2;
00101 } AngleBisector;
00102
00106 typedef struct Tangent
00107 {
00108     Shape base;
00109     Circle* circle;
00110     Point* point;
00111 } Tangent;
00112
00120 Point* point_create(CoordinateSystem* cs, Vector2 coordinates);
00129 Line* line_create(CoordinateSystem* cs, Point* p1, Point* p2);
00138 Circle* circle_create(CoordinateSystem* cs, Point* center, Point* perimeter_point);
00147 Parallel* parallel_create(CoordinateSystem* cs, Line* line, Point* point);
00156 Perpendicular* perpendicular_create(CoordinateSystem* cs, Line* line, Point* point);
00165 AngleBisector* angle_bisector_create(CoordinateSystem* cs, Line* line1, Line* line2);
00174 Tangent* tangent_create(CoordinateSystem* cs, Circle* circle, Point* point);
00175
00182 void shape_draw(CoordinateSystem* cs, Shape* self);
00189 void shape_update(CoordinateSystem* cs, Shape* self);
00197 void shape_translate(CoordinateSystem* cs, Shape* self, Vector2 translation);
00204 void shape_destroy(CoordinateSystem* cs, Shape* self);
00214 bool shape_overlap_point(CoordinateSystem* cs, Shape* self, Vector2 point);
00223 bool shape_is_defined_by(Shape* self, Shape* shape);

```

4.19 src/geometry/vector2/vector2.c File Reference

Functions

- [Vector2 vector2_create](#) (double x, double y)

- [Vector2 vector2_from_polar](#) (double angle, double length)
- [Vector2 vector2_from_point](#) (SDL_Point point)
- [Vector2 vector2_zero](#) ()
- [Vector2 vector2_one](#) ()
- [Vector2 vector2_up](#) ()
- [Vector2 vector2_down](#) ()
- [Vector2 vector2_left](#) ()
- [Vector2 vector2_right](#) ()
- [Vector2 vector2_add](#) (Vector2 a, Vector2 b)
- [Vector2 vector2_subtract](#) (Vector2 a, Vector2 b)
- [Vector2 vector2_scale](#) (Vector2 a, double b)
- [Vector2 vector2_negate](#) (Vector2 a)
- [Vector2 vector2_multiply](#) (Vector2 a, Vector2 b)
- [Vector2 vector2_divide](#) (Vector2 a, Vector2 b)
- double [vector2_dot](#) (Vector2 a, Vector2 b)
- double [vector2_cross](#) (Vector2 a, Vector2 b)
- double [vector2_length](#) (Vector2 a)
- double [vector2_distance](#) (Vector2 a, Vector2 b)
- double [vector2_angle](#) (Vector2 a)
- [Vector2 vector2_normalize](#) (Vector2 a)
- [Vector2 vector2_rotate90](#) (Vector2 a)
- [Vector2 vector2_rotate](#) (Vector2 a, double angle)
- [Vector2 vector2_reflect](#) (Vector2 a, Vector2 normal)

4.19.1 Function Documentation

4.19.1.1 vector2_add()

```
Vector2 vector2_add (
    Vector2 a,
    Vector2 b )
```

Adds two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

[Vector2](#) The sum of the two vectors

4.19.1.2 vector2_angle()

```
double vector2_angle (
    Vector2 a )
```

Calculates the angle of a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

double The angle of the vector

4.19.1.3 vector2_create()

```
Vector2 vector2_create (  
    double x,  
    double y )
```

Creates a vector from x and y values.

Parameters

<i>x</i>	The x value
<i>y</i>	The y value

Returns

Vector2 The vector

4.19.1.4 vector2_cross()

```
double vector2_cross (  
    Vector2 a,  
    Vector2 b )
```

Calculates the cross product of two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

double The cross product of the two vectors

4.19.1.5 vector2_distance()

```
double vector2_distance (  
    Vector2 a,  
    Vector2 b )
```

Calculates the distance between two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

double The distance between the two vectors

4.19.1.6 vector2_divide()

```
Vector2 vector2_divide (  
    Vector2 a,  
    Vector2 b )
```

Divides two vectors component-wise.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

Vector2 The quotient of the two vectors

4.19.1.7 vector2_dot()

```
double vector2_dot (  
    Vector2 a,  
    Vector2 b )
```

Calculates the dot product of two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

double The dot product of the two vectors

4.19.1.8 vector2_down()

```
Vector2 vector2_down ( )
```

Returns a vector with x value of 0 and y value of -1.

Returns

[Vector2](#) The vector

4.19.1.9 vector2_from_point()

```
Vector2 vector2_from_point (
    SDL_Point point )
```

Creates a vector from an SDL_Point.

Parameters

<i>point</i>	The point
--------------	-----------

Returns

[Vector2](#) The vector

4.19.1.10 vector2_from_polar()

```
Vector2 vector2_from_polar (
    double angle,
    double length )
```

Creates a vector from polar coordinates.

Parameters

<i>angle</i>	The angle
<i>length</i>	The length

Returns

[Vector2](#) The vector

4.19.1.11 vector2_left()

```
Vector2 vector2_left ( )
```

Returns a vector with x value of -1 and y value of 0.

Returns

[Vector2](#) The vector

4.19.1.12 vector2_length()

```
double vector2_length (  
    Vector2 a )
```

Calculates the length of a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

double The length of the vector

4.19.1.13 vector2_multiply()

```
Vector2 vector2_multiply (  
    Vector2 a,  
    Vector2 b )
```

Multiplies two vectors component-wise.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

Vector2 The product of the two vectors

4.19.1.14 vector2_negate()

```
Vector2 vector2_negate (  
    Vector2 a )
```

Negates a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

Vector2 The negated vector

4.19.1.15 vector2_normalize()

```
Vector2 vector2_normalize (  
    Vector2 a )
```

Normalizes a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

[Vector2](#) The normalized vector

4.19.1.16 vector2_one()

```
Vector2 vector2_one ( )
```

Returns a vector with x and y values of 1.

Returns

[Vector2](#) The vector

4.19.1.17 vector2_reflect()

```
Vector2 vector2_reflect (
    Vector2 a,
    Vector2 normal )
```

Reflects a vector over a normal.

Parameters

<i>a</i>	The vector
<i>normal</i>	The normal

Returns

[Vector2](#) The reflected vector

4.19.1.18 vector2_right()

```
Vector2 vector2_right ( )
```

Returns a vector with x value of 1 and y value of 0.

Returns

[Vector2](#) The vector

4.19.1.19 vector2_rotate()

```
Vector2 vector2_rotate (
    Vector2 a,
    double angle )
```

Rotates a vector by an angle.

Parameters

<i>a</i>	The vector
<i>angle</i>	The angle

Returns

[Vector2](#) The rotated vector

4.19.1.20 vector2_rotate90()

```
Vector2 vector2_rotate90 (  
    Vector2 a )
```

Rotates a vector by 90 degrees.

Parameters

<i>a</i>	The vector
----------	------------

Returns

[Vector2](#) The rotated vector

4.19.1.21 vector2_scale()

```
Vector2 vector2_scale (  
    Vector2 a,  
    double b )
```

Scales a vector by a scalar.

Parameters

<i>a</i>	The vector
<i>b</i>	The scale

Returns

[Vector2](#) The scaled vector

4.19.1.22 vector2_subtract()

```
Vector2 vector2_subtract (  
    Vector2 a,  
    Vector2 b )
```

Subtracts two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

[Vector2](#) The difference of the two vectors

4.19.1.23 vector2_up()

```
Vector2 vector2_up ( )
```

Returns a vector with x value of 0 and y value of 1.

Returns

[Vector2](#) The vector

4.19.1.24 vector2_zero()

```
Vector2 vector2_zero ( )
```

Returns a vector with x and y values of 0.

Returns

[Vector2](#) The vector

4.20 src/geometry/vector2/vector2.h File Reference

Classes

- struct [Vector2](#)

Functions

- [Vector2 vector2_create](#) (double x, double y)
- [Vector2 vector2_from_polar](#) (double angle, double length)
- [Vector2 vector2_from_point](#) (SDL_Point point)
- [Vector2 vector2_zero](#) ()
- [Vector2 vector2_one](#) ()
- [Vector2 vector2_up](#) ()
- [Vector2 vector2_down](#) ()
- [Vector2 vector2_left](#) ()
- [Vector2 vector2_right](#) ()
- [Vector2 vector2_add](#) (Vector2 a, Vector2 b)
- [Vector2 vector2_subtract](#) (Vector2 a, Vector2 b)
- [Vector2 vector2_scale](#) (Vector2 a, double b)
- [Vector2 vector2_negate](#) (Vector2 a)
- [Vector2 vector2_multiply](#) (Vector2 a, Vector2 b)
- [Vector2 vector2_divide](#) (Vector2 a, Vector2 b)
- [double vector2_dot](#) (Vector2 a, Vector2 b)
- [double vector2_cross](#) (Vector2 a, Vector2 b)
- [double vector2_length](#) (Vector2 a)
- [double vector2_distance](#) (Vector2 a, Vector2 b)
- [double vector2_angle](#) (Vector2 a)
- [Vector2 vector2_normalize](#) (Vector2 a)
- [Vector2 vector2_rotate90](#) (Vector2 a)
- [Vector2 vector2_rotate](#) (Vector2 a, double angle)
- [Vector2 vector2_reflect](#) (Vector2 a, Vector2 normal)

4.20.1 Typedef Documentation

4.20.1.1 Vector2

```
typedef struct Vector2 Vector2
```

A 2D vector, used for coordinate geometry.

4.20.2 Function Documentation

4.20.2.1 vector2_add()

```
Vector2 vector2_add (
    Vector2 a,
    Vector2 b )
```

Adds two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

[Vector2](#) The sum of the two vectors

4.20.2.2 vector2_angle()

```
double vector2_angle (  
    Vector2 a )
```

Calculates the angle of a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

double The angle of the vector

4.20.2.3 vector2_create()

```
Vector2 vector2_create (  
    double x,  
    double y )
```

Creates a vector from x and y values.

Parameters

<i>x</i>	The x value
<i>y</i>	The y value

Returns

[Vector2](#) The vector

4.20.2.4 vector2_cross()

```
double vector2_cross (  
    Vector2 a,  
    Vector2 b )
```

Calculates the cross product of two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

double The cross product of the two vectors

4.20.2.5 vector2_distance()

```
double vector2_distance (
    Vector2 a,
    Vector2 b )
```

Calculates the distance between two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

double The distance between the two vectors

4.20.2.6 vector2_divide()

```
Vector2 vector2_divide (
    Vector2 a,
    Vector2 b )
```

Divides two vectors component-wise.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

Vector2 The quotient of the two vectors

4.20.2.7 vector2_dot()

```
double vector2_dot (
    Vector2 a,
    Vector2 b )
```

Calculates the dot product of two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

double The dot product of the two vectors

4.20.2.8 vector2_down()

```
Vector2 vector2_down ( )
```

Returns a vector with x value of 0 and y value of -1.

Returns

[Vector2](#) The vector

4.20.2.9 vector2_from_point()

```
Vector2 vector2_from_point (
    SDL_Point point )
```

Creates a vector from an SDL_Point.

Parameters

<i>point</i>	The point
--------------	-----------

Returns

[Vector2](#) The vector

4.20.2.10 vector2_from_polar()

```
Vector2 vector2_from_polar (
    double angle,
    double length )
```

Creates a vector from polar coordinates.

Parameters

<i>angle</i>	The angle
<i>length</i>	The length

Returns

[Vector2](#) The vector

4.20.2.11 vector2_left()

```
Vector2 vector2_left ( )
```

Returns a vector with x value of -1 and y value of 0.

Returns

[Vector2](#) The vector

4.20.2.12 vector2_length()

```
double vector2_length (
    Vector2 a )
```

Calculates the length of a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

double The length of the vector

4.20.2.13 vector2_multiply()

```
Vector2 vector2_multiply (
    Vector2 a,
    Vector2 b )
```

Multiplies two vectors component-wise.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

[Vector2](#) The product of the two vectors

4.20.2.14 vector2_negate()

```
Vector2 vector2_negate (
    Vector2 a )
```

Negates a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

[Vector2](#) The negated vector

4.20.2.15 vector2_normalize()

```
Vector2 vector2_normalize (  
    Vector2 a )
```

Normalizes a vector.

Parameters

<i>a</i>	The vector
----------	------------

Returns

[Vector2](#) The normalized vector

4.20.2.16 vector2_one()

```
Vector2 vector2_one ( )
```

Returns a vector with x and y values of 1.

Returns

[Vector2](#) The vector

4.20.2.17 vector2_reflect()

```
Vector2 vector2_reflect (  
    Vector2 a,  
    Vector2 normal )
```

Reflects a vector over a normal.

Parameters

<i>a</i>	The vector
<i>normal</i>	The normal

Returns

[Vector2](#) The reflected vector

4.20.2.18 vector2_right()

```
Vector2 vector2_right ( )
```

Returns a vector with x value of 1 and y value of 0.

Returns

[Vector2](#) The vector

4.20.2.19 vector2_rotate()

```
Vector2 vector2_rotate (
    Vector2 a,
    double angle )
```

Rotates a vector by an angle.

Parameters

<i>a</i>	The vector
<i>angle</i>	The angle

Returns

[Vector2](#) The rotated vector

4.20.2.20 vector2_rotate90()

```
Vector2 vector2_rotate90 (
    Vector2 a )
```

Rotates a vector by 90 degrees.

Parameters

<i>a</i>	The vector
----------	------------

Returns

[Vector2](#) The rotated vector

4.20.2.21 vector2_scale()

```
Vector2 vector2_scale (
    Vector2 a,
    double b )
```

Scales a vector by a scalar.

Parameters

<i>a</i>	The vector
<i>b</i>	The scale

Returns

[Vector2](#) The scaled vector

4.20.2.22 vector2_subtract()

```
Vector2 vector2_subtract (
    Vector2 a,
    Vector2 b )
```

Subtracts two vectors.

Parameters

<i>a</i>	The first vector
<i>b</i>	The second vector

Returns

[Vector2](#) The difference of the two vectors

4.20.2.23 vector2_up()

```
Vector2 vector2_up ( )
```

Returns a vector with x value of 0 and y value of 1.

Returns

[Vector2](#) The vector

4.20.2.24 vector2_zero()

```
Vector2 vector2_zero ( )
```

Returns a vector with x and y values of 0.

Returns

[Vector2](#) The vector

4.21 vector2.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #ifdef _WIN32
00004     #include <SDL.h>
00005 #elif defined(__unix__) || defined(__linux__)
00006     #include <SDL2/SDL.h>
00007 #endif
00008
00012 typedef struct Vector2
00013 {
00014     double x, y;
00015 } Vector2;
00016
00024 Vector2 vector2_create(double x, double y);
00032 Vector2 vector2_from_polar(double angle, double length);
00039 Vector2 vector2_from_point(SDL_Point point);
00040
00046 Vector2 vector2_zero();
00052 Vector2 vector2_one();
00058 Vector2 vector2_up();
00064 Vector2 vector2_down();
00070 Vector2 vector2_left();
00076 Vector2 vector2_right();
00077
00085 Vector2 vector2_add(Vector2 a, Vector2 b);
00093 Vector2 vector2_subtract(Vector2 a, Vector2 b);
00101 Vector2 vector2_scale(Vector2 a, double b);
00108 Vector2 vector2_negate(Vector2 a);
00116 Vector2 vector2_multiply(Vector2 a, Vector2 b);
00124 Vector2 vector2_divide(Vector2 a, Vector2 b);
00132 double vector2_dot(Vector2 a, Vector2 b);
00140 double vector2_cross(Vector2 a, Vector2 b);
00147 double vector2_length(Vector2 a);
00155 double vector2_distance(Vector2 a, Vector2 b);
00162 double vector2_angle(Vector2 a);
00169 Vector2 vector2_normalize(Vector2 a);
00176 Vector2 vector2_rotate90(Vector2 a);
00184 Vector2 vector2_rotate(Vector2 a, double angle);
00192 Vector2 vector2_reflect(Vector2 a, Vector2 normal);

```

4.22 src/input/input.c File Reference

Functions

- bool [input_is_mouse_button_down](#) (int button)
- bool [input_is_mouse_button_pressed](#) (int button)
- bool [input_is_mouse_button_released](#) (int button)
- bool [input_is_key_down](#) (int key)
- bool [input_is_key_pressed](#) (int key)
- bool [input_is_key_released](#) (int key)
- SDL_Point [input_get_mouse_position](#) ()
- SDL_Point [input_get_mouse_motion](#) ()
- int [input_get_mouse_wheel_delta](#) ()
- void [_input_init](#) (InputData *input_data)
- void [_input_handle_event](#) (InputData *input_data, SDL_Event *event)
- void [_input_reset](#) (InputData *input_data)
- void [_input_close](#) (InputData *input_data)
- void [_input_set_target](#) (InputData *input_data)

4.22.1 Function Documentation

4.22.1.1 `_input_close()`

```
void _input_close (  
    InputData * input_data )
```

Frees input data resources (should not be called manually)

Parameters

<i>input_data</i>	The input data to close
-------------------	-------------------------

4.22.1.2 _input_handle_event()

```
void _input_handle_event (
    InputData * input_data,
    SDL_Event * event )
```

Handles an input event (should not be called manually)

Parameters

<i>input_data</i>	The input data to change based on the event
<i>event</i>	The event to handle

4.22.1.3 _input_init()

```
void _input_init (
    InputData * input_data )
```

Initializes the input data (should not be called manually)

Parameters

<i>input_data</i>	The input data to initialize
-------------------	------------------------------

4.22.1.4 _input_reset()

```
void _input_reset (
    InputData * input_data )
```

Resets the input data (should not be called manually)

Parameters

<i>input_data</i>	The input data to reset
-------------------	-------------------------

4.22.1.5 _input_set_target()

```
void _input_set_target (
    InputData * input_data )
```

Sets the target input data for the application (should not be called manually)

Parameters

<i>input_data</i>	The input data to set as target
-------------------	---------------------------------

4.22.1.6 input_get_mouse_motion()

```
SDL_Point input_get_mouse_motion ( )
```

Returns the mouse motion since the last frame.

Returns

SDL_Point The mouse motion

4.22.1.7 input_get_mouse_position()

```
SDL_Point input_get_mouse_position ( )
```

Returns the mouse position.

Returns

SDL_Point The mouse position

4.22.1.8 input_get_mouse_wheel_delta()

```
int input_get_mouse_wheel_delta ( )
```

Returns the mouse wheel delta since the last frame.

Returns

int The mouse wheel delta

4.22.1.9 input_is_key_down()

```
bool input_is_key_down (
    int key )
```

Returns if the key is down (being held)

Parameters

<i>key</i>	The key to check
------------	------------------

Returns

true Returns true if the key is held
false Returns false if the key is not held

4.22.1.10 input_is_key_pressed()

```
bool input_is_key_pressed (  
    int key )
```

Returns if the key has just been pressed.

Parameters

<i>key</i>	The key to check
------------	------------------

Returns

true Returns true if the key has just been pressed
false Returns false if the key has not been pressed

4.22.1.11 input_is_key_released()

```
bool input_is_key_released (  
    int key )
```

Returns if the key has just been released.

Parameters

<i>key</i>	The key to check
------------	------------------

Returns

true Returns true if the key has just been released
false Returns false if the key has not been released

4.22.1.12 input_is_mouse_button_down()

```
bool input_is_mouse_button_down (  
    int button )
```

Returns if the mouse button is down (being held)

Parameters

<i>button</i>	The mouse button to check
---------------	---------------------------

Returns

true Returns true if the mouse button is held
false Returns false if the mouse button is held

4.22.1.13 input_is_mouse_button_pressed()

```
bool input_is_mouse_button_pressed (
    int button )
```

Returns if the mouse button has just been pressed.

Parameters

<i>button</i>	The mouse button to check
---------------	---------------------------

Returns

true Returns true if the mouse button has just been pressed
false Returns false if the mouse button has not been pressed

4.22.1.14 input_is_mouse_button_released()

```
bool input_is_mouse_button_released (
    int button )
```

Returns if the mouse button has just been released.

Parameters

<i>button</i>	The mouse button to check
---------------	---------------------------

Returns

true Returns true if the mouse button has just been released
false Returns false if the mouse button has not been released

4.23 src/input/input.h File Reference**Classes**

- struct [InputData](#)

Functions

- bool [input_is_mouse_button_down](#) (int button)
- bool [input_is_mouse_button_pressed](#) (int button)
- bool [input_is_mouse_button_released](#) (int button)
- bool [input_is_key_down](#) (int key)
- bool [input_is_key_pressed](#) (int key)
- bool [input_is_key_released](#) (int key)
- SDL_Point [input_get_mouse_position](#) ()
- SDL_Point [input_get_mouse_motion](#) ()
- int [input_get_mouse_wheel_delta](#) ()
- void [_input_init](#) (InputData *input_data)
- void [_input_handle_event](#) (InputData *input_data, SDL_Event *event)
- void [_input_reset](#) (InputData *input_data)
- void [_input_close](#) (InputData *input_data)
- void [_input_set_target](#) (InputData *input_data)

4.23.1 Typedef Documentation

4.23.1.1 InputData

```
typedef struct InputData InputData
```

Holds the input data of a window (needed for press, hold, release events and mouse motion)

4.23.2 Function Documentation

4.23.2.1 _input_close()

```
void _input_close (
    InputData * input_data )
```

Frees input data resources (should not be called manually)

Parameters

<i>input_data</i>	The input data to close
-------------------	-------------------------

4.23.2.2 _input_handle_event()

```
void _input_handle_event (
    InputData * input_data,
    SDL_Event * event )
```

Handles an input event (should not be called manually)

Parameters

<i>input_data</i>	The input data to change based on the event
<i>event</i>	The event to handle

4.23.2.3 `_input_init()`

```
void _input_init (
    InputData * input_data )
```

Initializes the input data (should not be called manually)

Parameters

<i>input_data</i>	The input data to initialize
-------------------	------------------------------

4.23.2.4 `_input_reset()`

```
void _input_reset (
    InputData * input_data )
```

Resets the input data (should not be called manually)

Parameters

<i>input_data</i>	The input data to reset
-------------------	-------------------------

4.23.2.5 `_input_set_target()`

```
void _input_set_target (
    InputData * input_data )
```

Sets the target input data for the application (should not be called manually)

Parameters

<i>input_data</i>	The input data to set as target
-------------------	---------------------------------

4.23.2.6 `input_get_mouse_motion()`

```
SDL_Point input_get_mouse_motion ( )
```

Returns the mouse motion since the last frame.

Returns

SDL_Point The mouse motion

4.23.2.7 input_get_mouse_position()

```
SDL_Point input_get_mouse_position ( )
```

Returns the mouse position.

Returns

SDL_Point The mouse position

4.23.2.8 input_get_mouse_wheel_delta()

```
int input_get_mouse_wheel_delta ( )
```

Returns the mouse wheel delta since the last frame.

Returns

int The mouse wheel delta

4.23.2.9 input_is_key_down()

```
bool input_is_key_down (
    int key )
```

Returns if the key is down (being held)

Parameters

<i>key</i>	The key to check
------------	------------------

Returns

true Returns true if the key is held

false Returns false if the key is not held

4.23.2.10 input_is_key_pressed()

```
bool input_is_key_pressed (
    int key )
```

Returns if the key has just been pressed.

Parameters

<i>key</i>	The key to check
------------	------------------

Returns

true Returns true if the key has just been pressed
false Returns false if the key has not been pressed

4.23.2.11 input_is_key_released()

```
bool input_is_key_released (
    int key )
```

Returns if the key has just been released.

Parameters

<i>key</i>	The key to check
------------	------------------

Returns

true Returns true if the key has just been released
false Returns false if the key has not been released

4.23.2.12 input_is_mouse_button_down()

```
bool input_is_mouse_button_down (
    int button )
```

Returns if the mouse button is down (being held)

Parameters

<i>button</i>	The mouse button to check
---------------	---------------------------

Returns

true Returns true if the mouse button is held
false Returns false if the mouse button is held

4.23.2.13 input_is_mouse_button_pressed()

```
bool input_is_mouse_button_pressed (
    int button )
```

Returns if the mouse button has just been pressed.

Parameters

<i>button</i>	The mouse button to check
---------------	---------------------------

Returns

true Returns true if the mouse button has just been pressed
 false Returns false if the mouse button has not been pressed

4.23.2.14 input_is_mouse_button_released()

```
bool input_is_mouse_button_released (
    int button )
```

Returns if the mouse button has just been released.

Parameters

<i>button</i>	The mouse button to check
---------------	---------------------------

Returns

true Returns true if the mouse button has just been released
 false Returns false if the mouse button has not been released

4.24 input.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #ifndef _WIN32
00004     #include <SDL.h>
00005 #elif defined(__unix__) || defined(__linux__)
00006     #include <SDL2/SDL.h>
00007 #endif
00008
00009 #include <stdbool.h>
00010
00014 typedef struct InputData
00015 {
00016     //mouse
00017     bool current_mouse_button_state[5];
00018     bool old_mouse_button_state[5];
00019     SDL_Point current_mouse_position;
00020     SDL_Point old_mouse_position;
00021     int mouse_wheel_delta;
00022
00023     //keyboard
00024     Uint8* current_keyboard_state;
00025     Uint8* old_keyboard_state;
00026     int key_count;
00027 } InputData;
00028
00036 bool input_is_mouse_button_down(int button);
00044 bool input_is_mouse_button_pressed(int button);
00052 bool input_is_mouse_button_released(int button);
00053
00061 bool input_is_key_down(int key);
00069 bool input_is_key_pressed(int key);
00077 bool input_is_key_released(int key);
00078
00084 SDL_Point input_get_mouse_position();
00090 SDL_Point input_get_mouse_motion();
00096 int input_get_mouse_wheel_delta();
00097
00103 void _input_init(InputData* input_data);
00110 void _input_handle_event(InputData* input_data, SDL_Event* event);
00116 void _input_reset(InputData* input_data);
00122 void _input_close(InputData* input_data);
00128 void _input_set_target(InputData* input_data);
```

4.25 src/main.c File Reference

4.25.1 Detailed Description

Author

Örs Mándli (mandliors@gmail.com)

Version

0.1

Date

2023-11-25

Copyright

Copyright (c) 2023

4.25.2 Macro Definition Documentation

4.25.2.1 FPS

```
#define FPS 60
```

4.25.2.2 MOUSE_WHEEL_SENSITIVITY

```
#define MOUSE_WHEEL_SENSITIVITY 5
```

4.25.3 Typedef Documentation

4.25.3.1 State

```
typedef enum State State
```

Enumerator

4.25.4 Enumeration Type Documentation

4.25.4.1 State

```
enum State
```

Enumerator

STATE_POINTER	
STATE_CS_DRAGGED	
STATE_POINT	
STATE_LINE	
STATE_LINE_POINT1_PLACED	
STATE_CIRCLE	
STATE_CIRCLE_CENTER_PLACED	
STATE_PARALLEL	
STATE_PARALLEL_LINE_SELECTED	
STATE_PERPENDICULAR	
STATE_PERPENDICULAR_LINE_SELECTED	
STATE_ANGLE_BISECTOR	
STATE_ANGLE_BISECTOR_LINE1_SELECTED	
STATE_TANGENT	
STATE_TANGENT_LINE_SELECTED	
STATE_OPENING	
STATE_SAVEING	

4.25.5 Function Documentation

4.25.5.1 main()

```
int main (
    void )
```

4.25.5.2 on_angle_bisector_clicked() [1/2]

```
void on_angle_bisector_clicked (
    UIButton *self __attribute__((unused)) )
```

4.25.5.3 on_angle_bisector_clicked() [2/2]

```
void on_angle_bisector_clicked (
    UIButton * self )
```

4.25.5.4 on_cancel_button_clicked()

```
void on_cancel_button_clicked (
    UIButton * self )
```

4.25.5.5 on_canvas_size_changed()

```
void on_canvas_size_changed (
    UIContainer * self,
    SDL_Point size )
```

4.25.5.6 on_circle_clicked() [1/2]

```
void on_circle_clicked (
    UIButton *self __attribute__((unused)) )
```

4.25.5.7 on_circle_clicked() [2/2]

```
void on_circle_clicked (
    UIButton * self )
```

4.25.5.8 on_editmenu_clicked() [1/2]

```
void on_editmenu_clicked (
    UISplitButton *self __attribute__((unused)),
    Sint32 index __attribute__((unused)) )
```

4.25.5.9 on_editmenu_clicked() [2/2]

```
void on_editmenu_clicked (
    UISplitButton * self,
    Sint32 index )
```

4.25.5.10 on_filemenu_clicked() [1/2]

```
void on_filemenu_clicked (
    UISplitButton *self __attribute__((unused)),
    Sint32 index __attribute__((unused)) )
```

4.25.5.11 on_filemenu_clicked() [2/2]

```
void on_filemenu_clicked (
    UISplitButton * self,
    Sint32 index )
```

4.25.5.12 on_line_clicked() [1/2]

```
void on_line_clicked (
    UIButton *self  __attribute__((unused)) )
```

4.25.5.13 on_line_clicked() [2/2]

```
void on_line_clicked (
    UIButton * self )
```

4.25.5.14 on_open_button_clicked()

```
void on_open_button_clicked (
    UIButton * self )
```

4.25.5.15 on_parallel_clicked() [1/2]

```
void on_parallel_clicked (
    UIButton *self  __attribute__((unused)) )
```

4.25.5.16 on_parallel_clicked() [2/2]

```
void on_parallel_clicked (
    UIButton * self )
```

4.25.5.17 on_perpendicular_clicked() [1/2]

```
void on_perpendicular_clicked (
    UIButton *self  __attribute__((unused)) )
```

4.25.5.18 on_perpendicular_clicked() [2/2]

```
void on_perpendicular_clicked (
    UIButton * self )
```

4.25.5.19 on_point_clicked() [1/2]

```
void on_point_clicked (
    UIButton *self  __attribute__((unused)) )
```

4.25.5.20 on_point_clicked() [2/2]

```
void on_point_clicked (
    UIButton * self )
```


4.25.5.21 on_pointer_clicked() [1/2]

```
void on_pointer_clicked (
    UIButton *self  __attribute__((unused)) )
```

4.25.5.22 on_pointer_clicked() [2/2]

```
void on_pointer_clicked (
    UIButton * self )
```

4.25.5.23 on_save_button_clicked()

```
void on_save_button_clicked (
    UIButton * self )
```

4.25.5.24 on_tangent_clicked() [1/2]

```
void on_tangent_clicked (
    UIButton *self  __attribute__((unused)) )
```

4.25.5.25 on_tangent_clicked() [2/2]

```
void on_tangent_clicked (
    UIButton * self )
```

4.25.6 Variable Documentation**4.25.6.1 cs**

```
CoordinateSystem* cs
```

4.25.6.2 state

```
State state = STATE_POINTER
```

4.26 src/renderer/renderer.c File Reference

Functions

- void `renderer_set_default_font` (`Font *font`)
- void `renderer_set_clip_rect` (`int x`, `int y`, `int width`, `int height`)
- `Texture *` `renderer_create_framebuffer` (`int width`, `int height`)
- void `renderer_resize_framebuffer` (`Texture *framebuffer`, `int width`, `int height`)
- void `renderer_bind_framebuffer` (`Texture *framebuffer`)
- void `renderer_clear` (`Color color`)
- void `renderer_draw_pixel` (`int x`, `int y`, `Color color`)
- void `renderer_draw_line` (`int x1`, `int y1`, `int x2`, `int y2`, `int thickness`, `Color color`)
- void `renderer_draw_rect` (`int x`, `int y`, `int width`, `int height`, `Color color`)
- void `renderer_draw_filled_rect` (`int x`, `int y`, `int width`, `int height`, `Color color`)
- void `renderer_draw_circle` (`int x`, `int y`, `int radius`, `Color color`)
- void `renderer_draw_filled_circle` (`int x`, `int y`, `int radius`, `Color color`)
- void `renderer_draw_ellipse` (`int x`, `int y`, `int rx`, `int ry`, `Color color`)
- void `renderer_draw_filled_ellipse` (`int x`, `int y`, `int rx`, `int ry`, `Color color`)
- void `renderer_draw_triangle` (`int x1`, `int y1`, `int x2`, `int y2`, `int x3`, `int y3`, `Color color`)
- void `renderer_draw_filled_triangle` (`int x1`, `int y1`, `int x2`, `int y2`, `int x3`, `int y3`, `Color color`)
- void `renderer_draw_rounded_rect` (`int x`, `int y`, `int width`, `int height`, `int radius`, `Color color`)
- void `renderer_draw_filled_rounded_rect` (`int x`, `int y`, `int width`, `int height`, `int radius`, `Color color`)
- void `renderer_draw_polygon` (`const short *vx`, `const short *vy`, `int n`, `Color color`)
- void `renderer_draw_filled_polygon` (`const short *vx`, `const short *vy`, `int n`, `Color color`)
- void `renderer_draw_arc` (`int x`, `int y`, `int radius`, `int start`, `int end`, `Color color`)
- void `renderer_draw_pie` (`int x`, `int y`, `int radius`, `int start`, `int end`, `Color color`)
- void `renderer_draw_bezier` (`const short *vx`, `const short *vy`, `int n`, `int s`, `Color color`)
- void `renderer_draw_texture` (`Texture *texture`, `int x`, `int y`, `int width`, `int height`)
- void `renderer_draw_text` (`const char *text`, `int x`, `int y`, `Color color`)
- `SDL_Point` `renderer_query_text_size` (`const char *text`)
- void `_renderer_set_target` (`SDL_Renderer *renderer`)

4.26.1 Function Documentation

4.26.1.1 _renderer_set_target()

```
void _renderer_set_target (
    SDL_Renderer * renderer )
```

Sets the target renderer for the application (should not be called manually)

Parameters

<i>renderer</i>	The renderer to set as target
-----------------	-------------------------------

4.26.1.2 renderer_bind_framebuffer()

```
void renderer_bind_framebuffer (
    Texture * framebuffer )
```

Binds a framebuffer.

Parameters

<i>framebuffer</i>	The framebuffer to destroy
--------------------	----------------------------

4.26.1.3 `renderer_clear()`

```
void renderer_clear (
    Color color )
```

Clears the screen with a color.

Parameters

<i>color</i>	The color to clear the screen with
--------------	------------------------------------

4.26.1.4 `renderer_create_framebuffer()`

```
Texture * renderer_create_framebuffer (
    int width,
    int height )
```

Creates a new framebuffer.

Parameters

<i>width</i>	The width of the framebuffer
<i>height</i>	The height of the framebuffer

Returns

Texture* Returns the framebuffer

4.26.1.5 `renderer_draw_arc()`

```
void renderer_draw_arc (
    int x,
    int y,
    int radius,
    int start,
    int end,
    Color color )
```

Draws an arc (not filled)

Parameters

<i>x</i>	The x coordinate of the arc
<i>y</i>	The y coordinate of the arc

Parameters

<i>radius</i>	The radius of the arc
<i>start</i>	The start angle of the arc
<i>end</i>	The end angle of the arc
<i>color</i>	The color of the arc

4.26.1.6 `renderer_draw_bezier()`

```
void renderer_draw_bezier (
    const short * vx,
    const short * vy,
    int n,
    int s,
    Color color )
```

Draws a bezier curve.

Parameters

<i>vx</i>	The x coordinates of the points
<i>vy</i>	The y coordinates of the points
<i>n</i>	The number of points
<i>s</i>	The number of segments
<i>color</i>	The color of the bezier curve

4.26.1.7 `renderer_draw_circle()`

```
void renderer_draw_circle (
    int x,
    int y,
    int radius,
    Color color )
```

Draws a circle (not filled)

Parameters

<i>x</i>	The x coordinate of the circle
<i>y</i>	The y coordinate of the circle
<i>radius</i>	The radius of the circle
<i>color</i>	The color of the circle

4.26.1.8 `renderer_draw_ellipse()`

```
void renderer_draw_ellipse (
    int x,
```

```
int y,  
int rx,  
int ry,  
Color color )
```

Draws an ellipse (not filled, axis aligned)

Parameters

<i>x</i>	The x coordinate of the ellipse
<i>y</i>	The y coordinate of the ellipse
<i>rx</i>	The x radius of the ellipse
<i>ry</i>	The y radius of the ellipse
<i>color</i>	The color of the ellipse

4.26.1.9 renderer_draw_filled_circle()

```
void renderer_draw_filled_circle (  
    int x,  
    int y,  
    int radius,  
    Color color )
```

Draws a filled circle.

Parameters

<i>x</i>	The x coordinate of the circle
<i>y</i>	The y coordinate of the circle
<i>radius</i>	The radius of the circle
<i>color</i>	The color of the circle

4.26.1.10 renderer_draw_filled_ellipse()

```
void renderer_draw_filled_ellipse (  
    int x,  
    int y,  
    int rx,  
    int ry,  
    Color color )
```

Draws a filled ellipse (axis aligned)

Parameters

<i>x</i>	The x coordinate of the ellipse
<i>y</i>	The y coordinate of the ellipse
<i>rx</i>	The x radius of the ellipse
<i>ry</i>	The y radius of the ellipse
<i>color</i>	The color of the ellipse

4.26.1.11 `renderer_draw_filled_polygon()`

```
void renderer_draw_filled_polygon (
    const short * vx,
    const short * vy,
    int n,
    Color color )
```

Draws a filled polygon.

Parameters

<i>vx</i>	The x coordinates of the points
<i>vy</i>	The y coordinates of the points
<i>n</i>	The number of points
<i>color</i>	The color of the polygon

4.26.1.12 `renderer_draw_filled_rect()`

```
void renderer_draw_filled_rect (
    int x,
    int y,
    int width,
    int height,
    Color color )
```

Draws a filled rectangle.

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>color</i>	The color of the rectangle

4.26.1.13 `renderer_draw_filled_rounded_rect()`

```
void renderer_draw_filled_rounded_rect (
    int x,
    int y,
    int width,
    int height,
    int radius,
    Color color )
```

Draws a filled rounded rectangle.

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>radius</i>	The radius of the corners
<i>color</i>	The color of the rectangle

4.26.1.14 `renderer_draw_filled_triangle()`

```
void renderer_draw_filled_triangle (
    int x1,
    int y1,
    int x2,
    int y2,
    int x3,
    int y3,
    Color color )
```

Draws a filled triangle.

Parameters

<i>x1</i>	The x coordinate of the first point
<i>y1</i>	The y coordinate of the first point
<i>x2</i>	The x coordinate of the second point
<i>y2</i>	The y coordinate of the second point
<i>x3</i>	The x coordinate of the third point
<i>y3</i>	The y coordinate of the third point
<i>color</i>	The color of the triangle

4.26.1.15 `renderer_draw_line()`

```
void renderer_draw_line (
    int x1,
    int y1,
    int x2,
    int y2,
    int thickness,
    Color color )
```

Draws a line.

Parameters

<i>x1</i>	The x coordinate of the first point
<i>y1</i>	The y coordinate of the first point
<i>x2</i>	The x coordinate of the second point

Parameters

<i>y2</i>	The y coordinate of the second point
<i>thickness</i>	The thickness of the line
<i>color</i>	The color of the line

4.26.1.16 `renderer_draw_pie()`

```
void renderer_draw_pie (
    int x,
    int y,
    int radius,
    int start,
    int end,
    Color color )
```

Draws a filled arc (aka pie)

Parameters

<i>x</i>	The x coordinate of the arc
<i>y</i>	The y coordinate of the arc
<i>radius</i>	The radius of the arc
<i>start</i>	The start angle of the arc
<i>end</i>	The end angle of the arc
<i>color</i>	The color of the arc

4.26.1.17 `renderer_draw_pixel()`

```
void renderer_draw_pixel (
    int x,
    int y,
    Color color )
```

Draws a pixel.

Parameters

<i>x</i>	The x coordinate of the pixel
<i>y</i>	The y coordinate of the pixel
<i>color</i>	The color of the pixel

4.26.1.18 `renderer_draw_polygon()`

```
void renderer_draw_polygon (
    const short * vx,
    const short * vy,
```



```
int n,  
Color color )
```

Draws a polygon (not filled)

Parameters

<i>vx</i>	The x coordinates of the points
<i>vy</i>	The y coordinates of the points
<i>n</i>	The number of points
<i>color</i>	The color of the polygon

4.26.1.19 renderer_draw_rect()

```
void renderer_draw_rect (  
    int x,  
    int y,  
    int width,  
    int height,  
    Color color )
```

Draws a rectangle (not filled)

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>color</i>	The color of the rectangle

4.26.1.20 renderer_draw_rounded_rect()

```
void renderer_draw_rounded_rect (  
    int x,  
    int y,  
    int width,  
    int height,  
    int radius,  
    Color color )
```

Draws a rounded rectangle (not filled)

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>radius</i>	The radius of the corners
<i>color</i>	The color of the rectangle

4.26.1.21 `renderer_draw_text()`

```
void renderer_draw_text (
    const char * text,
    int x,
    int y,
    Color color )
```

Draws a text.

Parameters

<i>text</i>	The text to draw
<i>x</i>	The x coordinate of the text
<i>y</i>	The y coordinate of the text
<i>color</i>	The color of the text

4.26.1.22 `renderer_draw_texture()`

```
void renderer_draw_texture (
    Texture * texture,
    int x,
    int y,
    int width,
    int height )
```

Draws a texture.

Parameters

<i>texture</i>	The texture to draw
<i>x</i>	The x coordinate of the texture
<i>y</i>	The y coordinate of the texture
<i>width</i>	The width of the texture
<i>height</i>	The height of the texture

4.26.1.23 `renderer_draw_triangle()`

```
void renderer_draw_triangle (
    int x1,
    int y1,
    int x2,
    int y2,
    int x3,
    int y3,
    Color color )
```

Draws a triangle (not filled)

Parameters

<i>x1</i>	The x coordinate of the first point
<i>y1</i>	The y coordinate of the first point
<i>x2</i>	The x coordinate of the second point
<i>y2</i>	The y coordinate of the second point
<i>x3</i>	The x coordinate of the third point
<i>y3</i>	The y coordinate of the third point
<i>color</i>	The color of the triangle

4.26.1.24 `renderer_query_text_size()`

```
SDL_Point renderer_query_text_size (
    const char * text )
```

Returns the size of a text.

Parameters

<i>text</i>	The text to get the size of
-------------	-----------------------------

Returns

SDL_Point The size of the text

4.26.1.25 `renderer_reset_clip_rect()`

```
void renderer_reset_clip_rect ( )
```

Resets the clip rect of the renderer.

4.26.1.26 `renderer_resize_framebuffer()`

```
void renderer_resize_framebuffer (
    Texture * framebuffer,
    int width,
    int height )
```

Resizes a framebuffer.

Parameters

<i>framebuffer</i>	The framebuffer to resize
<i>width</i>	The new width of the framebuffer
<i>height</i>	The new height of the framebuffer

4.26.1.27 `renderer_set_clip_rect()`

```
void renderer_set_clip_rect (
    int x,
    int y,
    int width,
    int height )
```

Sets a clip rect for the renderer.

Parameters

<i>x</i>	The x coordinate of the clip rect
<i>y</i>	The y coordinate of the clip rect
<i>width</i>	The width of the clip rect
<i>height</i>	The height of the clip rect

4.26.1.28 `renderer_set_default_font()`

```
void renderer_set_default_font (
    Font * font )
```

Sets the default font for the renderer.

Parameters

<i>font</i>	The font to set as default
-------------	----------------------------

4.27 `src/renderer/renderer.h` File Reference

Functions

- void `renderer_set_default_font` (`Font` *font)
- void `renderer_set_clip_rect` (int x, int y, int width, int height)
- `Texture` * `renderer_create_framebuffer` (int width, int height)
- void `renderer_resize_framebuffer` (`Texture` *framebuffer, int width, int height)
- void `renderer_bind_framebuffer` (`Texture` *framebuffer)
- void `renderer_clear` (`Color` color)
- void `renderer_draw_pixel` (int x, int y, `Color` color)
- void `renderer_draw_line` (int x1, int y1, int x2, int y2, int thickness, `Color` color)
- void `renderer_draw_rect` (int x, int y, int width, int height, `Color` color)
- void `renderer_draw_filled_rect` (int x, int y, int width, int height, `Color` color)
- void `renderer_draw_circle` (int x, int y, int radius, `Color` color)
- void `renderer_draw_filled_circle` (int x, int y, int radius, `Color` color)
- void `renderer_draw_ellipse` (int x, int y, int rx, int ry, `Color` color)
- void `renderer_draw_filled_ellipse` (int x, int y, int rx, int ry, `Color` color)
- void `renderer_draw_triangle` (int x1, int y1, int x2, int y2, int x3, int y3, `Color` color)
- void `renderer_draw_filled_triangle` (int x1, int y1, int x2, int y2, int x3, int y3, `Color` color)
- void `renderer_draw_rounded_rect` (int x, int y, int width, int height, int radius, `Color` color)

- void [renderer_draw_filled_rounded_rect](#) (int x, int y, int width, int height, int radius, [Color](#) color)
- void [renderer_draw_polygon](#) (const short *vx, const short *vy, int n, [Color](#) color)
- void [renderer_draw_filled_polygon](#) (const short *vx, const short *vy, int n, [Color](#) color)
- void [renderer_draw_arc](#) (int x, int y, int radius, int start, int end, [Color](#) color)
- void [renderer_draw_pie](#) (int x, int y, int radius, int start, int end, [Color](#) color)
- void [renderer_draw_bezier](#) (const short *vx, const short *vy, int n, int s, [Color](#) color)
- void [renderer_draw_texture](#) ([Texture](#) *texture, int x, int y, int width, int height)
- void [renderer_draw_text](#) (const char *text, int x, int y, [Color](#) color)
- [SDL_Point](#) [renderer_query_text_size](#) (const char *text)
- void [_renderer_set_target](#) ([SDL_Renderer](#) *renderer)

4.27.1 Function Documentation

4.27.1.1 [_renderer_set_target\(\)](#)

```
void _renderer_set_target (
    SDL\_Renderer * renderer )
```

Sets the target renderer for the application (should not be called manually)

Parameters

<i>renderer</i>	The renderer to set as target
-----------------	-------------------------------

4.27.1.2 [renderer_bind_framebuffer\(\)](#)

```
void renderer_bind_framebuffer (
    Texture * framebuffer )
```

Binds a framebuffer.

Parameters

<i>framebuffer</i>	The framebuffer to destroy
--------------------	----------------------------

4.27.1.3 [renderer_clear\(\)](#)

```
void renderer_clear (
    Color color )
```

Clears the screen with a color.

Parameters

<i>color</i>	The color to clear the screen with
--------------	------------------------------------

4.27.1.4 `renderer_create_framebuffer()`

```
Texture * renderer_create_framebuffer (
    int width,
    int height )
```

Creates a new framebuffer.

Parameters

<i>width</i>	The width of the framebuffer
<i>height</i>	The height of the framebuffer

Returns

Texture* Returns the framebuffer

4.27.1.5 `renderer_draw_arc()`

```
void renderer_draw_arc (
    int x,
    int y,
    int radius,
    int start,
    int end,
    Color color )
```

Draws an arc (not filled)

Parameters

<i>x</i>	The x coordinate of the arc
<i>y</i>	The y coordinate of the arc
<i>radius</i>	The radius of the arc
<i>start</i>	The start angle of the arc
<i>end</i>	The end angle of the arc
<i>color</i>	The color of the arc

4.27.1.6 `renderer_draw_bezier()`

```
void renderer_draw_bezier (
    const short * vx,
    const short * vy,
    int n,
    int s,
    Color color )
```

Draws a bezier curve.

Parameters

<i>vx</i>	The x coordinates of the points
<i>vy</i>	The y coordinates of the points
<i>n</i>	The number of points
<i>s</i>	The number of segments
<i>color</i>	The color of the bezier curve

4.27.1.7 `renderer_draw_circle()`

```
void renderer_draw_circle (
    int x,
    int y,
    int radius,
    Color color )
```

Draws a circle (not filled)

Parameters

<i>x</i>	The x coordinate of the circle
<i>y</i>	The y coordinate of the circle
<i>radius</i>	The radius of the circle
<i>color</i>	The color of the circle

4.27.1.8 `renderer_draw_ellipse()`

```
void renderer_draw_ellipse (
    int x,
    int y,
    int rx,
    int ry,
    Color color )
```

Draws an ellipse (not filled, axis aligned)

Parameters

<i>x</i>	The x coordinate of the ellipse
<i>y</i>	The y coordinate of the ellipse
<i>rx</i>	The x radius of the ellipse
<i>ry</i>	The y radius of the ellipse
<i>color</i>	The color of the ellipse

4.27.1.9 `renderer_draw_filled_circle()`

```
void renderer_draw_filled_circle (
    int x,
```

```
int y,  
int radius,  
Color color )
```

Draws a filled circle.

Parameters

<i>x</i>	The x coordinate of the circle
<i>y</i>	The y coordinate of the circle
<i>radius</i>	The radius of the circle
<i>color</i>	The color of the circle

4.27.1.10 `renderer_draw_filled_ellipse()`

```
void renderer_draw_filled_ellipse (  
    int x,  
    int y,  
    int rx,  
    int ry,  
    Color color )
```

Draws a filled ellipse (axis aligned)

Parameters

<i>x</i>	The x coordinate of the ellipse
<i>y</i>	The y coordinate of the ellipse
<i>rx</i>	The x radius of the ellipse
<i>ry</i>	The y radius of the ellipse
<i>color</i>	The color of the ellipse

4.27.1.11 `renderer_draw_filled_polygon()`

```
void renderer_draw_filled_polygon (  
    const short * vx,  
    const short * vy,  
    int n,  
    Color color )
```

Draws a filled polygon.

Parameters

<i>vx</i>	The x coordinates of the points
<i>vy</i>	The y coordinates of the points
<i>n</i>	The number of points
<i>color</i>	The color of the polygon

4.27.1.12 `renderer_draw_filled_rect()`

```
void renderer_draw_filled_rect (
    int x,
    int y,
    int width,
    int height,
    Color color )
```

Draws a filled rectangle.

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>color</i>	The color of the rectangle

4.27.1.13 `renderer_draw_filled_rounded_rect()`

```
void renderer_draw_filled_rounded_rect (
    int x,
    int y,
    int width,
    int height,
    int radius,
    Color color )
```

Draws a filled rounded rectangle.

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>radius</i>	The radius of the corners
<i>color</i>	The color of the rectangle

4.27.1.14 `renderer_draw_filled_triangle()`

```
void renderer_draw_filled_triangle (
    int x1,
    int y1,
    int x2,
    int y2,
    int x3,
    int y3,
    Color color )
```

Draws a filled triangle.

Parameters

<i>x1</i>	The x coordinate of the first point
<i>y1</i>	The y coordinate of the first point
<i>x2</i>	The x coordinate of the second point
<i>y2</i>	The y coordinate of the second point
<i>x3</i>	The x coordinate of the third point
<i>y3</i>	The y coordinate of the third point
<i>color</i>	The color of the triangle

4.27.1.15 `renderer_draw_line()`

```
void renderer_draw_line (
    int x1,
    int y1,
    int x2,
    int y2,
    int thickness,
    Color color )
```

Draws a line.

Parameters

<i>x1</i>	The x coordinate of the first point
<i>y1</i>	The y coordinate of the first point
<i>x2</i>	The x coordinate of the second point
<i>y2</i>	The y coordinate of the second point
<i>thickness</i>	The thickness of the line
<i>color</i>	The color of the line

4.27.1.16 `renderer_draw_pie()`

```
void renderer_draw_pie (
    int x,
    int y,
    int radius,
    int start,
    int end,
    Color color )
```

Draws a filled arc (aka pie)

Parameters

<i>x</i>	The x coordinate of the arc
<i>y</i>	The y coordinate of the arc
<i>radius</i>	The radius of the arc

Parameters

<i>start</i>	The start angle of the arc
<i>end</i>	The end angle of the arc
<i>color</i>	The color of the arc

4.27.1.17 `renderer_draw_pixel()`

```
void renderer_draw_pixel (
    int x,
    int y,
    Color color )
```

Draws a pixel.

Parameters

<i>x</i>	The x coordinate of the pixel
<i>y</i>	The y coordinate of the pixel
<i>color</i>	The color of the pixel

4.27.1.18 `renderer_draw_polygon()`

```
void renderer_draw_polygon (
    const short * vx,
    const short * vy,
    int n,
    Color color )
```

Draws a polygon (not filled)

Parameters

<i>vx</i>	The x coordinates of the points
<i>vy</i>	The y coordinates of the points
<i>n</i>	The number of points
<i>color</i>	The color of the polygon

4.27.1.19 `renderer_draw_rect()`

```
void renderer_draw_rect (
    int x,
    int y,
    int width,
    int height,
    Color color )
```

Draws a rectangle (not filled)

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>color</i>	The color of the rectangle

4.27.1.20 `renderer_draw_rounded_rect()`

```
void renderer_draw_rounded_rect (
    int x,
    int y,
    int width,
    int height,
    int radius,
    Color color )
```

Draws a rounded rectangle (not filled)

Parameters

<i>x</i>	The x coordinate of the rectangle
<i>y</i>	The y coordinate of the rectangle
<i>width</i>	The width of the rectangle
<i>height</i>	The height of the rectangle
<i>radius</i>	The radius of the corners
<i>color</i>	The color of the rectangle

4.27.1.21 `renderer_draw_text()`

```
void renderer_draw_text (
    const char * text,
    int x,
    int y,
    Color color )
```

Draws a text.

Parameters

<i>text</i>	The text to draw
<i>x</i>	The x coordinate of the text
<i>y</i>	The y coordinate of the text
<i>color</i>	The color of the text

4.27.1.22 `renderer_draw_texture()`

```
void renderer_draw_texture (
    Texture * texture,
    int x,
    int y,
    int width,
    int height )
```

Draws a texture.

Parameters

<i>texture</i>	The texture to draw
<i>x</i>	The x coordinate of the texture
<i>y</i>	The y coordinate of the texture
<i>width</i>	The width of the texture
<i>height</i>	The height of the texture

4.27.1.23 `renderer_draw_triangle()`

```
void renderer_draw_triangle (
    int x1,
    int y1,
    int x2,
    int y2,
    int x3,
    int y3,
    Color color )
```

Draws a triangle (not filled)

Parameters

<i>x1</i>	The x coordinate of the first point
<i>y1</i>	The y coordinate of the first point
<i>x2</i>	The x coordinate of the second point
<i>y2</i>	The y coordinate of the second point
<i>x3</i>	The x coordinate of the third point
<i>y3</i>	The y coordinate of the third point
<i>color</i>	The color of the triangle

4.27.1.24 `renderer_query_text_size()`

```
SDL_Point renderer_query_text_size (
    const char * text )
```

Returns the size of a text.

Parameters

<i>text</i>	The text to get the size of
-------------	-----------------------------

Returns

SDL_Point The size of the text

4.27.1.25 renderer_reset_clip_rect()

```
void renderer_reset_clip_rect ( )
```

Resets the clip rect of the renderer.

4.27.1.26 renderer_resize_framebuffer()

```
void renderer_resize_framebuffer (
    Texture * framebuffer,
    int width,
    int height )
```

Resizes a framebuffer.

Parameters

<i>framebuffer</i>	The framebuffer to resize
<i>width</i>	The new width of the framebuffer
<i>height</i>	The new height of the framebuffer

4.27.1.27 renderer_set_clip_rect()

```
void renderer_set_clip_rect (
    int x,
    int y,
    int width,
    int height )
```

Sets a clip rect for the renderer.

Parameters

<i>x</i>	The x coordinate of the clip rect
<i>y</i>	The y coordinate of the clip rect
<i>width</i>	The width of the clip rect
<i>height</i>	The height of the clip rect

4.27.1.28 `renderer_set_default_font()`

```
void renderer_set_default_font (
    Font * font )
```

Sets the default font for the renderer.

Parameters

<i>font</i>	The font to set as default
-------------	----------------------------

4.28 `renderer.h`

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #ifndef _WIN32
00004     #include <SDL.h>
00005     #include <SDL2_gfxPrimitives.h>
00006 #elif defined(__unix__) || defined(__linux__)
00007     #include <SDL2/SDL.h>
00008     #include <SDL2/SDL2_gfxPrimitives.h>
00009 #endif
00010
00011 #include "../color/color.h"
00012 #include "../font/font.h"
00013 #include "../texture/texture.h"
00014
00020 void renderer_set_default_font(Font* font);
00029 void renderer_set_clip_rect(int x, int y, int width, int height);
00033 void renderer_reset_clip_rect();
00034
00042 Texture* renderer_create_framebuffer(int width, int height);
00050 void renderer_resize_framebuffer(Texture* framebuffer, int width, int height);
00056 void renderer_bind_framebuffer(Texture* framebuffer);
00057
00063 void renderer_clear(Color color);
00071 void renderer_draw_pixel(int x, int y, Color color);
00082 void renderer_draw_line(int x1, int y1, int x2, int y2, int thickness, Color color);
00092 void renderer_draw_rect(int x, int y, int width, int height, Color color);
00102 void renderer_draw_filled_rect(int x, int y, int width, int height, Color color);
00111 void renderer_draw_circle(int x, int y, int radius, Color color);
00120 void renderer_draw_filled_circle(int x, int y, int radius, Color color);
00130 void renderer_draw_ellipse(int x, int y, int rx, int ry, Color color);
00140 void renderer_draw_filled_ellipse(int x, int y, int rx, int ry, Color color);
00152 void renderer_draw_triangle(int x1, int y1, int x2, int y2, int x3, int y3, Color color);
00164 void renderer_draw_filled_triangle(int x1, int y1, int x2, int y2, int x3, int y3, Color color);
00175 void renderer_draw_rounded_rect(int x, int y, int width, int height, int radius, Color color);
00186 void renderer_draw_filled_rounded_rect(int x, int y, int width, int height, int radius, Color color);
00195 void renderer_draw_polygon(const short* vx, const short* vy, int n, Color color);
00204 void renderer_draw_filled_polygon(const short* vx, const short* vy, int n, Color color);
00215 void renderer_draw_arc(int x, int y, int radius, int start, int end, Color color);
00226 void renderer_draw_pie(int x, int y, int radius, int start, int end, Color color);
00236 void renderer_draw_bezier(const short* vx, const short* vy, int n, int s, Color color);
00246 void renderer_draw_texture(Texture* texture, int x, int y, int width, int height);
00255 void renderer_draw_text(const char* text, int x, int y, Color color);
00262 SDL_Point renderer_query_text_size(const char* text);
00263
00269 void _renderer_set_target(SDL_Renderer* renderer);
```

4.29 `src/texture/texture.c` File Reference

Functions

- `Texture* texture_load` (`SDL_Renderer*renderer`, `const char*path`)
- `void _texture_add` (`Texture*texture`)

4.29.1 Function Documentation

4.29.1.1 `_texture_add()`

```
void _texture_add (
    Texture * texture )
```

Adds a texture to the texture vector (should not be called manually)

Parameters

<i>texture</i>	The texture to add
----------------	--------------------

4.29.1.2 `_texture_close()`

```
void _texture_close ( )
```

Destroys the textures and the texture vector (should not be called directly)

4.29.1.3 `_texture_init()`

```
void _texture_init ( )
```

Creates the texture vector that contains all the loaded textures (should not be called directly, it is needed for the `_texture_close` function)

4.29.1.4 `texture_load()`

```
Texture * texture_load (
    SDL_Renderer * renderer,
    const char * path )
```

Loads a texture from a file (freed automatically when the program closes)

Parameters

<i>renderer</i>	The renderer to load the texture with
<i>path</i>	The path to the file

Returns

Texture* Returns the loaded texture

4.30 `src/texture/texture.h` File Reference

Classes

- struct `Texture`

Functions

- `Texture * texture_load` (SDL_Renderer *renderer, const char *path)
- `void _texture_add` (Texture *texture)

4.30.1 Typedef Documentation

4.30.1.1 Texture

```
typedef struct Texture Texture
```

The texture struct that holds the texture data (SDL_Texture*, width, height)

4.30.2 Function Documentation

4.30.2.1 _texture_add()

```
void _texture_add (  
    Texture * texture )
```

Adds a texture to the texture vector (should not be called manually)

Parameters

<i>texture</i>	The texture to add
----------------	--------------------

4.30.2.2 _texture_close()

```
void _texture_close ( )
```

Destroys the textures and the texture vector (should not be called directly)

4.30.2.3 _texture_init()

```
void _texture_init ( )
```

Creates the texture vector that contains all the loaded textures (should not be called directly, it is needed for the _texture_close function)

4.30.2.4 texture_load()

```
Texture * texture_load (  
    SDL_Renderer * renderer,  
    const char * path )
```

Loads a texture from a file (freed automatically when the program closes)

Parameters

<i>renderer</i>	The renderer to load the texture with
<i>path</i>	The path to the file

Returns

Texture* Returns the loaded texture

4.31 texture.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #ifdef _WIN32
00004     #include <SDL.h>
00005     #include <SDL_image.h>
00006 #elif defined(__unix__) || defined(__linux__)
00007     #include <SDL2/SDL.h>
00008     #include <SDL2/SDL_image.h>
00009 #endif
00010
00014 typedef struct Texture
00015 {
00016     SDL_Texture* texture;
00017     int width;
00018     int height;
00019 } Texture;
00020
00028 Texture* texture_load(SDL_Renderer* renderer, const char* path);
00029
00033 void _texture_init();
00039 void _texture_add(Texture* texture);
00043 void _texture_close();

```

4.32 src/ui/ui.c File Reference

Functions

- void [_ui_init](#) (UIData *ui_data, int width, int height)
- void [_ui_handle_event](#) (UIData *ui_data, SDL_Event *event)
- void [_ui_update](#) (UIData *ui_data)
- void [_ui_render](#) (UIData *ui_data)
- void [_ui_close](#) (UIData *ui_data)
- void [_ui_set_target](#) (UIData *ui_data)
- UIData * [_ui_get_target](#) ()

4.32.1 Function Documentation

4.32.1.1 _ui_close()

```

void _ui_close (
    UIData * ui_data )

```

Closes the ui (destroys the main container)

Parameters

<i>ui_data</i>	The ui data to close
----------------	----------------------

4.32.1.2 `_ui_get_target()`

```
UIData * _ui_get_target ( )
```

Returns the target ui data for the window.

Returns

UIData* The target ui data

4.32.1.3 `_ui_handle_event()`

```
void _ui_handle_event (
    UIData * ui_data,
    SDL_Event * event )
```

Handles an event that influences the ui.

Parameters

<i>ui_data</i>	The ui data to change based on the event
<i>event</i>	The event to handle

4.32.1.4 `_ui_init()`

```
void _ui_init (
    UIData * ui_data,
    int width,
    int height )
```

Initializes the ui (and creates the main container)

Parameters

<i>ui_data</i>	The ui data to initialize
<i>width</i>	The width of the window
<i>height</i>	The height of the window

4.32.1.5 `_ui_render()`

```
void _ui_render (
    UIData * ui_data )
```

Renders the ui elements recursively.

Parameters

<i>ui_data</i>	The ui data to render
----------------	-----------------------

4.32.1.6 `_ui_set_target()`

```
void _ui_set_target (
    UIData * ui_data )
```

Sets the target ui data for the window.

Parameters

<i>ui_data</i>	The ui data to set as target
----------------	------------------------------

4.32.1.7 `_ui_update()`

```
void _ui_update (
    UIData * ui_data )
```

Updates the ui elements recursively.

Parameters

<i>ui_data</i>	The ui data to update
----------------	-----------------------

4.32.2 Variable Documentation

4.32.2.1 `target_ui_data`

```
UIData* target_ui_data = NULL
```

4.33 `src/ui/ui.h` File Reference

Classes

- struct [UIData](#)

Functions

- void [_ui_init](#) ([UIData](#) *ui_data, int width, int height)
- void [_ui_handle_event](#) ([UIData](#) *ui_data, [SDL_Event](#) *event)
- void [_ui_update](#) ([UIData](#) *ui_data)
- void [_ui_render](#) ([UIData](#) *ui_data)
- void [_ui_close](#) ([UIData](#) *ui_data)
- void [_ui_set_target](#) ([UIData](#) *ui_data)
- [UIData](#) * [_ui_get_target](#) ()

4.33.1 Typedef Documentation

4.33.1.1 UIData

```
typedef struct UIData UIData
```

Holds the ui data of a window (contains the main container, the text input of the current frame, whether the backspace was pressed, whether the mouse is captured by a ui element and the expanded splitbutton)

4.33.2 Function Documentation

4.33.2.1 _ui_close()

```
void _ui_close (
    UIData * ui_data )
```

Closes the ui (destroys the main container)

Parameters

<i>ui_data</i>	The ui data to close
----------------	----------------------

4.33.2.2 _ui_get_target()

```
UIData * _ui_get_target ( )
```

Returns the target ui data for the window.

Returns

UIData* The target ui data

4.33.2.3 _ui_handle_event()

```
void _ui_handle_event (
    UIData * ui_data,
    SDL_Event * event )
```

Handles an event that influences the ui.

Parameters

<i>ui_data</i>	The ui data to change based on the event
<i>event</i>	The event to handle

4.33.2.4 `_ui_init()`

```
void _ui_init (
    UIData * ui_data,
    int width,
    int height )
```

Initializes the ui (and creates the main container)

Parameters

<i>ui_data</i>	The ui data to initialize
<i>width</i>	The width of the window
<i>height</i>	The height of the window

4.33.2.5 `_ui_render()`

```
void _ui_render (
    UIData * ui_data )
```

Renders the ui elements recursively.

Parameters

<i>ui_data</i>	The ui data to render
----------------	-----------------------

4.33.2.6 `_ui_set_target()`

```
void _ui_set_target (
    UIData * ui_data )
```

Sets the target ui data for the window.

Parameters

<i>ui_data</i>	The ui data to set as target
----------------	------------------------------

4.33.2.7 `_ui_update()`

```
void _ui_update (
    UIData * ui_data )
```

Updates the ui elements recursively.

Parameters

<i>ui_data</i>	The ui data to update
----------------	-----------------------

4.34 ui.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "ui_element/ui_element.h"
00004
00008 typedef struct UIData
00009 {
00010     UIContainer* main_container;
00011     char text_input[SDL_TEXTINPUTEVENT_TEXT_SIZE];
00012     bool backspace_pressed;
00013     bool mouse_captured;
00014     UISplitButton* expanded_splitbutton;
00015 } UIData;
00016
00024 void _ui_init(UIData* ui_data, int width, int height);
00031 void _ui_handle_event(UIData* ui_data, SDL_Event* event);
00037 void _ui_update(UIData* ui_data);
00043 void _ui_render(UIData* ui_data);
00049 void _ui_close(UIData* ui_data);
00055 void _ui_set_target(UIData* ui_data);
00061 UIData* _ui_get_target();
```

4.35 src/ui/ui_constraint/ui_constraint.c File Reference

Functions

- [UIConstraint new_pixel_constraint](#) (int value)
- [UIConstraint new_center_constraint](#) ()
- [UIConstraint new_relative_constraint](#) (double value)
- [UIConstraint new_offset_constraint](#) (double value)
- [UIConstraint new_aspect_constraint](#) (double value)
- [UIConstraints constraints_from_string](#) (const char *string)

4.35.1 Function Documentation

4.35.1.1 constraints_from_string()

```
UIConstraints constraints_from_string (
    const char * string )
```

Creates a new set of constraints using a string (this is the recommended way to create constraints) Pixel constraints are represented by a number followed by "p" Center constraints are represented by "c" Relative constraints are represented by a number followed by "r" Offset constraints are represented by a number followed by "o" Aspect constraints are represented by a number followed by "a" Constraints are separated by spaces example: "100p c 0.5r 1.0a" or "10o -100p 0.5a 0.8r".

Parameters

<i>string</i>	The string containing the constraints
---------------	---------------------------------------

Returns

[UIConstraints](#) The new constraints

4.35.1.2 new_aspect_constraint()

```
UIConstraint new_aspect_constraint (
    double value )
```

Creates a new aspect constraint.

Parameters

<i>value</i>	The ratio of the constraint between 0 and 1
--------------	---

Returns

The new constraint

4.35.1.3 new_center_constraint()

```
UIConstraint new_center_constraint ( )
```

Creates a new center constraint.

Returns

[UIConstraint](#)

4.35.1.4 new_offset_constraint()

```
UIConstraint new_offset_constraint (
    double value )
```

Creates a new offset constraint.

Parameters

<i>value</i>	The value of the constraint in pixels
--------------	---------------------------------------

Returns

The new constraint

4.35.1.5 new_pixel_constraint()

```
UIConstraint new_pixel_constraint (
    int value )
```

Creates a new pixel constraint.

Parameters

<i>value</i>	The value of the constraint in pixels
--------------	---------------------------------------

Returns

The new constraint

4.35.1.6 new_relative_constraint()

```
UIConstraint new_relative_constraint (
    double value )
```

Creates a new relative constraint.

Parameters

<i>value</i>	The value of the constraint between 0 and 1
--------------	---

Returns

The new constraint

4.36 src/ui/ui_constraint/ui_constraint.h File Reference

Classes

- struct [UIConstraint](#)
- struct [UIConstraints](#)

Functions

- [UIConstraint new_pixel_constraint](#) (int value)
- [UIConstraint new_center_constraint](#) ()
- [UIConstraint new_relative_constraint](#) (double value)
- [UIConstraint new_offset_constraint](#) (double value)
- [UIConstraint new_aspect_constraint](#) (double value)
- [UIConstraints constraints_from_string](#) (const char *string)

4.36.1 Typedef Documentation

4.36.1.1 ConstraintType

```
typedef enum ConstraintType ConstraintType
```

The constraint types available for [UIConstraints](#).

4.36.1.2 NSLayoutConstraint

```
typedef struct NSLayoutConstraint NSLayoutConstraint
```

A constraint for a [UIElement](#) A constraint is a value that can be used to calculate the position or size of a [UIElement](#). A pixel constraint is a fixed value in pixels (can be negative to measure it from the right). A center constraint represents the center of the parent element. A relative constraint is a value between 0 and 1 that represents the percentage of the parent element. An offset constraint represents the offset from the parent element in pixels. An aspect constraint is a value that represents the aspect ratio of a [UIElement](#).

4.36.1.3 UIConstraints

```
typedef struct UIConstraints UIConstraints
```

A set of constraints for a [UIElement](#).

4.36.2 Enumeration Type Documentation

4.36.2.1 ConstraintType

```
enum ConstraintType
```

The constraint types available for [UIConstraints](#).

Enumerator

CT_PIXEL	
CT_CENTER	
CT_RELATIVE	
CT_OFFSET	
CT_ASPECT	

4.36.3 Function Documentation

4.36.3.1 constraints_from_string()

```
UIConstraints constraints_from_string (
    const char * string )
```

Creates a new set of constraints using a string (this is the recommended way to create constraints) Pixel constraints are represented by a number followed by "p" Center constraints are represented by "c" Relative constraints are represented by a number followed by "r" Offset constraints are represented by a number followed by "o" Aspect constraints are represented by a number followed by "a" Constraints are separated by spaces example: "100p c 0.5r 1.0a" or "10o -100p 0.5a 0.8r".

Parameters

<i>string</i>	The string containing the constraints
---------------	---------------------------------------

Returns

[UIConstraints](#) The new constraints

4.36.3.2 new_aspect_constraint()

```
UIConstraint new_aspect_constraint (
    double value )
```

Creates a new aspect constraint.

Parameters

<i>value</i>	The ratio of the constraint between 0 and 1
--------------	---

Returns

The new constraint

4.36.3.3 new_center_constraint()

```
UIConstraint new_center_constraint ( )
```

Creates a new center constraint.

Returns

[UIConstraint](#)

4.36.3.4 new_offset_constraint()

```
UIConstraint new_offset_constraint (
    double value )
```

Creates a new offset constraint.

Parameters

<i>value</i>	The value of the constraint in pixels
--------------	---------------------------------------

Returns

The new constraint

4.36.3.5 new_pixel_constraint()

```
UIConstraint new_pixel_constraint (
    int value )
```

Creates a new pixel constraint.

Parameters

<i>value</i>	The value of the constraint in pixels
--------------	---------------------------------------

Returns

The new constraint

4.36.3.6 new_relative_constraint()

```
UIConstraint new_relative_constraint (
    double value )
```

Creates a new relative constraint.

Parameters

<i>value</i>	The value of the constraint between 0 and 1
--------------	---

Returns

The new constraint

4.37 ui_constraint.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00006 typedef enum ConstraintType
00007 {
00008     CT_PIXEL = 0,
00009     CT_CENTER,
00010     CT_RELATIVE,
00011     CT_OFFSET,
00012     CT_ASPECT
00013 } ConstraintType;
00014
00015
00025 typedef struct UIConstraint
00026 {
00027     double value;
00028     ConstraintType constraint_type;
00029 } UIConstraint;
00030
00034 typedef struct UIConstraints
00035 {
00036     UIConstraint x, y, width, height;
00037 } UIConstraints;
00038
00044 UIConstraint new_pixel_constraint(int value);
00050 UIConstraint new_center_constraint();
00056 UIConstraint new_relative_constraint(double value);
00062 UIConstraint new_offset_constraint(double value);
00068 UIConstraint new_aspect_constraint(double value);
00069
00083 UIConstraints constraints_from_string(const char* string);
```

4.38 src/ui/ui_element/ui_element.c File Reference

Functions

- `UIContainer * ui_create_container (UIContainer *parent, UIConstraints constraints, void(*on_size_changed)(UIContainer *self, SDL_Point size))`
- `UIPanel * ui_create_panel (UIContainer *parent, UIConstraints constraints, Color color, Color border_color, Uint32 border_width, Uint32 roundness)`
- `UILabel * ui_create_label (UIContainer *parent, UIConstraints constraints, const char *text, Color color)`
- `UIButton * ui_create_button (UIContainer *parent, UIConstraints constraints, const char *text, Color color, Color text_color, void(*on_click)(UIButton *self))`
- `UIImageButton * ui_create_imagebutton (UIContainer *parent, UIConstraints constraints, Texture *texture, void(*on_click)(UIImageButton *self))`
- `UITextbox * ui_create_textbox (UIContainer *parent, UIConstraints constraints, const char *text, Color color, Color text_color, void(*on_text_changed)(UITextbox *self, const char *text))`
- `UICheckbox * ui_create_checkbox (UIContainer *parent, UIConstraints constraints, Color checked_color, Color unchecked_color, void(*on_checked_changed)(UICheckbox *self, bool checked))`
- `UISlider * ui_create_slider (UIContainer *parent, UIConstraints constraints, double value, Color color, Color slider_color, void(*on_value_changed)(UISlider *self, double value))`
- `UIDropdownList * ui_create_dropdown (UIContainer *parent, UIConstraints constraints, char *items, Color color, Color text_color, void(*on_selection_changed)(UIDropdownList *self, Sint32 index))`
- `UISplitButton * ui_create_splitbutton (UIContainer *parent, UIConstraints constraints, char *items, Color color, Color text_color, void(*on_item_clicked)(UISplitButton *self, Sint32 index), bool auto_dropdown)`
- `void ui_show_element (UIElement *self)`
- `void ui_hide_element (UIElement *self)`
- `void _ui_container_update (UIElement *self)`
- `void _ui_container_recalculate (UIElement *sibling, UIElement *self)`
- `void _ui_container_render (UIElement *self)`
- `void _ui_container_destroy (UIElement *self)`

4.38.1 Typedef Documentation

4.38.1.1 _UIDropdownItem

```
typedef struct _UIDropdownItem _UIDropdownItem
```

4.38.1.2 _UISplitButtonItem

```
typedef struct _UISplitButtonItem _UISplitButtonItem
```

4.38.2 Function Documentation

4.38.2.1 _ui_container_destroy()

```
void _ui_container_destroy (
    UIElement * self )
```

Destroys a `UIContainer` and its children recursively.

Parameters

<i>self</i>	The UIContainer to destroy
-------------	--

4.38.2.2 `_ui_container_recalculate()`

```
void _ui_container_recalculate (
    UIElement * sibling,
    UIElement * self )
```

Recalculates the position and size of a [UIContainer](#) and its children.

Parameters

<i>sibling</i>	The sibling of the UIContainer (used for calculating the position in case of offset constraints)
<i>self</i>	The UIContainer to recalculate

4.38.2.3 `_ui_container_render()`

```
void _ui_container_render (
    UIElement * self )
```

Renders a [UIContainer](#) and its children recursively.

Parameters

<i>self</i>	The UIContainer to render
-------------	---

4.38.2.4 `_ui_container_update()`

```
void _ui_container_update (
    UIElement * self )
```

Updates a [UIContainer](#) and its children recursively.

Parameters

<i>self</i>	The UIContainer to update
-------------	---

4.38.2.5 `ui_create_button()`

```
UIButton * ui_create_button (
    UIContainer * parent,
    UIConstraints constraints,
    const char * text,
```

```

    Color color,
    Color text_color,
    void(*) (UIButton *self) on_click )

```

Creates a [UIButton](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>text</i>	The text
<i>color</i>	The button color
<i>text_color</i>	The text color
<i>on_click</i>	The on click callback (can be NULL)

Returns

UIButton* The created button

4.38.2.6 ui_create_checkbox()

```

UIButton * ui_create_checkbox (
    UIContainer * parent,
    UIConstraints constraints,
    Color checked_color,
    Color unchecked_color,
    void(*) (UIButton *self, bool checked) on_checked_changed )

```

Creates a [UIButton](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>checked_color</i>	The checked color
<i>unchecked_color</i>	The unchecked color
<i>on_checked_changed</i>	The on checked changed callback (can be NULL)

Returns

UIButton* The created checkbox

4.38.2.7 ui_create_container()

```

UIContainer * ui_create_container (
    UIContainer * parent,
    UIConstraints constraints,
    void(*) (UIContainer *self, SDL_Point size) on_size_changed )

```

Creates a [UIContainer](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>on_size_changed</i>	The on size changed callback (can be NULL)

Returns

UIContainer* The created container

4.38.2.8 ui_create_dropdown()

```
UIDropdownList * ui_create_dropdown (
    UIContainer * parent,
    UIConstraints constraints,
    char * items,
    Color color,
    Color text_color,
    void(*) (UIDropdownList *self, Sint32 index) on_selection_changed )
```

Creates a [UIDropdownList](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>items</i>	The items (semicolon separated)
<i>color</i>	The color
<i>text_color</i>	The text color
<i>on_selection_changed</i>	The on selection changed callback (can be NULL)

Returns

UIDropdownList* The created dropdown list

4.38.2.9 ui_create_imagebutton()

```
UIImageButton * ui_create_imagebutton (
    UIContainer * parent,
    UIConstraints constraints,
    Texture * texture,
    void(*) (UIImageButton *self) on_click )
```

Creates a [UIImageButton](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>texture</i>	The texture
<i>on_click</i>	The on click callback (can be NULL)

Returns

UIButton* The created image button

4.38.2.10 ui_create_label()

```
UILabel * ui_create_label (
    UIContainer * parent,
    UIConstraints constraints,
    const char * text,
    Color color )
```

Creates a UILabel.

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>text</i>	The text
<i>color</i>	The text color

Returns

UILabel* The created label

4.38.2.11 ui_create_panel()

```
UIPanel * ui_create_panel (
    UIContainer * parent,
    UIConstraints constraints,
    Color color,
    Color border_color,
    UInt32 border_width,
    UInt32 roundness )
```

Creates a UIPanel.

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>color</i>	The panel color
<i>border_color</i>	The border color
<i>border_width</i>	The border width
<i>roundness</i>	The roundness

Returns

UIPanel* The created panel

4.38.2.12 ui_create_slider()

```
UISlider * ui_create_slider (
    UIContainer * parent,
    UIConstraints constraints,
    double value,
    Color color,
    Color slider_color,
    void(*) (UISlider *self, double value) on_value_changed )
```

Creates a [UISlider](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>value</i>	The value
<i>color</i>	The color of the horizontal bar
<i>slider_color</i>	The slider color (the vertical bar)
<i>on_value_changed</i>	The on value changed callback (can be NULL)

Returns

UISlider* The created slider

4.38.2.13 ui_create_splitbutton()

```
UISplitButton * ui_create_splitbutton (
    UIContainer * parent,
    UIConstraints constraints,
    char * items,
    Color color,
    Color text_color,
    void(*) (UISplitButton *self, Sint32 index) on_item_clicked,
    bool auto_dropdown )
```

Creates a [UISplitButton](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>items</i>	The items (semicolon separated)
<i>color</i>	The color
<i>text_color</i>	The text color
<i>on_item_clicked</i>	The on item clicked callback (can be NULL)
<i>auto_dropdown</i>	Whether to automatically dropdown the list when the button is clicked

Returns

UISplitButton* The created split button

4.38.2.14 ui_create_textbox()

```
UITextbox * ui_create_textbox (
    UIContainer * parent,
    UIConstraints constraints,
    const char * text,
    Color color,
    Color text_color,
    void(*) (UITextbox *self, const char *text) on_text_changed )
```

Creates a [UITextbox](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>text</i>	The text
<i>color</i>	The textbox color
<i>text_color</i>	The text color
<i>on_text_changed</i>	The on text changed callback (can be NULL)

Returns

UITextbox* The created textbox

4.38.2.15 ui_hide_element()

```
void ui_hide_element (
    UIElement * self )
```

Hides a UI element (sets the shown flag to false)

Parameters

<i>self</i>	The UI element to hide
-------------	------------------------

4.38.2.16 ui_show_element()

```
void ui_show_element (
    UIElement * self )
```

Shows a UI element (sets the shown flag to true)

Parameters

<i>self</i>	The UI element to show
-------------	------------------------

4.39 src/ui/ui_element/ui_element.h File Reference

Classes

- struct [UIElement](#)
- struct [UIContainer](#)
- struct [UIPanel](#)
- struct [UILabel](#)
- struct [UIButton](#)
- struct [UIImageButton](#)
- struct [UITextbox](#)
- struct [UICheckbox](#)
- struct [UISlider](#)
- struct [UIDropdownList](#)
- struct [UISplitButton](#)

Functions

- [UIContainer](#) * [ui_create_container](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, void(*on_size_↔changed)([UIContainer](#) *self, [SDL_Point](#) size))
- [UIPanel](#) * [ui_create_panel](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, [Color](#) color, [Color](#) border_color, [Uint32](#) border_width, [Uint32](#) roundness)
- [UILabel](#) * [ui_create_label](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, const char *text, [Color](#) color)
- [UIButton](#) * [ui_create_button](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, const char *text, [Color](#) color, [Color](#) text_color, void(*on_click)([UIButton](#) *self))
- [UIImageButton](#) * [ui_create_imagebutton](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, [Texture](#) *texture, void(*on_click)([UIImageButton](#) *self))
- [UITextbox](#) * [ui_create_textbox](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, const char *text, [Color](#) color, [Color](#) text_color, void(*on_text_changed)([UITextbox](#) *self, const char *text))
- [UICheckbox](#) * [ui_create_checkbox](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, [Color](#) checked_color, [Color](#) unchecked_color, void(*on_checked_changed)([UICheckbox](#) *self, bool checked))
- [UISlider](#) * [ui_create_slider](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, double value, [Color](#) color, [Color](#) slider_color, void(*on_value_changed)([UISlider](#) *self, double value))
- [UIDropdownList](#) * [ui_create_dropdown](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, char *items, [Color](#) color, [Color](#) text_color, void(*on_selection_changed)([UIDropdownList](#) *self, [Sint32](#) index))
- [UISplitButton](#) * [ui_create_splitbutton](#) ([UIContainer](#) *parent, [UIConstraints](#) constraints, char *items, [Color](#) color, [Color](#) text_color, void(*on_item_clicked)([UISplitButton](#) *self, [Sint32](#) index), bool auto_dropdown)
- void [ui_show_element](#) ([UIElement](#) *self)
- void [ui_hide_element](#) ([UIElement](#) *self)
- void [_ui_container_update](#) ([UIElement](#) *self)
- void [_ui_container_recalculate](#) ([UIElement](#) *sibling, [UIElement](#) *self)
- void [_ui_container_render](#) ([UIElement](#) *self)
- void [_ui_container_destroy](#) ([UIElement](#) *self)

4.39.1 Macro Definition Documentation

4.39.1.1 UITEXT_MAX_LENGTH

```
#define UITEXT_MAX_LENGTH 50
```

4.39.2 Typedef Documentation

4.39.2.1 MouseState

```
typedef enum MouseState MouseState
```

Mouse state enum, needed for UI Elements.

4.39.2.2 UIButton

```
typedef struct UIButton UIButton
```

The UI button structure.

4.39.2.3 UIButtonClick

```
typedef void(* UIButtonClick) (UIButton *self)
```

4.39.2.4 UICheckbox

```
typedef struct UICheckbox UICheckbox
```

The UI checkbox structure.

4.39.2.5 UICheckboxCheckedChanged

```
typedef void(* UICheckboxCheckedChanged) (UICheckbox *self, bool checked)
```

4.39.2.6 UIContainer

```
typedef struct UIContainer UIContainer
```

The UI container structure (provides a container for other UI elements)

4.39.2.7 UIContainerSizeChanged

```
typedef void(* UIContainerSizeChanged) (UIContainer *self, SDL_Point size)
```

4.39.2.8 UIDropdownList

```
typedef struct UIDropdownList UIDropdownList
```

The UI dropdown list structure.

4.39.2.9 UIDropdownListSelectionChanged

```
typedef void(* UIDropdownListSelectionChanged) (UIDropdownList *self, Sint32 index)
```

4.39.2.10 UIElement

```
typedef struct UIElement UIElement
```

The base UI element structure (needed for polymorphism)

4.39.2.11 UIElementDestroy

```
typedef void(* UIElementDestroy) (UIElement *self)
```

4.39.2.12 UIElementRecalculate

```
typedef void(* UIElementRecalculate) (UIElement *sibling, UIElement *self)
```

4.39.2.13 UIElementRender

```
typedef void(* UIElementRender) (UIElement *self)
```

4.39.2.14 UIElementUpdate

```
typedef void(* UIElementUpdate) (UIElement *self)
```

4.39.2.15 UIImageButton

```
typedef struct UIImageButton UIImageButton
```

The UI image button structure.

4.39.2.16 UIImageButtonClick

```
typedef void(* UIImageButtonClick) (UIImageButton *self)
```

4.39.2.17 UILabel

```
typedef struct UILabel UILabel
```

The UI label structure.

4.39.2.18 UIPanel

```
typedef struct UIPanel UIPanel
```

The UI panel structure (colored panel with border)

4.39.2.19 UISlider

```
typedef struct UISlider UISlider
```

The UI slider structure.

4.39.2.20 UISliderValueChanged

```
typedef void(* UISliderValueChanged) (UISlider *self, double value)
```

4.39.2.21 UISplitButton

```
typedef struct UISplitButton UISplitButton
```

The UI split button structure.

4.39.2.22 UISplitButtonClicked

```
typedef void(* UISplitButtonClicked) (UISplitButton *self, Sint32 index)
```

4.39.2.23 UITextbox

```
typedef struct UITextbox UITextbox
```

The UI textbox structure (has a fixed length)

4.39.2.24 UITextboxTextChanged

```
typedef void(* UITextboxTextChanged) (UITextbox *self, const char *text)
```

4.39.3 Enumeration Type Documentation

4.39.3.1 MouseState

```
enum MouseState
```

Mouse state enum, needed for UI Elements.

Enumerator

MS_NONE	
MS_HOVER	
MS_PRESS	

4.39.4 Function Documentation

4.39.4.1 _ui_container_destroy()

```
void _ui_container_destroy (
    UIElement * self )
```

Destroys a [UIContainer](#) and its children recursively.

Parameters

<i>self</i>	The UIContainer to destroy
-------------	--

4.39.4.2 _ui_container_recalculate()

```
void _ui_container_recalculate (
    UIElement * sibling,
    UIElement * self )
```

Recalculates the position and size of a [UIContainer](#) and its children.

Parameters

<i>sibling</i>	The sibling of the UIContainer (used for calculating the position in case of offset constraints)
<i>self</i>	The UIContainer to recalculate

4.39.4.3 _ui_container_render()

```
void _ui_container_render (
    UIElement * self )
```

Renders a [UIContainer](#) and its children recursively.

Parameters

<i>self</i>	The UIContainer to render
-------------	---

4.39.4.4 `_ui_container_update()`

```
void _ui_container_update (
    UIElement * self )
```

Updates a [UIContainer](#) and its children recursively.

Parameters

<i>self</i>	The UIContainer to update
-------------	---

4.39.4.5 `ui_create_button()`

```
UIButton * ui_create_button (
    UIContainer * parent,
    UIConstraints constraints,
    const char * text,
    Color color,
    Color text_color,
    void(*) (UIButton *self) on_click )
```

Creates a [UIButton](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>text</i>	The text
<i>color</i>	The button color
<i>text_color</i>	The text color
<i>on_click</i>	The on click callback (can be NULL)

Returns

UIButton* The created button

4.39.4.6 `ui_create_checkbox()`

```
UICheckbox * ui_create_checkbox (
    UIContainer * parent,
    UIConstraints constraints,
    Color checked_color,
    Color unchecked_color,
    void(*) (UICheckbox *self, bool checked) on_checked_changed )
```

Creates a [UICheckbox](#).

Parameters

<i>parent</i>	The parent container
---------------	----------------------

Parameters

<i>constraints</i>	The constraints
<i>checked_color</i>	The checked color
<i>unchecked_color</i>	The unchecked color
<i>on_checked_changed</i>	The on checked changed callback (can be NULL)

Returns

UICheckbox* The created checkbox

4.39.4.7 ui_create_container()

```
UIContainer * ui_create_container (
    UIContainer * parent,
    UIConstraints constraints,
    void(*) (UIContainer *self, SDL_Point size) on_size_changed )
```

Creates a [UIContainer](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>on_size_changed</i>	The on size changed callback (can be NULL)

Returns

UIContainer* The created container

4.39.4.8 ui_create_dropdown()

```
UIDropdownList * ui_create_dropdown (
    UIContainer * parent,
    UIConstraints constraints,
    char * items,
    Color color,
    Color text_color,
    void(*) (UIDropdownList *self, Sint32 index) on_selection_changed )
```

Creates a [UIDropdownList](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>items</i>	The items (semicolon separated)
<i>color</i>	The color
<i>text_color</i>	The text color
<i>on_selection_changed</i>	The on selection changed callback (can be NULL)

Returns

UIDropdownList* The created dropdown list

4.39.4.9 ui_create_imagebutton()

```
UIImageButton * ui_create_imagebutton (
    UIContainer * parent,
    UIConstraints constraints,
    Texture * texture,
    void(*) (UIImageButton *self) on_click )
```

Creates a UIImageButton.

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>texture</i>	The texture
<i>on_click</i>	The on click callback (can be NULL)

Returns

UIImageButton* The created image button

4.39.4.10 ui_create_label()

```
UILabel * ui_create_label (
    UIContainer * parent,
    UIConstraints constraints,
    const char * text,
    Color color )
```

Creates a UILabel.

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>text</i>	The text
<i>color</i>	The text color

Returns

UILabel* The created label

4.39.4.11 ui_create_panel()

```
UIPanel * ui_create_panel (
    UIContainer * parent,
```

```

    UIConstraints constraints,
    Color color,
    Color border_color,
    Uint32 border_width,
    Uint32 roundness )

```

Creates a [UIPanel](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>color</i>	The panel color
<i>border_color</i>	The border color
<i>border_width</i>	The border width
<i>roundness</i>	The roundness

Returns

UIPanel* The created panel

4.39.4.12 ui_create_slider()

```

UISlider * ui_create_slider (
    UIContainer * parent,
    UIConstraints constraints,
    double value,
    Color color,
    Color slider_color,
    void(*) (UISlider *self, double value) on_value_changed )

```

Creates a [UISlider](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>value</i>	The value
<i>color</i>	The color of the horizontal bar
<i>slider_color</i>	The slider color (the vertical bar)
<i>on_value_changed</i>	The on value changed callback (can be NULL)

Returns

UISlider* The created slider

4.39.4.13 ui_create_splitbutton()

```

UISplitButton * ui_create_splitbutton (
    UIContainer * parent,

```

```

    UIConstraints constraints,
    char * items,
    Color color,
    Color text_color,
    void(*) (UISplitButton *self, Sint32 index) on_item_clicked,
    bool auto_dropdown )

```

Creates a [UISplitButton](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>items</i>	The items (semicolon separated)
<i>color</i>	The color
<i>text_color</i>	The text color
<i>on_item_clicked</i>	The on item clicked callback (can be NULL)
<i>auto_dropdown</i>	Whether to automatically dropdown the list when the button is clicked

Returns

UISplitButton* The created split button

4.39.4.14 ui_create_textbox()

```

UITextbox * ui_create_textbox (
    UIContainer * parent,
    UIConstraints constraints,
    const char * text,
    Color color,
    Color text_color,
    void(*) (UITextbox *self, const char *text) on_text_changed )

```

Creates a [UITextbox](#).

Parameters

<i>parent</i>	The parent container
<i>constraints</i>	The constraints
<i>text</i>	The text
<i>color</i>	The textbox color
<i>text_color</i>	The text color
<i>on_text_changed</i>	The on text changed callback (can be NULL)

Returns

UITextbox* The created textbox

4.39.4.15 ui_hide_element()

```

void ui_hide_element (
    UIElement * self )

```

Hides a UI element (sets the shown flag to false)

Parameters

<i>self</i>	The UI element to hide
-------------	------------------------

4.39.4.16 ui_show_element()

```
void ui_show_element (
    UIElement * self )
```

Shows a UI element (sets the shown flag to true)

Parameters

<i>self</i>	The UI element to show
-------------	------------------------

4.40 ui_element.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #ifdef _WIN32
00004     #include <SDL.h>
00005 #elif defined(__unix__) || defined(__linux__)
00006     #include <SDL2/SDL.h>
00007 #endif
00008
00009 #include "../ui_constraint/ui_constraint.h"
00010 #include "../utils/vector/vector.h"
00011 #include "../color/color.h"
00012 #include "../texture/texture.h"
00013
00014 #include <stdbool.h>
00015
00016 #define UITEXT_MAX_LENGTH 50
00017
00018 typedef struct UIElement UIElement;
00019 typedef void (*UIElementUpdate)(UIElement* self);
00020 typedef void (*UIElementRecalculate)(UIElement* sibling, UIElement* self);
00021 typedef void (*UIElementRender)(UIElement* self);
00022 typedef void (*UIElementDestroy)(UIElement* self);
00023
00027 typedef struct UIElement
00028 {
00029     UIElement* parent;
00030     UIConstraints constraints;
00031     SDL_Point position;
00032     SDL_Point size;
00033     bool shown;
00034
00035     UIElementUpdate update;
00036     UIElementRecalculate recalculate;
00037     UIElementRender render;
00038     UIElementDestroy destroy;
00039 } UIElement;
00040
00041 typedef struct UIContainer UIContainer;
00042 typedef void (*UIContainerSizeChanged)(UIContainer* self, SDL_Point size);
00043
00047 typedef struct UIContainer
00048 {
00049     UIElement base;
00050
00051     Vector* children;
00052     UIContainerSizeChanged on_size_changed;
00053 } UIContainer;
```

```

00054
00058 typedef struct UIPanel
00059 {
00060     UIElement base;
00061
00062     Color color;
00063     Color border_color;
00064     Uint32 border_width;
00065     Uint32 corner_radius;
00066 } UIPanel;
00067
00071 typedef struct UILabel
00072 {
00073     UIElement base;
00074
00075     char text[UITEXT_MAX_LENGTH + 1];
00076     Color color;
00077 } UILabel;
00078
00082 typedef enum MouseState { MS_NONE = 0, MS_HOVER, MS_PRESS } MouseState;
00083 typedef struct UIButton UIButton;
00084 typedef void (*UIButtonClick)(UIButton* self);
00085
00089 typedef struct UIButton
00090 {
00091     UIElement base;
00092
00093     char text[UITEXT_MAX_LENGTH + 1];
00094     SDL_Point text_position;
00095     Color color;
00096     Color text_color;
00097     Uint32 corner_radius;
00098     MouseState mouse_state;
00099     UIButtonClick on_click;
00100 } UIButton;
00101
00102 typedef struct UIImageButton UIImageButton;
00103 typedef void (*UIImageButtonClick)(UIImageButton* self);
00104
00108 typedef struct UIImageButton
00109 {
00110     UIElement base;
00111
00112     Texture* texture;
00113     MouseState mouse_state;
00114     UIImageButtonClick on_click;
00115 } UIImageButton;
00116
00117 typedef struct UITextbox UITextbox;
00118 typedef void (*UITextboxTextChanged)(UITextbox* self, const char* text);
00119
00123 typedef struct UITextbox
00124 {
00125     UIElement base;
00126
00127     char text[UITEXT_MAX_LENGTH + 1];
00128     Color color;
00129     Color text_color;
00130     Uint32 corner_radius;
00131     bool focused;
00132     MouseState mouse_state;
00133     UITextboxTextChanged on_text_changed;
00134 } UITextbox;
00135
00136 typedef struct UICheckbox UICheckbox;
00137 typedef void (*UICheckboxCheckedChanged)(UICheckbox* self, bool checked);
00138
00142 typedef struct UICheckbox
00143 {
00144     UIElement base;
00145
00146     bool checked;
00147     Color checked_color;
00148     Color unchecked_color;
00149     Uint32 corner_radius;
00150     MouseState mouse_state;
00151     UICheckboxCheckedChanged on_checked_changed;
00152 } UICheckbox;
00153
00154 typedef struct UISlider UISlider;
00155 typedef void (*UISliderValueChanged)(UISlider* self, double value);
00156
00160 typedef struct UISlider
00161 {
00162     UIElement base;
00163
00164     double value;

```

```

00165     Color color;
00166     Color slider_color;
00167     Uint32 thickness;
00168     Uint32 corner_radius;
00169     MouseState mouse_state;
00170     UISliderValueChanged on_value_changed;
00171 } UISlider;
00172
00173 typedef struct UIDropdownList UIDropdownList;
00174 typedef void (*UIDropdownListSelectionChanged) (UIDropdownList* self, Sint32 index);
00175
00176 typedef struct UIDropdownList
00177 {
00178     UIElement base;
00179
00180     Vector* items;
00181     Uint32 selected_item;
00182     bool expanded;
00183     Color color;
00184     Color text_color;
00185     Uint32 corner_radius;
00186     UIDropdownListSelectionChanged on_selection_changed;
00187 } UIDropdownList;
00188
00189 typedef struct UISplitButton UISplitButton;
00190 typedef void (*UISplitButtonClicked) (UISplitButton* self, Sint32 index);
00191
00192 typedef struct UISplitButton
00193 {
00194     UIElement base;
00195
00196     Vector* items;
00197     bool expanded;
00198     Color color;
00199     Color text_color;
00200     Uint32 corner_radius;
00201     UISplitButtonClicked on_item_clicked;
00202     bool auto_dropdown;
00203 } UISplitButton;
00204
00205 UIContainer* ui_create_container(UIContainer* parent, UIConstraints constraints, void
(*on_size_changed) (UIContainer* self, SDL_Point size));
00206 UIPanel* ui_create_panel(UIContainer* parent, UIConstraints constraints, Color color, Color
border_color, Uint32 border_width, Uint32 roundness);
00207 UILabel* ui_create_label(UIContainer* parent, UIConstraints constraints, const char* text, Color
color);
00208 UIButton* ui_create_button(UIContainer* parent, UIConstraints constraints, const char* text, Color
color, Color text_color, void (*on_click) (UIButton* self));
00209 UIImageButton* ui_create_imagebutton(UIContainer* parent, UIConstraints constraints, Texture* texture,
void (*on_click) (UIImageButton* self));
00210 UITextbox* ui_create_textbox(UIContainer* parent, UIConstraints constraints, const char* text, Color
color, Color text_color, void (*on_text_changed) (UITextbox* self, const char* text));
00211 UICheckbox* ui_create_checkbox(UIContainer* parent, UIConstraints constraints, Color checked_color,
Color unchecked_color, void (*on_checked_changed) (UICheckbox* self, bool checked));
00212 UISlider* ui_create_slider(UIContainer* parent, UIConstraints constraints, double value, Color color,
Color slider_color, void (*on_value_changed) (UISlider* self, double value));
00213 UIDropdownList* ui_create_dropdown(UIContainer* parent, UIConstraints constraints, char* items, Color
color, Color text_color, void (*on_selection_changed) (UIDropdownList* self, Sint32 index));
00214 UISplitButton* ui_create_splitbutton(UIContainer* parent, UIConstraints constraints, char* items,
Color color, Color text_color, void (*on_item_clicked) (UISplitButton* self, Sint32 index), bool
auto_dropdown);
00215
00216 void ui_show_element(UIElement* self);
00217 void ui_hide_element(UIElement* self);
00218
00219 void _ui_container_update(UIElement* self);
00220 void _ui_container_recalculate(UIElement* sibling, UIElement* self);
00221 void _ui_container_render(UIElement* self);
00222 void _ui_container_destroy(UIElement* self);

```

4.41 src/utils/math/math.c File Reference

Functions

- double [deg_to_rad](#) (double deg)
- double [rad_to_deg](#) (double rad)
- double [clamp](#) (double value, double min, double max)
- double [lerp](#) (double a, double b, double t)
- double [map](#) (double x, double min1, double max1, double min2, double max2)
- bool [check_collision_point_rect](#) (int px, int py, int rx, int ry, int rw, int rh)

4.41.1 Function Documentation

4.41.1.1 check_collision_point_rect()

```
bool check_collision_point_rect (
    int px,
    int py,
    int rx,
    int ry,
    int rw,
    int rh )
```

Checks if a point is inside a rectangle.

Parameters

<i>px</i>	The x coordinate of the point
<i>py</i>	The y coordinate of the point
<i>rx</i>	The x coordinate of the rectangle
<i>ry</i>	The y coordinate of the rectangle
<i>rw</i>	The width of the rectangle
<i>rh</i>	The height of the rectangle

Returns

true If the point is inside the rectangle
false If the point is outside the rectangle

4.41.1.2 clamp()

```
double clamp (
    double x,
    double min,
    double max )
```

Clamps a value between a minimum and maximum value.

Parameters

<i>x</i>	The value to clamp
<i>min</i>	The minimum value
<i>max</i>	The maximum value

Returns

double The clamped value

4.41.1.3 deg_to_rad()

```
double deg_to_rad (
```

```
double deg )
```

Converts degrees to radians.

Parameters

<i>deg</i>	The angle in degrees
------------	----------------------

Returns

double The angle in radians

4.41.1.4 lerp()

```
double lerp (  
    double a,  
    double b,  
    double t )
```

Linearly interpolates between two values.

Parameters

<i>a</i>	The first value
<i>b</i>	The second value
<i>t</i>	The interpolation value

Returns

double The interpolated value

4.41.1.5 map()

```
double map (  
    double x,  
    double min1,  
    double max1,  
    double min2,  
    double max2 )
```

Maps a value from one range to another.

Parameters

<i>x</i>	The value to map
<i>min1</i>	The minimum value of the first range
<i>max1</i>	The maximum value of the first range
<i>min2</i>	The minimum value of the second range
<i>max2</i>	The maximum value of the second range

Returns

double The mapped value

4.41.1.6 rad_to_deg()

```
double rad_to_deg (
    double rad )
```

Converts radians to degrees.

Parameters

<i>rad</i>	The angle in radians
------------	----------------------

Returns

double The angle in degrees

4.42 src/utils/math/math.h File Reference**Functions**

- double [deg_to_rad](#) (double deg)
- double [rad_to_deg](#) (double rad)
- double [clamp](#) (double x, double min, double max)
- double [lerp](#) (double a, double b, double t)
- double [map](#) (double x, double min1, double max1, double min2, double max2)
- bool [check_collision_point_rect](#) (int px, int py, int rx, int ry, int rw, int rh)

4.42.1 Macro Definition Documentation**4.42.1.1 HALF_PI**

```
#define HALF_PI 1.57079632679489661923
```

4.42.1.2 PI

```
#define PI 3.14159265358979323846
```

4.42.1.3 TWO_PI

```
#define TWO_PI 6.28318530717958647692
```

4.42.2 Function Documentation

4.42.2.1 `check_collision_point_rect()`

```
bool check_collision_point_rect (
    int px,
    int py,
    int rx,
    int ry,
    int rw,
    int rh )
```

Checks if a point is inside a rectangle.

Parameters

<i>px</i>	The x coordinate of the point
<i>py</i>	The y coordinate of the point
<i>rx</i>	The x coordinate of the rectangle
<i>ry</i>	The y coordinate of the rectangle
<i>rw</i>	The width of the rectangle
<i>rh</i>	The height of the rectangle

Returns

true If the point is inside the rectangle
false If the point is outside the rectangle

4.42.2.2 `clamp()`

```
double clamp (
    double x,
    double min,
    double max )
```

Clamps a value between a minimum and maximum value.

Parameters

<i>x</i>	The value to clamp
<i>min</i>	The minimum value
<i>max</i>	The maximum value

Returns

double The clamped value

4.42.2.3 `deg_to_rad()`

```
double deg_to_rad (
```

```
double deg )
```

Converts degrees to radians.

Parameters

<i>deg</i>	The angle in degrees
------------	----------------------

Returns

double The angle in radians

4.42.2.4 lerp()

```
double lerp (  
    double a,  
    double b,  
    double t )
```

Linearly interpolates between two values.

Parameters

<i>a</i>	The first value
<i>b</i>	The second value
<i>t</i>	The interpolation value

Returns

double The interpolated value

4.42.2.5 map()

```
double map (  
    double x,  
    double min1,  
    double max1,  
    double min2,  
    double max2 )
```

Maps a value from one range to another.

Parameters

<i>x</i>	The value to map
<i>min1</i>	The minimum value of the first range
<i>max1</i>	The maximum value of the first range
<i>min2</i>	The minimum value of the second range
<i>max2</i>	The maximum value of the second range

Returns

double The mapped value

4.42.2.6 rad_to_deg()

```
double rad_to_deg (
    double rad )
```

Converts radians to degrees.

Parameters

<i>rad</i>	The angle in radians
------------	----------------------

Returns

double The angle in degrees

4.43 math.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdbool.h>
00004
00005 #define PI 3.14159265358979323846
00006 #define TWO_PI 6.28318530717958647692
00007 #define HALF_PI 1.57079632679489661923
00008
00015 double deg_to_rad(double deg);
00022 double rad_to_deg(double rad);
00031 double clamp(double x, double min, double max);
00040 double lerp(double a, double b, double t);
00051 double map(double x, double min1, double max1, double min2, double max2);
00052
00065 bool check_collision_point_rect(int px, int py, int rx, int ry, int rw, int rh);
```

4.44 src/utils/vector/vector.c File Reference**Functions**

- [Vector](#) * [vector_create](#) (size_t capacity)
- void [vector_destroy](#) ([Vector](#) *vector)
- void * [vector_get](#) ([Vector](#) *vector, size_t idx)
- void [vector_set](#) ([Vector](#) *vector, size_t idx, void *value)
- void [vector_push_back](#) ([Vector](#) *vector, void *value)
- void * [vector_pop_back](#) ([Vector](#) *vector)
- void [vector_insert](#) ([Vector](#) *vector, size_t idx, void *value)
- bool [vector_contains](#) ([Vector](#) *vector, void *value)
- int [vector_index_of](#) ([Vector](#) *vector, void *value)
- void [vector_remove_at](#) ([Vector](#) *vector, size_t idx)
- void [vector_remove](#) ([Vector](#) *vector, void *value)
- void [vector_reserve](#) ([Vector](#) *vector, size_t capacity)
- size_t [vector_size](#) ([Vector](#) *vector)
- void [vector_clear](#) ([Vector](#) *vector)

4.44.1 Function Documentation

4.44.1.1 vector_clear()

```
void vector_clear (
    Vector * vector )
```

Clears the vector.

Parameters

<i>vector</i>	The vector to clear
---------------	---------------------

4.44.1.2 vector_contains()

```
bool vector_contains (
    Vector * vector,
    void * value )
```

Checks if the vector contains a value.

Parameters

<i>vector</i>	The vector to check
<i>value</i>	The value to check for

Returns

true If the vector contains the value
false If the vector does not contain the value

4.44.1.3 vector_create()

```
Vector * vector_create (
    size_t capacity )
```

Creates a new vector.

Parameters

<i>capacity</i>	The initial capacity of the vector
-----------------	------------------------------------

Returns

Vector* The created vector (must be freed with vector_destroy)

4.44.1.4 vector_destroy()

```
void vector_destroy (
    Vector * vector )
```

Destroys a vector.

Parameters

<i>vector</i>	The vector to destroy
---------------	-----------------------

4.44.1.5 vector_get()

```
void * vector_get (
    Vector * vector,
    size_t idx )
```

Gets the value at the specified index.

Parameters

<i>vector</i>	The vector to get the value from
<i>idx</i>	The index of the element to get

Returns

void* The value at the specified index

4.44.1.6 vector_index_of()

```
int vector_index_of (
    Vector * vector,
    void * value )
```

Returns the index of a value in the vector.

Parameters

<i>vector</i>	The vector to search
<i>value</i>	The value to search for

Returns

int The index of the value in the vector, or -1 if not found

4.44.1.7 vector_insert()

```
void vector_insert (
    Vector * vector,
```



```
size_t idx,  
void * value )
```

Inserts a value at the specified index.

Parameters

<i>vector</i>	The vector to insert into
<i>idx</i>	The index to insert at
<i>value</i>	The value to insert

4.44.1.8 vector_pop_back()

```
void * vector_pop_back (   
    Vector * vector )
```

Pops a value from the back of the vector.

Parameters

<i>vector</i>	The vector to pop from
---------------	------------------------

Returns

void* The popped value

4.44.1.9 vector_push_back()

```
void vector_push_back (   
    Vector * vector,   
    void * value )
```

Pushes a value to the back of the vector.

Parameters

<i>vector</i>	The vector to push to
<i>value</i>	The value to push

4.44.1.10 vector_remove()

```
void vector_remove (   
    Vector * vector,   
    void * value )
```

Removes the first occurrence of a value from the vector.

Parameters

<i>vector</i>	The vector to remove from
<i>value</i>	The value to remove

4.44.1.11 vector_remove_at()

```
void vector_remove_at (
    Vector * vector,
    size_t idx )
```

Removes the value at the specified index.

Parameters

<i>vector</i>	The vector to remove from
<i>idx</i>	The index of the value to remove

4.44.1.12 vector_reserve()

```
void vector_reserve (
    Vector * vector,
    size_t capacity )
```

Reserves a new capacity for the vector.

Parameters

<i>vector</i>	The vector to reserve for
<i>size</i>	The new capacity of the vector

4.44.1.13 vector_set()

```
void vector_set (
    Vector * vector,
    size_t idx,
    void * value )
```

Sets the value at the specified index.

Parameters

<i>vector</i>	The vector to set the value in
<i>idx</i>	The index of the element to set
<i>value</i>	The value to set

4.44.1.14 vector_size()

```
size_t vector_size (
    Vector * vector )
```

Returns the size of the vector.

Parameters

<i>vector</i>	The vector to get the size of
---------------	-------------------------------

Returns

size_t The size of the vector

4.45 src/utils/vector/vector.h File Reference

Classes

- struct [Vector](#)

Functions

- [Vector](#) * [vector_create](#) (size_t capacity)
- void [vector_destroy](#) ([Vector](#) *vector)
- void * [vector_get](#) ([Vector](#) *vector, size_t idx)
- void [vector_set](#) ([Vector](#) *vector, size_t idx, void *value)
- void [vector_push_back](#) ([Vector](#) *vector, void *value)
- void * [vector_pop_back](#) ([Vector](#) *vector)
- void [vector_insert](#) ([Vector](#) *vector, size_t idx, void *value)
- bool [vector_contains](#) ([Vector](#) *vector, void *value)
- int [vector_index_of](#) ([Vector](#) *vector, void *value)
- void [vector_remove_at](#) ([Vector](#) *vector, size_t idx)
- void [vector_remove](#) ([Vector](#) *vector, void *value)
- void [vector_reserve](#) ([Vector](#) *vector, size_t capacity)
- size_t [vector_size](#) ([Vector](#) *vector)
- void [vector_clear](#) ([Vector](#) *vector)

4.45.1 Typedef Documentation

4.45.1.1 Vector

```
typedef struct Vector Vector
```

A generic vector type (dynamic array for void pointers)

4.45.2 Function Documentation

4.45.2.1 vector_clear()

```
void vector_clear (
    Vector * vector )
```

Clears the vector.

Parameters

<i>vector</i>	The vector to clear
---------------	---------------------

4.45.2.2 vector_contains()

```
bool vector_contains (
    Vector * vector,
    void * value )
```

Checks if the vector contains a value.

Parameters

<i>vector</i>	The vector to check
<i>value</i>	The value to check for

Returns

true If the vector contains the value
false If the vector does not contain the value

4.45.2.3 vector_create()

```
Vector * vector_create (
    size_t capacity )
```

Creates a new vector.

Parameters

<i>capacity</i>	The initial capacity of the vector
-----------------	------------------------------------

Returns

*Vector** The created vector (must be freed with `vector_destroy`)

4.45.2.4 vector_destroy()

```
void vector_destroy (
    Vector * vector )
```

Destroys a vector.

Parameters

<i>vector</i>	The vector to destroy
---------------	-----------------------

4.45.2.5 vector_get()

```
void * vector_get (
    Vector * vector,
    size_t idx )
```

Gets the value at the specified index.

Parameters

<i>vector</i>	The vector to get the value from
<i>idx</i>	The index of the element to get

Returns

void* The value at the specified index

4.45.2.6 vector_index_of()

```
int vector_index_of (
    Vector * vector,
    void * value )
```

Returns the index of a value in the vector.

Parameters

<i>vector</i>	The vector to search
<i>value</i>	The value to search for

Returns

int The index of the value in the vector, or -1 if not found

4.45.2.7 vector_insert()

```
void vector_insert (
    Vector * vector,
    size_t idx,
    void * value )
```

Inserts a value at the specified index.

Parameters

<i>vector</i>	The vector to insert into
<i>idx</i>	The index to insert at
<i>value</i>	The value to insert

4.45.2.8 vector_pop_back()

```
void * vector_pop_back (
    Vector * vector )
```

Pops a value from the back of the vector.

Parameters

<i>vector</i>	The vector to pop from
---------------	------------------------

Returns

void* The popped value

4.45.2.9 vector_push_back()

```
void vector_push_back (
    Vector * vector,
    void * value )
```

Pushes a value to the back of the vector.

Parameters

<i>vector</i>	The vector to push to
<i>value</i>	The value to push

4.45.2.10 vector_remove()

```
void vector_remove (
    Vector * vector,
    void * value )
```

Removes the first occurrence of a value from the vector.

Parameters

<i>vector</i>	The vector to remove from
<i>value</i>	The value to remove

4.45.2.11 vector_remove_at()

```
void vector_remove_at (
    Vector * vector,
    size_t idx )
```

Removes the value at the specified index.

Parameters

<i>vector</i>	The vector to remove from
<i>idx</i>	The index of the value to remove

4.45.2.12 vector_reserve()

```
void vector_reserve (
    Vector * vector,
    size_t capacity )
```

Reserves a new capacity for the vector.

Parameters

<i>vector</i>	The vector to reserve for
<i>size</i>	The new capacity of the vector

4.45.2.13 vector_set()

```
void vector_set (
    Vector * vector,
    size_t idx,
    void * value )
```

Sets the value at the specified index.

Parameters

<i>vector</i>	The vector to set the value in
<i>idx</i>	The index of the element to set
<i>value</i>	The value to set

4.45.2.14 vector_size()

```
size_t vector_size (
    Vector * vector )
```

Returns the size of the vector.

Parameters

<i>vector</i>	The vector to get the size of
---------------	-------------------------------

Returns

size_t The size of the vector

4.46 vector.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdlib.h>
00004 #include <stdbool.h>
00005
00009 typedef struct Vector {
00010     size_t capacity;
00011     size_t size;
00012     void** data;
00013 } Vector;
00014
00021 Vector* vector_create(size_t capacity);
00027 void vector_destroy(Vector* vector);
00035 void* vector_get(Vector* vector, size_t idx);
00043 void vector_set(Vector* vector, size_t idx, void* value);
00050 void vector_push_back(Vector* vector, void* value);
00057 void* vector_pop_back(Vector* vector);
00065 void vector_insert(Vector* vector, size_t idx, void* value);
00074 bool vector_contains(Vector* vector, void* value);
00082 int vector_index_of(Vector* vector, void* value);
00089 void vector_remove_at(Vector* vector, size_t idx);
00096 void vector_remove(Vector* vector, void* value);
00103 void vector_reserve(Vector* vector, size_t capacity);
00110 size_t vector_size(Vector* vector);
00116 void vector_clear(Vector* vector);
```

4.47 src/window/window.c File Reference

Functions

- [Window](#) * [window_create](#) (const char *title, int width, int height, int flags)
- void [window_show](#) ([Window](#) *window)
- void [window_hide](#) ([Window](#) *window)
- void [window_focus](#) ([Window](#) *window)
- [UIContainer](#) * [window_get_main_container](#) ([Window](#) *window)
- void [_window_reset](#) ([Window](#) *window)
- void [_window_handle_event](#) ([Window](#) *window, [SDL_Event](#) *event)
- void [_window_update](#) ([Window](#) *window)
- void [_window_render](#) ([Window](#) *window)
- void [_window_close](#) ([Window](#) *window)

4.47.1 Function Documentation

4.47.1.1 _window_close()

```
void _window_close (
    Window * window )
```

Closes a window (closes the input and ui, destroys the renderer and the window, should not be called manually)

Parameters

<i>window</i>	The window to close
---------------	---------------------

4.47.1.2 `_window_handle_event()`

```
void _window_handle_event (
    Window * window,
    SDL_Event * event )
```

Handles an event for a window and calls the input and ui event handlers (should not be called manually)

Parameters

<i>window</i>	The window to handle the event for
<i>event</i>	The event to handle

4.47.1.3 `_window_render()`

```
void _window_render (
    Window * window )
```

Renders a window and the ui (should not be called manually)

Parameters

<i>window</i>	The window to render
---------------	----------------------

4.47.1.4 `_window_reset()`

```
void _window_reset (
    Window * window )
```

Resets a window (resets the input data and sets close_requested to false)

Parameters

<i>window</i>	The window to reset
---------------	---------------------

4.47.1.5 `_window_update()`

```
void _window_update (
    Window * window )
```

Updates a window and updates the ui (should not be called manually)

Parameters

<i>window</i>	The window to update
---------------	----------------------

4.47.1.6 window_create()

```
Window * window_create (
    const char * title,
    int width,
    int height,
    int flags )
```

Creates a new [Window](#).

Parameters

<i>title</i>	The title of the window
<i>width</i>	The width of the window
<i>height</i>	The height of the window
<i>flags</i>	The flags of the window

Returns

Window* The created window

4.47.1.7 window_focus()

```
void window_focus (
    Window * window )
```

Focuses a window.

Parameters

<i>window</i>	The window to focus
---------------	---------------------

4.47.1.8 window_get_main_container()

```
UIContainer * window_get_main_container (
    Window * window )
```

Returns the main container of a window (needed for the ui)

Parameters

<i>window</i>	The window to get the main container from
---------------	---

Returns

UIContainer* The main container of the window

4.47.1.9 window_hide()

```
void window_hide (
    Window * window )
```

Hides a window.

Parameters

<i>window</i>	The window to hide
---------------	--------------------

4.47.1.10 window_show()

```
void window_show (
    Window * window )
```

Shows a window.

Parameters

<i>window</i>	The window to show
---------------	--------------------

4.48 src/window/window.h File Reference

Classes

- struct [Window](#)

Functions

- [Window](#) * [window_create](#) (const char *title, int width, int height, int flags)
- void [window_show](#) ([Window](#) *window)
- void [window_hide](#) ([Window](#) *window)
- void [window_focus](#) ([Window](#) *window)
- [UIContainer](#) * [window_get_main_container](#) ([Window](#) *window)
- void [_window_reset](#) ([Window](#) *window)
- void [_window_handle_event](#) ([Window](#) *window, [SDL_Event](#) *event)
- void [_window_update](#) ([Window](#) *window)
- void [_window_render](#) ([Window](#) *window)
- void [_window_close](#) ([Window](#) *window)

4.48.1 Typedef Documentation

4.48.1.1 Window

```
typedef struct Window Window
```

The [Window](#) struct, contains an [SDL_Window](#), an [SDL_Renderer](#) and other window specific data, like the input data and the UI data, and a flag to check if the window is requested to be closed.

4.48.2 Function Documentation

4.48.2.1 `_window_close()`

```
void _window_close (
    Window * window )
```

Closes a window (closes the input and ui, destroys the renderer and the window, should not be called manually)

Parameters

<i>window</i>	The window to close
---------------	---------------------

4.48.2.2 `_window_handle_event()`

```
void _window_handle_event (
    Window * window,
    SDL_Event * event )
```

Handles an event for a window and calls the input and ui event handlers (should not be called manually)

Parameters

<i>window</i>	The window to handle the event for
<i>event</i>	The event to handle

4.48.2.3 `_window_render()`

```
void _window_render (
    Window * window )
```

Renders a window and the ui (should not be called manually)

Parameters

<i>window</i>	The window to render
---------------	----------------------

4.48.2.4 `_window_reset()`

```
void _window_reset (
    Window * window )
```

Resets a window (resets the input data and sets close_requested to false)

Parameters

<i>window</i>	The window to reset
---------------	---------------------

4.48.2.5 `_window_update()`

```
void _window_update (
    Window * window )
```

Updates a window and updates the ui (should not be called manually)

Parameters

<i>window</i>	The window to update
---------------	----------------------

4.48.2.6 `window_create()`

```
Window * window_create (
    const char * title,
    int width,
    int height,
    int flags )
```

Creates a new [Window](#).

Parameters

<i>title</i>	The title of the window
<i>width</i>	The width of the window
<i>height</i>	The height of the window
<i>flags</i>	The flags of the window

Returns

Window* The created window

4.48.2.7 `window_focus()`

```
void window_focus (
    Window * window )
```

Focuses a window.

Parameters

<i>window</i>	The window to focus
---------------	---------------------

4.48.2.8 `window_get_main_container()`

```
UIContainer * window_get_main_container (
    Window * window )
```

Returns the main container of a window (needed for the ui)

Parameters

<i>window</i>	The window to get the main container from
---------------	---

Returns

UIContainer* The main container of the window

4.48.2.9 window_hide()

```
void window_hide (
    Window * window )
```

Hides a window.

Parameters

<i>window</i>	The window to hide
---------------	--------------------

4.48.2.10 window_show()

```
void window_show (
    Window * window )
```

Shows a window.

Parameters

<i>window</i>	The window to show
---------------	--------------------

4.49 window.h

[Go to the documentation of this file.](#)

```
00001 #ifndef WINDOW_H
00002 #define WINDOW_H
00003
00004 #ifdef _WIN32
00005     #include <SDL.h>
00006 #elif defined(__unix__) || defined(__linux__)
00007     #include <SDL2/SDL.h>
00008 #endif
00009
00010 #include <stdbool.h>
00011 #include "../input/input.h"
00012 #include "../ui/ui.h"
00013 #include "../ui/ui_element/ui_element.h"
00014
00019 typedef struct Window
00020 {
00021     SDL_Window* window;
00022     SDL_Renderer* renderer;
00023     InputData input_data;
00024     UIData ui_data;
00025     bool close_requested;
```

```
00026 } Window;
00027
00037 Window* window_create(const char* title, int width, int height, int flags);
00043 void window_show(Window* window);
00049 void window_hide(Window* window);
00055 void window_focus(Window* window);
00062 UIContainer* window_get_main_container(Window* window);
00063
00069 void _window_reset(Window* window);
00076 void _window_handle_event(Window* window, SDL_Event* event);
00082 void _window_update(Window* window);
00088 void _window_render(Window* window);
00094 void _window_close(Window* window);
00095
00096 #endif
```


Index

- [_UIDropdownItem](#)
 - [ui_element.c, 143](#)
 - [_UISplitButtonItem](#)
 - [ui_element.c, 143](#)
 - [_app_add_window](#)
 - [app.c, 27](#)
 - [app.h, 30](#)
 - [_font_close](#)
 - [font.c, 41](#)
 - [font.h, 42](#)
 - [_font_init](#)
 - [font.c, 41](#)
 - [font.h, 42](#)
 - [_input_close](#)
 - [input.c, 93](#)
 - [input.h, 98](#)
 - [_input_handle_event](#)
 - [input.c, 94](#)
 - [input.h, 98](#)
 - [_input_init](#)
 - [input.c, 94](#)
 - [input.h, 99](#)
 - [_input_reset](#)
 - [input.c, 94](#)
 - [input.h, 99](#)
 - [_input_set_target](#)
 - [input.c, 94](#)
 - [input.h, 99](#)
 - [_renderer_set_target](#)
 - [renderer.c, 108](#)
 - [renderer.h, 119](#)
 - [_texture_add](#)
 - [texture.c, 130](#)
 - [texture.h, 131](#)
 - [_texture_close](#)
 - [texture.c, 130](#)
 - [texture.h, 131](#)
 - [_texture_init](#)
 - [texture.c, 130](#)
 - [texture.h, 131](#)
 - [_ui_close](#)
 - [ui.c, 132](#)
 - [ui.h, 135](#)
 - [_ui_container_destroy](#)
 - [ui_element.c, 143](#)
 - [ui_element.h, 154](#)
 - [_ui_container_recalculate](#)
 - [ui_element.c, 144](#)
 - [ui_element.h, 154](#)
 - [_ui_container_render](#)
 - [ui_element.c, 144](#)
 - [ui_element.h, 154](#)
 - [_ui_container_update](#)
 - [ui_element.c, 144](#)
 - [ui_element.h, 154](#)
 - [_ui_get_target](#)
 - [ui.c, 133](#)
 - [ui.h, 135](#)
 - [_ui_handle_event](#)
 - [ui.c, 133](#)
 - [ui.h, 135](#)
 - [_ui_init](#)
 - [ui.c, 133](#)
 - [ui.h, 135](#)
 - [_ui_render](#)
 - [ui.c, 133](#)
 - [ui.h, 136](#)
 - [_ui_set_target](#)
 - [ui.c, 134](#)
 - [ui.h, 136](#)
 - [_ui_update](#)
 - [ui.c, 134](#)
 - [ui.h, 136](#)
 - [_window_close](#)
 - [window.c, 178](#)
 - [window.h, 182](#)
 - [_window_handle_event](#)
 - [window.c, 178](#)
 - [window.h, 182](#)
 - [_window_render](#)
 - [window.c, 179](#)
 - [window.h, 182](#)
 - [_window_reset](#)
 - [window.c, 179](#)
 - [window.h, 182](#)
 - [_window_update](#)
 - [window.c, 179](#)
 - [window.h, 183](#)
 - [angle_bisector_create](#)
 - [shape.c, 59](#)
 - [shape.h, 68](#)
 - [AngleBisector, 5](#)
 - [base, 5](#)
 - [line1, 5](#)
 - [line2, 5](#)
 - [shape.h, 65](#)
 - [app.c](#)
 - [_app_add_window, 27](#)

- app_close, 27
- app_get_delta_time, 27
- app_get_target, 28
- app_get_time, 28
- app_get_windows, 28
- app_init, 28
- app_render, 28
- app_request_close, 29
- app_set_target, 29
- app_set_target_fps, 29
- app_update, 29
- app.h
 - _app_add_window, 30
 - app_close, 30
 - app_get_delta_time, 30
 - app_get_target, 30
 - app_get_time, 31
 - app_get_windows, 31
 - app_init, 31
 - app_render, 31
 - app_request_close, 31
 - app_set_target, 31
 - app_set_target_fps, 32
 - app_update, 32
 - AppData, 30
- app_close
 - app.c, 27
 - app.h, 30
- app_get_delta_time
 - app.c, 27
 - app.h, 30
- app_get_target
 - app.c, 28
 - app.h, 30
- app_get_time
 - app.c, 28
 - app.h, 31
- app_get_windows
 - app.c, 28
 - app.h, 31
- app_init
 - app.c, 28
 - app.h, 31
- app_render
 - app.c, 28
 - app.h, 31
- app_request_close
 - app.c, 29
 - app.h, 31
- app_set_target
 - app.c, 29
 - app.h, 31
- app_set_target_fps
 - app.c, 29
 - app.h, 32
- app_update
 - app.c, 29
 - app.h, 32
- AppData, 5
 - app.h, 30
 - delta_time, 6
 - frame_start, 6
 - last_frame_start, 6
 - target_frame_time, 6
 - windows, 6
- auto_dropdown
 - UISplitButton, 22
- backspace_pressed
 - UIData, 16
- base
 - AngleBisector, 5
 - Circle, 6
 - Line, 9
 - Parallel, 10
 - Perpendicular, 10
 - Point, 11
 - Tangent, 12
 - UIButton, 13
 - UICheckbox, 14
 - UIContainer, 16
 - UIDropdownList, 17
 - UIImageButton, 19
 - UILabel, 20
 - UIPanel, 20
 - UISlider, 21
 - UISplitButton, 22
 - UITextbox, 23
- BLACK
 - color.h, 36
- BLUE
 - color.h, 36
- border_color
 - UIPanel, 20
- border_width
 - UIPanel, 20
- capacity
 - Vector, 24
- center
 - Circle, 6
- check_collision_point_rect
 - math.c, 163
 - math.h, 166
- checked
 - UICheckbox, 14
- checked_color
 - UICheckbox, 14
- children
 - UIContainer, 16
- Circle, 6
 - base, 6
 - center, 6
 - perimeter_point, 6
 - shape.h, 65
- circle
 - Tangent, 12

- circle_create
 - shape.c, 60
 - shape.h, 68
- clamp
 - math.c, 163
 - math.h, 166
- close_requested
 - Window, 25
- Color
 - color.h, 37
- color
 - UIButton, 13
 - UIDropdownList, 17
 - UILabel, 20
 - UIPanel, 20
 - UISlider, 21
 - UISplitButton, 22
 - UITextbox, 23
- color.c
 - color_clever_shift, 33
 - color_fade, 33
 - color_from_grayscale, 33
 - color_from_hex, 34
 - color_from_hsv, 34
 - color_from_rgb, 34
 - color_from_rgba, 35
 - color_shift, 35
- color.h
 - BLACK, 36
 - BLUE, 36
 - Color, 37
 - color_clever_shift, 37
 - color_fade, 38
 - color_from_grayscale, 38
 - color_from_hex, 38
 - color_from_hsv, 39
 - color_from_rgb, 39
 - color_from_rgba, 39
 - color_shift, 40
 - CYAN, 36
 - DARK_GRAY, 36
 - GRAY, 36
 - GREEN, 36
 - MAGENTA, 37
 - RED, 37
 - TRANSPARENT, 37
 - WHITE, 37
 - YELLOW, 37
- color_clever_shift
 - color.c, 33
 - color.h, 37
- color_fade
 - color.c, 33
 - color.h, 38
- color_from_grayscale
 - color.c, 33
 - color.h, 38
- color_from_hex
 - color.c, 34
 - color.h, 38
- color_from_hsv
 - color.c, 34
 - color.h, 39
- color_from_rgb
 - color.c, 34
 - color.h, 39
- color_from_rgba
 - color.c, 35
 - color.h, 39
- color_shift
 - color.c, 35
 - color.h, 40
- constraint_type
 - UIConstraint, 15
- constraints
 - UIElement, 18
- constraints_from_string
 - ui_constraint.c, 137
 - ui_constraint.h, 140
- ConstraintType
 - ui_constraint.h, 139, 140
- coordinate_system.c
 - coordinate_system_clear, 43
 - coordinate_system_create, 44
 - coordinate_system_delete_selected_shapes, 44
 - coordinate_system_deselect_shape, 44
 - coordinate_system_deselect_shapes, 44
 - coordinate_system_destroy, 45
 - coordinate_system_destroy_shape, 45
 - coordinate_system_drag_selected_shapes, 45
 - coordinate_system_draw, 45
 - coordinate_system_get_hovered_shape, 46
 - coordinate_system_get_selected_shapes, 46
 - coordinate_system_is_hovered, 46
 - coordinate_system_load, 47
 - coordinate_system_save, 47
 - coordinate_system_select_all_shapes, 47
 - coordinate_system_select_shape, 47
 - coordinate_system_translate, 48
 - coordinate_system_update, 48
 - coordinate_system_update_dimensions, 48
 - coordinate_system_zoom, 49
 - coordinates_to_screen, 49
 - screen_to_coordinates, 49
- coordinate_system.h
 - coordinate_system_clear, 50
 - coordinate_system_create, 51
 - coordinate_system_delete_selected_shapes, 51
 - coordinate_system_deselect_shape, 51
 - coordinate_system_deselect_shapes, 51
 - coordinate_system_destroy, 52
 - coordinate_system_destroy_shape, 52
 - coordinate_system_drag_selected_shapes, 52
 - coordinate_system_draw, 53
 - coordinate_system_get_hovered_shape, 53
 - coordinate_system_get_selected_shapes, 53

- coordinate_system_is_hovered, 53
- coordinate_system_load, 54
- coordinate_system_save, 54
- coordinate_system_select_all_shapes, 54
- coordinate_system_select_shape, 54
- coordinate_system_translate, 55
- coordinate_system_update, 55
- coordinate_system_update_dimensions, 55
- coordinate_system_zoom, 56
- coordinates_to_screen, 56
- CoordinateSystem, 50
- INITIAL_ZOOM, 50
- screen_to_coordinates, 56
- coordinate_system_clear
 - coordinate_system.c, 43
 - coordinate_system.h, 50
- coordinate_system_create
 - coordinate_system.c, 44
 - coordinate_system.h, 51
- coordinate_system_delete_selected_shapes
 - coordinate_system.c, 44
 - coordinate_system.h, 51
- coordinate_system_deselect_shape
 - coordinate_system.c, 44
 - coordinate_system.h, 51
- coordinate_system_deselect_shapes
 - coordinate_system.c, 44
 - coordinate_system.h, 51
- coordinate_system_destroy
 - coordinate_system.c, 45
 - coordinate_system.h, 52
- coordinate_system_destroy_shape
 - coordinate_system.c, 45
 - coordinate_system.h, 52
- coordinate_system_drag_selected_shapes
 - coordinate_system.c, 45
 - coordinate_system.h, 52
- coordinate_system_draw
 - coordinate_system.c, 45
 - coordinate_system.h, 53
- coordinate_system_get_hovered_shape
 - coordinate_system.c, 46
 - coordinate_system.h, 53
- coordinate_system_get_selected_shapes
 - coordinate_system.c, 46
 - coordinate_system.h, 53
- coordinate_system_is_hovered
 - coordinate_system.c, 46
 - coordinate_system.h, 53
- coordinate_system_load
 - coordinate_system.c, 47
 - coordinate_system.h, 54
- coordinate_system_save
 - coordinate_system.c, 47
 - coordinate_system.h, 54
- coordinate_system_select_all_shapes
 - coordinate_system.c, 47
 - coordinate_system.h, 54
- coordinate_system_select_shape
 - coordinate_system.c, 47
 - coordinate_system.h, 54
- coordinate_system_translate
 - coordinate_system.c, 48
 - coordinate_system.h, 55
- coordinate_system_update
 - coordinate_system.c, 48
 - coordinate_system.h, 55
- coordinate_system_update_dimensions
 - coordinate_system.c, 48
 - coordinate_system.h, 55
- coordinate_system_zoom
 - coordinate_system.c, 49
 - coordinate_system.h, 56
- coordinates
 - Point, 11
- coordinates_to_screen
 - coordinate_system.c, 49
 - coordinate_system.h, 56
- CoordinateSystem, 7
 - coordinate_system.h, 50
 - intersection_points, 7
 - origin, 7
 - position, 7
 - shape.h, 65
 - shapes, 7
 - size, 7
 - zoom, 7
- corner_radius
 - UIButton, 13
 - UICheckbox, 14
 - UIDropdownList, 17
 - UIPanel, 20
 - UISlider, 21
 - UISplitButton, 22
 - UITextbox, 23
- cs
 - main.c, 107
- CT_ASPECT
 - ui_constraint.h, 140
- CT_CENTER
 - ui_constraint.h, 140
- CT_OFFSET
 - ui_constraint.h, 140
- CT_PIXEL
 - ui_constraint.h, 140
- CT_RELATIVE
 - ui_constraint.h, 140
- current_keyboard_state
 - InputData, 8
- current_mouse_button_state
 - InputData, 8
- current_mouse_position
 - InputData, 8
- CYAN
 - color.h, 36
- DARK_GRAY

- color.h, 36
- data
 - Vector, 24
- deg_to_rad
 - math.c, 163
 - math.h, 166
- delta_time
 - AppData, 6
- destroy
 - UIElement, 18
- dragged
 - Shape, 11
- EPSILON
 - intersection.c, 57
 - shape.c, 59
- expanded
 - UIDropdownList, 17
 - UISplitButton, 22
- expanded_splitbutton
 - UIData, 16
- focused
 - UITextbox, 23
- Font, 8
 - font, 8
 - font.h, 42
 - size, 8
- font
 - Font, 8
- font.c
 - _font_close, 41
 - _font_init, 41
 - font_load, 41
- font.h
 - _font_close, 42
 - _font_init, 42
 - Font, 42
 - font_load, 42
- font_load
 - font.c, 41
 - font.h, 42
- FPS
 - main.c, 103
- frame_start
 - AppData, 6
- GRAY
 - color.h, 36
- GREEN
 - color.h, 36
- HALF_PI
 - math.h, 165
- height
 - Texture, 12
 - UIConstraints, 15
- INITIAL_ZOOM
 - coordinate_system.h, 50
- input.c
 - _input_close, 93
 - _input_handle_event, 94
 - _input_init, 94
 - _input_reset, 94
 - _input_set_target, 94
 - input_get_mouse_motion, 95
 - input_get_mouse_position, 95
 - input_get_mouse_wheel_delta, 95
 - input_is_key_down, 95
 - input_is_key_pressed, 96
 - input_is_key_released, 96
 - input_is_mouse_button_down, 96
 - input_is_mouse_button_pressed, 97
 - input_is_mouse_button_released, 97
- input.h
 - _input_close, 98
 - _input_handle_event, 98
 - _input_init, 99
 - _input_reset, 99
 - _input_set_target, 99
 - input_get_mouse_motion, 99
 - input_get_mouse_position, 99
 - input_get_mouse_wheel_delta, 100
 - input_is_key_down, 100
 - input_is_key_pressed, 100
 - input_is_key_released, 101
 - input_is_mouse_button_down, 101
 - input_is_mouse_button_pressed, 101
 - input_is_mouse_button_released, 102
 - InputData, 98
- input_data
 - Window, 25
- input_get_mouse_motion
 - input.c, 95
 - input.h, 99
- input_get_mouse_position
 - input.c, 95
 - input.h, 99
- input_get_mouse_wheel_delta
 - input.c, 95
 - input.h, 100
- input_is_key_down
 - input.c, 95
 - input.h, 100
- input_is_key_pressed
 - input.c, 96
 - input.h, 100
- input_is_key_released
 - input.c, 96
 - input.h, 101
- input_is_mouse_button_down
 - input.c, 96
 - input.h, 101
- input_is_mouse_button_pressed
 - input.c, 97
 - input.h, 101

- input_is_mouse_button_released
 - input.c, 97
 - input.h, 102
- InputData, 8
 - current_keyboard_state, 8
 - current_mouse_button_state, 8
 - current_mouse_position, 8
 - input.h, 98
 - key_count, 8
 - mouse_wheel_delta, 8
 - old_keyboard_state, 9
 - old_mouse_button_state, 9
 - old_mouse_position, 9
- intersection.c
 - EPSILON, 57
 - intersection_get, 58
- intersection.h
 - intersection_get, 58
- intersection_get
 - intersection.c, 58
 - intersection.h, 58
- intersection_points
 - CoordinateSystem, 7
- items
 - UIDropdownList, 17
 - UISplitButton, 22
- key_count
 - InputData, 8
- last_frame_start
 - AppData, 6
- lerp
 - math.c, 164
 - math.h, 167
- Line, 9
 - base, 9
 - p1, 9
 - p2, 9
 - shape.h, 66
- line
 - Parallel, 10
 - Perpendicular, 10
- line1
 - AngleBisector, 5
- line2
 - AngleBisector, 5
- line_create
 - shape.c, 60
 - shape.h, 68
- MAGENTA
 - color.h, 37
- main
 - main.c, 104
- main.c
 - cs, 107
 - FPS, 103
 - main, 104
- MOUSE_WHEEL_SENSITIVITY, 103
 - on_angle_bisector_clicked, 104
 - on_cancel_button_clicked, 104
 - on_canvas_size_changed, 105
 - on_circle_clicked, 105
 - on_editmenu_clicked, 105
 - on_filemenu_clicked, 105
 - on_line_clicked, 105, 106
 - on_open_button_clicked, 106
 - on_parallel_clicked, 106
 - on_perpendicular_clicked, 106
 - on_point_clicked, 106
 - on_pointer_clicked, 106, 107
 - on_save_button_clicked, 107
 - on_tangent_clicked, 107
 - State, 103, 104
 - state, 107
 - STATE_ANGLE_BISECTOR, 104
 - STATE_ANGLE_BISECTOR_LINE1_SELECTED, 104
 - STATE_CIRCLE, 104
 - STATE_CIRCLE_CENTER_PLACED, 104
 - STATE_CS_DRAGGED, 104
 - STATE_LINE, 104
 - STATE_LINE_POINT1_PLACED, 104
 - STATE_OPENING, 104
 - STATE_PARALLEL, 104
 - STATE_PARALLEL_LINE_SELECTED, 104
 - STATE_PERPENDICULAR, 104
 - STATE_PERPENDICULAR_LINE_SELECTED, 104
 - STATE_POINT, 104
 - STATE_POINTER, 104
 - STATE_SAVEING, 104
 - STATE_TANGENT, 104
 - STATE_TANGENT_LINE_SELECTED, 104
- main_container
 - UIData, 16
- map
 - math.c, 164
 - math.h, 167
- math.c
 - check_collision_point_rect, 163
 - clamp, 163
 - deg_to_rad, 163
 - lerp, 164
 - map, 164
 - rad_to_deg, 165
- math.h
 - check_collision_point_rect, 166
 - clamp, 166
 - deg_to_rad, 166
 - HALF_PI, 165
 - lerp, 167
 - map, 167
 - PI, 165
 - rad_to_deg, 168
 - TWO_PI, 165

- mouse_captured
 - UIData, 16
- mouse_state
 - UIButton, 13
 - UICheckbox, 14
 - UIImageButton, 19
 - UISlider, 21
 - UITextbox, 23
- mouse_wheel_delta
 - InputData, 8
- MOUSE_WHEEL_SENSITIVITY
 - main.c, 103
- MouseState
 - ui_element.h, 151, 153
- MS_HOVER
 - ui_element.h, 154
- MS_NONE
 - ui_element.h, 154
- MS_PRESS
 - ui_element.h, 154
- new_aspect_constraint
 - ui_constraint.c, 137
 - ui_constraint.h, 141
- new_center_constraint
 - ui_constraint.c, 138
 - ui_constraint.h, 141
- new_offset_constraint
 - ui_constraint.c, 138
 - ui_constraint.h, 141
- new_pixel_constraint
 - ui_constraint.c, 138
 - ui_constraint.h, 141
- new_relative_constraint
 - ui_constraint.c, 139
 - ui_constraint.h, 142
- old_keyboard_state
 - InputData, 9
- old_mouse_button_state
 - InputData, 9
- old_mouse_position
 - InputData, 9
- on_angle_bisector_clicked
 - main.c, 104
- on_cancel_button_clicked
 - main.c, 104
- on_canvas_size_changed
 - main.c, 105
- on_checked_changed
 - UICheckbox, 14
- on_circle_clicked
 - main.c, 105
- on_click
 - UIButton, 13
 - UIImageButton, 19
- on_editmenu_clicked
 - main.c, 105
- on_filemenu_clicked
 - main.c, 105
- on_item_clicked
 - UISplitButton, 22
- on_line_clicked
 - main.c, 105, 106
- on_open_button_clicked
 - main.c, 106
- on_parallel_clicked
 - main.c, 106
- on_perpendicular_clicked
 - main.c, 106
- on_point_clicked
 - main.c, 106
- on_pointer_clicked
 - main.c, 106, 107
- on_save_button_clicked
 - main.c, 107
- on_selection_changed
 - UIDropdownList, 17
- on_size_changed
 - UIContainer, 16
- on_tangent_clicked
 - main.c, 107
- on_text_changed
 - UITextbox, 23
- on_value_changed
 - UISlider, 21
- origin
 - CoordinateSystem, 7
- OVERLAP_DISTANCE
 - shape.h, 65
- p1
 - Line, 9
- p2
 - Line, 9
- Parallel, 10
 - base, 10
 - line, 10
 - point, 10
 - shape.h, 66
- parallel_create
 - shape.c, 60
 - shape.h, 69
- parent
 - UIElement, 18
- perimeter_point
 - Circle, 6
- Perpendicular, 10
 - base, 10
 - line, 10
 - point, 10
 - shape.h, 66
- perpendicular_create
 - shape.c, 61
 - shape.h, 69
- PI
 - math.h, 165
- Point, 11

- base, 11
- coordinates, 11
- shape.h, 66
- point
 - Parallel, 10
 - Perpendicular, 10
 - Tangent, 12
- point_create
 - shape.c, 61
 - shape.h, 69
- position
 - CoordinateSystem, 7
 - UIElement, 18
- rad_to_deg
 - math.c, 165
 - math.h, 168
- recalculate
 - UIElement, 18
- RED
 - color.h, 37
- render
 - UIElement, 18
- renderer
 - Window, 25
- renderer.c
 - _renderer_set_target, 108
 - renderer_bind_framebuffer, 108
 - renderer_clear, 109
 - renderer_create_framebuffer, 109
 - renderer_draw_arc, 109
 - renderer_draw_bezier, 110
 - renderer_draw_circle, 110
 - renderer_draw_ellipse, 110
 - renderer_draw_filled_circle, 111
 - renderer_draw_filled_ellipse, 111
 - renderer_draw_filled_polygon, 112
 - renderer_draw_filled_rect, 112
 - renderer_draw_filled_rounded_rect, 112
 - renderer_draw_filled_triangle, 113
 - renderer_draw_line, 113
 - renderer_draw_pie, 114
 - renderer_draw_pixel, 114
 - renderer_draw_polygon, 114
 - renderer_draw_rect, 115
 - renderer_draw_rounded_rect, 115
 - renderer_draw_text, 116
 - renderer_draw_texture, 116
 - renderer_draw_triangle, 116
 - renderer_query_text_size, 117
 - renderer_reset_clip_rect, 117
 - renderer_resize_framebuffer, 117
 - renderer_set_clip_rect, 117
 - renderer_set_default_font, 118
- renderer.h
 - _renderer_set_target, 119
 - renderer_bind_framebuffer, 119
 - renderer_clear, 119
 - renderer_create_framebuffer, 119
 - renderer_draw_arc, 120
 - renderer_draw_bezier, 120
 - renderer_draw_circle, 121
 - renderer_draw_ellipse, 121
 - renderer_draw_filled_circle, 121
 - renderer_draw_filled_ellipse, 122
 - renderer_draw_filled_polygon, 122
 - renderer_draw_filled_rect, 122
 - renderer_draw_filled_rounded_rect, 123
 - renderer_draw_filled_triangle, 123
 - renderer_draw_line, 124
 - renderer_draw_pie, 124
 - renderer_draw_pixel, 125
 - renderer_draw_polygon, 125
 - renderer_draw_rect, 125
 - renderer_draw_rounded_rect, 126
 - renderer_draw_text, 126
 - renderer_draw_texture, 126
 - renderer_draw_triangle, 127
 - renderer_query_text_size, 127
 - renderer_reset_clip_rect, 128
 - renderer_resize_framebuffer, 128
 - renderer_set_clip_rect, 128
 - renderer_set_default_font, 128
- renderer_bind_framebuffer
 - renderer.c, 108
 - renderer.h, 119
- renderer_clear
 - renderer.c, 109
 - renderer.h, 119
- renderer_create_framebuffer
 - renderer.c, 109
 - renderer.h, 119
- renderer_draw_arc
 - renderer.c, 109
 - renderer.h, 120
- renderer_draw_bezier
 - renderer.c, 110
 - renderer.h, 120
- renderer_draw_circle
 - renderer.c, 110
 - renderer.h, 121
- renderer_draw_ellipse
 - renderer.c, 110
 - renderer.h, 121
- renderer_draw_filled_circle
 - renderer.c, 111
 - renderer.h, 121
- renderer_draw_filled_ellipse
 - renderer.c, 111
 - renderer.h, 122
- renderer_draw_filled_polygon
 - renderer.c, 112
 - renderer.h, 122
- renderer_draw_filled_rect
 - renderer.c, 112
 - renderer.h, 122
- renderer_draw_filled_rounded_rect

- renderer.c, [112](#)
- renderer.h, [123](#)
- renderer_draw_filled_triangle
 - renderer.c, [113](#)
 - renderer.h, [123](#)
- renderer_draw_line
 - renderer.c, [113](#)
 - renderer.h, [124](#)
- renderer_draw_pie
 - renderer.c, [114](#)
 - renderer.h, [124](#)
- renderer_draw_pixel
 - renderer.c, [114](#)
 - renderer.h, [125](#)
- renderer_draw_polygon
 - renderer.c, [114](#)
 - renderer.h, [125](#)
- renderer_draw_rect
 - renderer.c, [115](#)
 - renderer.h, [125](#)
- renderer_draw_rounded_rect
 - renderer.c, [115](#)
 - renderer.h, [126](#)
- renderer_draw_text
 - renderer.c, [116](#)
 - renderer.h, [126](#)
- renderer_draw_texture
 - renderer.c, [116](#)
 - renderer.h, [126](#)
- renderer_draw_triangle
 - renderer.c, [116](#)
 - renderer.h, [127](#)
- renderer_query_text_size
 - renderer.c, [117](#)
 - renderer.h, [127](#)
- renderer_reset_clip_rect
 - renderer.c, [117](#)
 - renderer.h, [128](#)
- renderer_resize_framebuffer
 - renderer.c, [117](#)
 - renderer.h, [128](#)
- renderer_set_clip_rect
 - renderer.c, [117](#)
 - renderer.h, [128](#)
- renderer_set_default_font
 - renderer.c, [118](#)
 - renderer.h, [128](#)
- screen_to_coordinates
 - coordinate_system.c, [49](#)
 - coordinate_system.h, [56](#)
- selected
 - Shape, [11](#)
- selected_item
 - UIDropdownList, [17](#)
- Shape, [11](#)
 - dragged, [11](#)
 - selected, [11](#)
 - shape.h, [66](#)
 - type, [11](#)
- shape.c
 - angle_bisector_create, [59](#)
 - circle_create, [60](#)
 - EPSILON, [59](#)
 - line_create, [60](#)
 - parallel_create, [60](#)
 - perpendicular_create, [61](#)
 - point_create, [61](#)
 - shape_destroy, [61](#)
 - shape_destroy_funcs, [64](#)
 - shape_draw, [62](#)
 - shape_draw_funcs, [64](#)
 - shape_is_defined_by, [62](#)
 - shape_is_defined_by_funcs, [64](#)
 - shape_overlap_point, [62](#)
 - shape_overlap_point_funcs, [64](#)
 - shape_translate, [63](#)
 - shape_translate_funcs, [64](#)
 - shape_update, [63](#)
 - tangent_create, [63](#)
- shape.h
 - angle_bisector_create, [68](#)
 - AngleBisector, [65](#)
 - Circle, [65](#)
 - circle_create, [68](#)
 - CoordinateSystem, [65](#)
 - Line, [66](#)
 - line_create, [68](#)
 - OVERLAP_DISTANCE, [65](#)
 - Parallel, [66](#)
 - parallel_create, [69](#)
 - Perpendicular, [66](#)
 - perpendicular_create, [69](#)
 - Point, [66](#)
 - point_create, [69](#)
 - Shape, [66](#)
 - shape_destroy, [70](#)
 - shape_draw, [70](#)
 - shape_is_defined_by, [70](#)
 - shape_overlap_point, [71](#)
 - shape_translate, [71](#)
 - shape_update, [71](#)
 - ShapeDestroy, [66](#)
 - ShapeDraw, [66](#)
 - ShapeIsDefinedBy, [66](#)
 - ShapeOverlapPoint, [67](#)
 - ShapeTranslate, [67](#)
 - ShapeType, [67](#)
 - ST_ANGLE_BISECTOR, [67](#)
 - ST_CIRCLE, [67](#)
 - ST_COUNT, [67](#)
 - ST_LINE, [67](#)
 - ST_PARALLEL, [67](#)
 - ST_PERPENDICULAR, [67](#)
 - ST_POINT, [67](#)
 - ST_TANGENT, [67](#)
 - Tangent, [67](#)

- tangent_create, 72
- shape_destroy
 - shape.c, 61
 - shape.h, 70
- shape_destroy_funcs
 - shape.c, 64
- shape_draw
 - shape.c, 62
 - shape.h, 70
- shape_draw_funcs
 - shape.c, 64
- shape_is_defined_by
 - shape.c, 62
 - shape.h, 70
- shape_is_defined_by_funcs
 - shape.c, 64
- shape_overlap_point
 - shape.c, 62
 - shape.h, 71
- shape_overlap_point_funcs
 - shape.c, 64
- shape_translate
 - shape.c, 63
 - shape.h, 71
- shape_translate_funcs
 - shape.c, 64
- shape_update
 - shape.c, 63
 - shape.h, 71
- ShapeDestroy
 - shape.h, 66
- ShapeDraw
 - shape.h, 66
- ShapeIsDefinedBy
 - shape.h, 66
- ShapeOverlapPoint
 - shape.h, 67
- shapes
 - CoordinateSystem, 7
- ShapeTranslate
 - shape.h, 67
- ShapeType
 - shape.h, 67
- shown
 - UIElement, 18
- size
 - CoordinateSystem, 7
 - Font, 8
 - UIElement, 18
 - Vector, 24
- slider_color
 - UISlider, 21
- src/app/app.c, 27
- src/app/app.h, 29, 32
- src/color/color.c, 33
- src/color/color.h, 36, 40
- src/font/font.c, 41
- src/font/font.h, 41, 42
- src/geometry/coordinate_system/coordinate_system.c, 43
- src/geometry/coordinate_system/coordinate_system.h, 50, 57
- src/geometry/intersection/intersection.c, 57
- src/geometry/intersection/intersection.h, 58
- src/geometry/shape/shape.c, 59
- src/geometry/shape/shape.h, 65, 72
- src/geometry/vector2/vector2.c, 73
- src/geometry/vector2/vector2.h, 83, 92
- src/input/input.c, 92
- src/input/input.h, 97, 102
- src/main.c, 103
- src/renderer/renderer.c, 108
- src/renderer/renderer.h, 118, 129
- src/texture/texture.c, 129
- src/texture/texture.h, 130, 132
- src/ui/ui.c, 132
- src/ui/ui.h, 134, 137
- src/ui/ui_constraint/ui_constraint.c, 137
- src/ui/ui_constraint/ui_constraint.h, 139, 142
- src/ui/ui_element/ui_element.c, 143
- src/ui/ui_element/ui_element.h, 150, 160
- src/utils/math/math.c, 162
- src/utils/math/math.h, 165, 168
- src/utils/vector/vector.c, 168
- src/utils/vector/vector.h, 173, 178
- src/window/window.c, 178
- src/window/window.h, 181, 185
- ST_ANGLE_BISECTOR
 - shape.h, 67
- ST_CIRCLE
 - shape.h, 67
- ST_COUNT
 - shape.h, 67
- ST_LINE
 - shape.h, 67
- ST_PARALLEL
 - shape.h, 67
- ST_PERPENDICULAR
 - shape.h, 67
- ST_POINT
 - shape.h, 67
- ST_TANGENT
 - shape.h, 67
- State
 - main.c, 103, 104
- state
 - main.c, 107
- STATE_ANGLE_BISECTOR
 - main.c, 104
- STATE_ANGLE_BISECTOR_LINE1_SELECTED
 - main.c, 104
- STATE_CIRCLE
 - main.c, 104
- STATE_CIRCLE_CENTER_PLACED
 - main.c, 104
- STATE_CS_DRAGGED

- main.c, [104](#)
- STATE_LINE
 - main.c, [104](#)
- STATE_LINE_POINT1_PLACED
 - main.c, [104](#)
- STATE_OPENING
 - main.c, [104](#)
- STATE_PARALLEL
 - main.c, [104](#)
- STATE_PARALLEL_LINE_SELECTED
 - main.c, [104](#)
- STATE_PERPENDICULAR
 - main.c, [104](#)
- STATE_PERPENDICULAR_LINE_SELECTED
 - main.c, [104](#)
- STATE_POINT
 - main.c, [104](#)
- STATE_POINTER
 - main.c, [104](#)
- STATE_SAVEING
 - main.c, [104](#)
- STATE_TANGENT
 - main.c, [104](#)
- STATE_TANGENT_LINE_SELECTED
 - main.c, [104](#)
- Tangent, [12](#)
 - base, [12](#)
 - circle, [12](#)
 - point, [12](#)
 - shape.h, [67](#)
- tangent_create
 - shape.c, [63](#)
 - shape.h, [72](#)
- target_frame_time
 - AppData, [6](#)
- target_ui_data
 - ui.c, [134](#)
- text
 - UIButton, [13](#)
 - UILabel, [20](#)
 - UITextbox, [23](#)
- text_color
 - UIButton, [13](#)
 - UIDropdownList, [17](#)
 - UISplitButton, [22](#)
 - UITextbox, [23](#)
- text_input
 - UIData, [16](#)
- text_position
 - UIButton, [13](#)
- Texture, [12](#)
 - height, [12](#)
 - texture, [12](#)
 - texture.h, [131](#)
 - width, [12](#)
- texture
 - Texture, [12](#)
 - UIImageButton, [19](#)
- texture.c
 - _texture_add, [130](#)
 - _texture_close, [130](#)
 - _texture_init, [130](#)
 - texture_load, [130](#)
- texture.h
 - _texture_add, [131](#)
 - _texture_close, [131](#)
 - _texture_init, [131](#)
 - Texture, [131](#)
 - texture_load, [131](#)
- texture_load
 - texture.c, [130](#)
 - texture.h, [131](#)
- thickness
 - UISlider, [21](#)
- TRANSPARENT
 - color.h, [37](#)
- TWO_PI
 - math.h, [165](#)
- type
 - Shape, [11](#)
- ui.c
 - _ui_close, [132](#)
 - _ui_get_target, [133](#)
 - _ui_handle_event, [133](#)
 - _ui_init, [133](#)
 - _ui_render, [133](#)
 - _ui_set_target, [134](#)
 - _ui_update, [134](#)
 - target_ui_data, [134](#)
- ui.h
 - _ui_close, [135](#)
 - _ui_get_target, [135](#)
 - _ui_handle_event, [135](#)
 - _ui_init, [135](#)
 - _ui_render, [136](#)
 - _ui_set_target, [136](#)
 - _ui_update, [136](#)
 - UIData, [135](#)
- ui_constraint.c
 - constraints_from_string, [137](#)
 - new_aspect_constraint, [137](#)
 - new_center_constraint, [138](#)
 - new_offset_constraint, [138](#)
 - new_pixel_constraint, [138](#)
 - new_relative_constraint, [139](#)
- ui_constraint.h
 - constraints_from_string, [140](#)
 - ConstraintType, [139](#), [140](#)
 - CT_ASPECT, [140](#)
 - CT_CENTER, [140](#)
 - CT_OFFSET, [140](#)
 - CT_PIXEL, [140](#)
 - CT_RELATIVE, [140](#)
 - new_aspect_constraint, [141](#)
 - new_center_constraint, [141](#)
 - new_offset_constraint, [141](#)

- new_pixel_constraint, 141
- new_relative_constraint, 142
- UIConstraint, 139
- UIConstraints, 140
- ui_create_button
 - ui_element.c, 144
 - ui_element.h, 155
- ui_create_checkbox
 - ui_element.c, 145
 - ui_element.h, 155
- ui_create_container
 - ui_element.c, 145
 - ui_element.h, 156
- ui_create_dropdown
 - ui_element.c, 146
 - ui_element.h, 156
- ui_create_imagebutton
 - ui_element.c, 146
 - ui_element.h, 157
- ui_create_label
 - ui_element.c, 147
 - ui_element.h, 157
- ui_create_panel
 - ui_element.c, 147
 - ui_element.h, 157
- ui_create_slider
 - ui_element.c, 147
 - ui_element.h, 158
- ui_create_splitbutton
 - ui_element.c, 148
 - ui_element.h, 158
- ui_create_textbox
 - ui_element.c, 148
 - ui_element.h, 159
- ui_data
 - Window, 25
- ui_element.c
 - _UIDropdownItem, 143
 - _UISplitButtonItem, 143
 - _ui_container_destroy, 143
 - _ui_container_recalculate, 144
 - _ui_container_render, 144
 - _ui_container_update, 144
 - ui_create_button, 144
 - ui_create_checkbox, 145
 - ui_create_container, 145
 - ui_create_dropdown, 146
 - ui_create_imagebutton, 146
 - ui_create_label, 147
 - ui_create_panel, 147
 - ui_create_slider, 147
 - ui_create_splitbutton, 148
 - ui_create_textbox, 148
 - ui_hide_element, 149
 - ui_show_element, 149
- ui_element.h
 - _ui_container_destroy, 154
 - _ui_container_recalculate, 154
 - _ui_container_render, 154
 - _ui_container_update, 154
 - MouseState, 151, 153
 - MS_HOVER, 154
 - MS_NONE, 154
 - MS_PRESS, 154
 - ui_create_button, 155
 - ui_create_checkbox, 155
 - ui_create_container, 156
 - ui_create_dropdown, 156
 - ui_create_imagebutton, 157
 - ui_create_label, 157
 - ui_create_panel, 157
 - ui_create_slider, 158
 - ui_create_splitbutton, 158
 - ui_create_textbox, 159
 - ui_hide_element, 159
 - ui_show_element, 160
 - UIButton, 151
 - UIButtonClick, 151
 - UICheckbox, 151
 - UICheckboxCheckedChanged, 151
 - UIContainer, 151
 - UIContainerSizeChanged, 151
 - UIDropdownList, 151
 - UIDropdownListSelectionChanged, 151
 - UIElement, 152
 - UIElementDestroy, 152
 - UIElementRecalculate, 152
 - UIElementRender, 152
 - UIElementUpdate, 152
 - UIImageButton, 152
 - UIImageButtonClick, 152
 - UILabel, 152
 - UIPanel, 152
 - UISlider, 153
 - UISliderValueChanged, 153
 - UISplitButton, 153
 - UISplitButtonClicked, 153
 - UITEXT_MAX_LENGTH, 150
 - UITextbox, 153
 - UITextboxTextChanged, 153
- ui_hide_element
 - ui_element.c, 149
 - ui_element.h, 159
- ui_show_element
 - ui_element.c, 149
 - ui_element.h, 160
- UIButton, 13
 - base, 13
 - color, 13
 - corner_radius, 13
 - mouse_state, 13
 - on_click, 13
 - text, 13
 - text_color, 13
 - text_position, 13
 - ui_element.h, 151

- UIButtonClick
 - ui_element.h, 151
- UICheckbox, 14
 - base, 14
 - checked, 14
 - checked_color, 14
 - corner_radius, 14
 - mouse_state, 14
 - on_checked_changed, 14
 - ui_element.h, 151
 - unchecked_color, 14
- UICheckboxCheckedChanged
 - ui_element.h, 151
- UIConstraint, 14
 - constraint_type, 15
 - ui_constraint.h, 139
 - value, 15
- UIConstraints, 15
 - height, 15
 - ui_constraint.h, 140
 - width, 15
 - x, 15
 - y, 15
- UIContainer, 15
 - base, 16
 - children, 16
 - on_size_changed, 16
 - ui_element.h, 151
- UIContainerSizeChanged
 - ui_element.h, 151
- UIData, 16
 - backspace_pressed, 16
 - expanded_splitbutton, 16
 - main_container, 16
 - mouse_captured, 16
 - text_input, 16
 - ui.h, 135
- UIDropdownList, 17
 - base, 17
 - color, 17
 - corner_radius, 17
 - expanded, 17
 - items, 17
 - on_selection_changed, 17
 - selected_item, 17
 - text_color, 17
 - ui_element.h, 151
- UIDropdownListSelectionChanged
 - ui_element.h, 151
- UIElement, 18
 - constraints, 18
 - destroy, 18
 - parent, 18
 - position, 18
 - recalculate, 18
 - render, 18
 - shown, 18
 - size, 18
 - ui_element.h, 152
 - update, 19
- UIElementDestroy
 - ui_element.h, 152
- UIElementRecalculate
 - ui_element.h, 152
- UIElementRender
 - ui_element.h, 152
- UIElementUpdate
 - ui_element.h, 152
- UIImageButton, 19
 - base, 19
 - mouse_state, 19
 - on_click, 19
 - texture, 19
 - ui_element.h, 152
- UIImageButtonClick
 - ui_element.h, 152
- UILabel, 19
 - base, 20
 - color, 20
 - text, 20
 - ui_element.h, 152
- UIPanel, 20
 - base, 20
 - border_color, 20
 - border_width, 20
 - color, 20
 - corner_radius, 20
 - ui_element.h, 152
- UISlider, 21
 - base, 21
 - color, 21
 - corner_radius, 21
 - mouse_state, 21
 - on_value_changed, 21
 - slider_color, 21
 - thickness, 21
 - ui_element.h, 153
 - value, 21
- UISliderValueChanged
 - ui_element.h, 153
- UISplitButton, 22
 - auto_dropdown, 22
 - base, 22
 - color, 22
 - corner_radius, 22
 - expanded, 22
 - items, 22
 - on_item_clicked, 22
 - text_color, 22
 - ui_element.h, 153
- UISplitButtonClicked
 - ui_element.h, 153
- UITEXT_MAX_LENGTH
 - ui_element.h, 150
- UITextbox, 23
 - base, 23

- color, [23](#)
- corner_radius, [23](#)
- focused, [23](#)
- mouse_state, [23](#)
- on_text_changed, [23](#)
- text, [23](#)
- text_color, [23](#)
- ui_element.h, [153](#)
- UITextboxTextChanged
 - ui_element.h, [153](#)
- unchecked_color
 - UICheckbox, [14](#)
- update
 - UIElement, [19](#)
- value
 - UIConstraint, [15](#)
 - UISlider, [21](#)
- Vector, [24](#)
 - capacity, [24](#)
 - data, [24](#)
 - size, [24](#)
 - vector.h, [173](#)
- vector.c
 - vector_clear, [169](#)
 - vector_contains, [169](#)
 - vector_create, [169](#)
 - vector_destroy, [169](#)
 - vector_get, [170](#)
 - vector_index_of, [170](#)
 - vector_insert, [170](#)
 - vector_pop_back, [171](#)
 - vector_push_back, [171](#)
 - vector_remove, [171](#)
 - vector_remove_at, [172](#)
 - vector_reserve, [172](#)
 - vector_set, [172](#)
 - vector_size, [172](#)
- vector.h
 - Vector, [173](#)
 - vector_clear, [173](#)
 - vector_contains, [174](#)
 - vector_create, [174](#)
 - vector_destroy, [174](#)
 - vector_get, [175](#)
 - vector_index_of, [175](#)
 - vector_insert, [175](#)
 - vector_pop_back, [175](#)
 - vector_push_back, [176](#)
 - vector_remove, [176](#)
 - vector_remove_at, [176](#)
 - vector_reserve, [177](#)
 - vector_set, [177](#)
 - vector_size, [177](#)
- Vector2, [24](#)
 - vector2.h, [84](#)
 - x, [24](#)
 - y, [24](#)
- vector2.c
 - vector2_add, [74](#)
 - vector2_angle, [74](#)
 - vector2_create, [75](#)
 - vector2_cross, [75](#)
 - vector2_distance, [75](#)
 - vector2_divide, [76](#)
 - vector2_dot, [76](#)
 - vector2_down, [76](#)
 - vector2_from_point, [77](#)
 - vector2_from_polar, [77](#)
 - vector2_left, [77](#)
 - vector2_length, [77](#)
 - vector2_multiply, [79](#)
 - vector2_negate, [79](#)
 - vector2_normalize, [79](#)
 - vector2_one, [80](#)
 - vector2_reflect, [80](#)
 - vector2_right, [80](#)
 - vector2_rotate, [80](#)
 - vector2_rotate90, [82](#)
 - vector2_scale, [82](#)
 - vector2_subtract, [82](#)
 - vector2_up, [83](#)
 - vector2_zero, [83](#)
- vector2.h
 - Vector2, [84](#)
 - vector2_add, [84](#)
 - vector2_angle, [85](#)
 - vector2_create, [85](#)
 - vector2_cross, [85](#)
 - vector2_distance, [86](#)
 - vector2_divide, [86](#)
 - vector2_dot, [86](#)
 - vector2_down, [87](#)
 - vector2_from_point, [87](#)
 - vector2_from_polar, [87](#)
 - vector2_left, [87](#)
 - vector2_length, [88](#)
 - vector2_multiply, [88](#)
 - vector2_negate, [88](#)
 - vector2_normalize, [89](#)
 - vector2_one, [89](#)
 - vector2_reflect, [89](#)
 - vector2_right, [90](#)
 - vector2_rotate, [90](#)
 - vector2_rotate90, [90](#)
 - vector2_scale, [90](#)
 - vector2_subtract, [91](#)
 - vector2_up, [91](#)
 - vector2_zero, [91](#)
- vector2_add
 - vector2.c, [74](#)
 - vector2.h, [84](#)
- vector2_angle
 - vector2.c, [74](#)
 - vector2.h, [85](#)
- vector2_create
 - vector2.c, [75](#)

- vector2.h, 85
- vector2_cross
 - vector2.c, 75
 - vector2.h, 85
- vector2_distance
 - vector2.c, 75
 - vector2.h, 86
- vector2_divide
 - vector2.c, 76
 - vector2.h, 86
- vector2_dot
 - vector2.c, 76
 - vector2.h, 86
- vector2_down
 - vector2.c, 76
 - vector2.h, 87
- vector2_from_point
 - vector2.c, 77
 - vector2.h, 87
- vector2_from_polar
 - vector2.c, 77
 - vector2.h, 87
- vector2_left
 - vector2.c, 77
 - vector2.h, 87
- vector2_length
 - vector2.c, 77
 - vector2.h, 88
- vector2_multiply
 - vector2.c, 79
 - vector2.h, 88
- vector2_negate
 - vector2.c, 79
 - vector2.h, 88
- vector2_normalize
 - vector2.c, 79
 - vector2.h, 89
- vector2_one
 - vector2.c, 80
 - vector2.h, 89
- vector2_reflect
 - vector2.c, 80
 - vector2.h, 89
- vector2_right
 - vector2.c, 80
 - vector2.h, 90
- vector2_rotate
 - vector2.c, 80
 - vector2.h, 90
- vector2_rotate90
 - vector2.c, 82
 - vector2.h, 90
- vector2_scale
 - vector2.c, 82
 - vector2.h, 90
- vector2_subtract
 - vector2.c, 82
 - vector2.h, 91
- vector2_up
 - vector2.c, 83
 - vector2.h, 91
- vector2_zero
 - vector2.c, 83
 - vector2.h, 91
- vector_clear
 - vector.c, 169
 - vector.h, 173
- vector_contains
 - vector.c, 169
 - vector.h, 174
- vector_create
 - vector.c, 169
 - vector.h, 174
- vector_destroy
 - vector.c, 169
 - vector.h, 174
- vector_get
 - vector.c, 170
 - vector.h, 175
- vector_index_of
 - vector.c, 170
 - vector.h, 175
- vector_insert
 - vector.c, 170
 - vector.h, 175
- vector_pop_back
 - vector.c, 171
 - vector.h, 175
- vector_push_back
 - vector.c, 171
 - vector.h, 176
- vector_remove
 - vector.c, 171
 - vector.h, 176
- vector_remove_at
 - vector.c, 172
 - vector.h, 176
- vector_reserve
 - vector.c, 172
 - vector.h, 177
- vector_set
 - vector.c, 172
 - vector.h, 177
- vector_size
 - vector.c, 172
 - vector.h, 177
- WHITE
 - color.h, 37
- width
 - Texture, 12
 - UIConstraints, 15
- Window, 25
 - close_requested, 25
 - input_data, 25
 - renderer, 25
 - ui_data, 25

- [window](#), [25](#)
 - [window.h](#), [181](#)
- [window](#)
 - [Window](#), [25](#)
- [window.c](#)
 - [_window_close](#), [178](#)
 - [_window_handle_event](#), [178](#)
 - [_window_render](#), [179](#)
 - [_window_reset](#), [179](#)
 - [_window_update](#), [179](#)
 - [window_create](#), [179](#)
 - [window_focus](#), [180](#)
 - [window_get_main_container](#), [180](#)
 - [window_hide](#), [180](#)
 - [window_show](#), [181](#)
- [window.h](#)
 - [_window_close](#), [182](#)
 - [_window_handle_event](#), [182](#)
 - [_window_render](#), [182](#)
 - [_window_reset](#), [182](#)
 - [_window_update](#), [183](#)
 - [Window](#), [181](#)
 - [window_create](#), [183](#)
 - [window_focus](#), [183](#)
 - [window_get_main_container](#), [183](#)
 - [window_hide](#), [185](#)
 - [window_show](#), [185](#)
- [window_create](#)
 - [window.c](#), [179](#)
 - [window.h](#), [183](#)
- [window_focus](#)
 - [window.c](#), [180](#)
 - [window.h](#), [183](#)
- [window_get_main_container](#)
 - [window.c](#), [180](#)
 - [window.h](#), [183](#)
- [window_hide](#)
 - [window.c](#), [180](#)
 - [window.h](#), [185](#)
- [window_show](#)
 - [window.c](#), [181](#)
 - [window.h](#), [185](#)
- [windows](#)
 - [AppData](#), [6](#)
- [x](#)
 - [UIConstraints](#), [15](#)
 - [Vector2](#), [24](#)
- [y](#)
 - [UIConstraints](#), [15](#)
 - [Vector2](#), [24](#)
- [YELLOW](#)
 - [color.h](#), [37](#)
- [zoom](#)
 - [CoordinateSystem](#), [7](#)