

Graphics

0.0.0

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	Canvas Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	6
3.1.2.1	origin	6
3.1.2.2	parent	6
3.1.2.3	size	6
3.1.2.4	surface	6
3.2	Circle Struct Reference	6
3.2.1	Detailed Description	7
3.2.2	Field Documentation	7
3.2.2.1	canvas	7
3.2.2.2	center	7
3.2.2.3	radius	7
3.3	Color Struct Reference	7
3.3.1	Detailed Description	7
3.3.2	Field Documentation	8
3.3.2.1	alpha	8

3.3.2.2	rgb	8
3.4	Event Struct Reference	8
3.4.1	Detailed Description	8
3.4.2	Field Documentation	9
3.4.2.1	arrows	9
3.4.2.2	quit	9
3.4.2.3	space	9
3.5	Image Struct Reference	9
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	canvas	10
3.5.2.2	surface	10
3.6	Line Struct Reference	10
3.6.1	Detailed Description	11
3.6.2	Field Documentation	11
3.6.2.1	a	11
3.6.2.2	b	11
3.6.2.3	canvas	11
3.7	Pixel Struct Reference	11
3.7.1	Detailed Description	12
3.7.2	Field Documentation	12
3.7.2.1	canvas	12
3.7.2.2	position	12
3.8	Point Struct Reference	13
3.8.1	Detailed Description	13
3.8.2	Field Documentation	13
3.8.2.1	x	13
3.8.2.2	y	13
3.9	Rectangle Struct Reference	13
3.9.1	Detailed Description	14

3.9.2	Field Documentation	14
3.9.2.1	canvas	14
3.9.2.2	origin	14
3.9.2.3	size	14
3.10	Sound Struct Reference	15
3.10.1	Detailed Description	15
3.10.2	Field Documentation	15
3.10.2.1	content	15
3.11	Sphere Struct Reference	15
3.11.1	Field Documentation	16
3.11.1.1	canvas	16
3.11.1.2	center	16
3.11.1.3	radius	16
3.12	Window Struct Reference	16
3.12.1	Field Documentation	17
3.12.1.1	position	17
3.12.1.2	size	17
3.12.1.3	title	17
3.12.1.4	window	17
4	File Documentation	19
4.1	calc.h File Reference	19
4.1.1	Detailed Description	20
4.1.2	Function Documentation	20
4.1.2.1	calc_alea_float(void)	20
4.1.2.2	calc_alea_int(const int min, const int max)	20
4.2	canvas.h File Reference	21
4.2.1	Detailed Description	23
4.2.2	Typedef Documentation	23
4.2.2.1	Canvas	23
4.2.3	Function Documentation	23

4.2.3.1	<code>canvas_blit(Canvas *canvas)</code>	23
4.2.3.2	<code>canvas_clear(Canvas *canvas)</code>	24
4.2.3.3	<code>canvas_collision_canvas(const Canvas *canvas1, const Canvas *canvas2) __attribute__((pure))</code>	24
4.2.3.4	<code>canvas_create(Canvas *canvas, const Point *size, const Point *origin, Canvas *parent)</code>	24
4.2.3.5	<code>canvas_create_from_window(Canvas *canvas, const Window *window)</code>	24
4.2.3.6	<code>canvas_draw_borders_in(Canvas *canvas, const Color *color)</code>	25
4.2.3.7	<code>canvas_draw_borders_out(Canvas *canvas, const Color *color)</code>	25
4.2.3.8	<code>canvas_fill(Canvas *canvas, const Color *color)</code>	25
4.2.3.9	<code>canvas_get_absolute_origin(const Canvas *canvas, Point *absoluteOrigin)</code>	25
4.2.3.10	<code>canvas_is_out_of_parent_bottom(const Canvas *canvas) __attribute__((pure))</code>	25
4.2.3.11	<code>canvas_is_out_of_parent_left(const Canvas *canvas) __attribute__((pure))</code>	26
4.2.3.12	<code>canvas_is_out_of_parent_right(const Canvas *canvas) __attribute__((pure))</code>	26
4.2.3.13	<code>canvas_is_out_of_parent_top(const Canvas *canvas) __attribute__((pure))</code>	26
4.2.3.14	<code>canvas_is_out_of_parent_x(const Canvas *canvas) __attribute__((pure))</code>	26
4.2.3.15	<code>canvas_is_out_of_parent_y(const Canvas *canvas) __attribute__((pure))</code>	27
4.2.3.16	<code>canvas_will_be_out_of_parent_bottom(const Canvas *canvas, const Point *move) __attribute__((pure))</code>	27
4.2.3.17	<code>canvas_will_be_out_of_parent_left(const Canvas *canvas, const Point *move) __attribute__((pure))</code>	27
4.2.3.18	<code>canvas_will_be_out_of_parent_right(const Canvas *canvas, const Point *move) __attribute__((pure))</code>	28
4.2.3.19	<code>canvas_will_be_out_of_parent_top(const Canvas *canvas, const Point *move) __attribute__((pure))</code>	28
4.2.3.20	<code>canvas_will_be_out_of_parent_x(const Canvas *canvas, const Point *move) __attribute__((pure))</code>	28
4.2.3.21	<code>canvas_will_be_out_of_parent_y(const Canvas *canvas, const Point *move) __attribute__((pure))</code>	28
4.3	<code>circle.h</code> File Reference	29
4.3.1	Detailed Description	30
4.3.2	Function Documentation	30
4.3.2.1	<code>circle_draw(const Circle *circle, const Color *color)</code>	30
4.3.2.2	<code>circle_draw_fill(const Circle *circle, const Color *color)</code>	31

4.4	color.h File Reference	31
4.4.1	Detailed Description	32
4.4.2	Function Documentation	33
4.4.2.1	color_get_blue(const Color *color) __attribute__((pure))	33
4.4.2.2	color_get_green(const Color *color) __attribute__((const))	33
4.4.2.3	color_get_red(const Color *color) __attribute__((const))	33
4.4.2.4	color_translate(const Color *color, SDL_Color *sdlColor)	33
4.5	error.h File Reference	33
4.5.1	Detailed Description	35
4.5.2	Function Documentation	35
4.5.2.1	error_quit(void) __attribute__((noreturn))	35
4.6	event.h File Reference	35
4.6.1	Detailed Description	36
4.6.2	Function Documentation	36
4.6.2.1	event_create(Event *newEvent)	36
4.6.2.2	event_update(Event *event)	37
4.7	graphics.h File Reference	37
4.7.1	Detailed Description	38
4.8	image.h File Reference	38
4.8.1	Detailed Description	39
4.8.2	Function Documentation	39
4.8.2.1	image_blit_naive(const Image *image)	39
4.8.2.2	image_blit_scaled(const Image *image)	40
4.8.2.3	image_load(Image *image, const char *pathToImg)	40
4.8.2.4	image_unload(Image *image)	40
4.9	line.h File Reference	40
4.9.1	Detailed Description	43
4.9.2	Function Documentation	43
4.9.2.1	line_draw(const Line *line, const Color *color)	43
4.9.2.2	line_draw_bis(const Line *line, const Color *color)	43

4.9.2.3	<code>line_draw_ter(const Line *line, const Color *color)</code>	43
4.10	mouse.h File Reference	43
4.10.1	Detailed Description	45
4.10.2	Function Documentation	45
4.10.2.1	<code>mouse_hide(void)</code>	45
4.10.2.2	<code>mouse_is_hidden(void)</code>	45
4.10.2.3	<code>mouse_is_shown(void)</code>	45
4.10.2.4	<code>mouse_show(void)</code>	45
4.10.2.5	<code>mouse_wait_click(const Window *window, Point *click)</code>	45
4.11	pixel.h File Reference	46
4.11.1	Function Documentation	47
4.11.1.1	<code>pixel_draw(const Pixel *pixel, const Color *color)</code>	47
4.12	point.h File Reference	48
4.12.1	Detailed Description	49
4.12.2	Function Documentation	49
4.12.2.1	<code>point_are_equals(const Point p1, const Point p2) __attribute__((const))</code>	49
4.12.2.2	<code>point_distance(const Point a, const Point b)</code>	49
4.12.2.3	<code>point_max_x(const Point a, const Point b)</code>	50
4.12.2.4	<code>point_max_y(const Point a, const Point b)</code>	50
4.12.2.5	<code>point_min_x(const Point a, const Point b)</code>	51
4.12.2.6	<code>point_min_y(const Point a, const Point b)</code>	51
4.13	rectangle.h File Reference	51
4.13.1	Function Documentation	53
4.13.1.1	<code>rectangle_contains_absolute_point(const Rectangle *rect, const Point *p)</code>	53
4.13.1.2	<code>rectangle_contains_point(const Rectangle *rect, const Point *p) __attribute__((pure))</code>	53
4.13.1.3	<code>rectangle_draw(const Rectangle *rectangle, const Color *color)</code>	53
4.13.1.4	<code>rectangle_draw_fill(const Rectangle *rectangle, const Color *color)</code>	53
4.14	screen.h File Reference	54
4.14.1	Detailed Description	54
4.14.2	Function Documentation	55

4.14.2.1	<code>screen_get_size(Point *screenSize)</code>	55
4.15	sound.h File Reference	55
4.15.1	Detailed Description	56
4.15.2	Function Documentation	56
4.15.2.1	<code>sound_free(Sound *sound)</code>	56
4.15.2.2	<code>sound_load(const char *pathToFile, Sound *sound)</code>	57
4.15.2.3	<code>sound_pause(void)</code>	57
4.15.2.4	<code>sound_play(const Sound *music)</code>	57
4.15.2.5	<code>sound_play_once(const Sound *music)</code>	57
4.15.2.6	<code>sound_resume(void)</code>	57
4.15.2.7	<code>sound_stop(void)</code>	57
4.16	sphere.h File Reference	58
4.16.1	Detailed Description	59
4.16.2	Function Documentation	59
4.16.2.1	<code>sphere_draw_fill(const Sphere *sphere, const Color *color)</code>	59
4.17	startstop.h File Reference	59
4.17.1	Detailed Description	61
4.17.2	Function Documentation	61
4.17.2.1	<code>graphics_start(const Uint32 flags)</code>	61
4.17.2.2	<code>graphics_stop(void)</code>	61
4.18	window.h File Reference	61
4.18.1	Detailed Description	62
4.18.2	Function Documentation	63
4.18.2.1	<code>window_create(Window *window, char *title, const Point *position, const Point *size, const Uint32 flags)</code>	63
4.18.2.2	<code>window_destroy(Window *window)</code>	64
4.18.2.3	<code>window_update(Window *window)</code>	64
	Index	65

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Canvas	A Canvas is part of a Window or of another Canvas , on which it's possible to draw	5
Circle	A struct used to represent a circle	6
Color	A struct used to represent a RGBA color	7
Event	A struct used to represent events, i.e. user input	8
Image	A struct representing an image	9
Line	A struct used to represent a line segment	10
Pixel	A struct used to represent a pixel	11
Point	A struct used to represent a point	13
Rectangle	A struct used to represent a rectanglec	13
Sound	A struct used to store a sound	15
Sphere	15
Window	16

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

calc.h	Some maths functions	19
canvas.h	Everything related to Canvas	21
circle.h	Everything related to Circle	29
color.h	Everything related to Color	31
error.h	Everything related to errors and warnings handling	33
event.h	Everything related to events, i.e. user input	35
graphics.h	The main lib file	37
image.h	Everything related to Image	38
line.h	Everything related to Line	40
mouse.h	Everything related to the mouse	43
pixel.h	46
point.h	Everything related to Point	48
rectangle.h	51
screen.h	Everything related to the screen	54
sound.h	Everything related to Sound	55
sphere.h	Everything related to Sphere	58
startstop.h	Everything related to graphics' start and stop functions	59
window.h	Everything related to Window	61

Chapter 3

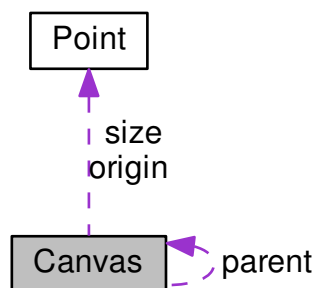
Data Structure Documentation

3.1 Canvas Struct Reference

A [Canvas](#) is part of a [Window](#) or of another [Canvas](#), on which it's possible to draw.

```
#include <canvas.h>
```

Collaboration diagram for Canvas:



Data Fields

- `SDL_Surface *` [surface](#)
- [Point](#) `size`
- [Point](#) `origin`
- `struct Canvas *` [parent](#)

3.1.1 Detailed Description

A [Canvas](#) is part of a [Window](#) or of another [Canvas](#), on which it's possible to draw.

3.1.2 Field Documentation

3.1.2.1 Point Canvas::origin

[Point](#) representing the origin of the [Canvas](#), user can set and get it safely.

3.1.2.2 struct Canvas* Canvas::parent

Pointer to the [Canvas](#) representing the parent of the [Canvas](#), i.e. the one one which it will be blitted, if the [Canvas](#) is the root [Canvas](#) representing the whole [Window](#) it points to NULL.

3.1.2.3 Point Canvas::size

[Point](#) representing the size of the [Canvas](#), usefull to get the value quickly, but user shouldn't change it.

3.1.2.4 SDL_Surface* Canvas::surface

Pointer to the `SDL_Surface` used to store the content of the [Canvas](#), user shouldn't have to touch this.

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

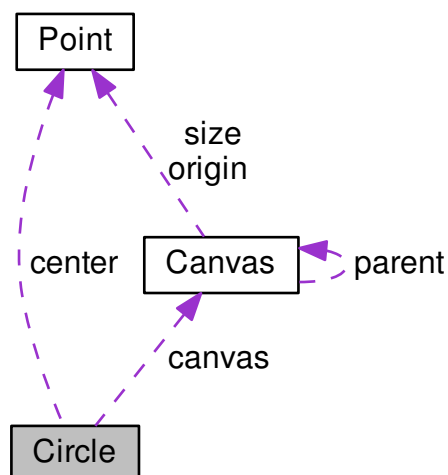
- [canvas.h](#)

3.2 Circle Struct Reference

A struct used to represent a circle.

```
#include <circle.h>
```

Collaboration diagram for Circle:



Data Fields

- [Point](#) [center](#)
- int [radius](#)
- [Canvas](#) * [canvas](#)

3.2.1 Detailed Description

A struct used to represent a circle.

A struct used to represent a sphere.

3.2.2 Field Documentation

3.2.2.1 [Canvas](#)* [Circle::canvas](#)

Pointer to the [Canvas](#) the [Circle](#) belongs to.

3.2.2.2 [Point](#) [Circle::center](#)

[Point](#) representing the center of the circle, must be relative to its [Canvas](#).

3.2.2.3 int [Circle::radius](#)

int representing the radius of the circle.

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

- [circle.h](#)

3.3 Color Struct Reference

A struct used to represent a RGBA color.

```
#include <color.h>
```

Data Fields

- Uint32 [rgb](#)
- Uint8 [alpha](#)

3.3.1 Detailed Description

A struct used to represent a RGBA color.

3.3.2 Field Documentation

3.3.2.1 Uint8 Color::alpha

Uint32 representing the alpha component of the color.

3.3.2.2 Uint32 Color::rgb

Uint32 representing the RGB component of the color.

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

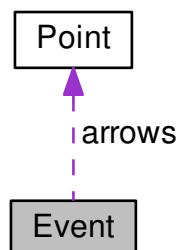
- [color.h](#)

3.4 Event Struct Reference

A struct used to represent events, i.e. user input.

```
#include <event.h>
```

Collaboration diagram for Event:



Data Fields

- bool [quit](#)
- bool [space](#)
- [Point](#) [arrows](#)

3.4.1 Detailed Description

A struct used to represent events, i.e. user input.

3.4.2 Field Documentation

3.4.2.1 Point Event::arrows

[Point](#) representing the arrow keys.

3.4.2.2 bool Event::quit

bool containing true if user press one of the exit key or close the current [Window](#), else contain false.

3.4.2.3 bool Event::space

bool containing true if user press the space key, else contain false.

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

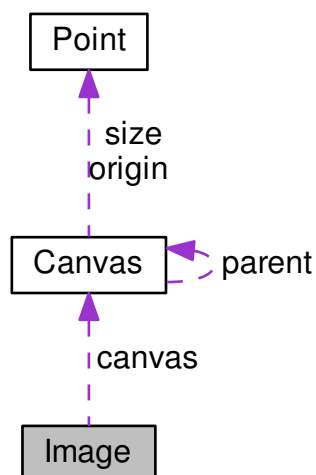
- [event.h](#)

3.5 Image Struct Reference

A struct representing an image.

```
#include <image.h>
```

Collaboration diagram for Image:



Data Fields

- `SDL_Surface *` [surface](#)
- [Canvas](#) * [canvas](#)

3.5.1 Detailed Description

A struct representing an image.

A struct representing a [Window](#).

3.5.2 Field Documentation

3.5.2.1 `Canvas*` `Image::canvas`

Pointer to the [Canvas](#) the [Image](#) belongs to.

3.5.2.2 `SDL_Surface*` `Image::surface`

Pointer to the `SDL_Surface` used to store the content of the image, user shouldn't have to touch this.

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

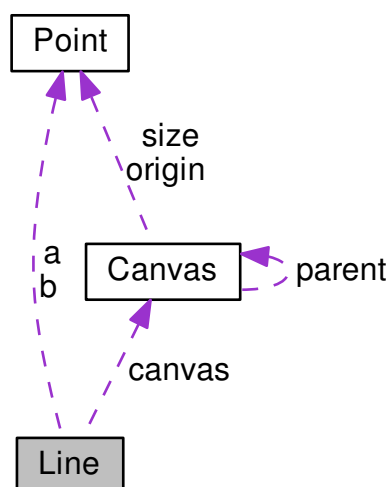
- [image.h](#)

3.6 Line Struct Reference

A struct used to represent a line segment.

```
#include <line.h>
```

Collaboration diagram for Line:



Data Fields

- [Point a](#)
- [Point b](#)
- [Canvas](#) * [canvas](#)

3.6.1 Detailed Description

A struct used to represent a line segment.

3.6.2 Field Documentation

3.6.2.1 Point Line::a

The first point of the line segment.

3.6.2.2 Point Line::b

The last point of the line segment.

3.6.2.3 Canvas* Line::canvas

The [Canvas](#) the [Line](#) belongs to.

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

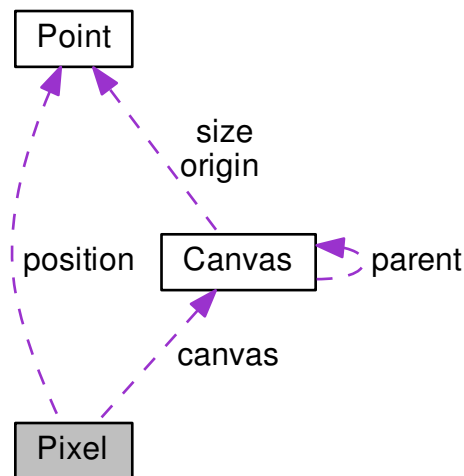
- [line.h](#)

3.7 Pixel Struct Reference

A struct used to represent a pixel.

```
#include <pixel.h>
```

Collaboration diagram for Pixel:



Data Fields

- [Point position](#)
- [Canvas * canvas](#)

3.7.1 Detailed Description

A struct used to represent a pixel.

3.7.2 Field Documentation

3.7.2.1 Canvas* Pixel::canvas

Pointer to the [Canvas](#) the [Pixel](#) belongs to.

3.7.2.2 Point Pixel::position

[Point](#) representing the position of the [Pixel](#).

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

- [pixel.h](#)

3.8 Point Struct Reference

A struct used to represent a point.

```
#include <point.h>
```

Data Fields

- `int x`
- `int y`

3.8.1 Detailed Description

A struct used to represent a point.

3.8.2 Field Documentation

3.8.2.1 `int Point::x`

The value of the point on the x-coordinate.

3.8.2.2 `int Point::y`

The value of the point on the y-coordinate.

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

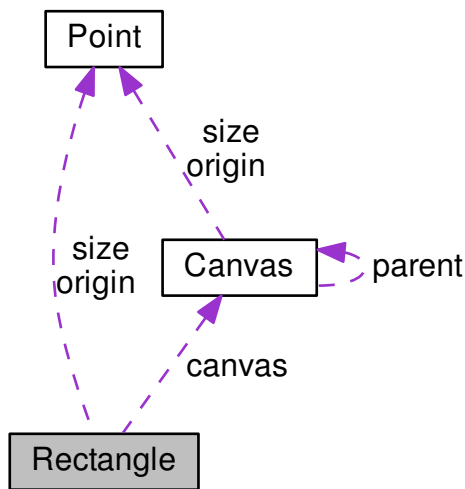
- `point.h`

3.9 Rectangle Struct Reference

A struct used to represent a rectangle.

```
#include <rectangle.h>
```

Collaboration diagram for Rectangle:



Data Fields

- [Point origin](#)
- [Point size](#)
- [Canvas * canvas](#)

3.9.1 Detailed Description

A struct used to represent a rectangle.

3.9.2 Field Documentation

3.9.2.1 `Canvas* Rectangle::canvas`

Pointer to the [Canvas](#) the [Rectangle](#) belongs to.

3.9.2.2 `Point Rectangle::origin`

[Point](#) representing the origin of the [Rectangle](#) on its [Canvas](#).

3.9.2.3 `Point Rectangle::size`

[Point](#) representing the size of the [Canvas](#).

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

- [rectangle.h](#)

3.10 Sound Struct Reference

A struct used to store a sound.

```
#include <sound.h>
```

Data Fields

- `Mix_Music *` [content](#)

3.10.1 Detailed Description

A struct used to store a sound.

3.10.2 Field Documentation

3.10.2.1 `Mix_Music*` `Sound::content`

Pointer to the `Mix_Music` used to store the sound's content.

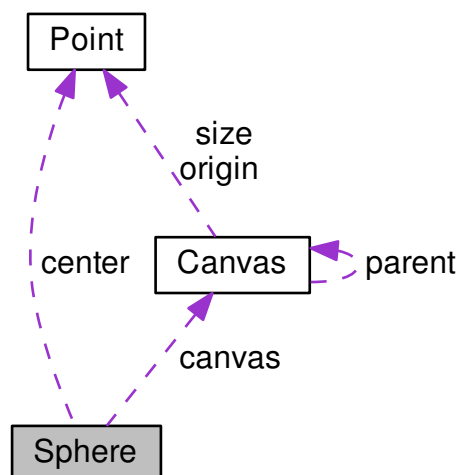
The documentation for this struct was generated from the following file:

- [sound.h](#)

3.11 Sphere Struct Reference

```
#include <sphere.h>
```

Collaboration diagram for Sphere:



Data Fields

- [Point](#) center
- int [radius](#)
- [Canvas](#) * [canvas](#)

3.11.1 Field Documentation

3.11.1.1 [Canvas](#)* [Sphere](#)::[canvas](#)

Pointer to the [Canvas](#) the [Sphere](#) belongs to.

3.11.1.2 [Point](#) [Sphere](#)::[center](#)

[Point](#) representing the center of the sphere, must be relative to its [Canvas](#).

3.11.1.3 int [Sphere](#)::[radius](#)

int representing the radius of the sphere.

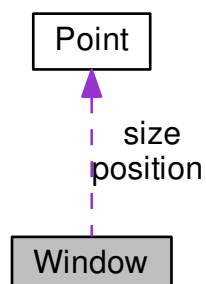
The documentation for this struct was generated from the following file:

- [sphere.h](#)

3.12 Window Struct Reference

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

Collaboration diagram for Window:



Data Fields

- `SDL_Window *` [window](#)
- `char *` [title](#)
- [Point](#) [position](#)
- [Point](#) [size](#)

3.12.1 Field Documentation

3.12.1.1 `Point Window::position`

[Point](#) representing the window's position.

3.12.1.2 `Point Window::size`

[Point](#) representing the window's size.

3.12.1.3 `char* Window::title`

The window's title.

3.12.1.4 `SDL_Window* Window::window`

Pointer to the `SDL_Window` used to store the window.

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

- [window.h](#)

Chapter 4

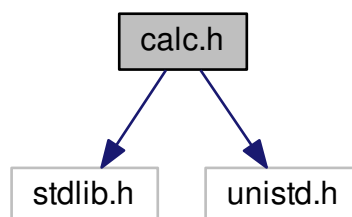
File Documentation

4.1 calc.h File Reference

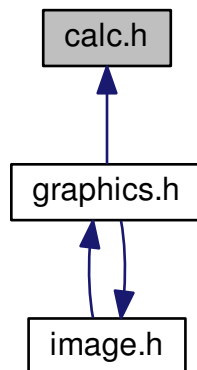
Some maths functions.

```
#include <stdlib.h>
#include <unistd.h>
```

Include dependency graph for calc.h:



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



Functions

- float `calc_alea_float` (void)
Function to get a random float x in [0 ; 1[.
- int `calc_alea_int` (const int min, const int max)
Function to get a random int.

4.1.1 Detailed Description

Some maths functions.

4.1.2 Function Documentation

4.1.2.1 float `calc_alea_float` (void)

Function to get a random float x in [0 ; 1[.

Returns

The random float.

4.1.2.2 int `calc_alea_int` (const int *min*, const int *max*)

Function to get a random int.

Parameters

<i>min</i>	The minimum value for the random int.
<i>max</i>	The maximum value for the random int.

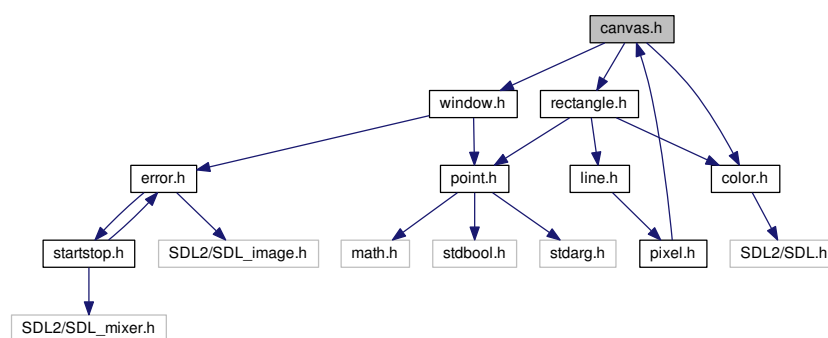
Returns

The random int.

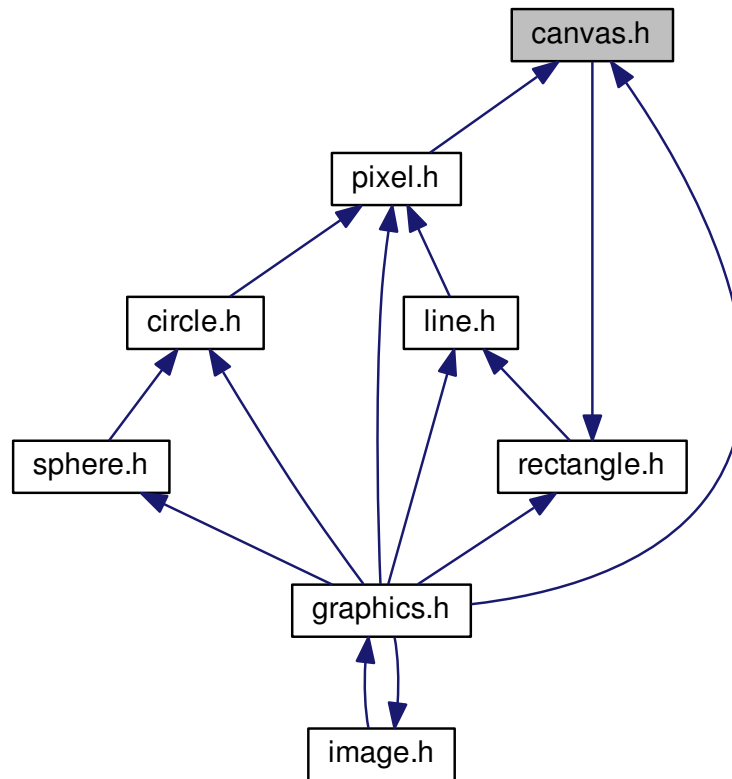
4.2 canvas.h File Reference

Everything related to [Canvas](#).

```
#include "window.h"
#include "color.h"
#include "rectangle.h"
Include dependency graph for canvas.h:
```



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



Data Structures

- struct [Canvas](#)

A [Canvas](#) is part of a [Window](#) or of another [Canvas](#), on which it's possible to draw.

Typedefs

- typedef struct [Canvas](#) [Canvas](#)

Functions

- bool [canvas_collision_canvas](#) (const [Canvas](#) *canvas1, const [Canvas](#) *canvas2) __attribute__((pure))
Function to detect collision between two [Canvas](#).
- bool [canvas_is_out_of_parent_bottom](#) (const [Canvas](#) *canvas) __attribute__((pure))
Function to know if a [Canvas](#) is under its parent.
- bool [canvas_is_out_of_parent_left](#) (const [Canvas](#) *canvas) __attribute__((pure))
Function to know if a [Canvas](#) is out of its parent's left side.
- bool [canvas_is_out_of_parent_right](#) (const [Canvas](#) *canvas) __attribute__((pure))

- Function to know if a [Canvas](#) is out of its parent's right side.*

 - bool [canvas_is_out_of_parent_top](#) (const [Canvas](#) *canvas) `__attribute__((pure))`
- Function to know if a [Canvas](#) is upper its parent's.*

 - bool [canvas_is_out_of_parent_x](#) (const [Canvas](#) *canvas) `__attribute__((pure))`
- Function to know if a [Canvas](#) is outside of its parent's on the X axis.*

 - bool [canvas_is_out_of_parent_y](#) (const [Canvas](#) *canvas) `__attribute__((pure))`
- Function to know if a [Canvas](#) is outside of its parent's on the Y axis.*

 - bool [canvas_will_be_out_of_parent_bottom](#) (const [Canvas](#) *canvas, const [Point](#) *move) `__attribute__((pure))`
- Function to know if a [Canvas](#) will be under its parent after moving its origin.*

 - bool [canvas_will_be_out_of_parent_left](#) (const [Canvas](#) *canvas, const [Point](#) *move) `__attribute__((pure))`
- Function to know if a [Canvas](#) will be out of its parent's left side after moving its origin.*

 - bool [canvas_will_be_out_of_parent_right](#) (const [Canvas](#) *canvas, const [Point](#) *move) `__attribute__((pure))`
- Function to know if a [Canvas](#) will be out of its parent's right side after moving its origin.*

 - bool [canvas_will_be_out_of_parent_top](#) (const [Canvas](#) *canvas, const [Point](#) *move) `__attribute__((pure))`
- Function to know if a [Canvas](#) will be upper its parent after moving its origin.*

 - bool [canvas_will_be_out_of_parent_x](#) (const [Canvas](#) *canvas, const [Point](#) *move) `__attribute__((pure))`
- Function to know if a [Canvas](#) will be outside of its parent on the X axis after moving its origin.*

 - bool [canvas_will_be_out_of_parent_y](#) (const [Canvas](#) *canvas, const [Point](#) *move) `__attribute__((pure))`
- Function to know if a [Canvas](#) will be outside of its parent on the Y axis after moving its origin.*

 - void [canvas_blit](#) ([Canvas](#) *canvas)
- Function to blit a [Canvas](#) on its parent.*

 - void [canvas_create](#) ([Canvas](#) *canvas, const [Point](#) *size, const [Point](#) *origin, [Canvas](#) *parent)
- Function to create a [Canvas](#).*

 - void [canvas_clear](#) ([Canvas](#) *canvas)
- Function to clear a [Canvas](#), i.e. filling it with black.*

 - void [canvas_create_from_window](#) ([Canvas](#) *canvas, const [Window](#) *window)
- Function to create a [Canvas](#) from a [Window](#), it will fill the whole window.*

 - void [canvas_draw_borders_in](#) ([Canvas](#) *canvas, const [Color](#) *color)
- Function to draw a 1 pixel border inside of a [Canvas](#).*

 - void [canvas_draw_borders_out](#) ([Canvas](#) *canvas, const [Color](#) *color)
- Function to draw a 1 pixel border outside of a [Canvas](#).*

 - void [canvas_fill](#) ([Canvas](#) *canvas, const [Color](#) *color)
- Function to fill a [Canvas](#) with a [Color](#).*

 - void [canvas_get_absolute_origin](#) (const [Canvas](#) *canvas, [Point](#) *absoluteOrigin)
- Function to get the origin of a [Canvas](#) on the [Window](#), instead of on its parent.*

4.2.1 Detailed Description

Everything related to [Canvas](#).

4.2.2 Typedef Documentation

4.2.2.1 typedef struct Canvas Canvas

4.2.3 Function Documentation

4.2.3.1 void canvas_blit (Canvas * canvas)

Function to blit a [Canvas](#) on its parent.

Parameters

<i>canvas</i>	A pointer to the Canvas to blit.
---------------	--

4.2.3.2 void canvas_clear ([Canvas](#) * *canvas*)

Function to clear a [Canvas](#), i.e. filling it with black.

Parameters

<i>canvas</i>	A pointer to the Canvas to clear.
---------------	---

4.2.3.3 bool canvas_collision_canvas (const [Canvas](#) * *canvas1*, const [Canvas](#) * *canvas2*)

Function to detect collision between two [Canvas](#).

Parameters

<i>canvas1</i>	A pointer to the first Canvas .
<i>canvas2</i>	A pointer to the second Canvas .

Returns

If the two [Canvas](#) collide returns true, else, returns false.

4.2.3.4 void canvas_create ([Canvas](#) * *canvas*, const [Point](#) * *size*, const [Point](#) * *origin*, [Canvas](#) * *parent*)

Function to create a [Canvas](#).

Parameters

<i>canvas</i>	A pointer to the Canvas to create.
<i>size</i>	A pointer to a Point representing the wanted size for the Canvas .
<i>origin</i>	A pointer to a Point representing the wanted origin for the Canvas .
<i>parent</i>	A pointer to the Canvas wanted as the parent of the Canvas to create.

4.2.3.5 void canvas_create_from_window ([Canvas](#) * *canvas*, const [Window](#) * *window*)

Function to create a [Canvas](#) from a [Window](#), it will fill the whole window.

Parameters

<i>canvas</i>	A pointer to the Canvas to create.
<i>window</i>	A pointer to the Window from which the Canvas should be created.

4.2.3.6 void canvas_draw_borders_in (Canvas * *canvas*, const Color * *color*)

Function to draw a 1 pixel border inside of a [Canvas](#).

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>color</i>	A pointer to the Color wanted for the border.

4.2.3.7 void canvas_draw_borders_out (Canvas * *canvas*, const Color * *color*)

Function to draw a 1 pixel border outside of a [Canvas](#).

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>color</i>	A pointer to the Color wanted for the border.

4.2.3.8 void canvas_fill (Canvas * *canvas*, const Color * *color*)

Function to fill a [Canvas](#) with a [Color](#).

Parameters

<i>canvas</i>	A pointer to the Canvas to fill.
<i>color</i>	A pointer to the Color wanted to fill the Canvas .

4.2.3.9 void canvas_get_absolute_origin (const Canvas * *canvas*, Point * *absoluteOrigin*)

Function to get the origin of a [Canvas](#) on the [Window](#), instead of on its parent.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>absoluteOrigin</i>	A pointer to the Point in which the origin will be stored.

4.2.3.10 bool canvas_is_out_of_parent_bottom (const Canvas * *canvas*)

Function to know if a [Canvas](#) is under its parent.

Parameters

<i>canvas</i>	A pointer to the Canvas .
---------------	---

Returns

If the [Canvas](#) is under its parent, returns true, else, returns false.

4.2.3.11 bool canvas_is_out_of_parent_left (const Canvas * canvas)

Function to know if a [Canvas](#) is out of its parent's left side.

Parameters

<i>canvas</i>	A pointer to the Canvas .
---------------	---

Returns

If the [Canvas](#) is out of its parent's left side, returns true, else, returns false.

4.2.3.12 bool canvas_is_out_of_parent_right (const Canvas * canvas)

Function to know if a [Canvas](#) is out of its parent's right side.

Parameters

<i>canvas</i>	A pointer to the Canvas .
---------------	---

Returns

If the [Canvas](#) is out of its parent's right side, returns true, else, returns false.

4.2.3.13 bool canvas_is_out_of_parent_top (const Canvas * canvas)

Function to know if a [Canvas](#) is upper its parent's.

Parameters

<i>canvas</i>	A pointer to the Canvas .
---------------	---

Returns

If the canvas is upper, returns true, else, returns false.

4.2.3.14 bool canvas_is_out_of_parent_x (const Canvas * canvas)

Function to know if a [Canvas](#) is outside of its parent's on the X axis.

Parameters

<i>canvas</i>	A pointer to the Canvas .
---------------	---

Returns

If the [Canvas](#) is outside, returns true, else, returns false.

4.2.3.15 `bool canvas_is_out_of_parent_y (const Canvas * canvas)`

Function to know if a [Canvas](#) is outside of its parent's on the Y axis.

Parameters

<i>canvas</i>	A pointer to the Canvas .
---------------	---

Returns

If the [Canvas](#) is outside, returns true, else, returns false.

4.2.3.16 `bool canvas_will_be_out_of_parent_bottom (const Canvas * canvas, const Point * move)`

Function to know if a [Canvas](#) will be under its parent after moving its origin.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>move</i>	A pointer to the Point representing the origin's move.

Returns

If the [Canvas](#) will be under its parent, returns true, else, returns false.

4.2.3.17 `bool canvas_will_be_out_of_parent_left (const Canvas * canvas, const Point * move)`

Function to know if a [Canvas](#) will be out of its parent's left side after moving its origin.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>move</i>	A pointer to the Point representing the origin's move.

Returns

If the [Canvas](#) will be out of its parent's left side, returns true, else, returns false.

4.2.3.18 `bool canvas_will_be_out_of_parent_right (const Canvas * canvas, const Point * move)`

Function to know if a [Canvas](#) will be out of its parent's right side after moving its origin.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>move</i>	A pointer to the Point representing the origin's move.

Returns

If the [Canvas](#) will be out of its parent's right side, returns true, else, returns false.

4.2.3.19 `bool canvas_will_be_out_of_parent_top (const Canvas * canvas, const Point * move)`

Function to know if a [Canvas](#) will be upper its parent after moving its origin.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>move</i>	A pointer to the point representing the origin's move.

Returns

If the [Canvas](#) will be upper its parent, returns true, else, returns false.

4.2.3.20 `bool canvas_will_be_out_of_parent_x (const Canvas * canvas, const Point * move)`

Function to know if a [Canvas](#) will be outside of its parent on the X axis after moving its origin.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>move</i>	A pointer to the point representing the origin's move.

Returns

If the [Canvas](#) will be outside of its parent on the X axis, returns true, else, returns false.

4.2.3.21 `bool canvas_will_be_out_of_parent_y (const Canvas * canvas, const Point * move)`

Function to know if a [Canvas](#) will be outside of its parent on the Y axis after moving its origin.

Parameters

<i>canvas</i>	A pointer to the Canvas .
<i>move</i>	A pointer to the point representing the origin's move.

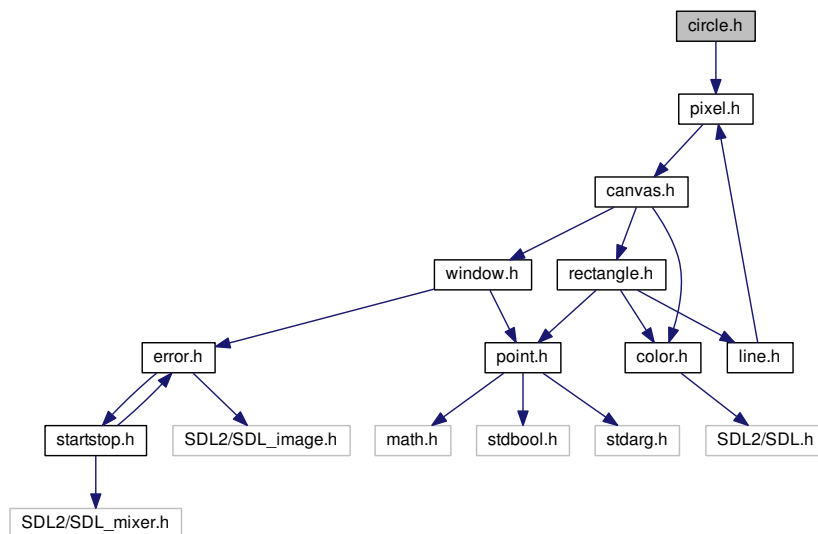
Returns

If the [Canvas](#) will be outside of its parent on the Y axis, returns true, else, returns false.

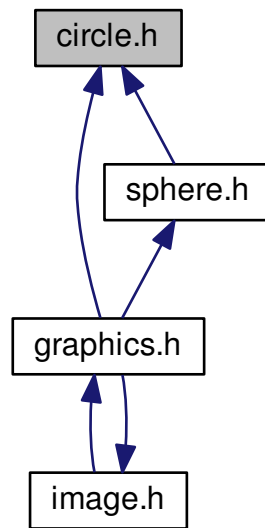
4.3 circle.h File Reference

Everything related to [Circle](#).

```
#include "pixel.h"
Include dependency graph for circle.h:
```



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



Data Structures

- struct [Circle](#)
A struct used to represent a circle.

Functions

- void [circle_draw](#) (const [Circle](#) *circle, const [Color](#) *color)
Function to draw a [Circle](#).
- void [circle_draw_fill](#) (const [Circle](#) *circle, const [Color](#) *color)
Function to draw a filled [Circle](#).

4.3.1 Detailed Description

Everything related to [Circle](#).

4.3.2 Function Documentation

4.3.2.1 void circle_draw (const Circle * circle, const Color * color)

Function to draw a [Circle](#).

Parameters

<i>circle</i>	A pointer to the Circle to draw.
<i>color</i>	A pointer to the Color to use to draw the Circle .

4.3.2.2 void circle_draw_fill (const [Circle](#) * *circle*, const [Color](#) * *color*)

Function to draw a filled [Circle](#).

Parameters

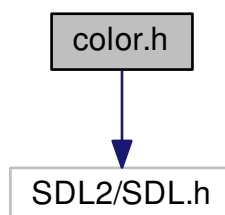
<i>circle</i>	A pointer to the Circle to draw.
<i>color</i>	A pointer to the Color to use to draw the Circle .

4.4 color.h File Reference

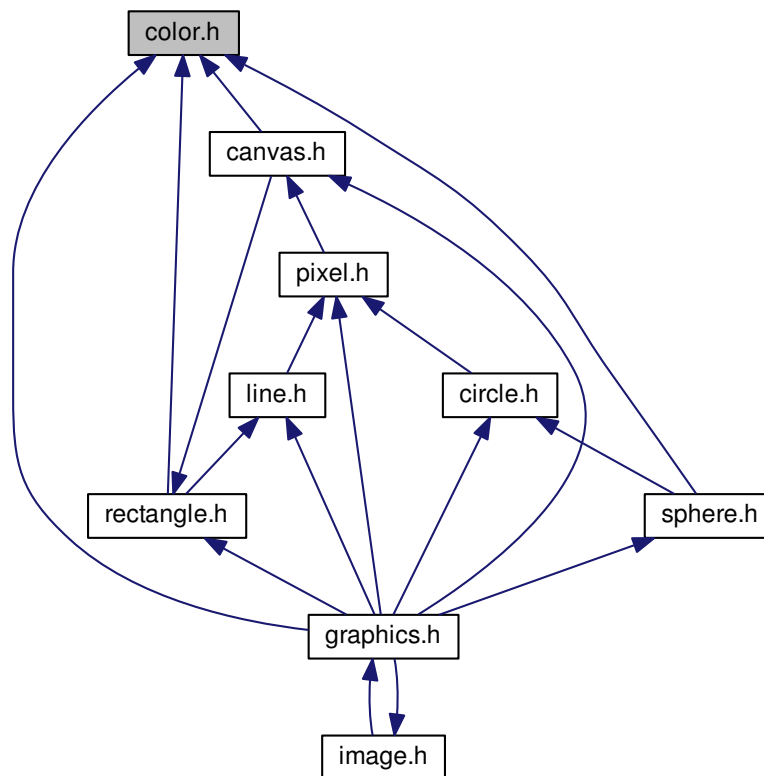
Everything related to [Color](#).

```
#include <SDL2/SDL.h>
```

Include dependency graph for color.h:



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



Data Structures

- struct [Color](#)

A struct used to represent a RGBA color.

Functions

- void [color_translate](#) (const [Color](#) *color, SDL_Color *sdlColor)
- Uint8 [color_get_red](#) (const [Color](#) *color) __attribute__((const))
Function to get the red component of a [Color](#).
- Uint8 [color_get_green](#) (const [Color](#) *color) __attribute__((const))
Function to get the green component of a [Color](#).
- Uint8 [color_get_blue](#) (const [Color](#) *color) __attribute__((pure))
Function to get the blue component of a [Color](#).

4.4.1 Detailed Description

Everything related to [Color](#).

4.4.2 Function Documentation

4.4.2.1 Uint8 color_get_blue (const Color * color)

Function to get the blue component of a [Color](#).

Parameters

<i>canvas1</i>	A pointer to the Color .
----------------	--

Returns

The blue component in a Uint8.

4.4.2.2 Uint8 color_get_green (const Color * color) const

Function to get the green component of a [Color](#).

Parameters

<i>canvas1</i>	A pointer to the Color .
----------------	--

Returns

The green component in a Uint8.

4.4.2.3 Uint8 color_get_red (const Color * color) const

Function to get the red component of a [Color](#).

Parameters

<i>canvas1</i>	A pointer to the Color .
----------------	--

Returns

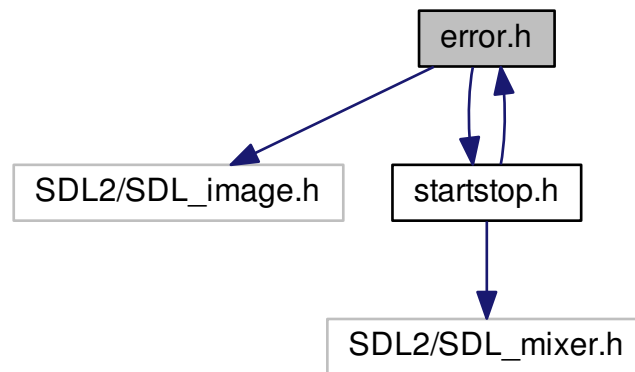
The red component in a Uint8.

4.4.2.4 void color_translate (const Color * color, SDL_Color * sdlColor)

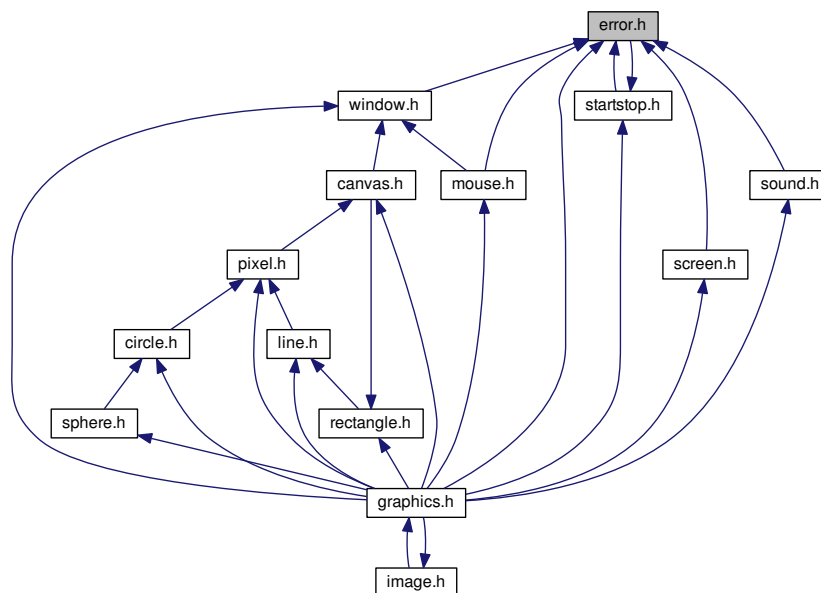
4.5 error.h File Reference

Everything related to errors and warnings handling.

```
#include <SDL2/SDL_image.h>
#include "startstop.h"
Include dependency graph for error.h:
```



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



Functions

- void `error_quit` (void) `__attribute__((noreturn))`

Function to quit after an error, will stop graphics and SDL components and stop the program.

4.5.1 Detailed Description

Everything related to errors and warnings handling.

4.5.2 Function Documentation

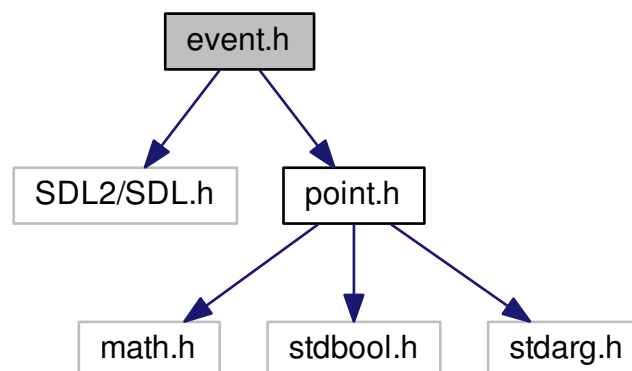
4.5.2.1 void error_quit (void)

Function to quit after an error, will stop graphics and SDL components and stop the program.

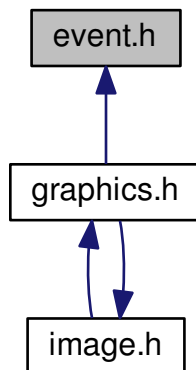
4.6 event.h File Reference

Everything related to events, i.e. user input.

```
#include <SDL2/SDL.h>
#include "point.h"
Include dependency graph for event.h:
```



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



Data Structures

- struct [Event](#)
A struct used to represent events, i.e. user input.

Functions

- void [event_create](#) ([Event](#) *newEvent)
Function to create an [Event](#).
- void [event_update](#) ([Event](#) *event)
Function to update an [Event](#).

4.6.1 Detailed Description

Everything related to events, i.e. user input.

4.6.2 Function Documentation

4.6.2.1 void [event_create](#) ([Event](#) * *newEvent*)

Function to create an [Event](#).

Parameters

<i>newEvent</i>	A pointer to the Event to create.
-----------------	---

4.6.2.2 void event_update (Event * event)

Function to update an [Event](#).

Parameters

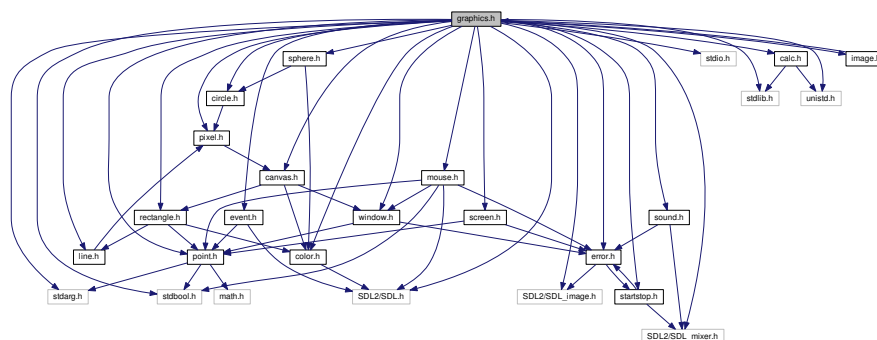
<i>newEvent</i>	A pointer to the Event to update.
-----------------	---

4.7 graphics.h File Reference

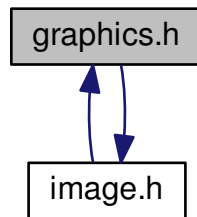
The main lib file.

```
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
#include "point.h"
#include "pixel.h"
#include "canvas.h"
#include "line.h"
#include "window.h"
#include "screen.h"
#include "color.h"
#include "circle.h"
#include "sound.h"
#include "calc.h"
#include "rectangle.h"
#include "event.h"
#include "sphere.h"
#include "image.h"
#include "error.h"
#include "startstop.h"
#include "mouse.h"
```

Include dependency graph for graphics.h:



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



4.7.1 Detailed Description

The main lib file.

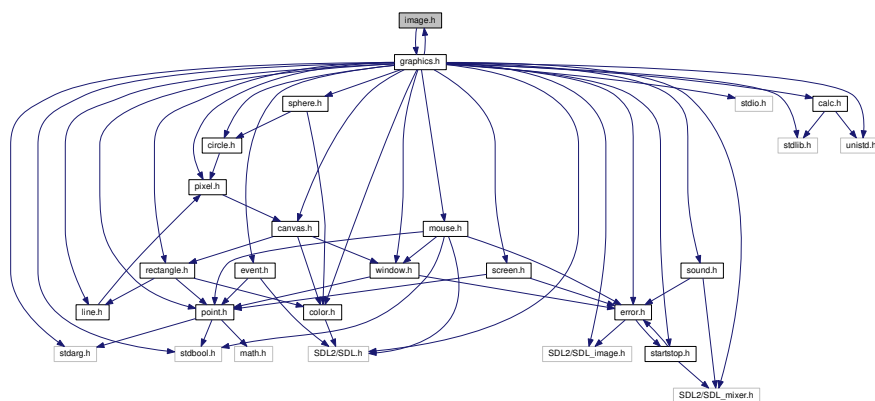
It's the file to include when using the lib in a program. It includes all the others headers and dependencies.

4.8 image.h File Reference

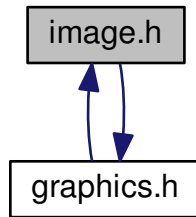
Everything related to [Image](#).

```
#include "graphics.h"
```

Include dependency graph for image.h:



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



Data Structures

- struct [Image](#)
A struct representing an image.

Functions

- void [image_blit_naive](#) (const [Image](#) *image)
Function to blit an [Image](#) on its [Canvas](#), it will be blitted "as is", even if the [Image](#) is bigger than its [Canvas](#).
- void [image_blit_scaled](#) (const [Image](#) *image)
Function to blit an [Image](#) on its [Canvas](#), it will be scaled, i.e. will fill the [Canvas](#) perfectly.
- void [image_load](#) ([Image](#) *image, const char *pathToImg)
Function to load an image into an [Image](#) struct.
- void [image_unload](#) ([Image](#) *image)
Function to unload an [Image](#), i.e. to free it.

4.8.1 Detailed Description

Everything related to [Image](#).

4.8.2 Function Documentation

4.8.2.1 void image_blit_naive (const [Image](#) * image)

Function to blit an [Image](#) on its [Canvas](#), it will be blitted "as is", even if the [Image](#) is bigger than its [Canvas](#).

Parameters

<i>image</i>	A pointer to the Image to blit.
--------------	---

4.8.2.2 void image_blit_scaled (const Image * image)

Function to blit an [Image](#) on its [Canvas](#), it will be scaled, i.e. will fill the [Canvas](#) perfectly.

Parameters

<i>image</i>	A pointer to the Image to blit.
--------------	---

4.8.2.3 void image_load (Image * image, const char * pathToImg)

Function to load an image into an [Image](#) struct.

Parameters

<i>image</i>	A pointer to the Image used to store the loaded image.
<i>pathToImg</i>	The path to the image to load.

4.8.2.4 void image_unload (Image * image)

Function to unload an [Image](#), i.e. to free it.

Parameters

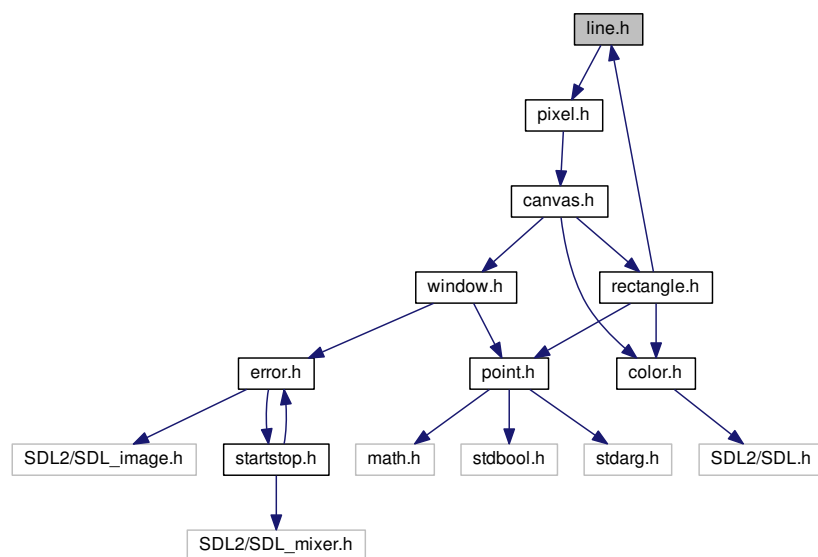
<i>image</i>	A pointer to the Image to unload.
--------------	---

4.9 line.h File Reference

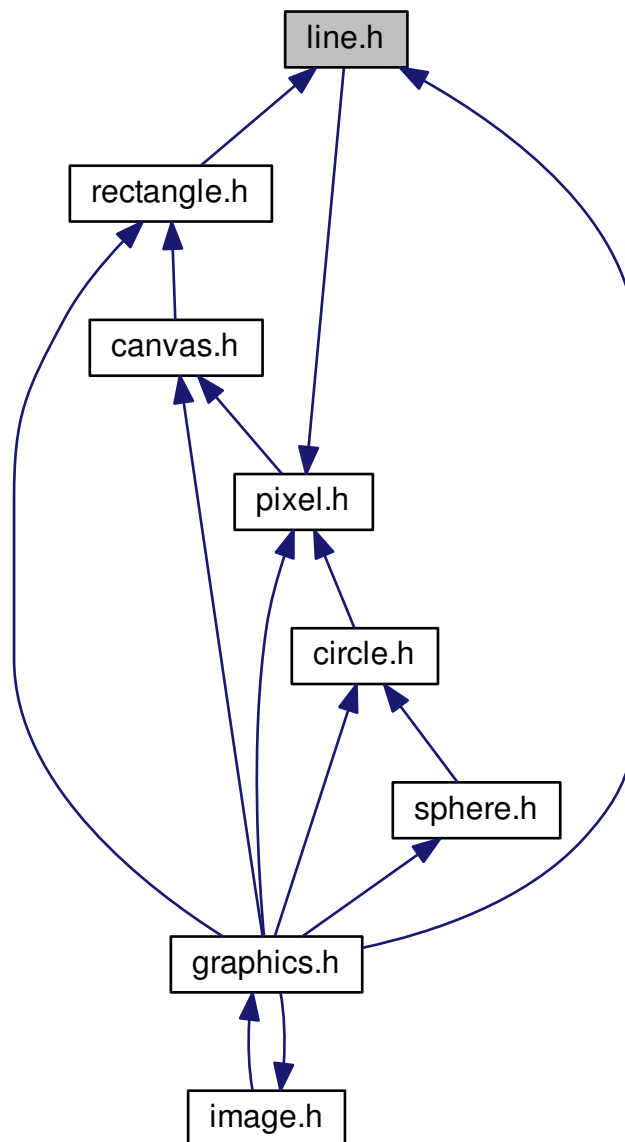
Everything related to [Line](#).

```
#include "pixel.h"
```

Include dependency graph for line.h:



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



Data Structures

- struct [Line](#)
A struct used to represent a line segment.

Functions

- void [line_draw](#) (const [Line](#) *line, const [Color](#) *color)
Function to draw a line. The best one.

- void [line_draw_bis](#) (const [Line](#) *line, const [Color](#) *color)
Function to draw a line. Use floats and thus, is slower than line_draw.
- void [line_draw_ter](#) (const [Line](#) *line, const [Color](#) *color)
Function to draw a line. Is very fast, but, draws lines with blanks.

4.9.1 Detailed Description

Everything related to [Line](#).

Everything related to [Pixel](#).

4.9.2 Function Documentation

4.9.2.1 void line_draw (const [Line](#) * line, const [Color](#) * color)

Function to draw a line. The best one.

Parameters

<i>line</i>	A pointer to the Line to draw.
<i>color</i>	A pointer to the Color to use to draw the Line .

4.9.2.2 void line_draw_bis (const [Line](#) * line, const [Color](#) * color)

Function to draw a line. Use floats and thus, is slower than line_draw.

Parameters

<i>line</i>	A pointer to the Line to draw.
<i>color</i>	A pointer to the Color to use to draw the Line .

4.9.2.3 void line_draw_ter (const [Line](#) * line, const [Color](#) * color)

Function to draw a line. Is very fast, but, draws lines with blanks.

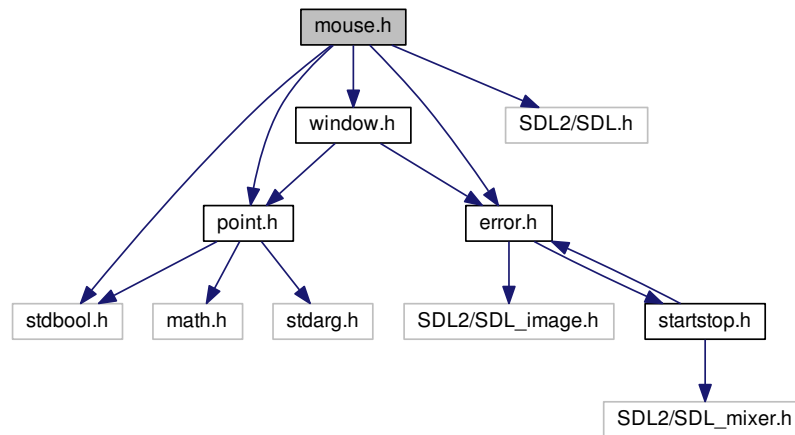
Parameters

<i>line</i>	A pointer to the Line to draw.
<i>color</i>	A pointer to the Color to use to draw the Line .

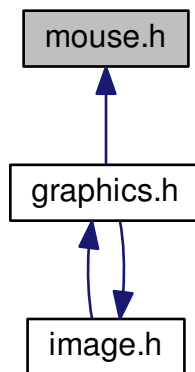
4.10 mouse.h File Reference

Everything related to the mouse.

```
#include <stdbool.h>
#include <SDL2/SDL.h>
#include "error.h"
#include "point.h"
#include "window.h"
Include dependency graph for mouse.h:
```



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



Functions

- void `mouse_hide` (void)
Function to hide the mouse cursor.
- void `mouse_show` (void)
- void `mouse_wait_click` (const `Window` *window, `Point` *click)

Function to wait a click and store it into a [Point](#).

- bool [mouse_is_hidden](#) (void)

Function to know if the cursor is hidden.

- bool [mouse_is_shown](#) (void)

Function to know if the cursor is shown.

4.10.1 Detailed Description

Everything related to the mouse.

4.10.2 Function Documentation

4.10.2.1 void mouse_hide (void)

Function to hide the mouse cursor.

Function to show the mouse cursor.

4.10.2.2 bool mouse_is_hidden (void)

Function to know if the cursor is hidden.

Returns

Returns true if the cursor is hidden, false otherwise.

4.10.2.3 bool mouse_is_shown (void)

Function to know if the cursor is shown.

Returns

Returns true if the cursor is shown, false otherwise.

4.10.2.4 void mouse_show (void)

4.10.2.5 void mouse_wait_click (const Window * window, Point * click)

Function to wait a click and store it into a [Point](#).

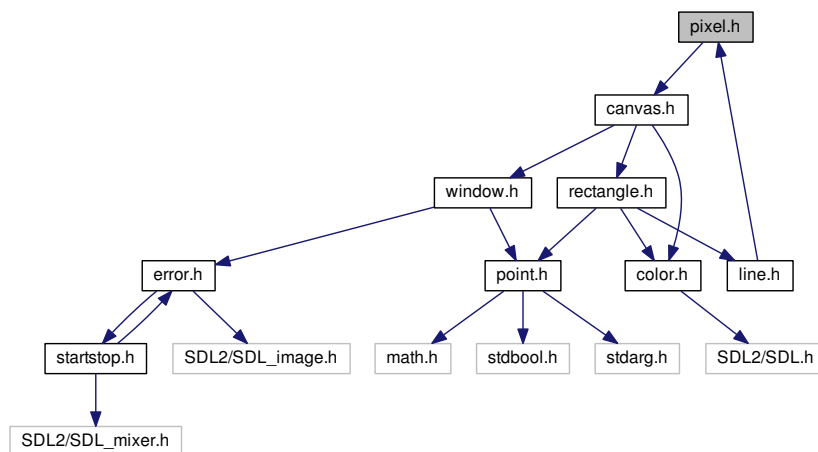
Parameters

<i>window</i>	A pointer to the Window on which the click is waited.
<i>color</i>	A pointer to the Point on which the click position must be stored.

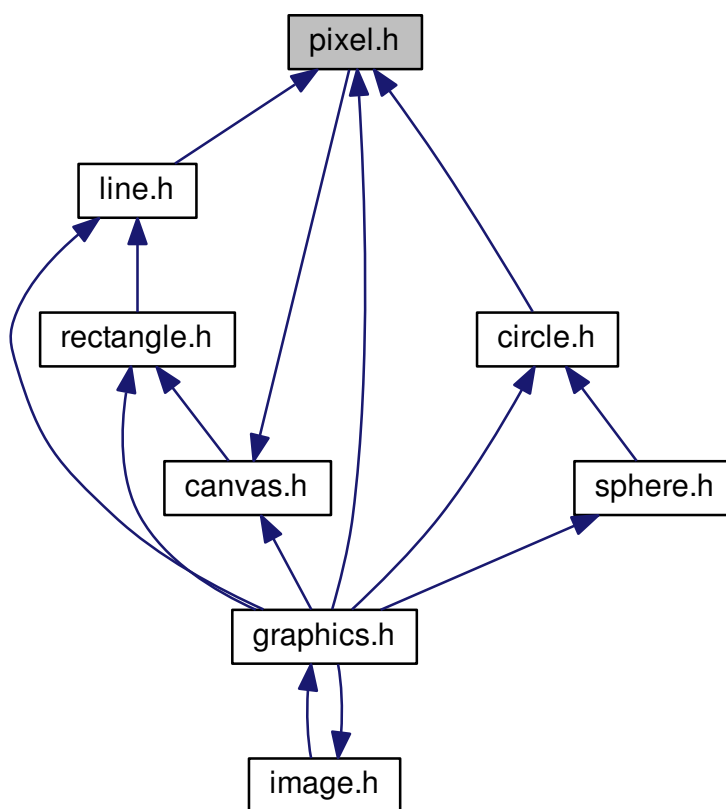
4.11 pixel.h File Reference

```
#include "canvas.h"
```

Include dependency graph for pixel.h:



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



Data Structures

- struct [Pixel](#)
A struct used to represent a pixel.

Functions

- void [pixel_draw](#) (const [Pixel](#) *pixel, const [Color](#) *color)
Function to draw a pixel.

4.11.1 Function Documentation

4.11.1.1 void pixel_draw (const Pixel * pixel, const Color * color)

Function to draw a pixel.

Parameters

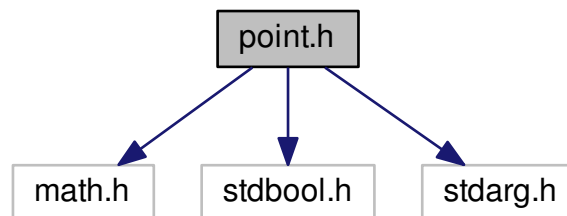
<i>line</i>	A pointer to the Pixel to draw.
<i>color</i>	A pointer to the Color to use to draw the Pixel .

4.12 point.h File Reference

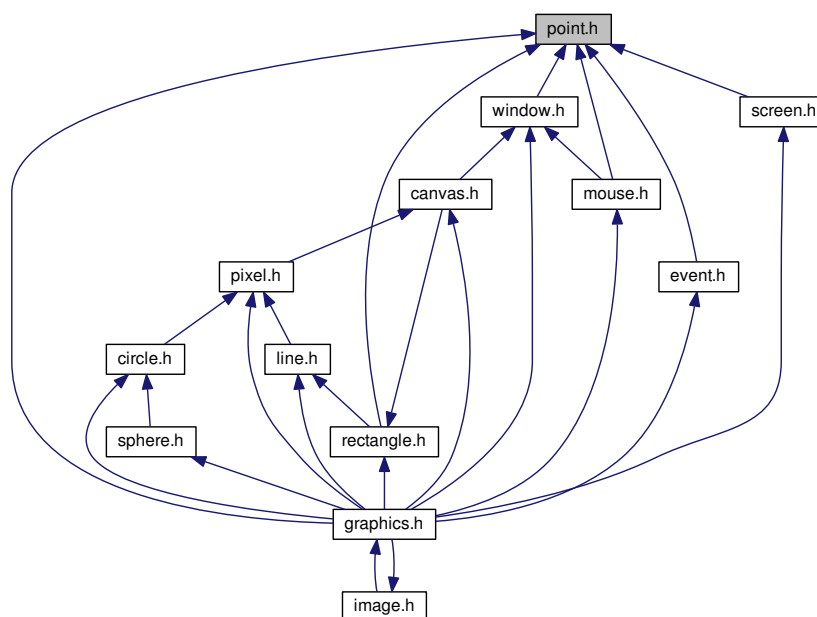
Everything related to [Point](#).

```
#include <math.h>
#include <stdbool.h>
#include <stdarg.h>
```

Include dependency graph for point.h:



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



Data Structures

- struct [Point](#)

A struct used to represent a point.

Functions

- bool [point_are_equals](#) (const [Point](#) p1, const [Point](#) p2) `__attribute__((const))`
Function to know if two [Point](#) are equals.
- int [point_distance](#) (const [Point](#) a, const [Point](#) b)
Function to get the distance between two [Point](#).
- [Point point_max_x](#) (const [Point](#) a, const [Point](#) b)
Function to compare two [Point](#) and getting the one with the biggest x.
- [Point point_max_y](#) (const [Point](#) a, const [Point](#) b)
Function to compare two [Point](#) and getting the one with the biggest y.
- [Point point_min_x](#) (const [Point](#) a, const [Point](#) b)
- [Point point_min_y](#) (const [Point](#) a, const [Point](#) b)

4.12.1 Detailed Description

Everything related to [Point](#).

Everything related to [Rectangle](#).

4.12.2 Function Documentation

4.12.2.1 bool point_are_equals (const [Point](#) p1, const [Point](#) p2) const

Function to know if two [Point](#) are equals.

Parameters

<i>p1</i>	The first Point .
<i>p2</i>	The second Point .

Returns

Return true if they're equals, false otherwise.

4.12.2.2 int point_distance (const [Point](#) a, const [Point](#) b)

Function to get the distance between two [Point](#).

Parameters

<i>a</i>	The first Point .
<i>b</i>	The second Point .

Returns

The distance between the two [Point](#), in an int.

4.12.2.3 Point point_max_x (const Point a, const Point b)

Function to compare two [Point](#) and getting the one with the biggest x.

Function to compare two [Point](#) and getting the one with the smallest y.

Function to compare two [Point](#) and getting the one with the smallest x.

Parameters

<i>a</i>	The first Point .
<i>b</i>	The second Point .

Returns

The [Point](#) with the biggest x.

Parameters

<i>a</i>	The first Point .
<i>b</i>	The second Point .

Returns

The [Point](#) with the smallest x.

Parameters

<i>a</i>	The first Point .
<i>b</i>	The second Point .

Returns

The [Point](#) with the smallest y.

4.12.2.4 Point point_max_y (const Point a, const Point b)

Function to compare two [Point](#) and getting the one with the biggest y.

Parameters

<i>a</i>	The first Point .
<i>b</i>	The second Point .

Returns

The [Point](#) with the biggest y.

4.12.2.5 `Point point_min_x (const Point a, const Point b)`

4.12.2.6 `Point point_min_y (const Point a, const Point b)`

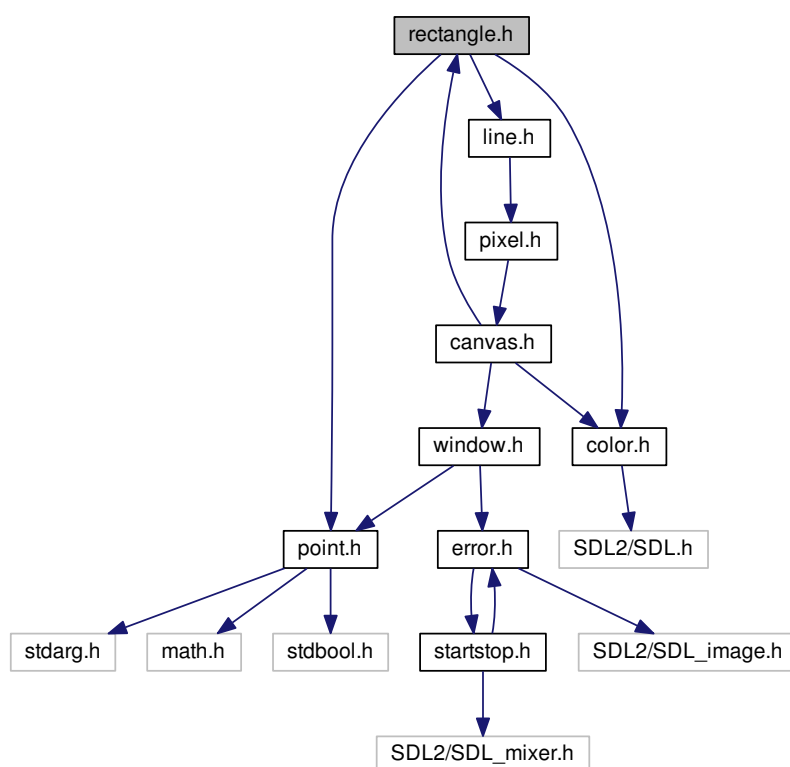
4.13 rectangle.h File Reference

```
#include "point.h"
```

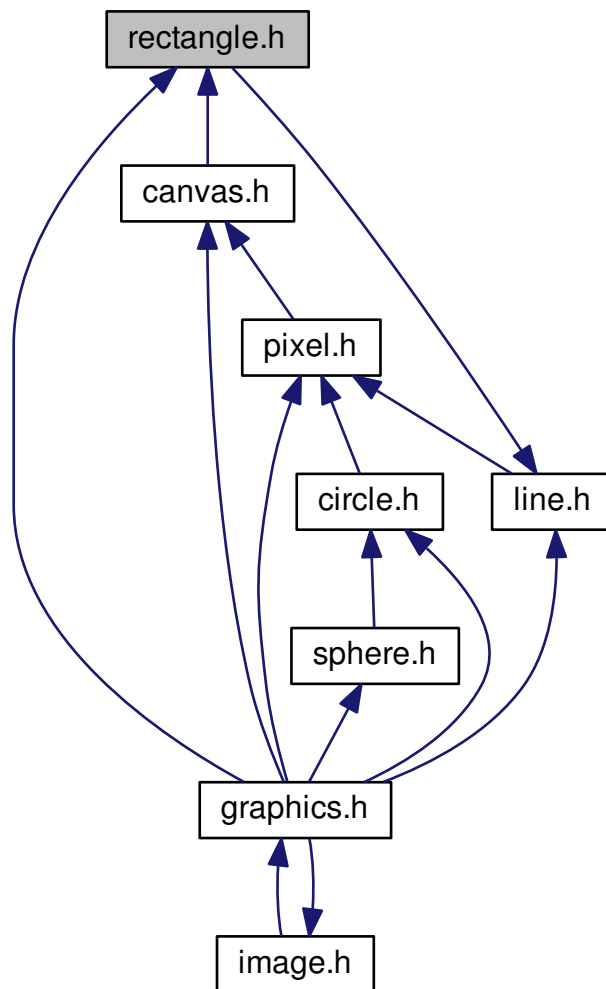
```
#include "line.h"
```

```
#include "color.h"
```

Include dependency graph for rectangle.h:



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



Data Structures

- struct [Rectangle](#)
A struct used to represent a rectangle.

Functions

- void [rectangle_draw](#) (const [Rectangle](#) *rectangle, const [Color](#) *color)
Function to draw a [Rectangle](#).
- void [rectangle_draw_fill](#) (const [Rectangle](#) *rectangle, const [Color](#) *color)
Function to draw a filled [Rectangle](#).
- bool [rectangle_contains_point](#) (const [Rectangle](#) *rect, const [Point](#) *p) __attribute__((pure))
Function to know if a rectangle contains a [Point](#).
- bool [rectangle_contains_absolute_point](#) (const [Rectangle](#) *rect, const [Point](#) *p)
Function to know if a rectangle contains a [Point](#), when the point have absolute coordinates, i.e. relative to the current [Window](#).

4.13.1 Function Documentation

4.13.1.1 `bool rectangle_contains_absolute_point (const Rectangle * rect, const Point * p)`

Function to know if a rectangle contains a [Point](#), when the point have absolute coordinates, i.e. relative to the current [Window](#).

Parameters

<i>rect</i>	A pointer to the Rectangle .
<i>p</i>	A pointer to the Point .

Returns

Returns true if the [Rectangle](#) contains the [Point](#), false otherwise.

4.13.1.2 `bool rectangle_contains_point (const Rectangle * rect, const Point * p)`

Function to know if a rectangle contains a [Point](#).

Parameters

<i>rect</i>	A pointer to the Rectangle .
<i>p</i>	A pointer to the Point .

Returns

Returns true if the [Rectangle](#) contains the [Point](#), false otherwise.

4.13.1.3 `void rectangle_draw (const Rectangle * rectangle, const Color * color)`

Function to draw a [Rectangle](#).

Parameters

<i>circle</i>	A pointer to the Rectangle to draw.
<i>color</i>	A pointer to the Color to use to draw the Rectangle .

4.13.1.4 `void rectangle_draw_fill (const Rectangle * rectangle, const Color * color)`

Function to draw a filled [Rectangle](#).

Parameters

<i>circle</i>	A pointer to the Rectangle to draw.
<i>color</i>	A pointer to the Color to use to draw the Rectangle .

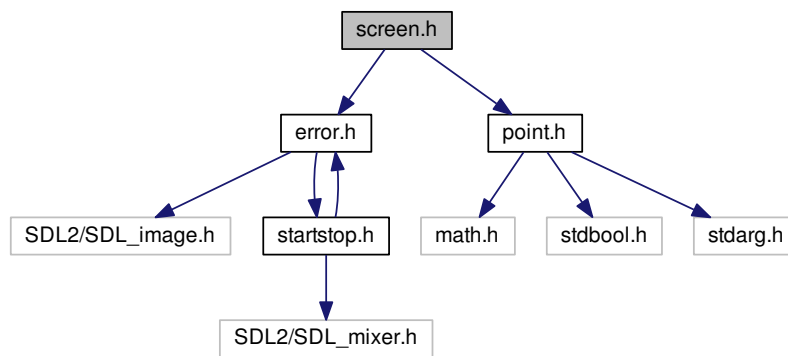
4.14 screen.h File Reference

Everything related to the screen.

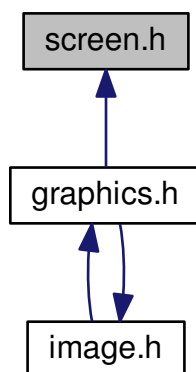
```
#include "error.h"
```

```
#include "point.h"
```

Include dependency graph for screen.h:



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



Functions

- void [screen_get_size](#) ([Point](#) *screenSize)
Function to get the screen's size.

4.14.1 Detailed Description

Everything related to the screen.

4.14.2 Function Documentation

4.14.2.1 void screen_get_size (Point * screenSize)

Function to get the screen's size.

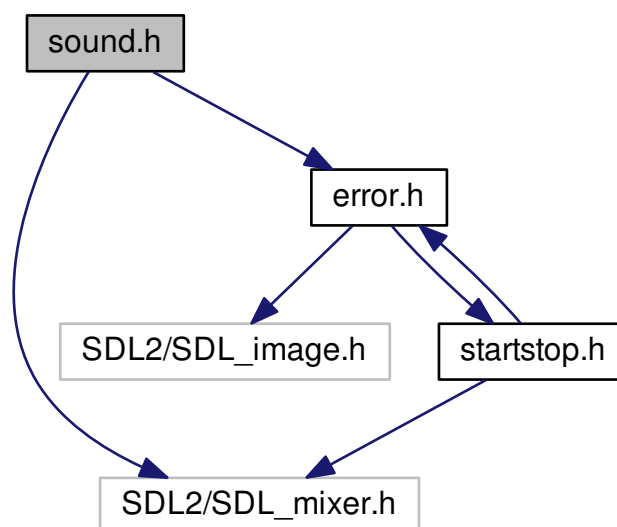
Parameters

<i>screenSize</i>	A pointer to the Point in which the screen's size must be stored.
-------------------	---

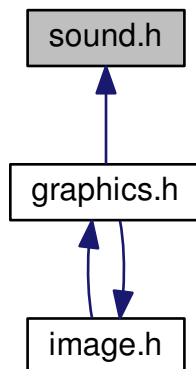
4.15 sound.h File Reference

Everything related to [Sound](#).

```
#include <SDL2/SDL_mixer.h>
#include "error.h"
Include dependency graph for sound.h:
```



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



Data Structures

- struct [Sound](#)
A struct used to store a sound.

Functions

- void [sound_load](#) (const char *pathToFile, [Sound](#) *sound)
Function to load a sound into a [Sound](#) struct.
- void [sound_play](#) (const [Sound](#) *music)
Function to play a sound indefinitely.
- void [sound_play_once](#) (const [Sound](#) *music)
Function to play a sound once.
- void [sound_free](#) ([Sound](#) *sound)
Function to free a [Sound](#), i.e. to unload it.
- void [sound_stop](#) (void)
Function to stop the current played [Sound](#).
- void [sound_pause](#) (void)
Function to pause the current played [Sound](#).
- void [sound_resume](#) (void)
Function to resume the current paused [Sound](#).

4.15.1 Detailed Description

Everything related to [Sound](#).

4.15.2 Function Documentation

4.15.2.1 void sound_free ([Sound](#) * sound)

Function to free a [Sound](#), i.e. to unload it.

Parameters

<i>sound</i>	A pointer to the Sound to free.
--------------	---

4.15.2.2 void sound_load (const char * *fileName*, **Sound** * *sound*)

Function to load a sound into a [Sound](#) struct.

Parameters

<i>pathToFile</i>	The path to the file to load.
<i>sound</i>	Pointer to the Sound in which the file must be stored.

4.15.2.3 void sound_pause (void)

Function to pause the current played [Sound](#).

4.15.2.4 void sound_play (const **Sound** * *music*)

Function to play a sound indefinitely.

Parameters

<i>music</i>	A pointer to the Sound to play.
--------------	---

4.15.2.5 void sound_play_once (const **Sound** * *music*)

Function to play a sound once.

Parameters

<i>music</i>	A pointer to the Sound to play.
--------------	---

4.15.2.6 void sound_resume (void)

Function to resume the current paused [Sound](#).

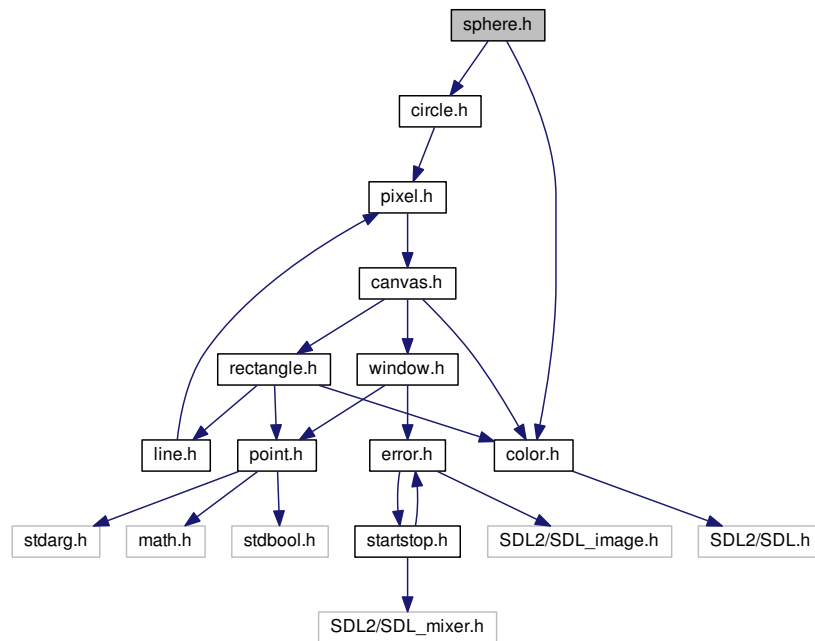
4.15.2.7 void sound_stop (void)

Function to stop the current played [Sound](#).

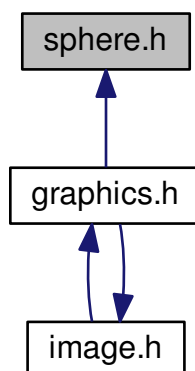
4.16 sphere.h File Reference

Everything related to [Sphere](#).

```
#include "circle.h"
#include "color.h"
Include dependency graph for sphere.h:
```



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



Data Structures

- struct [Sphere](#)

Functions

- void [sphere_draw_fill](#) (const [Sphere](#) *sphere, const [Color](#) *color)
Function to draw a filled [Sphere](#).

4.16.1 Detailed Description

Everything related to [Sphere](#).

4.16.2 Function Documentation

4.16.2.1 void sphere_draw_fill (const [Sphere](#) * *sphere*, const [Color](#) * *color*)

Function to draw a filled [Sphere](#).

Parameters

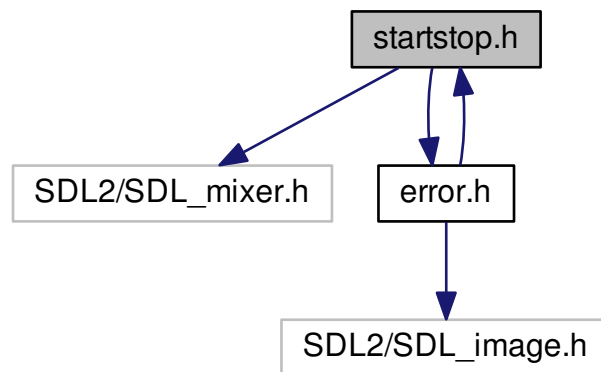
<i>sphere</i>	A pointer to the Sphere to draw.
<i>color</i>	A pointer to the Color to use to draw the Sphere .

4.17 startstop.h File Reference

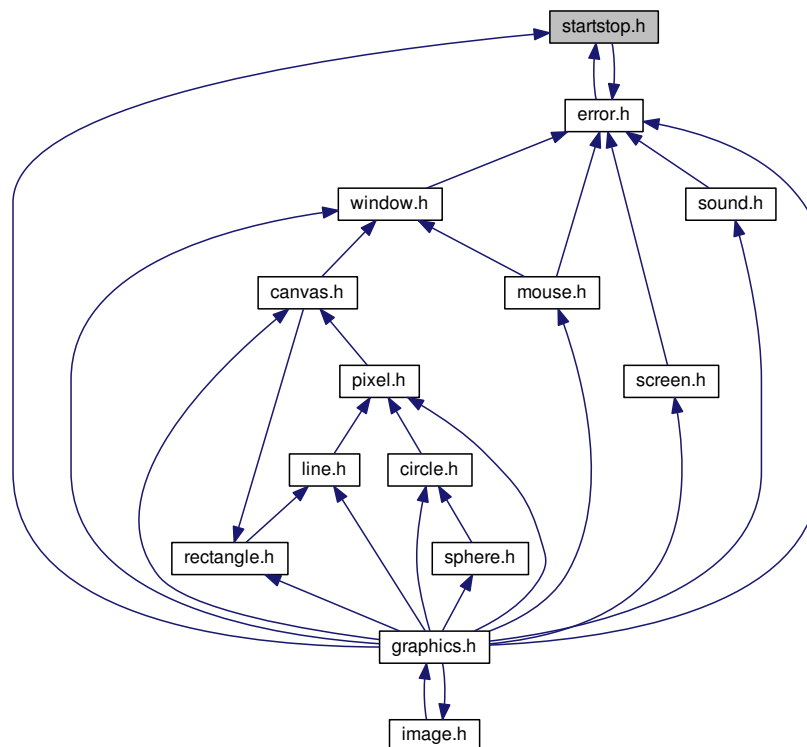
Everything related to graphics' start and stop functions.

```
#include <SDL2/SDL_mixer.h>
#include "error.h"
```

Include dependency graph for startstop.h:



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



Functions

- void [graphics_start](#) (const Uint32 flags)

Function to start graphics.

- void [graphics_stop](#) (void)

Function to stop graphics.

4.17.1 Detailed Description

Everything related to graphics' start and stop functions.

4.17.2 Function Documentation

4.17.2.1 void [graphics_start](#) (const Uint32 *flags*)

Function to start graphics.

Parameters

<i>flags</i>	A list of SDL flags, if you don't know, use <code>SDL_INIT_EVERYTHING</code> , or see <code>SDL_Init</code> doc.
--------------	--

4.17.2.2 void [graphics_stop](#) (void)

Function to stop graphics.

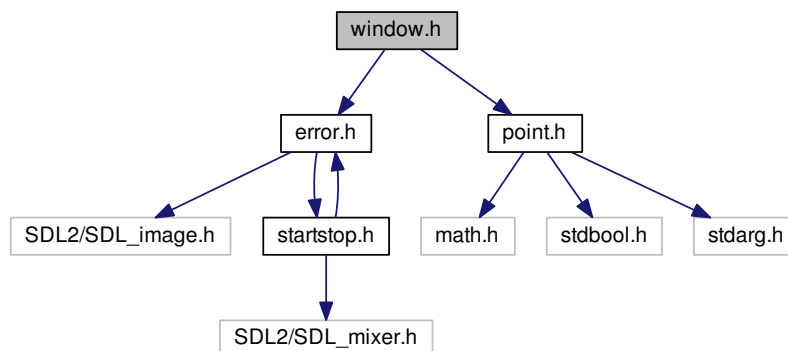
4.18 window.h File Reference

Everything related to [Window](#).

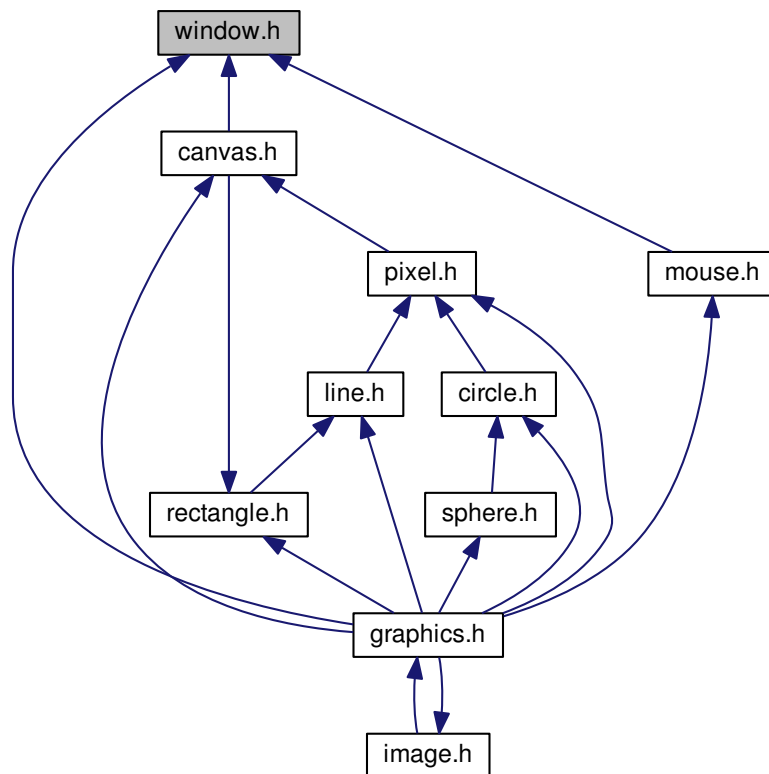
```
#include "error.h"
```

```
#include "point.h"
```

Include dependency graph for window.h:



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



Data Structures

- struct [Window](#)

Functions

- void [window_create](#) ([Window](#) *window, char *title, const [Point](#) *position, const [Point](#) *size, const Uint32 flags)
Function to create a [Window](#).
- void [window_destroy](#) ([Window](#) *window)
Function to destroy a [Window](#).
- void [window_update](#) ([Window](#) *window)
Function to update a [Window](#).

4.18.1 Detailed Description

Everything related to [Window](#).

4.18.2 Function Documentation

4.18.2.1 void window_create (Window * *window*, char * *title*, const Point * *position*, const Point * *size*, const Uint32 *flags*)

Function to create a [Window](#).

Parameters

<i>window</i>	A pointer to the Window in which the new Window will be stored.
<i>title</i>	The title wanted for the Window .
<i>position</i>	A pointer to a Point representing the position wanted for the Window .
<i>size</i>	A pointer to a Point representing the size wanted for the Window .
<i>flags</i>	A list of SDL flags, if you don't know, use SDL_WINDOW_SHOWN, or see SDL_CreateWindow doc.

4.18.2.2 void window_destroy ([Window](#) * *window*)

Function to destroy a [Window](#).

Parameters

<i>window</i>	A pointer to the Window to destroy.
---------------	---

4.18.2.3 void window_update ([Window](#) * *window*)

Function to update a [Window](#).

Parameters

<i>window</i>	A pointer to the Window to update.
---------------	--

Index

a

Line, [11](#)

alpha

Color, [8](#)

arrows

Event, [9](#)

b

Line, [11](#)

calc.h, [19](#)

calc_alea_float, [20](#)

calc_alea_int, [20](#)

calc_alea_float

calc.h, [20](#)

calc_alea_int

calc.h, [20](#)

Canvas, [5](#)

canvas.h, [23](#)

origin, [6](#)

parent, [6](#)

size, [6](#)

surface, [6](#)

canvas

Circle, [7](#)

Image, [10](#)

Line, [11](#)

Pixel, [12](#)

Rectangle, [14](#)

Sphere, [16](#)

canvas.h, [21](#)

Canvas, [23](#)

canvas_blit, [23](#)

canvas_clear, [24](#)

canvas_collision_canvas, [24](#)

canvas_create, [24](#)

canvas_create_from_window, [24](#)

canvas_draw_borders_in, [25](#)

canvas_draw_borders_out, [25](#)

canvas_fill, [25](#)

canvas_get_absolute_origin, [25](#)

canvas_is_out_of_parent_bottom, [25](#)

canvas_is_out_of_parent_left, [26](#)

canvas_is_out_of_parent_right, [26](#)

canvas_is_out_of_parent_top, [26](#)

canvas_is_out_of_parent_x, [26](#)

canvas_is_out_of_parent_y, [27](#)

canvas_will_be_out_of_parent_bottom, [27](#)

canvas_will_be_out_of_parent_left, [27](#)

canvas_will_be_out_of_parent_right, [28](#)

canvas_will_be_out_of_parent_top, [28](#)

canvas_will_be_out_of_parent_x, [28](#)

canvas_will_be_out_of_parent_y, [28](#)

canvas_blit

canvas.h, [23](#)

canvas_clear

canvas.h, [24](#)

canvas_collision_canvas

canvas.h, [24](#)

canvas_create

canvas.h, [24](#)

canvas_create_from_window

canvas.h, [24](#)

canvas_draw_borders_in

canvas.h, [25](#)

canvas_draw_borders_out

canvas.h, [25](#)

canvas_fill

canvas.h, [25](#)

canvas_get_absolute_origin

canvas.h, [25](#)

canvas_is_out_of_parent_bottom

canvas.h, [25](#)

canvas_is_out_of_parent_left

canvas.h, [26](#)

canvas_is_out_of_parent_right

canvas.h, [26](#)

canvas_is_out_of_parent_top

canvas.h, [26](#)

canvas_is_out_of_parent_x

canvas.h, [26](#)

canvas_is_out_of_parent_y

canvas.h, [27](#)

canvas_will_be_out_of_parent_bottom

canvas.h, [27](#)

canvas_will_be_out_of_parent_left

canvas.h, [27](#)

canvas_will_be_out_of_parent_right

canvas.h, [28](#)

canvas_will_be_out_of_parent_top

canvas.h, [28](#)

canvas_will_be_out_of_parent_x

canvas.h, [28](#)

canvas_will_be_out_of_parent_y

canvas.h, [28](#)

center

Circle, [7](#)

Sphere, [16](#)

Circle, [6](#)

- canvas, 7
 - center, 7
 - radius, 7
- circle.h, 29
 - circle_draw, 30
 - circle_draw_fill, 31
- circle_draw
 - circle.h, 30
- circle_draw_fill
 - circle.h, 31
- Color, 7
 - alpha, 8
 - rgb, 8
- color.h, 31
 - color_get_blue, 33
 - color_get_green, 33
 - color_get_red, 33
 - color_translate, 33
- color_get_blue
 - color.h, 33
- color_get_green
 - color.h, 33
- color_get_red
 - color.h, 33
- color_translate
 - color.h, 33
- content
 - Sound, 15
- error.h, 33
 - error_quit, 35
- error_quit
 - error.h, 35
- Event, 8
 - arrows, 9
 - quit, 9
 - space, 9
- event.h, 35
 - event_create, 36
 - event_update, 36
- event_create
 - event.h, 36
- event_update
 - event.h, 36
- graphics.h, 37
- graphics_start
 - startstop.h, 61
- graphics_stop
 - startstop.h, 61
- Image, 9
 - canvas, 10
 - surface, 10
- image.h, 38
 - image_blit_naive, 39
 - image_blit_scaled, 39
 - image_load, 40
 - image_unload, 40
- image_blit_naive
 - image.h, 39
- image_blit_scaled
 - image.h, 39
- image_load
 - image.h, 40
- image_unload
 - image.h, 40
- Line, 10
 - a, 11
 - b, 11
 - canvas, 11
- line.h, 40
 - line_draw, 43
 - line_draw_bis, 43
 - line_draw_ter, 43
- line_draw
 - line.h, 43
- line_draw_bis
 - line.h, 43
- line_draw_ter
 - line.h, 43
- mouse.h, 43
 - mouse_hide, 45
 - mouse_is_hidden, 45
 - mouse_is_shown, 45
 - mouse_show, 45
 - mouse_wait_click, 45
- mouse_hide
 - mouse.h, 45
- mouse_is_hidden
 - mouse.h, 45
- mouse_is_shown
 - mouse.h, 45
- mouse_show
 - mouse.h, 45
- mouse_wait_click
 - mouse.h, 45
- origin
 - Canvas, 6
 - Rectangle, 14
- parent
 - Canvas, 6
- Pixel, 11
 - canvas, 12
 - position, 12
- pixel.h, 46
 - pixel_draw, 47
- pixel_draw
 - pixel.h, 47
- Point, 13
 - x, 13
 - y, 13
- point.h, 48
 - point_are_equals, 49

- point_distance, [49](#)
- point_max_x, [50](#)
- point_max_y, [50](#)
- point_min_x, [51](#)
- point_min_y, [51](#)
- point_are_equals
 - point.h, [49](#)
- point_distance
 - point.h, [49](#)
- point_max_x
 - point.h, [50](#)
- point_max_y
 - point.h, [50](#)
- point_min_x
 - point.h, [51](#)
- point_min_y
 - point.h, [51](#)
- position
 - Pixel, [12](#)
 - Window, [17](#)
- quit
 - Event, [9](#)
- radius
 - Circle, [7](#)
 - Sphere, [16](#)
- Rectangle, [13](#)
 - canvas, [14](#)
 - origin, [14](#)
 - size, [14](#)
- rectangle.h, [51](#)
 - rectangle_contains_absolute_point, [53](#)
 - rectangle_contains_point, [53](#)
 - rectangle_draw, [53](#)
 - rectangle_draw_fill, [53](#)
- rectangle_contains_absolute_point
 - rectangle.h, [53](#)
- rectangle_contains_point
 - rectangle.h, [53](#)
- rectangle_draw
 - rectangle.h, [53](#)
- rectangle_draw_fill
 - rectangle.h, [53](#)
- rgb
 - Color, [8](#)
- screen.h, [54](#)
 - screen_get_size, [55](#)
- screen_get_size
 - screen.h, [55](#)
- size
 - Canvas, [6](#)
 - Rectangle, [14](#)
 - Window, [17](#)
- Sound, [15](#)
 - content, [15](#)
- sound.h, [55](#)
 - sound_free, [56](#)
 - sound_load, [57](#)
 - sound_pause, [57](#)
 - sound_play, [57](#)
 - sound_play_once, [57](#)
 - sound_resume, [57](#)
 - sound_stop, [57](#)
- sound_free
 - sound.h, [56](#)
- sound_load
 - sound.h, [57](#)
- sound_pause
 - sound.h, [57](#)
- sound_play
 - sound.h, [57](#)
- sound_play_once
 - sound.h, [57](#)
- sound_resume
 - sound.h, [57](#)
- sound_stop
 - sound.h, [57](#)
- space
 - Event, [9](#)
- Sphere, [15](#)
 - canvas, [16](#)
 - center, [16](#)
 - radius, [16](#)
- sphere.h, [58](#)
 - sphere_draw_fill, [59](#)
- sphere_draw_fill
 - sphere.h, [59](#)
- startstop.h, [59](#)
 - graphics_start, [61](#)
 - graphics_stop, [61](#)
- surface
 - Canvas, [6](#)
 - Image, [10](#)
- title
 - Window, [17](#)
- Window, [16](#)
 - position, [17](#)
 - size, [17](#)
 - title, [17](#)
 - window, [17](#)
- window
 - Window, [17](#)
- window.h, [61](#)
 - window_create, [63](#)
 - window_destroy, [64](#)
 - window_update, [64](#)
- window_create
 - window.h, [63](#)
- window_destroy
 - window.h, [64](#)
- window_update
 - window.h, [64](#)
- x

Point, [13](#)

y

Point, [13](#)