

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	Field Documentation	8
3.4.1.1	arrows	8
3.4.1.2	quit	8
3.4.1.3	space	8
3.5	Image Struct Reference	9
3.5.1	Field Documentation	9
3.5.1.1	canvas	9
3.5.1.2	surface	9
3.6	Line Struct Reference	10
3.6.1	Field Documentation	10
3.6.1.1	a	10
3.6.1.2	b	10
3.6.1.3	canvas	10
3.7	Pixel Struct Reference	11
3.7.1	Field Documentation	11
3.7.1.1	canvas	11
3.7.1.2	position	11
3.8	Point Struct Reference	11
3.8.1	Field Documentation	12
3.8.1.1	x	12
3.8.1.2	y	12
3.9	Rectangle Struct Reference	12
3.9.1	Field Documentation	13
3.9.1.1	canvas	13
3.9.1.2	origin	13
3.9.1.3	size	13
3.10	Sound Struct Reference	13
3.10.1	Field Documentation	13
3.10.1.1	content	13
3.11	Sphere Struct Reference	13
3.11.1	Field Documentation	14
3.11.1.1	canvas	14
3.11.1.2	center	14
3.11.1.3	radius	14
3.12	Window Struct Reference	14
3.12.1	Field Documentation	15
3.12.1.1	position	15
3.12.1.2	size	15
3.12.1.3	title	15
3.12.1.4	window	15

4 File Documentation	17
4.1 calc.h File Reference	17
4.1.1 Detailed Description	18
4.1.2 Function Documentation	18
4.1.2.1 calc_alea_float(void)	18
4.1.2.2 calc_alea_int(const int min, const int max)	18
4.2 canvas.h File Reference	19
4.2.1 Detailed Description	21
4.2.2 Typedef Documentation	21
4.2.2.1 Canvas	21
4.2.3 Function Documentation	21
4.2.3.1 canvas_blit(Canvas *canvas)	21
4.2.3.2 canvas_clear(Canvas *canvas)	22
4.2.3.3 canvas_collision_canvas(const Canvas *canvas1, const Canvas *canvas2) __attribute__((pure))	22
4.2.3.4 canvas_create(Canvas *canvas, const Point *size, const Point *origin, Canvas *parent)	22
4.2.3.5 canvas_create_from_window(Canvas *canvas, const Window *window)	22
4.2.3.6 canvas_draw_borders_in(Canvas *canvas, const Color *color)	23
4.2.3.7 canvas_draw_borders_out(Canvas *canvas, const Color *color)	23
4.2.3.8 canvas_fill(Canvas *canvas, const Color *color)	23
4.2.3.9 canvas_get_absolute_origin(const Canvas *canvas, Point *absoluteOrigin)	23
4.2.3.10 canvas_is_out_of_parent_bottom(const Canvas *canvas) __attribute__((pure))	23
4.2.3.11 canvas_is_out_of_parent_left(const Canvas *canvas) __attribute__((pure))	24
4.2.3.12 canvas_is_out_of_parent_right(const Canvas *canvas) __attribute__((pure))	24
4.2.3.13 canvas_is_out_of_parent_top(const Canvas *canvas) __attribute__((pure))	24
4.2.3.14 canvas_is_out_of_parent_x(const Canvas *canvas) __attribute__((pure))	24
4.2.3.15 canvas_is_out_of_parent_y(const Canvas *canvas) __attribute__((pure))	25
4.2.3.16 canvas_will_be_out_of_parent_bottom(const Canvas *canvas, const Point *move) __attribute__((pure))	25
4.2.3.17 canvas_will_be_out_of_parent_left(const Canvas *canvas, const Point *move) __attribute__((pure))	25

4.2.3.18	<code>canvas_will_be_out_of_parent_right(const Canvas *canvas, const Point *move)</code> <code>__attribute__((pure))</code>	26
4.2.3.19	<code>canvas_will_be_out_of_parent_top(const Canvas *canvas, const Point *move)</code> \leftrightarrow <code>__attribute__((pure))</code>	26
4.2.3.20	<code>canvas_will_be_out_of_parent_x(const Canvas *canvas, const Point *move)</code> \leftrightarrow <code>__attribute__((pure))</code>	26
4.2.3.21	<code>canvas_will_be_out_of_parent_y(const Canvas *canvas, const Point *move)</code> \leftrightarrow <code>__attribute__((pure))</code>	26
4.3	<code>circle.h</code> File Reference	27
4.3.1	Detailed Description	28
4.3.2	Function Documentation	28
4.3.2.1	<code>circle_draw(const Circle *circle, const Color *color)</code>	28
4.3.2.2	<code>circle_draw_fill(const Circle *circle, const Color *color)</code>	29
4.4	<code>color.h</code> File Reference	29
4.4.1	Detailed Description	30
4.4.2	Function Documentation	31
4.4.2.1	<code>color_get_blue(const Color *color)</code> <code>__attribute__((pure))</code>	31
4.4.2.2	<code>color_get_green(const Color *color)</code> <code>__attribute__((const))</code>	31
4.4.2.3	<code>color_get_red(const Color *color)</code> <code>__attribute__((const))</code>	31
4.4.2.4	<code>color_translate(const Color *color, SDL_Color *sdlColor)</code>	31
4.5	<code>error.h</code> File Reference	31
4.5.1	Function Documentation	33
4.5.1.1	<code>error_quit(void)</code> <code>__attribute__((noreturn))</code>	33
4.6	<code>event.h</code> File Reference	33
4.6.1	Function Documentation	34
4.6.1.1	<code>event_create(Event *newEvent)</code>	34
4.6.1.2	<code>event_update(Event *event)</code>	34
4.7	<code>graphics.h</code> File Reference	34
4.8	<code>image.h</code> File Reference	35
4.8.1	Function Documentation	36
4.8.1.1	<code>image_blit_naive(const Image *image)</code>	36
4.8.1.2	<code>image_blit_scaled(const Image *image)</code>	36

4.8.1.3	<code>image_load(Image *image, const char *pathToImg)</code>	36
4.8.1.4	<code>image_unload(Image *image)</code>	36
4.9	<code>line.h</code> File Reference	36
4.9.1	Function Documentation	38
4.9.1.1	<code>line_draw(const Line *line, const Color *color)</code>	38
4.9.1.2	<code>line_draw_bis(const Line *line, const Color *color)</code>	38
4.9.1.3	<code>line_draw_ter(const Line *line, const Color *color)</code>	38
4.10	<code>mouse.h</code> File Reference	38
4.10.1	Function Documentation	39
4.10.1.1	<code>mouse_hide(void)</code>	39
4.10.1.2	<code>mouse_is_hidden(void)</code>	39
4.10.1.3	<code>mouse_is_shown(void)</code>	39
4.10.1.4	<code>mouse_show(void)</code>	39
4.10.1.5	<code>mouse_wait_click(const Window *window, Point *click)</code>	39
4.11	<code>pixel.h</code> File Reference	39
4.11.1	Function Documentation	40
4.11.1.1	<code>pixel_draw(const Pixel *pixel, const Color *color)</code>	40
4.12	<code>point.h</code> File Reference	41
4.12.1	Function Documentation	42
4.12.1.1	<code>point_are_equals(const Point p1, const Point p2) __attribute__((const))</code>	42
4.12.1.2	<code>point_distance(const Point a, const Point b)</code>	42
4.12.1.3	<code>point_max_x(const Point a, const Point b)</code>	42
4.12.1.4	<code>point_max_y(const Point a, const Point b)</code>	42
4.12.1.5	<code>point_min_x(const Point a, const Point b)</code>	42
4.12.1.6	<code>point_min_y(const Point a, const Point b)</code>	42
4.13	<code>rectangle.h</code> File Reference	42
4.13.1	Function Documentation	45
4.13.1.1	<code>rectangle_contains_absolute_point(const Rectangle *rect, const Point *p)</code>	45
4.13.1.2	<code>rectangle_contains_point(const Rectangle *rect, const Point *p) __attribute__((pure))</code>	45
4.13.1.3	<code>rectangle_draw(const Rectangle *rectangle, const Color *color)</code>	45

4.13.1.4	<code>rectangle_draw_fill(const Rectangle *rectangle, const Color *color)</code>	45
4.14	screen.h File Reference	45
4.14.1	Function Documentation	46
4.14.1.1	<code>screen_get_size(Point *screenSize)</code>	46
4.15	sound.h File Reference	46
4.15.1	Function Documentation	47
4.15.1.1	<code>sound_free(Sound *sound)</code>	47
4.15.1.2	<code>sound_load(const char *fileName, Sound *sound)</code>	47
4.15.1.3	<code>sound_pause(void)</code>	47
4.15.1.4	<code>sound_play(const Sound *music)</code>	47
4.15.1.5	<code>sound_play_once(const Sound *music)</code>	47
4.15.1.6	<code>sound_resume(void)</code>	47
4.15.1.7	<code>sound_stop(void)</code>	48
4.16	sphere.h File Reference	48
4.16.1	Function Documentation	49
4.16.1.1	<code>sphere_draw_fill(const Sphere *sphere, const Color *color)</code>	49
4.17	startstop.h File Reference	49
4.17.1	Function Documentation	50
4.17.1.1	<code>graphics_start(const Uint32 flags)</code>	50
4.17.1.2	<code>graphics_stop(void)</code>	50
4.18	window.h File Reference	50
4.18.1	Function Documentation	52
4.18.1.1	<code>window_create(Window *window, char *title, const Point *position, const Point *size, const Uint32 flags)</code>	52
4.18.1.2	<code>window_destroy(Window *window)</code>	52
4.18.1.3	<code>window_update(Window *window)</code>	52
Index		53

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	8
Image	9
Line	10
Pixel	11
Point	11
Rectangle	12
Sound	13
Sphere	13
Window	14

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

calc.h	Some maths functions	17
canvas.h	Everything related to Canvas	19
circle.h	Everything related to Circle	27
color.h	Everything related to Color	29
error.h	31
event.h	33
graphics.h	34
image.h	35
line.h	36
mouse.h	38
pixel.h	39
point.h	41
rectangle.h	42
screen.h	45
sound.h	46
sphere.h	48
startstop.h	49
window.h	50

Chapter 3

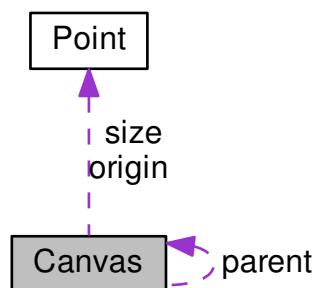
Data Structure Documentation

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:

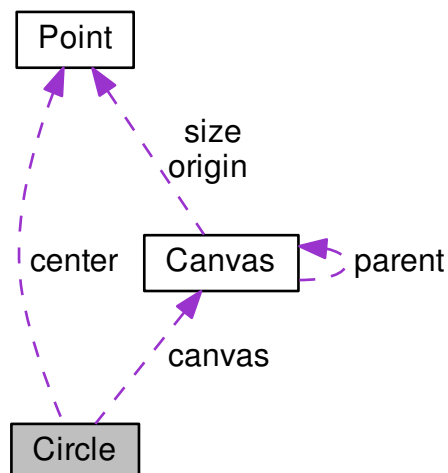
- [canvas.h](#)

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.

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.

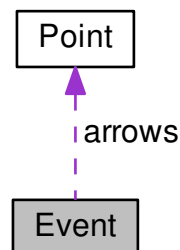
The documentation for this struct was generated from the following file:

- [color.h](#)

3.4 Event Struct Reference

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

Collaboration diagram for Event:



Data Fields

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

3.4.1 Field Documentation

3.4.1.1 Point Event::arrows

3.4.1.2 bool Event::quit

3.4.1.3 bool Event::space

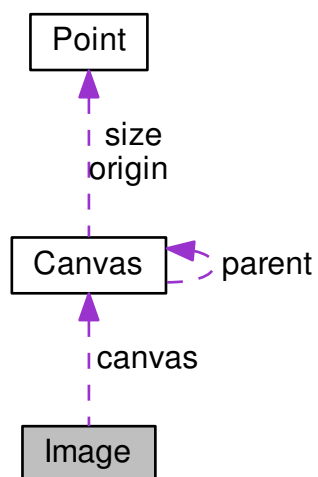
The documentation for this struct was generated from the following file:

- [event.h](#)

3.5 Image Struct Reference

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

Collaboration diagram for Image:



Data Fields

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

3.5.1 Field Documentation

3.5.1.1 `Canvas*` `Image::canvas`

3.5.1.2 `SDL_Surface*` `Image::surface`

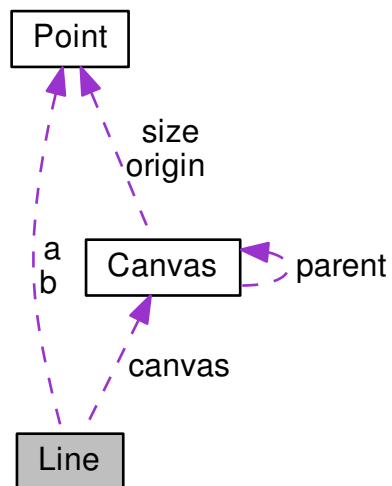
The documentation for this struct was generated from the following file:

- [image.h](#)

3.6 Line Struct Reference

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

Collaboration diagram for Line:



Data Fields

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

3.6.1 Field Documentation

3.6.1.1 Point Line::a

3.6.1.2 Point Line::b

3.6.1.3 Canvas* Line::canvas

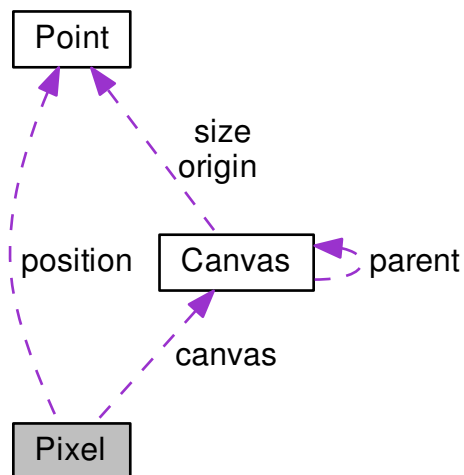
The documentation for this struct was generated from the following file:

- [line.h](#)

3.7 Pixel Struct Reference

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

Collaboration diagram for Pixel:



Data Fields

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

3.7.1 Field Documentation

3.7.1.1 Canvas* Pixel::canvas

3.7.1.2 Point Pixel::position

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

- [pixel.h](#)

3.8 Point Struct Reference

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

Data Fields

- [int x](#)
- [int y](#)

3.8.1 Field Documentation

3.8.1.1 `int Point::x`

3.8.1.2 `int Point::y`

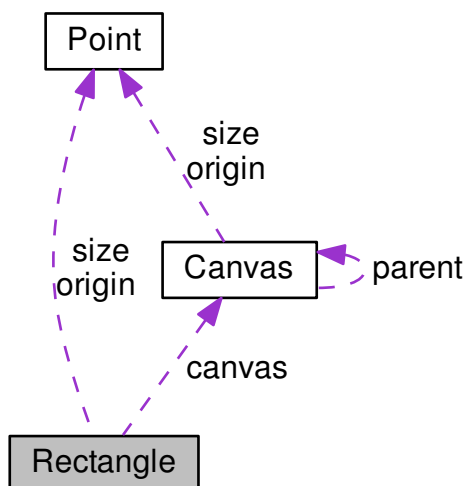
The documentation for this struct was generated from the following file:

- [point.h](#)

3.9 Rectangle Struct Reference

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

Collaboration diagram for Rectangle:



Data Fields

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

3.9.1 Field Documentation

3.9.1.1 Canvas* Rectangle::canvas

3.9.1.2 Point Rectangle::origin

3.9.1.3 Point Rectangle::size

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

- [rectangle.h](#)

3.10 Sound Struct Reference

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

Data Fields

- Mix_Music * [content](#)

3.10.1 Field Documentation

3.10.1.1 Mix_Music* Sound::content

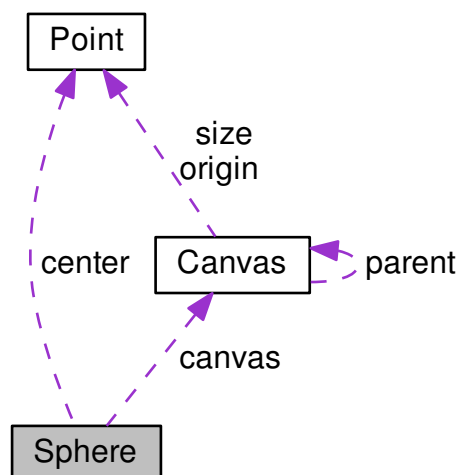
The documentation for this struct was generated from the following file:

- [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

3.11.1.2 Point Sphere::center

3.11.1.3 int Sphere::radius

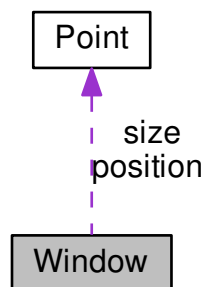
The documentation for this struct was generated from the following file:

- [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`

3.12.1.2 `Point Window::size`

3.12.1.3 `char* Window::title`

3.12.1.4 `SDL_Window* Window::window`

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

- [window.h](#)

Chapter 4

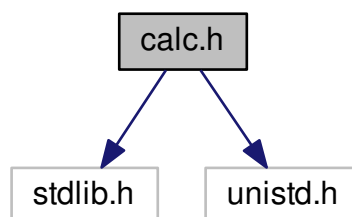
File Documentation

4.1 calc.h File Reference

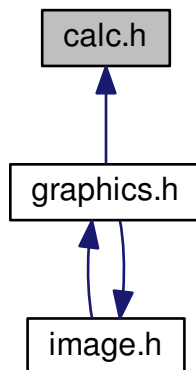
Some maths functions.

```
#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.

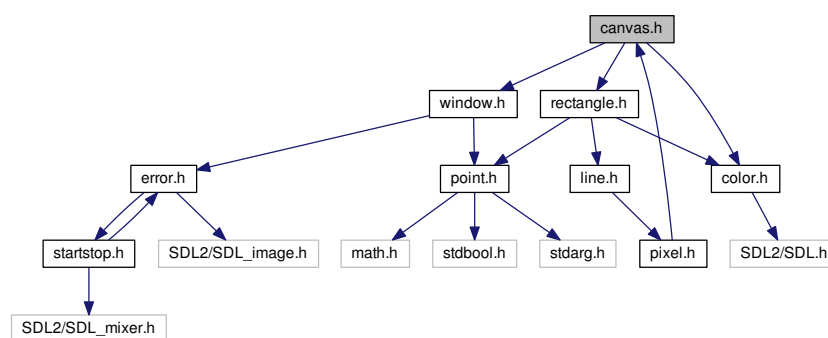
Returns

The random int.

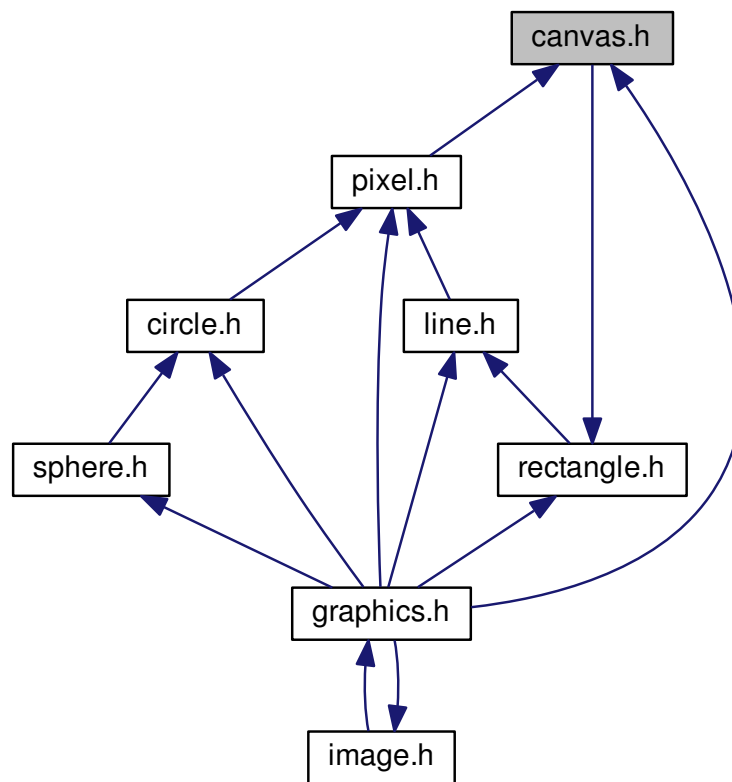
4.2 canvas.h File Reference

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.

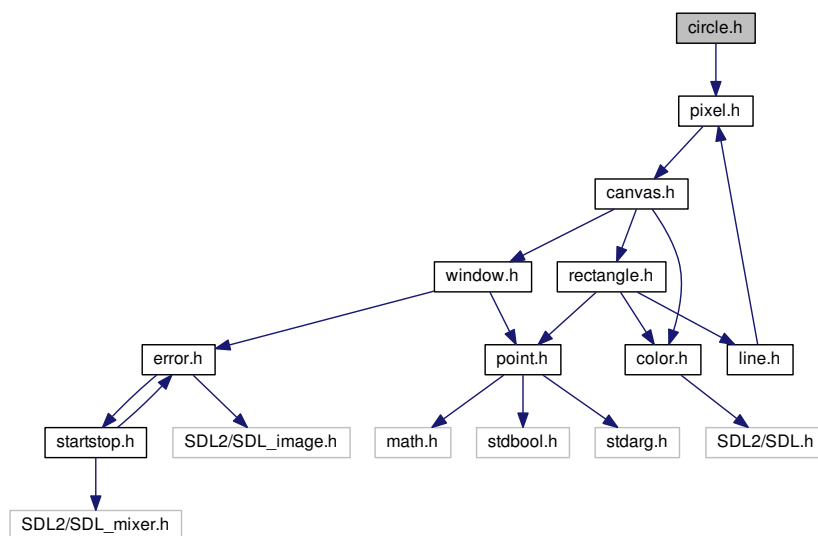
Returns

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

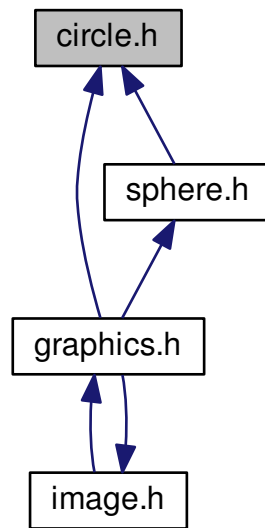
4.3 circle.h File Reference

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

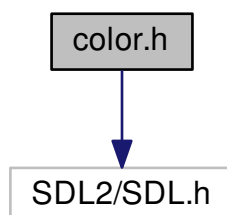
<i>circle</i>	A pointer to the Circle to draw.
<i>color</i>	A pointer to the Color to use to draw the Circle .

4.4 color.h File Reference

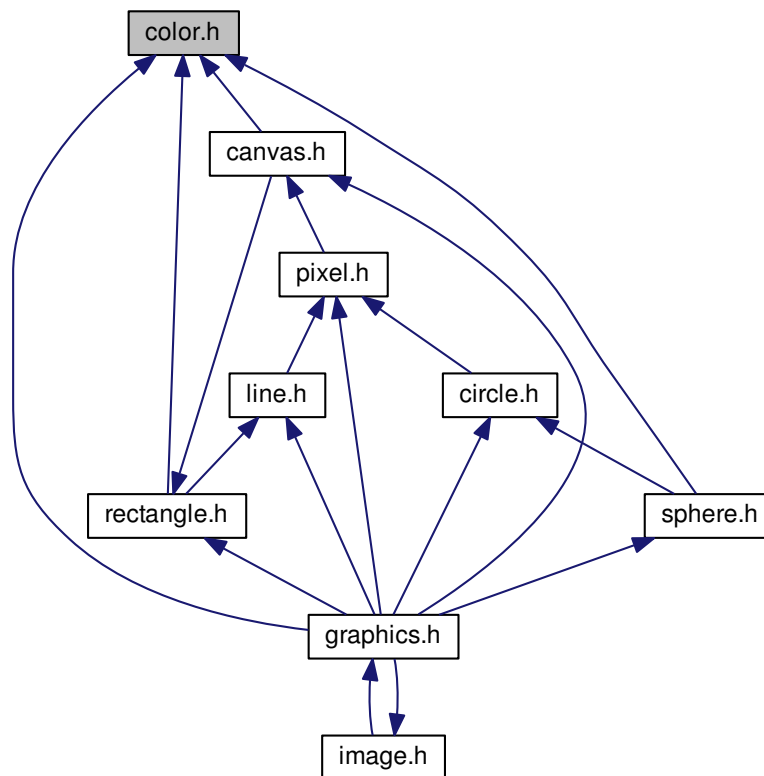
Everything related to [Color](#).

```
#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.

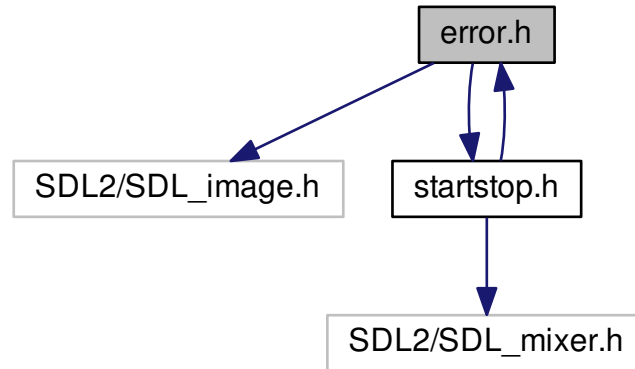
4.4.2.4 void color_translate (const Color * color, SDL_Color * sdlColor)

4.5 error.h File Reference

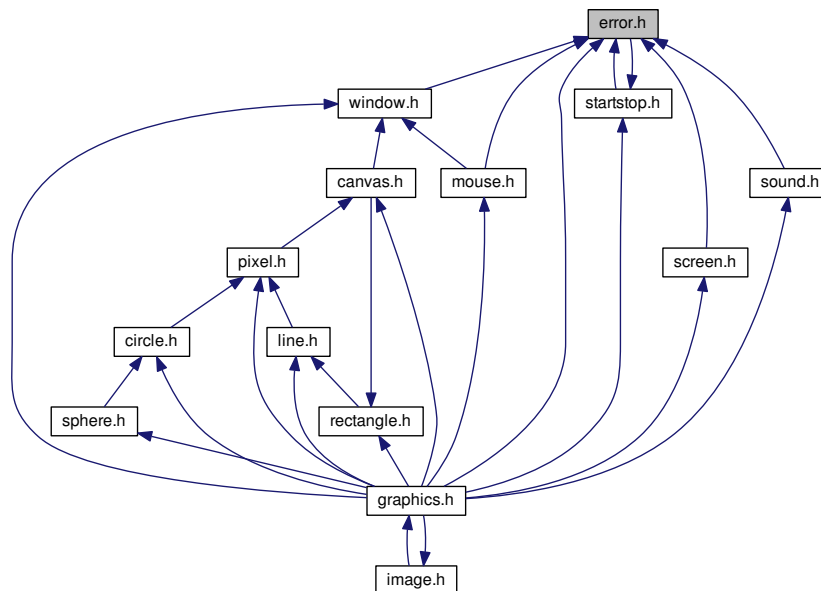
```
#include <SDL2/SDL_image.h>
```

```
#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))`

4.5.1 Function Documentation

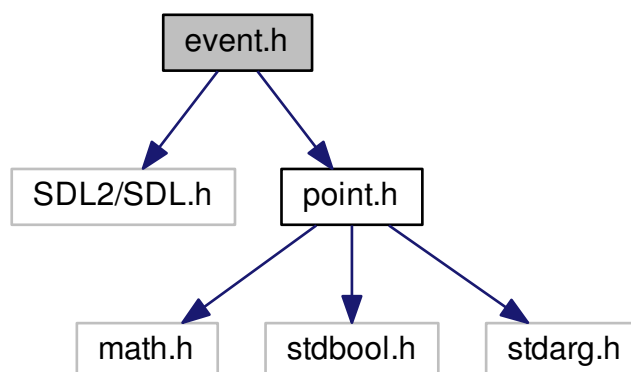
4.5.1.1 void error_quit (void)

4.6 event.h File Reference

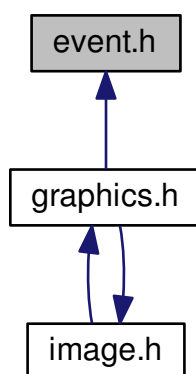
```
#include <SDL2/SDL.h>
```

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

Include dependency graph for event.h:



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



Data Structures

- struct [Event](#)

Functions

- void [event_create](#) ([Event](#) *newEvent)
- void [event_update](#) ([Event](#) *event)

4.6.1 Function Documentation

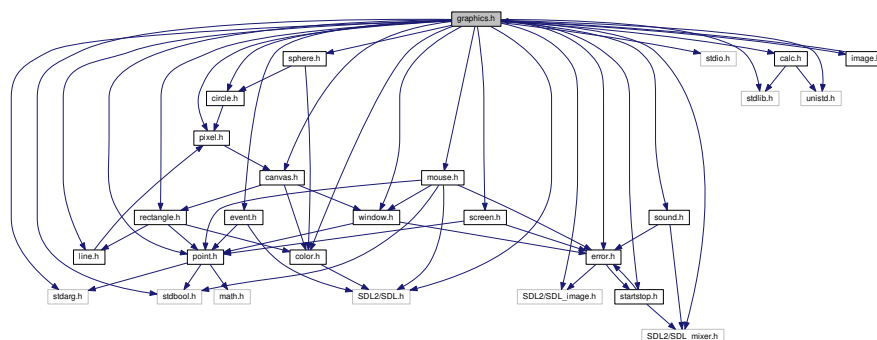
4.6.1.1 void [event_create](#) ([Event](#) * *newEvent*)

4.6.1.2 void [event_update](#) ([Event](#) * *event*)

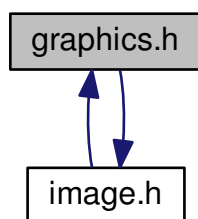
4.7 [graphics.h](#) File Reference

```
#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](#):



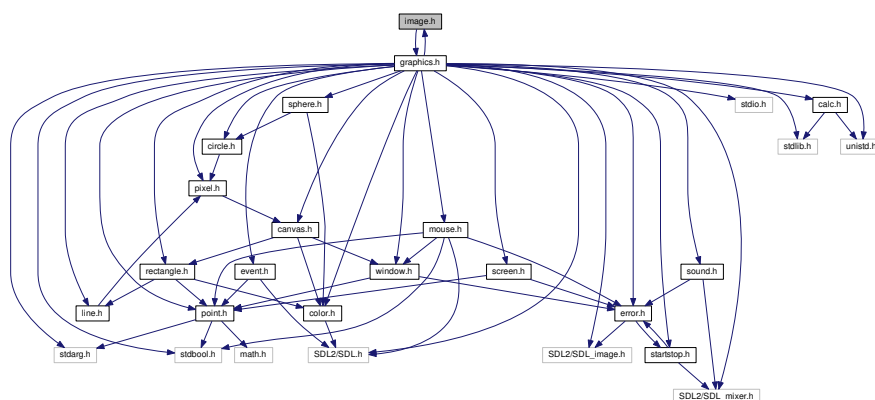
This graph shows which files directly or indirectly include this file:



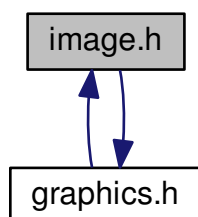
4.8 image.h File Reference

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

Include dependency graph for image.h:



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



Data Structures

- struct [Image](#)

Functions

- void [image_blit_naive](#) (const [Image](#) *image)
- void [image_blit_scaled](#) (const [Image](#) *image)
- void [image_load](#) ([Image](#) *image, const char *pathToImg)
- void [image_unload](#) ([Image](#) *image)

4.8.1 Function Documentation

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

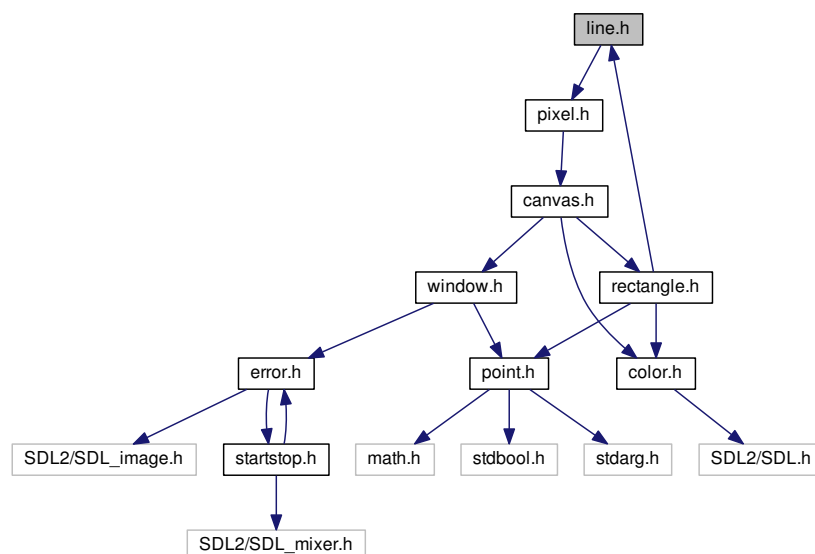
4.8.1.2 void [image_blit_scaled](#) (const [Image](#) * *image*)

4.8.1.3 void [image_load](#) ([Image](#) * *image*, const char * *pathToImg*)

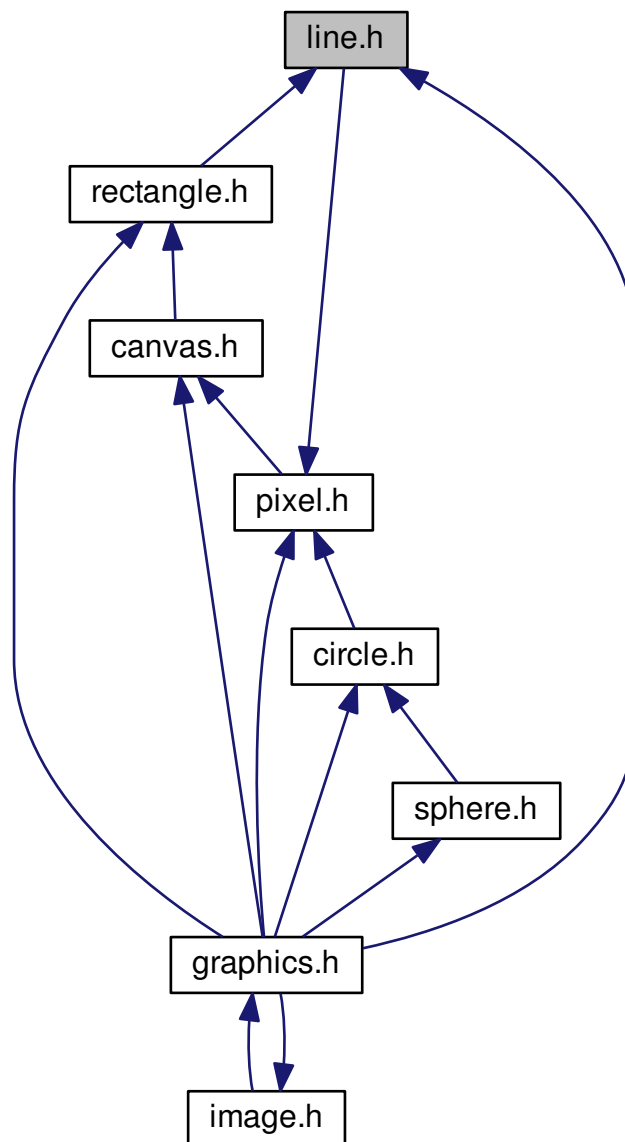
4.8.1.4 void [image_unload](#) ([Image](#) * *image*)

4.9 line.h File Reference

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



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



Data Structures

- struct [Line](#)

Functions

- void [line_draw](#) (const [Line](#) *line, const [Color](#) *color)
- void [line_draw_bis](#) (const [Line](#) *line, const [Color](#) *color)
- void [line_draw_ter](#) (const [Line](#) *line, const [Color](#) *color)

4.9.1 Function Documentation

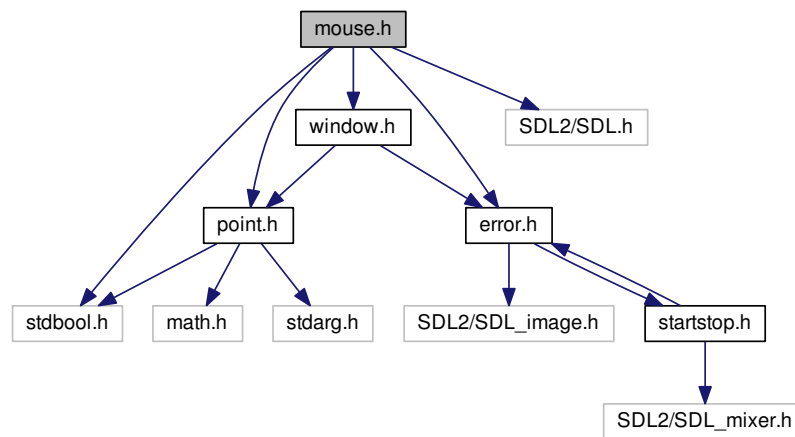
4.9.1.1 void line_draw (const Line * line, const Color * color)

4.9.1.2 void line_draw_bis (const Line * line, const Color * color)

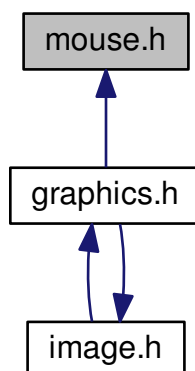
4.9.1.3 void line_draw_ter (const Line * line, const Color * color)

4.10 mouse.h File Reference

```
#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)
- void [mouse_show](#) (void)
- void [mouse_wait_click](#) (const [Window](#) *window, [Point](#) *click)
- bool [mouse_is_hidden](#) (void)
- bool [mouse_is_shown](#) (void)

4.10.1 Function Documentation

4.10.1.1 void [mouse_hide](#) (void)

4.10.1.2 bool [mouse_is_hidden](#) (void)

4.10.1.3 bool [mouse_is_shown](#) (void)

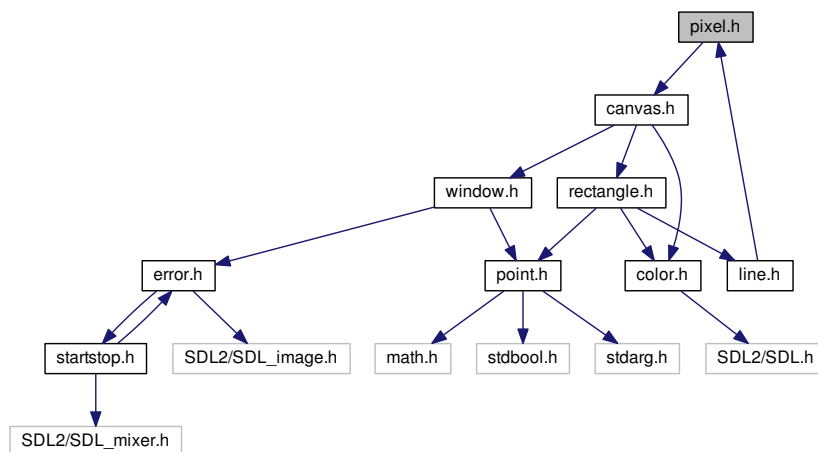
4.10.1.4 void [mouse_show](#) (void)

4.10.1.5 void [mouse_wait_click](#) (const [Window](#) * *window*, [Point](#) * *click*)

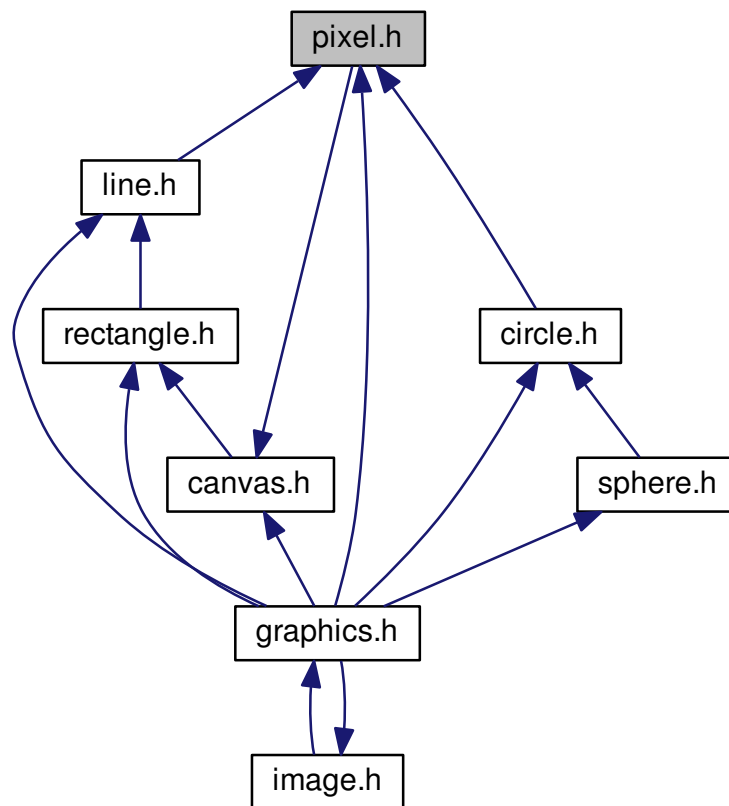
4.11 pixel.h File Reference

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

Include dependency graph for pixel.h:



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



Data Structures

- struct [Pixel](#)

Functions

- void [pixel_draw](#) (const [Pixel](#) *pixel, const [Color](#) *color)

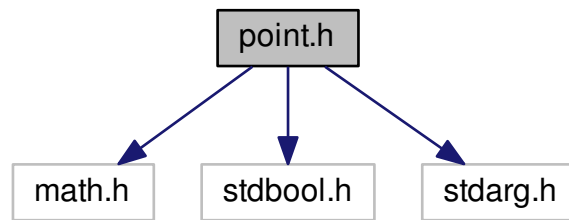
4.11.1 Function Documentation

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

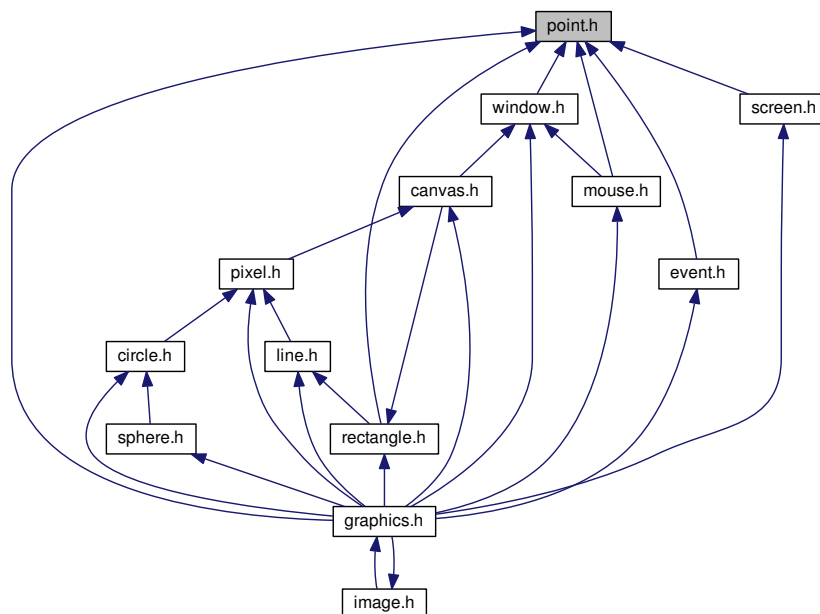
4.12 point.h File Reference

```
#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](#)

Functions

- `bool point_are_equals` (const `Point` `p1`, const `Point` `p2`) `__attribute__((const))`
- `int point_distance` (const `Point` `a`, const `Point` `b`)
- `Point point_max_x` (const `Point` `a`, const `Point` `b`)
- `Point point_max_y` (const `Point` `a`, const `Point` `b`)
- `Point point_min_x` (const `Point` `a`, const `Point` `b`)
- `Point point_min_y` (const `Point` `a`, const `Point` `b`)

4.12.1 Function Documentation

4.12.1.1 `bool point_are_equals (const Point p1, const Point p2) const`

4.12.1.2 `int point_distance (const Point a, const Point b)`

4.12.1.3 `Point point_max_x (const Point a, const Point b)`

4.12.1.4 `Point point_max_y (const Point a, const Point b)`

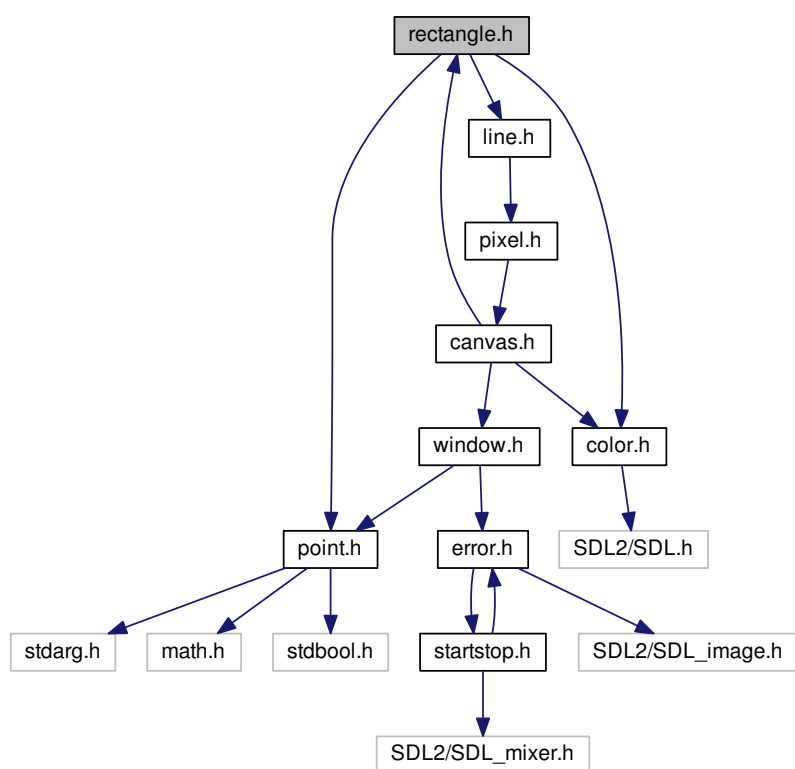
4.12.1.5 `Point point_min_x (const Point a, const Point b)`

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

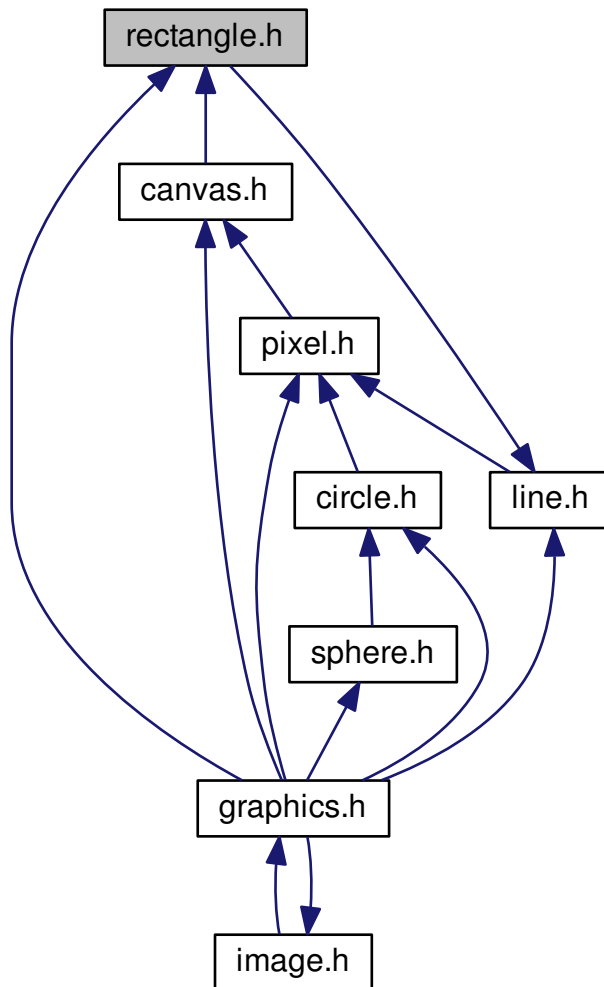
4.13 rectangle.h File Reference

```
#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](#)

Functions

- void [rectangle_draw](#) (const [Rectangle](#) *rectangle, const [Color](#) *color)
- void [rectangle_draw_fill](#) (const [Rectangle](#) *rectangle, const [Color](#) *color)
- bool [rectangle_contains_point](#) (const [Rectangle](#) *rect, const [Point](#) *p) `__attribute__((pure))`
- bool [rectangle_contains_absolute_point](#) (const [Rectangle](#) *rect, const [Point](#) *p)

4.13.1 Function Documentation

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

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

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

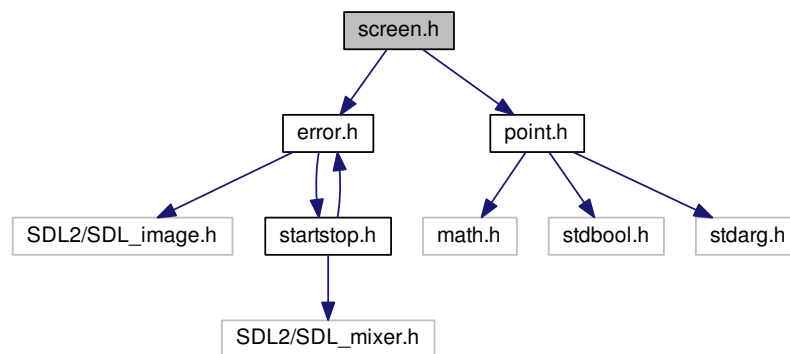
4.13.1.4 `void rectangle_draw_fill (const Rectangle * rectangle, const Color * color)`

4.14 screen.h File Reference

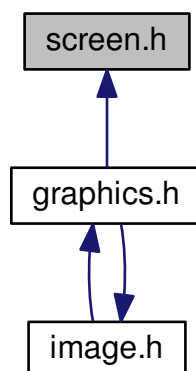
```
#include "error.h"
```

```
#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)

4.14.1 Function Documentation

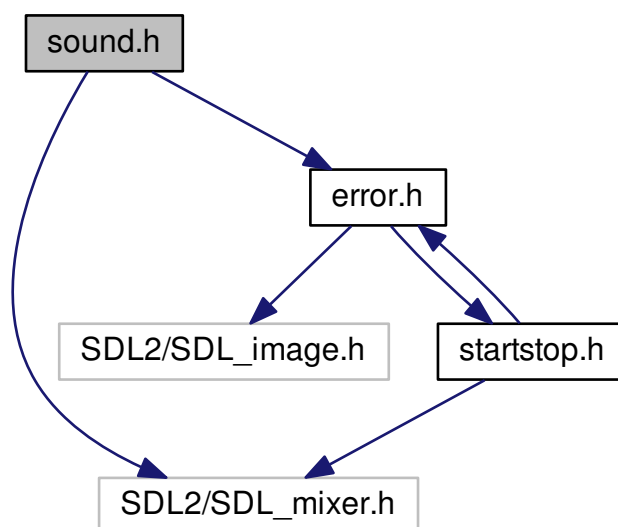
4.14.1.1 void [screen_get_size](#) ([Point](#) * *screenSize*)

4.15 sound.h File Reference

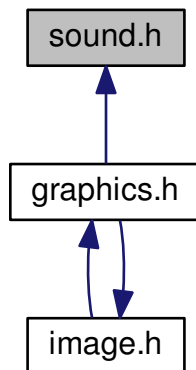
```
#include <SDL2/SDL_mixer.h>
```

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

Include dependency graph for sound.h:



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



Data Structures

- struct [Sound](#)

Functions

- void [sound_load](#) (const char *fileName, [Sound](#) *sound)
- void [sound_play](#) (const [Sound](#) *music)
- void [sound_play_once](#) (const [Sound](#) *music)
- void [sound_free](#) ([Sound](#) *sound)
- void [sound_stop](#) (void)
- void [sound_pause](#) (void)
- void [sound_resume](#) (void)

4.15.1 Function Documentation

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

4.15.1.2 void [sound_load](#) (const char * *fileName*, [Sound](#) * *sound*)

4.15.1.3 void [sound_pause](#) (void)

4.15.1.4 void [sound_play](#) (const [Sound](#) * *music*)

4.15.1.5 void [sound_play_once](#) (const [Sound](#) * *music*)

4.15.1.6 void [sound_resume](#) (void)

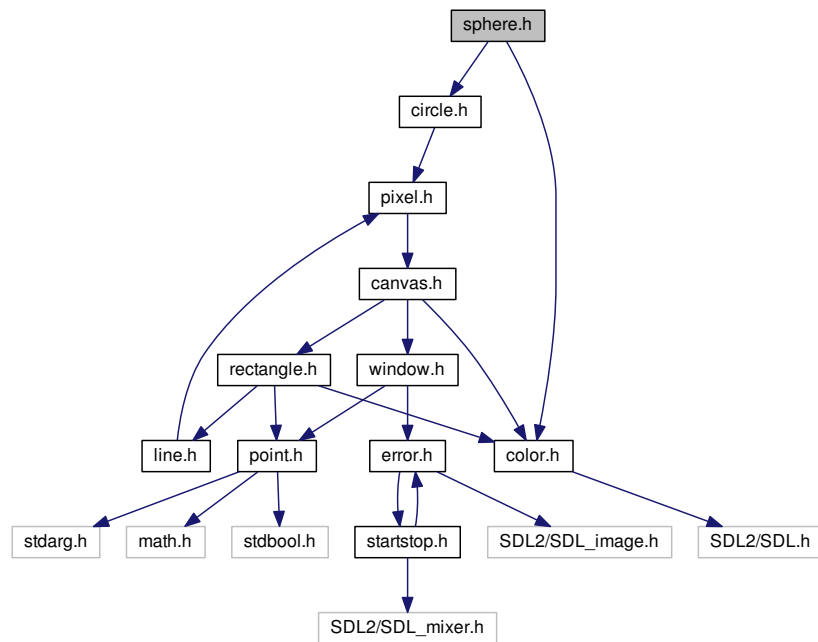
4.15.1.7 void sound_stop (void)

4.16 sphere.h File Reference

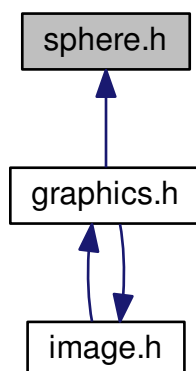
```
#include "circle.h"
```

```
#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)

4.16.1 Function Documentation

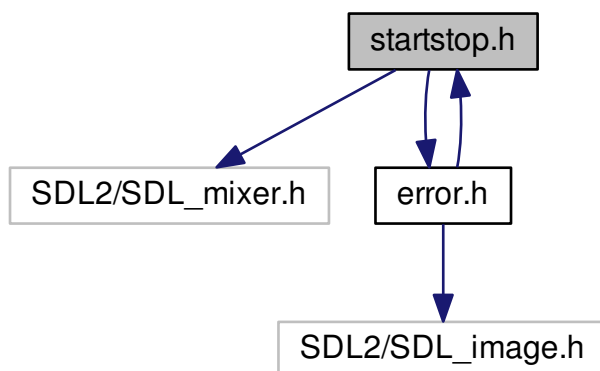
4.16.1.1 void [sphere_draw_fill](#) (const [Sphere](#) * *sphere*, const [Color](#) * *color*)

4.17 startstop.h File Reference

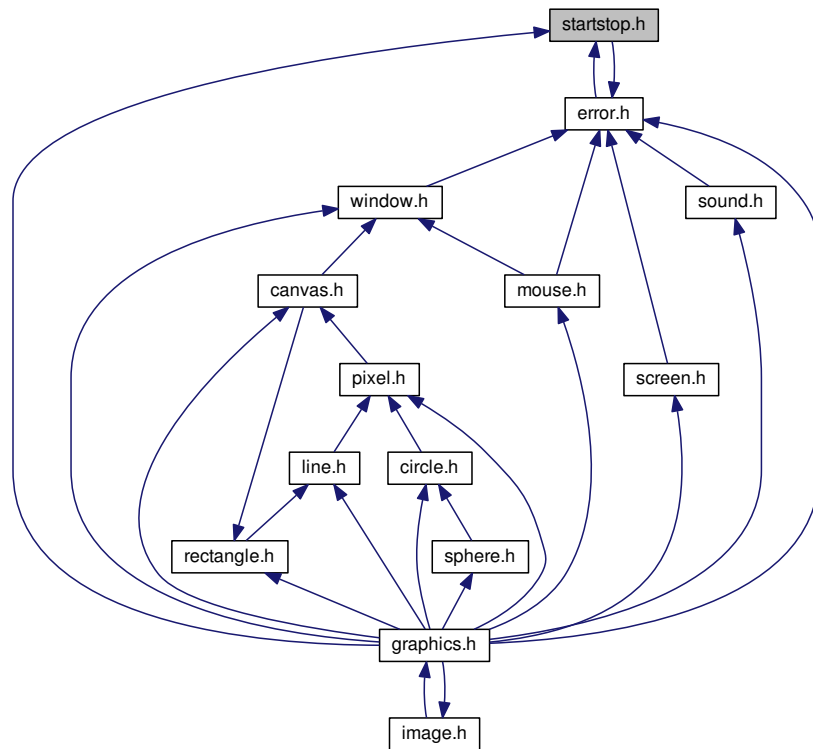
```
#include <SDL2/SDL_mixer.h>
```

```
#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)
- void [graphics_stop](#) (void)

4.17.1 Function Documentation

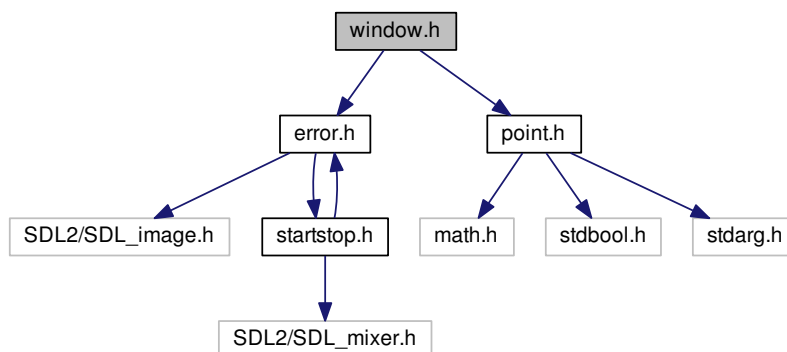
4.17.1.1 void [graphics_start](#) (const Uint32 flags)

4.17.1.2 void [graphics_stop](#) (void)

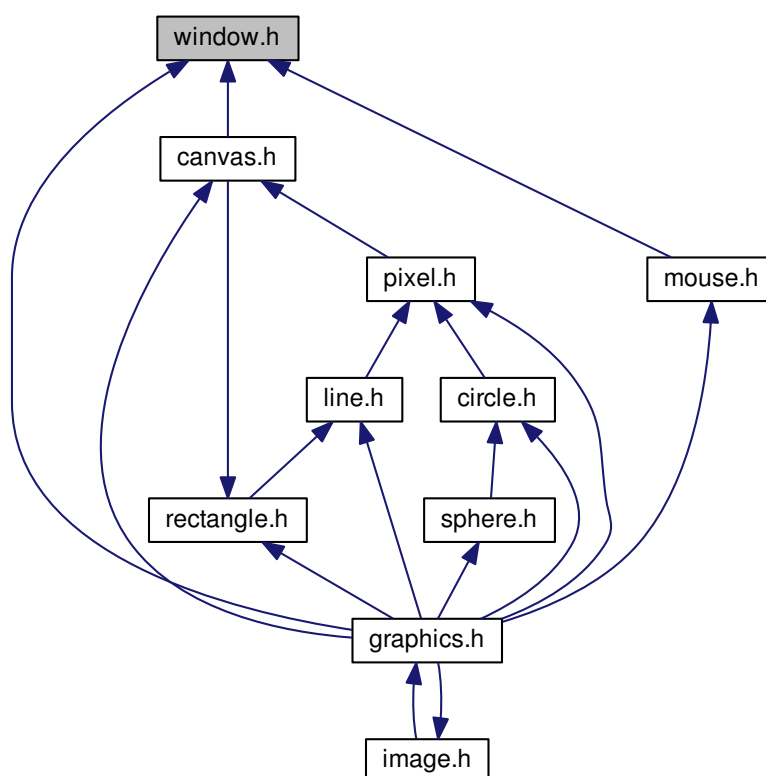
4.18 window.h File Reference

```
#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)
- void `window_destroy` (`Window` *window)
- void `window_update` (`Window` *window)

4.18.1 Function Documentation

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

4.18.1.2 void `window_destroy` (`Window` * *window*)

4.18.1.3 void `window_update` (`Window` * *window*)

Index

a

Line, [10](#)

alpha

Color, [8](#)

arrows

Event, [8](#)

b

Line, [10](#)

calc.h, [17](#)

calc_alea_float, [18](#)

calc_alea_int, [18](#)

calc_alea_float

calc.h, [18](#)

calc_alea_int

calc.h, [18](#)

Canvas, [5](#)

canvas.h, [21](#)

origin, [6](#)

parent, [6](#)

size, [6](#)

surface, [6](#)

canvas

Circle, [7](#)

Image, [9](#)

Line, [10](#)

Pixel, [11](#)

Rectangle, [13](#)

Sphere, [14](#)

canvas.h, [19](#)

Canvas, [21](#)

canvas_blit, [21](#)

canvas_clear, [22](#)

canvas_collision_canvas, [22](#)

canvas_create, [22](#)

canvas_create_from_window, [22](#)

canvas_draw_borders_in, [23](#)

canvas_draw_borders_out, [23](#)

canvas_fill, [23](#)

canvas_get_absolute_origin, [23](#)

canvas_is_out_of_parent_bottom, [23](#)

canvas_is_out_of_parent_left, [24](#)

canvas_is_out_of_parent_right, [24](#)

canvas_is_out_of_parent_top, [24](#)

canvas_is_out_of_parent_x, [24](#)

canvas_is_out_of_parent_y, [25](#)

canvas_will_be_out_of_parent_bottom, [25](#)

canvas_will_be_out_of_parent_left, [25](#)

canvas_will_be_out_of_parent_right, [26](#)

canvas_will_be_out_of_parent_top, [26](#)

canvas_will_be_out_of_parent_x, [26](#)

canvas_will_be_out_of_parent_y, [26](#)

canvas_blit

canvas.h, [21](#)

canvas_clear

canvas.h, [22](#)

canvas_collision_canvas

canvas.h, [22](#)

canvas_create

canvas.h, [22](#)

canvas_create_from_window

canvas.h, [22](#)

canvas_draw_borders_in

canvas.h, [23](#)

canvas_draw_borders_out

canvas.h, [23](#)

canvas_fill

canvas.h, [23](#)

canvas_get_absolute_origin

canvas.h, [23](#)

canvas_is_out_of_parent_bottom

canvas.h, [23](#)

canvas_is_out_of_parent_left

canvas.h, [24](#)

canvas_is_out_of_parent_right

canvas.h, [24](#)

canvas_is_out_of_parent_top

canvas.h, [24](#)

canvas_is_out_of_parent_x

canvas.h, [24](#)

canvas_is_out_of_parent_y

canvas.h, [25](#)

canvas_will_be_out_of_parent_bottom

canvas.h, [25](#)

canvas_will_be_out_of_parent_left

canvas.h, [25](#)

canvas_will_be_out_of_parent_right

canvas.h, [26](#)

canvas_will_be_out_of_parent_top

canvas.h, [26](#)

canvas_will_be_out_of_parent_x

canvas.h, [26](#)

canvas_will_be_out_of_parent_y

canvas.h, [26](#)

center

Circle, [7](#)

Sphere, [14](#)

Circle, [6](#)

- canvas, 7
- center, 7
- radius, 7
- circle.h, 27
 - circle_draw, 28
 - circle_draw_fill, 29
- circle_draw
 - circle.h, 28
- circle_draw_fill
 - circle.h, 29
- Color, 7
 - alpha, 8
 - rgb, 8
- color.h, 29
 - color_get_blue, 31
 - color_get_green, 31
 - color_get_red, 31
 - color_translate, 31
- color_get_blue
 - color.h, 31
- color_get_green
 - color.h, 31
- color_get_red
 - color.h, 31
- color_translate
 - color.h, 31
- content
 - Sound, 13
- error.h, 31
 - error_quit, 33
- error_quit
 - error.h, 33
- Event, 8
 - arrows, 8
 - quit, 8
 - space, 8
- event.h, 33
 - event_create, 34
 - event_update, 34
- event_create
 - event.h, 34
- event_update
 - event.h, 34
- graphics.h, 34
- graphics_start
 - startstop.h, 50
- graphics_stop
 - startstop.h, 50
- Image, 9
 - canvas, 9
 - surface, 9
- image.h, 35
 - image_blit_naive, 36
 - image_blit_scaled, 36
 - image_load, 36
 - image_unload, 36
- image_blit_naive
 - image.h, 36
- image_blit_scaled
 - image.h, 36
- image_load
 - image.h, 36
- image_unload
 - image.h, 36
- Line, 10
 - a, 10
 - b, 10
 - canvas, 10
- line.h, 36
 - line_draw, 38
 - line_draw_bis, 38
 - line_draw_ter, 38
- line_draw
 - line.h, 38
- line_draw_bis
 - line.h, 38
- line_draw_ter
 - line.h, 38
- mouse.h, 38
 - mouse_hide, 39
 - mouse_is_hidden, 39
 - mouse_is_shown, 39
 - mouse_show, 39
 - mouse_wait_click, 39
- mouse_hide
 - mouse.h, 39
- mouse_is_hidden
 - mouse.h, 39
- mouse_is_shown
 - mouse.h, 39
- mouse_show
 - mouse.h, 39
- mouse_wait_click
 - mouse.h, 39
- origin
 - Canvas, 6
 - Rectangle, 13
- parent
 - Canvas, 6
- Pixel, 11
 - canvas, 11
 - position, 11
- pixel.h, 39
 - pixel_draw, 40
- pixel_draw
 - pixel.h, 40
- Point, 11
 - x, 12
 - y, 12
- point.h, 41
 - point_are_equals, 42

- point_distance, 42
- point_max_x, 42
- point_max_y, 42
- point_min_x, 42
- point_min_y, 42
- point_are_equals
 - point.h, 42
- point_distance
 - point.h, 42
- point_max_x
 - point.h, 42
- point_max_y
 - point.h, 42
- point_min_x
 - point.h, 42
- point_min_y
 - point.h, 42
- position
 - Pixel, 11
 - Window, 15
- quit
 - Event, 8
- radius
 - Circle, 7
 - Sphere, 14
- Rectangle, 12
 - canvas, 13
 - origin, 13
 - size, 13
- rectangle.h, 42
 - rectangle_contains_absolute_point, 45
 - rectangle_contains_point, 45
 - rectangle_draw, 45
 - rectangle_draw_fill, 45
- rectangle_contains_absolute_point
 - rectangle.h, 45
- rectangle_contains_point
 - rectangle.h, 45
- rectangle_draw
 - rectangle.h, 45
- rectangle_draw_fill
 - rectangle.h, 45
- rgb
 - Color, 8
- screen.h, 45
 - screen_get_size, 46
- screen_get_size
 - screen.h, 46
- size
 - Canvas, 6
 - Rectangle, 13
 - Window, 15
- Sound, 13
 - content, 13
- sound.h, 46
 - sound_free, 47
 - sound_load, 47
 - sound_pause, 47
 - sound_play, 47
 - sound_play_once, 47
 - sound_resume, 47
 - sound_stop, 47
- sound_free
 - sound.h, 47
- sound_load
 - sound.h, 47
- sound_pause
 - sound.h, 47
- sound_play
 - sound.h, 47
- sound_play_once
 - sound.h, 47
- sound_resume
 - sound.h, 47
- sound_stop
 - sound.h, 47
- space
 - Event, 8
- Sphere, 13
 - canvas, 14
 - center, 14
 - radius, 14
- sphere.h, 48
 - sphere_draw_fill, 49
- sphere_draw_fill
 - sphere.h, 49
- startstop.h, 49
 - graphics_start, 50
 - graphics_stop, 50
- surface
 - Canvas, 6
 - Image, 9
- title
 - Window, 15
- Window, 14
 - position, 15
 - size, 15
 - title, 15
 - window, 15
- window
 - Window, 15
- window.h, 50
 - window_create, 52
 - window_destroy, 52
 - window_update, 52
- window_create
 - window.h, 52
- window_destroy
 - window.h, 52
- window_update
 - window.h, 52
- x

Point, [12](#)

y

Point, [12](#)