Graphics

0.0.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

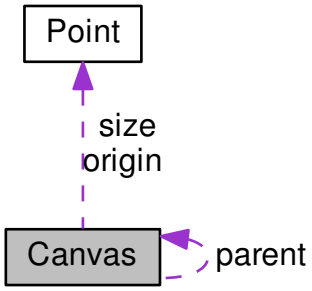
Data Structure Documentation

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 souldn'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:

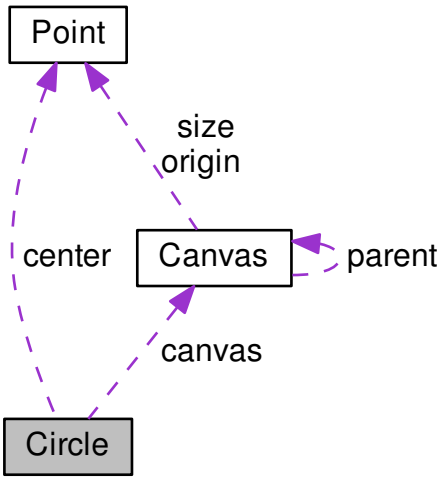
- canvas.h

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:

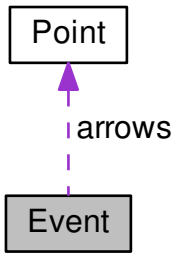
- color.h

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:

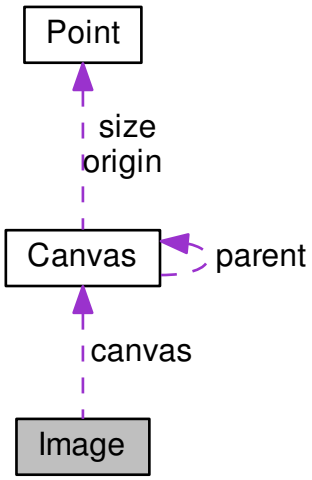
- event.h

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:

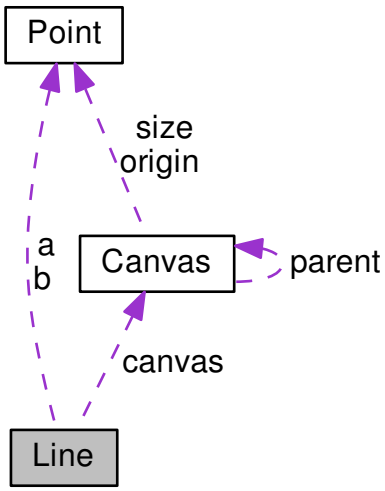
- image.h

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:

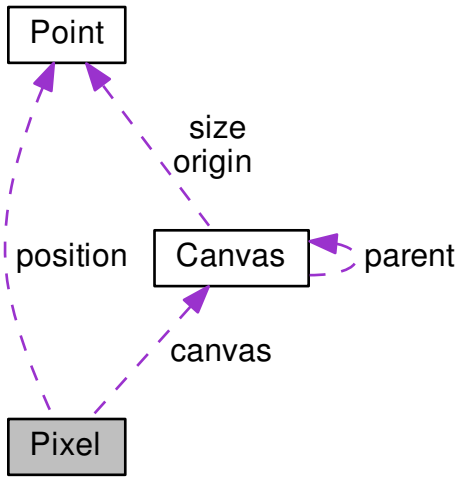
- line.h

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:

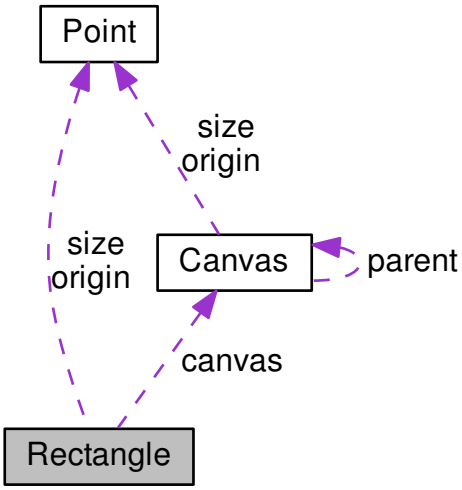
- point.h

3.9 Rectangle Struct Reference

A struct used to represent a rectanglec.

`#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 rectanglec.

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.

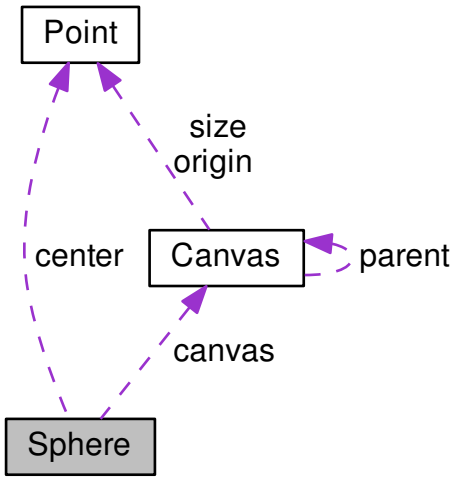
The documentation for this struct was generated from the following file:

- 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.

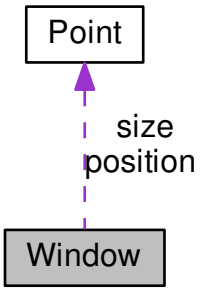
The documentation for this struct was generated from the following file:

- 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

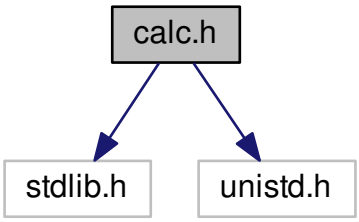
File Documentation

4.1 calc.h File Reference

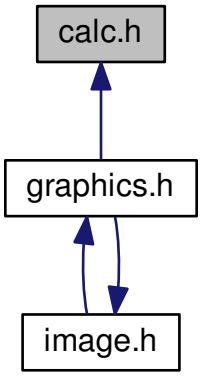
Some maths functions.

```
#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

min	The minimun value for the random int.
max	The maximum value for the random int.

Returns

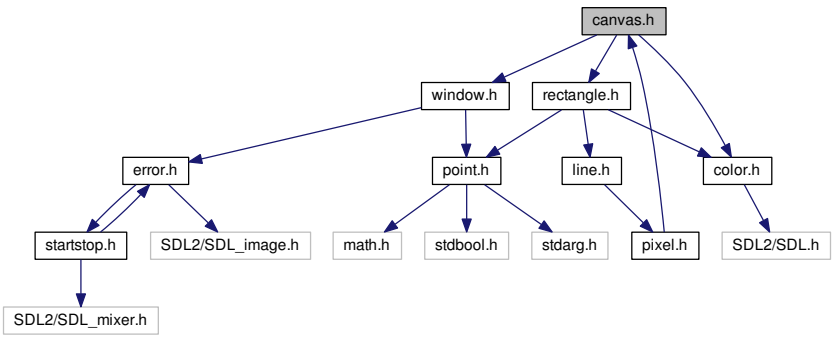
The random int.

4.2 canvas.h File Reference

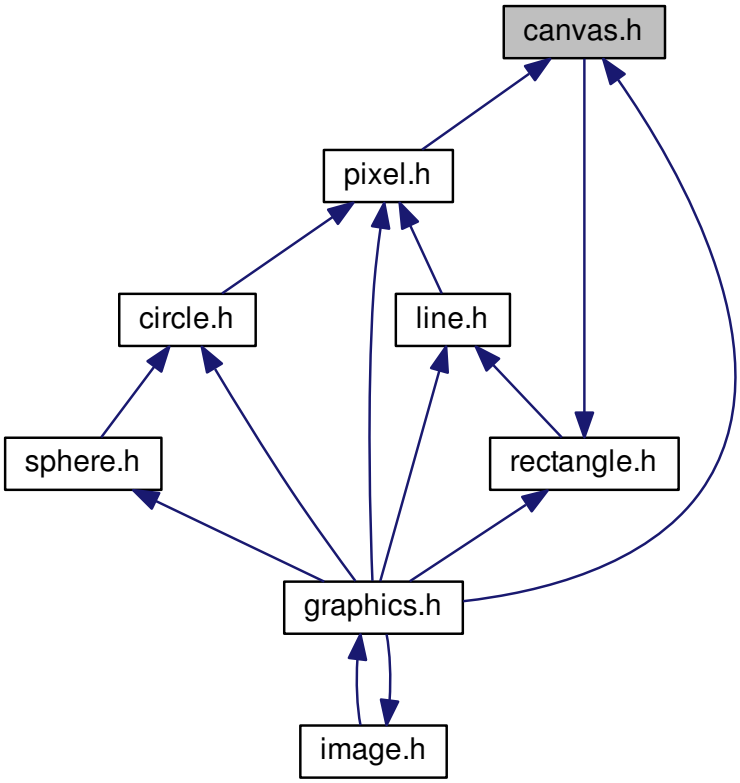
Everything related to Canvas.

```
#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

canvas	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

canvas	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

canvas1	A pointer to the first Canvas.
canvas2	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

canvas	A pointer to the Canvas to create.
size	A pointer to a Point representing the wanted size for the Canvas.
origin	A pointer to a Point representig the wanter origin for the Canvas.
parent	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

canvas	A pointer to the Canvas to create.
window	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

canvas	A pointer to the Canvas.
color	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

canvas	A pointer to the Canvas.
color	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

canvas	A pointer to the Canvas to fill.
color	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

canvas	A pointer to the Canvas.
absoluteOrigin	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

canvas	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

canvas	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

canvas	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

canvas	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

canvas	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

canvas	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

canvas	A pointer to the Canvas.
move	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

canvas	A pointer to the Canvas.
move	A pointer to the Point representing the origin's move.

Returns

If the Canvas will be 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

canvas	A pointer to the Canvas.
move	A pointer to the Point representing the origin's move.

Returns

If the Canvas will be 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

canvas	A pointer to the Canvas.
move	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

canvas	A pointer to the Canvas.
move	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

canvas	A pointer to the Canvas.
move	A pointer to the point representing the origin's move.

Returns

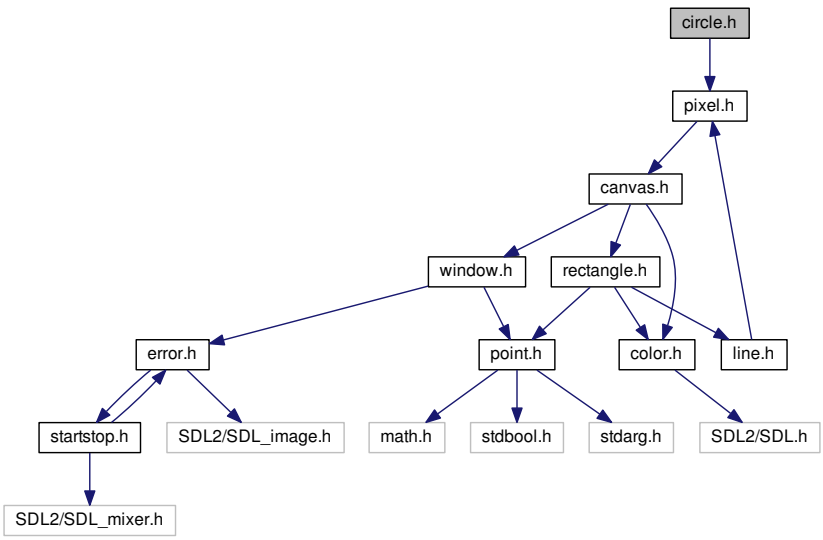
If the Canvas will be outside of its parent on the Y axis, returns true, else, returns false.

4.3 circle.h File Reference

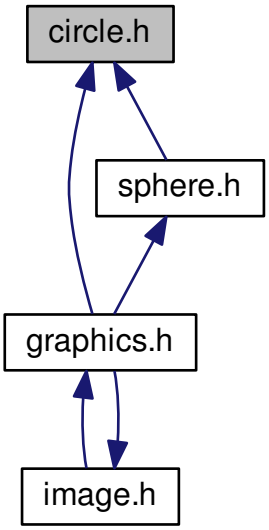
Everything related to Circle.

```
#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

circle	A pointer to the Circle to draw.
color	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

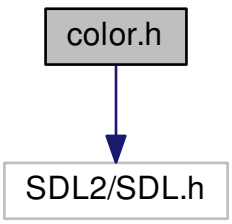
circle	A pointer to the Circle to draw.
color	A pointer to the Color to use to draw the Circle.

4.4 color.h File Reference

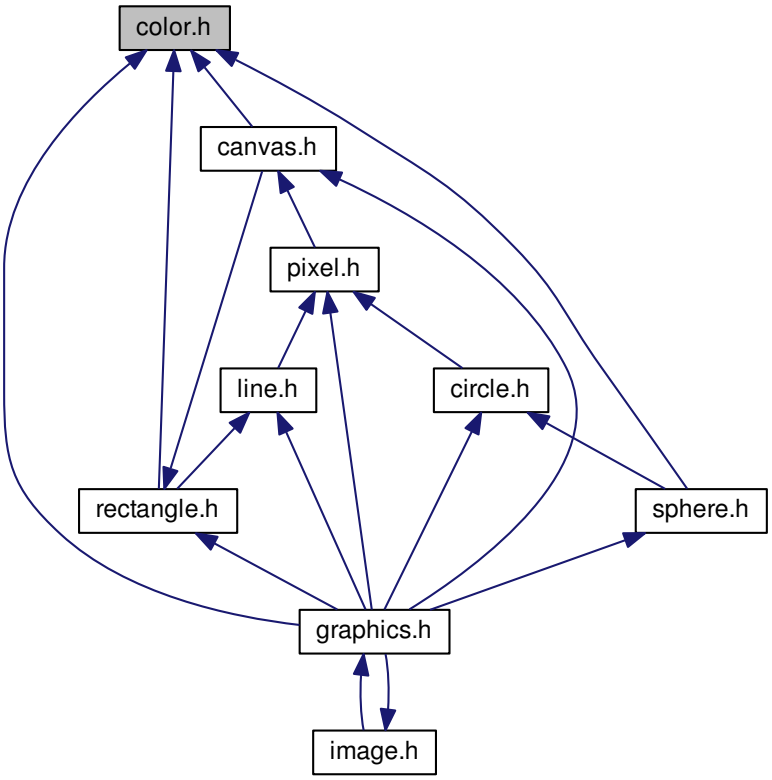
Everything related to Color.

```
#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

canvas1	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

canvas1	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

canvas1	A pointer to the Color.

Returns

The red component in a Uint8.

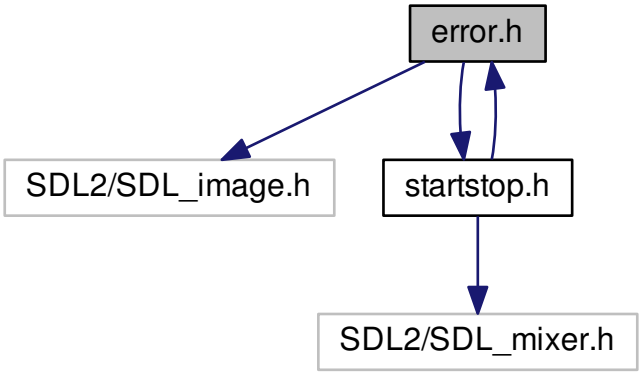
4.4.2.4 void color_translate (const Color * *color*, SDL_Color * *sdlColor*)

4.5 error.h File Reference

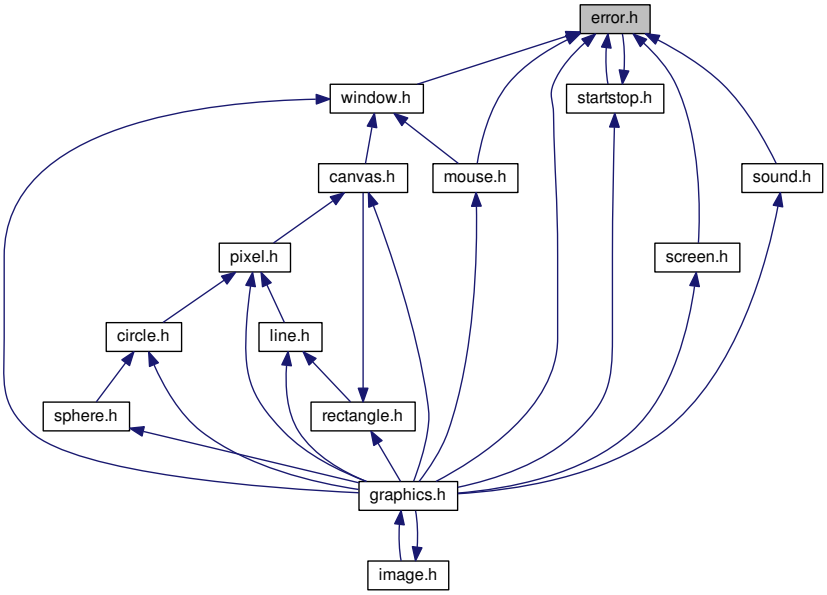
Everything related to errors and warnings handling.

```
#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)

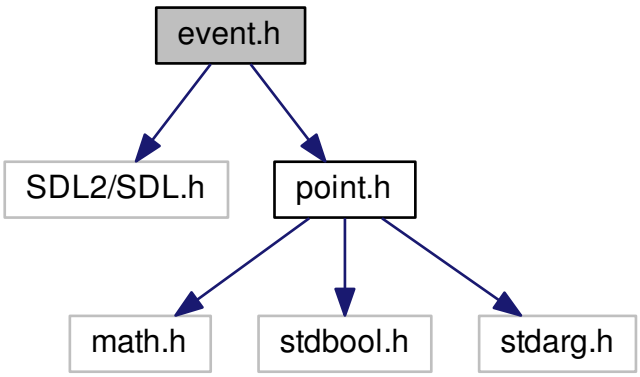
Function to quit after an error, will stop graphics and SDL components and stop the program.

4.6 event.h File Reference

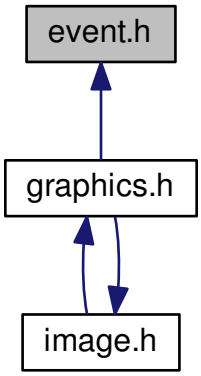
Everything related to events, i.e. user input.

```
#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

newEvent	A pointer to the Event to create.

4.6.2.2 void event_update (Event * *event*)

Function to update an Event.

Parameters

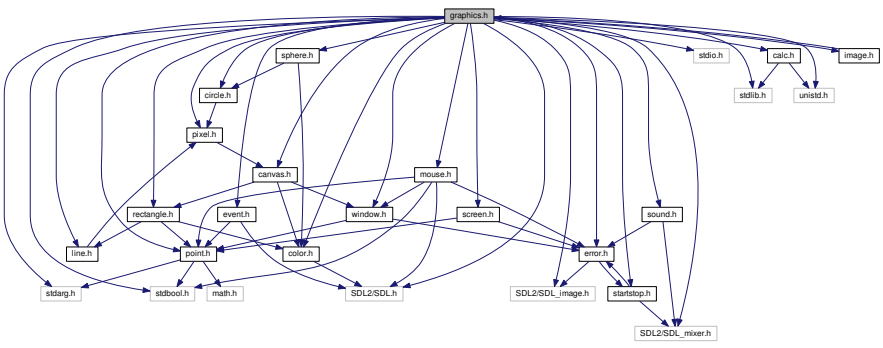
newEvent	A pointer to the Event to update.

4.7 graphics.h File Reference

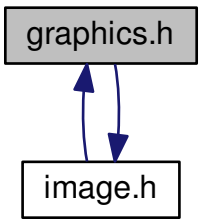
The main lib file.

```
#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.

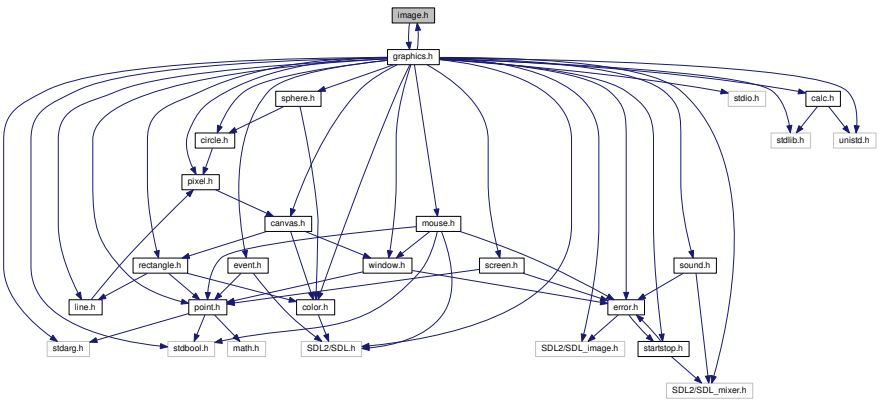
It's the file to include when using the lib in a program. It includes all the others headers and dependencies.

4.8 image.h File Reference

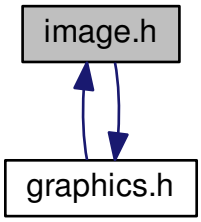
Everything related to Image.

```
#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

image	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

image	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

image	A pointer to the Image used to store the loaded image.
pathToImg	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

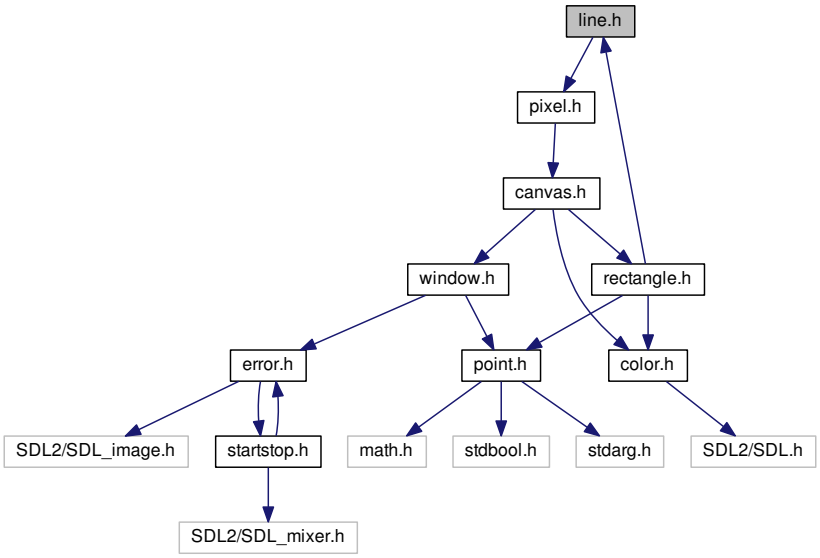
image	A pointer to the Image to unload.

4.9 line.h File Reference

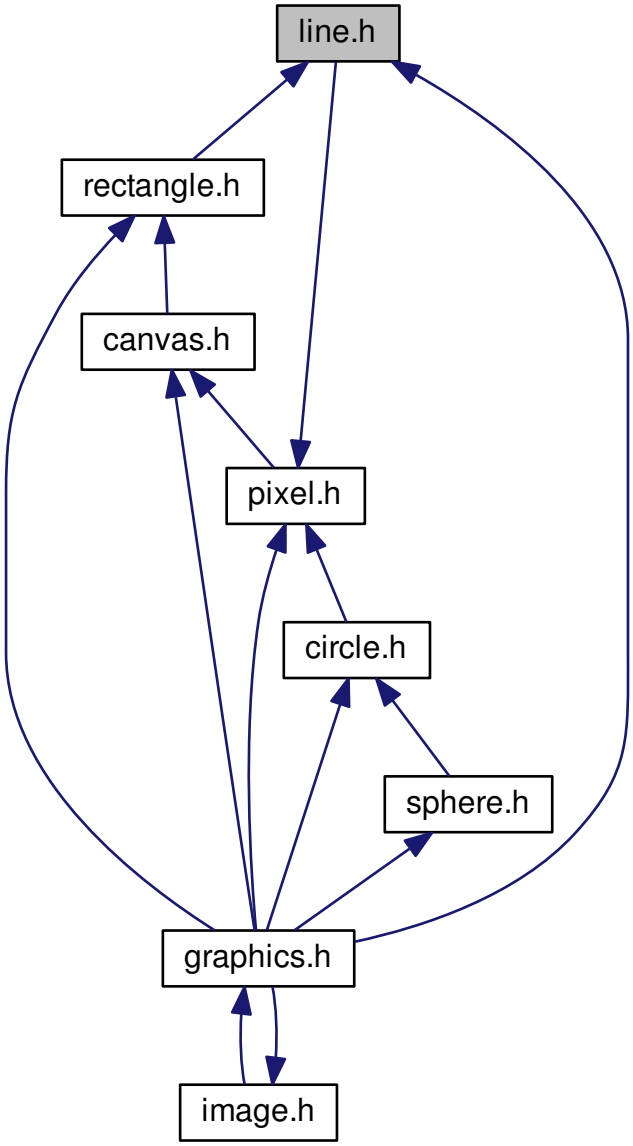
Everything related to Line.

```
#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

line	A pointer to the Line to draw.
color	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

line	A pointer to the Line to draw.
color	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

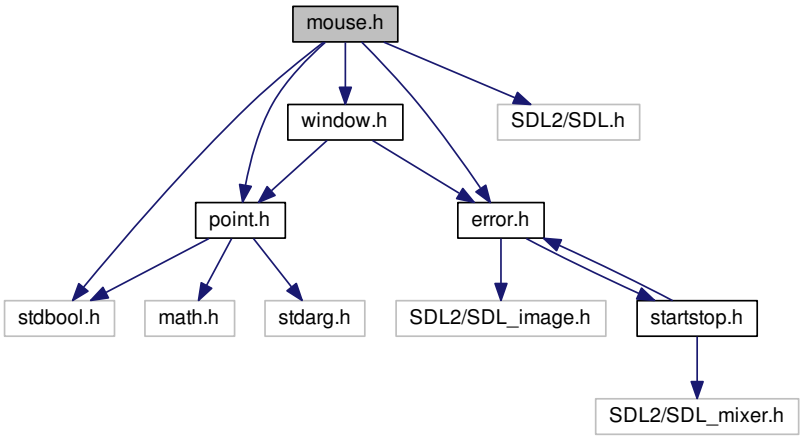
line	A pointer to the Line to draw.
color	A pointer to the Color to use to draw the Line.

4.10 mouse.h File Reference

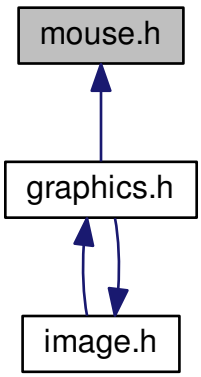
Everything related to the mouse.

```
#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.

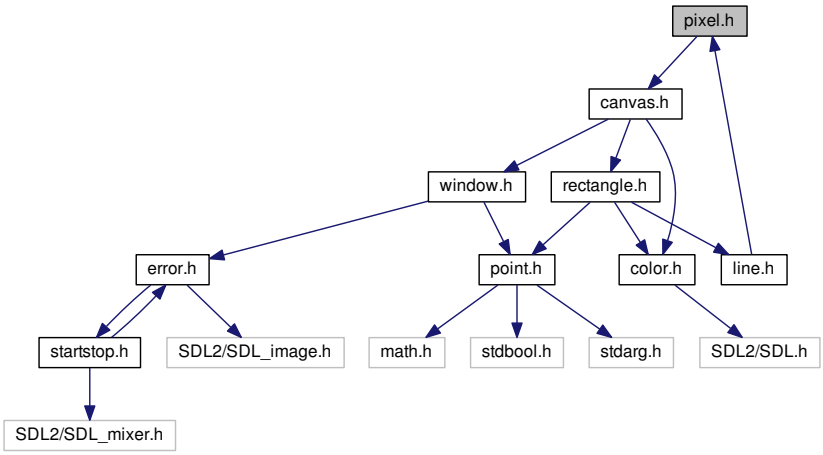
Parameters

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

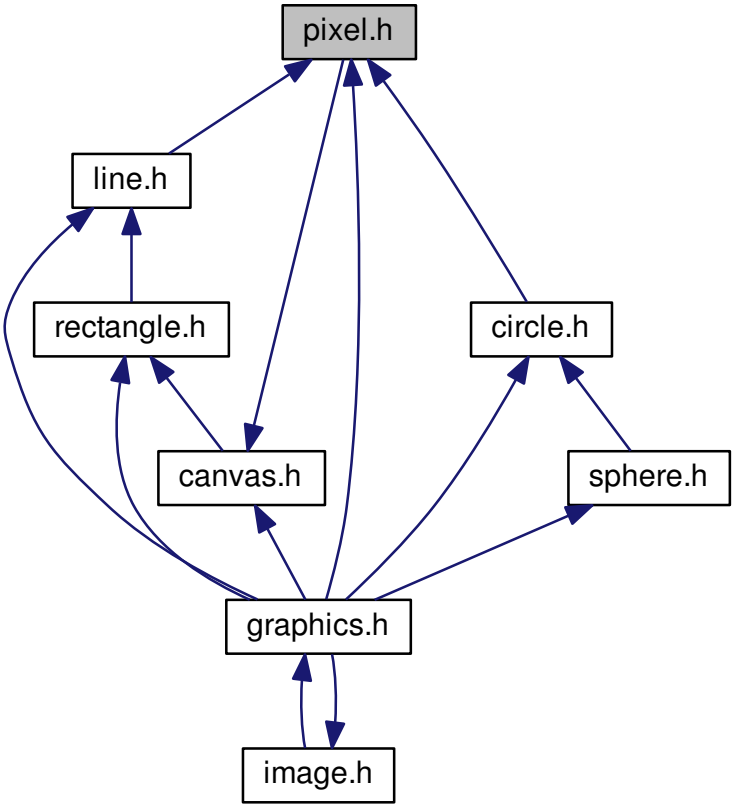
4.11 pixel.h File Reference

```
#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

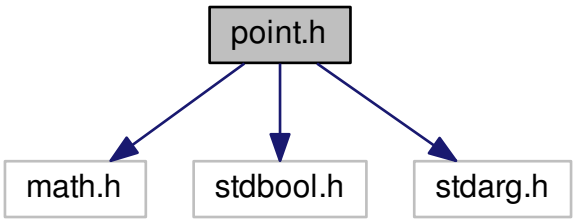
line	A pointer to the Pixel to draw.
color	A pointer to the Color to use to draw the Pixel.

4.12 point.h File Reference

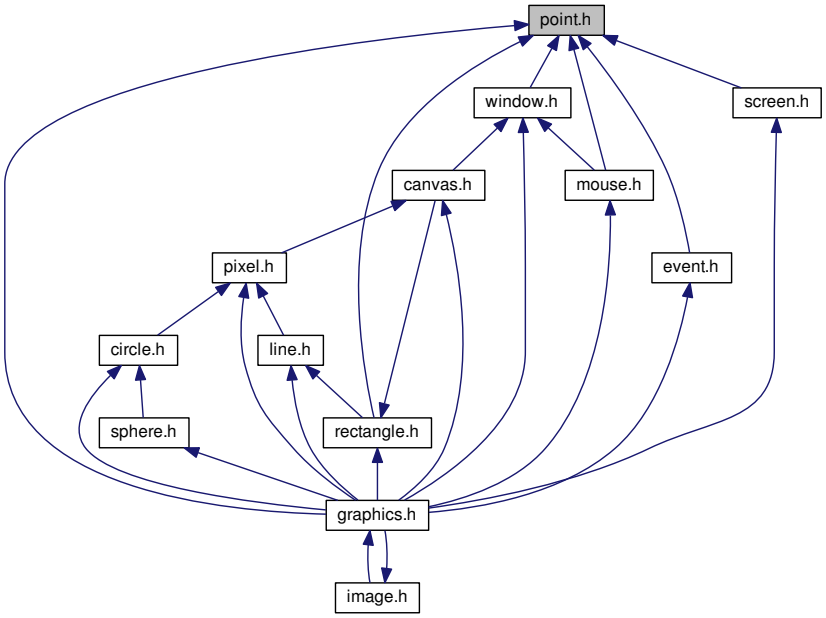
Everything related to Point.

```
#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

p1	The first Point.
p2	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

a	The first Point.
b	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

a	The first Point.
b	The second Point.

Returns

The Point with the biggest x.

Parameters

a	The first Point.
b	The second Point.

Returns

The Point with the smallest x.

Parameters

a	The first Point.
b	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

a	The first Point.
b	The second Point.

Returns

The Point with the biggest y.

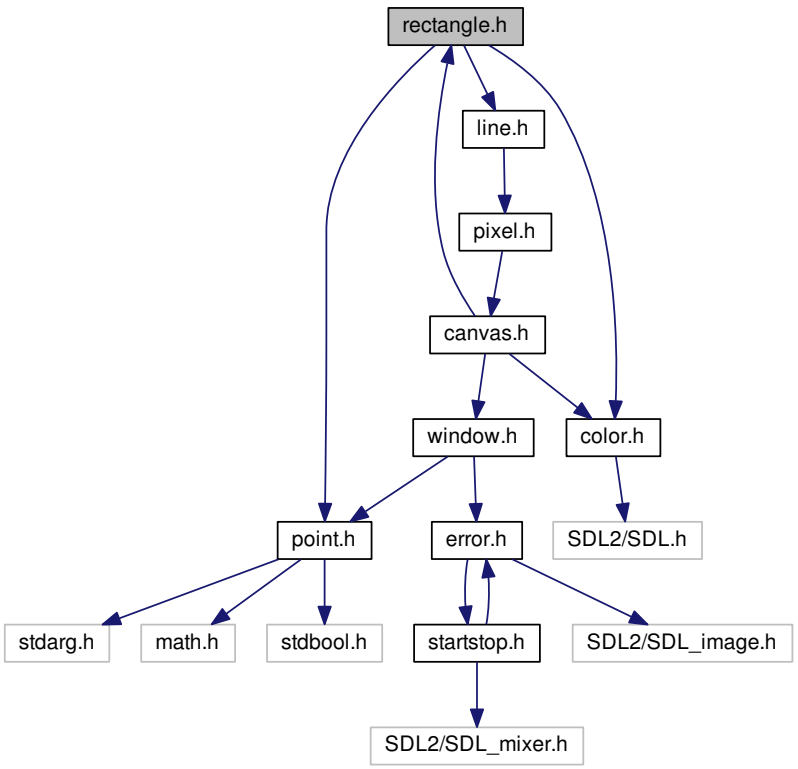
4.12.2.5 Point point_min_x (const Point *a*, const Point *b*)

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

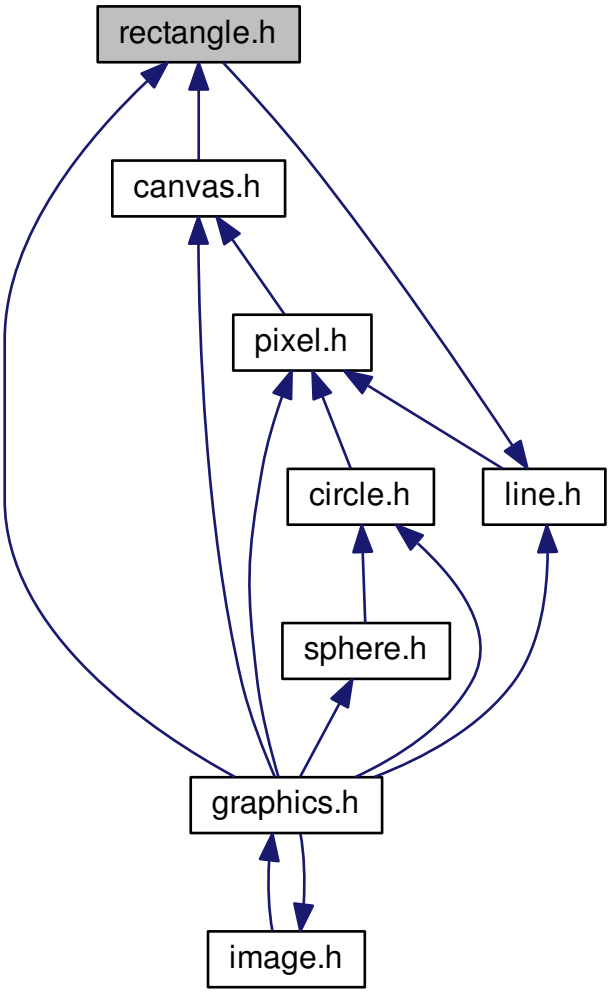
4.13 rectangle.h File Reference

```
#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 rectanglec.

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

rect	A pointer to the Rectangle.
p	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

rect	A pointer to the Rectangle.
p	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

circle	A pointer to the Rectangle to draw.
color	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

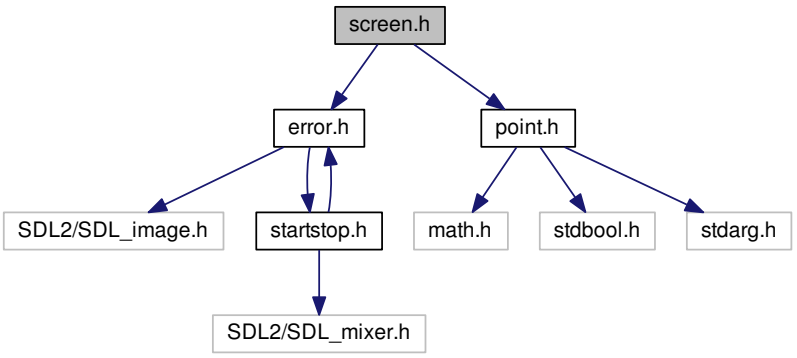
circle	A pointer to the Rectangle to draw.
color	A pointer to the Color to use to draw the Rectangle.

4.14 screen.h File Reference

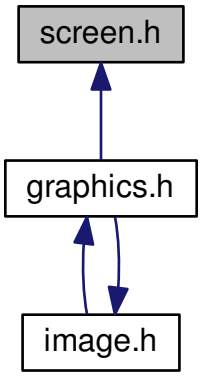
Everything related to the screen.

```
#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

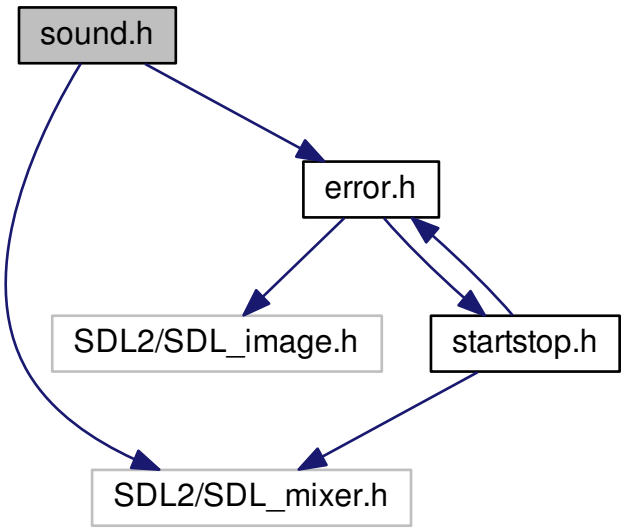
screenSize	A pointer to the Point in which the screen's size must be stored.

4.15 sound.h File Reference

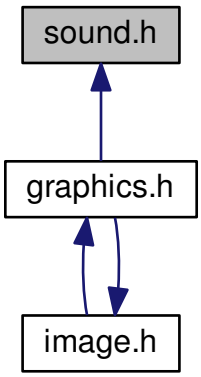
Everything related to Sound.

```
#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

sound	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

pathToFile	The path to the file to load.
sound	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

music	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

music	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)

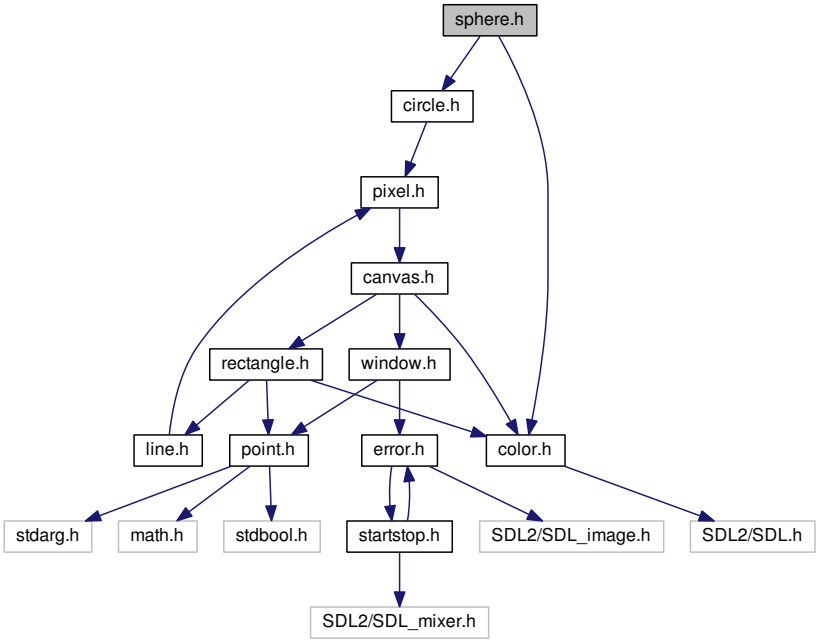
Function to stop the current played Sound.

4.16 sphere.h File Reference

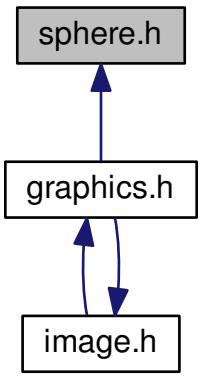
Everything related to Sphere.

```
#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

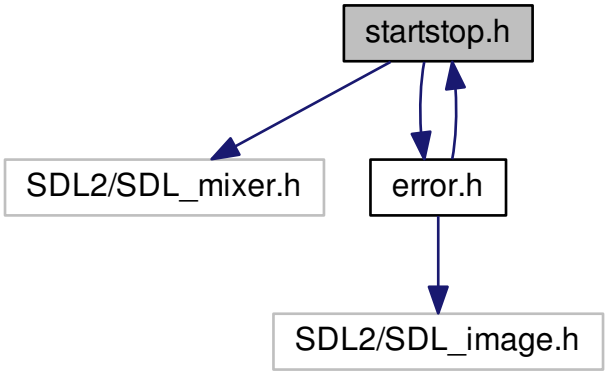
sphere	A pointer to the Sphere to draw.
color	A pointer to the Color to use to draw the Sphere.

4.17 startstop.h File Reference

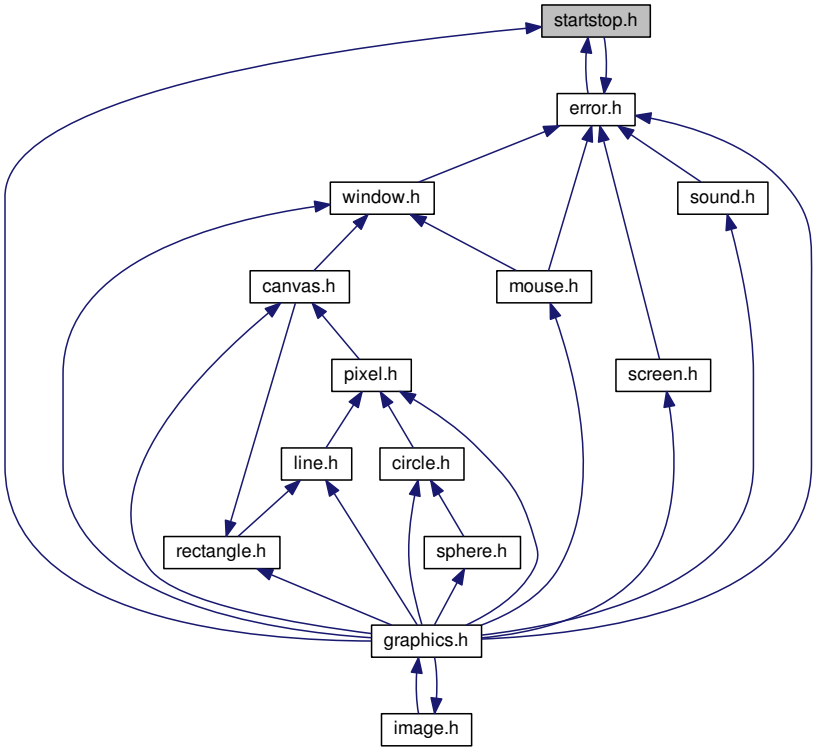
Everything related to graphics' start and stop functions.

```
#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

flags	A list of SDL flags, if you don't know, use SDL_INIT_EVERYTHING, or see SDL_Init doc.

4.17.2.2 void graphics_stop (void)

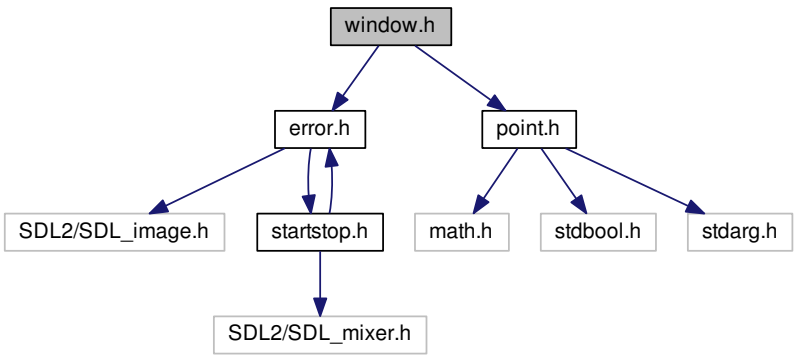
Function to stop graphics.

4.18 window.h File Reference

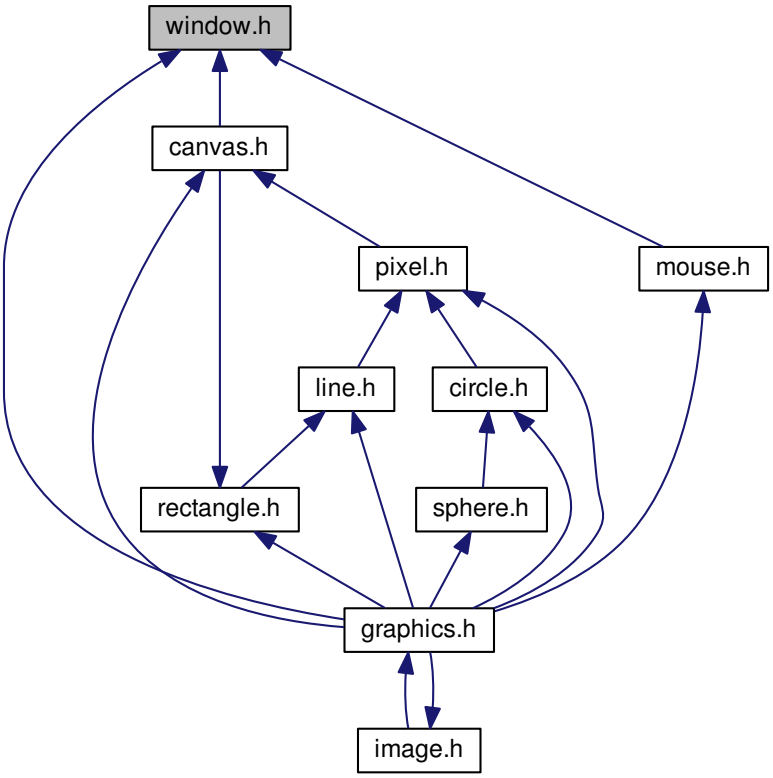
Everything related to Window.

```
#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 destory 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

window	A pointer to the Window in which the new Window will be stored.
title	The title wanted for the Window.
position	A pointer to a Point representing the position wanted for the Window.
size	A pointer to a Point representing the size wanted for the Window.
flags	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 destory a Window.

Parameters

window	A pointer to the Window to destroy.

4.18.2.3 void window_update (Window * *window*)

Function to update a Window.

Parameters

window	A pointer to the Window to update.

Index