

Super Martin Level Editor

Generated by Doxygen 1.8.6

Mon Jun 2 2014 13:59:31

## Contents

<b>1</b>	<b>Data Structure Documentation</b>	<b>1</b>
1.1	Cursor Struct Reference	1
1.1.1	Detailed Description	1
1.1.2	Field Documentation	1
1.2	Input Struct Reference	2
1.2.1	Detailed Description	2
1.2.2	Field Documentation	2
1.3	Level Struct Reference	2
1.3.1	Detailed Description	3
1.3.2	Field Documentation	3
1.4	Map Struct Reference	3
1.4.1	Detailed Description	4
1.4.2	Field Documentation	4
<b>2</b>	<b>File Documentation</b>	<b>4</b>
2.1	const.h File Reference	4
2.1.1	Detailed Description	5
2.1.2	Macro Definition Documentation	5
2.2	file.c File Reference	6
2.2.1	Detailed Description	6
2.2.2	Function Documentation	6
2.3	file.h File Reference	7
2.3.1	Detailed Description	7
2.3.2	Function Documentation	7
2.4	file_level.c File Reference	8
2.4.1	Detailed Description	8
2.4.2	Function Documentation	8
2.5	file_level.h File Reference	11
2.5.1	Detailed Description	12
2.5.2	Macro Definition Documentation	12
2.5.3	Function Documentation	12
2.6	game.c File Reference	15
2.6.1	Detailed Description	15
2.6.2	Function Documentation	15
2.7	game.h File Reference	17
2.7.1	Detailed Description	17
2.7.2	Function Documentation	17
2.8	image.c File Reference	18

2.8.1	Detailed Description	19
2.8.2	Function Documentation	19
2.9	image.h File Reference	19
2.9.1	Detailed Description	20
2.9.2	Function Documentation	20
2.10	input.c File Reference	20
2.10.1	Detailed Description	20
2.10.2	Function Documentation	21
2.11	main.c File Reference	22
2.11.1	Detailed Description	22
2.11.2	Function Documentation	22
2.12	map.c File Reference	23
2.12.1	Detailed Description	24
2.12.2	Function Documentation	24
2.13	map.h File Reference	29
2.13.1	Detailed Description	30
2.13.2	Function Documentation	30
2.14	menu.c File Reference	35
2.14.1	Detailed Description	36
2.14.2	Function Documentation	36
2.15	menu.h File Reference	37
2.15.1	Detailed Description	38
2.15.2	Function Documentation	38
2.16	menu_level.c File Reference	39
2.16.1	Detailed Description	39
2.16.2	Function Documentation	40
2.17	menu_level.h File Reference	40
2.17.1	Detailed Description	41
2.17.2	Function Documentation	41
2.18	menu_option.c File Reference	42
2.18.1	Detailed Description	42
2.18.2	Function Documentation	43
2.19	menu_option.h File Reference	46
2.19.1	Detailed Description	46
2.19.2	Function Documentation	47
2.20	option.c File Reference	49
2.20.1	Detailed Description	49
2.20.2	Function Documentation	49
2.21	option.h File Reference	50
2.21.1	Detailed Description	51

2.21.2	Function Documentation	51
2.22	share.c File Reference	52
2.22.1	Detailed Description	52
2.22.2	Function Documentation	52
2.23	share.h File Reference	53
2.23.1	Detailed Description	53
2.23.2	Function Documentation	53
2.24	text.c File Reference	53
2.24.1	Detailed Description	54
2.24.2	Function Documentation	54
2.25	text.h File Reference	55
2.25.1	Detailed Description	55
2.25.2	Function Documentation	55
<b>Index</b>		<b>57</b>

## 1 Data Structure Documentation

### 1.1 Cursor Struct Reference

```
#include <structures.h>
```

#### Data Fields

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

#### 1.1.1 Detailed Description

The cursor structure

#### 1.1.2 Field Documentation

##### 1.1.2.1 char [tileID](#)

The tile ID

##### 1.1.2.2 int [x](#)

The x coordinate

##### 1.1.2.3 int [y](#)

The y coordinate

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

- [structures.h](#)

## 1.2 Input Struct Reference

```
#include <structures.h>
```

### Data Fields

- char `key` [SDLK\_LAST]
- char `mouse` [6]
- int `mouseX`
- int `mouseY`
- int `quit`

### 1.2.1 Detailed Description

The input structure

### 1.2.2 Field Documentation

#### 1.2.2.1 char `key`[SDLK\_LAST]

all the keyboard keys : 1 the key is pushed, 0 isn't

#### 1.2.2.2 char `mouse`[6]

all the mouse buttons : 1 the button is pushed, 0 isn't

#### 1.2.2.3 int `mouseX`

The x coordinate of the mouse

#### 1.2.2.4 int `mouseY`

The y coordinate of the mouse

#### 1.2.2.5 int `quit`

is 1 is the SDL\_QUIT event happens

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

- `structures.h`

## 1.3 Level Struct Reference

```
#include <structures.h>
```

### Data Fields

- unsigned char \*\* `map`
- int `width`
- int `height`
- int `timer_level`
- char `tileSet` [MAX\_LENGTH\_FILE\_NAME]
- char `background` [MAX\_LENGTH\_FILE\_NAME]
- char `music` [MAX\_LENGTH\_FILE\_NAME]

### 1.3.1 Detailed Description

The level structure

### 1.3.2 Field Documentation

#### 1.3.2.1 char background[MAX\_LENGTH\_FILE\_NAME]

The background

#### 1.3.2.2 int height

The height

#### 1.3.2.3 unsigned char\*\* map

The map

#### 1.3.2.4 char music[MAX\_LENGTH\_FILE\_NAME]

The music

#### 1.3.2.5 char tileSet[MAX\_LENGTH\_FILE\_NAME]

The tilset

#### 1.3.2.6 int timer\_level

The timer level

#### 1.3.2.7 int width

The width

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

- structures.h

## 1.4 Map Struct Reference

```
#include <structures.h>
```

Collaboration diagram for Map:



## Data Fields

- [Level \\* lvl](#)
- [int xScroll](#)
- [int screenWidth](#)
- [int screenHeight](#)

### 1.4.1 Detailed Description

The map structure

### 1.4.2 Field Documentation

#### 1.4.2.1 [Level\\* lvl](#)

The level

#### 1.4.2.2 [int screenHeight](#)

The screen height

#### 1.4.2.3 [int screenWidth](#)

The Screen width

#### 1.4.2.4 [int xScroll](#)

The xscroll

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

- [structures.h](#)

## 2 File Documentation

### 2.1 [const.h](#) File Reference

contient les constantes du programme

#### Macros

- [#define TILE\\_SIZE 16](#)
- [#define NB\\_TILES\\_X 80](#)
- [#define NB\\_TILES\\_Y 45](#)
- [#define SCREEN\\_WIDTH TILE\\_SIZE \\* NB\\_TILES\\_X](#)
- [#define SCREEN\\_HEIGHT TILE\\_SIZE \\* NB\\_TILES\\_Y](#)
- [#define TILESET\\_SIZE 20](#)
- [#define OPTIONS\\_PER\\_COLUMN 9](#)
- [#define FPS 60](#)
- [#define SCROLLING\\_MARGIN 2](#)
- [#define MAX\\_LENGTH\\_FILE\\_NAME 100](#)
- [#define min\(a, b\) \(a<=b?a:b\)](#)

## Enumerations

- enum {  
    **VOID =0, GROUND, COIN =7, ROCK,**  
    **SPRING, HAMMER, HEART, ADDLIFE,**  
    **ENEMY, CANON\_R, CANON\_L, CANON\_B,**  
    **TREE, FLOWER, CLOUD, PLATFORMV,**  
    **PLATFORMH }**
- enum { **RIGHT, LEFT, UP, DOWN }**

### 2.1.1 Detailed Description

contient les constantes du programme

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-02-27

### 2.1.2 Macro Definition Documentation

#### 2.1.2.1 #define FPS 60

The FPS

#### 2.1.2.2 #define MAX\_LENGTH\_FILE\_NAME 100

The size max of the filenames

#### 2.1.2.3 #define min( a, b ) (a<=b?a:b)

mix

#### 2.1.2.4 #define NB\_TILES\_X 80

The number of tile on x

#### 2.1.2.5 #define NB\_TILES\_Y 45

The number of tile on y

#### 2.1.2.6 #define OPTIONS\_PER\_COLUMN 9

The number of options per column

#### 2.1.2.7 #define SCREEN\_HEIGHT TILE\_SIZE \* NB\_TILES\_Y

The screen height

#### 2.1.2.8 #define SCREEN\_WIDTH TILE\_SIZE \* NB\_TILES\_X

The screen width

#### 2.1.2.9 #define SCROLLING\_MARGIN 2

The scrolling margin



### 2.1.2.10 #define TILE\_SIZE 16

The tile size

### 2.1.2.11 #define TILESET\_SIZE 20

The tilset size

## 2.2 file.c File Reference

Functions to access to the files.

```
#include "file.h"
```

### Functions

- FILE \* [openFile](#) (char name[], char mode[])
- int [closeFile](#) (FILE \*ptr\_file)

### 2.2.1 Detailed Description

Functions to access to the files.

#### Author

Remi BERTHO, Glenn HERROU

#### Date

2014-05-12

### 2.2.2 Function Documentation

#### 2.2.2.1 int closeFile ( FILE \* *ptr\_file* )

Close the given file

#### Parameters

in	<i>*ptr_file</i>	the file
----	------------------	----------

#### Returns

0 if the file has been succesfully closed, 1 otherwise

#### 2.2.2.2 FILE \* openFile ( char *name*[], char *mode*[] )

Open a file which path is name with the given mode

#### Parameters

in	<i>name</i> []	name of the file
----	----------------	------------------

in	mode[]	the opening mode
----	--------	------------------

#### Returns

a pointer if the opening has been succesfull, NULL otherwise

## 2.3 file.h File Reference

#### file.c header

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
```

#### Functions

- FILE \* [openFile](#) (char nome[], char mode[])
- int [closeFile](#) (FILE \*ptr\_fichier)

#### 2.3.1 Detailed Description

##### file.c header

##### Author

Remi BERTHO, Glenn HERROU

##### Date

2014-05-12

#### 2.3.2 Function Documentation

##### 2.3.2.1 int closeFile ( FILE \* ptr\_file )

Close the given file

##### Parameters

in	*ptr_file	the file
----	-----------	----------

#### Returns

0 if the file has been succesfully closed, 1 otherwise

##### 2.3.2.2 FILE\* openFile ( char name[], char mode[] )

Open a file which path is name with the given mode

##### Parameters

in	name[]	name of the file
----	--------	------------------

<code>in</code>	<code>mode[]</code>	the opening mode
-----------------	---------------------	------------------

#### Returns

a pointer if the opening has been succesfull, NULL otherwise

## 2.4 file\_level.c File Reference

Management of the level files.

```
#include "file_level.h"
```

#### Functions

- `Level * openLevel (char *file_name)`
- `void closeLevel (Level *lvl)`
- `Level * initLevel (Level *lvl)`
- `void writeLevel (char *file_name, Level *lvl)`
- `char ** readLevelFile (int *nb_lvl)`
- `void closeLevelList (char **level_names, int nb_lvl)`
- `Level * adaptSizeLevel (Level *lvl)`
- `int searchEndLevel (Level *lvl)`

### 2.4.1 Detailed Description

Management of the level files.

#### Author

Remi BERTHO, Glenn HERROU

#### Date

2014-05-12

#### Version

2.0

### 2.4.2 Function Documentation

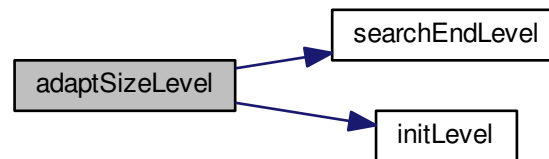
#### 2.4.2.1 `Level * adaptSizeLevel ( Level * lvl )`

Adapt the width of a level

#### Parameters

<code>in, out</code>	<code>lvl</code>	the level to adapt
----------------------	------------------	--------------------

Here is the call graph for this function:



#### 2.4.2.2 void closeLevel ( Level \* lvl )

Close a level by freeing its allocated memory

##### Parameters

out	lvl	The level
-----	-----	-----------

#### 2.4.2.3 void closeLevelList ( char \*\* level\_names, int nb\_lvl )

Free the array containing the list of the levels

##### Parameters

in, out	level_names	the list of existing levels
in	nb_lvl	the number of existing levels

#### 2.4.2.4 Level \* initLevel ( Level \* lvl )

Initialize a level. The width and the Height of the level must be stored in the level before calling this function.

##### Parameters

out	lvl	The level
-----	-----	-----------

##### Returns

a pointer on the level initialized

#### 2.4.2.5 Level \* openLevel ( char \* file\_name )

Open a level file and store the level corresponding to the file

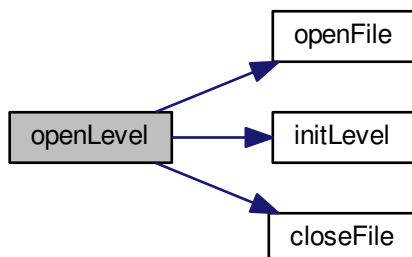
##### Parameters

in	file_name	the name of the level file
----	-----------	----------------------------

**Returns**

a pointer to the level created

Here is the call graph for this function:

**2.4.2.6 char \*\* readLevelFile ( int \* nb\_lv )**

Read the file including the list of existing levels

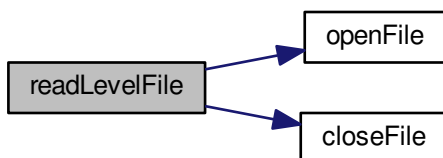
**Parameters**

out	<i>nb_lv</i>	the number of level in the file
-----	--------------	---------------------------------

**Returns**

a pointer to an array of strings containing the list of the levels

Here is the call graph for this function:

**2.4.2.7 int searchEndLevel ( Level \* lv )**

Search the end of a level

## Parameters

in, out	lvl	the level
---------	-----	-----------

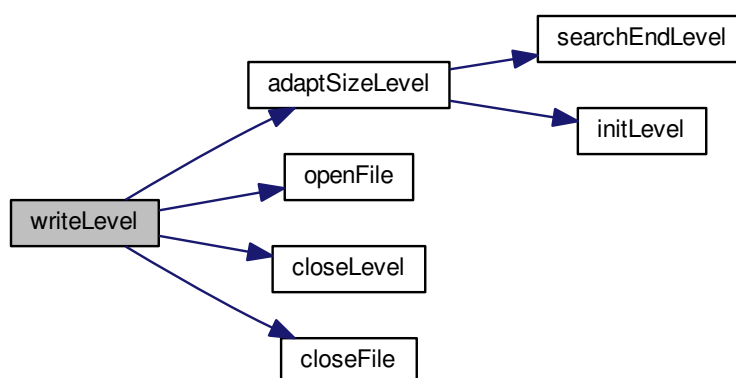
## 2.4.2.8 void writeLevel ( char \* file\_name, Level \* lvl )

Write the given level in the given file

## Parameters

in	lvl	the file
in	file_name	the name of the file

Here is the call graph for this function:



## 2.5 file\_level.h File Reference

## file\_level.c header

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "file.h"
#include "const.h"
#include "structures.h"

```

## Macros

- `#define BUFFER_SIZE 2`

## Functions

- `Level * openLevel (char *file_name)`
- `void closeLevel (Level *lvl)`
- `Level * initLevel (Level *lvl)`
- `void writeLevel (char *file_name, Level *lvl)`
- `char ** readLevelFile (int *nb_lvl)`

- void [closeLevelList](#) (char \*\*level\_names, int nb\_lvl)
- [Level \\*](#) [adaptSizeLevel](#) ([Level \\*](#)lvl)
- int [searchEndLevel](#) ([Level \\*](#)lvl)

### 2.5.1 Detailed Description

[file\\_level.c](#) header

#### Author

Remi BERTHO, Glenn HERROU

#### Date

2014-05-12

#### Version

2.0

### 2.5.2 Macro Definition Documentation

#### 2.5.2.1 #define BUFFER\_SIZE 2

The buffer size

### 2.5.3 Function Documentation

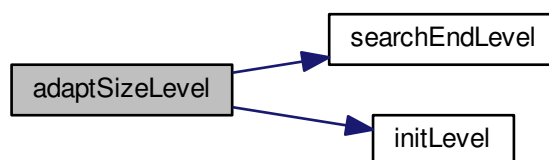
#### 2.5.3.1 [Level\\*](#) [adaptSizeLevel](#) ( [Level \\*](#) lvl )

Adapt the width of a level

##### Parameters

<i>in, out</i>	<i>lvl</i>	the level to adapt
----------------	------------	--------------------

Here is the call graph for this function:



#### 2.5.3.2 void [closeLevel](#) ( [Level \\*](#) lvl )

Close a level by freeing its allocated memory

## Parameters

out	<i>lvl</i>	The level
-----	------------	-----------

2.5.3.3 void closeLevelList ( char \*\* *level\_names*, int *nb\_lvl* )

Free the array containing the list of the levels

## Parameters

in, out	<i>level_names</i>	the list of existing levels
in	<i>nb_lvl</i>	the number of existing levels

2.5.3.4 Level\* initLevel ( Level \* *lvl* )

Initialize a level. The width and the Height of the level must be stored in the level before calling this function.

## Parameters

out	<i>lvl</i>	The level
-----	------------	-----------

## Returns

a pointer on the level initialized

2.5.3.5 Level\* openLevel ( char \* *file\_name* )

Open a level file and store the level corresponding to the file

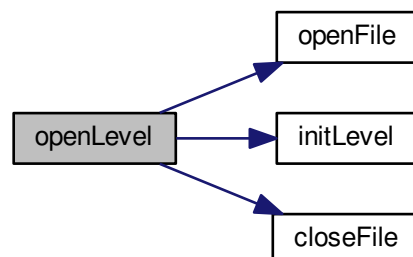
## Parameters

in	<i>file_name</i>	the name of the level file
----	------------------	----------------------------

## Returns

a pointer to the level created

Here is the call graph for this function:

2.5.3.6 char\*\* readLevelFile ( int \* *nb\_lvl* )

Read the file including the list of existing levels



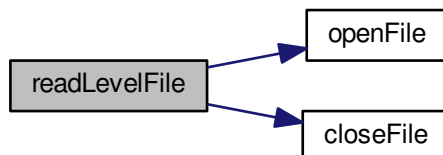
**Parameters**

out	<i>nb_lvl</i>	the number of level in the file
-----	---------------	---------------------------------

**Returns**

a pointer to an array of strings containing the list of the levels

Here is the call graph for this function:

**2.5.3.7 int searchEndLevel ( Level \* lvl )**

Search the end of a level

**Parameters**

in, out	<i>lvl</i>	the level
---------	------------	-----------

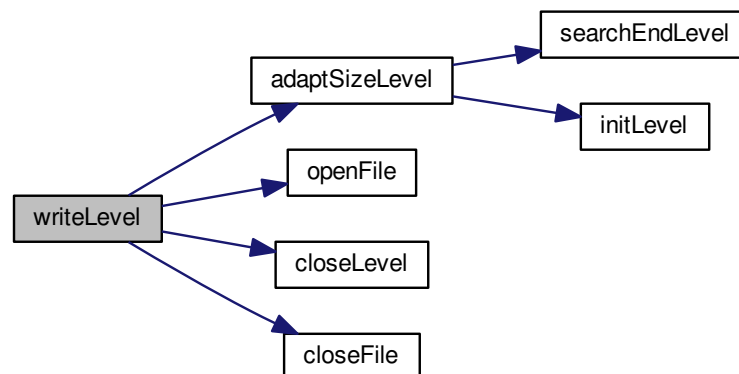
**2.5.3.8 void writeLevel ( char \* file\_name, Level \* lvl )**

Write the given level in the given file

**Parameters**

in	<i>lvl</i>	the file
in	<i>file_name</i>	the name of the file

Here is the call graph for this function:



## 2.6 game.c File Reference

Contain the main functions of the game.

```
#include "game.h"
```

### Functions

- void [play](#) (SDL\_Surface \*screen, char \*level\_name, SDLKey \*kc)
- void [printConfirmation](#) (SDL\_Surface \*screen, [Input](#) \*in, int \*go)

### 2.6.1 Detailed Description

Contain the main functions of the game.

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-05-15

### 2.6.2 Function Documentation

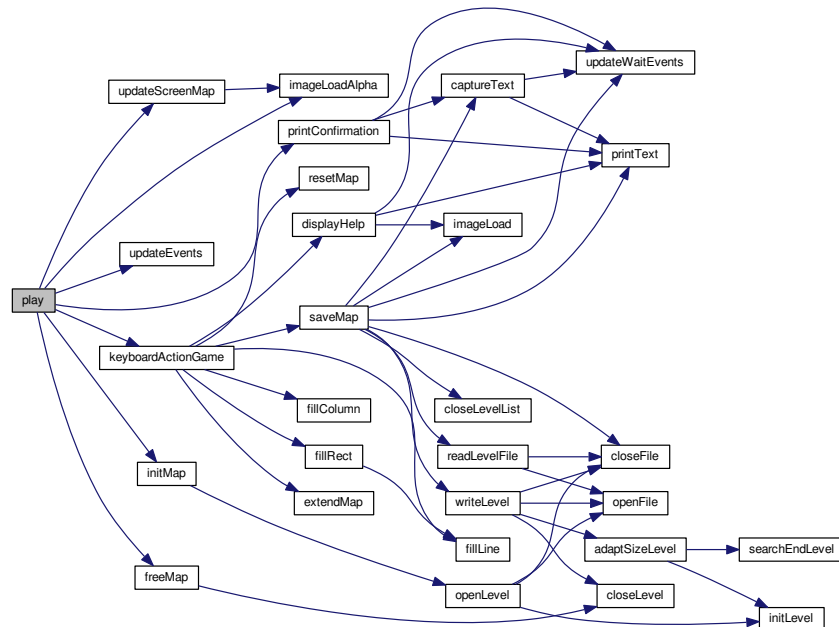
#### 2.6.2.1 void play ( SDL\_Surface \* screen, char \* level\_name, SDLKey \* kc )

Include the main loop of the game

**Parameters**

in, out	<i>screen</i>	The screen of the game
in	<i>level_name</i>	The name of the level
in	<i>kc</i>	The keyboard configuration

Here is the call graph for this function:



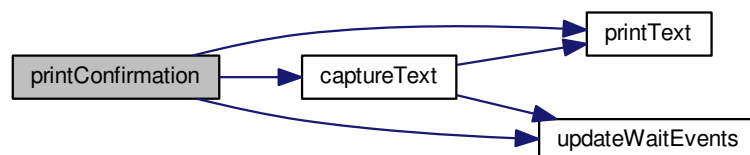
### 2.6.2.2 void printConfirmation ( SDL\_Surface \* *screen*, Input \* *in*, int \* *go* )

Display the confirmation screen, before leaving the edition of a level

**Parameters**

out	<i>screen</i>	The screen of the game
in	<i>in</i>	The input structure
out	<i>go</i>	The main loop activation

Here is the call graph for this function:



## 2.7 game.h File Reference

### game.c header

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include "const.h"
#include "structures.h"
#include "text.h"
#include "share.h"
#include "file_level.h"
#include "image.h"
#include "map.h"
#include "input.h"
```

### Functions

- void [play](#) (SDL\_Surface \*screen, char \*level\_name, SDLKey \*kc)
- void [printConfirmation](#) (SDL\_Surface \*screen, [Input](#) \*in, int \*go)

### 2.7.1 Detailed Description

#### game.c header

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-05-15

### 2.7.2 Function Documentation

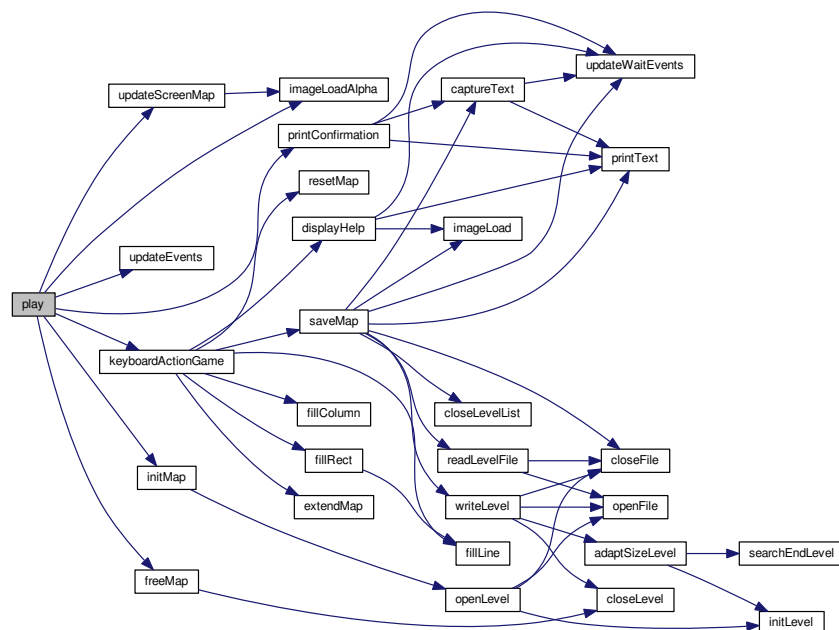
#### 2.7.2.1 void play ( SDL\_Surface \* screen, char \* level\_name, SDLKey \* kc )

Include the main loop of the game

#### Parameters

in, out	<i>screen</i>	The screen of the game
in	<i>level_name</i>	The name of the level
in	<i>kc</i>	The keyboard configuration

Here is the call graph for this function:



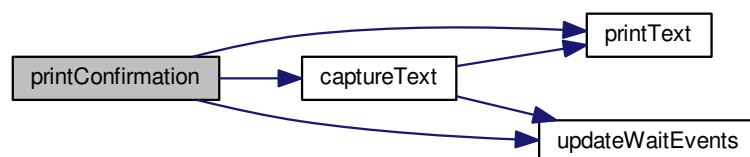
### 2.7.2.2 void printConfirmation ( SDL\_Surface \* screen, Input \* in, int \* go )

Display the confirmation screen, before leaving the edition of a level

#### Parameters

out	<i>screen</i>	The screen of the game
in	<i>in</i>	The input structure
out	<i>go</i>	The main loop activation

Here is the call graph for this function:



## 2.8 image.c File Reference

Contain the functions managing the images.

```
#include "image.h"
```

## Functions

- SDL\_Surface \* [imageLoad](#) (char \*file\_name)
- SDL\_Surface \* [imageLoadAlpha](#) (char \*file\_name)

### 2.8.1 Detailed Description

Contain the functions managing the images.

#### Author

Rémi BERTHO

#### Date

2014-02-27

### 2.8.2 Function Documentation

#### 2.8.2.1 SDL\_Surface \* imageLoad ( char \* file\_name )

Load an image

##### Parameters

in	<i>file_name</i>	the name of the image file
----	------------------	----------------------------

##### Returns

a pointer on the SDL\_Surface created

#### 2.8.2.2 SDL\_Surface \* imageLoadAlpha ( char \* file\_name )

Load an image with alpha management

##### Parameters

in	<i>file_name</i>	the name of the image file
----	------------------	----------------------------

##### Returns

a pointer on the SDL\_Surface created

## 2.9 image.h File Reference

contient les fonction liées aux images

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include "const.h"
```

## Functions

- SDL\_Surface \* [imageLoad](#) (char \*file\_name)
- SDL\_Surface \* [imageLoadAlpha](#) (char \*file\_name)

### 2.9.1 Detailed Description

contient les fonction liées aux images

#### Author

Rémi BERTHO

#### Date

2014-02-27

### 2.9.2 Function Documentation

#### 2.9.2.1 SDL\_Surface\* imageLoad ( char \* file\_name )

Load an image

##### Parameters

in	<i>file_name</i>	the name of the image file
----	------------------	----------------------------

##### Returns

a pointer on the SDL\_Surface created

#### 2.9.2.2 SDL\_Surface\* imageLoadAlpha ( char \* file\_name )

Load an image with alpha management

##### Parameters

in	<i>file_name</i>	the name of the image file
----	------------------	----------------------------

##### Returns

a pointer on the SDL\_Surface created

## 2.10 input.c File Reference

```
#include "input.h"
```

#### Functions

- void [updateEvents](#) (Input \*in)
- void [keyboardActionGame](#) (SDL\_Surface \*screen, Input \*in, Map \*m, Cursor \*cursor, SDLKey \*kc)
- int [updateWaitEvents](#) (Input \*in)
- int [keyboardActionMenu](#) (Input \*in, int \*cursorPos, int \*select, int nb\_options)

### 2.10.1 Detailed Description

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-03-18

## 2.10.2 Function Documentation

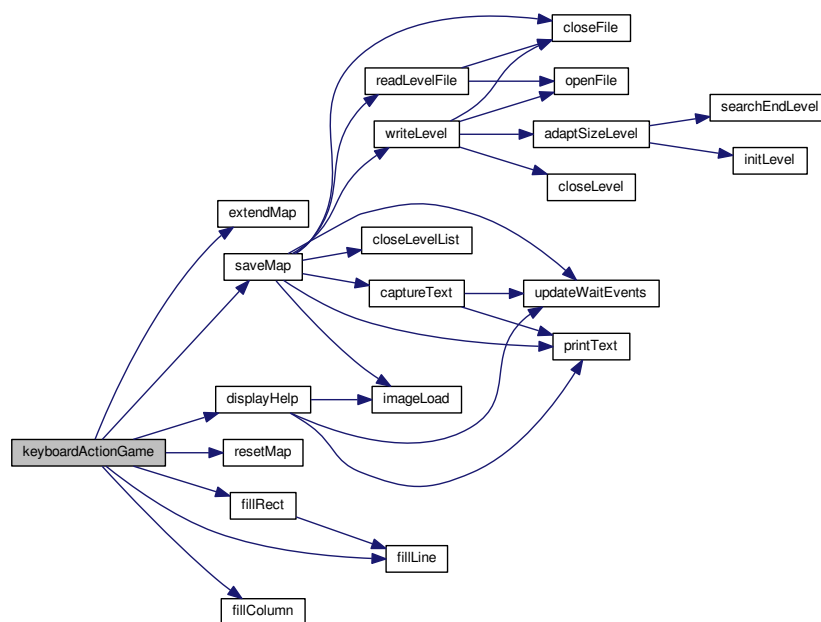
2.10.2.1 void keyboardActionGame ( SDL\_Surface \* *screen*, Input \* *in*, Map \* *m*, Cursor \* *cursor*, SDLKey \* *kc* )

perform action commanded by keyboard action

## Parameters

in, out	<i>screen</i>	The screen of the game
in, out	<i>in</i>	the input structure
in, out	<i>m</i>	the map to update
in, out	<i>cursor</i>	the cursor structure
in	<i>kc</i>	the keyboard bindings

Here is the call graph for this function:

2.10.2.2 void keyboardActionMenu ( Input \* *in*, int \* *cursorPos*, int \* *select*, int *nb\_options* )

perform menu action commanded by keyboard action

## Parameters

in	<i>in</i>	the input structure
out	<i>cursorPos</i>	the cursor position
out	<i>select</i>	boolean about selecting the option or quit to title screen
in	<i>nb_options</i>	the number of options of the menu

2.10.2.3 void updateEvents ( Input \* *in* )

get keyboard input with a SDL\_PollEvent



**Parameters**

out	in	the input structure
-----	----	---------------------

**2.10.2.4 int updateWaitEvents ( Input \* in )**

get keyboard input with a SDL\_WaitEvent

**Parameters**

out	in	the input structure
-----	----	---------------------

**Returns**

1 if a key is activated

**2.11 main.c File Reference**

```
#include "game.h"
#include "const.h"
#include "menu.h"
#include "menu_level.h"
#include "menu_option.h"
#include "option.h"
```

**Functions**

- int [main](#) (int argc, char \*argv[])

**2.11.1 Detailed Description****Author**

Xavier COPONET

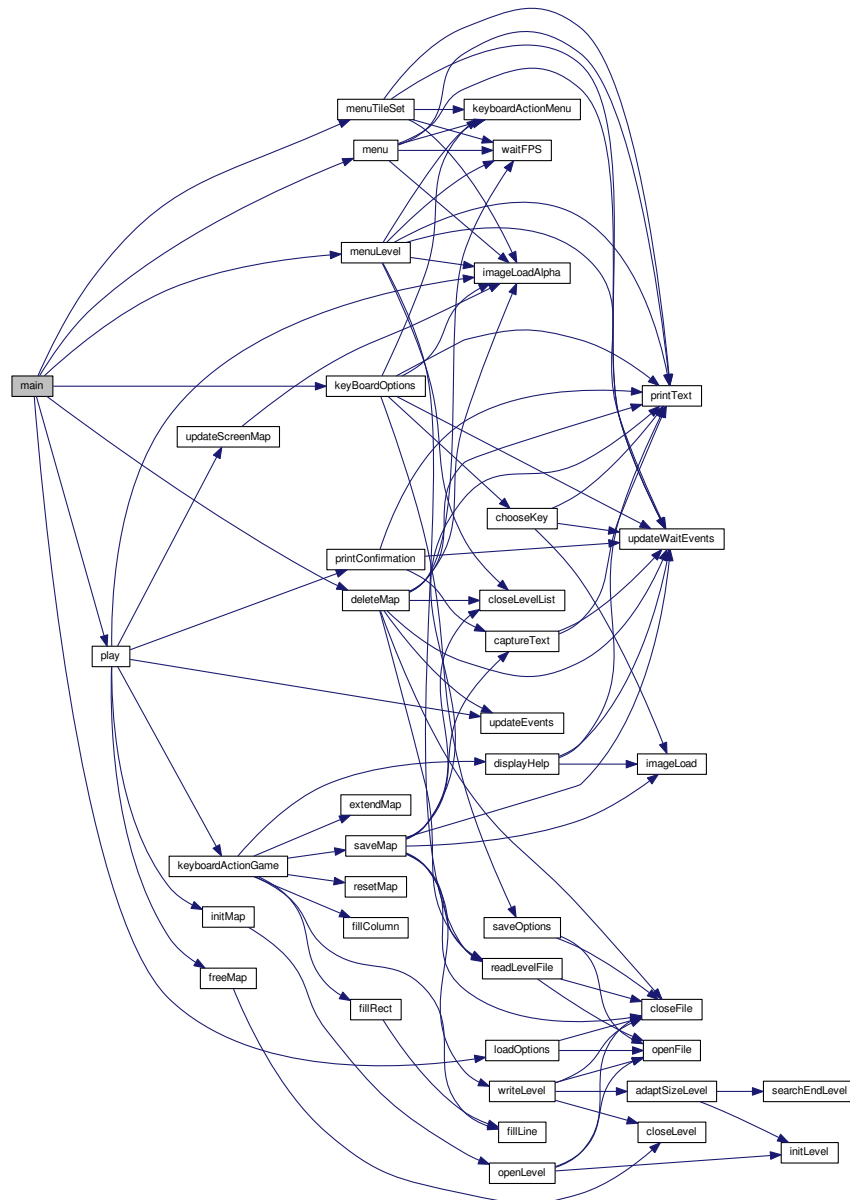
**Date**

2014-02-27

**2.11.2 Function Documentation****2.11.2.1 int main ( int argc, char \* argv[] )****Main****Parameters**

in, out	argc	argc
in, out	argv	argv

Here is the call graph for this function:



## 2.12 map.c File Reference

Management of the map.

```
#include "map.h"
```

### Functions

- void [updateScreenMap](#) (SDL\_Surface \*screen, [Map](#) \*m, char \*tileset, [Cursor](#) \*cursor)
- [Map](#) \* [initMap](#) (SDL\_Surface \*screen, char \*level\_name)
- void [fillLine](#) ([Map](#) \*m, int line, int column, char tileID)
- void [fillColumn](#) ([Map](#) \*m, int line, int column, char tileID)

- void `fillRect` (`Map *m`, int line, int column, char tileID)
- void `displayHelp` (`SDL_Surface *screen`, `SDLKey *kc`)
- void `saveMap` (`SDL_Surface *screen`, `Map *m`)
- void `deleteMap` (`SDL_Surface *screen`, char `*map_name`, char `*map_path`)
- void `extendMap` (`Map *m`)
- void `resetMap` (`Map *m`)
- void `freeMap` (`Map *m`)

### 2.12.1 Detailed Description

Management of the map.

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-05-18

### 2.12.2 Function Documentation

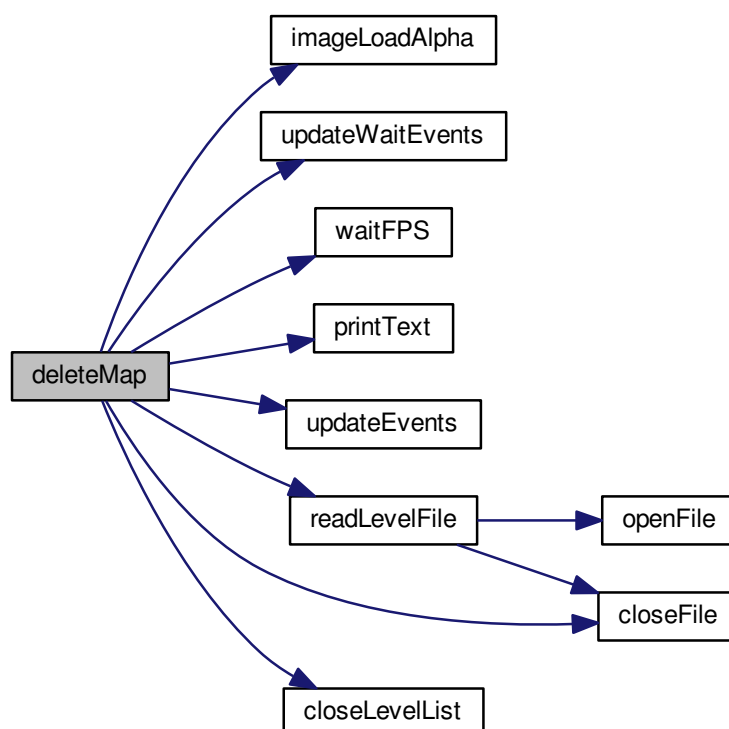
#### 2.12.2.1 void `deleteMap` ( `SDL_Surface * screen`, char `* map_name`, char `* map_path` )

Delete a the map file and update the level list file

#### Parameters

in, out	<i>screen</i>	The screen of the game
in	<i>map_name</i>	The name of the map to delete
in	<i>map_path</i>	The path to the file of the map to delete

Here is the call graph for this function:



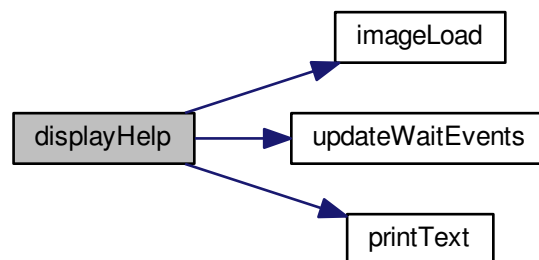
#### 2.12.2.2 void displayHelp ( SDL\_Surface \* screen, SDLKey \* kc )

Display the list of keybindings on the screen

##### Parameters

<code>in, out</code>	<code>screen</code>	The screen of the game
<code>in</code>	<code>kc</code>	The array containing the keybindings

Here is the call graph for this function:



#### 2.12.2.3 void extendMap ( Map \* m )

Extend the width of a map

Parameters

out	<i>m</i>	The map to extend
-----	----------	-------------------

#### 2.12.2.4 void fillColumn ( Map \* m, int line, int column, char tileID )

Fill a column with the current tile. The filling stops if another non-void tile is reached. This function works only with ground tiles

Parameters

in	<i>m</i>	The map
in	<i>line</i>	The line pointed by the cursor
in	<i>column</i>	The column pointed by the cursor
in	<i>tileID</i>	the tile ID

#### 2.12.2.5 void fillLine ( Map \* m, int line, int column, char tileID )

Fill a line with the current tile. The filling stops if another non-void tile is reached. This function works only with ground tiles

Parameters

in	<i>m</i>	The map
in	<i>line</i>	The line pointed by the cursor
in	<i>column</i>	The column pointed by the cursor
in	<i>tileID</i>	the tile ID

#### 2.12.2.6 void fillRect ( Map \* m, int line, int column, char tileID )

Fill a rectangle with the current tile. The filling stops if another non-void tile is reached. This function works only with ground tiles

## Parameters

in	<i>m</i>	The map
in	<i>line</i>	The line pointed by the cursor
in	<i>column</i>	The column pointed by the cursor
in	<i>tileID</i>	the tile ID

Here is the call graph for this function:



### 2.12.2.7 void freeMap ( Map \* m )

Free memory allocated to the map

## Parameters

in, out	<i>m</i>	the map
---------	----------	---------

Here is the call graph for this function:



### 2.12.2.8 Map \* initMap ( SDL\_Surface \* screen, char \* level\_name )

initialize the map

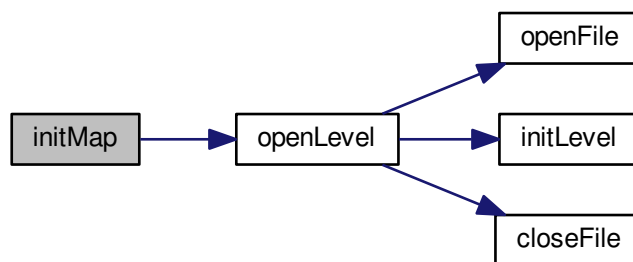
## Parameters

in	<i>screen</i>	game screen
in	<i>level_name</i>	lvl name

**Returns**

a pointer on the map

Here is the call graph for this function:



#### 2.12.2.9 void resetMap ( Map \* m )

Fill a map with blank tiles. This function doesn't change the current map file.

**Parameters**

in, out	<i>m</i>	The map to reinit
---------	----------	-------------------

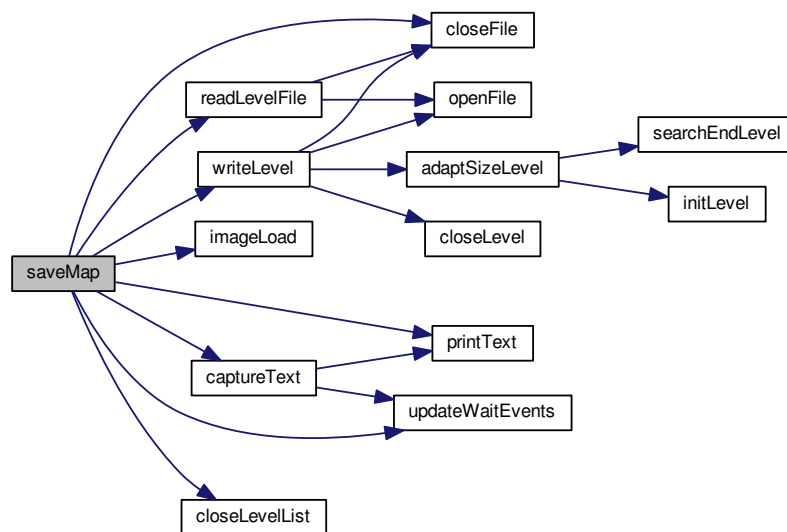
#### 2.12.2.10 void saveMap ( SDL\_Surface \* screen, Map \* m )

Save the map in a new file and update the file 'level' containing the map list

**Parameters**

in, out	<i>screen</i>	The screen of the game
in	<i>m</i>	The map to save

Here is the call graph for this function:



#### 2.12.2.11 void updateScreenMap ( SDL\_Surface \* screen, Map \* m, char \* tileset, Cursor \* cursor )

update and display the map on the screen

##### Parameters

in, out	<i>screen</i>	The screen of the game
in	<i>m</i>	The map
in	<i>tileset</i>	The level tileset
in	<i>cursor</i>	The mouse cursor

Here is the call graph for this function:



## 2.13 map.h File Reference

[map.c](#) header



```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include "image.h"
#include "file_level.h"
#include "input.h"
#include "text.h"
#include "share.h"
```

## Functions

- void [updateScreenMap](#) (SDL\_Surface \*screen, [Map](#) \*m, char \*tileset, [Cursor](#) \*cursor)
- [Map](#) \* [initMap](#) (SDL\_Surface \*screen, char \*level\_name)
- void [fillLine](#) ([Map](#) \*m, int line, int column, char tileID)
- void [fillColumn](#) ([Map](#) \*m, int line, int column, char tileID)
- void [fillRect](#) ([Map](#) \*m, int line, int column, char tileID)
- void [displayHelp](#) (SDL\_Surface \*screen, SDLKey \*kc)
- void [saveMap](#) (SDL\_Surface \*screen, [Map](#) \*m)
- void [deleteMap](#) (SDL\_Surface \*screen, char \*map\_name, char \*map\_path)
- void [extendMap](#) ([Map](#) \*m)
- void [resetMap](#) ([Map](#) \*m)
- void [freeMap](#) ([Map](#) \*m)

### 2.13.1 Detailed Description

[map.c](#) header

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-05-18

### 2.13.2 Function Documentation

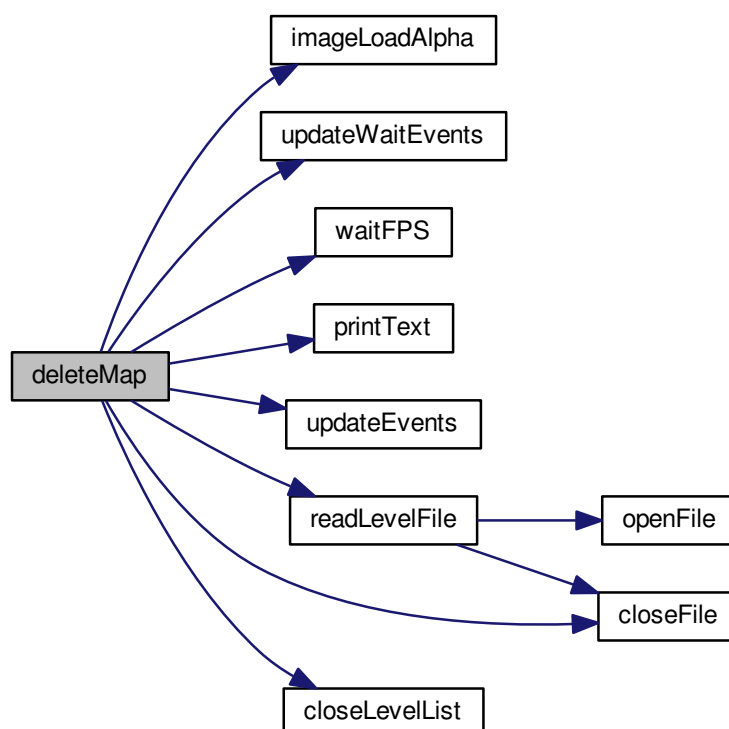
#### 2.13.2.1 void [deleteMap](#) ( SDL\_Surface \* *screen*, char \* *map\_name*, char \* *map\_path* )

Delete a the map file and update the level list file

#### Parameters

<i>in, out</i>	<i>screen</i>	The screen of the game
<i>in</i>	<i>map_name</i>	The name of the map to delete
<i>in</i>	<i>map_path</i>	The path to the file of the map to delete

Here is the call graph for this function:



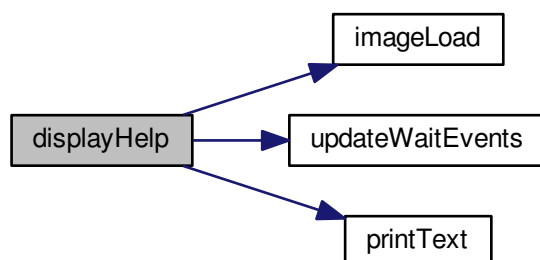
#### 2.13.2.2 void displayHelp ( SDL\_Surface \* screen, SDLKey \* kc )

Display the list of keybindings on the screen

##### Parameters

<code>in, out</code>	<code>screen</code>	The screen of the game
<code>in</code>	<code>kc</code>	The array containing the keybindings

Here is the call graph for this function:



#### 2.13.2.3 void extendMap ( Map \* m )

Extend the width of a map

##### Parameters

out	<i>m</i>	The map to extend
-----	----------	-------------------

#### 2.13.2.4 void fillColumn ( Map \* m, int line, int column, char tileID )

Fill a column with the current tile. The filling stops if another non-void tile is reached. This function works only with ground tiles

##### Parameters

in	<i>m</i>	The map
in	<i>line</i>	The line pointed by the cursor
in	<i>column</i>	The column pointed by the cursor
in	<i>tileID</i>	the tile ID

#### 2.13.2.5 void fillLine ( Map \* m, int line, int column, char tileID )

Fill a line with the current tile. The filling stops if another non-void tile is reached. This function works only with ground tiles

##### Parameters

in	<i>m</i>	The map
in	<i>line</i>	The line pointed by the cursor
in	<i>column</i>	The column pointed by the cursor
in	<i>tileID</i>	the tile ID

#### 2.13.2.6 void fillRect ( Map \* m, int line, int column, char tileID )

Fill a rectangle with the current tile. The filling stops if another non-void tile is reached. This function works only with ground tiles

## Parameters

in	<i>m</i>	The map
in	<i>line</i>	The line pointed by the cursor
in	<i>column</i>	The column pointed by the cursor
in	<i>tileID</i>	the tile ID

Here is the call graph for this function:



## 2.13.2.7 void freeMap ( Map \* m )

Free memory allocated to the map

## Parameters

in, out	<i>m</i>	the map
---------	----------	---------

Here is the call graph for this function:



## 2.13.2.8 Map\* initMap ( SDL\_Surface \* screen, char \* level\_name )

initialize the map

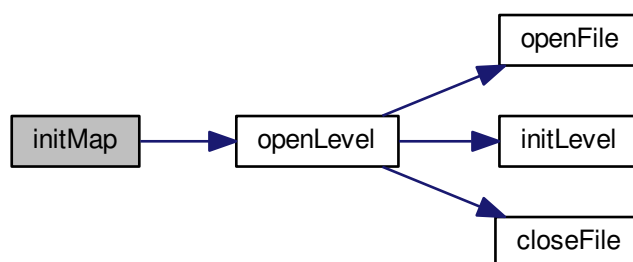
## Parameters

in	<i>screen</i>	game screen
in	<i>level_name</i>	lvl name

**Returns**

a pointer on the map

Here is the call graph for this function:



#### 2.13.2.9 void resetMap ( Map \* m )

Fill a map with blank tiles. This function doesn't change the current map file.

**Parameters**

in, out	<i>m</i>	The map to reinit
---------	----------	-------------------

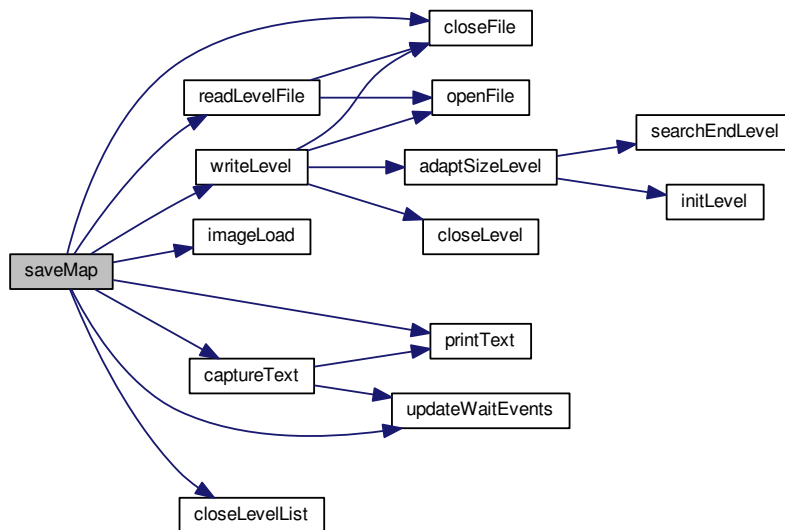
#### 2.13.2.10 void saveMap ( SDL\_Surface \* screen, Map \* m )

Save the map in a new file and update the file 'level' containing the map list

**Parameters**

in, out	<i>screen</i>	The screen of the game
in	<i>m</i>	The map to save

Here is the call graph for this function:



#### 2.13.2.11 void updateScreenMap ( SDL\_Surface \* screen, Map \* m, char \* tileset, Cursor \* cursor )

update and display the map on the screen

##### Parameters

in, out	<i>screen</i>	The screen of the game
in	<i>m</i>	The map
in	<i>tileset</i>	The level tileset
in	<i>cursor</i>	The mouse cursor

Here is the call graph for this function:



## 2.14 menu.c File Reference

Contain the main menu management.

```
#include "menu.h"
```

##### Functions

- int [menu](#) (SDL\_Surface \*screen, int \*choice, int \*go)

- int `menuTileSet` (SDL\_Surface \*screen, char tileSet\_name[MAX\_LENGTH\_FILE\_NAME])

#### 2.14.1 Detailed Description

Contain the main menu management.

Author

Glenn HERROU

Date

2014-04-20

#### 2.14.2 Function Documentation

##### 2.14.2.1 int menu ( SDL\_Surface \* screen, int \* choice, int \* go )

Display the menu on the screen

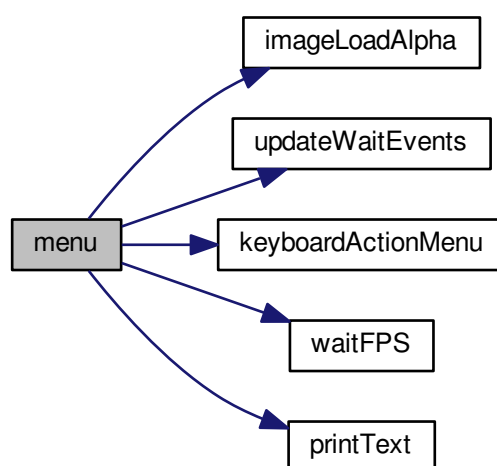
Parameters

out	<i>screen</i>	the screen of the game
out	<i>choice</i>	the option selected
out	<i>go</i>	the main loop validation

Returns

1 if an option has been selected

Here is the call graph for this function:



##### 2.14.2.2 int menuTileSet ( SDL\_Surface \* screen, char tileSet\_name[MAX\_LENGTH\_FILE\_NAME] )

Display the tileset menu on the screen

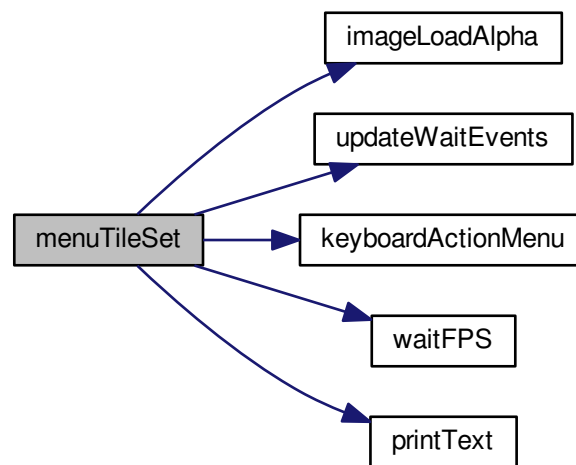
## Parameters

out	<i>screen</i>	the screen of the game
out	<i>tileSet_name</i>	The name of the tileSet selected

## Returns

1 if a tileset has been selected

Here is the call graph for this function:



## 2.15 menu.h File Reference

header de [menu.c](#)

```

#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include <SDL/SDL_ttf.h>
#include "const.h"
#include "text.h"
#include "share.h"
#include "image.h"
#include "input.h"

```

## Functions

- int [menu](#) (SDL\_Surface \*screen, int \*choice, int \*go)
- int [menuTileSet](#) (SDL\_Surface \*screen, char tileSet\_name[MAX\_LENGTH\_FILE\_NAME])



### 2.15.1 Detailed Description

header de [menu.c](#)

#### Author

Xavier COPONET

#### Date

2014-02-27

### 2.15.2 Function Documentation

#### 2.15.2.1 `int menu ( SDL_Surface * screen, int * choice, int * go )`

Display the menu on the screen

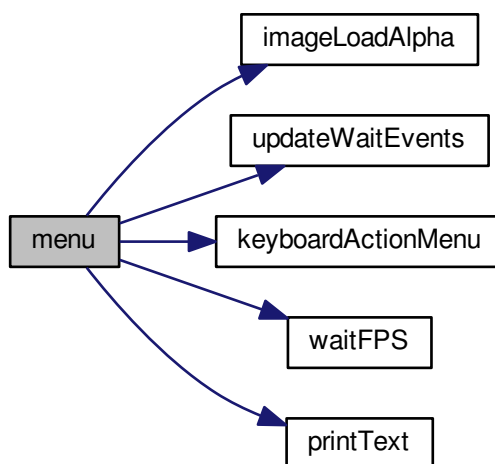
#### Parameters

out	<i>screen</i>	the screen of the game
out	<i>choice</i>	the option selected
out	<i>go</i>	the main loop validation

#### Returns

1 if an option has been selected

Here is the call graph for this function:



#### 2.15.2.2 `int menuTileSet ( SDL_Surface * screen, char tileSet_name[MAX_LENGTH_FILE_NAME] )`

Display the tileset menu on the screen

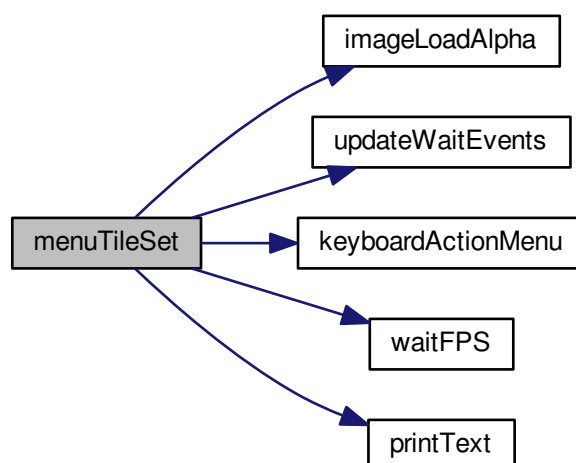
## Parameters

out	<i>screen</i>	the screen of the game
out	<i>tileSet_name</i>	The name of the tileSet selected

## Returns

1 if a tileset has been selected

Here is the call graph for this function:



## 2.16 menu\_level.c File Reference

Contain the level menu management.

```
#include "menu_level.h"
```

## Functions

- int [menuLevel](#) (SDL\_Surface \*screen, char level\_name[[MAX\\_LENGTH\\_FILE\\_NAME](#)], char level\_path[[MAX\\_LENGTH\\_FILE\\_NAME](#)])

## 2.16.1 Detailed Description

Contain the level menu management.

## Author

Remi BERTHO, Glenn HERROU

## Date

2014-05-10

Version

2.0

## 2.16.2 Function Documentation

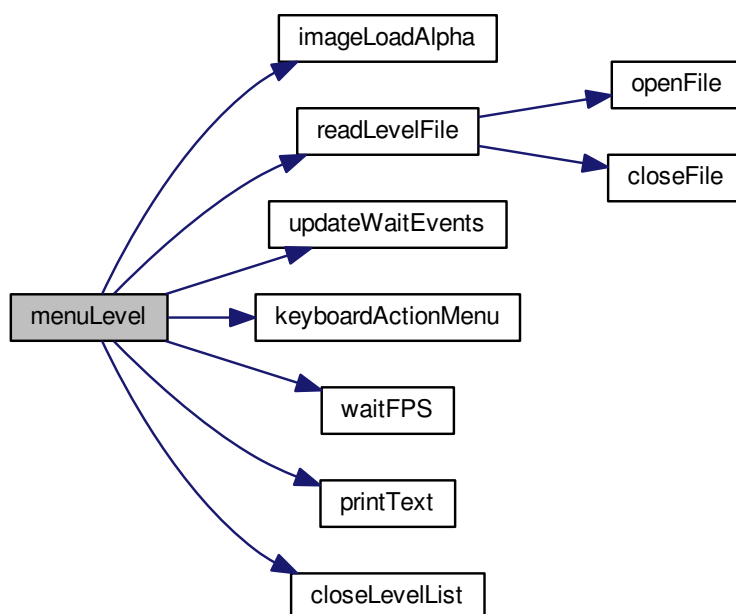
2.16.2.1 `int menuLevel ( SDL_Surface * screen, char level_name[MAX_LENGTH_FILE_NAME], char level_path[MAX_LENGTH_FILE_NAME] )`

Display the level menu on the screen

Parameters

out	<i>screen</i>	The screen of the game
out	<i>level_name</i>	The level name
out	<i>level_path</i>	The level file path

Here is the call graph for this function:



## 2.17 menu\_level.h File Reference

[menu\\_level.c](#) header

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include <SDL/SDL_ttf.h>
#include "const.h"
#include "file_level.h"
#include "share.h"
#include "text.h"
#include "image.h"
#include "input.h"
```

## Functions

- int [menuLevel](#) (SDL\_Surface \*screen, char level\_name[MAX\_LENGTH\_FILE\_NAME], char level\_path[MAX\_LENGTH\_FILE\_NAME])

### 2.17.1 Detailed Description

[menu\\_level.c](#) header

#### Author

Remi BERTHO, Glenn HERROU

#### Date

2014-05-10

#### Version

2.0

### 2.17.2 Function Documentation

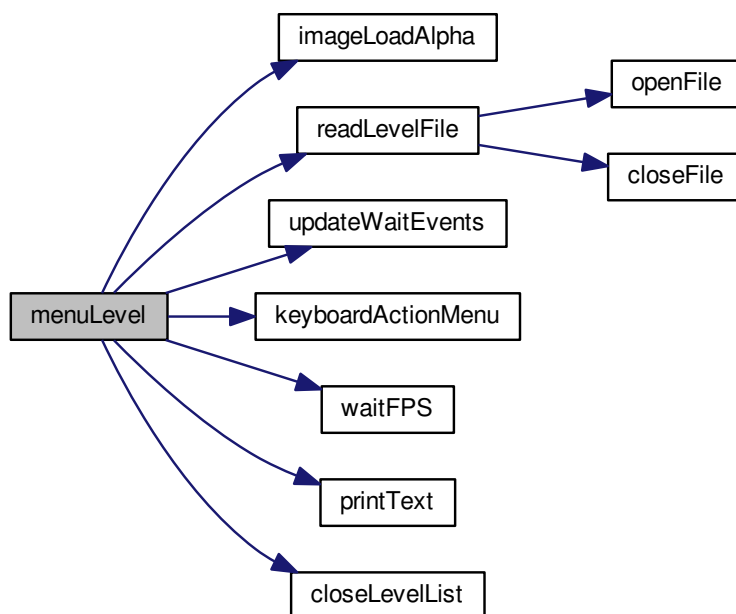
**2.17.2.1** int [menuLevel](#) ( SDL\_Surface \* *screen*, char *level\_name*[MAX\_LENGTH\_FILE\_NAME], char *level\_path*[MAX\_LENGTH\_FILE\_NAME] )

Display the level menu on the screen

#### Parameters

out	<i>screen</i>	The screen of the game
out	<i>level_name</i>	The level name
out	<i>level_path</i>	The level file path

Here is the call graph for this function:



## 2.18 menu\_option.c File Reference

contain the option menu functions

```
#include "menu_option.h"
```

### Functions

- int [menuOptions](#) (SDL\_Surface \*screen, int \*go, SDLKey \*kc)
- void [keyBoardOptions](#) (SDL\_Surface \*screen, int \*go, SDLKey \*kc)
- void [chooseKey](#) (SDL\_Surface \*screen, Input \*in, char \*action, SDLKey \*kc, int nb)

### 2.18.1 Detailed Description

contain the option menu functions

#### Author

Xavier COPONET

#### Date

2014-04-27

## 2.18.2 Function Documentation

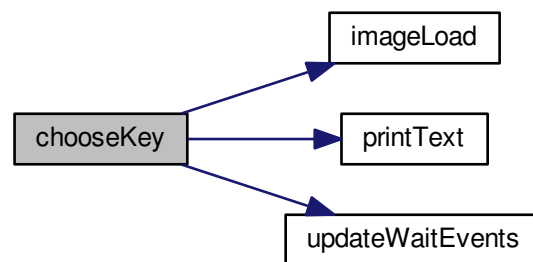
### 2.18.2.1 void chooseKey ( SDL\_Surface \* *screen*, Input \* *in*, char \* *action*, SDLKey \* *kc*, int *nb* )

print the message asking the player to choose a key and wait until the player press a key and deals with this key

**Parameters**

out	<i>screen</i>	the game screen
in, out	<i>in</i>	the input structure
in	<i>action</i>	the action which the key has to be choosen
out	<i>kc</i>	the keyboard configuration
in	<i>nb</i>	the number of the action

Here is the call graph for this function:



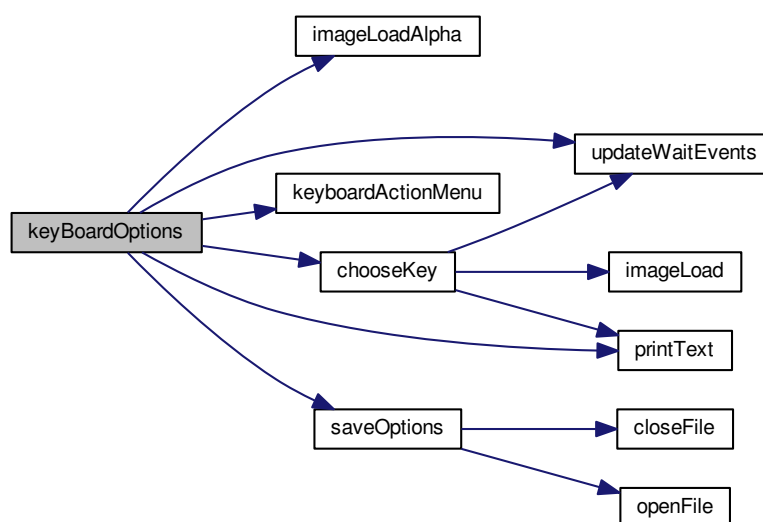
### 2.18.2.2 void keyBoardOptions ( SDL\_Surface \* *screen*, int \* *go*, SDLKey \* *kc* )

print the keyboard options and deals with the user choises

**Parameters**

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
in, out	<i>kc</i>	the keyboard config

Here is the call graph for this function:



### 2.18.2.3 int menuOptions ( SDL\_Surface \* *screen*, int \* *go*, SDLKey \* *kc* )

print the option menu on the screen

#### Parameters

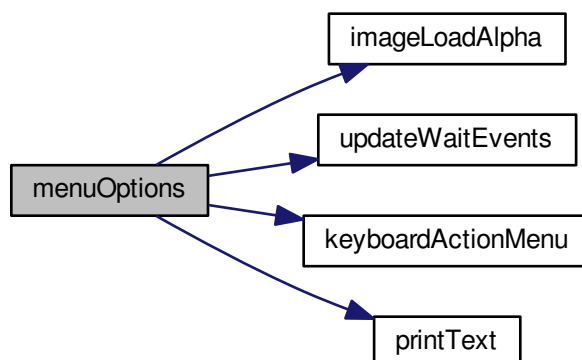
out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
in, out	<i>kc</i>	the keyboard configuration

#### Returns

the number of the option which is choosen, -1 if esc



Here is the call graph for this function:



## 2.19 menu\_option.h File Reference

[menu\\_option.c](#) header

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include <SDL/SDL_ttf.h>
#include "const.h"
#include "text.h"
#include "share.h"
#include "image.h"
#include "input.h"
#include "option.h"
```

### Functions

- int [menuOptions](#) (SDL\_Surface \*screen, int \*go, SDLKey \*kc)
- void [keyBoardOptions](#) (SDL\_Surface \*screen, int \*go, SDLKey \*kc)
- void [chooseKey](#) (SDL\_Surface \*screen, [Input](#) \*in, char \*action, SDLKey \*kc, int nb)

#### 2.19.1 Detailed Description

[menu\\_option.c](#) header

Author

Xavier COPONET

Date

2014-04-27

## 2.19.2 Function Documentation

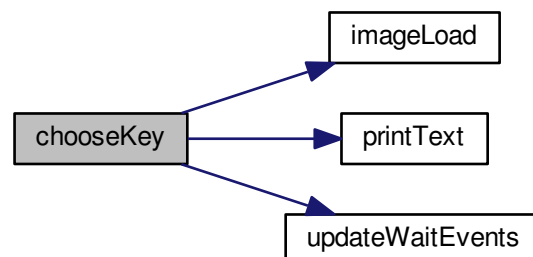
## 2.19.2.1 void chooseKey ( SDL\_Surface \* screen, Input \* in, char \* action, SDLKey \* kc, int nb )

print the message asking the player to choose a key and wait until the player press a key and deals with this key

## Parameters

out	<i>screen</i>	the game screen
in, out	<i>in</i>	the input structure
in	<i>action</i>	the action which the key has to be choosen
out	<i>kc</i>	the keyboard configuration
in	<i>nb</i>	the number of the action

Here is the call graph for this function:



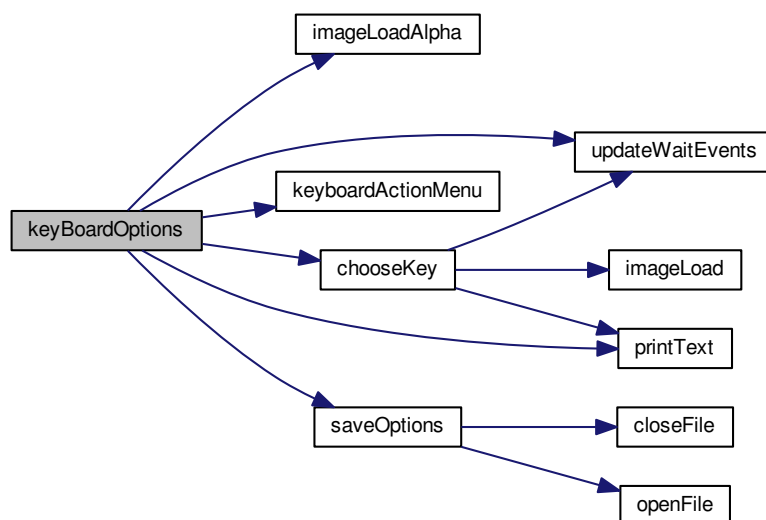
## 2.19.2.2 void keyBoardOptions ( SDL\_Surface \* screen, int \* go, SDLKey \* kc )

print the keyboard options and deals with the user choises

## Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
in, out	<i>kc</i>	the keyboard config

Here is the call graph for this function:



### 2.19.2.3 int menuOptions ( SDL\_Surface \* *screen*, int \* *go*, SDLKey \* *kc* )

print the option menu on the screen

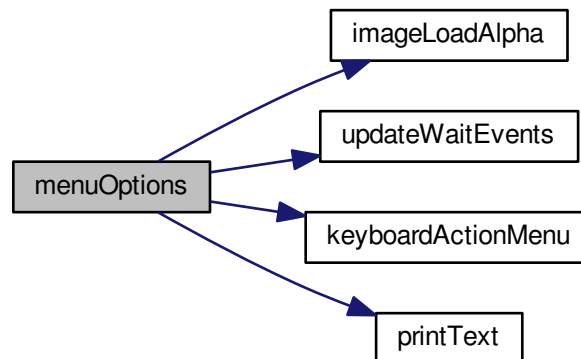
#### Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
in, out	<i>kc</i>	the keyboard configuration

#### Returns

the number of the option which is choosen, -1 if esc

Here is the call graph for this function:



## 2.20 option.c File Reference

contains the functions that manipulate the options

```
#include "option.h"
```

### Functions

- void [loadOptions](#) (char confFile[], SDLKey \*kc)
- void [saveOptions](#) (char confFile[], SDLKey \*kc)

### 2.20.1 Detailed Description

contains the functions that manipulate the options

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-04-28

### 2.20.2 Function Documentation

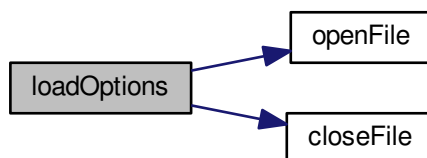
#### 2.20.2.1 void loadOptions ( char *confFile*[], SDLKey \* *kc* )

Load the options from the config file

**Parameters**

in	<i>confFile</i>	the config file path
out	<i>kc</i>	the keyboard configuration structure

Here is the call graph for this function:



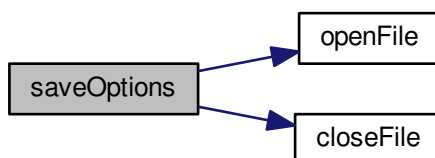
### 2.20.2.2 void saveOptions ( char *confFile*[], SDLKey \* *kc* )

save the options to the config file

**Parameters**

in	<i>confFile</i>	the config file path
in	<i>kc</i>	the keyboard configuration structure

Here is the call graph for this function:



## 2.21 option.h File Reference

### [option.c](#) header

```

#include "file.h"
#include "structures.h"
#include "input.h"

```

**Functions**

- void [loadOptions](#) (char *confFile*[], SDLKey \**kc*)
- void [saveOptions](#) (char *confFile*[], SDLKey \**kc*)

## 2.21.1 Detailed Description

[option.c](#) header

## Author

Xavier COPONET

## Date

2014-04-28

## 2.21.2 Function Documentation

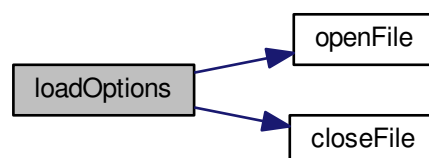
2.21.2.1 void loadOptions ( char *confFile*[], SDLKey \* *kc* )

Load the options from the config file

## Parameters

in	<i>confFile</i>	the config file path
out	<i>kc</i>	the keyboard configuration structure

Here is the call graph for this function:

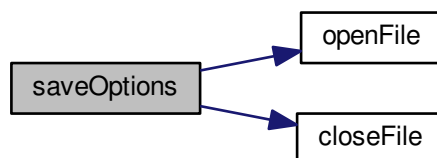
2.21.2.2 void saveOptions ( char *confFile*[], SDLKey \* *kc* )

save the options to the config file

## Parameters

in	<i>confFile</i>	the config file path
in	<i>kc</i>	the keyboard configuration structure

Here is the call graph for this function:



## 2.22 share.c File Reference

Management of FPS rate.

```
#include "share.h"
```

### Functions

- void `waitFPS` (int \*previous\_time, int \*current\_time)

### 2.22.1 Detailed Description

Management of FPS rate.

#### Author

Remi BERTHO

#### Date

15/03/14

#### Version

1.0

### 2.22.2 Function Documentation

#### 2.22.2.1 void waitFPS ( int \* previous\_time, int \* current\_time )

Function managing the fps rate

#### Parameters

in, out	<i>previous_time</i>	The previous time
in, out	<i>current_time</i>	The current time

## 2.23 share.h File Reference

### share.c header

```
#include "const.h"  
#include <SDL/SDL.h>
```

### Functions

- void [waitFPS](#) (int \*previous\_time, int \*current\_time)

#### 2.23.1 Detailed Description

### share.c header

#### Author

Remi BERTHO

#### Date

15/03/14

#### Version

1.0

#### 2.23.2 Function Documentation

##### 2.23.2.1 void waitFPS ( int \* *previous\_time*, int \* *current\_time* )

Function managing the fps rate

#### Parameters

in, out	<i>previous_time</i>	The previous time
in, out	<i>current_time</i>	The current time

## 2.24 text.c File Reference

Management of the display of text on the screen.

```
#include "text.h"
```

### Functions

- void [printText](#) (SDL\_Surface \*screen, SDL\_Rect \*posText, char \*text, int r, int g, int b, char \*font, int ptSize, int mode)
- void [captureText](#) (SDL\_Surface \*screen, SDL\_Rect posText, char \*text, int text\_length, int r, int g, int b, char \*font, int text\_size, int \*go)



### 2.24.1 Detailed Description

Management of the display of text on the screen.

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-04-27

### 2.24.2 Function Documentation

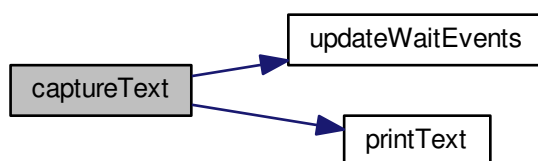
**2.24.2.1** `void captureText ( SDL_Surface * screen, SDL_Rect posText, char * text, int text_length, int r, int g, int b, char * font, int text_size, int * go )`

Capture the text corresponding to the keyboard inputs and display it on the screen at the given position

#### Parameters

out	<i>screen</i>	The screen of the game
in	<i>posText</i>	The position of the text. If NULL, the text is centered
out	<i>text</i>	The text to display
in	<i>r</i>	red value
in	<i>g</i>	green value
in	<i>b</i>	blue value
in	<i>text_length</i>	the text length
in	<i>font</i>	The path to the font file
in	<i>text_size</i>	The text size
out	<i>go</i>	The main loop validation

Here is the call graph for this function:



**2.24.2.2** `void printText ( SDL_Surface * screen, SDL_Rect * posText, char * text, int r, int g, int b, char * font, int ptSize, int mode )`

Display the given text on the screen, at the given position

#### Parameters

out	<i>screen</i>	The screen of the game
in	<i>posText</i>	The position of the text. If NULL, the text is centered
in	<i>text</i>	The text to display
in	<i>r</i>	red value
in	<i>g</i>	green value
in	<i>b</i>	blue value
in	<i>font</i>	The path to the font file
in	<i>ptSize</i>	The text size
in	<i>mode</i>	The writing mode : 0 (Solid), 1 (Blended)

## 2.25 text.h File Reference

### text.c header

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include "structures.h"
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include <SDL/SDL_ttf.h>
#include "input.h"
```

### Functions

- void [printText](#) (SDL\_Surface \*screen, SDL\_Rect \*posText, char \*text, int r, int g, int b, char \*font, int ptSize, int mode)
- void [captureText](#) (SDL\_Surface \*screen, SDL\_Rect posText, char \*text, int text\_length, int r, int g, int b, char \*font, int text\_size, int \*go)

#### 2.25.1 Detailed Description

### text.c header

#### Author

Xavier COPONET, Glenn HERROU

#### Date

2014-04-27

#### 2.25.2 Function Documentation

**2.25.2.1 void captureText ( SDL\_Surface \* *screen*, SDL\_Rect *posText*, char \* *text*, int *text\_length*, int *r*, int *g*, int *b*, char \* *font*, int *text\_size*, int \* *go* )**

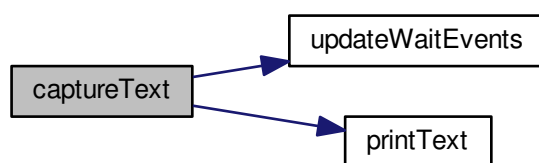
Capture the text corresponding to the keyboard inputs and display it on the screen at the given position

#### Parameters

---

out	<i>screen</i>	The screen of the game
in	<i>posText</i>	The position of the text. If NULL, the text is centered
out	<i>text</i>	The text to display
in	<i>r</i>	red value
in	<i>g</i>	green value
in	<i>b</i>	blue value
in	<i>text_length</i>	the text length
in	<i>font</i>	The path to the font file
in	<i>text_size</i>	The text size
out	<i>go</i>	The main loop validation

Here is the call graph for this function:



**2.25.2.2** void printText ( SDL\_Surface \* *screen*, SDL\_Rect \* *posText*, char \* *text*, int *r*, int *g*, int *b*, char \* *font*, int *ptSize*, int *mode* )

Display the given text on the screen, at the given position

#### Parameters

out	<i>screen</i>	The screen of the game
in	<i>posText</i>	The position of the text. If NULL, the text is centered
in	<i>text</i>	The text to display
in	<i>r</i>	red value
in	<i>g</i>	green value
in	<i>b</i>	blue value
in	<i>font</i>	The path to the font file
in	<i>ptSize</i>	The text size
in	<i>mode</i>	The writing mode : 0 (Solid), 1 (Blended)

## Index

- adaptSizeLevel
  - file\_level.c, [8](#)
  - file\_level.h, [12](#)
- BUFFER\_SIZE
  - file\_level.h, [12](#)
- background
  - Level, [3](#)
- captureText
  - text.c, [54](#)
  - text.h, [55](#)
- chooseKey
  - menu\_option.c, [43](#)
  - menu\_option.h, [47](#)
- closeFile
  - file.c, [6](#)
  - file.h, [7](#)
- closeLevel
  - file\_level.c, [9](#)
  - file\_level.h, [12](#)
- closeLevelList
  - file\_level.c, [9](#)
  - file\_level.h, [13](#)
- const.h, [4](#)
  - FPS, [5](#)
  - min, [5](#)
  - NB\_TILES\_X, [5](#)
  - NB\_TILES\_Y, [5](#)
  - OPTIONS\_PER\_COLUMN, [5](#)
  - SCREEN\_HEIGHT, [5](#)
  - SCREEN\_WIDTH, [5](#)
  - SCROLLING\_MARGIN, [5](#)
  - TILE\_SIZE, [5](#)
  - TILESET\_SIZE, [6](#)
- Cursor, [1](#)
  - tileID, [1](#)
  - x, [1](#)
  - y, [1](#)
- deleteMap
  - map.c, [24](#)
  - map.h, [30](#)
- displayHelp
  - map.c, [25](#)
  - map.h, [31](#)
- extendMap
  - map.c, [26](#)
  - map.h, [32](#)
- FPS
  - const.h, [5](#)
- file.c, [6](#)
  - closeFile, [6](#)
  - openFile, [6](#)
- file.h, [7](#)
  - closeFile, [7](#)
  - openFile, [7](#)
- file\_level.c, [8](#)
  - adaptSizeLevel, [8](#)
  - closeLevel, [9](#)
  - closeLevelList, [9](#)
  - initLevel, [9](#)
  - openLevel, [9](#)
  - readLevelFile, [10](#)
  - searchEndLevel, [10](#)
  - writeLevel, [11](#)
- file\_level.h, [11](#)
  - adaptSizeLevel, [12](#)
  - BUFFER\_SIZE, [12](#)
  - closeLevel, [12](#)
  - closeLevelList, [13](#)
  - initLevel, [13](#)
  - openLevel, [13](#)
  - readLevelFile, [13](#)
  - searchEndLevel, [14](#)
  - writeLevel, [14](#)
- fillColumn
  - map.c, [26](#)
  - map.h, [32](#)
- fillLine
  - map.c, [26](#)
  - map.h, [32](#)
- fillRect
  - map.c, [26](#)
  - map.h, [32](#)
- freeMap
  - map.c, [27](#)
  - map.h, [33](#)
- game.c, [15](#)
  - play, [15](#)
  - printConfirmation, [16](#)
- game.h, [17](#)
  - play, [17](#)
  - printConfirmation, [18](#)
- height
  - Level, [3](#)
- image.c, [18](#)
  - imageLoad, [19](#)
  - imageLoadAlpha, [19](#)
- image.h, [19](#)
  - imageLoad, [20](#)
  - imageLoadAlpha, [20](#)
- imageLoad
  - image.c, [19](#)
  - image.h, [20](#)
- imageLoadAlpha
  - image.c, [19](#)

- image.h, 20
- initLevel
  - file\_level.c, 9
  - file\_level.h, 13
- initMap
  - map.c, 27
  - map.h, 33
- Input, 2
  - key, 2
  - mouse, 2
  - mouseX, 2
  - mouseY, 2
  - quit, 2
- input.c, 20
  - keyboardActionGame, 21
  - keyboardActionMenu, 21
  - updateEvents, 21
  - updateWaitEvents, 22
- key
  - Input, 2
- keyBoardOptions
  - menu\_option.c, 44
  - menu\_option.h, 47
- keyboardActionGame
  - input.c, 21
- keyboardActionMenu
  - input.c, 21
- Level, 2
  - background, 3
  - height, 3
  - map, 3
  - music, 3
  - tileSet, 3
  - timer\_level, 3
  - width, 3
- loadOptions
  - option.c, 49
  - option.h, 51
- lvl
  - Map, 4
- main
  - main.c, 22
- main.c, 22
  - main, 22
- Map, 3
  - lvl, 4
  - screenHeight, 4
  - screenWidth, 4
  - xScroll, 4
- map
  - Level, 3
- map.c, 23
  - deleteMap, 24
  - displayHelp, 25
  - extendMap, 26
  - fillColumn, 26
  - fillLine, 26
  - fillRect, 26
  - freeMap, 27
  - initMap, 27
  - resetMap, 28
  - saveMap, 28
  - updateScreenMap, 29
- map.h, 29
  - deleteMap, 30
  - displayHelp, 31
  - extendMap, 32
  - fillColumn, 32
  - fillLine, 32
  - fillRect, 32
  - freeMap, 33
  - initMap, 33
  - resetMap, 34
  - saveMap, 34
  - updateScreenMap, 35
- menu
  - menu.c, 36
  - menu.h, 38
- menu.c, 35
  - menu, 36
  - menuTileSet, 36
- menu.h, 37
  - menu, 38
  - menuTileSet, 38
- menu\_level.c, 39
  - menuLevel, 40
- menu\_level.h, 40
  - menuLevel, 41
- menu\_option.c, 42
  - chooseKey, 43
  - keyBoardOptions, 44
  - menuOptions, 45
- menu\_option.h, 46
  - chooseKey, 47
  - keyBoardOptions, 47
  - menuOptions, 48
- menuLevel
  - menu\_level.c, 40
  - menu\_level.h, 41
- menuOptions
  - menu\_option.c, 45
  - menu\_option.h, 48
- menuTileSet
  - menu.c, 36
  - menu.h, 38
- min
  - const.h, 5
- mouse
  - Input, 2
- mouseX
  - Input, 2
- mouseY
  - Input, 2
- music

- Level, 3
- NB\_TILES\_X
  - const.h, 5
- NB\_TILES\_Y
  - const.h, 5
- OPTIONS\_PER\_COLUMN
  - const.h, 5
- openFile
  - file.c, 6
  - file.h, 7
- openLevel
  - file\_level.c, 9
  - file\_level.h, 13
- option.c, 49
  - loadOptions, 49
  - saveOptions, 50
- option.h, 50
  - loadOptions, 51
  - saveOptions, 51
- play
  - game.c, 15
  - game.h, 17
- printConfirmation
  - game.c, 16
  - game.h, 18
- printText
  - text.c, 54
  - text.h, 56
- quit
  - Input, 2
- readLevelFile
  - file\_level.c, 10
  - file\_level.h, 13
- resetMap
  - map.c, 28
  - map.h, 34
- SCREEN\_HEIGHT
  - const.h, 5
- SCREEN\_WIDTH
  - const.h, 5
- SCROLLING\_MARGIN
  - const.h, 5
- saveMap
  - map.c, 28
  - map.h, 34
- saveOptions
  - option.c, 50
  - option.h, 51
- screenHeight
  - Map, 4
- screenWidth
  - Map, 4
- searchEndLevel
  - file\_level.c, 10
- file\_level.h, 14
- share.c, 52
  - waitFPS, 52
- share.h, 53
  - waitFPS, 53
- TILE\_SIZE
  - const.h, 5
- TILESET\_SIZE
  - const.h, 6
- text.c, 53
  - captureText, 54
  - printText, 54
- text.h, 55
  - captureText, 55
  - printText, 56
- tileID
  - Cursor, 1
- tileSet
  - Level, 3
- timer\_level
  - Level, 3
- updateEvents
  - input.c, 21
- updateScreenMap
  - map.c, 29
  - map.h, 35
- updateWaitEvents
  - input.c, 22
- waitFPS
  - share.c, 52
  - share.h, 53
- width
  - Level, 3
- writeLevel
  - file\_level.c, 11
  - file\_level.h, 14
- x
  - Cursor, 1
- xScroll
  - Map, 4
- y
  - Cursor, 1