

Super Martin

Generated by Doxygen 1.8.6

Mon Jun 2 2014 13:58:39

Contents

1	Data Structure Documentation	2
1.1	Character Struct Reference	2
1.1.1	Detailed Description	2
1.1.2	Field Documentation	2
1.2	Input Struct Reference	4
1.2.1	Detailed Description	4
1.2.2	Field Documentation	4
1.3	Level Struct Reference	5
1.3.1	Detailed Description	5
1.3.2	Field Documentation	5
1.4	list Struct Reference	6
1.4.1	Detailed Description	6
1.4.2	Field Documentation	7
1.5	Map Struct Reference	7
1.5.1	Detailed Description	7
1.5.2	Field Documentation	7
1.6	node Struct Reference	8
1.6.1	Detailed Description	8
1.6.2	Field Documentation	8
1.7	platform Struct Reference	9
1.7.1	Detailed Description	9
1.7.2	Field Documentation	9
1.8	platformSet Struct Reference	10
1.8.1	Detailed Description	10
1.8.2	Field Documentation	10
1.9	Player Struct Reference	11
1.9.1	Detailed Description	11
1.9.2	Field Documentation	11
1.10	projectile Struct Reference	11
1.10.1	Detailed Description	11
1.10.2	Field Documentation	12
1.11	projectileSet Struct Reference	12
1.11.1	Detailed Description	12
1.11.2	Field Documentation	12
1.12	Sound Struct Reference	13
1.12.1	Detailed Description	13
1.12.2	Field Documentation	13

2	File Documentation	14
2.1	character.c File Reference	14
2.1.1	Detailed Description	14
2.1.2	Function Documentation	14
2.2	character.h File Reference	18
2.2.1	Detailed Description	19
2.2.2	Macro Definition Documentation	19
2.2.3	Function Documentation	19
2.3	const.h File Reference	24
2.3.1	Detailed Description	25
2.3.2	Macro Definition Documentation	25
2.4	enemies.c File Reference	26
2.4.1	Detailed Description	27
2.4.2	Function Documentation	27
2.5	enemies.h File Reference	39
2.5.1	Detailed Description	39
2.5.2	Function Documentation	40
2.6	file.c File Reference	51
2.6.1	Detailed Description	51
2.6.2	Function Documentation	51
2.7	file.h File Reference	52
2.7.1	Detailed Description	52
2.7.2	Function Documentation	52
2.8	file_level.c File Reference	53
2.8.1	Detailed Description	53
2.8.2	Function Documentation	53
2.9	file_level.h File Reference	55
2.9.1	Detailed Description	55
2.9.2	Macro Definition Documentation	56
2.9.3	Function Documentation	56
2.10	game.c File Reference	57
2.10.1	Detailed Description	58
2.10.2	Function Documentation	58
2.11	game.h File Reference	64
2.11.1	Detailed Description	65
2.11.2	Function Documentation	65
2.12	image.c File Reference	72
2.12.1	Detailed Description	73
2.12.2	Function Documentation	73
2.13	image.h File Reference	73

2.13.1 Detailed Description	74
2.13.2 Function Documentation	74
2.14 input.c File Reference	75
2.14.1 Detailed Description	75
2.14.2 Function Documentation	75
2.15 main.c File Reference	77
2.15.1 Detailed Description	77
2.15.2 Function Documentation	78
2.16 map.c File Reference	79
2.16.1 Detailed Description	80
2.16.2 Function Documentation	80
2.17 map.h File Reference	82
2.17.1 Detailed Description	82
2.17.2 Function Documentation	83
2.18 menu.c File Reference	84
2.18.1 Detailed Description	85
2.18.2 Function Documentation	85
2.19 menu.h File Reference	88
2.19.1 Detailed Description	89
2.19.2 Function Documentation	89
2.20 menu_level.c File Reference	92
2.20.1 Detailed Description	92
2.20.2 Function Documentation	93
2.21 menu_level.h File Reference	94
2.21.1 Detailed Description	94
2.21.2 Function Documentation	94
2.22 menu_option.c File Reference	95
2.22.1 Detailed Description	96
2.22.2 Function Documentation	96
2.23 menu_option.h File Reference	99
2.23.1 Detailed Description	99
2.23.2 Function Documentation	100
2.24 mobile_platform.c File Reference	103
2.24.1 Detailed Description	103
2.24.2 Function Documentation	104
2.25 mobile_platform.h File Reference	107
2.25.1 Detailed Description	107
2.25.2 Function Documentation	107
2.26 option.c File Reference	110
2.26.1 Detailed Description	111

2.26.2	Function Documentation	111
2.27	option.h File Reference	113
2.27.1	Detailed Description	113
2.27.2	Function Documentation	113
2.28	player.c File Reference	115
2.28.1	Detailed Description	116
2.28.2	Function Documentation	116
2.29	player.h File Reference	118
2.29.1	Detailed Description	119
2.29.2	Function Documentation	119
2.30	projectile.c File Reference	122
2.30.1	Detailed Description	122
2.30.2	Function Documentation	123
2.31	projectile.h File Reference	125
2.31.1	Detailed Description	125
2.31.2	Function Documentation	126
2.32	share.c File Reference	128
2.32.1	Detailed Description	128
2.32.2	Function Documentation	129
2.33	share.h File Reference	129
2.33.1	Detailed Description	129
2.33.2	Function Documentation	129
2.34	sound.c File Reference	129
2.34.1	Detailed Description	130
2.34.2	Function Documentation	130
2.35	sound.h File Reference	131
2.35.1	Detailed Description	131
2.35.2	Function Documentation	132
2.36	structures.h File Reference	132
2.36.1	Detailed Description	133
2.37	text.c File Reference	133
2.37.1	Detailed Description	133
2.37.2	Function Documentation	134
2.38	text.h File Reference	135
2.38.1	Detailed Description	136
2.38.2	Function Documentation	136

1 Data Structure Documentation

1.1 Character Struct Reference

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

Data Fields

- int `isNpc`
- `SDL_Surface *` `tile`
- `SDL_Rect` `location`
- int `saveX`
- int `saveY`
- float `dirX`
- float `dirY`
- int `isRight`
- int `isOnGround`
- int `doubleJump`
- int `wallJump`
- int `hp`
- int `hpMax`
- int `nbLives`
- int `countStars`
- int `isHurt`
- int `isFalling`
- int `moving`
- int `OnPlatform`
- int `nbProjectile`

1.1.1 Detailed Description

the game characters : player and npc

1.1.2 Field Documentation

1.1.2.1 int `countStars`

character points with caught stars

1.1.2.2 float `dirX`

the direction vectors of the character

1.1.2.3 float `dirY`

the direction vectors of the character

1.1.2.4 int `doubleJump`

0 when not jumping, 1 if made 1 jump (ie can make a double jump), 2 if have made double jump

1.1.2.5 int `hp`

character hit points, dead when 0

1.1.2.6 int hpMax

The HP max

1.1.2.7 int isFalling

indicate if the character is falling

1.1.2.8 int isHurt

indicate if the character was hurt recently

1.1.2.9 int isNpc

the type of npc, 0 if not a npc,

1.1.2.10 int isOnGround

indicate if the character is on the ground

1.1.2.11 int isRight

indicate the character's direction (1 right, 0 left)

1.1.2.12 SDL_Rect location

the location of the character

1.1.2.13 int moving

indicate the number of moving

1.1.2.14 int nbLives

nb of the character has

1.1.2.15 int nbProjectile

indicates the number of projectiles the character has

1.1.2.16 int OnPlatform

indicates if the character is on the platform number x, -1 if not on a platform

1.1.2.17 int saveX

save the position of the pnj to know if he is blocked at the next loop iteration

1.1.2.18 int saveY

save the position of the pnj to know if he is blocked at the next loop iteration

1.1.2.19 SDL_Surface* tile

the tile set of the character

1.1.2.20 int wallJump

indicates if can do wall jump, 0 don't, 1 wall at right, 2 wall at left

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

- [structures.h](#)

1.2 Input Struct Reference

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

Data Fields

- char `key` [SDLK_LAST]
- int `space`
- int `quit`
- int `isJoystick`
- int `useJoystick`
- SDL_Joystick * `joystick`
- char * `button`
- int * `axes`
- int * `hat`
- int `hatMoved`

1.2.1 Detailed Description

the global input structure

1.2.2 Field Documentation

1.2.2.1 int* axes

the joystick axes value : between -32768 and 32767

1.2.2.2 char* button

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

1.2.2.3 int* hat

the joystick hats value : SDLK_HAT_CENTERED, SDLK_HAT_UP, SDLK_HAT_RIGHT, SDLK_HAT_DOWN, SDLK_HAT_LEFT, SDLK_HAT_RIGHTUP, SDLK_HAT_RIGHTDOWN, SDLK_HAT_LEFTUP, SDLK_HAT_LEFTDOWN

1.2.2.4 int hatMoved

indicates if a hat has been moved during the last updateEvent

1.2.2.5 int isJoystick

indicate if there is a joystick plugged in

1.2.2.6 SDL_Joystick* joystick

the joystick

1.2.2.7 char key[SDLK_LAST]

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

1.2.2.8 int quit

is 1 is the SDLK_QUIT event happens

1.2.2.9 int space

Space

1.2.2.10 int useJoystick

indicate if the joystick is willing to be used

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

- input.h

1.3 Level Struct Reference

```
#include <const.h>
```

Data Fields

- unsigned char ** [map](#)
- int [width](#)
- int [height](#)
- int [timer_level](#)
- char [tileSet](#) [MAX_SIZE_FILE_NAME]
- char [tileSet2](#) [MAX_SIZE_FILE_NAME]
- int [tileSetUse](#)
- char [background](#) [MAX_SIZE_FILE_NAME]
- char [music](#) [MAX_SIZE_FILE_NAME]

1.3.1 Detailed Description

The level structure

1.3.2 Field Documentation

1.3.2.1 char background[MAX_SIZE_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_SIZE_FILE_NAME]

The music

1.3.2.5 char tileSet[MAX_SIZE_FILE_NAME]

The tilset

1.3.2.6 char tileSet2[MAX_SIZE_FILE_NAME]

The tilset 2

1.3.2.7 int tileSetUse

The tilset which is used

1.3.2.8 int timer_level

The timer level

1.3.2.9 int width

The width

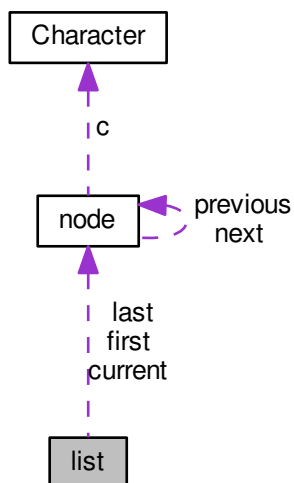
The documentation for this struct was generated from the following file:

- [const.h](#)

1.4 list Struct Reference

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

Collaboration diagram for list:



Data Fields

- [node * first](#)
- [node * current](#)
- [node * last](#)

1.4.1 Detailed Description

the linked list that stock the ennemies

1.4.2 Field Documentation

1.4.2.1 `node* current`

the list's current node

1.4.2.2 `node* first`

the list's first node

1.4.2.3 `node* last`

the list's last node

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

- [structures.h](#)

1.5 Map Struct Reference

```
#include <const.h>
```

Collaboration diagram for Map:



Data Fields

- `Level * lvl`
- `int xScroll`
- `int screenWidth`
- `int screenHeight`

1.5.1 Detailed Description

The map structure

1.5.2 Field Documentation

1.5.2.1 `Level* lvl`

The level

1.5.2.2 int screenHeight

The screen height

1.5.2.3 int screenWidth

The Screen width

1.5.2.4 int xScroll

The xscroll

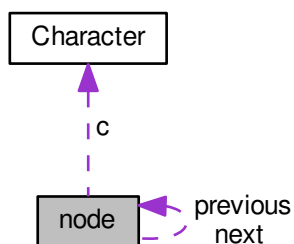
The documentation for this struct was generated from the following file:

- [const.h](#)

1.6 node Struct Reference

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

Collaboration diagram for node:



Data Fields

- [Character * c](#)
- [struct node * next](#)
- [struct node * previous](#)

1.6.1 Detailed Description

node for the enemy list

1.6.2 Field Documentation

1.6.2.1 Character* c

characater of the node

1.6.2.2 struct node* next

next node of the linked list

1.6.2.3 struct node* previous

previous node of the linked list

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

- [structures.h](#)

1.7 platform Struct Reference

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

Data Fields

- SDL_Surface * [sprite](#)
- SDL_Rect [location](#)
- int [xMin](#)
- int [xMax](#)
- int [yMin](#)
- int [yMax](#)
- int [type](#)
- int [direction](#)
- int [speed](#)

1.7.1 Detailed Description

a mobile platform

1.7.2 Field Documentation

1.7.2.1 int direction

the platform direction

1.7.2.2 SDL_Rect location

the platform location

1.7.2.3 int speed

platform speed

1.7.2.4 SDL_Surface* sprite

the platform's sprite

1.7.2.5 int type

0 if horizontal movement, 1 if vertical

1.7.2.6 int xMax

x high limit for displacement

1.7.2.7 int xMin

x low limit for displacement

1.7.2.8 int yMax

y hight limit for deplacement

1.7.2.9 int yMin

y hight limit for deplacement

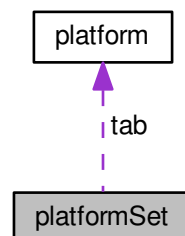
The documentation for this struct was generated from the following file:

- [structures.h](#)

1.8 platformSet Struct Reference

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

Collaboration diagram for platformSet:



Data Fields

- [platform](#) * [tab](#) [[MAX_NB_PLATFORM](#)]
- [int](#) [nb](#)

1.8.1 Detailed Description

the set of the mobile platform

1.8.2 Field Documentation

1.8.2.1 int nb

the number of platform

1.8.2.2 platform* tab[MAX_NB_PLATFORM]

the platform set

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

- [structures.h](#)

1.9 Player Struct Reference

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

Data Fields

- int [levelMax](#)
- int [nbProjectile](#)
- int [nbLives](#)
- int [nbCoins](#)

1.9.1 Detailed Description

The player

1.9.2 Field Documentation

1.9.2.1 int levelMax

The level max

1.9.2.2 int nbCoins

The number of coins

1.9.2.3 int nbLives

The number of life

1.9.2.4 int nbProjectile

The number of projectile

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

- [structures.h](#)

1.10 projectile Struct Reference

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

Data Fields

- SDL_Surface * [sprite](#)
- SDL_Rect [location](#)
- int [direction](#)
- int [fromNPC](#)

1.10.1 Detailed Description

a projectile structure

1.10.2 Field Documentation

1.10.2.1 int direction

the platform direction

1.10.2.2 int fromNPC

indicates if the projectile belongs to the player (0) or to a npc

1.10.2.3 SDL_Rect location

the platform location

1.10.2.4 SDL_Surface* sprite

the platform's sprite

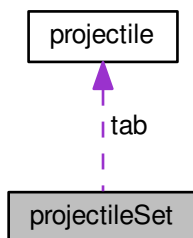
The documentation for this struct was generated from the following file:

- [structures.h](#)

1.11 projectileSet Struct Reference

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

Collaboration diagram for projectileSet:



Data Fields

- [projectile](#) * [tab](#) [[MAX_NB_PROJECTILE](#)]
- int [nb](#)
- int [projectileThrown](#)

1.11.1 Detailed Description

the set of the projectiles

1.11.2 Field Documentation

1.11.2.1 int nb

the number of projectiles on the map

1.11.2.2 int projectileThrown

indicates if a projectile has been thrown by the player and the key wasn't released yet

1.11.2.3 projectile* tab[MAX_NB_PROJECTILE]

the projectile set

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

- [structures.h](#)

1.12 Sound Struct Reference

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

Data Fields

- FMOD_SYSTEM * **sys**
- FMOD_CHANNEL * [music](#)
- FMOD_CHANNEL * [fx](#)
- FMOD_SOUND * [mscSound](#)
- FMOD_SOUND * [fxSound](#)
- float [musicVolume](#)
- float [fxVolume](#)

1.12.1 Detailed Description

the sound gestion structure

1.12.2 Field Documentation

1.12.2.1 FMOD_CHANNEL* fx

the music channel

1.12.2.2 FMOD_SOUND* fxSound

the music sound

1.12.2.3 float fxVolume

the music volume

1.12.2.4 FMOD_SOUND* mscSound

the effects channel

1.12.2.5 FMOD_CHANNEL* music

the sound system

1.12.2.6 float musicVolume

the effects sound

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

- [sound.h](#)

2 File Documentation

2.1 character.c File Reference

manipulate character

```
#include "character.h"
```

Functions

- [Character](#) * [createCharacter](#) (char *tile, int x, int y, int npc, int nbProjectile, int nbCoins, int nbLifes)
- void [freeCharacters](#) ([Character](#) *c)
- int [moveCharacter](#) ([Character](#) *c, float move_left, float move_right, int jump, [Map](#) *m, float *speed, [list](#) *l, [Sound](#) *sound_sys, [platformSet](#) *ps)
- int [tryMovement](#) ([Character](#) *c, int vx, int vy, [Map](#) *m, [list](#) *l, [platformSet](#) *ps, [Sound](#) *sound_sys)
- int [collisionSprite](#) (SDL_Rect s1, SDL_Rect s2)
- void [blitCharacter](#) (SDL_Surface *screen, [Character](#) *c, [Map](#) *m)
- void [presiseMoveCharacter](#) ([Character](#) *c, int vx, int vy, [Map](#) *m, [list](#) *l, [platformSet](#) *ps)
- int [checkWall](#) ([Character](#) *c, [Map](#) *m)
- int [checkFall](#) ([Character](#) *c, [Map](#) *m, [platformSet](#) *ps)

2.1.1 Detailed Description

manipulate character

Author

Xavier COPONET

Date

2014-02-27

2.1.2 Function Documentation

2.1.2.1 void blitCharacter (SDL_Surface * screen, Character * c, Map * m)

blit the character

Parameters

in, out	<i>screen</i>	game screen
---------	---------------	-------------

in	<i>c</i>	the character
in	<i>m</i>	game map

2.1.2.2 int checkFall (Character * *c*, Map * *m*, platformSet * *ps*)

tests if the character's futur position is over a void tile

Parameters

in	<i>c</i>	the monster/character to be tested
in	<i>m</i>	the game map
in	<i>ps</i>	the platform set

Returns

1 if void tile, 0 if not

2.1.2.3 int checkWall (Character * *c*, Map * *m*)

tests if the character's futur position is next to a wall tile

Parameters

in	<i>c</i>	the monster/character to be tested
in	<i>m</i>	the game map

Returns

1 if wall tile, 0 if not

2.1.2.4 int collisionSprite (SDL_Rect *s1*, SDL_Rect *s2*)

int [collisionSprite](#)(SDL_Rect *s1*, SDL_Rect *s2*) determine if there is a collision beteewen two sprites

Parameters

in	<i>s1</i>	the first sprite
in	<i>s2</i>	the second sprite

Returns

3 if there is a collision and *s1* is below *s2*, 2 if there is a collision and *s1* is over *s2*, 0 if there is no collision

2.1.2.5 Character * createCharacter (char * *tile*, int *x*, int *y*, int *npc*, int *nbProjectile*, int *nbCoins*, int *nbLifes*)

create a character

Parameters

in	<i>tile</i>	character tileSet address
in	<i>x</i>	character's x location
in	<i>y</i>	character's y location
in	<i>npc</i>	type of npc if creating a npc, 0 if not
in	<i>nbProjectile</i>	the number of projectiles the character has

in	<i>nbCoins</i>	the number of coins the character has
in	<i>nbLives</i>	the number of life the character has

Returns

character structure pointer

Here is the call graph for this function:

**2.1.2.6 void freeCharacters (Character * c)**

Free a character

Parameters

in, out	<i>c</i>	the character
---------	----------	---------------

2.1.2.7 int moveCharacter (Character * c, float move_left, float move_right, int jump, Map * m, float * speed, list * l, Sound * sound_sys, platformSet * ps)

move player according to the direction

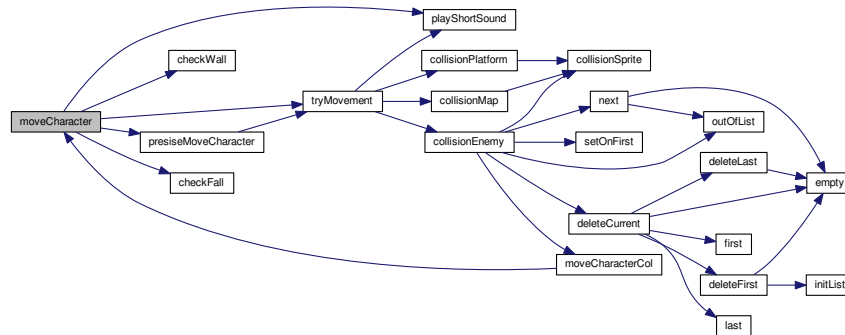
Parameters

in, out	<i>c</i>	the character
in	<i>move_left</i>	indicates if must go to the left
in	<i>move_right</i>	indicates if must go to the right
in	<i>jump</i>	indicates if must jump
in	<i>m</i>	level map
in	<i>speed</i>	movement speed
in, out	<i>l</i>	the enemy list
out	<i>sound_sys</i>	the sound system
out	<i>ps</i>	the platform set

Returns

1 if character was moved without using the precise movement function, 0 if not

Here is the call graph for this function:



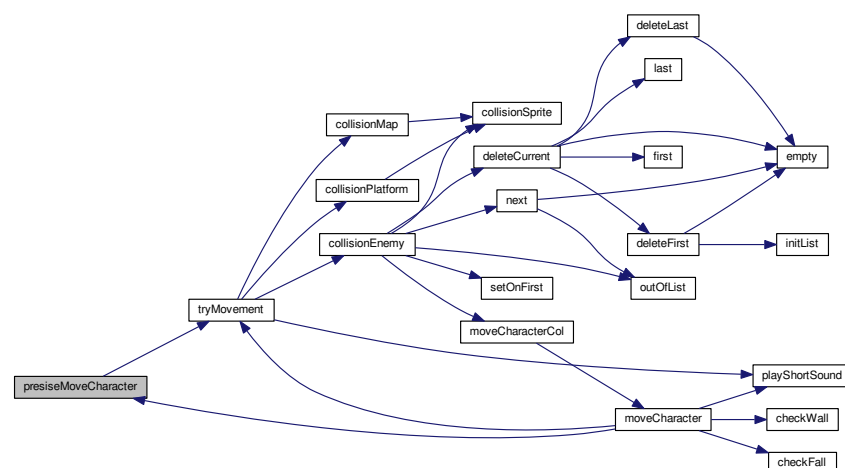
2.1.2.8 void presiseMoveCharacter (Character * c, int vx, int vy, Map * m, list * l, platformSet * ps)

make a more precise move of a character if he can still move but the distance between it and the obstacle is less than its speed

Parameters

in, out	<i>c</i>	the caractere
in	<i>m</i>	the map
in	<i>vx</i>	the horizontal component of the movement vector
in	<i>vy</i>	the vertical component of the movement vector
in, out	<i>l</i>	the enemy list
out	<i>ps</i>	the platform set

Here is the call graph for this function:



2.1.2.9 int tryMovement (Character * c, int vx, int vy, Map * m, list * l, platformSet * ps, Sound * sound_sys)

try to move a character

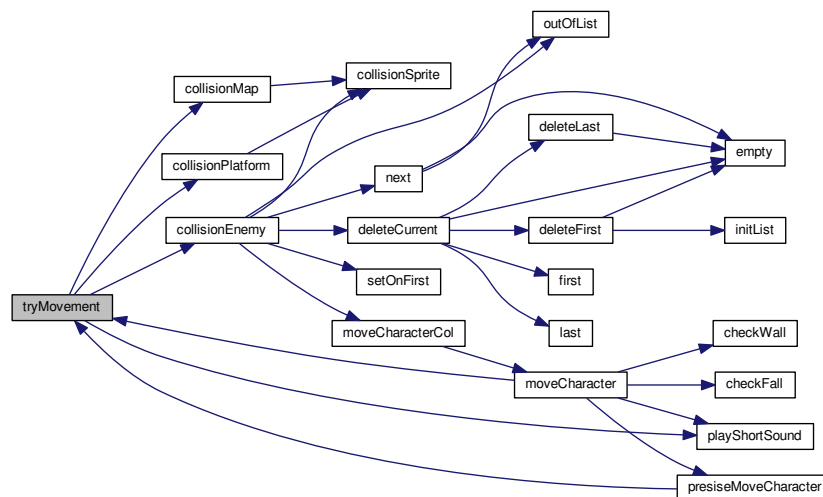
Parameters

in, out	<i>c</i>	the character
in	<i>vx</i>	the horizontal component of the movement vector
in	<i>vy</i>	the vertical component of the movement vector
in	<i>m</i>	the map the character is on
in, out	<i>l</i>	the enemy list
out	<i>ps</i>	the platform set
out	<i>sound_sys</i>	the game sound system

Returns

1 if the character can be moved, 0 if not

Here is the call graph for this function:

**2.2 character.h File Reference**

header de [character.c](#)

```

#include "const.h"
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>
#include "file_level.h"
#include "share.h"
#include "map.h"
#include "structures.h"
#include "image.h"
#include "enemies.h"
#include "mobile_platform.h"
#include "sound.h"

```

Macros

- `#define SGN(X) (((X)==0)?(0):(((X)<0)?(-1):(1)))`
- `#define ABS(X) (((X)<0)?(-(X)):(X))`

Functions

- `int moveCharacter (Character *c, float move_left, float move_right, int jump, Map *m, float *speed, list *l, Sound *sound_sys, platformSet *ps)`
- `void freeCharacters (Character *c)`
- `Character * createCharacter (char *tile, int x, int y, int npc, int nbProjectile, int nbCoins, int nbLives)`
- `void blitCharacter (SDL_Surface *screen, Character *c, Map *m)`
- `int tryMovement (Character *c, int vx, int vy, Map *m, list *l, platformSet *ps, Sound *sound_sys)`
- `void presiseMoveCharacter (Character *c, int vx, int vy, Map *m, list *l, platformSet *ps)`
- `int collisionSprite (SDL_Rect s1, SDL_Rect s2)`
- `int checkFall (Character *c, Map *m, platformSet *ps)`
- `int checkWall (Character *c, Map *m)`

2.2.1 Detailed Description

header de [character.c](#)

Author

Xavier COPONET

Date

2014-02-27

2.2.2 Macro Definition Documentation

2.2.2.1 `#define ABS(X) (((X)<0)?(-(X)):(X))`

X absolute value

2.2.2.2 `#define SGN(X) (((X)==0)?(0):(((X)<0)?(-1):(1)))`

X sign

2.2.3 Function Documentation

2.2.3.1 `void blitCharacter (SDL_Surface * screen, Character * c, Map * m)`

blit the character

Parameters

<i>in, out</i>	<i>screen</i>	game screen
<i>in</i>	<i>c</i>	the character
<i>in</i>	<i>m</i>	game map

2.2.3.2 `int checkFall (Character * c, Map * m, platformSet * ps)`

tests if the character's futur position is over a void tile

Parameters

in	<i>c</i>	the monster/character to be tested
in	<i>m</i>	the game map
in	<i>ps</i>	the platform set

Returns

1 if void tile, 0 if not

2.2.3.3 int checkWall (Character * c, Map * m)

tests if the character's futur position is next to a wall tile

Parameters

in	<i>c</i>	the monster/character to be tested
in	<i>m</i>	the game map

Returns

1 if wall tile, 0 if not

2.2.3.4 int collisionSprite (SDL_Rect s1, SDL_Rect s2)

int [collisionSprite\(SDL_Rect s1, SDL_Rect s2\)](#) determine if there is a collision beteewen two sprites

Parameters

in	<i>s1</i>	the first sprite
in	<i>s2</i>	the second sprite

Returns

3 if there is a collision and s1 is below s2, 2 if there is a collision and s1 is over s2, 0 if there is no collision

2.2.3.5 Character* createCharacter (char * tile, int x, int y, int npc, int nbProjectile, int nbCoins, int nbLifes)

create a character

Parameters

in	<i>tile</i>	character tileSet address
in	<i>x</i>	character's x location
in	<i>y</i>	character's y location
in	<i>npc</i>	type of npc if creating a npc, 0 if not
in	<i>nbProjectile</i>	the number of projectiles the character has
in	<i>nbCoins</i>	the number of coins the character has
in	<i>nbLifes</i>	the number of life the character has

Returns

character structure pointer

Here is the call graph for this function:



2.2.3.6 void freeCharacters (Character * c)

Free a character

Parameters

in, out	c	the character
---------	---	---------------

2.2.3.7 int moveCharacter (Character * c, float move_left, float move_right, int jump, Map * m, float * speed, list * l, Sound * sound_sys, platformSet * ps)

move player according to the direction

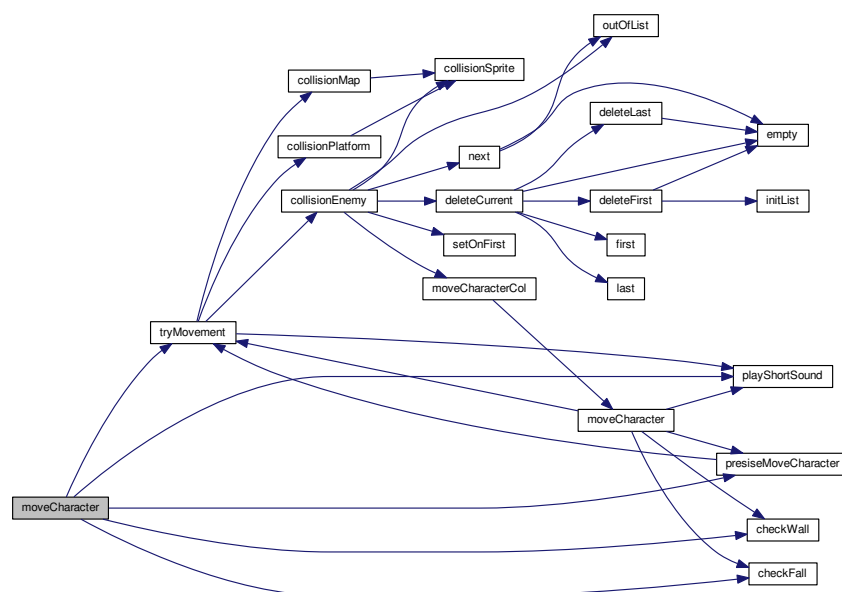
Parameters

in, out	c	the character
in	move_left	indicates if must go to the left
in	move_right	indicates if must go to the right
in	jump	indicates if must jump
in	m	level map
in	speed	movement speed
in, out	l	the enemy list
out	sound_sys	the sound system
out	ps	the platform set

Returns

1 if character was moved without using the precise movement function, 0 if not

Here is the call graph for this function:



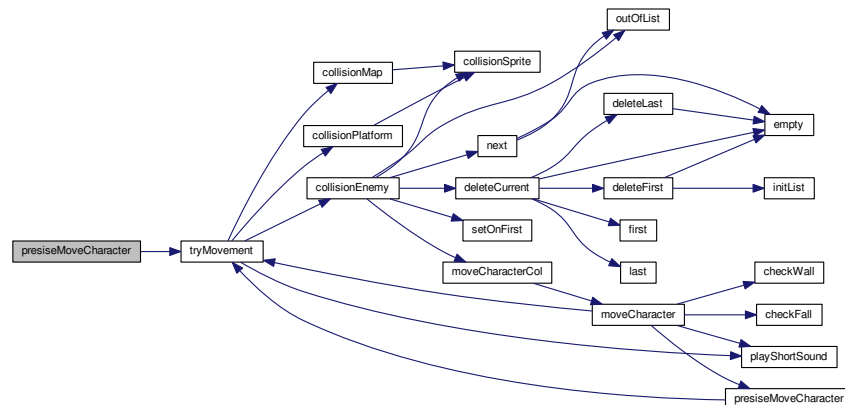
2.2.3.8 void preciseMoveCharacter (Character * c, int vx, int vy, Map * m, list * l, platformSet * ps)

make a more precise move of a character if he can still move but the distance between it and the obstacle is less than its speed

Parameters

in, out	<i>c</i>	the character
in	<i>m</i>	the map
in	<i>vx</i>	the horizontal component of the movement vector
in	<i>vy</i>	the vertical component of the movement vector
in, out	<i>l</i>	the enemy list
out	<i>ps</i>	the platform set

Here is the call graph for this function:



2.2.3.9 int tryMovement (Character * c, int vx, int vy, Map * m, list * l, platformSet * ps, Sound * sound_sys)

try to move a character

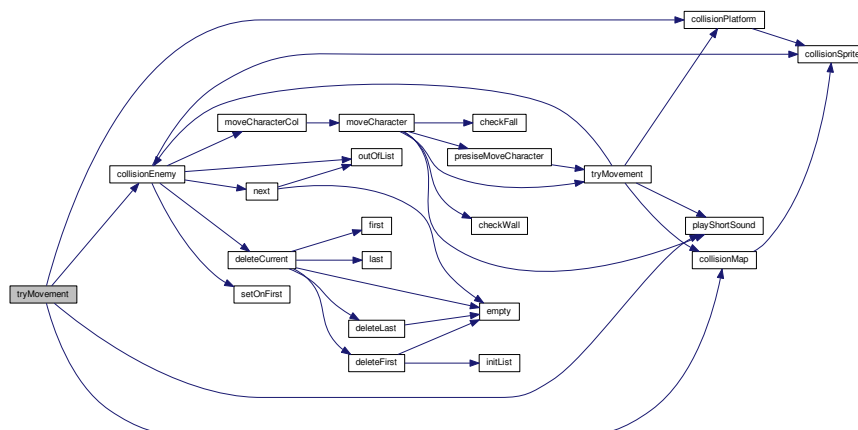
Parameters

in, out	c	the character
in	vx	the horizontal component of the movement vector
in	vy	the vertical component of the movement vector
in	m	the map the character is on
in, out	l	the enemy list
out	ps	the platform set
out	sound_sys	the game sound system

Returns

1 if the character can be moved, 0 if not

Here is the call graph for this function:



2.3 const.h File Reference

containe the program constantes

Data Structures

- struct [Level](#)
- struct [Map](#)

Macros

- #define [TILE_SIZE](#) 16
- #define [SCREEN_WIDTH](#) 1280
- #define [SCREEN_HEIGHT](#) 720
- #define [FPS](#) 60
- #define [MAX_SIZE_FILE_NAME](#) 100
- #define [MARGE_SCROLLING](#) 2
- #define [DEPLACEMENT_POURCENTAGE](#) 0
- #define [GRAVITY_SPEED](#) 1
- #define [JUMP_HEIGHT](#) 13
- #define [MAX_SPEED](#) 5
- #define [SPRING_HEIGHT](#) 22
- #define [COLLISION_ADJUSTMENT](#) 9
- #define [IMG_END_SIZE](#) 80
- #define [NB_TILE_MARYO_WIDTH](#) 4
- #define [NB_TILE_MARYO_HEIGHT](#) 2
- #define [TILE_MAX](#) 18
- #define [FRENQUENCY_CHANGE_MOVING](#) 5
- #define [MAX_NB_PLATFORM](#) 30
- #define [PLATFORM_SPEED](#) 1
- #define [MAX_NB_PROJECTILE](#) 30
- #define [PROJECTILE_SPEED](#) 10
- #define [FRENQUENCY_ROCKET_LAUNCH](#) 2000
- #define [NB_KEY](#) 6
- #define [min](#)(a, b) (a<=b?a:b)

Enumerations

- enum {
 VOID =0, **GROUND**, **COIN** =7, **ROCK**,
 SPRING, **HAMMER**, **HEART**, **ADDLIFE**,
 ENEMY, **TREE**, **FLOWER**, **CLOUD**,
 CANON_L =17, **CANON_R**, **CANON_B** }
- enum { **RIGHT**, **LEFT**, **UP**, **DOWN** }
- enum {
 L =0, **R**, **J**, **P**,
 H }

2.3.1 Detailed Description

containe the program constantes

Author

Xavier COPONET

Date

2014-02-27

2.3.2 Macro Definition Documentation

2.3.2.1 `#define COLLISION_ADJUSTMENT 9`

Collision ajustement

2.3.2.2 `#define DEPLACEMENT_POURCENTAGE 0`

The displacement pourcentage of scrooling

2.3.2.3 `#define FPS 60`

The FPS

2.3.2.4 `#define FRENQUENCY_CHANGE_MOVING 5`

The frequency of changing the legs of maryo when moving

2.3.2.5 `#define FRENQUENCY_ROCKET_LAUNCH 2000`

The frequency of rocket launch

2.3.2.6 `#define GRAVITY_SPEED 1`

The gravity speed

2.3.2.7 `#define IMG_END_SIZE 80`

The image size end

2.3.2.8 `#define JUMP_HEIGHT 13`

The jump height

2.3.2.9 `#define MARGE_SCROLLING 2`

The marge of scrolling

2.3.2.10 `#define MAX_NB_PLATFORM 30`

The the number max of platform

2.3.2.11 `#define MAX_NB_PROJECTILE 30`

The number max of projectile

2.3.2.12 `#define MAX_SIZE_FILE_NAME 100`

The size max of the filenames

2.3.2.13 `#define MAX_SPEED 5`

The max speed

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

mix

2.3.2.15 `#define NB_KEY 6`

The number of key

2.3.2.16 `#define NB_TILE_MARYO_HEIGHT 2`

The number of tile height of maryo

2.3.2.17 `#define NB_TILE_MARYO_WIDTH 4`

The number of tile width of maryo

2.3.2.18 `#define PLATFORM_SPEED 1`

The platform speed

2.3.2.19 `#define PROJECTILE_SPEED 10`

The projectile speed

2.3.2.20 `#define SCREEN_HEIGHT 720`

The screen height

2.3.2.21 `#define SCREEN_WIDTH 1280`

The screen width

2.3.2.22 `#define SPRING_HEIGHT 22`

The spring height

2.3.2.23 `#define TILE_MAX 18`

The the tile max

2.3.2.24 `#define TILE_SIZE 16`

The tile size

2.4 `enemies.c` File Reference

contain enemies gestion function

```
#include "enemies.h"
```

Functions

- void [createEnemy](#) (char *tile, int x, int y, [list](#) *l, int type)
- void [freeEnemies](#) ([list](#) *)
- void [blitEnemies](#) (SDL_Surface *screen, [list](#) *l, [Map](#) *m)

- int `collisionEnemy` (`Character *c`, `list *l`, `Map *m`)
- void `moveEnemies` (`list *l`, `Map *m`, `list *p`, `projectileSet *ps`, `int *launch`)
- int `moveCharacterCol` (`Character *c`, `int move_left`, `int move_right`, `Map *m`)
- `node * newNode` (`Character *c`, `node *n`, `node *p`)
- void `initList` (`list *l`)
- int `empty` (`list *l`)
- int `first` (`list *l`)
- int `last` (`list *l`)
- int `outOfList` (`list *l`)
- void `setOnFirst` (`list *l`)
- void `setOnLast` (`list *l`)
- void `next` (`list *l`)
- void `previous` (`list *l`)
- `Character * getCurrent` (`list *l`)
- int `insertFirst` (`list *l`, `Character *c`)
- int `insertLast` (`list *l`, `Character *c`)
- int `insertAfterCurrent` (`list *l`, `Character *c`)
- int `insertBeforeCurrent` (`list *l`, `Character *c`)
- `Character * deleteFirst` (`list *l`)
- `Character * deleteLast` (`list *l`)
- `Character * deleteCurrent` (`list *l`)

2.4.1 Detailed Description

contain enemies gestion function

Author

Xavier COPONET

Date

2014-04-14

2.4.2 Function Documentation

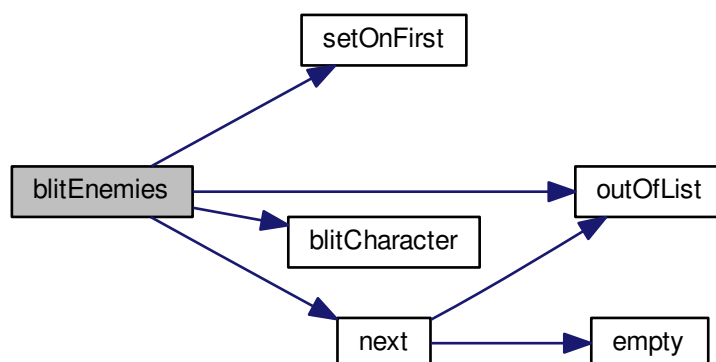
2.4.2.1 void blitEnemies (SDL_Surface * screen, list * l, Map * m)

blit the enemies

Parameters

<code>in, out</code>	<code>screen</code>	game screen
<code>in, out</code>	<code>m</code>	the map
<code>in, out</code>	<code>l</code>	the enemy list

Here is the call graph for this function:



2.4.2.2 int collisionEnemy (Character * c, list * l, Map * m)

determine if there is a collision between the player sprite and an enemy and deals with

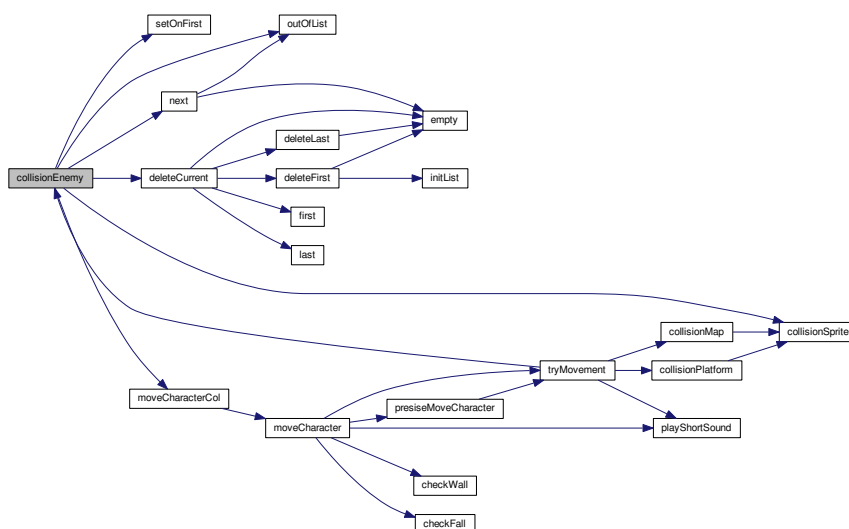
Parameters

in, out	c	the player
in, out	l	the enemy list, change the current node
in	m	the game map

Returns

1 if there is a collision, 0 if not

Here is the call graph for this function:



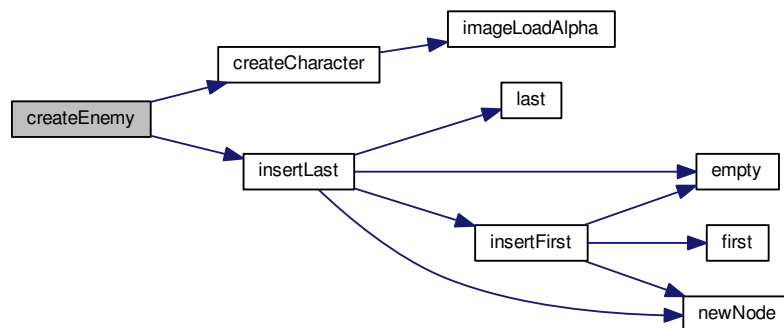
2.4.2.3 void createEnemy (char * *tile*, int *x*, int *y*, list * *l*, int *type*)

creates an enemy and adds it to an enemies list

Parameters

in	<i>tile</i>	the tilset name
in	<i>x</i>	enemy's x location
in	<i>y</i>	enemy's y location
out	<i>l</i>	enemies list
in	<i>type</i>	the type of enemy

Here is the call graph for this function:

**2.4.2.4 enemy * deleteCurrent (list * l)**

delete the current node

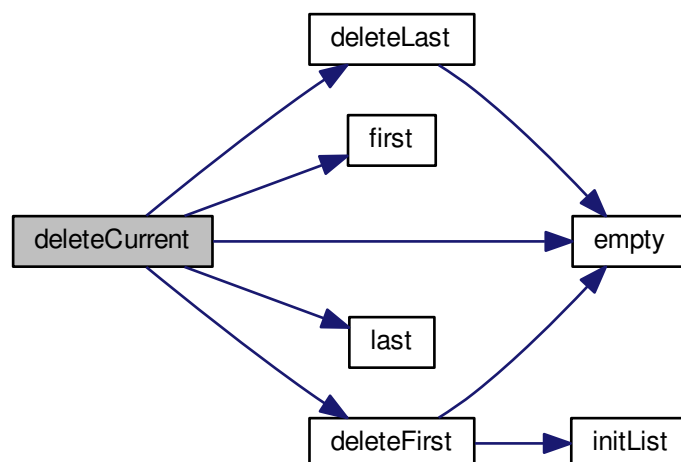
Parameters

out	<i>l</i>	the list which has to be modified
-----	----------	-----------------------------------

Returns

the current node's enemy, NULL if empty list

Here is the call graph for this function:



2.4.2.5 enemy * deleteFirst (list * l)

delete the first node

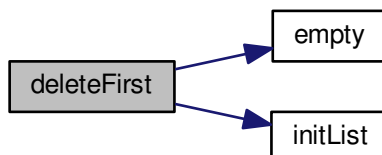
Parameters

out	/	the list which has to be modified
-----	---	-----------------------------------

Returns

the first node's enemy, NULL if empty list

Here is the call graph for this function:



2.4.2.6 enemy * deleteLast (list * l)

delete the last node

Parameters

out	/	the list which has to be modified
-----	---	-----------------------------------

Returns

the last node's enemy, NULL if empty list

Here is the call graph for this function:

**2.4.2.7 int empty (list * /)**

tests if the list is empty

Parameters

in	/	the list to be tested
----	---	-----------------------

Returns

1 if the list is empty, 0 if not

2.4.2.8 int first (list * /)

tests if the current node is the first node

Parameters

in	/	the list to be tested
----	---	-----------------------

Returns

1 if the current node is the first node, 0 if not

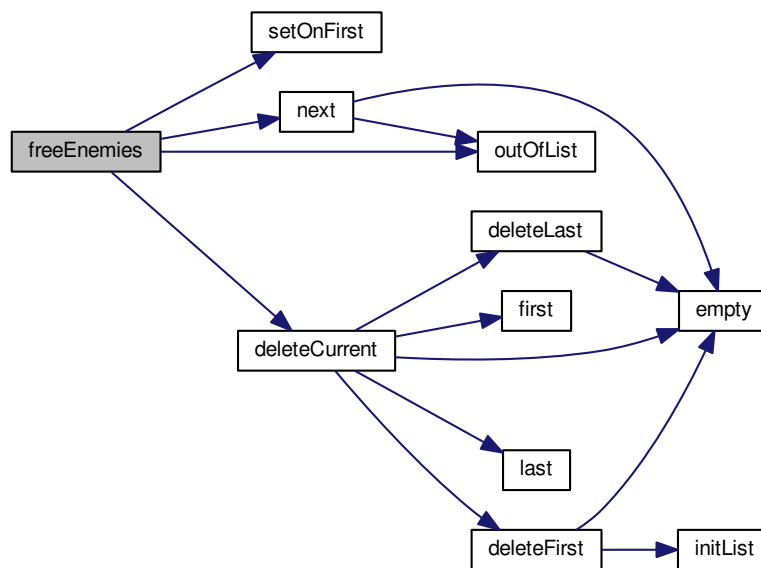
2.4.2.9 void freeEnemies (list * l)

free all the enemies and the list

Parameters

out	/	the enemy list
-----	---	----------------

Here is the call graph for this function:

**2.4.2.10 enemy * getCurrent (list * l)**

get the character of the current node

Parameters

in	/	the list to be modified
----	---	-------------------------

2.4.2.11 void initList (list * l)

initialize the enemy list

Parameters

out	/	the list to be initialized
-----	---	----------------------------

2.4.2.12 int insertAfterCurrent (list * l, Character * c)

insert a enemy just after the current node

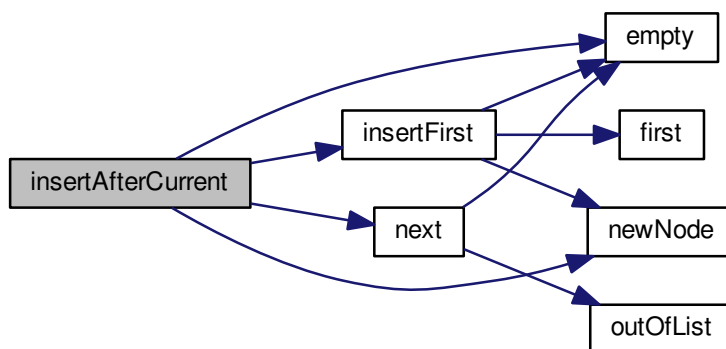
Parameters

out	/	the list in which the enemy has to be inserted
in	c	the character to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:



2.4.2.13 int insertBeforeCurrent (list * l, Character * c)

insert a enemy just before the current node

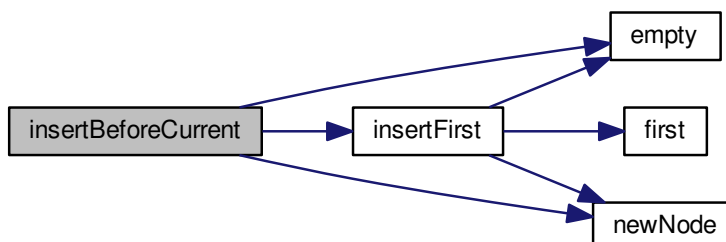
Parameters

out	/	the list in which the enemy has to be inserted
in	c	the character to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:



2.4.2.14 int insertFirst (list * l, Character * c)

insert a enemy as first node

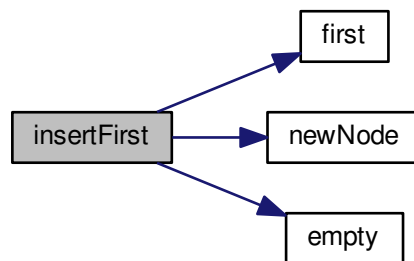
Parameters

out	<i>l</i>	the list in which the enemy has to be inserted
in	<i>c</i>	the charcter to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:

**2.4.2.15 int insertLast (list * l, Character * c)**

insert a enemy as last node

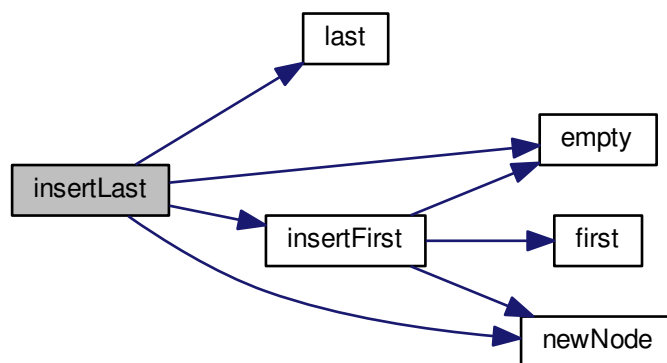
Parameters

out	<i>l</i>	the list in which the enemy has to be inserted
in	<i>c</i>	the character to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:

**2.4.2.16 int last (list * l)**

tests if the current node is the last node

Parameters

in	/	the list to be tested
----	---	-----------------------

Returns

1 if the current node is the last node, 0 if not

2.4.2.17 int moveCharacterCol (Character * c, int move_left, int move_right, Map * m)

moves the character if it's hurt by an enemy

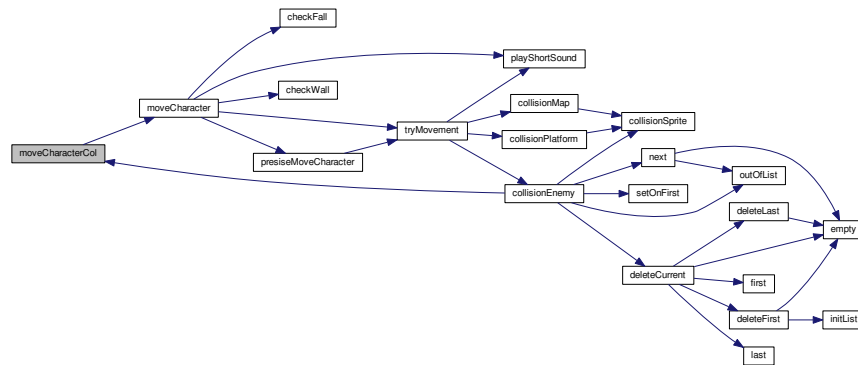
Parameters

in, out	c	the character
in, out	move_left	indicate if the character must move left
in, out	move_right	indicate if the character must move right
in	m	level map

Returns

1 if character was moved without using the precise movement function, 0 if not

Here is the call graph for this function:



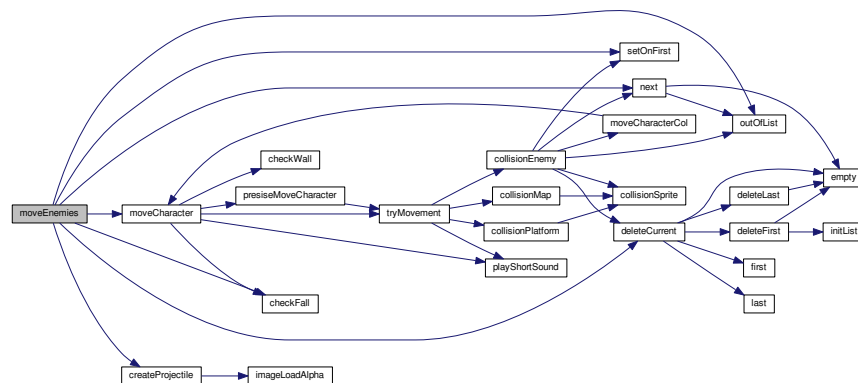
2.4.2.18 void moveEnemies (list * *l*, Map * *m*, list * *p*, projectileSet * *ps*, int * *launch*)

make the enemies moving

Parameters

in, out	<i>l</i>	the enemy list
in	<i>m</i>	the game map
in, out	<i>p</i>	the player list
out	<i>ps</i>	the projectile set
in	<i>launch</i>	if 1, canons can fire an rocket

Here is the call graph for this function:



2.4.2.19 node * newNode (Character * *c*, node * *n*, node * *p*)

creates a new node

Parameters

in	<i>c</i>	the character of the node
in	<i>n</i>	the next node
in	<i>p</i>	the previous node

Returns

a pointer on the created node

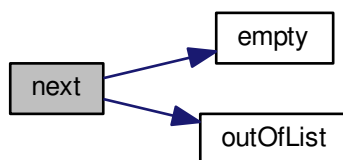
2.4.2.20 void next (list * l)

set the current node on its next node

Parameters

out	<i>l</i>	the list to be modified
-----	----------	-------------------------

Here is the call graph for this function:

**2.4.2.21 int outOfList (list * l)**

tests if the current node is in the list

Parameters

in	<i>l</i>	the list to be tested
----	----------	-----------------------

Returns

1 if the current node is in the list, 0 if not

2.4.2.22 void previous (list * l)

set the current node on its previous node

Parameters

out	<i>l</i>	the list to be modified
-----	----------	-------------------------

2.4.2.23 void setOnFirst (list * l)

set the current node on the first node

Parameters

out	/	the list to be modified
-----	---	-------------------------

2.4.2.24 void setOnLast (list * l)

set the current node on the last node

Parameters

out	/	the list to be modified
-----	---	-------------------------

2.5 enemies.h File Reference

enemies.c header

```
#include "character.h"
#include "projectile.h"
#include "const.h"
#include "structures.h"
```

Functions

- [node](#) * [newNode](#) ([Character](#) *c, [node](#) *n, [node](#) *p)
- void [initList](#) ([list](#) *l)
- int [empty](#) ([list](#) *l)
- int [first](#) ([list](#) *l)
- int [last](#) ([list](#) *l)
- int [outOfList](#) ([list](#) *l)
- void [setOnFirst](#) ([list](#) *l)
- void [setOnLast](#) ([list](#) *l)
- void [next](#) ([list](#) *l)
- void [previous](#) ([list](#) *l)
- [Character](#) * [getCurrent](#) ([list](#) *l)
- int [insertFirst](#) ([list](#) *l, [Character](#) *c)
- int [insertLast](#) ([list](#) *l, [Character](#) *c)
- int [insertAfterCurrent](#) ([list](#) *l, [Character](#) *c)
- int [insertBeforeCurrent](#) ([list](#) *l, [Character](#) *c)
- [Character](#) * [deleteFirst](#) ([list](#) *l)
- [Character](#) * [deleteLast](#) ([list](#) *l)
- [Character](#) * [deleteCurrent](#) ([list](#) *l)
- void [createEnemy](#) (char *tile, int x, int y, [list](#) *l, int type)
- void [freeEnemies](#) ([list](#) *l)
- void [blitEnemies](#) (SDL_Surface *screen, [list](#) *l, [Map](#) *m)
- int [collisionEnemy](#) ([Character](#) *c, [list](#) *l, [Map](#) *m)
- void [moveEnemies](#) ([list](#) *l, [Map](#) *m, [list](#) *p, [projectileSet](#) *ps, int *launch)
- int [moveCharacterCol](#) ([Character](#) *c, int move_left, int move_right, [Map](#) *m)

2.5.1 Detailed Description

enemies.c header

Author

Xavier COPONET

Date

2014-02-27

2.5.2 Function Documentation

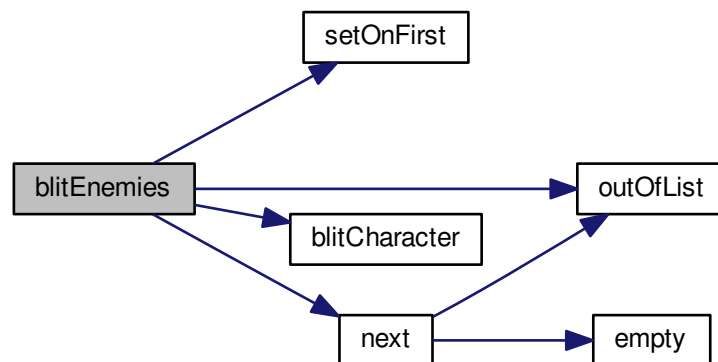
2.5.2.1 void blitEnemies (SDL_Surface * *screen*, list * *l*, Map * *m*)

blit the enemies

Parameters

in, out	<i>screen</i>	game screen
in, out	<i>m</i>	the map
in, out	<i>l</i>	the enemy list

Here is the call graph for this function:



2.5.2.2 int collisionEnemy (Character * *c*, list * *l*, Map * *m*)

determine if there is a collision between the player sprite and an enemy and deals with

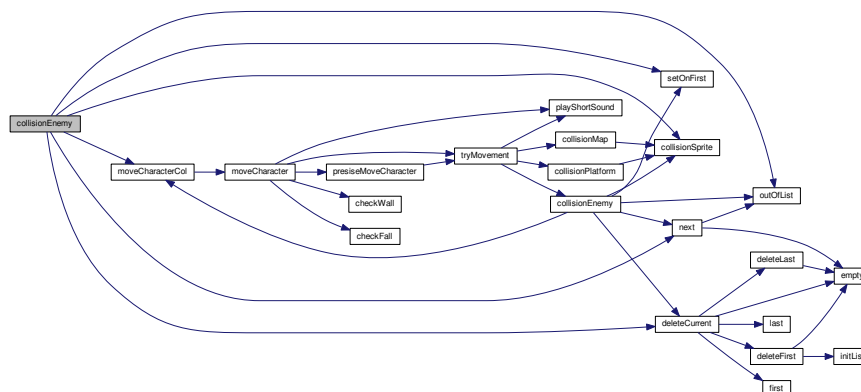
Parameters

in, out	<i>c</i>	the player
in, out	<i>l</i>	the enemy list, change the current node
in	<i>m</i>	the game map

Returns

1 if there is a collision, 0 if not

Here is the call graph for this function:



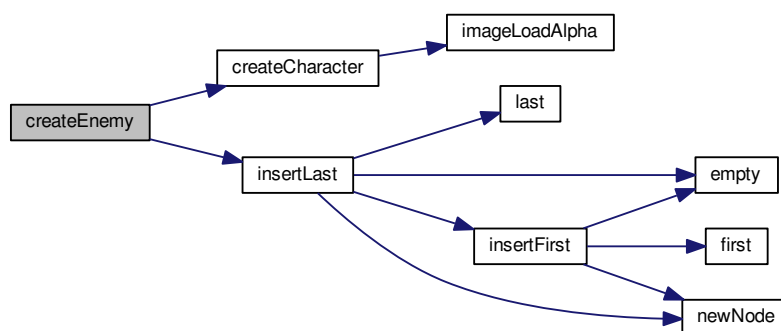
2.5.2.3 void createEnemy (char * *tile*, int *x*, int *y*, list * *l*, int *type*)

creates an enemy and adds it to an enemies list

Parameters

in	<i>tile</i>	the tilset name
in	<i>x</i>	enemy's x location
in	<i>y</i>	enemy's y location
out	<i>l</i>	enemies list
in	<i>type</i>	the type of enemy

Here is the call graph for this function:



2.5.2.4 Character* deleteCurrent (list * *l*)

delete the current node

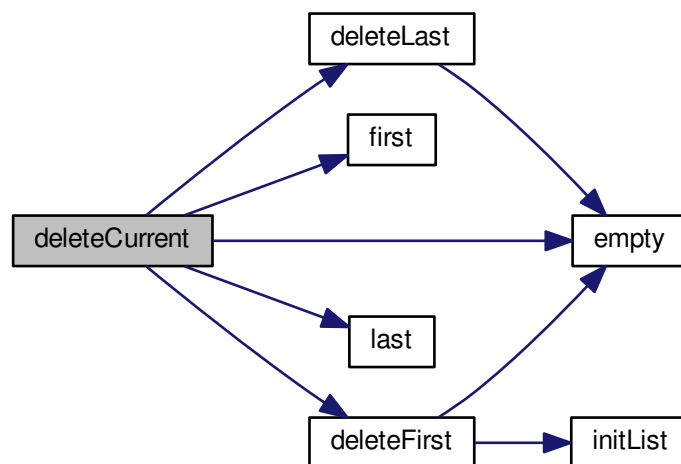
Parameters

out	/	the list which has to be modified
-----	---	-----------------------------------

Returns

the current node's enemy, NULL if empty list

Here is the call graph for this function:

**2.5.2.5 Character* deleteFirst (list * /)**

delete the first node

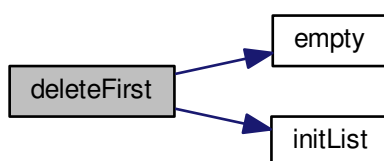
Parameters

out	/	the list which has to be modified
-----	---	-----------------------------------

Returns

the first node's enemy, NULL if empty list

Here is the call graph for this function:



2.5.2.6 Character* deleteLast (list * l)

delete the last node

Parameters

out	/	the list which has to be modified
-----	---	-----------------------------------

Returns

the last node's enemy, NULL if empty list

Here is the call graph for this function:

**2.5.2.7 int empty (list * /)**

tests if the list is empty

Parameters

in	/	the list to be tested
----	---	-----------------------

Returns

1 if the list is empty, 0 if not

2.5.2.8 int first (list * /)

tests if the current node is the first node

Parameters

in	/	the list to be tested
----	---	-----------------------

Returns

1 if the current node is the first node, 0 if not

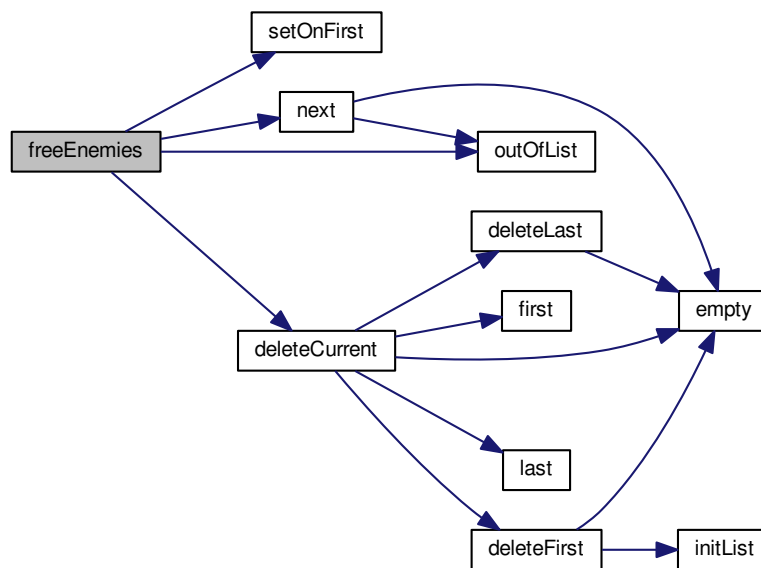
2.5.2.9 void freeEnemies (list * l)

free all the enemies and the list

Parameters

out	/	the enemy list
-----	---	----------------

Here is the call graph for this function:

**2.5.2.10 Character* getCurrent (list * l)**

get the character of the current node

Parameters

in	/	the list to be modified
----	---	-------------------------

2.5.2.11 void initList (list * l)

initialize the enemy list

Parameters

out	/	the list to be initialized
-----	---	----------------------------

2.5.2.12 int insertAfterCurrent (list * l, Character * c)

insert a enemy just after the current node

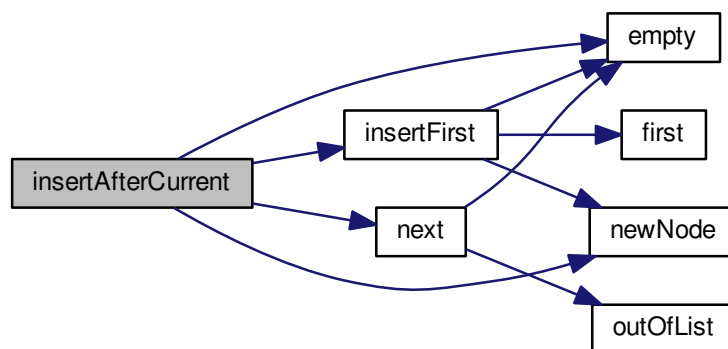
Parameters

out	/	the list in which the enemy has to be inserted
in	c	the character to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:



2.5.2.13 int insertBeforeCurrent (list * l, Character * c)

insert a enemy just before the current node

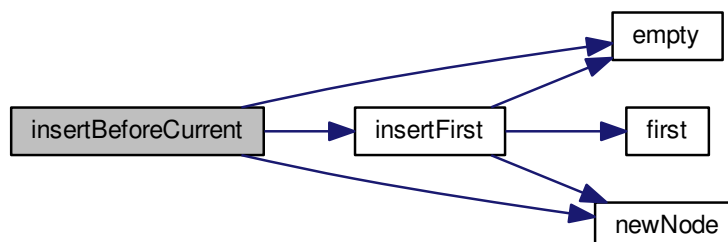
Parameters

out	/	the list in which the enemy has to be inserted
in	c	the character to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:



2.5.2.14 int insertFirst (list * l, Character * c)

insert a enemy as first node

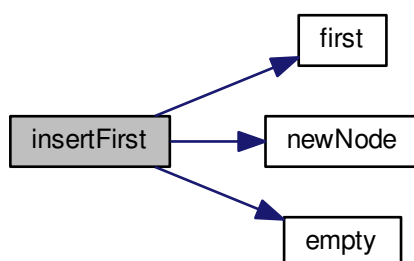
Parameters

out	l	the list in which the enemy has to be inserted
in	c	the charcter to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:



2.5.2.15 int insertLast (list * l, Character * c)

insert a enemy as last node

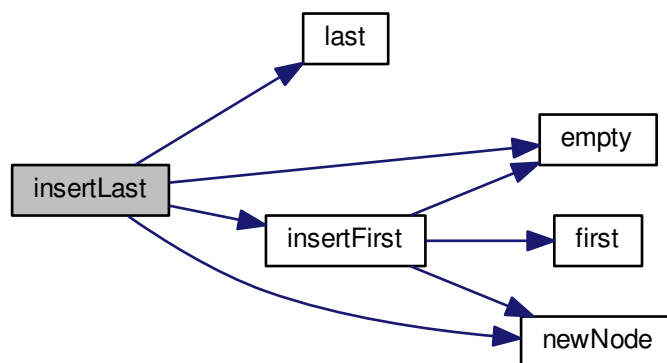
Parameters

out	l	the list in which the enemy has to be inserted
in	c	the character to be inserted

Returns

1 if enemy inserted, 0 if failure

Here is the call graph for this function:

**2.5.2.16 int last (list * l)**

tests if the current node is the last node

Parameters

in	/	the list to be tested
----	---	-----------------------

Returns

1 if the current node is the last node, 0 if not

2.5.2.17 int moveCharacterCol (Character * c, int move_left, int move_right, Map * m)

moves the character if it's hurt by an enemy

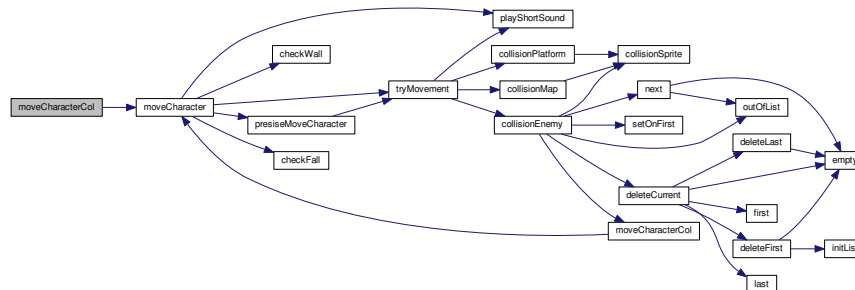
Parameters

in, out	c	the character
in, out	move_left	indicate if the character must move left
in, out	move_right	indicate if the character must move right
in	m	level map

Returns

1 if character was moved without using the precise movement function, 0 if not

Here is the call graph for this function:



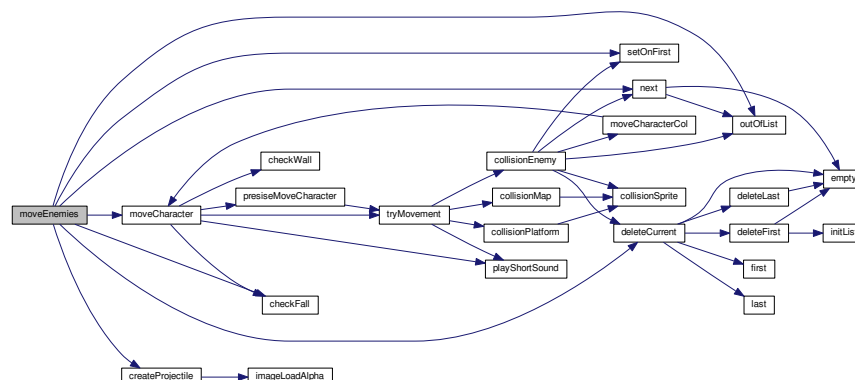
2.5.2.18 void moveEnemies (list * *l*, Map * *m*, list * *p*, projectileSet * *ps*, int * *launch*)

make the enemies moving

Parameters

in, out	<i>l</i>	the enemy list
in	<i>m</i>	the game map
in, out	<i>p</i>	the player list
out	<i>ps</i>	the projectile set
in	<i>launch</i>	if 1, canons can fire an rocket

Here is the call graph for this function:



2.5.2.19 node* newNode (Character * *c*, node * *n*, node * *p*)

creates a new node

Parameters

in	<i>c</i>	the character of the node
in	<i>n</i>	the next node
in	<i>p</i>	the previous node

Returns

a pointer on the created node

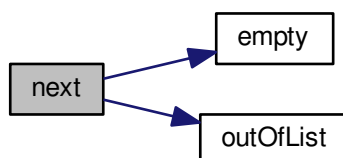
2.5.2.20 void next (list * l)

set the current node on its next node

Parameters

out	<i>l</i>	the list to be modified
-----	----------	-------------------------

Here is the call graph for this function:

**2.5.2.21 int outOfList (list * l)**

tests if the current node is in the list

Parameters

in	<i>l</i>	the list to be tested
----	----------	-----------------------

Returns

1 if the current node is in the list, 0 if not

2.5.2.22 void previous (list * l)

set the current node on its previous node

Parameters

out	<i>l</i>	the list to be modified
-----	----------	-------------------------

2.5.2.23 void setOnFirst (list * l)

set the current node on the first node

Parameters

out	/	the list to be modified
-----	---	-------------------------

2.5.2.24 void setOnLast (list * l)

set the current node on the last node

Parameters

out	/	the list to be modified
-----	---	-------------------------

2.6 file.c File Reference

file access functions

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

Functions

- FILE * [openFile](#) (char name[], char mode[])
- int [closeFile](#) (FILE *ptr_file)

2.6.1 Detailed Description

file access functions

Author

Remi BERTHO

Date

15/03/14

2.6.2 Function Documentation

2.6.2.1 int closeFile (FILE * ptr_file)

close a file

Parameters

in	*ptr_file	the file to be closed
----	-----------	-----------------------

Returns

int 0 if the file was succefully closed, 1 if not

2.6.2.2 FILE * openFile (char name[], char mode[])

open a file

Parameters

in	<i>name</i>	the file name/path
in	<i>mode</i>	the opening mode

Returns

a pointer on the opened file, NULL if error

2.7 file.h File Reference

Prototypes des fonctions d'accès aux fichiers.

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

Functions

- FILE * [openFile](#) (char name[], char mode[])
- int [closeFile](#) (FILE *ptr_fichier)

2.7.1 Detailed Description

Prototypes des fonctions d'accès aux fichiers.

Author

Remi BERTHO

Date

15/03/14

2.7.2 Function Documentation**2.7.2.1 int closeFile (FILE * *ptr_file*)**

close a file

Parameters

in	<i>*ptr_file</i>	the file to be closed
----	------------------	-----------------------

Returns

int 0 if the file was successfully closed, 1 if not

2.7.2.2 FILE* openFile (char *name*[], char *mode*[])

open a file

Parameters

in	<i>name</i>	the file name/path
in	<i>mode</i>	the opening mode

Returns

a pointer on the opened file, NULL if error

2.8 file_level.c File Reference

map file gestion

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

Functions

- [Level *](#) [openLevel](#) (char *file_name, [list](#) *l, [platformSet](#) *ps)
- void [closeLevel](#) ([Level](#) *lvl)
- [Level *](#) [initLevel](#) ([Level](#) *lvl)
- char ** [readLevelFile](#) (char *file_path, int *nb_lvl)
- void [closeLevelList](#) (char **level_names, int nb_lvl)

2.8.1 Detailed Description

map file gestion

Author

Remi BERTHO

Date

15/03/14

Version

1.0

2.8.2 Function Documentation

2.8.2.1 void closeLevel (Level * lvl)

close a level freeing its allocated memory

Parameters

out	<i>lvl</i>	the level to be closed
-----	------------	------------------------

2.8.2.2 void closeLevelList (char ** level_names, int nb_lvl)

desallocate the level name list

Parameters

in, out	<i>level_names</i>	level name list
in	<i>nb_lvl</i>	number of level

2.8.2.3 Level * initLevel (Level * lvl)

Initialize a level assuming its width and height fields are already set

Parameters

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

Returns

a pointer on the level structure

2.8.2.4 Level * openLevel (char * file_name, list * l, platformSet * ps)

Open a map file and stock the map and the enemies

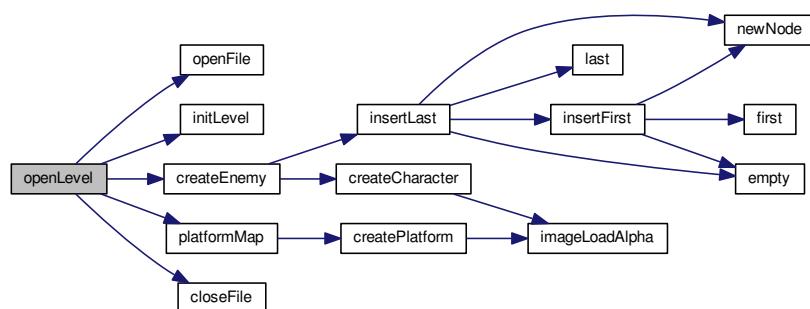
Parameters

in	<i>file_name</i>	the map file name
out	<i>l</i>	the enemy list to stock the enemies.
out	<i>ps</i>	the platform set for mobile platforms

Returns

a pointer on the level structure

Here is the call graph for this function:

**2.8.2.5 char ** readLevelFile (char * file_path, int * nb_lvl)**

read a file level

Parameters

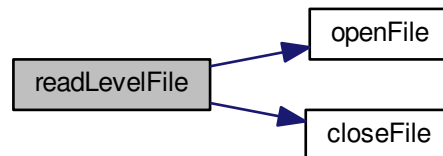
out	<i>nb_lvl</i>	number of level
-----	---------------	-----------------

out	<i>file_path</i>	the file path
-----	------------------	---------------

Returns

pointer on the level list created

Here is the call graph for this function:

**2.9 file_level.h File Reference**

Gestion des fichiers de carte.

```

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

Macros

- `#define BUFFER_SIZE 2`

Functions

- `Level * openLevel (char *file_name, list *, platformSet *ps)`
- `void closeLevel (Level *lvl)`
- `Level * initLevel (Level *lvl)`
- `char ** readLevelFile (char *file_path, int *nb_lvl)`
- `void closeLevelList (char **level_names, int nb_lvl)`

2.9.1 Detailed Description

Gestion des fichiers de carte.

Author

Remi BERTHO

Date

15/03/14

Version

1.0

2.9.2 Macro Definition Documentation

2.9.2.1 #define BUFFER_SIZE 2

The buffer size

2.9.3 Function Documentation

2.9.3.1 void closeLevel (Level * lvl)

close a level freeing its allocated memory

Parameters

out	lvl	the level to be closed
-----	-----	------------------------

2.9.3.2 void closeLevelList (char ** level_names, int nb_lvl)

desallocate the level name list

Parameters

in, out	level_names	level name list
in	nb_lvl	number of level

2.9.3.3 Level* initLevel (Level * lvl)

Initialize a level assuming its width and height fields are already set

Parameters

out	lvl	the level
-----	-----	-----------

Returns

a pointer on the level structure

2.9.3.4 Level* openLevel (char * file_name, list * l, platformSet * ps)

Open a map file and stock the map and the enemies

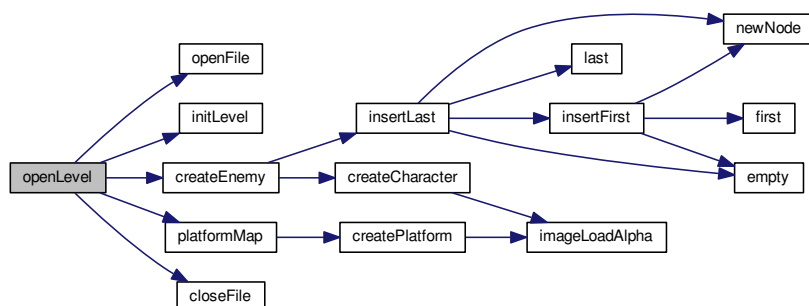
Parameters

in	file_name	the map file name
out	l	the enemy list to stock the enemies.
out	ps	the platform set for mobile platforms

Returns

a pointer on the level structure

Here is the call graph for this function:



2.9.3.5 char** readLevelFile (char * file_path, int * nb_lvl)

read a file level

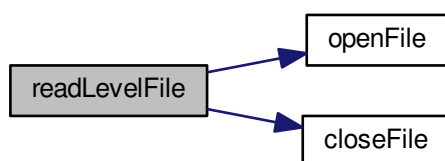
Parameters

out	<i>nb_lvl</i>	number of level
out	<i>file_path</i>	the file path

Returns

pointer on the level list created

Here is the call graph for this function:



2.10 game.c File Reference

contient les fonction liées au jeu

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

Functions

- int [play](#) (SDL_Surface *screen, char *level_name, [Sound](#) *sound_sys, int *go, SDLKey *kc, [Input](#) *in, [Player](#) *player, char player_name[MAX_SIZE_FILE_NAME], int currentLevel, int nb_lv)
- void [printGameOver](#) (SDL_Surface *screen, int *go, [Input](#) *in, [Sound](#) *sound_sys)
- void [printWin](#) (SDL_Surface *screen, int *go, [Input](#) *in, [Sound](#) *sound_sys)
- void [move](#) (float move_left, float move_right, int jump, [Character](#) *player, [Map](#) *m, float *speed, int *acceleration, [list](#) *l, [Sound](#) *sound_sys, [platformSet](#) *ps)
- void [updateSpeed](#) (float *speed, int acceleration)
- void [printPause](#) (SDL_Surface *screen, [Input](#) *in, int *time, int *go, SDLKey *kc)
- Uint32 [decrement](#) (Uint32 interval, void *parameter)
- Uint32 [rocketLaunch](#) (Uint32 interval, void *parameter)
- void [printHUD](#) (SDL_Surface *screen, [Character](#) *player, [Map](#) *m)

2.10.1 Detailed Description

contient les fonction liées au jeu

Author

Xavier COPONET

Date

2014-02-27

2.10.2 Function Documentation

2.10.2.1 Uint32 decrement (Uint32 interval, void * parameter)

the callback function to decrement the time indicator

Parameters

in	<i>interval</i>	the interval between two calls of the function
out	<i>parameter</i>	the time indicator

Returns

the interval between two calls of the function

2.10.2.2 void move (float move_left, float move_right, int jump, Character * player, Map * m, float * speed, int * acceleration, list * l, Sound * sound_sys, platformSet * ps)

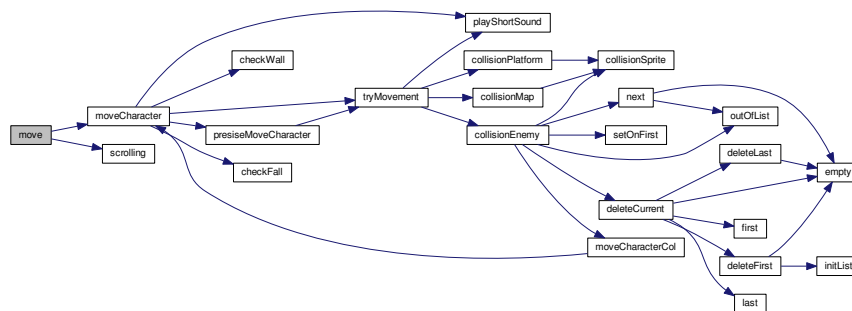
moves the player and scrolls the screen if needed

Parameters

in	<i>move_left</i>	1 if move to the left
in	<i>move_right</i>	1 if move to the right
in	<i>jump</i>	1 if jump
in	<i>player</i>	the player
in	<i>m</i>	the game map

in	<i>speed</i>	the movement speed
out	<i>acceleration</i>	the acceleration of the player
in, out	<i>l</i>	the enemy list
out	<i>sound_sys</i>	the game sound system
out	<i>ps</i>	the platform set

Here is the call graph for this function:



2.10.2.3 `int play (SDL_Surface * screen, char * level_name, Sound * sound_sys, int * go, SDLKey * kc, Input * in, Player * player, char player_name[MAX_SIZE_FILE_NAME], int currentLevel, int nb_lvl)`

initialize a game map and contain the main loop for the game

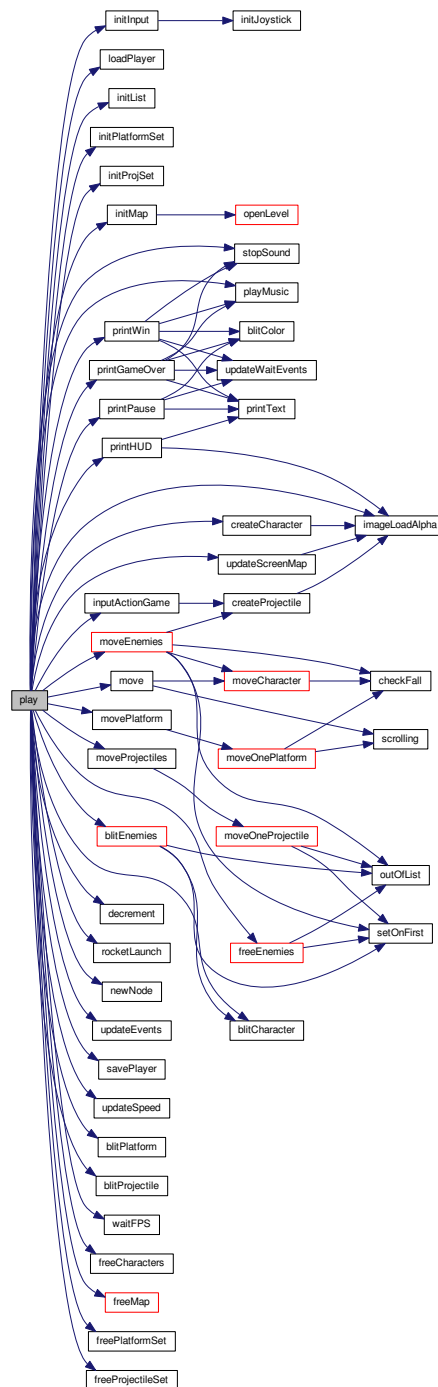
Parameters

in, out	<i>screen</i>	the gamin screen
in	<i>level_name</i>	the name of the level to be played
out	<i>sound_sys</i>	the game sound system
in	<i>kc</i>	the keyboard configuration structure
in, out	<i>go</i>	the software main loop validation
in, out	<i>in</i>	the input gestion structure
in, out	<i>player</i>	the save player structure
in	<i>player_name</i>	the current player name
in	<i>nb_lvl</i>	the number of level
in	<i>currentLevel</i>	the current level

Returns

1 if the maryo dies, 0 if he wins or if he quits the level

Here is the call graph for this function:



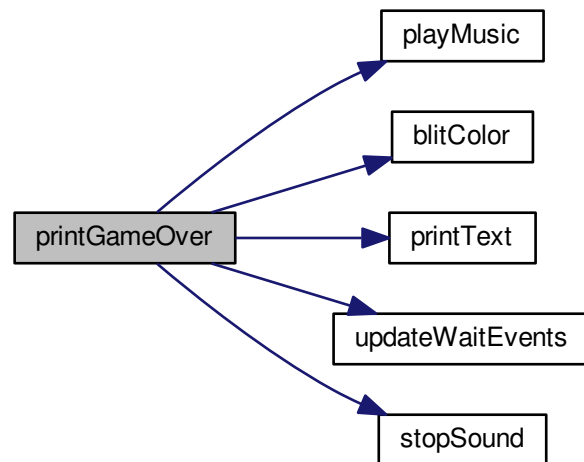
2.10.2.4 void printGameOver (SDL_Surface * screen, int * go, Input * in, Sound * sound_sys)

print the game over screen and wait until the player press a key

Parameters

out	<i>screen</i>	the game screen
out	<i>go</i>	the game function main loop validation
in, out	<i>in</i>	the input structure
out	<i>sound_sys</i>	the sound system

Here is the call graph for this function:



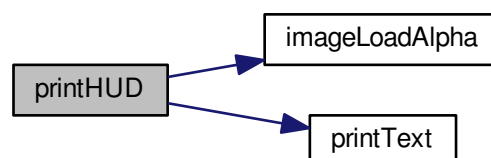
2.10.2.5 void printHUD (SDL_Surface * *screen*, Character * *player*, Map * *m*)

print the player HUD on the screen

Parameters

in, out	<i>screen</i>	the game screen
in	<i>player</i>	the player
in	<i>m</i>	the game map

Here is the call graph for this function:



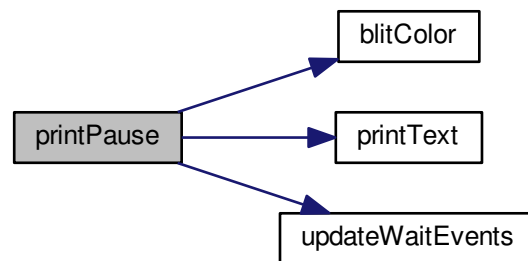
2.10.2.6 void printPause (SDL_Surface * *screen*, Input * *in*, int * *time*, int * *go*, SDLKey * *kc*)

print the pause screen and wait until the player press the pause key

Parameters

out	<i>screen</i>	the game screen
out	<i>go</i>	the game function main loop validation
in, out	<i>in</i>	the input structure
in	<i>time</i>	the current time of the level
in	<i>kc</i>	the keyboard configuration

Here is the call graph for this function:



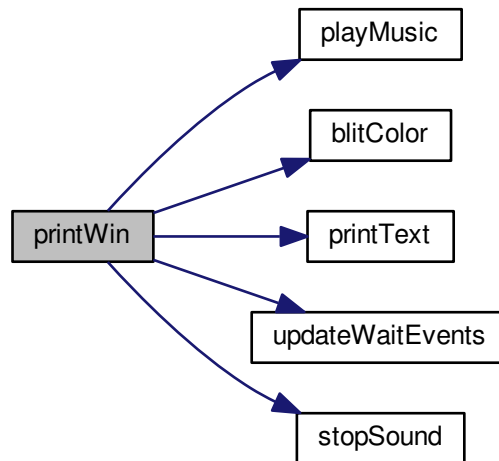
2.10.2.7 void printWin (SDL_Surface * *screen*, int * *go*, Input * *in*, Sound * *sound_sys*)

print the win screen and wait until the player press a key

Parameters

out	<i>screen</i>	the game screen
out	<i>go</i>	the game function main loop validation
in, out	<i>in</i>	the input structure
out	<i>sound_sys</i>	the sound system

Here is the call graph for this function:



2.10.2.8 Uint32 rocketLaunch (Uint32 interval, void * parameter)

the callback function to flip the rocket launch validation

Parameters

in	<i>interval</i>	the interval between two calls of the function
out	<i>parameter</i>	the launch validation

Returns

the interval between two calls of the function

2.10.2.9 void updateSpeed (float * speed, int acceleration)

update the player speed in correlation with its acceleration

Parameters

out	<i>speed</i>	the player speed
out	<i>acceleration</i>	the player acceleration

2.11 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 "text.h"
#include "sound.h"
#include "share.h"
#include "character.h"
#include "file_level.h"
#include "image.h"
#include "map.h"
#include "input.h"
#include "mobile_platform.h"
#include "projectile.h"
#include "player.h"
#include "enemies.h"
```

Functions

- int [play](#) (SDL_Surface *screen, char *level_name, [Sound](#) *sound_sys, int *go, SDLKey *kc, [Input](#) *in, [Player](#) *player, char player_name[MAX_SIZE_FILE_NAME], int currentLevel, int nb_lv)
- void [printGameOver](#) (SDL_Surface *screen, int *go, [Input](#) *in, [Sound](#) *sound_sys)
- void [move](#) (float move_left, float move_right, int jump, [Character](#) *player, [Map](#) *m, float *speed, int *acceleration, [list](#) *l, [Sound](#) *sound_sys, [platformSet](#) *ps)
- void [printWin](#) (SDL_Surface *screen, int *go, [Input](#) *in, [Sound](#) *sound_sys)
- void [updateSpeed](#) (float *speed, int acceleration)
- void [printPause](#) (SDL_Surface *screen, [Input](#) *in, int *time, int *go, SDLKey *kc)
- Uint32 [decrement](#) (Uint32 interval, void *parameter)
- Uint32 [rocketLaunch](#) (Uint32 interval, void *parameter)
- void [printHUD](#) (SDL_Surface *screen, [Character](#) *player, [Map](#) *m)

2.11.1 Detailed Description

[game.c](#) header

Author

Xavier COPONET

Date

2014-02-27

2.11.2 Function Documentation

2.11.2.1 Uint32 decrement (Uint32 interval, void * parameter)

the callback function to decrement the time indicator

Parameters

in	<i>interval</i>	the interval between two calls of the function
out	<i>parameter</i>	the time indicator

Returns

the interval between two calls of the function

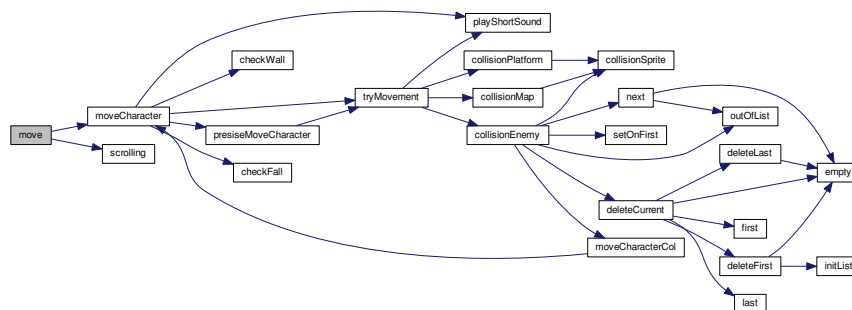
2.11.2.2 void move (float *move_left*, float *move_right*, int *jump*, Character * *player*, Map * *m*, float * *speed*, int * *acceleration*, list * *l*, Sound * *sound_sys*, platformSet * *ps*)

moves the player and scrolls the screen if needed

Parameters

in	<i>move_left</i>	1 if move to the left
in	<i>move_right</i>	1 if move to the right
in	<i>jump</i>	1 if jump
in	<i>player</i>	the player
in	<i>m</i>	the game map
in	<i>speed</i>	the movement speed
out	<i>acceleration</i>	the acceleration of the player
in, out	<i>l</i>	the enemy list
out	<i>sound_sys</i>	the game sound system
out	<i>ps</i>	the platform set

Here is the call graph for this function:



2.11.2.3 int play (SDL_Surface * *screen*, char * *level_name*, Sound * *sound_sys*, int * *go*, SDLKey * *kc*, Input * *in*, Player * *player*, char *player_name*[MAX_SIZE_FILE_NAME], int *currentLevel*, int *nb_lvl*)

initialize a game map and contain the main loop for the game

Parameters

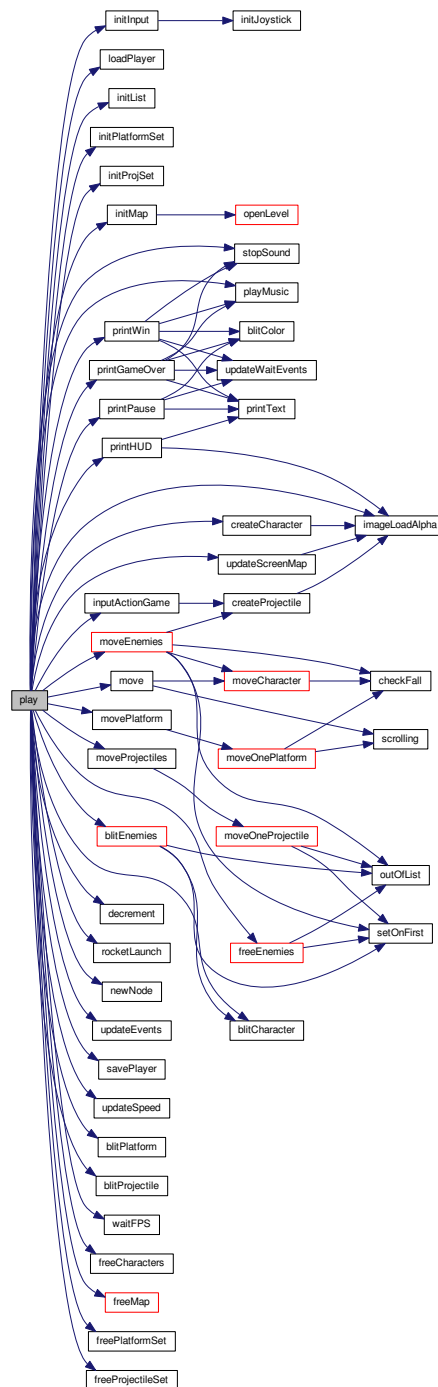
in, out	<i>screen</i>	the gamin screen
in	<i>level_name</i>	the name of the level to be played
out	<i>sound_sys</i>	the game sound system
in	<i>kc</i>	the keyboard configuration structure

in,out	<i>go</i>	the software main loop validation
in,out	<i>in</i>	the input gestion structure
in,out	<i>player</i>	the save player structure
in	<i>player_name</i>	the current player name
in	<i>nb_lvl</i>	the number of level
in	<i>currentLevel</i>	the current level

Returns

1 if the maryo dies, 0 if he wins or if he quits the level

Here is the call graph for this function:



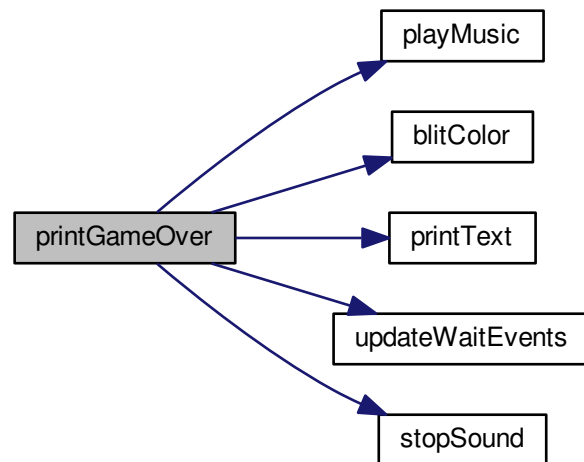
2.11.2.4 void printGameOver (SDL_Surface * screen, int * go, Input * in, Sound * sound_sys)

print the game over screen and wait until the player press a key

Parameters

out	<i>screen</i>	the game screen
out	<i>go</i>	the game function main loop validation
in, out	<i>in</i>	the input structure
out	<i>sound_sys</i>	the sound system

Here is the call graph for this function:



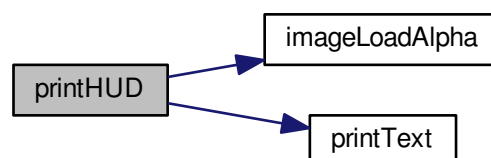
2.11.2.5 void printHUD (SDL_Surface * *screen*, Character * *player*, Map * *m*)

print the player HUD on the screen

Parameters

in, out	<i>screen</i>	the game screen
in	<i>player</i>	the player
in	<i>m</i>	the game map

Here is the call graph for this function:



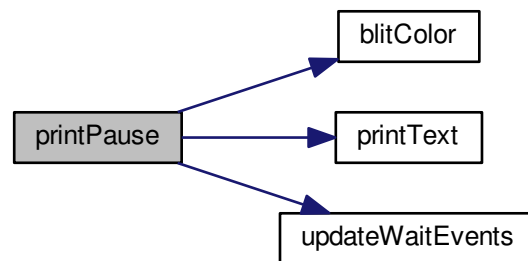
2.11.2.6 void printPause (SDL_Surface * *screen*, Input * *in*, int * *time*, int * *go*, SDLKey * *kc*)

print the pause screen and wait until the player press the pause key

Parameters

out	<i>screen</i>	the game screen
out	<i>go</i>	the game function main loop validation
in, out	<i>in</i>	the input structure
in	<i>time</i>	the current time of the level
in	<i>kc</i>	the keyboard configuration

Here is the call graph for this function:



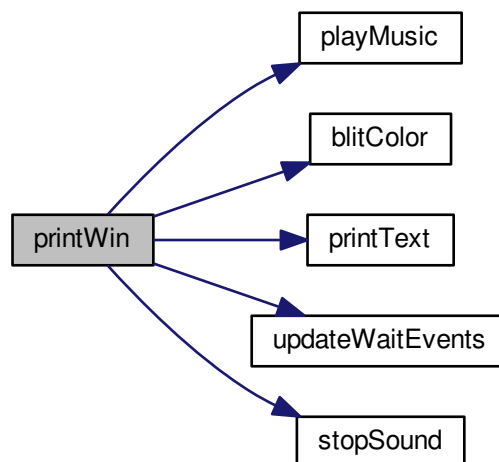
2.11.2.7 void printWin (SDL_Surface * *screen*, int * *go*, Input * *in*, Sound * *sound_sys*)

print the win screen and wait until the player press a key

Parameters

out	<i>screen</i>	the game screen
out	<i>go</i>	the game function main loop validation
in, out	<i>in</i>	the input structure
out	<i>sound_sys</i>	the sound system

Here is the call graph for this function:



2.11.2.8 Uint32 rocketLaunch (Uint32 interval, void * parameter)

the callback function to flip the rocket launch validation

Parameters

in	<i>interval</i>	the interval between two calls of the function
out	<i>parameter</i>	the launch validation

Returns

the interval between two calls of the function

2.11.2.9 void updateSpeed (float * speed, int acceleration)

update the player speed in correlation with its acceleration

Parameters

out	<i>speed</i>	the player speed
out	<i>acceleration</i>	the player acceleration

2.12 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`)
- `void` [blitColor](#) (`Uint32 red`, `Uint32 green`, `Uint32 blue`, `int alpha`, `SDL_Surface *screen`)

2.12.1 Detailed Description

Contain the functions managing the images.

Author

Rémi BERTHO

Date

2014-02-27

2.12.2 Function Documentation

2.12.2.1 `void blitColor (Uint32 red, Uint32 green, Uint32 blue, int alpha, SDL_Surface * screen)`

Blit a color on the screen

Parameters

<i>in</i>	<i>red</i>	the red of the color
<i>in</i>	<i>green</i>	the green of the color
<i>in</i>	<i>blue</i>	the blue of the color
<i>in</i>	<i>alpha</i>	the transparency of the image file
<i>in</i>	<i>screen</i>	the screen

2.12.2.2 `SDL_Surface * imageLoad (char * file_name)`

Load an image

Parameters

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

Returns

a pointer on the `SDL_Surface` created

2.12.2.3 `SDL_Surface * imageLoadAlpha (char * file_name)`

Load an image with alpha management

Parameters

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

Returns

a pointer on the `SDL_Surface` created

2.13 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`)
- `void` [blitColor](#) (`Uint32 red`, `Uint32 green`, `Uint32 blue`, `int alpha`, `SDL_Surface *screen`)

2.13.1 Detailed Description

contient les fonction liées aux images

Author

Rémi BERTHO

Date

2014-02-27

2.13.2 Function Documentation

2.13.2.1 `void blitColor (Uint32 red, Uint32 green, Uint32 blue, int alpha, SDL_Surface * screen)`

Blit a color on the screen

Parameters

<code>in</code>	<code>red</code>	the red of the color
<code>in</code>	<code>green</code>	the green of the color
<code>in</code>	<code>blue</code>	the blue of the color
<code>in</code>	<code>alpha</code>	the transparency of the image file
<code>in</code>	<code>screen</code>	the screen

2.13.2.2 `SDL_Surface* imageLoad (char * file_name)`

Load an image

Parameters

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

Returns

a pointer on the `SDL_Surface` created

2.13.2.3 `SDL_Surface* imageLoadAlpha (char * file_name)`

Load an image with alpha management

Parameters

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

Returns

a pointer on the SDL_Surface created

2.14 input.c File Reference

the funtions to deal with the player inputs

```
#include "input.h"
#include "SDL/SDL_joystick.h"
#include "projectile.h"
```

Functions

- void [initInput](#) ([Input](#) *in)
- void [initJoystick](#) ([Input](#) *in)
- void [freeInput](#) ([Input](#) *in)
- int [updateEvents](#) ([Input](#) *in, int *go)
- void [inputActionGame](#) ([Input](#) *in, float *move_left, float *move_right, int *jump, int *pause, [Character](#) *player, int *acceleration, [SDLKey](#) *kc, [projectileSet](#) *ps)
- int [updateWaitEvents](#) ([Input](#) *in, int *go)
- void [inputActionMenu](#) ([Input](#) *in, int *cursorPos, int *play_level, int nb_options)

2.14.1 Detailed Description

the funtions to deal with the player inputs

Author

Xavier COPONET

Date

2014-03-18

2.14.2 Function Documentation

2.14.2.1 void freeInput (Input * in)

free the input structure

Parameters

<i>out</i>	<i>in</i>	the input structure
------------	-----------	---------------------

2.14.2.2 void initInput (Input * in)

initialize the input structure

Parameters

out	<i>in</i>	the input structure to be initialized
-----	-----------	---------------------------------------

Here is the call graph for this function:

**2.14.2.3 void initJoystick (Input * in)**

initialize the joystic fiels of the input structure

Parameters

out	<i>in</i>	the joystick input structure to be initialized
-----	-----------	--

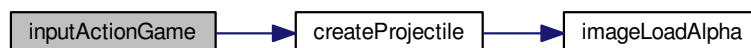
2.14.2.4 void inputActionGame (Input * in, float * move_left, float * move_right, int * jump, int * pause, Character * player, int * acceleration, SDLKey * kc, projectileSet * ps)

perform action command by keyboard or joystick action

Parameters

in	<i>in</i>	the input structure
out	<i>move_left</i>	the left movement boolean
out	<i>move_right</i>	the right movement boolean
out	<i>jump</i>	the jump boolean
out	<i>pause</i>	the pause boolean
in	<i>player</i>	the Player
in	<i>acceleration</i>	the acceleration
in	<i>kc</i>	the keyboard configuration structure
out	<i>ps</i>	the projectile set

Here is the call graph for this function:

**2.14.2.5 void inputActionMenu (Input * in, int * cursorPos, int * play_level, int nb_lvl)**

perform action command by keyboard action

Parameters

in	<i>in</i>	the input structure
out	<i>cursorPos</i>	cursor position
out	<i>play_level</i>	play level
in	<i>nb_lvl</i>	the number of level

2.14.2.6 int updateEvents (Input * *in*, int * *go*)

recuperate keyboard/joystick input with a SDL_PollEvent

Parameters

out	<i>in</i>	the input structure
out	<i>go</i>	the software main loop validation

Returns

1 if a key is activated

2.14.2.7 int updateWaitEvents (Input * *in*, int * *go*)

recuperate keyboard input with a SDL_WaitEvent

Parameters

out	<i>in</i>	the input structure
in, out	<i>go</i>	the software main loop validation

Returns

1 if a key is activated

2.15 main.c File Reference

```
#include "game.h"
#include "const.h"
#include "menu.h"
#include "menu_level.h"
#include "sound.h"
#include "menu_option.h"
#include "option.h"
#include "structures.h"
#include "input.h"
#include "player.h"
```

Functions

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

2.15.1 Detailed Description**Author**

Xavier COPONET

Date

2014-02-27

2.15.2 Function Documentation

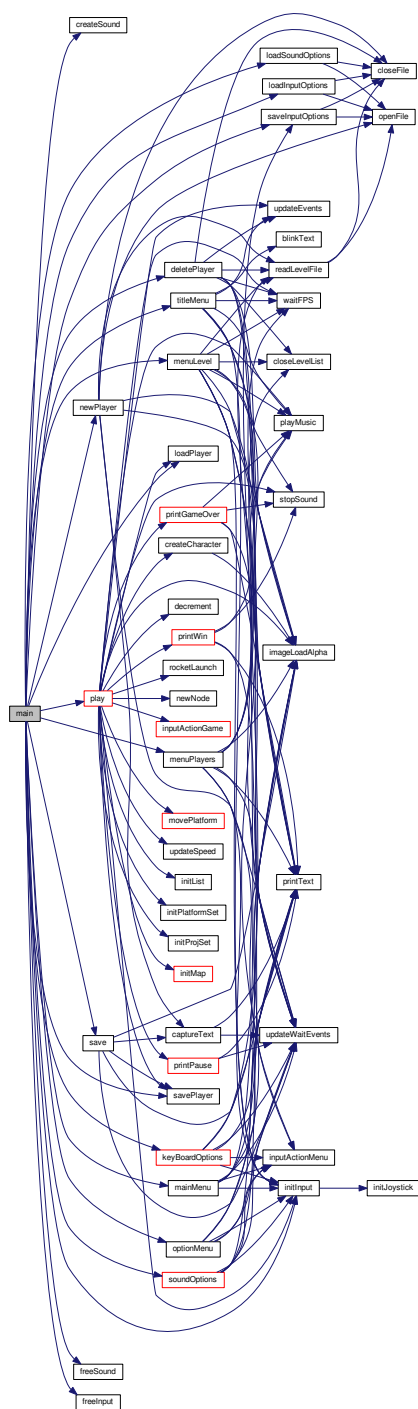
2.15.2.1 `int main (int argc, char * argv[])`

Main

Parameters

<code>in, out</code>	<code><i>argc</i></code>	<code>argc</code>
<code>in, out</code>	<code><i>argv</i></code>	<code>argv</code>

Here is the call graph for this function:



2.16 map.c File Reference

loading and displaying the map

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

Functions

- void [updateScreenMap](#) (SDL_Surface *screen, [Map](#) *m, char *tileset)
- void [scrolling](#) ([Map](#) *m, int direction, float speed)
- [Map](#) * [initMap](#) (SDL_Surface *screen, char *level_name, [list](#) *, [platformSet](#) *ps)
- void [freeMap](#) ([Map](#) *m)
- int [collisionMap](#) (SDL_Rect r, [Map](#) *m, int type)

2.16.1 Detailed Description

loading and displaying the map

Author

Xavier COPONET

Date

2014-03-18

2.16.2 Function Documentation

2.16.2.1 int collisionMap (SDL_Rect *r*, Map * *m*, int *type*)

determine if there is a collision beteewen a sprite and a "wall" of the map

Parameters

in	<i>r</i>	SDL_Rect corresponding to the sprite
in	<i>m</i>	map
in	<i>type</i>	0 if not a projectile

Returns

1 if there is a collision, 0 if not, 2 if collision with star/coin, 3 if spring

Here is the call graph for this function:



2.16.2.2 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:



2.16.2.3 Map * initMap (SDL_Surface * screen, char * level_name, list * l, platformSet * ps)

initialize the map

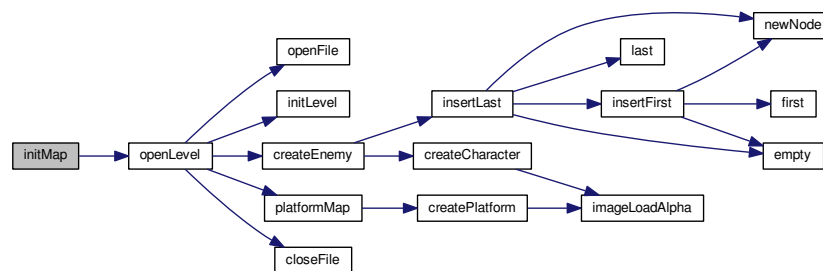
Parameters

in	<i>screen</i>	game screen
in	<i>level_name</i>	lvl name
out	<i>l</i>	the enemy list that stocks the enemies
out	<i>ps</i>	the platform set for the mobile platforms

Returns

pointer on the map

Here is the call graph for this function:



2.16.2.4 void scrolling (Map * m, int direction, float speed)

scroll the map

Parameters

in, out	<i>m</i>	the lvl
in	<i>direction</i>	scrolling direction

in	<i>speed</i>	scrolling speed
----	--------------	-----------------

2.16.2.5 void updateScreenMap (SDL_Surface * *screen*, Map * *m*, char * *tileset*)

update and display the map

Parameters

in, out	<i>screen</i>	
in	<i>m</i>	The map
in	<i>tileset</i>	the level tileset

Here is the call graph for this function:



2.17 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"

```

Functions

- void [updateScreenMap](#) (SDL_Surface *screen, [Map](#) *m, char *tileset)
- void [scrolling](#) ([Map](#) *m, int direction, float speed)
- [Map](#) * [initMap](#) (SDL_Surface *screen, char *level_name, [list](#) *, [platformSet](#) *ps)
- void [freeMap](#) ([Map](#) *m)
- int [collisionMap](#) (SDL_Rect r, [Map](#) *m, int type)

2.17.1 Detailed Description

map.c header

Author

Xavier COPONET

Date

2014-03-18

2.17.2 Function Documentation

2.17.2.1 int collisionMap (SDL_Rect *r*, Map * *m*, int *type*)

determine if there is a collision between a sprite and a "wall" of the map

Parameters

in	<i>r</i>	SDL_Rect corresponding to the sprite
in	<i>m</i>	map
in	<i>type</i>	0 if not a projectile

Returns

1 if there is a collision, 0 if not, 2 if collision with star/coin, 3 if spring

Here is the call graph for this function:

2.17.2.2 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:

2.17.2.3 Map* initMap (SDL_Surface * *screen*, char * *level_name*, list * *l*, platformSet * *ps*)

initialize the map

Parameters

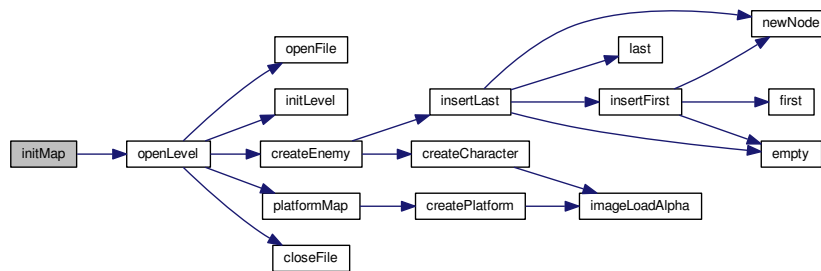
in	<i>screen</i>	game screen
----	---------------	-------------

in	<i>level_name</i>	lvl name
out	<i>l</i>	the enemy list that stocks the enemies
out	<i>ps</i>	the platform set for the mobile platforms

Returns

pointer on the map

Here is the call graph for this function:

**2.17.2.4 void scrolling (Map * m, int direction, float speed)**

scroll the map

Parameters

in, out	<i>m</i>	the lvl
in	<i>direction</i>	scrolling direction
in	<i>speed</i>	scrolling speed

2.17.2.5 void updateScreenMap (SDL_Surface * screen, Map * m, char * tileset)

update and display the map

Parameters

in, out	<i>screen</i>	
in	<i>m</i>	The map
in	<i>tileset</i>	the level tileset

Here is the call graph for this function:

**2.18 menu.c File Reference**

contains some functions tied to the title and main menu


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

Functions

- int [titleMenu](#) (SDL_Surface *screen, int *go, [Sound](#) *sound_sys, [Input](#) *in)
- Uint32 [blinkText](#) (Uint32 interval, void *param)
- int [mainMenu](#) (SDL_Surface *screen, int *go, [Sound](#) *sound_sys, char *player_name, [Input](#) *in)
- int [menuPlayers](#) (SDL_Surface *screen, char player_name[MAX_SIZE_FILE_NAME], int *go, [Sound](#) *sound_sys, [Input](#) *in)

2.18.1 Detailed Description

contains some functions tied to the title and main menu

Author

Xavier COPONET

Date

2014-02-27

2.18.2 Function Documentation

2.18.2.1 Uint32 [blinkText](#) (Uint32 *interval*, void * *param*)

toggle the printing text boolean (timer callback function)

Parameters

in	<i>interval</i>	the interval between two callback of the function
in	<i>param</i>	a parameter

Returns

1000 if the boolean is right, 600 if not

2.18.2.2 int [mainMenu](#) (SDL_Surface * *screen*, int * *go*, [Sound](#) * *sound_sys*, char * *player_name*, [Input](#) * *in*)

print the main menu on the screen

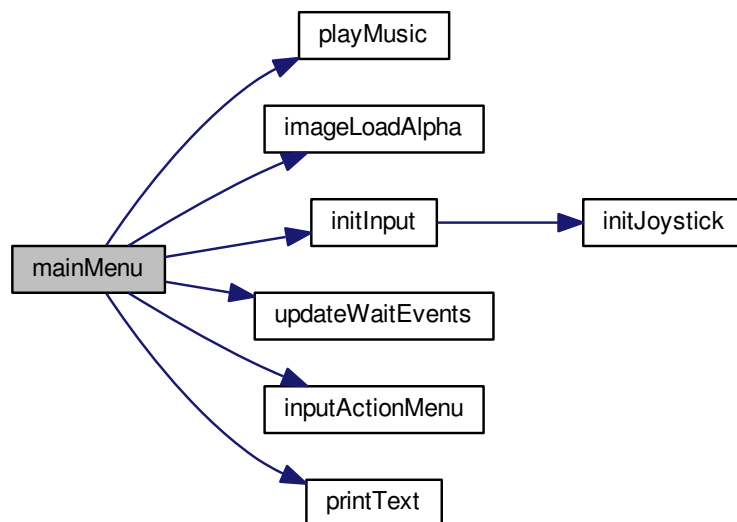
Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
out	<i>sound_sys</i>	sound system
in	<i>player_name</i>	the current player name
in, out	<i>in</i>	the input structure

Returns

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

Here is the call graph for this function:



2.18.2.3 `int menuPlayers (SDL_Surface * screen, char player_name[MAX_SIZE_FILE_NAME], int * go, Sound * sound_sys, Input * in)`

Menu to choose the player

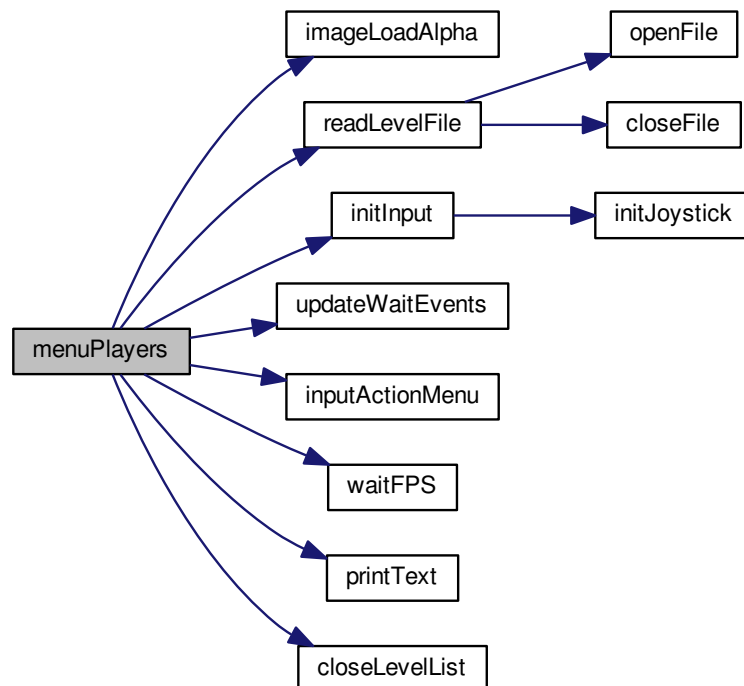
Parameters

out	<i>screen</i>	game screen
out	<i>player_name</i>	the name of the current player
in, out	<i>go</i>	main loop validation
in, out	<i>sound_sys</i>	the sound system
in, out	<i>in</i>	the input structure

Returns

2 if the option NewPlayer has been choosen, 1 if a player has been choosen, -1 if esc

Here is the call graph for this function:



2.18.2.4 int titleMenu (SDL_Surface * *screen*, int * *go*, Sound * *sound_sys*, Input * *in*)

print the title menu on the screen

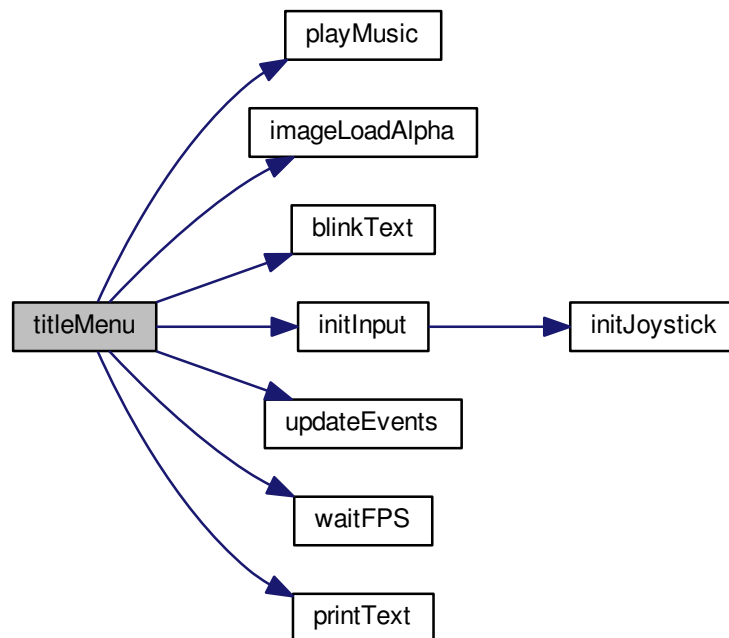
Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
out	<i>sound_sys</i>	sound
in, out	<i>in</i>	the input structure

Returns

1 if the enter key has been pushed

Here is the call graph for this function:

**2.19 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 "sound.h"
#include "share.h"
#include "image.h"
#include "input.h"

```

Functions

- int [titleMenu](#) (SDL_Surface *screen, int *go, [Sound](#) *sound_sys, [Input](#) *in)
- Uint32 [blinkText](#) (Uint32 interval, void *param)
- int [mainMenu](#) (SDL_Surface *screen, int *go, [Sound](#) *sound_sys, char *player_name, [Input](#) *in)

- int [menuPlayers](#) (SDL_Surface *screen, char player_name[MAX_SIZE_FILE_NAME], int *go, [Sound](#) *sound_sys, [Input](#) *in)

2.19.1 Detailed Description

header de [menu.c](#)

Author

Xavier COPONET

Date

2014-02-27

2.19.2 Function Documentation

2.19.2.1 Uint32 blinkText (Uint32 interval, void * param)

toggle the printing text boolean (timer callback function)

Parameters

in	<i>interval</i>	the interval between two callback of the function
in	<i>param</i>	a parameter

Returns

1000 if the boolean is right, 600 if not

2.19.2.2 int mainMenu (SDL_Surface * screen, int * go, Sound * sound_sys, char * player_name, Input * in)

print the main menu on the screen

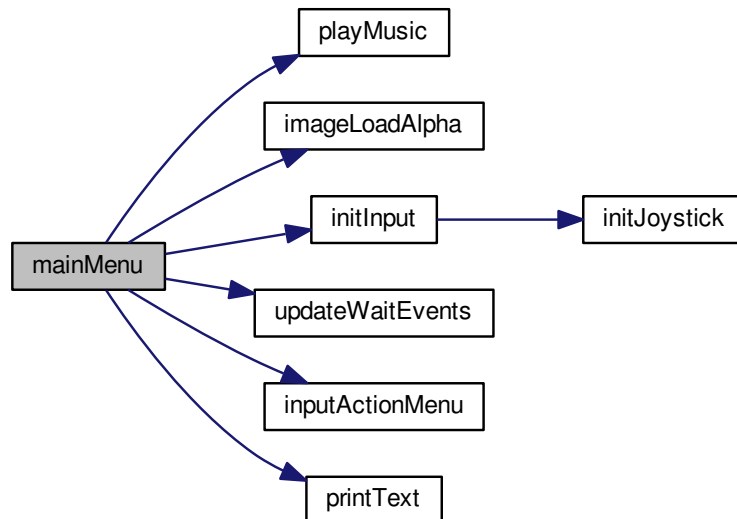
Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
out	<i>sound_sys</i>	sound system
in	<i>player_name</i>	the current player name
in, out	<i>in</i>	the input structure

Returns

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

Here is the call graph for this function:



2.19.2.3 `int menuPlayers (SDL_Surface * screen, char player_name[MAX_SIZE_FILE_NAME], int * go, Sound * sound_sys, Input * in)`

Menu to choose the player

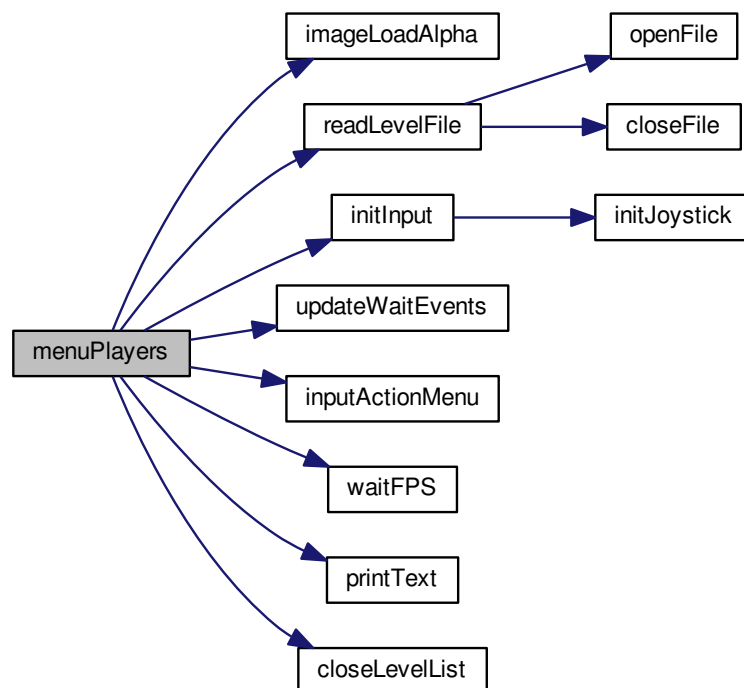
Parameters

out	<i>screen</i>	game screen
out	<i>player_name</i>	the name of the current player
in, out	<i>go</i>	main loop validation
in, out	<i>sound_sys</i>	the sound system
in, out	<i>in</i>	the input structure

Returns

2 if the option NewPlayer has been choosen, 1 if a player has been choosen, -1 if esc

Here is the call graph for this function:



2.19.2.4 int titleMenu (SDL_Surface * *screen*, int * *go*, Sound * *sound_sys*, Input * *in*)

print the title menu on the screen

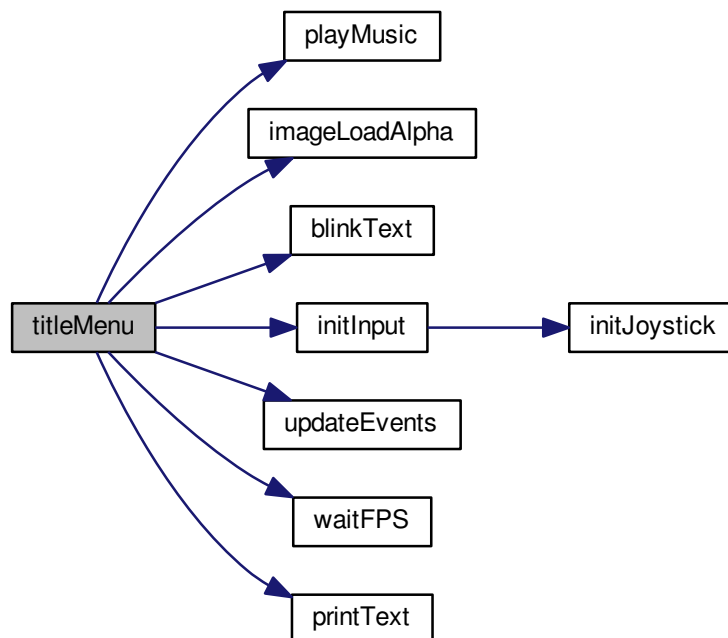
Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
out	<i>sound_sys</i>	sound
in, out	<i>in</i>	the input structure

Returns

1 if the enter key has been pushed

Here is the call graph for this function:

**2.20 menu_level.c File Reference**

level choose menu

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

Functions

- int [menuLevel](#) (SDL_Surface *screen, char level_name[[MAX_SIZE_FILE_NAME](#)], [Sound](#) *sound_sys, char player_name[[MAX_SIZE_FILE_NAME](#)], [Player](#) *player, int *go, int *nb_lvl, [Input](#) *in)

2.20.1 Detailed Description

level choose menu

Author

Remi BERTHO

Date

15/03/14

2.20.2 Function Documentation

2.20.2.1 int menuLevel (SDL_Surface * screen, char level_name[MAX_SIZE_FILE_NAME], Sound * sound_sys, char player_name[MAX_SIZE_FILE_NAME], Player * player, int * go, int * nb_lv, Input * in)

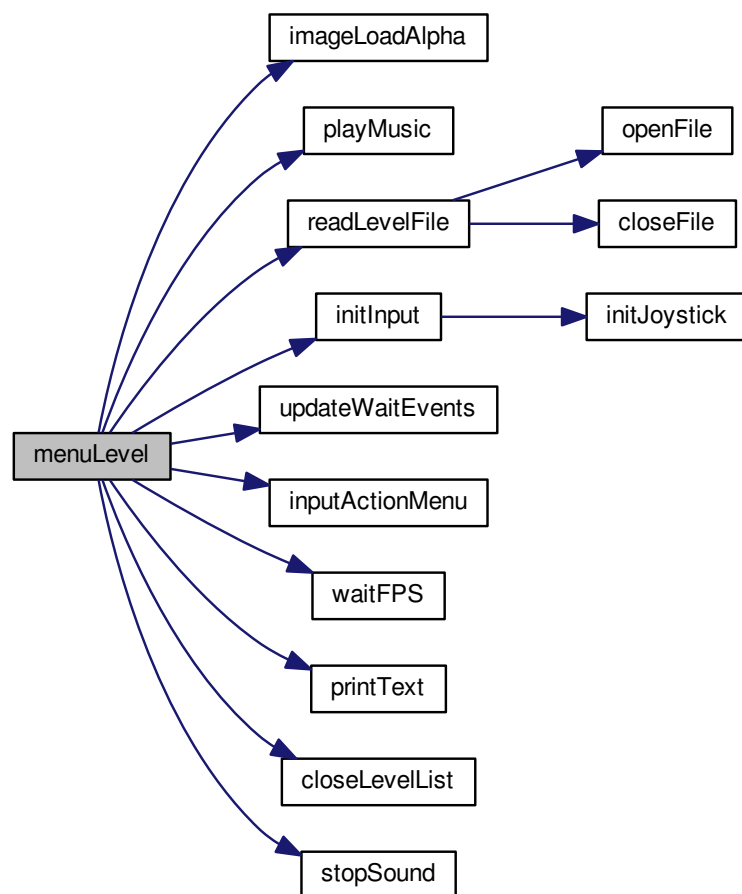
Parameters

out	screen	game screen
out	level_name	the name of the level we will want to launch
in, out	sound_sys	the sound system
in, out	player	the player structure
in, out	go	the soft main loop validation
in, out	player_name	the player name
in	nb_lv	the number of level
in, out	in	the input structure

Returns

1 if a level has been choosen, 0 if not

Here is the call graph for this function:



2.21 menu_level.h File Reference

Menu gerant le choix du niveau.

```
#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 "structures.h"
#include "file_level.h"
#include "share.h"
#include "text.h"
#include "sound.h"
#include "image.h"
#include "input.h"
```

Functions

- int [menuLevel](#) (SDL_Surface *screen, char level_name[[MAX_SIZE_FILE_NAME](#)], [Sound](#) *sound_sys, char player_name[[MAX_SIZE_FILE_NAME](#)], [Player](#) *player, int *go, int *nb_lvl, [Input](#) *in)

2.21.1 Detailed Description

Menu gerant le choix du niveau.

Author

Remi BERTHO

Date

15/03/14

Version

2.0

2.21.2 Function Documentation

2.21.2.1 int [menuLevel](#) (SDL_Surface * *screen*, char *level_name*[[MAX_SIZE_FILE_NAME](#)], [Sound](#) * *sound_sys*, char *player_name*[[MAX_SIZE_FILE_NAME](#)], [Player](#) * *player*, int * *go*, int * *nb_lvl*, [Input](#) * *in*)

Parameters

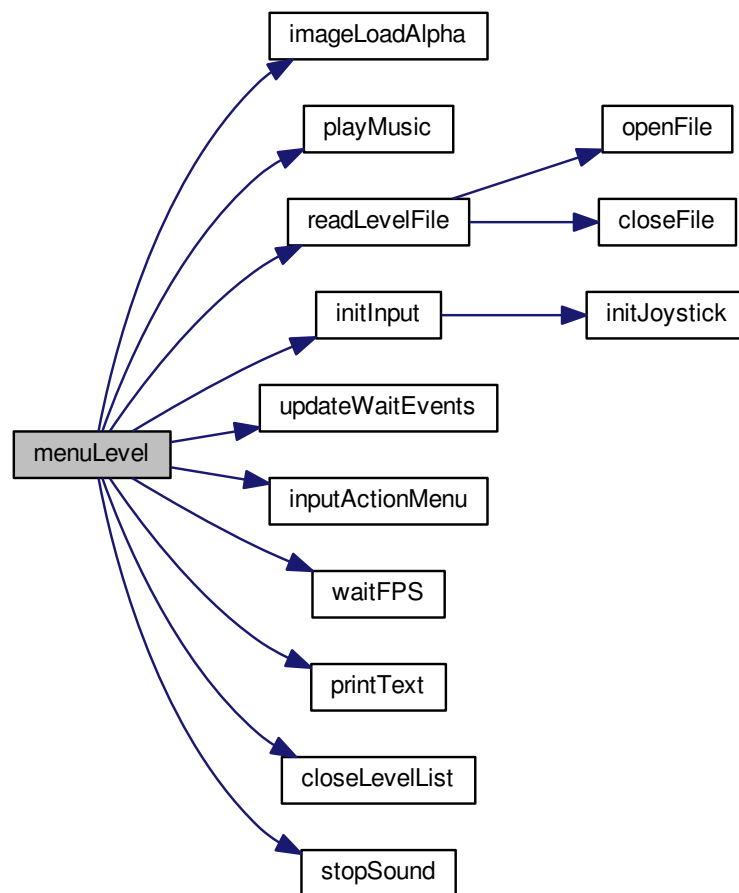
out	<i>screen</i>	game screen
out	<i>level_name</i>	the name of the level we will want to launch
in, out	<i>sound_sys</i>	the sound system
in, out	<i>player</i>	the player structure

in, out	<i>go</i>	the soft main loop validation
in, out	<i>player_name</i>	the player name
in	<i>nb_lvl</i>	the number of level
in, out	<i>in</i>	the input structure

Returns

1 if a level has been choosen, 0 if not

Here is the call graph for this function:

**2.22 menu_option.c File Reference**

contains the option menu functions

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

Functions

- int `optionMenu` (SDL_Surface *screen, int *go, Sound *sound_sys, SDLKey *kc, Input *in)

- void `soundOptions` (SDL_Surface *screen, int *go, Sound *sound_sys, Input *in)
- void `keyBoardOptions` (SDL_Surface *screen, int *go, SDLKey *kc, Input *in, char *player_name)
- void `chooseKey` (SDL_Surface *screen, Input *in, char *action, SDLKey *kc, int nb)

2.22.1 Detailed Description

contains the option menu functions

Author

X.COPONET

Date

2014-04-27

2.22.2 Function Documentation

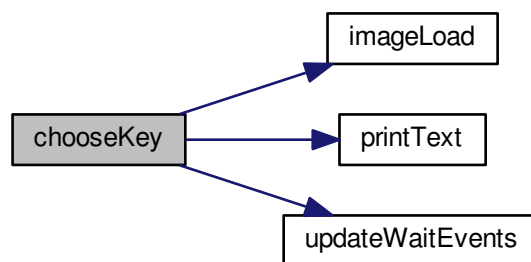
2.22.2.1 void chooseKey (SDL_Surface * screen, Input * in, char * action, SDLKey * kc, int nb)

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:



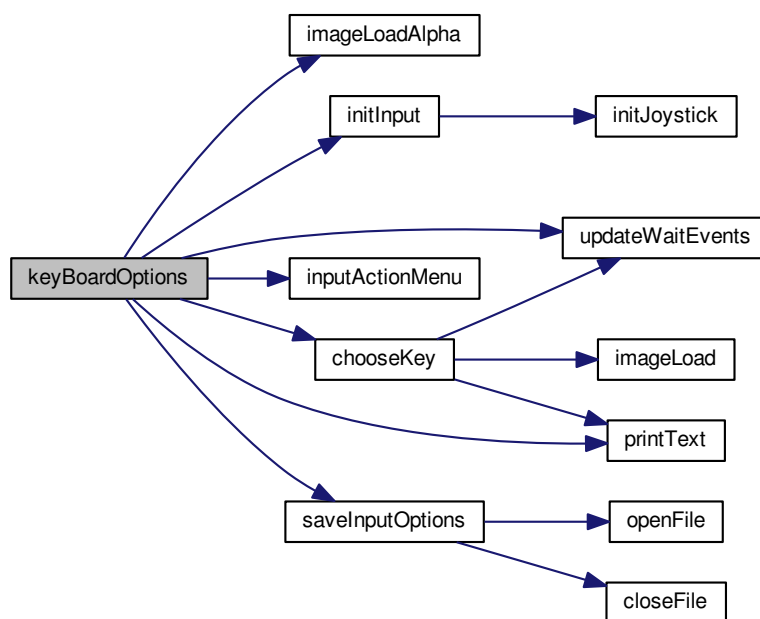
2.22.2.2 void keyBoardOptions (SDL_Surface * screen, int * go, SDLKey * kc, Input * in, char * player_name)

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 structure
in, out	<i>in</i>	the input structure
in	<i>player_name</i>	the current player name

Here is the call graph for this function:



2.22.2.3 int optionMenu (SDL_Surface * screen, int * go, Sound * sound_sys, SDLKey * kc, Input * in)

print the option menu on the screen

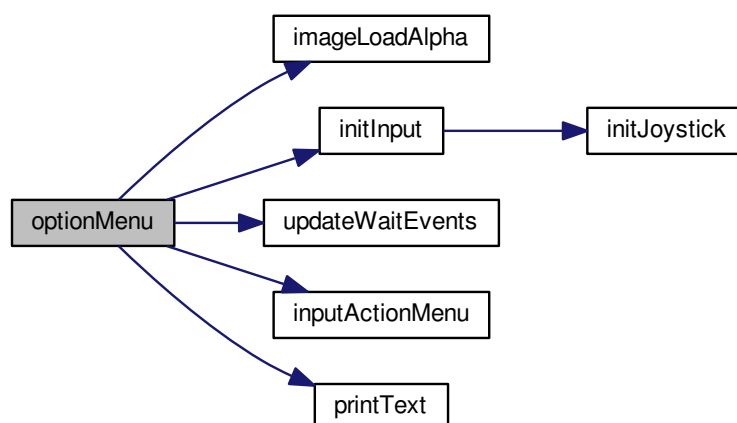
Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
in, out	<i>sound_sys</i>	sound system
in, out	<i>kc</i>	the keyboard configuration structure
in, out	<i>in</i>	the input structure

Returns

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

Here is the call graph for this function:



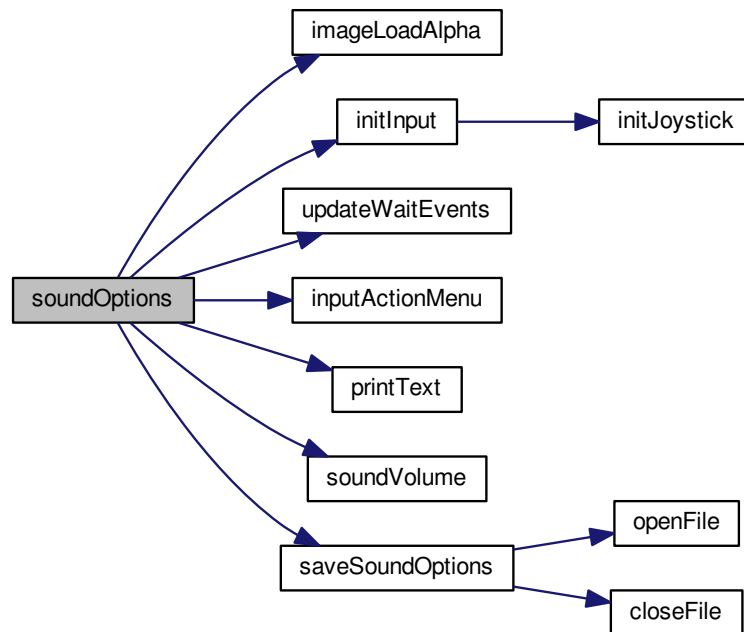
2.22.2.4 void soundOptions (SDL_Surface * screen, int * go, Sound * sound_sys, Input * in)

print the sound 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>sound_sys</i>	sound system
in, out	<i>in</i>	the input structure
in, out	<i>sound_sys</i>	the sound system

Here is the call graph for this function:



2.23 menu_option.h File Reference

```

#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 "sound.h"
#include "share.h"
#include "image.h"
#include "input.h"
#include "option.h"

```

Functions

- int [optionMenu](#) (SDL_Surface *screen, int *go, [Sound](#) *sound_sys, SDLKey *kc, [Input](#) *in)
- void [soundOptions](#) (SDL_Surface *screen, int *go, [Sound](#) *sound_sys, [Input](#) *in)
- void [keyBoardOptions](#) (SDL_Surface *screen, int *go, SDLKey *kc, [Input](#) *in, char *player_name)
- void [chooseKey](#) (SDL_Surface *screen, [Input](#) *in, char *action, SDLKey *kc, int nb)

2.23.1 Detailed Description

[menu_option.h](#)

Author

X.COPONET

Date

2014-04-27

2.23.2 Function Documentation

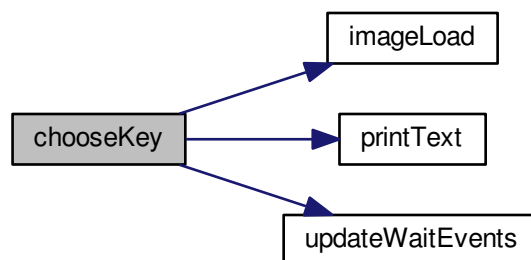
2.23.2.1 void chooseKey (SDL_Surface * *screen*, Input * *in*, char * *action*, SDLKey * *kc*, int *nb*)

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:

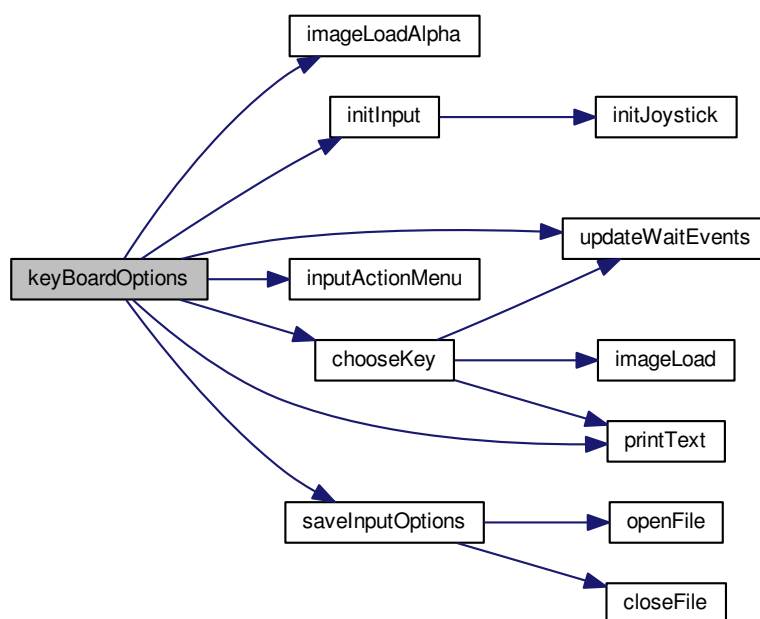
2.23.2.2 void keyBoardOptions (SDL_Surface * *screen*, int * *go*, SDLKey * *kc*, Input * *in*, char * *player_name*)

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 structure
in, out	<i>in</i>	the input structure
in	<i>player_name</i>	the current player name

Here is the call graph for this function:



2.23.2.3 int optionMenu (SDL_Surface * screen, int * go, Sound * sound_sys, SDLKey * kc, Input * in)

print the option menu on the screen

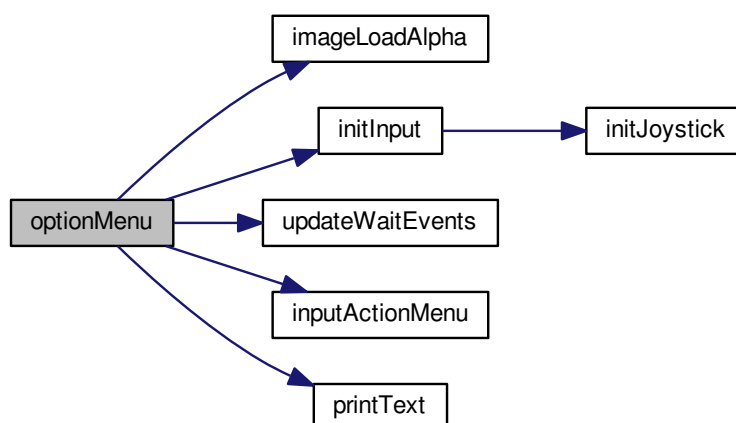
Parameters

out	<i>screen</i>	the game screen
in, out	<i>go</i>	main loop validation
in, out	<i>sound_sys</i>	sound system
in, out	<i>kc</i>	the keyboard configuration structure
in, out	<i>in</i>	the input structure

Returns

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

Here is the call graph for this function:



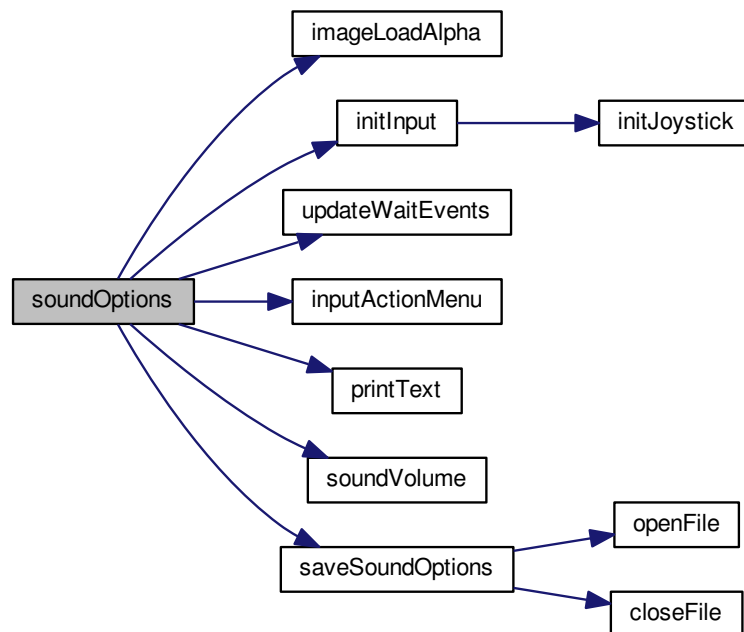
2.23.2.4 void soundOptions (SDL_Surface * screen, int * go, Sound * sound_sys, Input * in)

print the sound 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>sound_sys</i>	sound system
in, out	<i>in</i>	the input structure
in, out	<i>sound_sys</i>	the sound system

Here is the call graph for this function:



2.24 mobile_platform.c File Reference

contains the functions to deal with the mobile platforms

```
#include "mobile_platform.h"
```

Functions

- void [initPlatformSet](#) ([platformSet](#) *ps)
- void [createPlatform](#) ([platformSet](#) *ps, int x1, int y1, int x2, int y2)
- void [blitPlatform](#) (SDL_Surface *screen, [platformSet](#) *ps, [Map](#) *m)
- void [movePlatform](#) ([Character](#) *c, [platformSet](#) *ps, [list](#) *l, [Map](#) *m)
- void [moveOnePlatform](#) ([Character](#) *c, [platform](#) *p, [list](#) *l, int nb, [Map](#) *m)
- int [collisionPlatform](#) ([Character](#) *c, [platformSet](#) *ps, SDL_Rect futureLocation)
- void [freePlatformSet](#) ([platformSet](#) *ps)
- void [platformMap](#) ([platformSet](#) *ps, SDL_Rect array[], SDL_Rect mark, int vert)

2.24.1 Detailed Description

contains the functions to deal with the mobile platforms

Author

X.COPONET

Date

2014-05-01

2.24.2 Function Documentation

2.24.2.1 void blitPlatform (SDL_Surface * *screen*, platformSet * *ps*, Map * *m*)

blit the platforms on the game screen

Parameters

in, out	<i>screen</i>	game screen
in, out	<i>ps</i>	the platform set
in	<i>m</i>	the current level map

2.24.2.2 int collisionPlatform (Character * *c*, platformSet * *ps*, SDL_Rect *futureLocation*)

determine if there is a collision between the player and a mobile platform and deals with

Parameters

in, out	<i>c</i>	the player
in, out	<i>ps</i>	the platform set
in	<i>futureLocation</i>	the tryMovement variable to test the future position

Returns

1 if there is a collision, 0 if not

Here is the call graph for this function:

2.24.2.3 void createPlatform (platformSet * *ps*, int *x1*, int *y1*, int *x2*, int *y2*)

creates of new platform and adds it to the platform set

Parameters

in, out	<i>ps</i>	the platform set
in	<i>x1</i>	the x low limit for displacement
in	<i>x2</i>	the x high limit for displacement
in	<i>y1</i>	the y low limit for displacement
in	<i>y2</i>	the y high limit for displacement

Here is the call graph for this function:



2.24.2.4 void freePlatformSet (platformSet * *ps*)

free all the platforms

Parameters

in, out	<i>ps</i>	the platform set
---------	-----------	------------------

2.24.2.5 void initPlatformSet (platformSet * *ps*)

initialize a platform set

Parameters

in	<i>ps</i>	the platform set to be initialized
----	-----------	------------------------------------

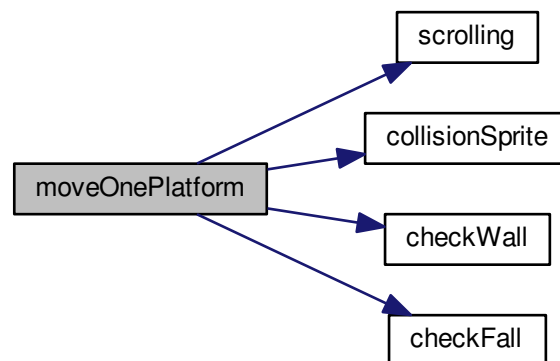
2.24.2.6 void moveOnePlatform (Character * *c*, platform * *p*, list * *l*, int *nb*, Map * *m*)

moves one platforms

Parameters

in, out	<i>c</i>	the player
in, out	<i>p</i>	the platform
in, out	<i>l</i>	the enemy list
in	<i>nb</i>	the number of the platform which is moved
in	<i>m</i>	the game map

Here is the call graph for this function:



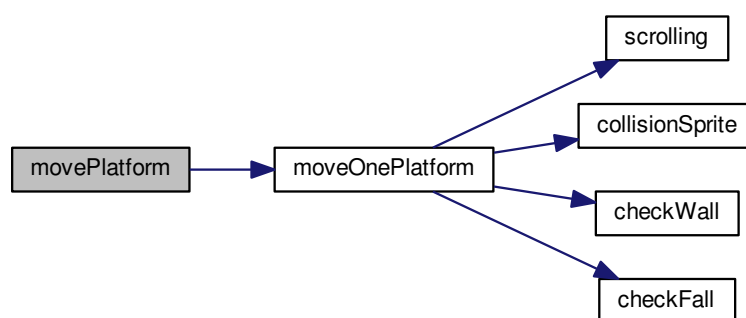
2.24.2.7 `void movePlatform (Character * c, platformSet * ps, list * l, Map * m)`

moves all the platforms

Parameters

in, out	<i>c</i>	the player
in, out	<i>ps</i>	the platform set
in, out	<i>l</i>	the enemy list
in	<i>m</i>	the game map

Here is the call graph for this function:



2.24.2.8 `void platformMap (platformSet * ps, SDL_Rect array[], SDL_Rect mark, int vert)`

takes a limit mark for a vertical displacement platform and creates a new platform if finds another limit mark which match it, stocks it if doesn't find another limit mark

Parameters

out	<i>ps</i>	the platform set
in, out	<i>array</i>	the array that stocks the limit marks
in	<i>mark</i>	the mark which has to be dealt with
in	<i>vert</i>	indicates if vertical movement(1) platform or horizontal (0)

Here is the call graph for this function:



2.25 mobile_platform.h File Reference

[mobile_platform.c](#) header

```
#include "structures.h"
#include "image.h"
#include "character.h"
```

Functions

- void [initPlatformSet](#) ([platformSet](#) *ps)
- void [createPlatform](#) ([platformSet](#) *ps, int x1, int y1, int x2, int y2)
- void [blitPlatform](#) (SDL_Surface *screen, [platformSet](#) *ps, [Map](#) *m)
- void [movePlatform](#) ([Character](#) *c, [platformSet](#) *ps, [list](#) *l, [Map](#) *m)
- void [moveOnePlatform](#) ([Character](#) *c, [platform](#) *p, [list](#) *l, int nb, [Map](#) *m)
- int [collisionPlatform](#) ([Character](#) *c, [platformSet](#) *ps, SDL_Rect futureLocation)
- void [freePlatformSet](#) ([platformSet](#) *ps)
- void [platformMap](#) ([platformSet](#) *ps, SDL_Rect array[], SDL_Rect mark, int vert)

2.25.1 Detailed Description

[mobile_platform.c](#) header

Author

X.COPONET

Date

2014-05-01

2.25.2 Function Documentation

2.25.2.1 void blitPlatform (SDL_Surface * screen, platformSet * ps, Map * m)

blit the platforms on the game screen

Parameters

in, out	<i>screen</i>	game screen
in, out	<i>ps</i>	the platform set
in	<i>m</i>	the current level map

2.25.2.2 int collisionPlatform (Character * c, platformSet * ps, SDL_Rect futureLocation)

determine if there is a collision between the player and a mobile platform and deals with

Parameters

in, out	<i>c</i>	the player
in, out	<i>ps</i>	the platform set
in	<i>futureLocation</i>	the tryMovement variable to test the future position

Returns

1 if there is a collision, 0 if not

Here is the call graph for this function:

**2.25.2.3 void createPlatform (platformSet * ps, int x1, int y1, int x2, int y2)**

creates of new platform and adds it to the platform set

Parameters

in, out	<i>ps</i>	the platform set
in	<i>x1</i>	the x low limit for deplacement
in	<i>x2</i>	the x high limit for deplacement
in	<i>y1</i>	the y low limit for deplacement
in	<i>y2</i>	the y high limit for deplacement

Here is the call graph for this function:

**2.25.2.4 void freePlatformSet (platformSet * ps)**

free all the platforms

Parameters

in, out	<i>ps</i>	the platform set
---------	-----------	------------------

2.25.2.5 void initPlatformSet (platformSet * *ps*)

initialize a platform set

Parameters

in	<i>ps</i>	the platform set to be initialized
----	-----------	------------------------------------

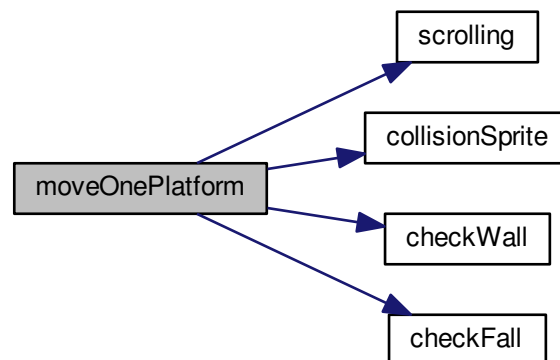
2.25.2.6 void moveOnePlatform (Character * *c*, platform * *p*, list * *l*, int *nb*, Map * *m*)

moves one platforms

Parameters

in, out	<i>c</i>	the player
in, out	<i>p</i>	the platform
in, out	<i>l</i>	the enemy list
in	<i>nb</i>	the number of the platform which is moved
in	<i>m</i>	the game map

Here is the call graph for this function:

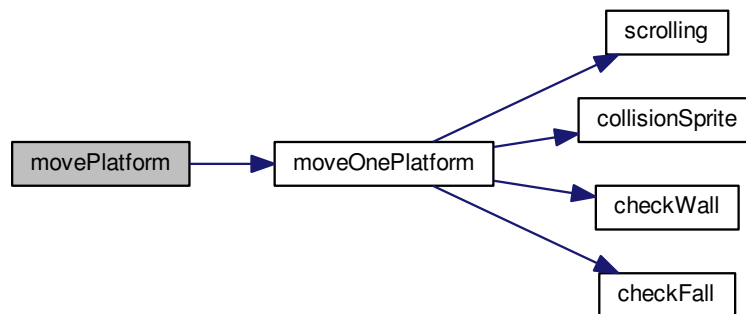
2.25.2.7 void movePlatform (Character * *c*, platformSet * *ps*, list * *l*, Map * *m*)

moves all the platforms

Parameters

in, out	<i>c</i>	the player
in, out	<i>ps</i>	the platform set
in, out	<i>l</i>	the enemy list
in	<i>m</i>	the game map

Here is the call graph for this function:



2.25.2.8 void platformMap (platformSet * ps, SDL_Rect array[], SDL_Rect mark, int vert)

takes a limit mark for a vertical displacement platform and creates a new platform if finds another limit mark which match it, stocks it if doesn't find another limit mark

Parameters

out	<i>ps</i>	the platform set
in, out	<i>array</i>	the array that stocks the limit marks
in	<i>mark</i>	the mark which has to be dealt with
in	<i>vert</i>	indicates if vertical movement(1) platform or horizontal (0)

Here is the call graph for this function:



2.26 option.c File Reference

contains the funtions that manipulate the options

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

Functions

- void [loadSoundOptions](#) (char confFile[], [Sound](#) *soundSys)
- void [saveSoundOptions](#) (char confFile[], [Sound](#) *soundSys)
- void [loadInputOptions](#) (char player_name[], [SDLKey](#) *kc, [Input](#) *in)
- void [saveInputOptions](#) (char player_name[], [SDLKey](#) *kc, [Input](#) *in)

2.26.1 Detailed Description

contains the funtions that manipulate the options

Author

X.COPONET

Date

2014-04-28

2.26.2 Function Documentation

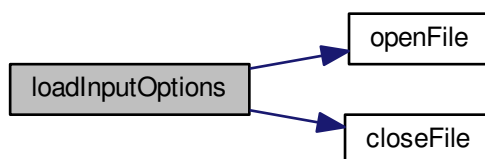
2.26.2.1 void loadInputOptions (char *player_name*[], SDLKey * *kc*, Input * *in*)

load the input options from the player input config file

Parameters

in	<i>player_name</i>	the current player's name
out	<i>kc</i>	the keyboard configuration structure
out	<i>in</i>	the input structure

Here is the call graph for this function:



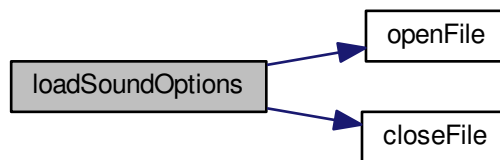
2.26.2.2 void loadSoundOptions (char *confFile*[], Sound * *soundSys*)

load the sound options from the sound config file

Parameters

in	<i>confFile</i>	the config file path
out	<i>soundSys</i>	the sound system

Here is the call graph for this function:



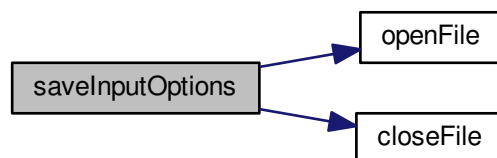
2.26.2.3 void saveInputOptions (char *player_name*[], SDLKey * *kc*, Input * *in*)

save the input options to the player input config file

Parameters

in	<i>player_name</i>	the current player name
out	<i>kc</i>	the keyboard configuration structure
out	<i>in</i>	the input structure

Here is the call graph for this function:



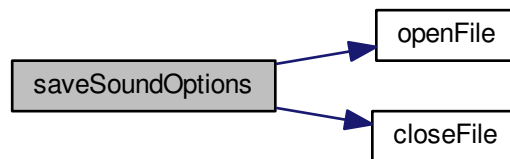
2.26.2.4 void saveSoundOptions (char *confFile*[], Sound * *soundSys*)

save the sound options to the config file

Parameters

in	<i>confFile</i>	the config file path
in	<i>soundSys</i>	the sound system

Here is the call graph for this function:



2.27 option.h File Reference

[option.c](#) header

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

Functions

- void [loadSoundOptions](#) (char confFile[], [Sound](#) *soundSys)
- void [saveSoundOptions](#) (char confFile[], [Sound](#) *soundSys)
- void [loadInputOptions](#) (char player_name[], [SDLKey](#) *kc, [Input](#) *in)
- void [saveInputOptions](#) (char player_name[], [SDLKey](#) *kc, [Input](#) *in)

2.27.1 Detailed Description

[option.c](#) header

Author

Xavier COPONET

Date

2014-04-28

2.27.2 Function Documentation

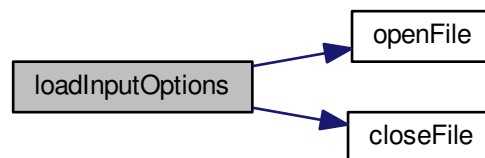
2.27.2.1 void [loadInputOptions](#) (char *player_name*[], [SDLKey](#) * *kc*, [Input](#) * *in*)

load the input options from the player input config file

Parameters

in	<i>player_name</i>	the current player's name
out	<i>kc</i>	the keyboard configuration structure
out	<i>in</i>	the input structure

Here is the call graph for this function:



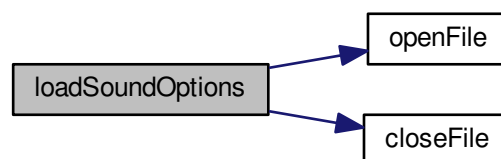
2.27.2.2 void loadSoundOptions (char *confFile*[], Sound * *soundSys*)

load the sound options from the sound config file

Parameters

in	<i>confFile</i>	the config file path
out	<i>soundSys</i>	the sound system

Here is the call graph for this function:



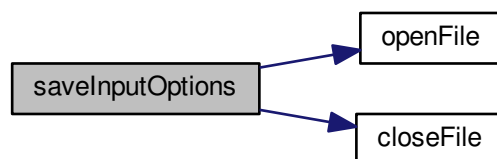
2.27.2.3 void saveInputOptions (char *player_name*[], SDLKey * *kc*, Input * *in*)

save the input options to the player input config file

Parameters

in	<i>player_name</i>	the current player name
out	<i>kc</i>	the keyboard configuration structure
out	<i>in</i>	the input structure

Here is the call graph for this function:



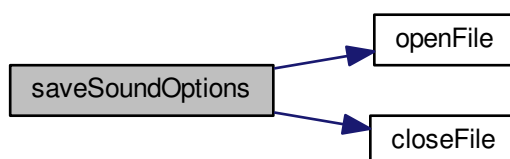
2.27.2.4 void saveSoundOptions (char *confFile*[], Sound * *soundSys*)

save the sound options to the config file

Parameters

in	<i>confFile</i>	the config file path
in	<i>soundSys</i>	the sound system

Here is the call graph for this function:



2.28 player.c File Reference

Management of the player system.

```
#include "player.h"
```

Functions

- int [newPlayer](#) (SDL_Surface *screen, char player_name[[MAX_SIZE_FILE_NAME](#)], Sound *s, int *go)
- void [loadPlayer](#) (char fileSave[[MAX_SIZE_FILE_NAME](#)], char player_name[[MAX_SIZE_FILE_NAME](#)], Player *player)
- int [savePlayer](#) (char fileSave[[MAX_SIZE_FILE_NAME](#)], char player_name[[MAX_SIZE_FILE_NAME](#)], Player *player)
- void [save](#) (SDL_Surface *screen, char fileSave[[MAX_SIZE_FILE_NAME](#)], char player_name[[MAX_SIZE_FILE_NAME](#)], Player *player, int *go)
- void [deletePlayer](#) (SDL_Surface *screen, char fileSave[[MAX_SIZE_FILE_NAME](#)], char player_name[[MAX_SIZE_FILE_NAME](#)])

2.28.1 Detailed Description

Management of the player system.

Author

Glenn HERROU

Date

06/05/14

Version

1.0

2.28.2 Function Documentation

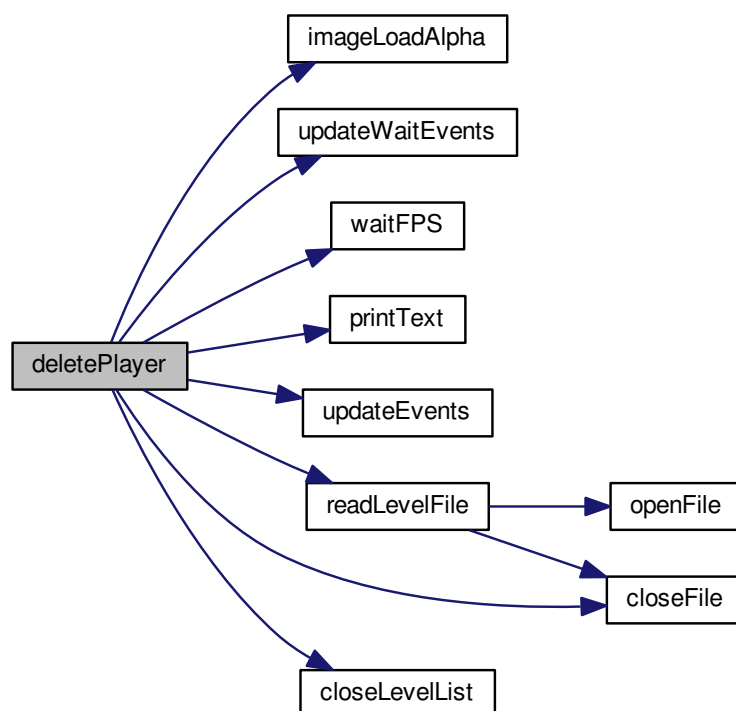
2.28.2.1 `void deletePlayer (SDL_Surface * screen, char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME])`

Delete the current player in the player list.

Parameters

in, out	<i>screen</i>	The screen of the game
in, out	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player

Here is the call graph for this function:



2.28.2.2 void loadPlayer (char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player * player)

Load the progression of the given player from the binary file named fileSave

Parameters

in	fileSave	The path to the binary file containing the progression of each player
in	player_name	The name of the current player
out	player	The player structure where the progression will be loaded

2.28.2.3 int newPlayer (SDL_Surface * screen, char player_name[MAX_SIZE_FILE_NAME], Sound * s, int * go)

Display the interface to create a new player

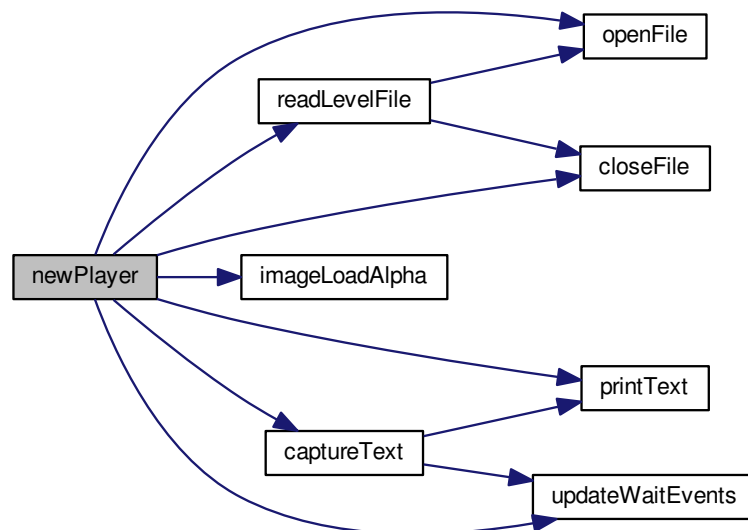
Parameters

in, out	screen	The screen of the game
out	player_name	The name of the new player
out	s	the sound system
out	go	the main loop validation

Returns

1 if a new player has been created, 0 otherwise

Here is the call graph for this function:



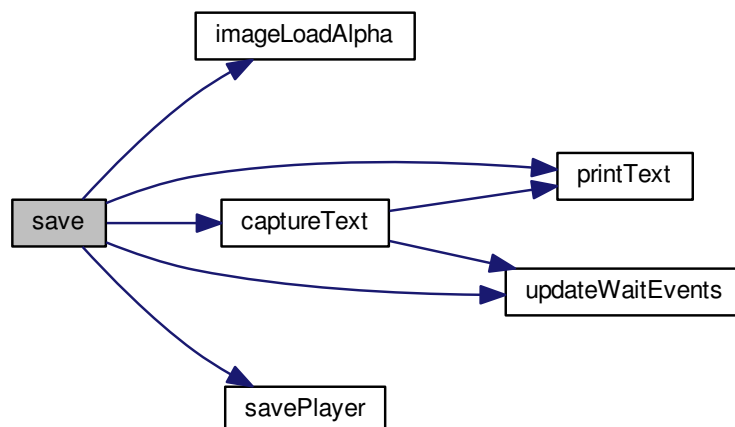
2.28.2.4 void save (SDL_Surface * screen, char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player * player, int * go)

Display the interface to save the player progression

Parameters

in, out	<i>screen</i>	The screen of the game
in	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player
out	<i>player</i>	The player structure where the progression is stored
out	<i>go</i>	The main loop validation

Here is the call graph for this function:



2.28.2.5 `int savePlayer (char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player * player)`

Save the progression of the given player in the binary file named fileSave

Parameters

in	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player
out	<i>player</i>	The player structure where the progression is stored

2.29 player.h File Reference

[player.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 "structures.h"
#include "file_level.h"
#include "share.h"
#include "text.h"
#include "sound.h"
#include "image.h"
#include "input.h"
```

Functions

- int [newPlayer](#) (SDL_Surface *screen, char player_name[MAX_SIZE_FILE_NAME], Sound *s, int *go)
- void [loadPlayer](#) (char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player *player)
- int [savePlayer](#) (char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player *player)
- void [save](#) (SDL_Surface *screen, char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player *player, int *go)
- void [deletePlayer](#) (SDL_Surface *screen, char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME])

2.29.1 Detailed Description

[player.c](#) header

Author

Glenn HERROU

Date

06/05/14

Version

1.0

2.29.2 Function Documentation

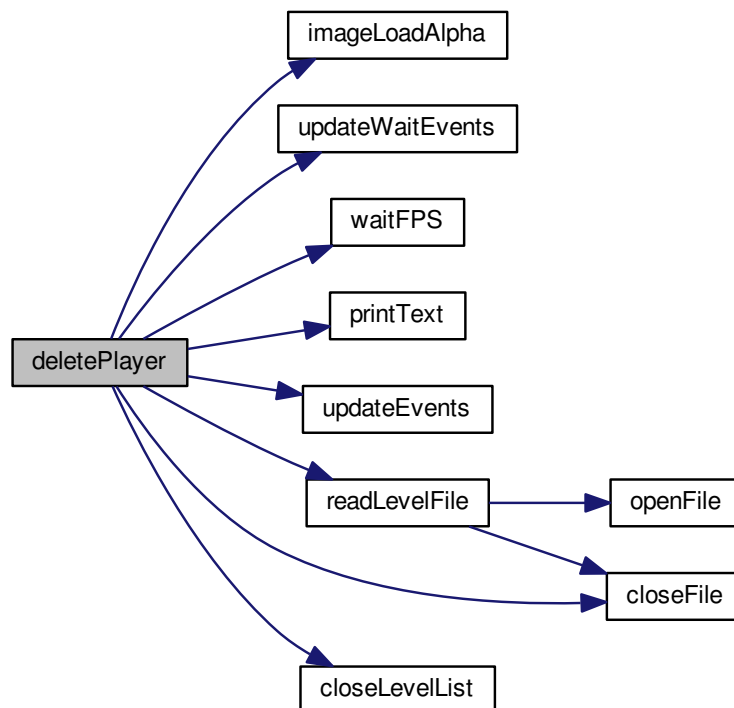
2.29.2.1 void [deletePlayer](#) (SDL_Surface * screen, char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME])

Delete the current player in the player list.

Parameters

in, out	<i>screen</i>	The screen of the game
in, out	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player

Here is the call graph for this function:



2.29.2.2 `void loadPlayer (char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player * player)`

Load the progression of the given player from the binary file named fileSave

Parameters

in	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player
out	<i>player</i>	The player structure where the progression will be loaded

2.29.2.3 `int newPlayer (SDL_Surface * screen, char player_name[MAX_SIZE_FILE_NAME], Sound * s, int * go)`

Display the interface to create a new player

Parameters

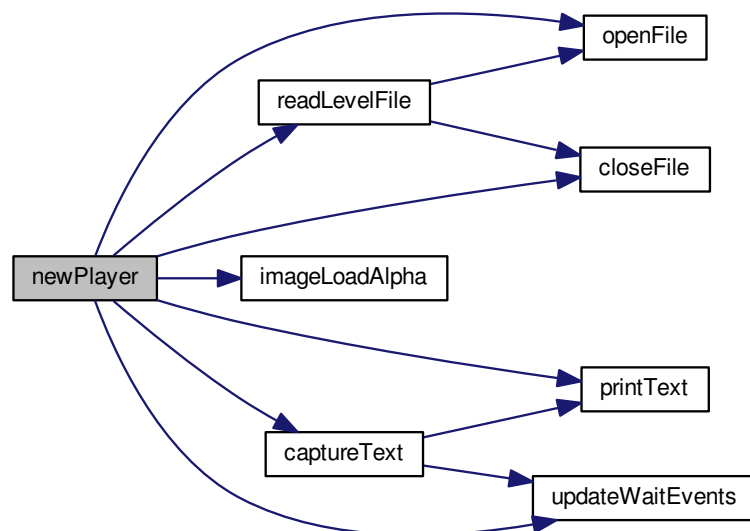
in, out	<i>screen</i>	The screen of the game
---------	---------------	------------------------

out	<i>player_name</i>	The name of the new player
out	<i>s</i>	the sound system
out	<i>go</i>	the main loop validation

Returns

1 if a new player has been created, 0 otherwise

Here is the call graph for this function:



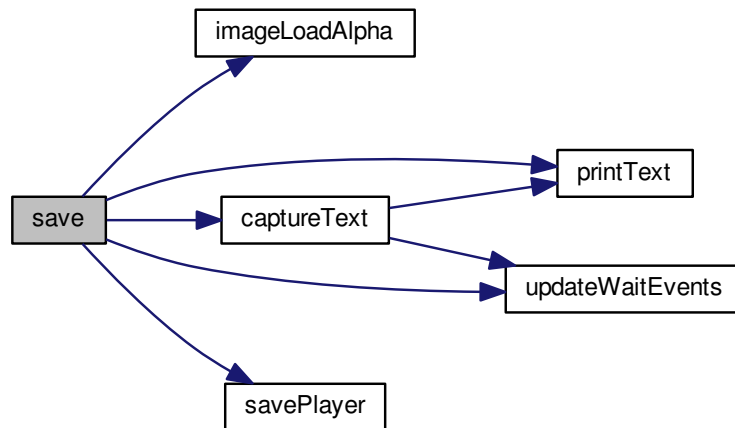
2.29.2.4 void save (SDL_Surface * *screen*, char *fileSave*[MAX_SIZE_FILE_NAME], char *player_name*[MAX_SIZE_FILE_NAME], Player * *player*, int * *go*)

Display the interface to save the player progression

Parameters

in, out	<i>screen</i>	The screen of the game
in	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player
out	<i>player</i>	The player structure where the progression is stored
out	<i>go</i>	The main loop validation

Here is the call graph for this function:



2.29.2.5 `int savePlayer (char fileSave[MAX_SIZE_FILE_NAME], char player_name[MAX_SIZE_FILE_NAME], Player * player)`

Save the progression of the given player in the binary file named fileSave

Parameters

in	<i>fileSave</i>	The path to the binary file containing the progression of each player
in	<i>player_name</i>	The name of the current player
out	<i>player</i>	The player structure where the progression is stored

2.30 projectile.c File Reference

contains the functions to deal with the projectiles

```
#include "projectile.h"
```

Functions

- void `initProjSet` (`projectileSet *projSet`)
- void `freeProjectileSet` (`projectileSet *ps`)
- void `createProjectile` (`projectileSet *projSet`, `char *pathSprite`, `int dir`, `int x`, `int y`, `int fromNPC`)
- void `deleteProjectile` (`projectileSet *ps`, `int nb`)
- void `blitProjectile` (`SDL_Surface *screen`, `projectileSet *ps`, `Map *m`)
- void `moveProjectiles` (`Character *c`, `Map *m`, `projectileSet *ps`, `list *enemyList`)
- void `moveOneProjectile` (`Character *c`, `Map *m`, `projectileSet *ps`, `list *l`, `int nb`)

2.30.1 Detailed Description

contains the functions to deal with the projectiles

Author

X.COPONET

Date

2014-05-08

2.30.2 Function Documentation

2.30.2.1 void blitProjectile (SDL_Surface * *screen*, projectileSet * *ps*, Map * *m*)

blit the projectiles on the game screen

Parameters

in, out	<i>screen</i>	game screen
in, out	<i>ps</i>	the projectile set
in	<i>m</i>	the current level map

2.30.2.2 void createProjectile (projectileSet * *projSet*, char * *pathSprite*, int *dir*, int *x*, int *y*, int *fromNPC*)

creates a projectile and adds it to the projectile set

Parameters

in, out	<i>projSet</i>	the projectile set
in, out	<i>pathSprite</i>	the path for the sprite
in	<i>dir</i>	the projectile's direction
in	<i>x</i>	the start absciss coordinate of the projectile
in	<i>y</i>	the start ordinate coordinate of the projectile
in	<i>fromNPC</i>	indicates if from npc or not

Here is the call graph for this function:

2.30.2.3 void deleteProjectile (projectileSet * *ps*, int *nb*)

delete a projectile

Parameters

out	<i>ps</i>	the projectile Set
in	<i>nb</i>	the number of the projectile which has to be deleted

2.30.2.4 void freeProjectileSet (projectileSet * *ps*)

free all the projectiles

Parameters

in, out	<i>ps</i>	the projectile set
---------	-----------	--------------------

2.30.2.5 void initProjSet (projectileSet * *projSet*)

initialize a projectile set

Parameters

out	<i>projSet</i>	the projectile set to be initialized
-----	----------------	--------------------------------------

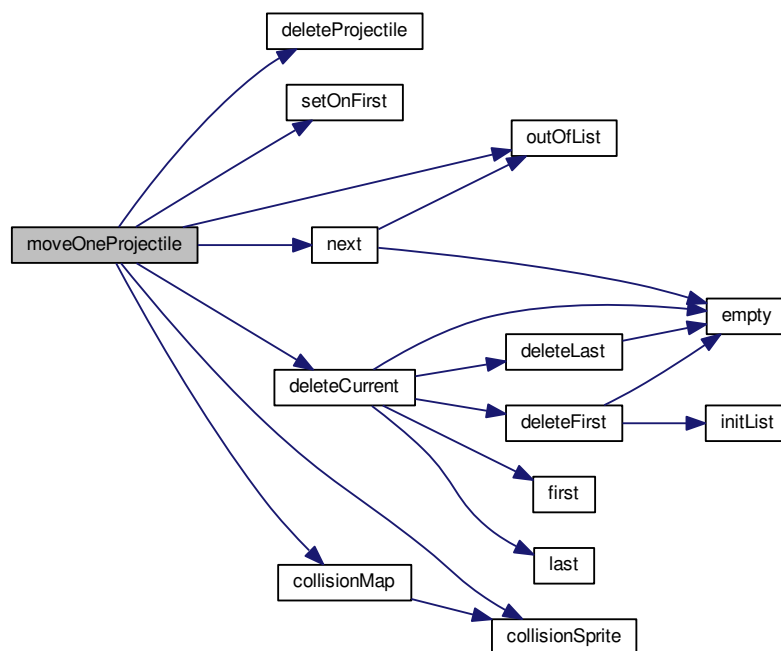
2.30.2.6 void moveOneProjectile (Character * *c*, Map * *m*, projectileSet * *ps*, list * *l*, int *nb*)

moves one projectile

Parameters

in, out	<i>c</i>	the player
in, out	<i>m</i>	the game map
in, out	<i>ps</i>	the projectileSet
in, out	<i>l</i>	the enemy list
in	<i>nb</i>	the number of the projectile which is moved

Here is the call graph for this function:

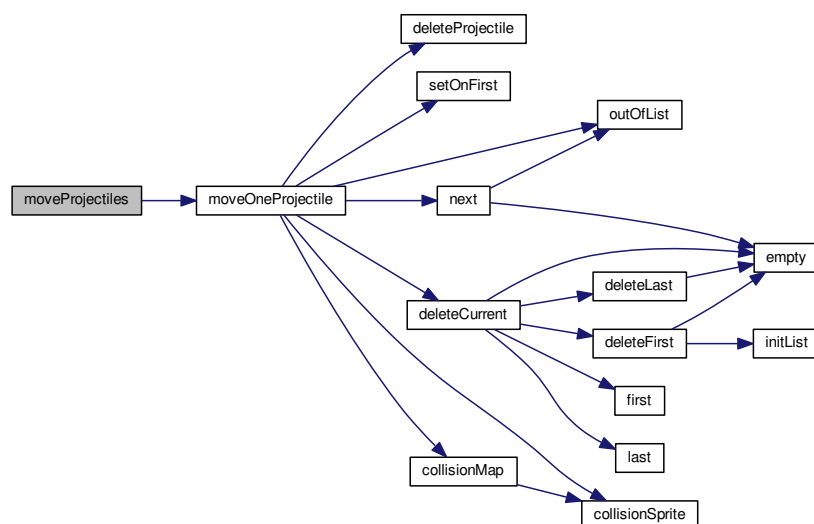
2.30.2.7 void moveProjectiles (Character * *c*, Map * *m*, projectileSet * *ps*, list * *enemyList*)

moves all the projectiles

Parameters

in, out	<i>c</i>	the player
in, out	<i>m</i>	the game map
in, out	<i>ps</i>	the projectile set
in, out	<i>enemyList</i>	the enemy list

Here is the call graph for this function:



2.31 projectile.h File Reference

projectile.c header

```

#include "structures.h"
#include "image.h"
#include "enemies.h"
#include <errno.h>

```

Functions

- void `initProjSet` (`projectileSet` *projSet)
- void `freeProjectileSet` (`projectileSet` *ps)
- void `createProjectile` (`projectileSet` *projSet, char *pathSprite, int dir, int x, int y, int fromNPC)
- void `deleteProjectile` (`projectileSet` *ps, int nb)
- void `blitProjectile` (SDL_Surface *screen, `projectileSet` *ps, `Map` *m)
- void `moveProjectiles` (`Character` *c, `Map` *m, `projectileSet` *ps, `list` *enemyList)
- void `moveOneProjectile` (`Character` *c, `Map` *m, `projectileSet` *ps, `list` *l, int nb)

2.31.1 Detailed Description

projectile.c header

Author

X.COPONET

Date

2014-05-08

2.31.2 Function Documentation

2.31.2.1 void blitProjectile (SDL_Surface * *screen*, projectileSet * *ps*, Map * *m*)

blit the projectiles on the game screen

Parameters

in, out	<i>screen</i>	game screen
in, out	<i>ps</i>	the projectile set
in	<i>m</i>	the current level map

2.31.2.2 void createProjectile (projectileSet * *projSet*, char * *pathSprite*, int *dir*, int *x*, int *y*, int *fromNPC*)

creates a projectile and adds it to the projectile set

Parameters

in, out	<i>projSet</i>	the projectile set
in, out	<i>pathSprite</i>	the path for the sprite
in	<i>dir</i>	the projectile's direction
in	<i>x</i>	the start absciss coordinate of the projectile
in	<i>y</i>	the start ordinate coordinate of the projectile
in	<i>fromNPC</i>	indicates if from npc or not

Here is the call graph for this function:

2.31.2.3 void deleteProjectile (projectileSet * *ps*, int *nb*)

delete a projectile

Parameters

out	<i>ps</i>	the projectile Set
in	<i>nb</i>	the number of the projectile which has to be deleted

2.31.2.4 void freeProjectileSet (projectileSet * *ps*)

free all the projectiles

Parameters

in, out	<i>ps</i>	the projectile set
---------	-----------	--------------------

2.31.2.5 void initProjSet (projectileSet * *projSet*)

initialize a projectile set

Parameters

out	<i>projSet</i>	the projectile set to be initialized
-----	----------------	--------------------------------------

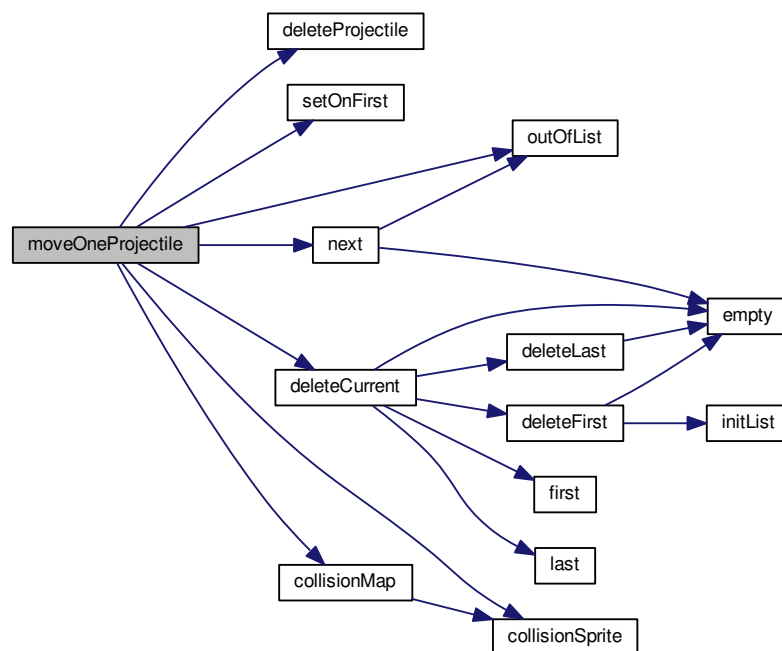
2.31.2.6 void moveOneProjectile (Character * *c*, Map * *m*, projectileSet * *ps*, list * *l*, int *nb*)

moves one projectile

Parameters

in, out	<i>c</i>	the player
in, out	<i>m</i>	the game map
in, out	<i>ps</i>	the projectileSet
in, out	<i>l</i>	the enemy list
in	<i>nb</i>	the number of the projectile which is moved

Here is the call graph for this function:

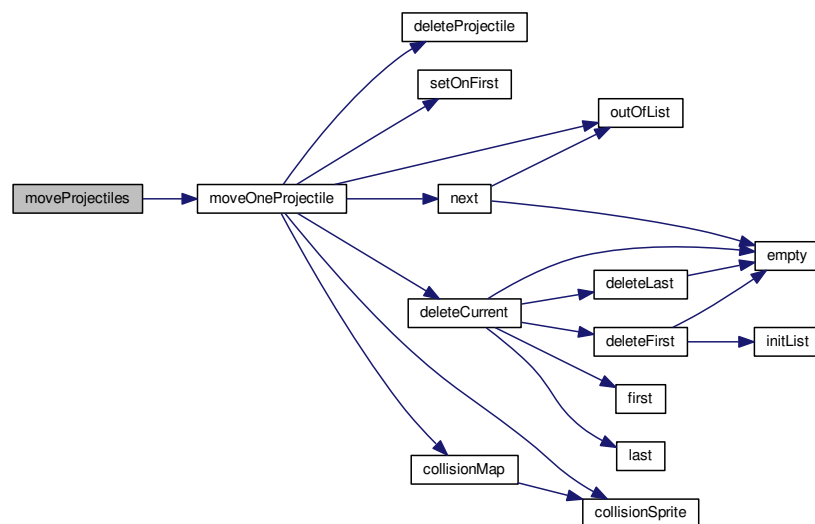
2.31.2.7 void moveProjectiles (Character * *c*, Map * *m*, projectileSet * *ps*, list * *enemyList*)

moves all the projectiles

Parameters

in, out	<i>c</i>	the player
in, out	<i>m</i>	the game map
in, out	<i>ps</i>	the projectile set
in, out	<i>enemyList</i>	the enemy list

Here is the call graph for this function:

**2.32 share.c File Reference**

Management of FPS rate.

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

Functions

- void [waitFPS](#) (int *previous_time, int *current_time)

2.32.1 Detailed Description

Management of FPS rate.

Author

Remi BERTHO

Date

15/03/14

Version

1.0

2.32.2 Function Documentation

2.32.2.1 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

2.33 share.h File Reference

share.c header

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

Functions

- void [waitFPS](#) (int *previous_time, int *current_time)

2.33.1 Detailed Description

share.c header

Author

Remi BERTHO

Date

15/03/14

Version

1.0

2.33.2 Function Documentation

2.33.2.1 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

2.34 sound.c File Reference

contains the sound playing function

```
#include "sound.h"
```

Functions

- `Sound * createSound (void)`
- `void playMusic (char *file, Sound *s)`
- `void playShortSound (char *file, Sound *s)`
- `void freeSound (Sound *s)`
- `void stopSound (Sound *s, int chan)`
- `void soundVolume (Sound *s, int chan, float volume)`

2.34.1 Detailed Description

contains the sound playing function

Author

Xavier COPONET

Date

2014-02-27

2.34.2 Function Documentation

2.34.2.1 `sound * createSound (void)`

create a sound structure

Returns

the sound structure

2.34.2.2 `void freeSound (Sound * s)`

release the sound

Parameters

out	s	the sound
-----	---	-----------

2.34.2.3 `void playMusic (char * file, Sound * s)`

play a long sound file (music)

Parameters

in	file	the sound file to be played
out	s	the sound system we manipulate

2.34.2.4 `void playShortSound (char * file, Sound * s)`

play a short sound file (effect sound)

Parameters

in	file	the sound file to be played
----	------	-----------------------------

out	s	the sound system we manipulate
-----	---	--------------------------------

2.34.2.5 void soundVolume (Sound * s, int chan, float volume)

set the sound volume

Parameters

out	s	the sound system
in	chan	the channel which the volume's has to be changed
in	volume	the sound volume : [0.0 : no sound ; 1.0 (default) max power]

2.34.2.6 void stopSound (Sound * s, int chan)

stop the sound

Parameters

out	s	the sound system
in	chan	the channel which has to be stoped

2.35 sound.h File Reference

header de [sound.c](#)

```
#include <FMOD/fmod.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
```

Data Structures

- struct [Sound](#)

Functions

- [Sound * createSound](#) (void)
- void [playMusic](#) (char *file, [Sound *s](#))
- void [playShortSound](#) (char *file, [Sound *s](#))
- void [freeSound](#) ([Sound *s](#))
- void [stopSound](#) ([Sound *s](#), int chan)
- void [soundVolume](#) ([Sound *s](#), int chan, float volume)

2.35.1 Detailed Description

header de [sound.c](#)

Author

Xavier COPONET

Date

2014-02-27

2.35.2 Function Documentation

2.35.2.1 **Sound*** createSound (void)

create a sound structure

Returns

the sound structure

2.35.2.2 void freeSound (**Sound *** s)

release the sound

Parameters

out	s	the sound
-----	---	-----------

2.35.2.3 void playMusic (char * *file*, **Sound *** s)

play a long sound file (music)

Parameters

in	<i>file</i>	the sound file to be played
out	s	the sound system we manipulate

2.35.2.4 void playShortSound (char * *file*, **Sound *** s)

play a short sound file (effect sound)

Parameters

in	<i>file</i>	the sound file to be played
out	s	the sound system we manipulate

2.35.2.5 void soundVolume (**Sound *** s, int *chan*, float *volume*)

set the sound volume

Parameters

out	s	the sound system
in	<i>chan</i>	the channel which the volume's has to be changed
in	<i>volume</i>	the sound volume : [0.0 : no sound ; 1.0 (default) max power]

2.35.2.6 void stopSound (**Sound *** s, int *chan*)

stop the sound

Parameters

out	s	the sound system
in	<i>chan</i>	the channel which has to be stoped

2.36 structures.h File Reference

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


Data Structures

- struct [Character](#)
- struct [Player](#)
- struct [node](#)
- struct [list](#)
- struct [platform](#)
- struct [platformSet](#)
- struct [projectile](#)
- struct [projectileSet](#)

Typedefs

- typedef struct [node](#) **node**

2.36.1 Detailed Description

contain the definition of some structures

Author

X.COPONET

Date

2014-04-15

2.37 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, int *ret)

2.37.1 Detailed Description

Management of the display of text on the screen.

Author

Xavier COPONET, Glenn HERROU

Date

2014-04-27

2.37.2 Function Documentation

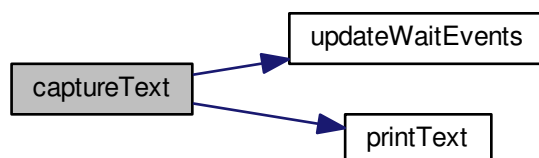
2.37.2.1 void captureText (SDL_Surface * *screen*, SDL_Rect *posText*, char * *text*, int *text_length*, int *r*, int *g*, int *b*, char * *font*, int *text_size*, int * *go*, int * *ret*)

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
in, out	<i>ret</i>	The press return indicator

Here is the call graph for this function:



2.37.2.2 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)

2.38 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, int *ret)

2.38.1 Detailed Description

[text.c](#) header

Author

Xavier COPONET, Glenn HERROU

Date

2014-04-27

2.38.2 Function Documentation

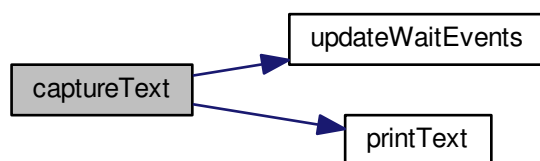
2.38.2.1 void [captureText](#) (SDL_Surface * *screen*, SDL_Rect *posText*, char * *text*, int *text_length*, int *r*, int *g*, int *b*, char * *font*, int *text_size*, int * *go*, int * *ret*)

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
in, out	<i>ret</i>	The press return indicator

Here is the call graph for this function:



2.38.2.2 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

ABS

character.h, 19

axes

Input, 4

BUFFER_SIZE

file_level.h, 56

background

Level, 5

blinkText

menu.c, 85

menu.h, 89

blitCharacter

character.c, 14

character.h, 19

blitColor

image.c, 73

image.h, 74

blitEnemies

enemies.c, 27

enemies.h, 40

blitPlatform

mobile_platform.c, 104

mobile_platform.h, 107

blitProjectile

projectile.c, 123

projectile.h, 126

button

Input, 4

c

node, 8

COLLISION_ADJUSTMENT

const.h, 25

captureText

text.c, 134

text.h, 136

Character, 2

countStars, 2

dirX, 2

dirY, 2

doubleJump, 2

hp, 2

hpMax, 2

isFalling, 3

isHurt, 3

isNPC, 3

isOnGround, 3

isRight, 3

location, 3

moving, 3

nbLives, 3

nbProjectile, 3

OnPlatform, 3

saveX, 3

saveY, 3

tile, 3

wallJump, 3

character.c, 14

blitCharacter, 14

checkFall, 15

checkWall, 15

collisionSprite, 15

createCharacter, 15

freeCharacters, 16

moveCharacter, 16

presiseMoveCharacter, 17

tryMovement, 17

character.h, 18

ABS, 19

blitCharacter, 19

checkFall, 19

checkWall, 20

collisionSprite, 20

createCharacter, 20

freeCharacters, 21

moveCharacter, 21

presiseMoveCharacter, 22

SGN, 19

tryMovement, 23

checkFall

character.c, 15

character.h, 19

checkWall

character.c, 15

character.h, 20

chooseKey

menu_option.c, 96

menu_option.h, 100

closeFile

file.c, 51

file.h, 52

closeLevel

file_level.c, 53

file_level.h, 56

closeLevelList

file_level.c, 53

file_level.h, 56

collisionEnemy

enemies.c, 28

enemies.h, 40

collisionMap

map.c, 80

map.h, 83

collisionPlatform

mobile_platform.c, 104

mobile_platform.h, 108

collisionSprite

character.c, 15

character.h, 20

const.h, 24

- COLLISION_ADJUSTMENT, 25
- FPS, 25
- GRAVITY_SPEED, 25
- IMG_END_SIZE, 25
- JUMP_HEIGHT, 25
- MARGE_SCROLLING, 25
- MAX_NB_PLATFORM, 25
- MAX_NB_PROJECTILE, 25
- MAX_SIZE_FILE_NAME, 25
- MAX_SPEED, 25
- min, 26
- NB_KEY, 26
- PLATFORM_SPEED, 26
- PROJECTILE_SPEED, 26
- SCREEN_HEIGHT, 26
- SCREEN_WIDTH, 26
- SPRING_HEIGHT, 26
- TILE_MAX, 26
- TILE_SIZE, 26
- countStars
 - Character, 2
- createCharacter
 - character.c, 15
 - character.h, 20
- createEnemy
 - enemies.c, 28
 - enemies.h, 41
- createPlatform
 - mobile_platform.c, 104
 - mobile_platform.h, 108
- createProjectile
 - projectile.c, 123
 - projectile.h, 126
- createSound
 - sound.c, 130
 - sound.h, 132
- current
 - list, 7
- decrement
 - game.c, 58
 - game.h, 65
- deleteCurrent
 - enemies.c, 30
 - enemies.h, 41
- deleteFirst
 - enemies.c, 31
 - enemies.h, 42
- deleteLast
 - enemies.c, 31
 - enemies.h, 42
- deletePlayer
 - player.c, 116
 - player.h, 119
- deleteProjectile
 - projectile.c, 123
 - projectile.h, 126
- dirX
 - Character, 2
- dirY
 - Character, 2
- direction
 - platform, 9
 - projectile, 12
- doubleJump
 - Character, 2
- empty
 - enemies.c, 32
 - enemies.h, 44
- enemies.c, 26
 - blitEnemies, 27
 - collisionEnemy, 28
 - createEnemy, 28
 - deleteCurrent, 30
 - deleteFirst, 31
 - deleteLast, 31
 - empty, 32
 - first, 32
 - freeEnemies, 33
 - getCurrent, 33
 - initList, 33
 - insertAfterCurrent, 33
 - insertBeforeCurrent, 34
 - insertFirst, 35
 - insertLast, 35
 - last, 36
 - moveCharacterCol, 36
 - moveEnemies, 37
 - newNode, 37
 - next, 38
 - outOfList, 38
 - previous, 38
 - setOnFirst, 38
 - setOnLast, 39
- enemies.h, 39
 - blitEnemies, 40
 - collisionEnemy, 40
 - createEnemy, 41
 - deleteCurrent, 41
 - deleteFirst, 42
 - deleteLast, 42
 - empty, 44
 - first, 44
 - freeEnemies, 45
 - getCurrent, 45
 - initList, 45
 - insertAfterCurrent, 45
 - insertBeforeCurrent, 46
 - insertFirst, 47
 - insertLast, 47
 - last, 48
 - moveCharacterCol, 48
 - moveEnemies, 49
 - newNode, 49
 - next, 50
 - outOfList, 50
 - previous, 50

- setOnFirst, 50
 - setOnLast, 51
- FPS
 - const.h, 25
- file.c, 51
 - closeFile, 51
 - openFile, 51
- file.h, 52
 - closeFile, 52
 - openFile, 52
- file_level.c, 53
 - closeLevel, 53
 - closeLevelList, 53
 - initLevel, 54
 - openLevel, 54
 - readLevelFile, 54
- file_level.h, 55
 - BUFFER_SIZE, 56
 - closeLevel, 56
 - closeLevelList, 56
 - initLevel, 56
 - openLevel, 56
 - readLevelFile, 57
- first
 - enemies.c, 32
 - enemies.h, 44
 - list, 7
- freeCharacters
 - character.c, 16
 - character.h, 21
- freeEnemies
 - enemies.c, 33
 - enemies.h, 45
- freeInput
 - input.c, 75
- freeMap
 - map.c, 80
 - map.h, 83
- freePlatformSet
 - mobile_platform.c, 105
 - mobile_platform.h, 108
- freeProjectileSet
 - projectile.c, 123
 - projectile.h, 126
- freeSound
 - sound.c, 130
 - sound.h, 132
- fromNPC
 - projectile, 12
- fx
 - Sound, 13
- fxSound
 - Sound, 13
- fxVolume
 - Sound, 13
- GRAVITY_SPEED
 - const.h, 25
- game.c, 57
 - decrement, 58
 - move, 58
 - play, 59
 - printGameOver, 60
 - printHUD, 61
 - printPause, 61
 - printWin, 63
 - rocketLaunch, 64
 - updateSpeed, 64
- game.h, 64
 - decrement, 65
 - move, 66
 - play, 66
 - printGameOver, 68
 - printHUD, 69
 - printPause, 69
 - printWin, 71
 - rocketLaunch, 72
 - updateSpeed, 72
- getCurrent
 - enemies.c, 33
 - enemies.h, 45
- hat
 - Input, 4
- hatMoved
 - Input, 4
- height
 - Level, 5
- hp
 - Character, 2
- hpMax
 - Character, 2
- IMG_END_SIZE
 - const.h, 25
- image.c, 72
 - blitColor, 73
 - imageLoad, 73
 - imageLoadAlpha, 73
- image.h, 73
 - blitColor, 74
 - imageLoad, 74
 - imageLoadAlpha, 74
- imageLoad
 - image.c, 73
 - image.h, 74
- imageLoadAlpha
 - image.c, 73
 - image.h, 74
- initInput
 - input.c, 75
- initJoystick
 - input.c, 76
- initLevel
 - file_level.c, 54
 - file_level.h, 56
- initList

- enemies.c, 33
- enemies.h, 45
- initMap
 - map.c, 81
 - map.h, 83
- initPlatformSet
 - mobile_platform.c, 105
 - mobile_platform.h, 109
- initProjSet
 - projectile.c, 124
 - projectile.h, 127
- Input, 4
 - axes, 4
 - button, 4
 - hat, 4
 - hatMoved, 4
 - isJoystick, 4
 - joystick, 4
 - key, 4
 - quit, 4
 - space, 4
 - useJoystick, 5
- input.c, 75
 - freeInput, 75
 - initInput, 75
 - initJoystick, 76
 - inputActionGame, 76
 - inputActionMenu, 76
 - updateEvents, 77
 - updateWaitEvents, 77
- inputActionGame
 - input.c, 76
- inputActionMenu
 - input.c, 76
- insertAfterCurrent
 - enemies.c, 33
 - enemies.h, 45
- insertBeforeCurrent
 - enemies.c, 34
 - enemies.h, 46
- insertFirst
 - enemies.c, 35
 - enemies.h, 47
- insertLast
 - enemies.c, 35
 - enemies.h, 47
- isFalling
 - Character, 3
- isHurt
 - Character, 3
- isJoystick
 - Input, 4
- isNPC
 - Character, 3
- isOnGround
 - Character, 3
- isRight
 - Character, 3
- JUMP_HEIGHT
 - const.h, 25
- joystick
 - Input, 4
- key
 - Input, 4
- keyBoardOptions
 - menu_option.c, 96
 - menu_option.h, 100
- last
 - enemies.c, 36
 - enemies.h, 48
 - list, 7
- Level, 5
 - background, 5
 - height, 5
 - map, 5
 - music, 5
 - tileSet, 5
 - tileSet2, 5
 - tileSetUse, 5
 - timer_level, 6
 - width, 6
- levelMax
 - Player, 11
- list, 6
 - current, 7
 - first, 7
 - last, 7
- loadInputOptions
 - option.c, 111
 - option.h, 113
- loadPlayer
 - player.c, 116
 - player.h, 120
- loadSoundOptions
 - option.c, 111
 - option.h, 114
- location
 - Character, 3
 - platform, 9
 - projectile, 12
- lvl
 - Map, 7
- MARGE_SCROLLING
 - const.h, 25
- MAX_NB_PLATFORM
 - const.h, 25
- MAX_NB_PROJECTILE
 - const.h, 25
- MAX_SIZE_FILE_NAME
 - const.h, 25
- MAX_SPEED
 - const.h, 25
- main
 - main.c, 78

- main.c, 77
 - main, 78
- mainMenu
 - menu.c, 85
 - menu.h, 89
- Map, 7
 - lvl, 7
 - screenHeight, 7
 - screenWidth, 8
 - xScroll, 8
- map
 - Level, 5
- map.c, 79
 - collisionMap, 80
 - freeMap, 80
 - initMap, 81
 - scrolling, 81
 - updateScreenMap, 82
- map.h, 82
 - collisionMap, 83
 - freeMap, 83
 - initMap, 83
 - scrolling, 84
 - updateScreenMap, 84
- menu.c, 84
 - blinkText, 85
 - mainMenu, 85
 - menuPlayers, 86
 - titleMenu, 87
- menu.h, 88
 - blinkText, 89
 - mainMenu, 89
 - menuPlayers, 90
 - titleMenu, 91
- menu_level.c, 92
 - menuLevel, 93
- menu_level.h, 94
 - menuLevel, 94
- menu_option.c, 95
 - chooseKey, 96
 - keyBoardOptions, 96
 - optionMenu, 97
 - soundOptions, 98
- menu_option.h, 99
 - chooseKey, 100
 - keyBoardOptions, 100
 - optionMenu, 101
 - soundOptions, 102
- menuLevel
 - menu_level.c, 93
 - menu_level.h, 94
- menuPlayers
 - menu.c, 86
 - menu.h, 90
- min
 - const.h, 26
- mobile_platform.c, 103
 - blitPlatform, 104
 - collisionPlatform, 104
 - createPlatform, 104
 - freePlatformSet, 105
 - initPlatformSet, 105
 - moveOnePlatform, 105
 - movePlatform, 106
 - platformMap, 106
- mobile_platform.h, 107
 - blitPlatform, 107
 - collisionPlatform, 108
 - createPlatform, 108
 - freePlatformSet, 108
 - initPlatformSet, 109
 - moveOnePlatform, 109
 - movePlatform, 109
 - platformMap, 110
- move
 - game.c, 58
 - game.h, 66
- moveCharacter
 - character.c, 16
 - character.h, 21
- moveCharacterCol
 - enemies.c, 36
 - enemies.h, 48
- moveEnemies
 - enemies.c, 37
 - enemies.h, 49
- moveOnePlatform
 - mobile_platform.c, 105
 - mobile_platform.h, 109
- moveOneProjectile
 - projectile.c, 124
 - projectile.h, 127
- movePlatform
 - mobile_platform.c, 106
 - mobile_platform.h, 109
- moveProjectiles
 - projectile.c, 124
 - projectile.h, 127
- moving
 - Character, 3
- mscSound
 - Sound, 13
- music
 - Level, 5
 - Sound, 13
- musicVolume
 - Sound, 13
- NB_KEY
 - const.h, 26
- NB_TILE_MARYO_WIDTH
 - const.h, 26
- nb
 - platformSet, 10
 - projectileSet, 12
- nbCoins
 - Player, 11

- nbLives
 - Character, 3
 - Player, 11
- nbProjectile
 - Character, 3
 - Player, 11
- newNode
 - enemies.c, 37
 - enemies.h, 49
- newPlayer
 - player.c, 117
 - player.h, 120
- next
 - enemies.c, 38
 - enemies.h, 50
 - node, 8
- node, 8
 - c, 8
 - next, 8
 - previous, 8
- OnPlatform
 - Character, 3
- openFile
 - file.c, 51
 - file.h, 52
- openLevel
 - file_level.c, 54
 - file_level.h, 56
- option.c, 110
 - loadInputOptions, 111
 - loadSoundOptions, 111
 - saveInputOptions, 112
 - saveSoundOptions, 112
- option.h, 113
 - loadInputOptions, 113
 - loadSoundOptions, 114
 - saveInputOptions, 114
 - saveSoundOptions, 115
- optionMenu
 - menu_option.c, 97
 - menu_option.h, 101
- outOfList
 - enemies.c, 38
 - enemies.h, 50
- PLATFORM_SPEED
 - const.h, 26
- PROJECTILE_SPEED
 - const.h, 26
- platform, 9
 - direction, 9
 - location, 9
 - speed, 9
 - sprite, 9
 - type, 9
 - xMax, 9
 - xMin, 9
 - yMax, 9
 - yMin, 10
- platformMap
 - mobile_platform.c, 106
 - mobile_platform.h, 110
- platformSet, 10
 - nb, 10
 - tab, 10
- play
 - game.c, 59
 - game.h, 66
- playMusic
 - sound.c, 130
 - sound.h, 132
- playShortSound
 - sound.c, 130
 - sound.h, 132
- Player, 11
 - levelMax, 11
 - nbCoins, 11
 - nbLives, 11
 - nbProjectile, 11
- player.c, 115
 - deletePlayer, 116
 - loadPlayer, 116
 - newPlayer, 117
 - save, 117
 - savePlayer, 118
- player.h, 118
 - deletePlayer, 119
 - loadPlayer, 120
 - newPlayer, 120
 - save, 121
 - savePlayer, 122
- presiseMoveCharacter
 - character.c, 17
 - character.h, 22
- previous
 - enemies.c, 38
 - enemies.h, 50
 - node, 8
- printGameOver
 - game.c, 60
 - game.h, 68
- printHUD
 - game.c, 61
 - game.h, 69
- printPause
 - game.c, 61
 - game.h, 69
- printText
 - text.c, 135
 - text.h, 137
- printWin
 - game.c, 63
 - game.h, 71
- projectile, 11
 - direction, 12
 - fromNPC, 12

- location, 12
- sprite, 12
- projectile.c, 122
 - blitProjectile, 123
 - createProjectile, 123
 - deleteProjectile, 123
 - freeProjectileSet, 123
 - initProjSet, 124
 - moveOneProjectile, 124
 - moveProjectiles, 124
- projectile.h, 125
 - blitProjectile, 126
 - createProjectile, 126
 - deleteProjectile, 126
 - freeProjectileSet, 126
 - initProjSet, 127
 - moveOneProjectile, 127
 - moveProjectiles, 127
- projectileSet, 12
 - nb, 12
 - projectileThrown, 13
 - tab, 13
- projectileThrown
 - projectileSet, 13
- quit
 - Input, 4
- readLevelFile
 - file_level.c, 54
 - file_level.h, 57
- rocketLaunch
 - game.c, 64
 - game.h, 72
- SCREEN_HEIGHT
 - const.h, 26
- SCREEN_WIDTH
 - const.h, 26
- SGN
 - character.h, 19
- SPRING_HEIGHT
 - const.h, 26
- save
 - player.c, 117
 - player.h, 121
- saveInputOptions
 - option.c, 112
 - option.h, 114
- savePlayer
 - player.c, 118
 - player.h, 122
- saveSoundOptions
 - option.c, 112
 - option.h, 115
- saveX
 - Character, 3
- saveY
 - Character, 3
- screenHeight
 - Map, 7
- screenWidth
 - Map, 8
- scrolling
 - map.c, 81
 - map.h, 84
- setOnFirst
 - enemies.c, 38
 - enemies.h, 50
- setOnLast
 - enemies.c, 39
 - enemies.h, 51
- share.c, 128
 - waitFPS, 129
- share.h, 129
 - waitFPS, 129
- Sound, 13
 - fx, 13
 - fxSound, 13
 - fxVolume, 13
 - mscSound, 13
 - music, 13
 - musicVolume, 13
- sound.c, 129
 - createSound, 130
 - freeSound, 130
 - playMusic, 130
 - playShortSound, 130
 - soundVolume, 131
 - stopSound, 131
- sound.h, 131
 - createSound, 132
 - freeSound, 132
 - playMusic, 132
 - playShortSound, 132
 - soundVolume, 132
 - stopSound, 132
- soundOptions
 - menu_option.c, 98
 - menu_option.h, 102
- soundVolume
 - sound.c, 131
 - sound.h, 132
- space
 - Input, 4
- speed
 - platform, 9
- sprite
 - platform, 9
 - projectile, 12
- stopSound
 - sound.c, 131
 - sound.h, 132
- structures.h, 132
- TILE_MAX
 - const.h, 26
- TILE_SIZE

- const.h, 26
- tab
 - platformSet, 10
 - projectileSet, 13
- text.c, 133
 - captureText, 134
 - printText, 135
- text.h, 135
 - captureText, 136
 - printText, 137
- tile
 - Character, 3
- tileSet
 - Level, 5
- tileSet2
 - Level, 5
- tileSetUse
 - Level, 5
- timer_level
 - Level, 6
- titleMenu
 - menu.c, 87
 - menu.h, 91
- tryMovement
 - character.c, 17
 - character.h, 23
- type
 - platform, 9
- updateEvents
 - input.c, 77
- updateScreenMap
 - map.c, 82
 - map.h, 84
- updateSpeed
 - game.c, 64
 - game.h, 72
- updateWaitEvents
 - input.c, 77
- useJoystick
 - Input, 5
- waitFPS
 - share.c, 129
 - share.h, 129
- wallJump
 - Character, 3
- width
 - Level, 6
- xMax
 - platform, 9
- xMin
 - platform, 9
- xScroll
 - Map, 8
- yMax
 - platform, 9
- yMin
 - platform, 10