Super Martin

Generated by Doxygen 1.8.6

Mon Jun 2 2014 13:58:39

Contents

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 nbLifes
- 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 diraction (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 nbLifes

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 : SDL_HAT_CENTERED, SDL_HAT_UP, SDL_HAT_RIGHT, SDL_HAT_DOWN, SDL_HAT-_LEFT, SDL_HAT_RIGHTUP, SDL_HAT_RIGHTDOWN, SDL_HAT_LEFTUP, SDL_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 SDL_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

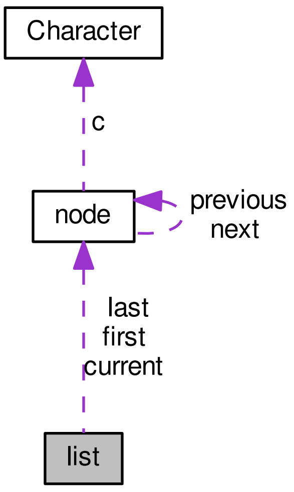
The documentation for this struct was generated from the following file:

- 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

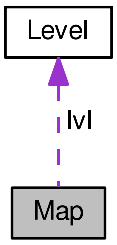
The documentation for this struct was generated from the following file:

- 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

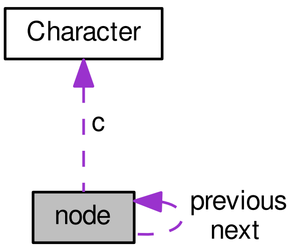
The documentation for this struct was generated from the following file:

- 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 hight limit for deplacement

1.7.2.7 int xMin

x low limit for deplacement

1.7.2.8 int yMax

y hight limit for deplacement

1.7.2.9 int yMin

y hight limit for deplacement

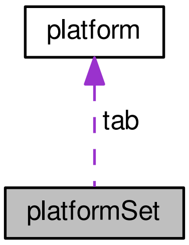
The documentation for this struct was generated from the following file:

- 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 nbLifes
- 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 nbLifes

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

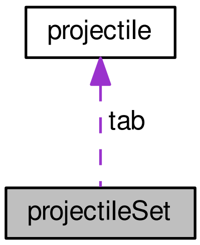
The documentation for this struct was generated from the following file:

- 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	*screen*	game screen

in	*c*	the character
in	*m*	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	*c*	the monster/character to be tested
in	*m*	the game map
in	*ps*	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	*c*	the monster/character to be tested
in	*m*	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	*s1*	the first sprite
in	*s2*	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

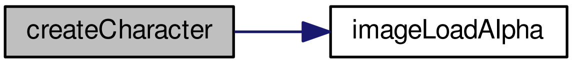
in	*tile*	character tileSet address
in	*x*	character's x location
in	*y*	character's y location
in	*npc*	type of npc if creating a npc, 0 if not
in	*nbProjectile*	the number of projectiles the character has

in	*nbCoins*	the number of coins the character has
in	*nbLifes*	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	*c*	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

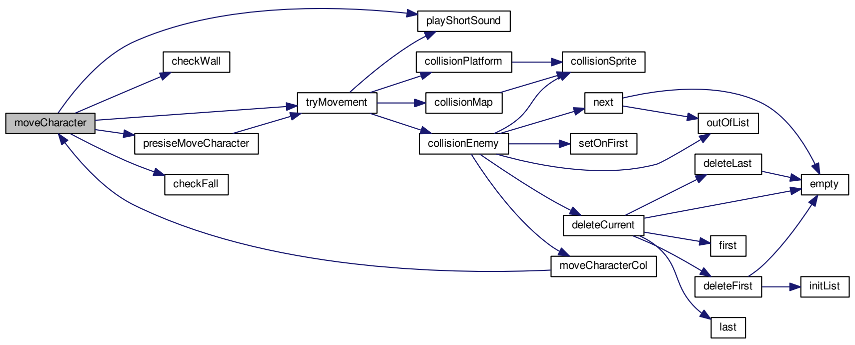
Parameters

in,out	*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:



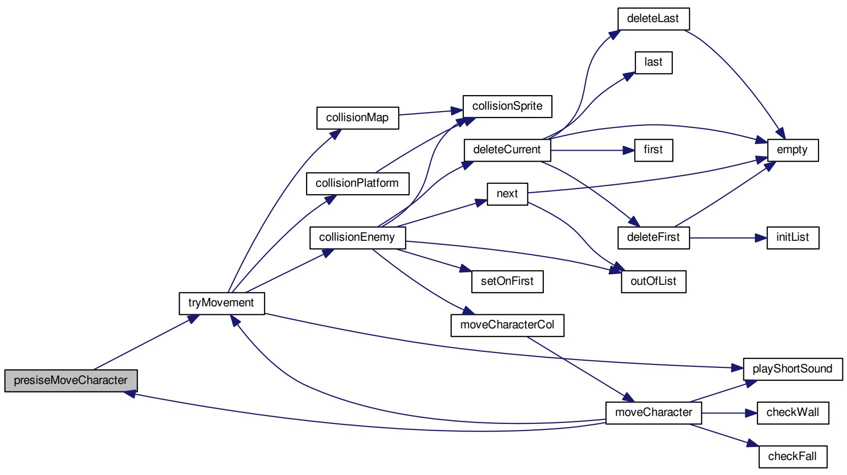
2.1.2.8 void presiseMoveCharacter (**Character** ∗ *c,* int *vx,* int *vy,* **Map** ∗ *m,* **list** ∗ *l,* **platformSet** ∗ *ps*)

make a more presise 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	*c*	the charactere
in	*m*	the map
in	*vx*	the horizontal component of the movement vector
in	*vy*	the vertical component of the movement vector
in,out	*l*	the enemy list
out	*ps*	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

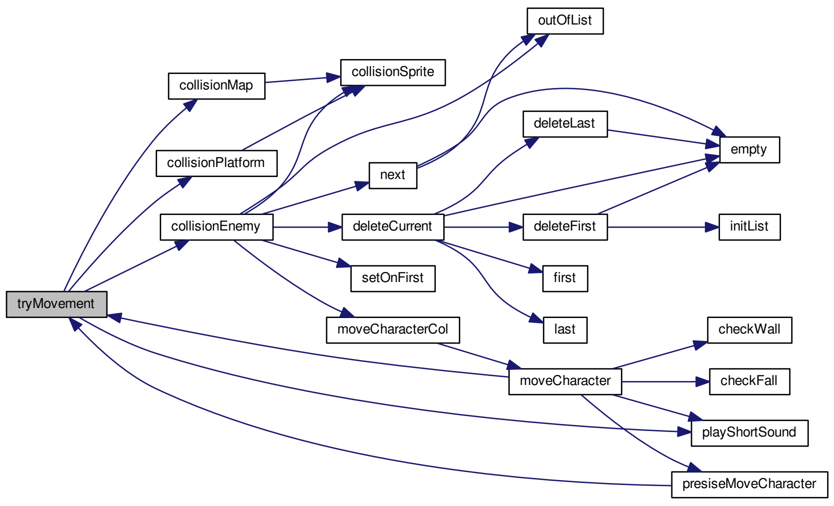
Parameters

`in,out`	*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:



2.2 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 nbLifes)
- 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

in,out	*screen*	game screen
in	*c*	the character
in	*m*	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`	*c*	the monster/character to be tested
`in`	*m*	the game map
`in`	*ps*	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`	*c*	the monster/character to be tested
`in`	*m*	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`	*s1*	the first sprite
`in`	*s2*	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

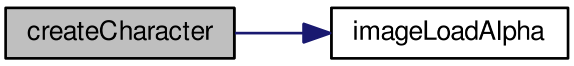
Parameters

`in`	*tile*	character tileSet address
`in`	*x*	character's x location
`in`	*y*	character's y location
`in`	*npc*	type of npc if creating a npc, 0 if not
`in`	*nbProjectile*	the number of projectiles the character has
`in`	*nbCoins*	the number of coins the character has
`in`	*nbLifes*	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

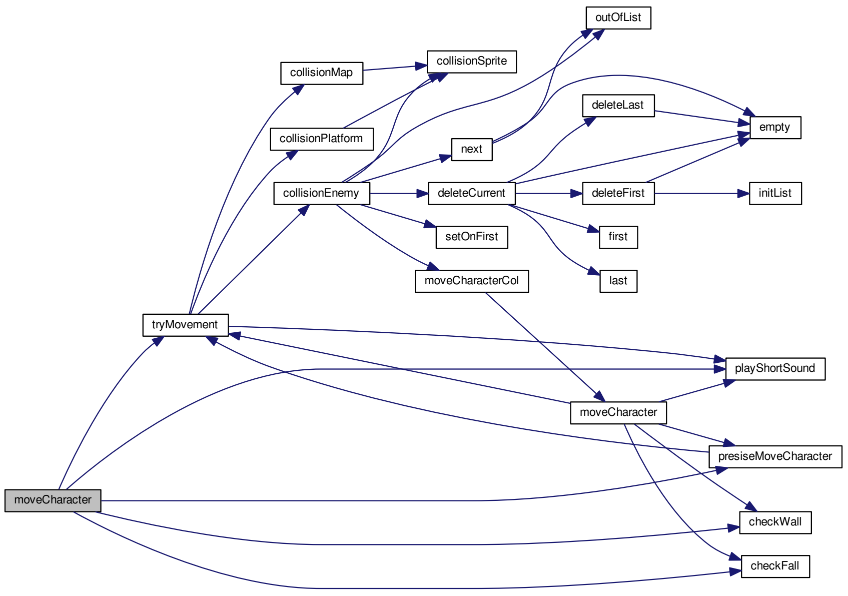
Parameters

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:



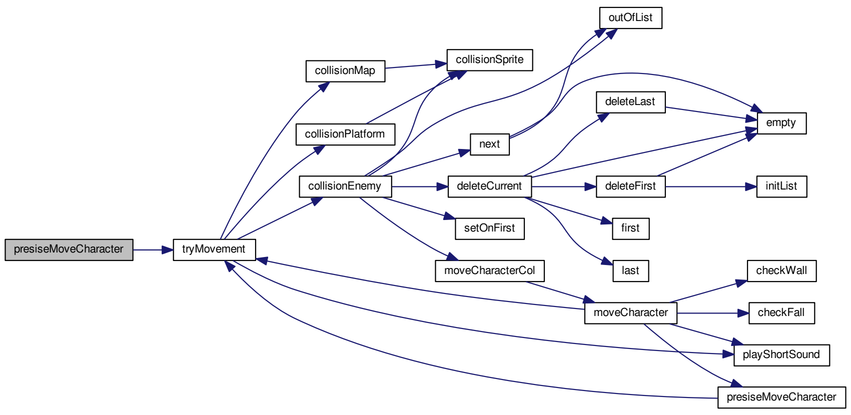
2.2.3.8 void presiseMoveCharacter (**Character** ∗ *c,* int *vx,* int *vy,* **Map** ∗ *m,* **list** ∗ *l,* **platformSet** ∗ *ps*)

make a more presise 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	*c*	the charactere
in	*m*	the map
in	*vx*	the horizontal component of the movement vector
in	*vy*	the vertical component of the movement vector
in,out	*l*	the enemy list
out	*ps*	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

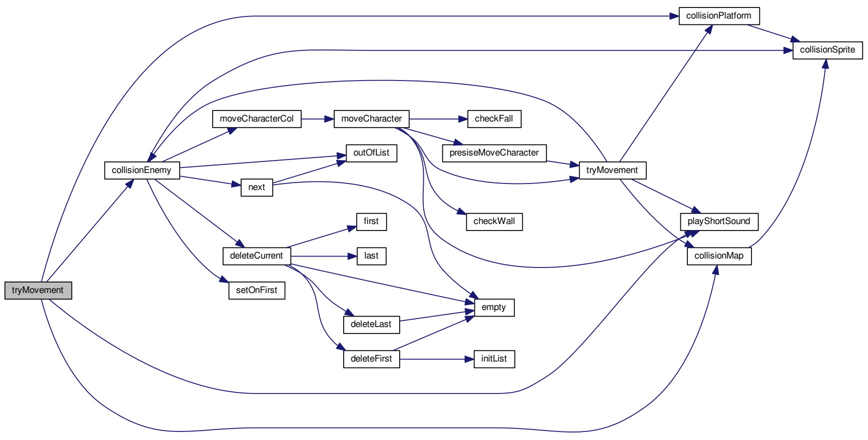
Parameters

`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 adjustement

2.3.2.2 #define DEPLACEMENT_POURCENTAGE 0

The deplacement 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 ∗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)
- 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

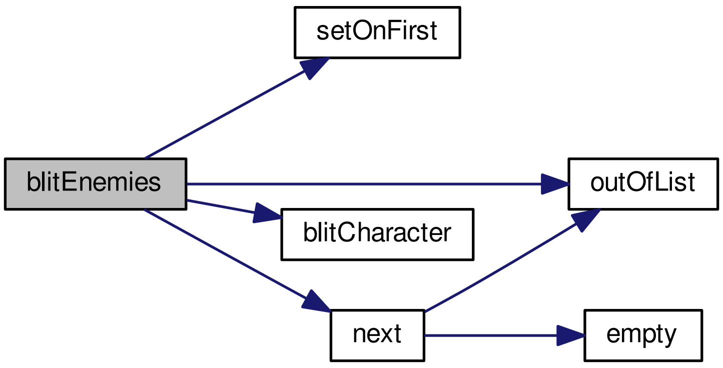
2.4.2.1 void blitEnemies (SDL_Surface ∗ *screen,* list ∗ *l,* Map ∗ *m*)

blit the enemies

Parameters

in,out	*screen*	game screen
in,out	*m*	the map
in,out	*l*	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 beteewen the player sprite and an enemy and deals with

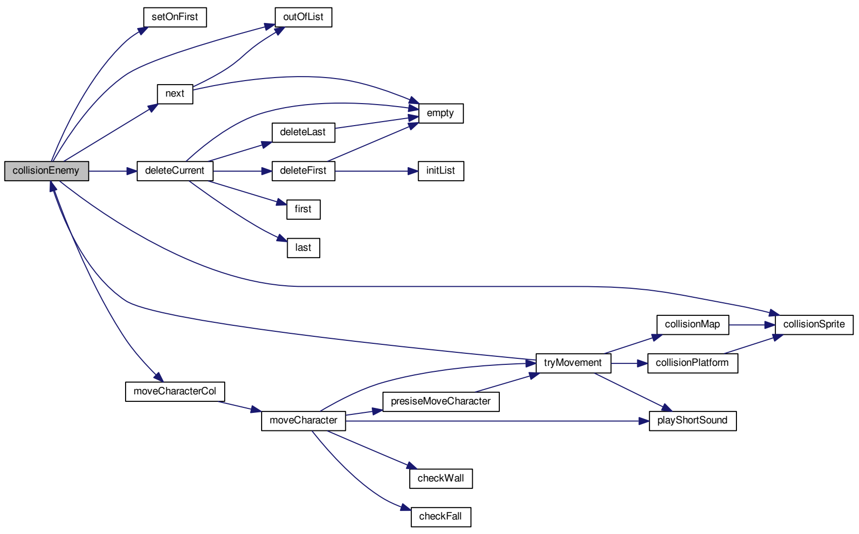
Parameters

`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:



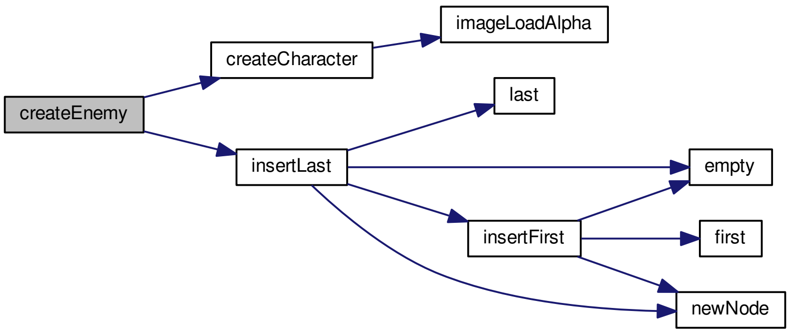
2.4.2.3 void createEnemy (char ∗ *tile,* int *x,* int *y,* list ∗ *l,* int *type*)

creates an enemy and adds it to an enemies list

Parameters

`in`	*tile*	the tilset name
`in`	*x*	enemy's x location
`in`	*y*	enemy's y location
`out`	*l*	enemies list
`in`	*type*	the type of enemy

Here is the call graph for this function:



2.4.2.4 enemy ∗ deleteCurrent (list ∗ *l*)

delete the current node

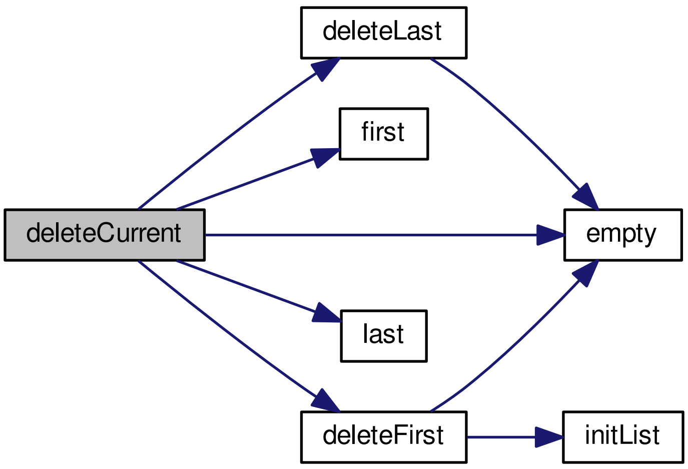
Parameters

`out`	*l*	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

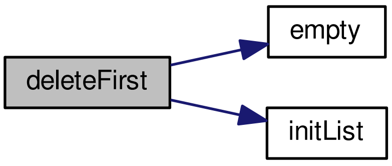
Parameters

out	*l*	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

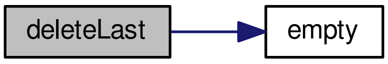
Parameters

out	*l*	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 ∗ *l*)

tests if the list is empty

Parameters

in	*l*	the list to be tested

Returns

1 if the list is empty, 0 if not

2.4.2.8 int first (list ∗ *l*)

tests if the current node is the first node

Parameters

in	*l*	the list to be tested

Returns

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

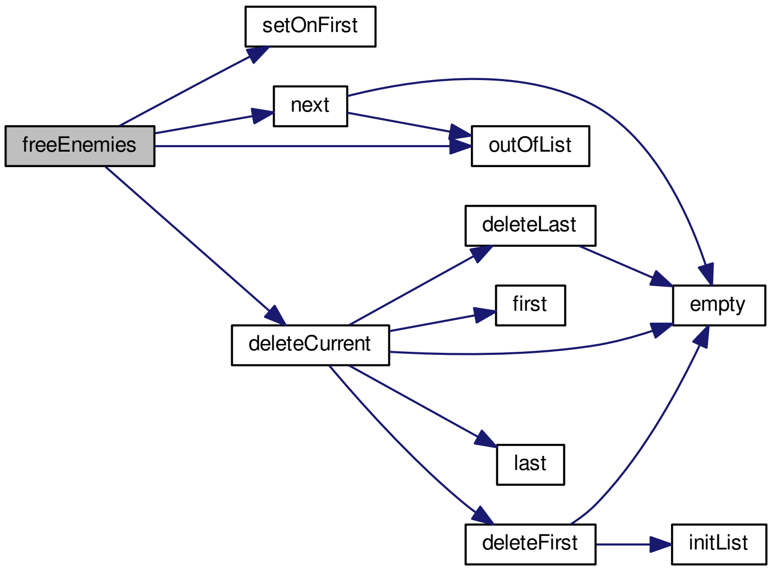
2.4.2.9 void freeEnemies (**list** ∗ *l*)

free all the enemies and the list

Parameters

out	*l*	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	*l*	the list to be modified

2.4.2.11 void initList (**list** ∗ *l*)

initialize the enemy list

Parameters

out	*l*	the list to be initalized

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

insert a enemy just after the current node

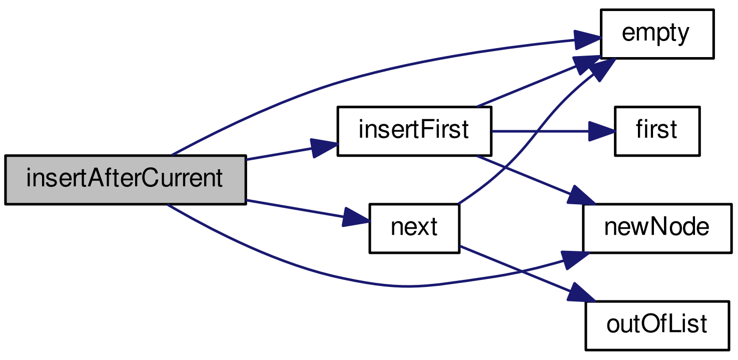
Parameters

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:



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

insert a enemy just before the current node

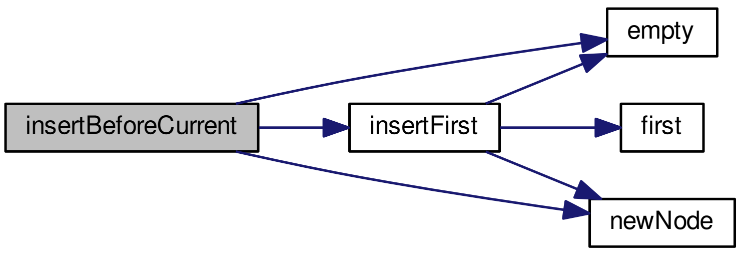
Parameters

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:



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

insert a enemy as first node

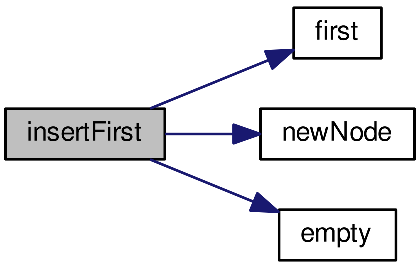
Parameters

`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:



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

insert a enemy as last node

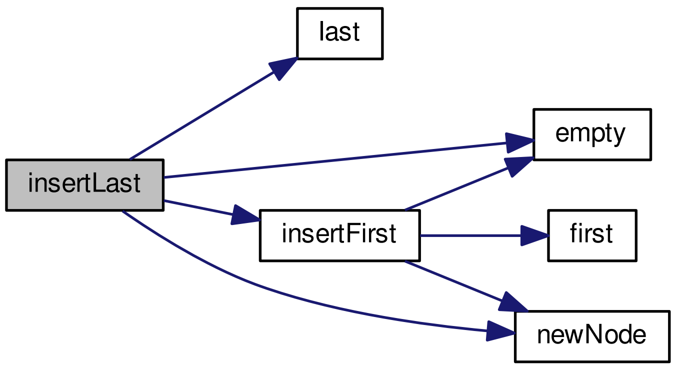
Parameters

`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:



2.4.2.16 int last (**list** ∗ *l*)

tests if the current node is the last node

Parameters

`in`	*l*	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

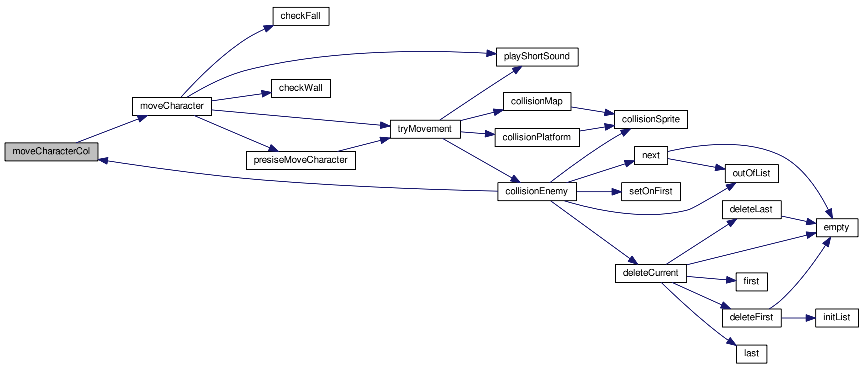
Parameters

`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:



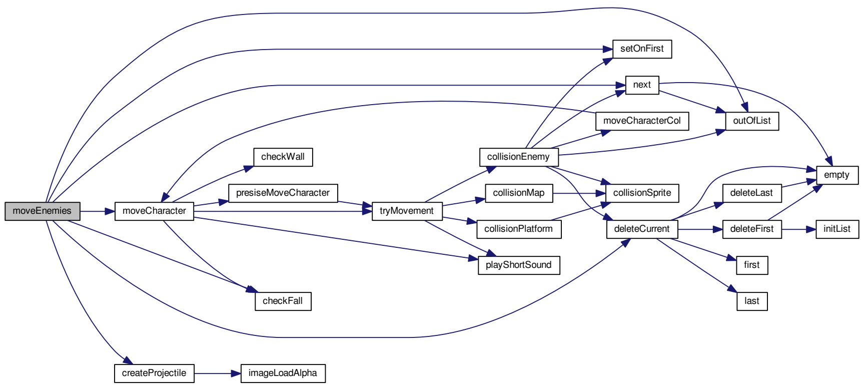
2.4.2.18 void moveEnemies (list ∗ *l,* Map ∗ *m,* list ∗ *p,* projectileSet ∗ *ps,* int ∗ *launch*)

make the enemies moving

Parameters

`in,out`	*l*	the enemy list
`in`	*m*	the game map
`in,out`	*p*	the player list
`out`	*ps*	the projectile set
`in`	*launch*	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	*c*	the character of the node
in	*n*	the next node
in	*p*	the previous node

Returns

a pointer on the created node

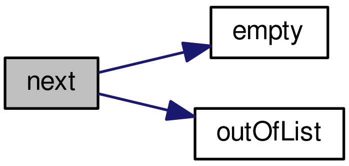
2.4.2.20 void next (list ∗ *l*)

set the current node on its next node

Parameters

out	*l*	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	*l*	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	*l*	the list to be modified

2.4.2.23 void setOnFirst (list ∗ *l*)

set the current node on the first node

Parameters

out	*l*	the list to be modified

2.4.2.24 void setOnLast (list ∗ *l*)

set the current node on the last node

Parameters

out	*l*	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

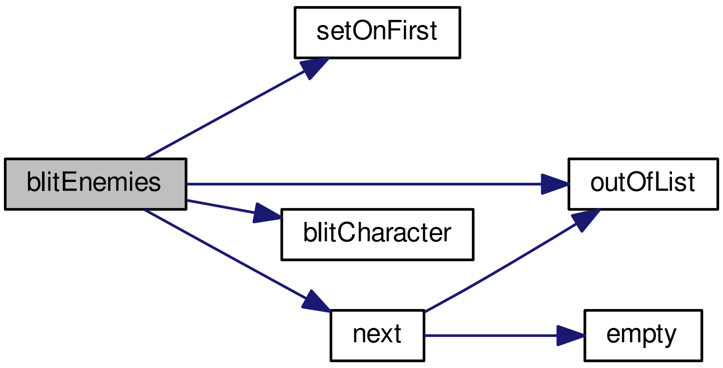
2.5.2.1 void blitEnemies (SDL_Surface ∗ *screen*, list ∗ *l*, Map ∗ *m*)

blit the enemies

Parameters

`in,out`	*screen*	game screen
`in,out`	*m*	the map
`in,out`	*l*	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 beteewen the player sprite and an enemy and deals with

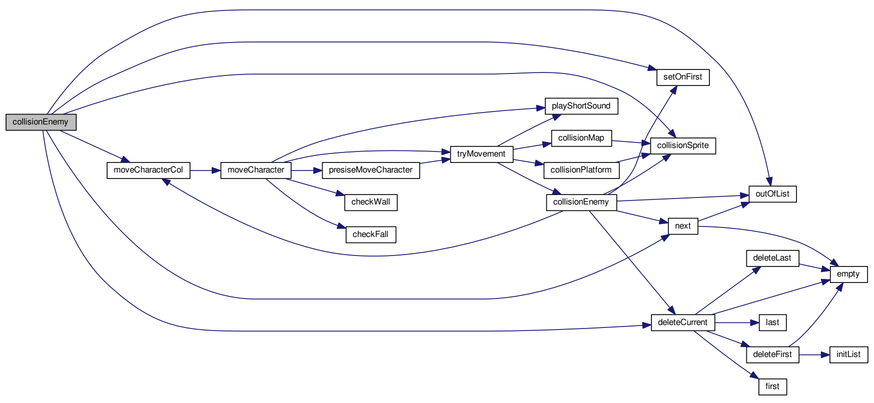
Parameters

`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:



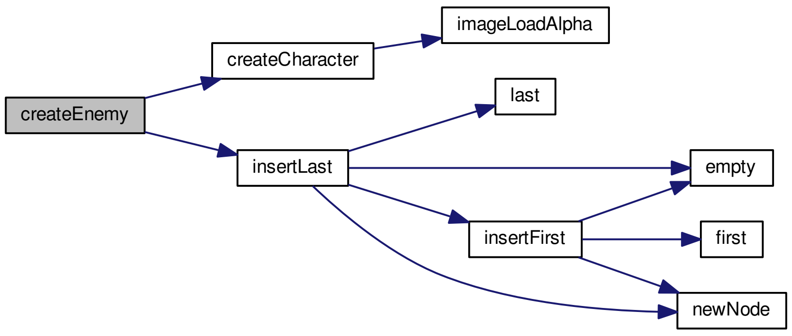
2.5.2.3 void createEnemy (char ∗ *tile,* int *x,* int *y,* list ∗ *l,* int *type*)

creates an enemy and adds it to an enemies list

Parameters

in	*tile*	the tilset name
in	*x*	enemy's x location
in	*y*	enemy's y location
out	*l*	enemies list
in	*type*	the type of enemy

Here is the call graph for this function:



2.5.2.4 Character∗ deleteCurrent (list ∗ *l*)

delete the current node

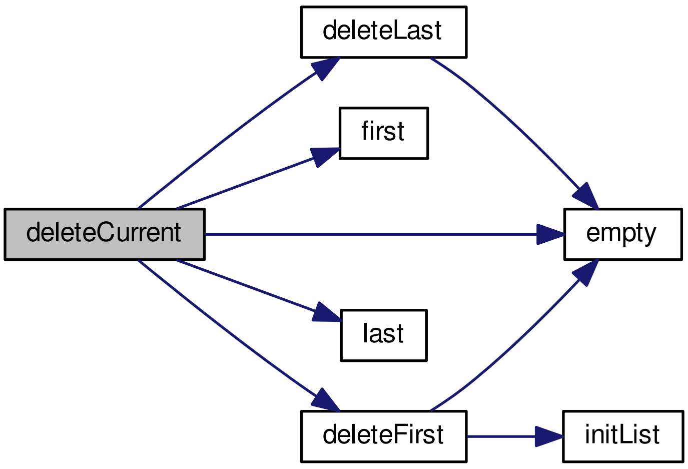
Parameters

`out`	*l*	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 ∗ *l*)

delete the first node

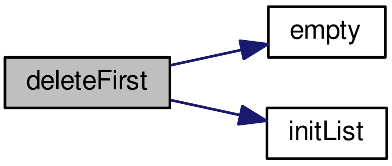
Parameters

`out`	*l*	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

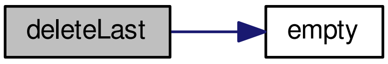
Parameters

out	*l*	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 ∗ *l*)

tests if the list is empty

Parameters

in	*l*	the list to be tested

Returns

1 if the list is empty, 0 if not

2.5.2.8 int first (list ∗ *l*)

tests if the current node is the first node

Parameters

in	*l*	the list to be tested

Returns

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

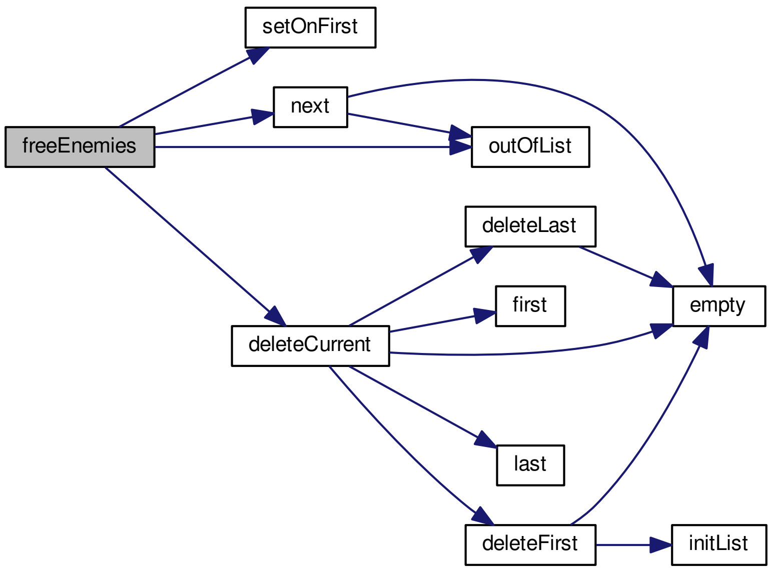
2.5.2.9 void freeEnemies (list ∗ *l*)

free all the enemies and the list

Parameters

out	*l*	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	*l*	the list to be modified

2.5.2.11 void initList (list ∗ *l*)

initialize the enemy list

Parameters

out	*l*	the list to be initalized

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

insert a enemy just after the current node

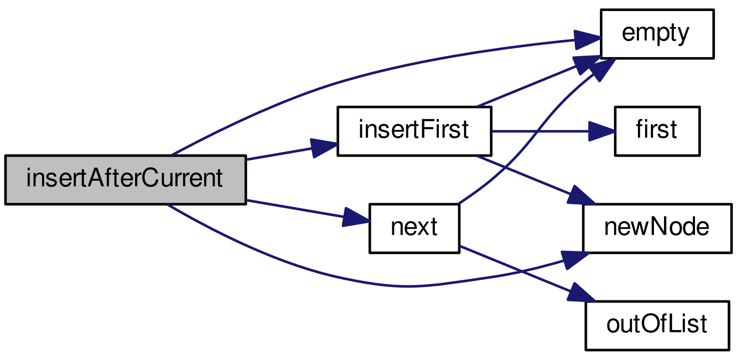
Parameters

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:



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

insert a enemy just before the current node

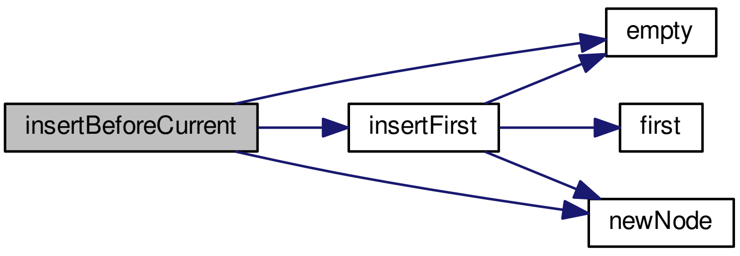
Parameters

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:



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

insert a enemy as first node

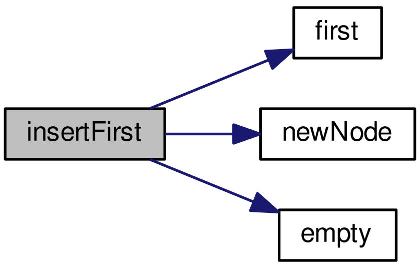
Parameters

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

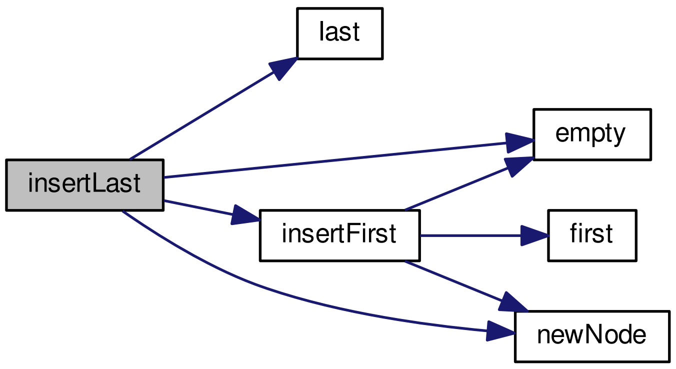
Parameters

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`	*l*	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

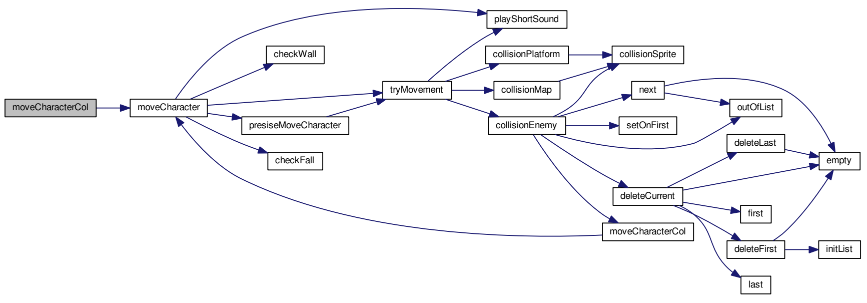
Parameters

`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:



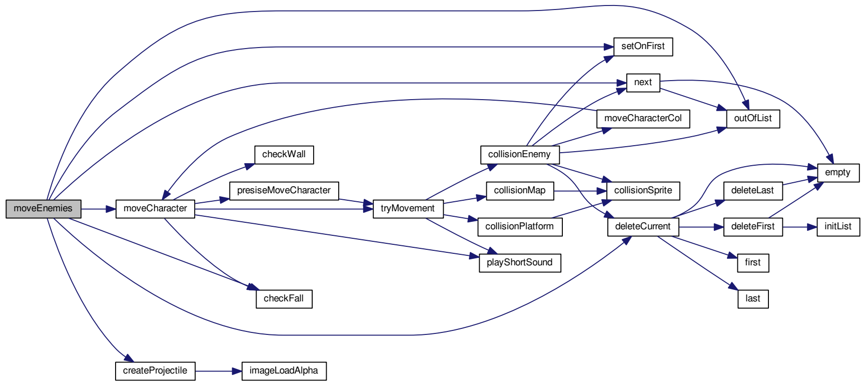
2.5.2.18 void moveEnemies (list ∗ *l*, Map ∗ *m*, list ∗ *p*, projectileSet ∗ *ps*, int ∗ *launch*)

make the enemies moving

Parameters

`in,out`	*l*	the enemy list
`in`	*m*	the game map
`in,out`	*p*	the player list
`out`	*ps*	the projectile set
`in`	*launch*	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	*c*	the character of the node
in	*n*	the next node
in	*p*	the previous node

Returns

a pointer on the created node

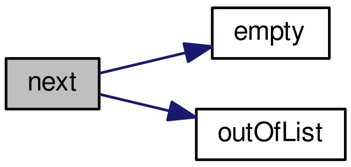
2.5.2.20 void next (list ∗ *l*)

set the current node on its next node

Parameters

out	*l*	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	*l*	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	*l*	the list to be modified

2.5.2.23 void setOnFirst (list ∗ *l*)

set the current node on the first node

Parameters

out	*l*	the list to be modified

2.5.2.24 void setOnLast (list ∗ *l*)

set the current node on the last node

Parameters

out	*l*	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 succefuly closed, 1 if not

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

open a file

Parameters

`in`	*name*	the file name/path
`in`	*mode*	the opening mode

Returns

a pointer on the opened file, NULL if error

2.7 file.h File Reference

Prototypes des fonctions d'acces 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'acces 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`	**ptr_file*	the file to be closed

Returns

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

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

open a file

Parameters

`in`	*name*	the file name/path
`in`	*mode*	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`	*lvl*	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`	*level_names*	level name list
`in`	*nb_lvl*	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`	*lvl*	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

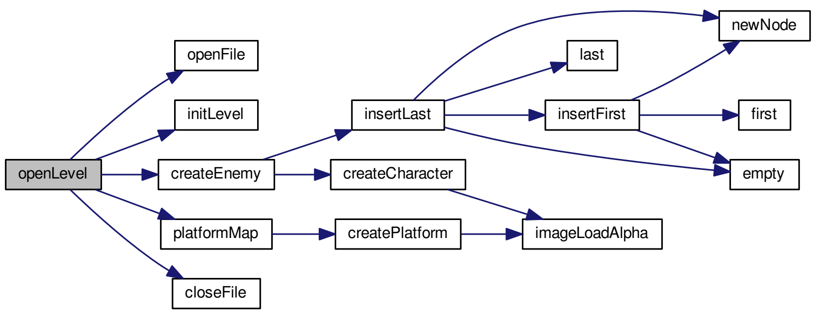
Parameters

`in`	*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:



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

read a file level

Parameters

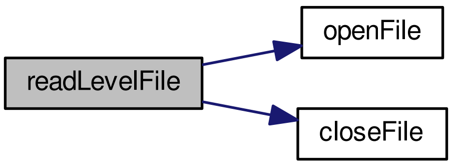
`out`	*nb_lvl*	number of level

out	*file_path*	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 ∗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)

2.9.1 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

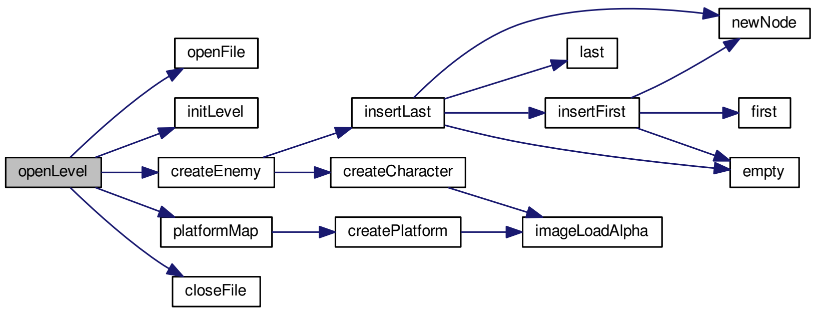
Parameters

`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

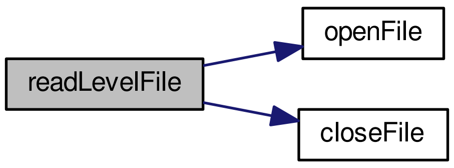
Parameters

out	*nb_lvl*	number of level
out	*file_path*	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_lvl)
- 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	*interval*	the interval between two calls of the function
out	*parameter*	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*)

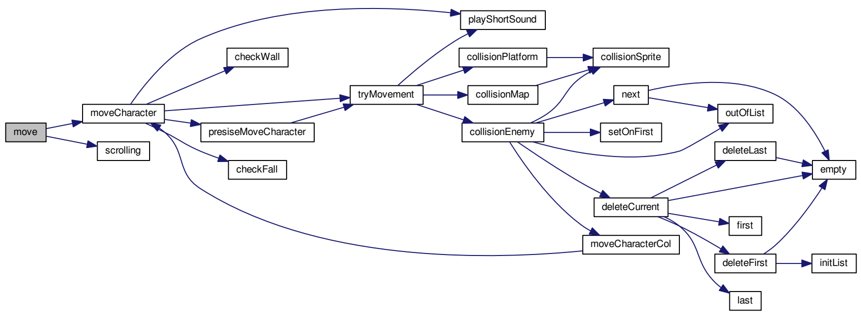
moves the player and scrolls the screen if needed

Parameters

in	*move_left*	1 if move to the left
in	*move_right*	1 if move to the right
in	*jump*	1 if jump
in	*player*	the player
in	*m*	the game map

in	*speed*	the movement speed
out	*acceleration*	the acceleration of the player
in,out	*l*	the enemy list
out	*sound_sys*	the game sound system
out	*ps*	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

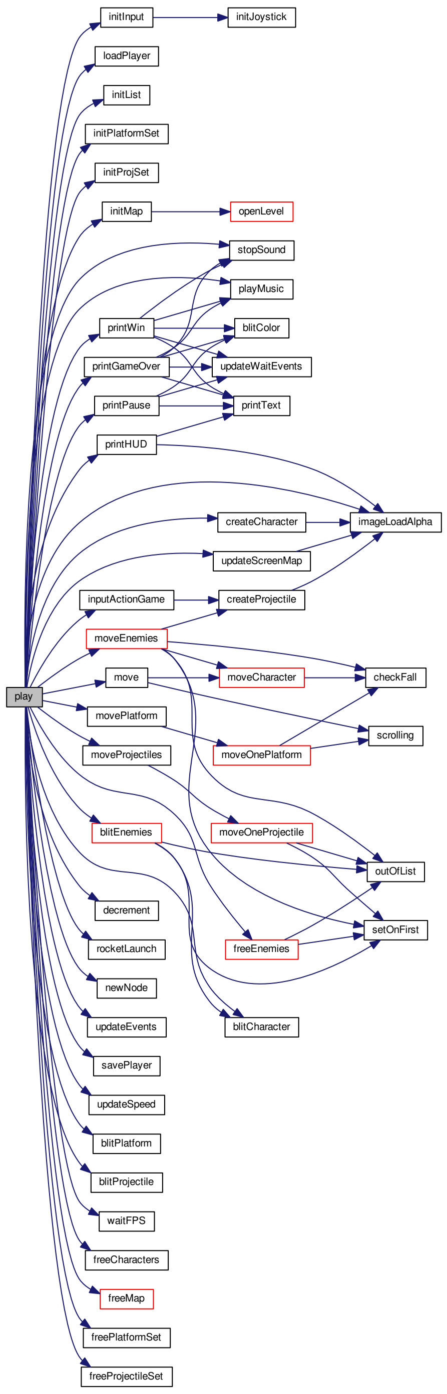
Parameters

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



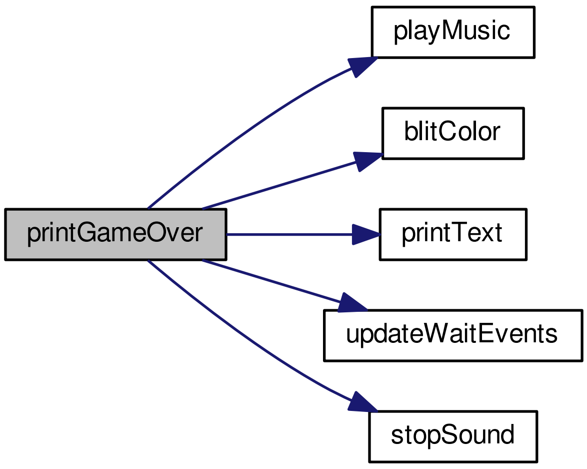
2.10.2.4 void printGameOver (SDL_Surface ∗ *screen*, int ∗ *go*, Input ∗ *in*, Sound ∗ *sound_sys*)

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

Parameters

out	*screen*	the game screen
out	*go*	the game function main loop validation
in,out	*in*	the input structure
out	*sound_sys*	the sound system

Here is the call graph for this function:



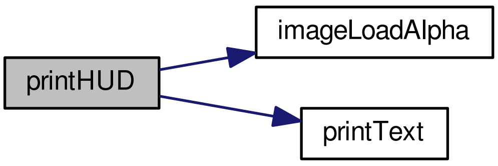
2.10.2.5 void printHUD (SDL_Surface ∗ *screen,* Character ∗ *player,* Map ∗ *m*)

print the player HUD on the screen

Parameters

in,out	*screen*	the game screen
in	*player*	the player
in	*m*	the game map

Here is the call graph for this function:



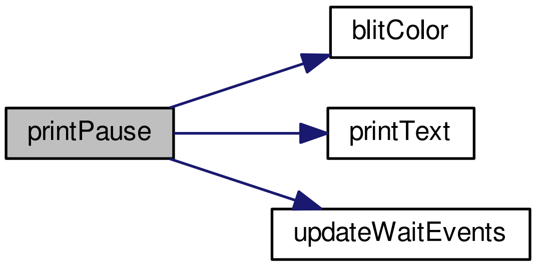
2.10.2.6 void printPause (SDL_Surface ∗ *screen*, Input ∗ *in*, int ∗ *time*, int ∗ *go*, SDLKey ∗ *kc*)

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

Parameters

`out`	*screen*	the game screen
`out`	*go*	the game function main loop validation
`in,out`	*in*	the input structure
`in`	*time*	the current time of the level
`in`	*kc*	the keyboard configuration

Here is the call graph for this function:



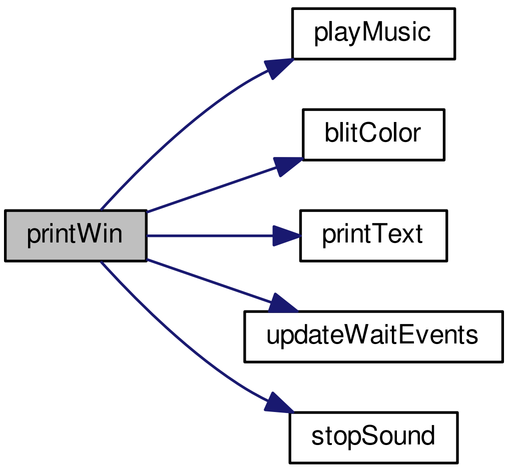
2.10.2.7 void printWin (SDL_Surface ∗ *screen,* int ∗ *go,* Input ∗ *in,* Sound ∗ *sound_sys*)

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

Parameters

`out`	*screen*	the game screen
`out`	*go*	the game function main loop validation
`in,out`	*in*	the input structure
`out`	*sound_sys*	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	*interval*	the interval between two calls of the function
out	*parameter*	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	*speed*	the player speed
out	*acceleration*	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_lvl)
- 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`	*interval*	the interval between two calls of the function
`out`	*parameter*	the time indicator

Returns

the interval between two calls of the function

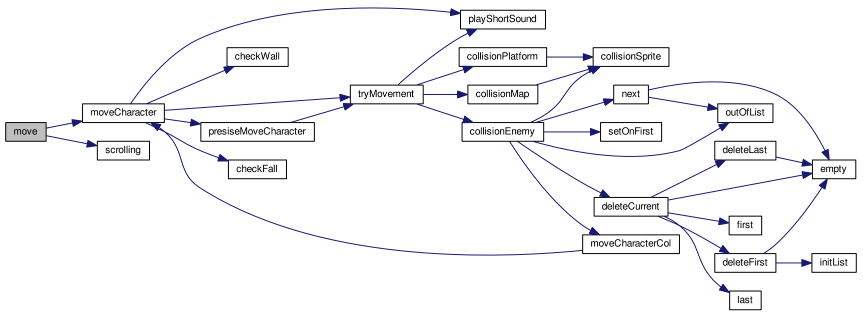
2.11.2.2 void move (float *move_left,* float *move_right,* int *jump,* Character ∗ *player,* Map ∗ *m,* float ∗ *speed,* int ∗ *acceleration,* list ∗ *l,* Sound ∗ *sound_sys,* platformSet ∗ *ps*)

moves the player and scrolls the screen if needed

Parameters

`in`	*move_left*	1 if move to the left
`in`	*move_right*	1 if move to the right
`in`	*jump*	1 if jump
`in`	*player*	the player
`in`	*m*	the game map
`in`	*speed*	the movement speed
`out`	*acceleration*	the acceleration of the player
`in,out`	*l*	the enemy list
`out`	*sound_sys*	the game sound system
`out`	*ps*	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

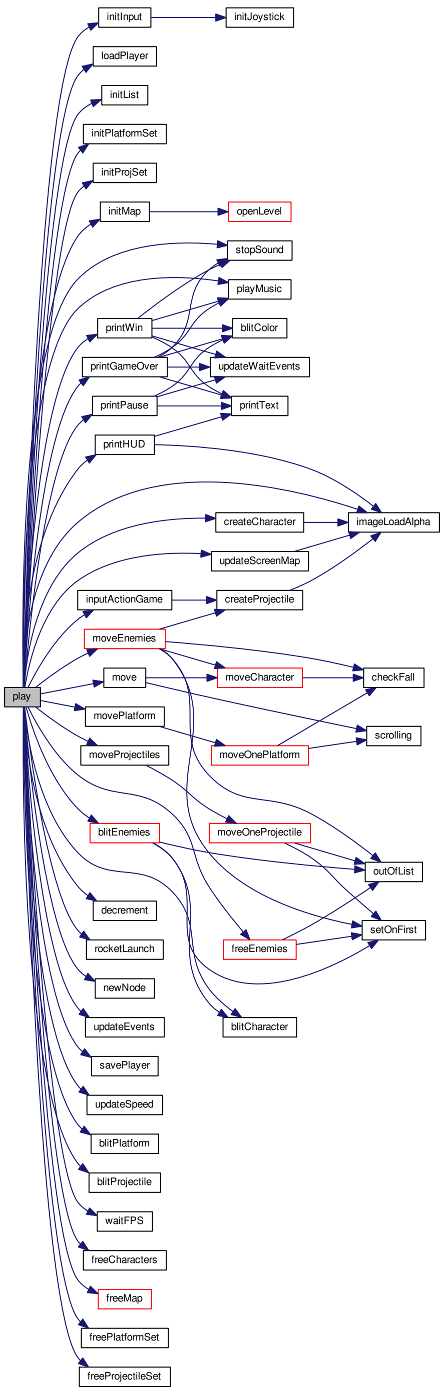
`in,out`	*screen*	the gamin screen
`in`	*level_name*	the name of the level to be played
`out`	*sound_sys*	the game sound system
`in`	*kc*	the keyboard configuration structure

`in,out`	*go*	the software main loop validation
`in,out`	*in*	the input gestion structure
`in,out`	*player*	the save player structure
`in`	*player_name*	the current player name
`in`	*nb_lvl*	the number of level
`in`	*currentLevel*	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:



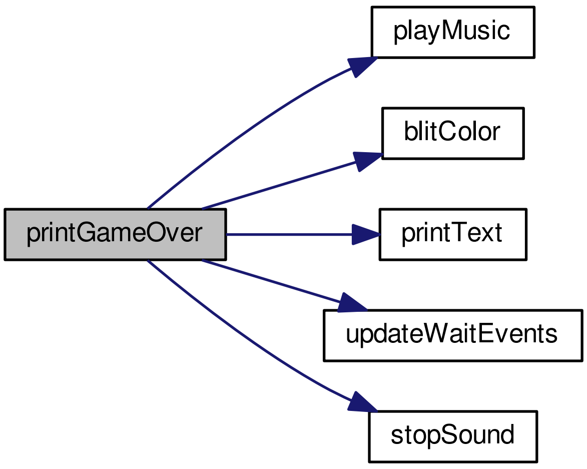
2.11.2.4 void printGameOver (SDL_Surface * *screen*, int * *go*, Input * *in*, Sound * *sound_sys*)

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

Parameters

out	*screen*	the game screen
out	*go*	the game function main loop validation
in,out	*in*	the input structure
out	*sound_sys*	the sound system

Here is the call graph for this function:



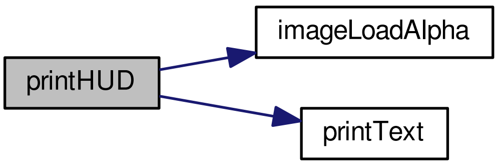
2.11.2.5 void printHUD (SDL_Surface ∗ *screen,* Character ∗ *player,* Map ∗ *m*)

print the player HUD on the screen

Parameters

in,out	*screen*	the game screen
in	*player*	the player
in	*m*	the game map

Here is the call graph for this function:



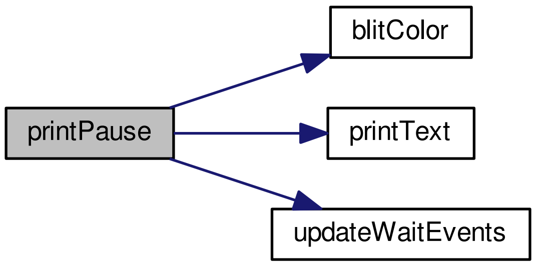
2.11.2.6 void printPause (SDL_Surface ∗ *screen*, Input ∗ *in*, int ∗ *time*, int ∗ *go*, SDLKey ∗ *kc*)

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

Parameters

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

Here is the call graph for this function:



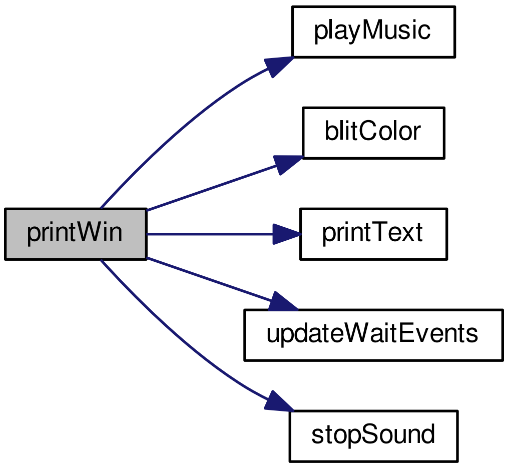
2.11.2.7 void printWin (SDL_Surface ∗ *screen*, int ∗ *go*, Input ∗ *in*, Sound ∗ *sound_sys*)

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

Parameters

out	*screen*	the game screen
out	*go*	the game function main loop validation
in,out	*in*	the input structure
out	*sound_sys*	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	*interval*	the interval between two calls of the function
out	*parameter*	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	*speed*	the player speed
out	*acceleration*	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

in	*red*	the red of the color
in	*green*	the green of the color
in	*blue*	the blue of the color
in	*alpha*	the transparency of the image file
in	*screen*	the screen

2.12.2.2 SDL_Surface ∗ imageLoad (char ∗ *file_name*)

Load an image

Parameters

in	*file_name*	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

in	*file_name*	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

`in`	*red*	the red of the color
`in`	*green*	the green of the color
`in`	*blue*	the blue of the color
`in`	*alpha*	the transparency of the image file
`in`	*screen*	the screen

2.13.2.2 SDL_Surface∗ imageLoad (char ∗ *file_name*)

Load an image

Parameters

`in`	*file_name*	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

in	*file_name*	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

out	*in*	the input structure

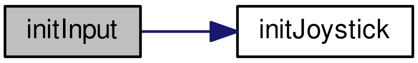
2.14.2.2 void initInput (**Input** ∗ *in*)

initialize the input structure

Parameters

`out`	*in*	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`	*in*	the joystick input structure to be initialized

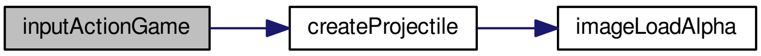
2.14.2.4 void inputActionGame (**Input** ∗ *in,* float ∗ *move_left,* float ∗ *move_right,* int ∗ *jump,* int ∗ *pause,* **Character** ∗ *player,* int ∗ *acceleration,* SDLKey ∗ *kc,* **projectileSet** ∗ *ps*)

perform action command by keyboard or joystick action

Parameters

`in`	*in*	the input structure
`out`	*move_left*	the left movement boolean
`out`	*move_right*	the right movement boolean
`out`	*jump*	the jump boolean
`out`	*pause*	the pause boolean
`in`	*player*	the Player
`in`	*acceleration*	the acceleration
`in`	*kc*	the keyboard configuration structure
`out`	*ps*	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	*in*	the input structure
out	*cursorPos*	cursor position
out	*play_level*	play level
in	*nb_lvl*	the number of level

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

recuperate keyboard/joystick input with a SDL_PollEvent

Parameters

out	*in*	the input structure
out	*go*	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	*in*	the input structure
in,out	*go*	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

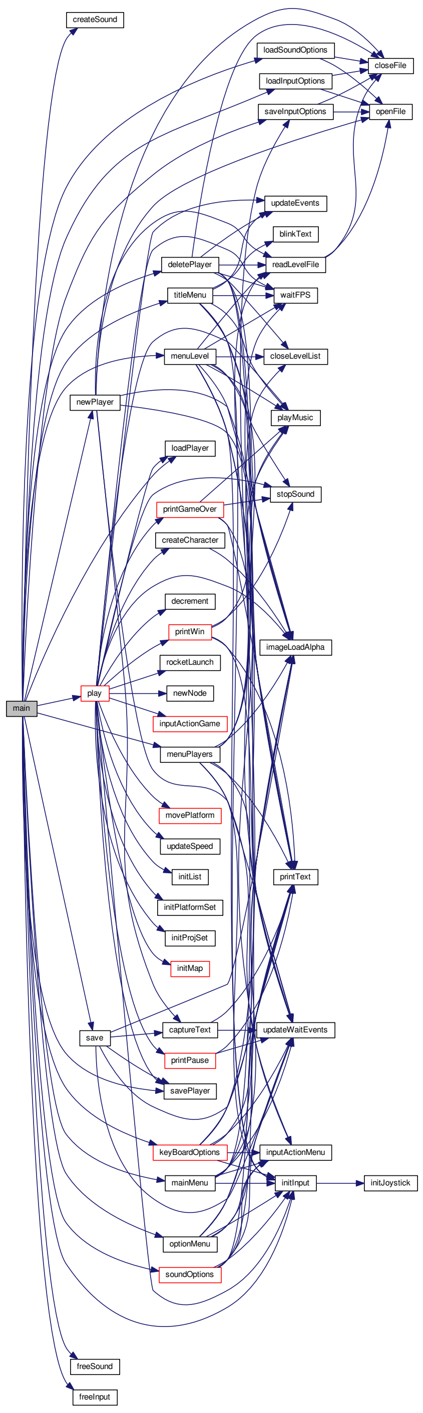
2.15.2.1 int main (int *argc,* char ∗ *argv[ ]*)

Main

Parameters

`in,out`	*argc*	argc
`in,out`	*argv*	argv

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 ∗l, 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

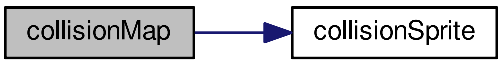
Parameters

`in`	*r*	SDL_Rect corresponding to the sprite
`in`	*m*	map
`in`	*type*	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:



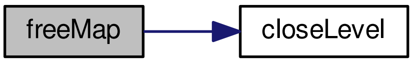
2.16.2.2 void freeMap (Map ∗ *m*)

free memory allocated to the map

Parameters

in,out	*m*	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

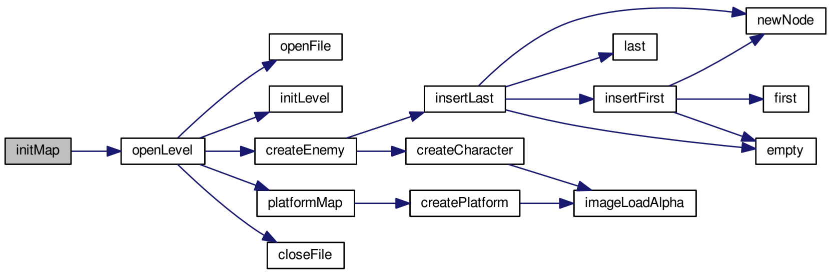
Parameters

in	*screen*	game screen
in	*level_name*	lvl name
out	*l*	the enemy list that stocks the enemies
out	*ps*	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	*m*	the lvl
in	*direction*	scrolling direction

in	*speed*	scrolling speed

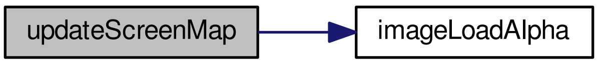
2.16.2.5 void updateScreenMap (SDL_Surface ∗ *screen*, Map ∗ *m*, char ∗ *tileset*)

update and display the map

Parameters

in,out	*screen*	
in	*m*	The map
in	*tileset*	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 ∗l, 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 beteewen a sprite and a "wall" of the map

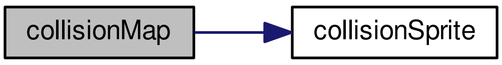
Parameters

in	*r*	SDL_Rect corresponding to the sprite
in	*m*	map
in	*type*	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:



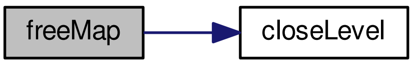
2.17.2.2 void freeMap (Map ∗ *m*)

free memory allocated to the map

Parameters

in,out	*m*	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

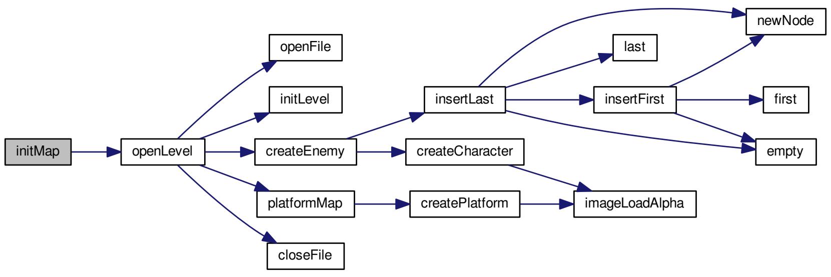
in	*screen*	game screen

in	*level_name*	lvl name
out	*l*	the enemy list that stocks the enemies
out	*ps*	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	*m*	the lvl
in	*direction*	scrolling direction
in	*speed*	scrolling speed

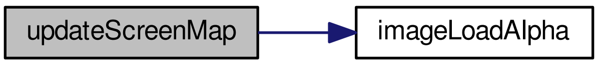
2.17.2.5 void updateScreenMap (SDL_Surface ∗ *screen,* **Map** ∗ *m,* char ∗ *tileset*)

update and display the map

Parameters

in,out	*screen*	
in	*m*	The map
in	*tileset*	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	*interval*	the interval between two callback of the function
in	*param*	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

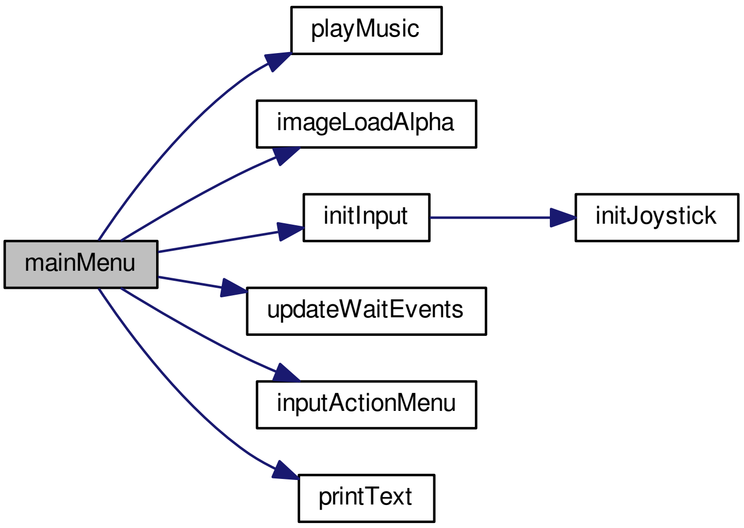
Parameters

out	*screen*	the game screen
in,out	*go*	main loop validation
out	*sound_sys*	sound system
in	*player_name*	the current player name
in,out	*in*	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

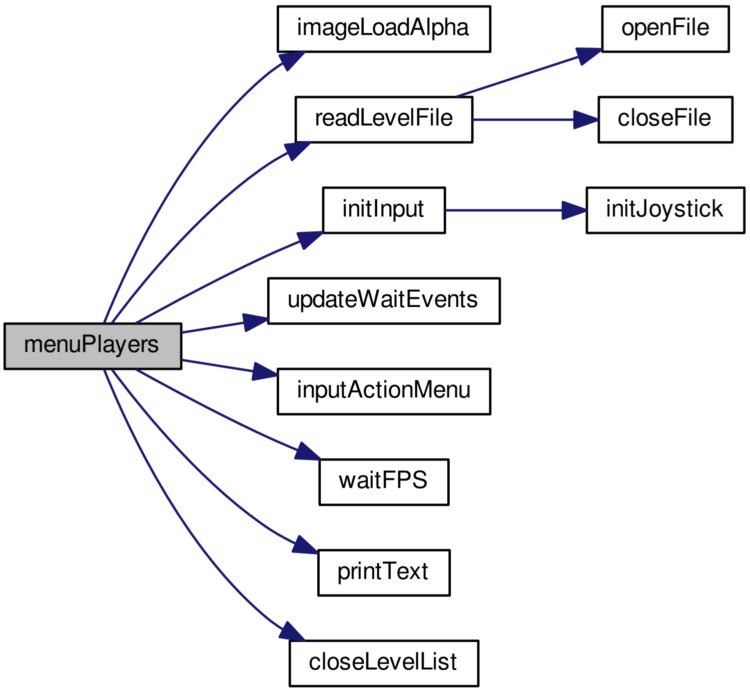
Parameters

`out`	*screen*	game screen
`out`	*player_name*	the name of the current player
`in,out`	*go*	main loop validation
`in,out`	*sound_sys*	the sound system
`in,out`	*in*	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

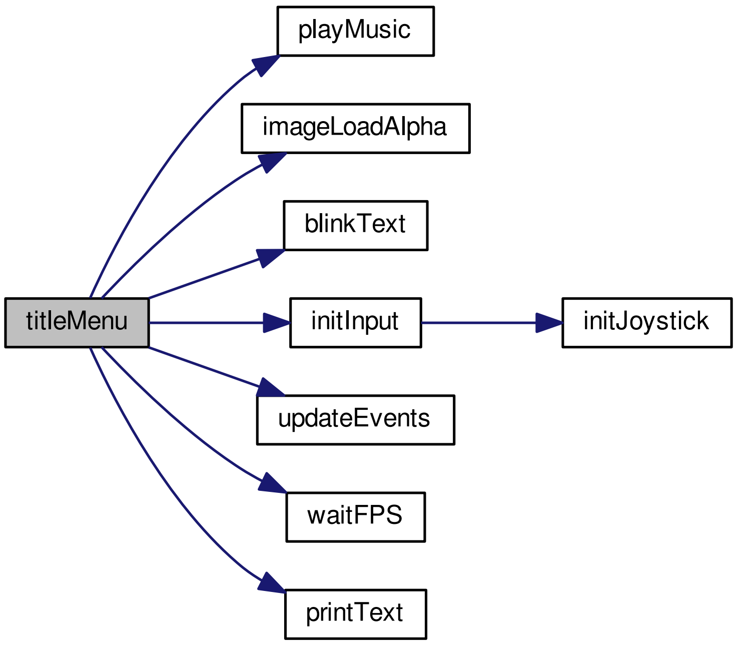
Parameters

out	*screen*	the game screen
in,out	*go*	main loop validation
out	*sound_sys*	sound
in,out	*in*	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	*interval*	the interval between two callback of the function
in	*param*	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

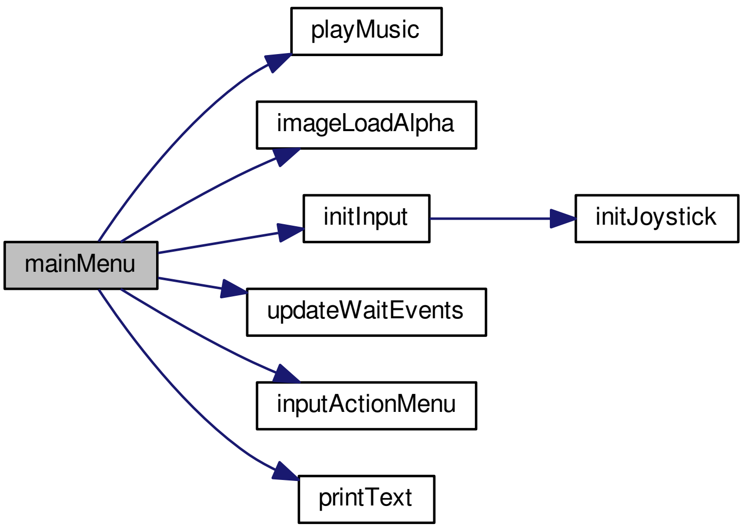
Parameters

out	*screen*	the game screen
in,out	*go*	main loop validation
out	*sound_sys*	sound system
in	*player_name*	the current player name
in,out	*in*	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

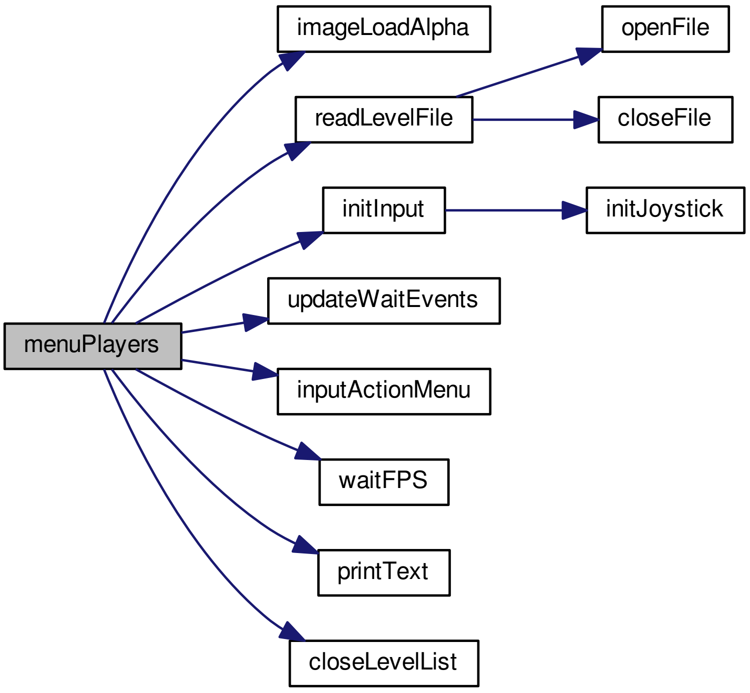
Parameters

`out`	*screen*	game screen
`out`	*player_name*	the name of the current player
`in,out`	*go*	main loop validation
`in,out`	*sound_sys*	the sound system
`in,out`	*in*	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

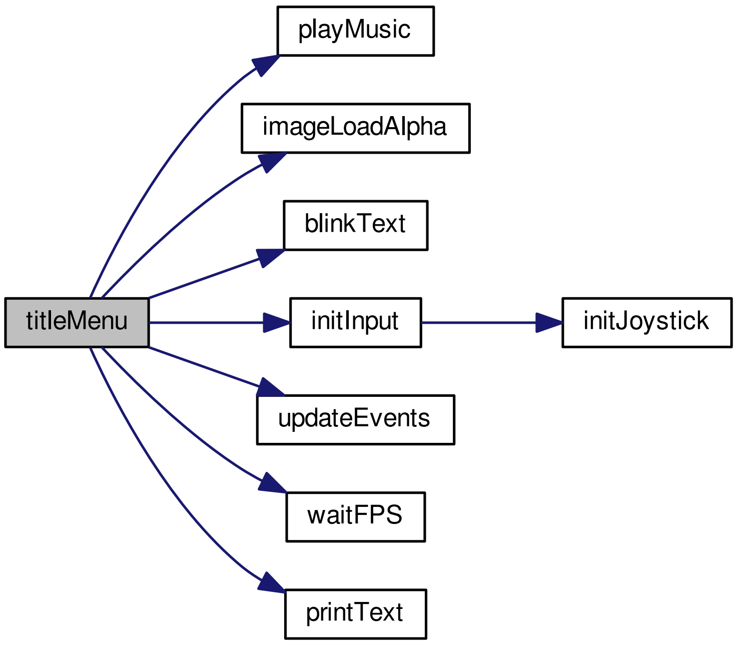
Parameters

`out`	*screen*	the game screen
`in,out`	*go*	main loop validation
`out`	*sound_sys*	sound
`in,out`	*in*	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_lvl*, Input ∗ *in*)

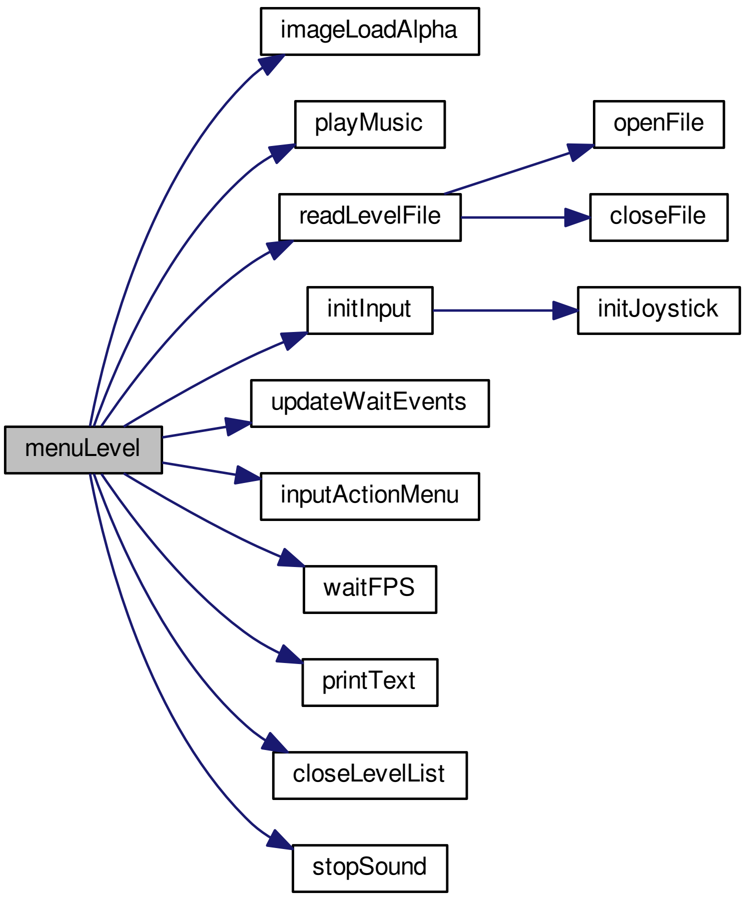
Parameters

`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_lvl*	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

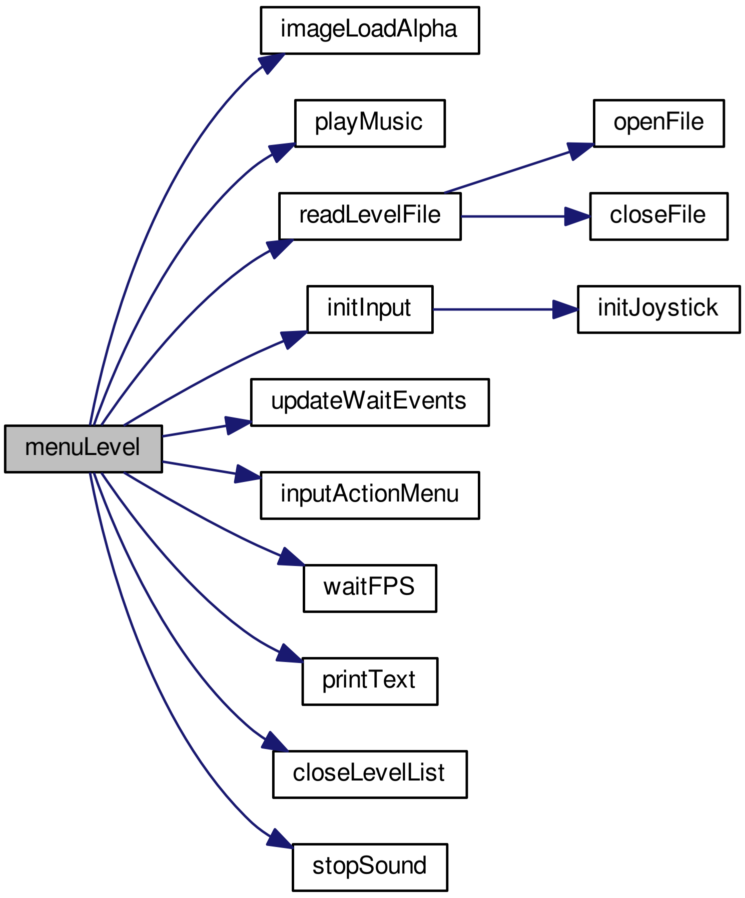
out	*screen*	game screen
out	*level_name*	the name of the level we will want to launch
in,out	*sound_sys*	the sound system
in,out	*player*	the player structure

`in,out`	*go*	the soft main loop validation
`in,out`	*player_name*	the player name
`in`	*nb_lvl*	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:



2.22 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

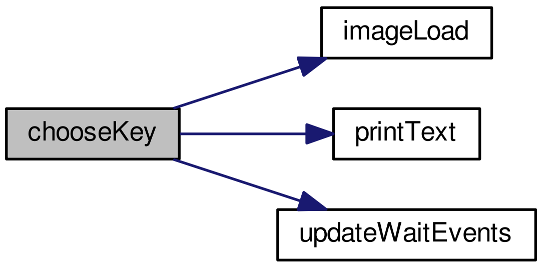
2.22.2.1 void chooseKey (SDL_Surface ∗ *screen,* Input ∗ *in,* char ∗ *action,* SDLKey ∗ *kc,* int *nb*)

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

Here is the call graph for this function:



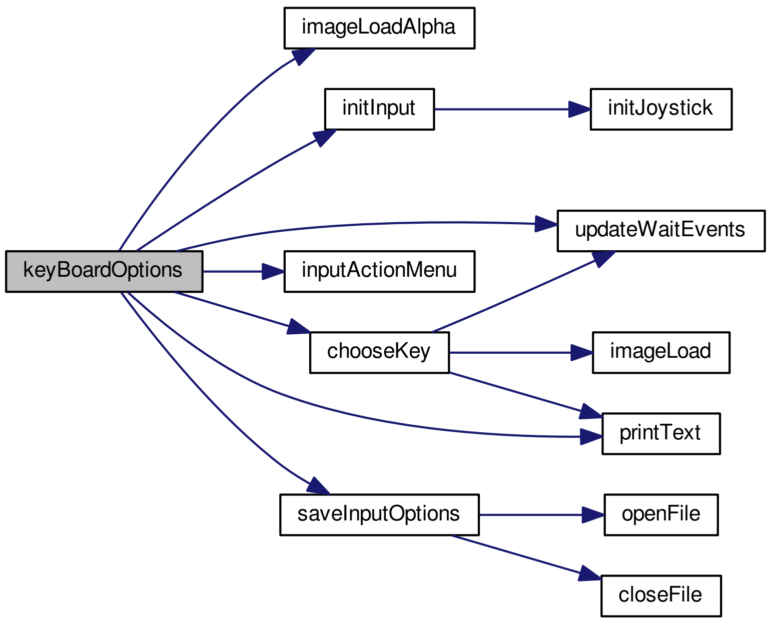
2.22.2.2 void keyBoardOptions (SDL_Surface ∗ *screen,* int ∗ *go,* SDLKey ∗ *kc,* Input ∗ *in,* char ∗ *player_name*)

print the keyboard options and deals with the user choises

Parameters

out	*screen*	the game screen
in,out	*go*	main loop validation
in,out	*kc*	the keyboard config structure
in,out	*in*	the input structure
in	*player_name*	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

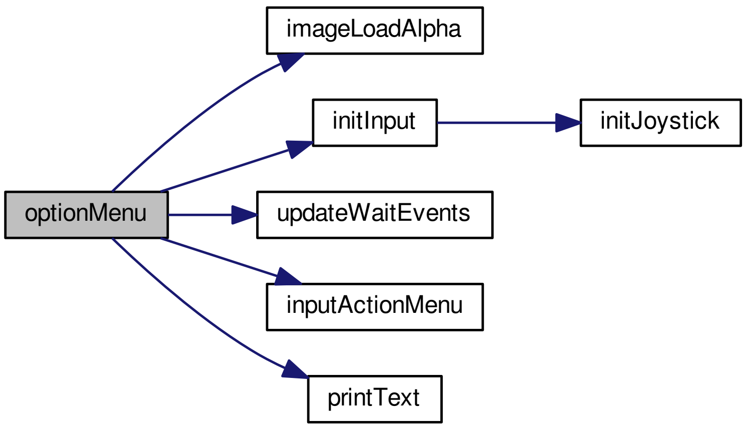
Parameters

out	*screen*	the game screen
in,out	*go*	main loop validation
in,out	*sound_sys*	sound system
in,out	*kc*	the keyboard configuration structure
in,out	*in*	the input structure

Returns

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

Here is the call graph for this function:



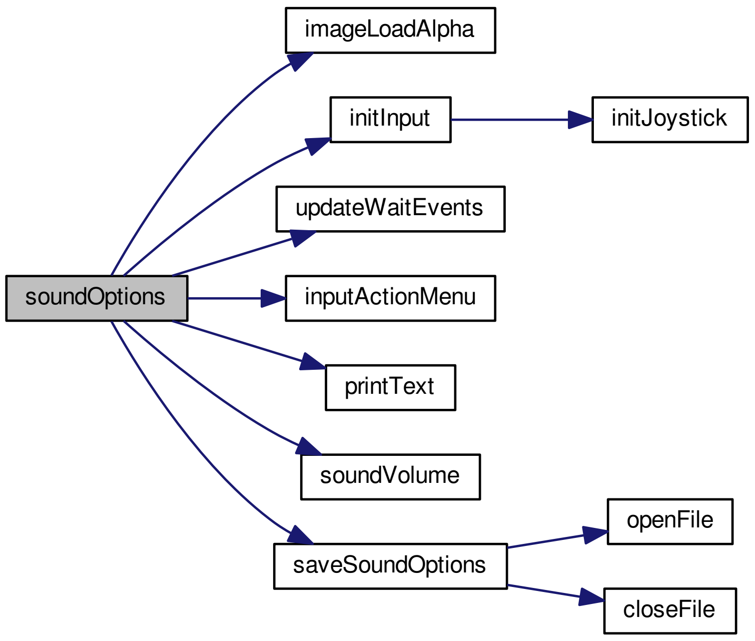
2.22.2.4 void soundOptions (SDL_Surface ∗ *screen,* int ∗ *go,* Sound ∗ *sound_sys,* Input ∗ *in*)

print the sound options and deals with the user choises

Parameters

`out`	*screen*	the game screen
`in,out`	*go*	main loop validation
`in,out`	*sound_sys*	sound system
`in,out`	*in*	the input structure
`in,out`	*sound_sys*	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

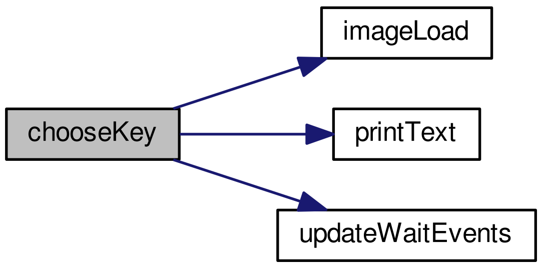
2.23.2.1 void chooseKey (SDL_Surface ∗ *screen*, Input ∗ *in*, char ∗ *action*, SDLKey ∗ *kc*, int *nb*)

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`	*screen*	the game screen
`in,out`	*in*	the input structure
`in`	*action*	the action which the key has to be choosen
`out`	*kc*	the keyboard configuration
`in`	*nb*	the number of the action

Here is the call graph for this function:



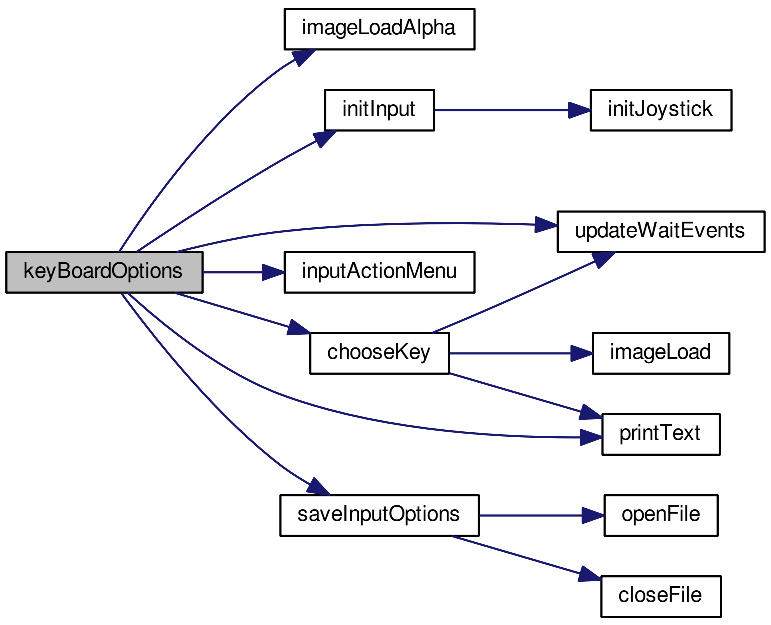
2.23.2.2 void keyBoardOptions (SDL_Surface ∗ *screen*, int ∗ *go*, SDLKey ∗ *kc*, Input ∗ *in*, char ∗ *player_name*)

print the keyboard options and deals with the user choises

Parameters

`out`	*screen*	the game screen
`in,out`	*go*	main loop validation
`in,out`	*kc*	the keyboard config structure
`in,out`	*in*	the input structure
`in`	*player_name*	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

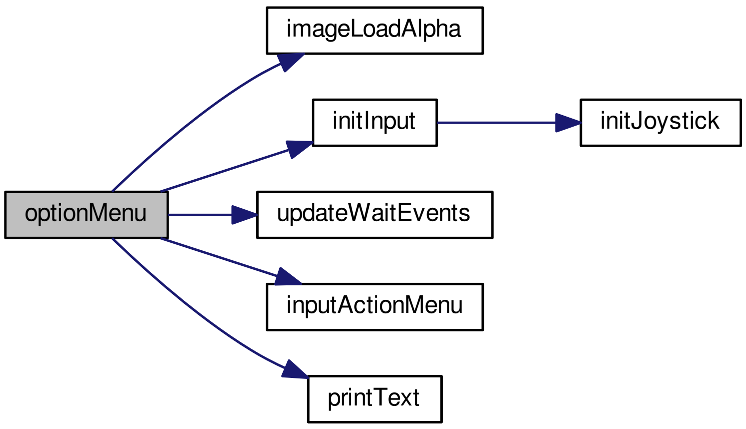
Parameters

`out`	*screen*	the game screen
`in,out`	*go*	main loop validation
`in,out`	*sound_sys*	sound system
`in,out`	*kc*	the keyboard configuration structure
`in,out`	*in*	the input structure

Returns

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

Here is the call graph for this function:



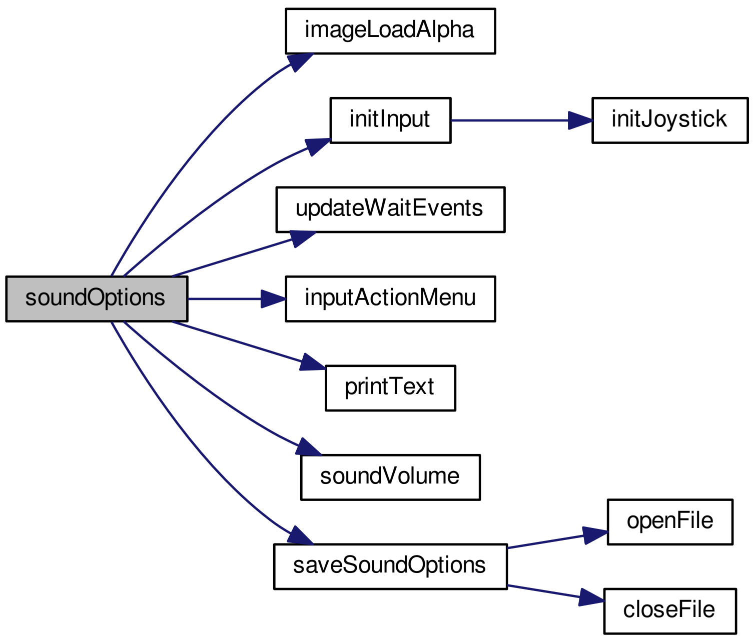
2.23.2.4 void soundOptions (SDL_Surface ∗ *screen,* int ∗ *go,* Sound ∗ *sound_sys,* Input ∗ *in*)

print the sound options and deals with the user choises

Parameters

out	*screen*	the game screen
in,out	*go*	main loop validation
in,out	*sound_sys*	sound system
in,out	*in*	the input structure
in,out	*sound_sys*	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`	*screen*	game screen
`in,out`	*ps*	the platform set
`in`	*m*	the current level map

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

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

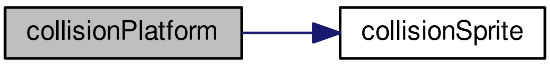
Parameters

`in,out`	*c*	the player
`in,out`	*ps*	the platform set
`in`	*futureLocation*	the tryMovement variabla to test the future position

Returns

1 if there is a collision, 0 if not

Here is the call graph for this function:



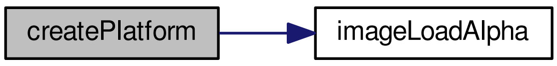
2.24.2.3 void createPlatform (platformSet ∗ *ps*, int *x1*, int *y1*, int *x2*, int *y2*)

creates of new platform and adds it to the platform set

Parameters

`in,out`	*ps*	the platform set
`in`	*x1*	the x low limit for deplacement
`in`	*x2*	the x high limit for deplacement
`in`	*y1*	the y low limit for deplacement
`in`	*y2*	the y high limit for deplacement

Here is the call graph for this function:



2.24.2.4 void freePlatformSet (**platformSet** ∗ *ps*)

free all the platforms

Parameters

`in,out`	*ps*	the platform set

2.24.2.5 void initPlatformSet (**platformSet** ∗ *ps*)

initialize a platform set

Parameters

`in`	*ps*	the platform set to be initialized

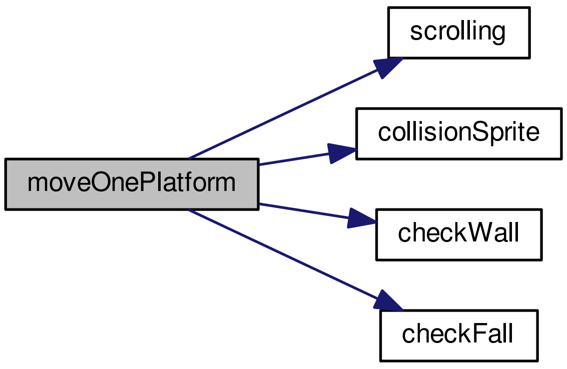
2.24.2.6 void moveOnePlatform (**Character** ∗ *c*, **platform** ∗ *p*, **list** ∗ *l*, int *nb*, **Map** ∗ *m*)

moves one platforms

Parameters

`in,out`	*c*	the player
`in,out`	*p*	the platform
`in,out`	*l*	the enemy list
`in`	*nb*	the number of the platform which is moved
`in`	*m*	the game map

Here is the call graph for this function:



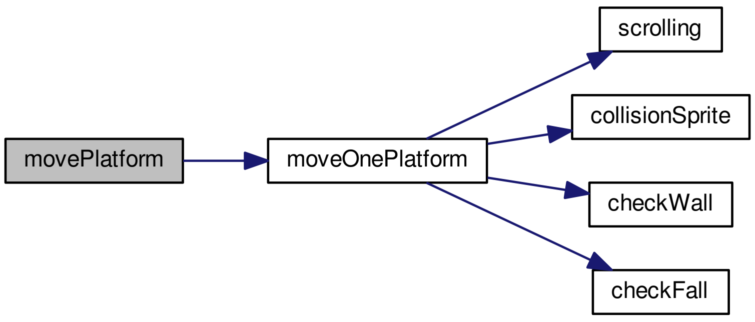
2.24.2.7 void movePlatform (**Character** ∗ *c*, **platformSet** ∗ *ps*, **list** ∗ *l*, **Map** ∗ *m*)

moves all the platforms

Parameters

`in,out`	*c*	the player
`in,out`	*ps*	the platform set
`in,out`	*l*	the enemy list
`in`	*m*	the game map

Here is the call graph for this function:



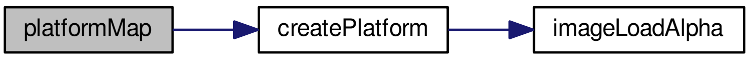
2.24.2.8 void platformMap (**platformSet** ∗ *ps*, SDL_Rect *array[ ]*, SDL_Rect *mark*, int *vert*)

takes a limit mark for a vertical deplacement 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	*ps*	the platform set
in,out	*array*	the array that stocks the limit marks
in	*mark*	the mark which has to be dealt with
in	*vert*	indicates if vertical movement(1) plarform 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`	*screen*	game screen
`in,out`	*ps*	the platform set
`in`	*m*	the current level map

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

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

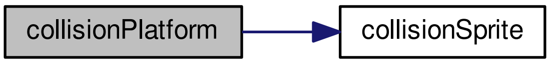
Parameters

`in,out`	*c*	the player
`in,out`	*ps*	the platform set
`in`	*futureLocation*	the tryMovement variabla to test the future position

Returns

1 if there is a collision, 0 if not

Here is the call graph for this function:



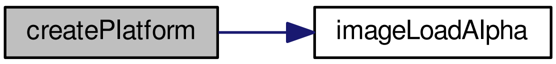
2.25.2.3 void createPlatform (**platformSet** ∗ *ps*, int *x1*, int *y1*, int *x2*, int *y2*)

creates of new platform and adds it to the platform set

Parameters

`in,out`	*ps*	the platform set
`in`	*x1*	the x low limit for deplacement
`in`	*x2*	the x high limit for deplacement
`in`	*y1*	the y low limit for deplacement
`in`	*y2*	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`	*ps*	the platform set

2.25.2.5 void initPlatformSet (**platformSet** ∗ *ps*)

initialize a platform set

Parameters

`in`	*ps*	the platform set to be initialized

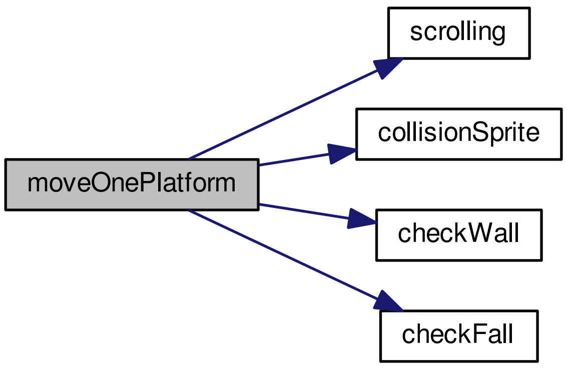
2.25.2.6 void moveOnePlatform (**Character** ∗ *c*, **platform** ∗ *p*, **list** ∗ *l*, int *nb*, **Map** ∗ *m*)

moves one platforms

Parameters

`in,out`	*c*	the player
`in,out`	*p*	the platform
`in,out`	*l*	the enemy list
`in`	*nb*	the number of the platform which is moved
`in`	*m*	the game map

Here is the call graph for this function:



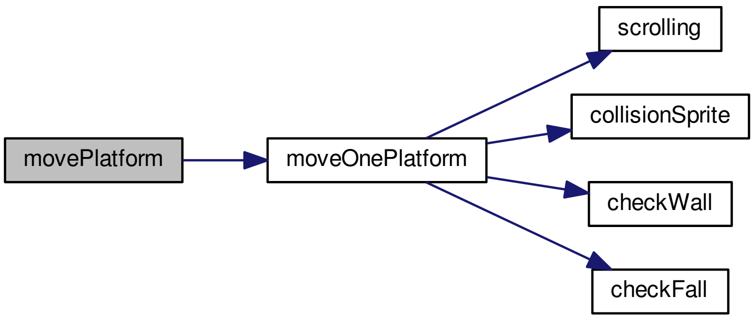
2.25.2.7 void movePlatform (**Character** ∗ *c*, **platformSet** ∗ *ps*, **list** ∗ *l*, **Map** ∗ *m*)

moves all the platforms

Parameters

`in,out`	*c*	the player
`in,out`	*ps*	the platform set
`in,out`	*l*	the enemy list
`in`	*m*	the game map

Here is the call graph for this function:



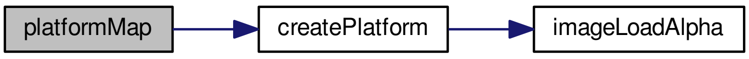
2.25.2.8 void platformMap (**platformSet** ∗ *ps,* SDL_Rect *array[ ],* SDL_Rect *mark,* int *vert*)

takes a limit mark for a vertical deplacement 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`	*ps*	the platform set
`in,out`	*array*	the array that stocks the limit marks
`in`	*mark*	the mark which has to be dealt with
`in`	*vert*	indicates if vertical movement(1) plarform 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

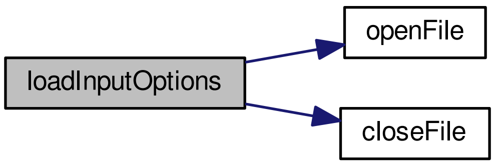
2.26.2.1 void loadInputOptions (char *player_name[ ]*, SDLKey ∗ *kc*, Input ∗ *in*)

load the input options from the player input config file

Parameters

`in`	*player_name*	the current player's name
`out`	*kc*	the keyboard configuration structure
`out`	*in*	the input structure

Here is the call graph for this function:



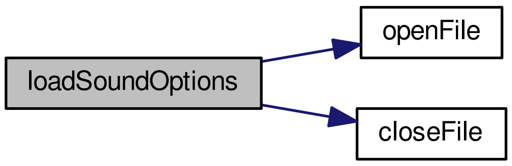
2.26.2.2 void loadSoundOptions (char *confFile[ ]*, Sound ∗ *soundSys*)

load the sound options from the sound config file

Parameters

`in`	*confFile*	the config file path
`out`	*soundSys*	the sound system

Here is the call graph for this function:



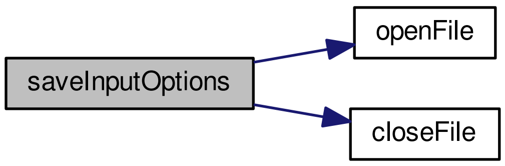
2.26.2.3 void saveInputOptions (char *player_name[ ]*, SDLKey ∗ *kc*, Input ∗ *in*)

save the input options to the player input config file

Parameters

in	*player_name*	the current player name
out	*kc*	the keyboard configuration structure
out	*in*	the input structure

Here is the call graph for this function:



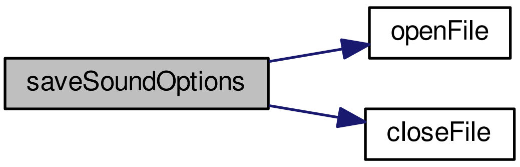
2.26.2.4 void saveSoundOptions (char *confFile[ ]*, Sound ∗ *soundSys*)

save the sound options to the config file

Parameters

in	*confFile*	the config file path
in	*soundSys*	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

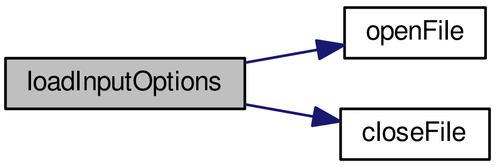
2.27.2.1 void loadInputOptions (char *player_name[],* SDLKey ∗ *kc,* Input ∗ *in*)

load the input options from the player input config file

Parameters

in	*player_name*	the current player's name
out	*kc*	the keyboard configuration structure
out	*in*	the input structure

Here is the call graph for this function:



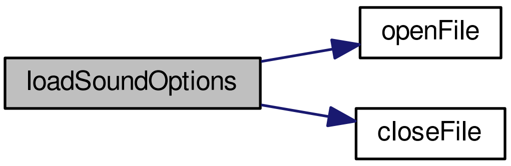
2.27.2.2 void loadSoundOptions (char *confFile[ ]*, Sound ∗ *soundSys*)

load the sound options from the sound config file

Parameters

in	*confFile*	the config file path
out	*soundSys*	the sound system

Here is the call graph for this function:



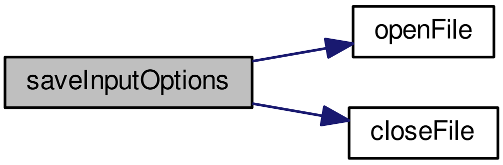
2.27.2.3 void saveInputOptions (char *player_name[ ]*, SDLKey ∗ *kc*, Input ∗ *in*)

save the input options to the player input config file

Parameters

in	*player_name*	the current player name
out	*kc*	the keyboard configuration structure
out	*in*	the input structure

Here is the call graph for this function:



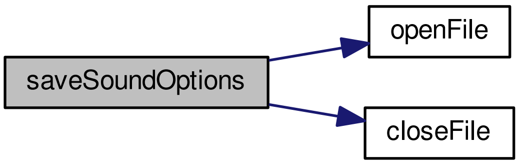
2.27.2.4 void saveSoundOptions (char *confFile[ ]*, Sound ∗ *soundSys*)

save the sound options to the config file

Parameters

`in`	*confFile*	the config file path
`in`	*soundSys*	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_F-ILE_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

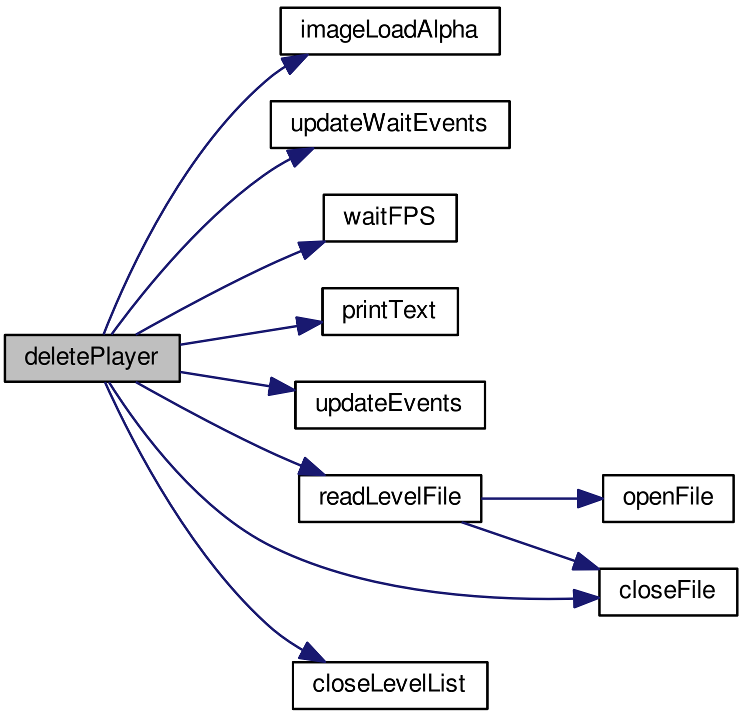
2.28.2.1 void deletePlayer (SDL_Surface ∗ *screen*, char *fileSave[MAX_SIZE_FILE_NAME]*, char *player_name[MAX_SIZE_FILE_NAME]*)

Delete the current player in the player list.

Parameters

`in,out`	*screen*	The screen of the game
`in,out`	*fileSave*	The path to the binary file containing the progression of each player
`in`	*player_name*	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

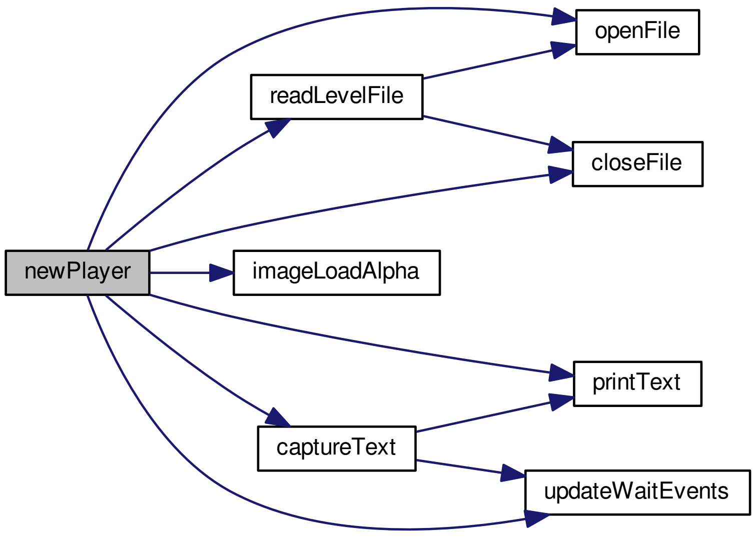
Parameters

`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:



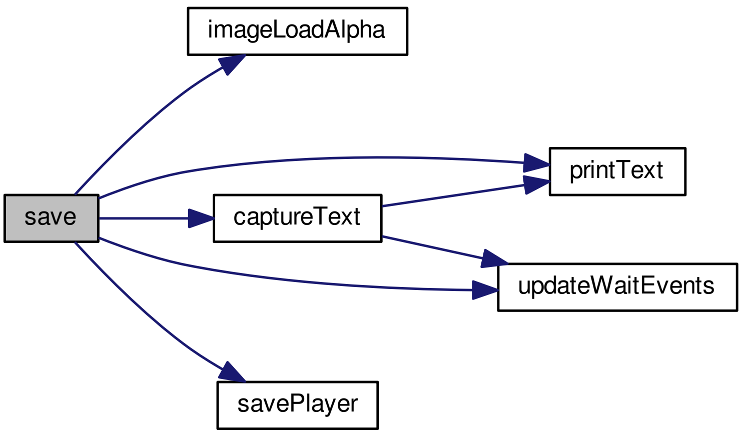
2.28.2.4 void save (SDL_Surface ∗ *screen*, char *fileSave[MAX_SIZE_FILE_NAME]*, char *player_name[MAX_SIZE_FILE_NAME]*, **Player** ∗ *player*, int ∗ *go*)

Display the interface to save the player progression

Parameters

`in,out`	*screen*	The screen of the game
`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 is stored
`out`	*go*	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`	*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 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_F-ILE_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

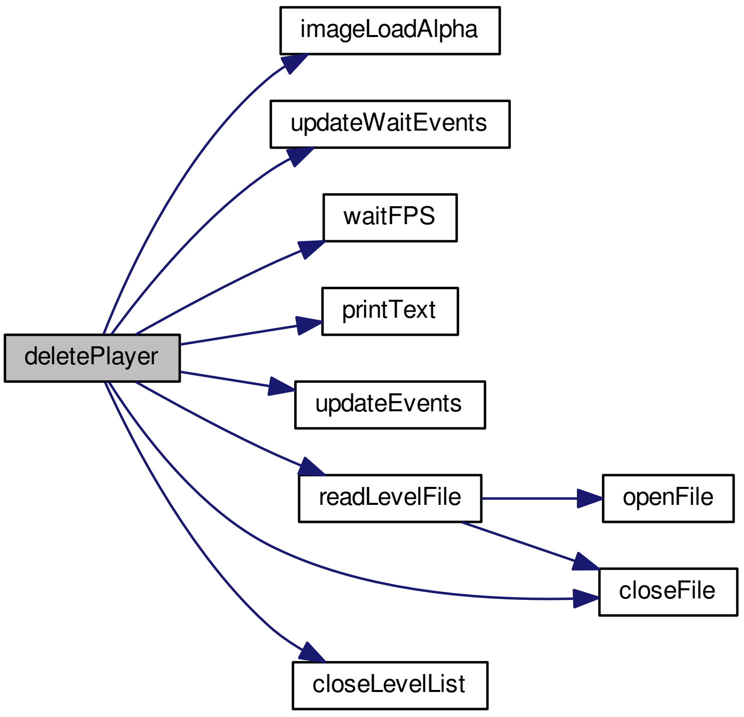
2.29.2.1 void deletePlayer (SDL_Surface ∗ *screen*, char *fileSave[MAX_SIZE_FILE_NAME]*, char *player_name[MAX_SIZE_FILE_NAME]*)

Delete the current player in the player list.

Parameters

in,out	*screen*	The screen of the game
in,out	*fileSave*	The path to the binary file containing the progression of each player
in	*player_name*	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	*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

2.29.2.3 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

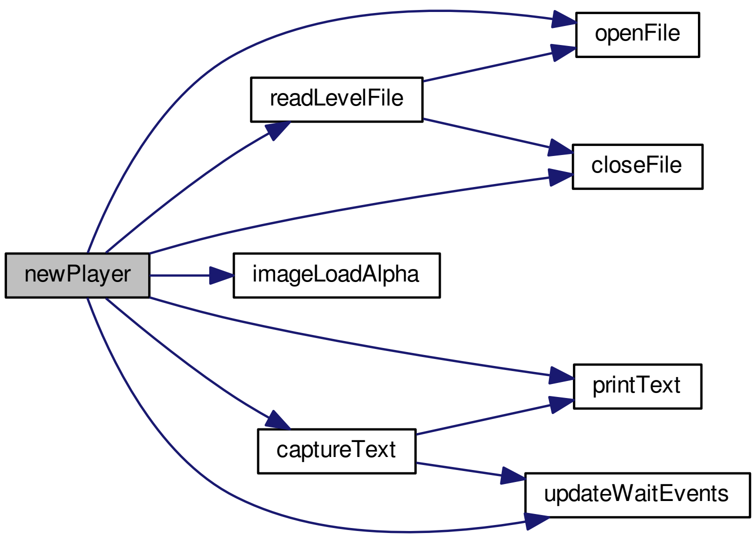
in,out	*screen*	The screen of the game

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:



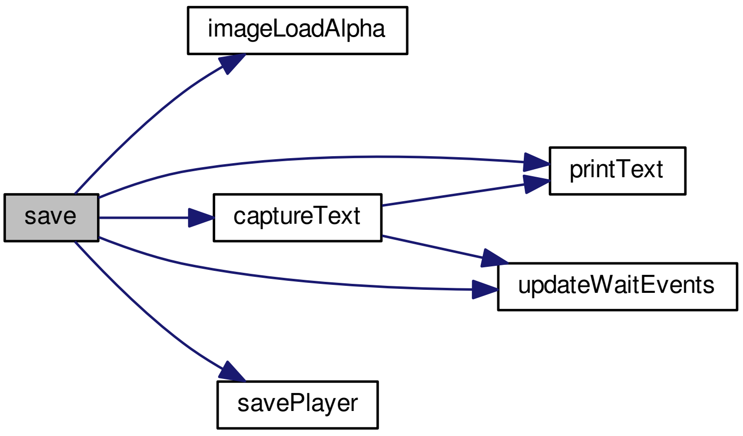
2.29.2.4 void save (SDL_Surface ∗ *screen*, char *fileSave[MAX_SIZE_FILE_NAME]*, char *player_name[MAX_SIZE_FILE_NAME]*, Player ∗ *player*, int ∗ *go*)

Display the interface to save the player progression

Parameters

in,out	*screen*	The screen of the game
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 is stored
out	*go*	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	*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 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`	*screen*	game screen
`in,out`	*ps*	the projectile set
`in`	*m*	the current level map

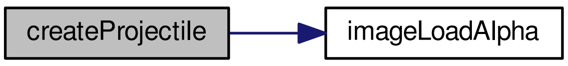
2.30.2.2 void createProjectile (projectileSet * *projSet*, char * *pathSprite*, int *dir*, int *x*, int *y*, int *fromNPC*)

creates a projectile and adds it to the projectile set

Parameters

`in,out`	*projSet*	the projectile set
`in,out`	*pathSprite*	the path for the sprite
`in`	*dir*	the projectile's direction
`in`	*x*	the start absciss coordinate of the projectile
`in`	*y*	the start ordinate coordinate of the projectile
`in`	*fromNPC*	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`	*ps*	the projectile Set
`in`	*nb*	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`	*ps*	the projectile set

2.30.2.5 void initProjSet (projectileSet ∗ *projSet*)

initialize a projectile set

Parameters

`out`	*projSet*	the projectile set to be initialized

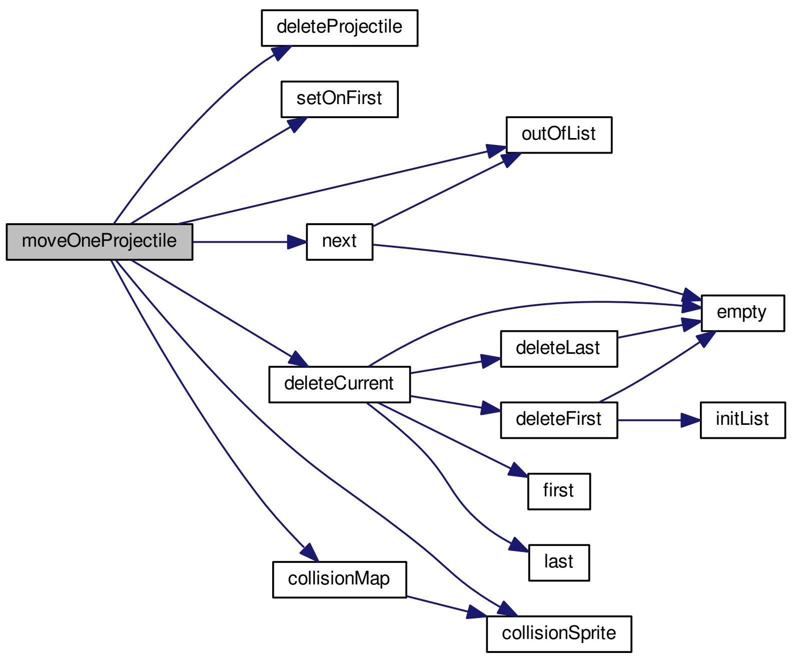
2.30.2.6 void moveOneProjectile (Character ∗ *c*, Map ∗ *m*, projectileSet ∗ *ps*, list ∗ *l*, int *nb*)

moves one projectile

Parameters

`in,out`	*c*	the player
`in,out`	*m*	the game map
`in,out`	*ps*	the projectileSet
`in,out`	*l*	the enemy list
`in`	*nb*	the number of the projectile which is moved

Here is the call graph for this function:



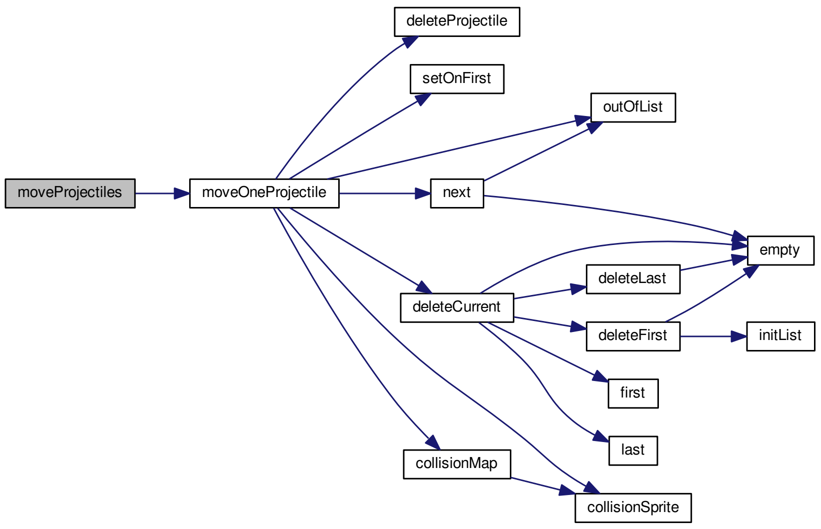
2.30.2.7 void moveProjectiles (Character ∗ *c*, Map ∗ *m*, projectileSet ∗ *ps*, list ∗ *enemyList*)

moves all the projectiles

Parameters

`in,out`	*c*	the player
`in,out`	*m*	the game map
`in,out`	*ps*	the projectile set
`in,out`	*enemyList*	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`	*screen*	game screen
`in,out`	*ps*	the projectile set
`in`	*m*	the current level map

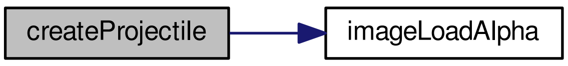
2.31.2.2 void createProjectile (projectileSet ∗ *projSet*, char ∗ *pathSprite*, int *dir*, int *x*, int *y*, int *fromNPC*)

creates a projectile and adds it to the projectile set

Parameters

`in,out`	*projSet*	the projectile set
`in,out`	*pathSprite*	the path for the sprite
`in`	*dir*	the projectile's direction
`in`	*x*	the start absciss coordinate of the projectile
`in`	*y*	the start ordinate coordinate of the projectile
`in`	*fromNPC*	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`	*ps*	the projectile Set
`in`	*nb*	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	*ps*	the projectile set

2.31.2.5 void initProjSet (**projectileSet** ∗ *projSet*)

initialize a projectile set

Parameters

out	*projSet*	the projectile set to be initialized

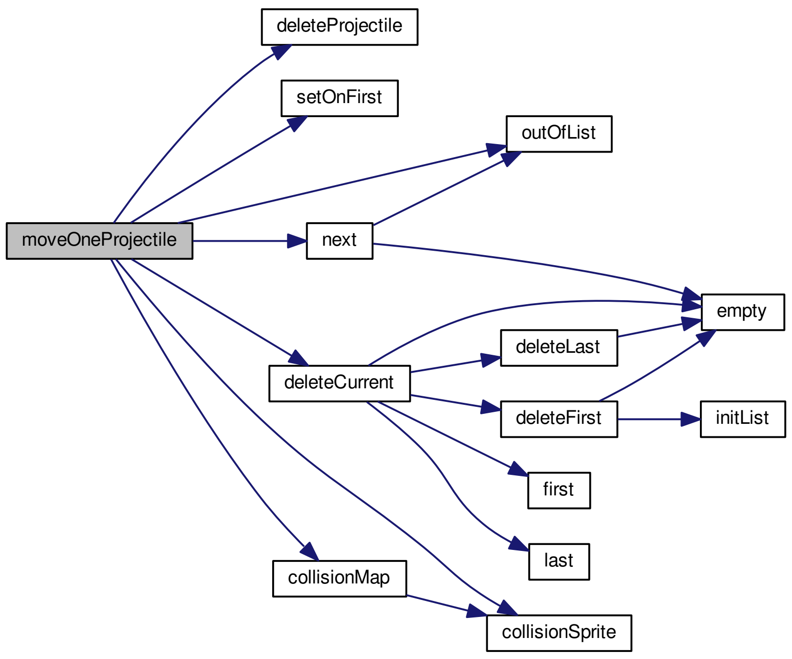
2.31.2.6 void moveOneProjectile (**Character** ∗ *c*, **Map** ∗ *m*, **projectileSet** ∗ *ps*, **list** ∗ *l*, int *nb*)

moves one projectile

Parameters

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

Here is the call graph for this function:



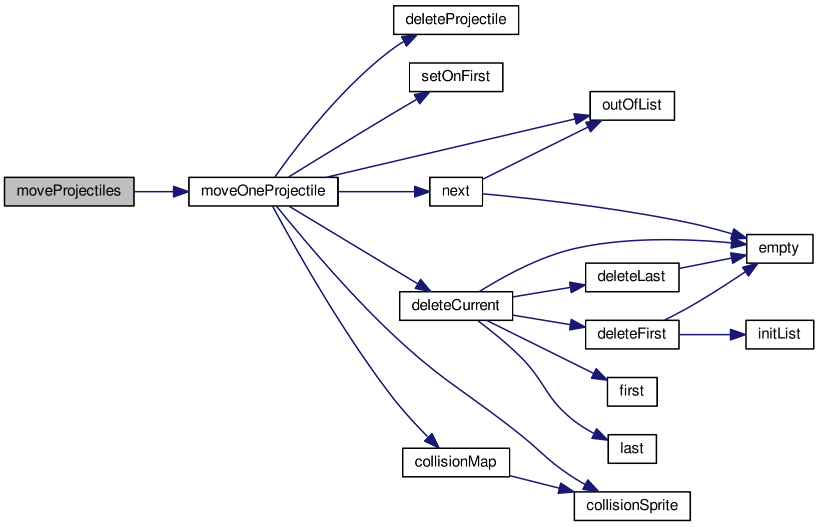
2.31.2.7 void moveProjectiles (**Character** ∗ *c*, **Map** ∗ *m*, **projectileSet** ∗ *ps*, **list** ∗ *enemyList*)

moves all the projectiles

Parameters

`in,out`	*c*	the player
`in,out`	*m*	the game map
`in,out`	*ps*	the projectile set
`in,out`	*enemyList*	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`	*previous_time*	The previous time
`in,out`	*current_time*	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`	*previous_time*	The previous time
`in,out`	*current_time*	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`	*file*	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`	*file*	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`	*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]

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

stop the sound

Parameters

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

2.36 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

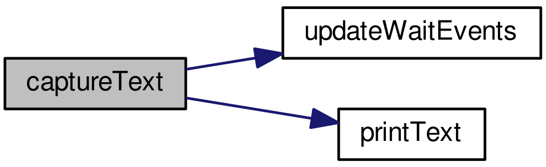
2.37.2.1 void captureText (SDL_Surface ∗ *screen*, SDL_Rect *posText*, char ∗ *text*, int *text_length*, int *r*, int *g*, int *b*, char ∗ *font*, int *text_size*, int ∗ *go*, int ∗ *ret*)

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

Parameters

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

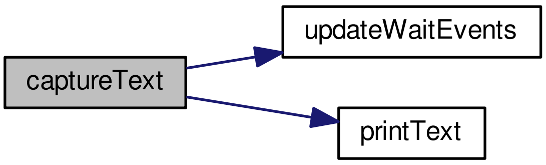
2.38.2.1 void captureText (SDL_Surface ∗ *screen,* SDL_Rect *posText,* char ∗ *text,* int *text_length,* int *r,* int *g,* int *b,* char ∗ *font,* int *text_size,* int ∗ *go,* int ∗ *ret*)

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

Parameters

out	*screen*	The screen of the game
in	*posText*	The position of the text. If NULL, the text is centered
out	*text*	The text to display
in	*r*	red value
in	*g*	green value
in	*b*	blue value
in	*text_length*	the text length
in	*font*	The path to the font file
in	*text_size*	The text size
out	*go*	The main loop validation
in,out	*ret*	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`	*screen*	The screen of the game
`in`	*posText*	The position of the text. If NULL, the text is centered
`in`	*text*	The text to display
`in`	*r*	red value
`in`	*g*	green value
`in`	*b*	blue value
`in`	*font*	The path to the font file
`in`	*ptSize*	The text size
`in`	*mode*	The writing mode : 0 (Solid), 1 (Blended)

Index