

My Project

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 sprite_s Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 h	5
3.1.2.2 w	5
3.1.2.3 x	6
3.1.2.4 y	6
3.2 textures_s Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 background	6
3.2.2.2 ligne_arrive	7
3.2.2.3 meteorite	7
3.2.2.4 vaisseau	7
3.3 world_s Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Member Data Documentation	7
3.3.2.1 gameover	8
3.3.2.2 ligne_arrive	8
3.3.2.3 mur	8
3.3.2.4 vaisseau	8
4 File Documentation	9
4.1 main.c File Reference	9
4.1.1 Detailed Description	11
4.1.2 Function Documentation	11
4.1.2.1 apply_background()	11
4.1.2.2 apply_meteorite()	11
4.1.2.3 apply_sprite()	12
4.1.2.4 clean()	12
4.1.2.5 clean_data()	12
4.1.2.6 clean_textures()	13
4.1.2.7 handle_events()	13
4.1.2.8 init()	13
4.1.2.9 init_data()	14
4.1.2.10 init_sprite()	14

4.1.2.11 init_textures()	14
4.1.2.12 is_game_over()	15
4.1.2.13 print_sprite()	15
4.1.2.14 refresh_graphics()	15
4.1.2.15 update_data()	16
4.2 sdl2-light.c File Reference	16
4.2.1 Detailed Description	16
4.2.2 Function Documentation	17
4.2.2.1 apply_texture()	17
4.2.2.2 clean_sdl()	17
4.2.2.3 clean_texture()	18
4.2.2.4 clear_renderer()	18
4.2.2.5 init_sdl()	18
4.2.2.6 load_image()	19
4.2.2.7 pause()	19
4.2.2.8 update_screen()	19
4.3 sdl2-light.h File Reference	20
4.3.1 Detailed Description	20
4.3.2 Function Documentation	20
4.3.2.1 apply_texture()	20
4.3.2.2 clean_sdl()	21
4.3.2.3 clean_texture()	21
4.3.2.4 clear_renderer()	21
4.3.2.5 init_sdl()	22
4.3.2.6 load_image()	22
4.3.2.7 pause()	22
4.3.2.8 update_screen()	23
Index	25

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

sprite_s	Représentation d'un sprite du jeu	5
textures_s	Représentation pour stocker les textures nécessaires à l'affichage graphique	6
world_s	Représentation du monde du jeu	7

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

main.c	Programme principal initial du niveau 1	9
sdl2-light.c	Sur-couche de SDL2 pour simplifier son utilisation pour le projet	16
sdl2-light.h	En-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet	20

Chapter 3

Class Documentation

3.1 `sprite_s` Struct Reference

Représentation d'un sprite du jeu.

Public Attributes

- int `x`
- int `y`
- int `h`
- int `w`

3.1.1 Detailed Description

Représentation d'un sprite du jeu.

3.1.2 Member Data Documentation

3.1.2.1 `h`

```
int sprite_s::h
```

Champ indiquant la hauteur

3.1.2.2 `w`

```
int sprite_s::w
```

Champ indiquant la largeur

3.1.2.3 x

```
int sprite_s::x
```

Champ indiquant l'abscisse de la position

3.1.2.4 y

```
int sprite_s::y
```

Champ indiquant l'ordonnée de la position

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

- [main.c](#)

3.2 textures_s Struct Reference

Représentation pour stocker les textures nécessaires à l'affichage graphique.

Public Attributes

- SDL_Texture * [background](#)
- SDL_Texture * [vaisseau](#)
- SDL_Texture * [ligne_arrive](#)
- SDL_Texture * [meteorite](#)

3.2.1 Detailed Description

Représentation pour stocker les textures nécessaires à l'affichage graphique.

3.2.2 Member Data Documentation

3.2.2.1 background

```
SDL_Texture* textures_s::background
```

Texture liée à l'image du fond de l'écran.

3.2.2.2 ligne_arrive

```
SDL_Texture* textures_s::ligne_arrive
```

Texture liée à la ligne d'arrivée

3.2.2.3 meteorite

```
SDL_Texture* textures_s::meteorite
```

Texture liée à une météorite

3.2.2.4 vaisseau

```
SDL_Texture* textures_s::vaisseau
```

Texture liée à l'image du vaisseau.

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

- [main.c](#)

3.3 world_s Struct Reference

Représentation du monde du jeu.

Public Attributes

- [sprite_t vaisseau](#)
- [sprite_t ligne_arrive](#)
- [sprite_t mur](#)
- int **vy**
- int [gameover](#)

3.3.1 Detailed Description

Représentation du monde du jeu.

3.3.2 Member Data Documentation

3.3.2.1 gameover

```
int world_s::gameover
```

Champ indiquant si l'on est à la fin du jeu

3.3.2.2 ligne_arrive

```
sprite_t world_s::ligne_arrive
```

Champ représentant la ligne d'arrivée

3.3.2.3 mur

```
sprite_t world_s::mur
```

Champ représentant un mur de météorites

3.3.2.4 vaisseau

```
sprite_t world_s::vaisseau
```

Champ représentant le sprite vaisseau

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

- [main.c](#)

Chapter 4

File Documentation

4.1 main.c File Reference

Programme principal initial du niveau 1.

```
#include "sdl2-light.h"  
#include <stdio.h>
```

Classes

- struct [sprite_s](#)
Représentation d'un sprite du jeu.
- struct [textures_s](#)
Représentation pour stocker les textures nécessaires à l'affichage graphique.
- struct [world_s](#)
Représentation du monde du jeu.

Macros

- #define [SCREEN_WIDTH](#) 300
Largeur de l'écran de jeu.
- #define [SCREEN_HEIGHT](#) 480
Hauteur de l'écran de jeu.
- #define [SHIP_SIZE](#) 32
Taille d'un vaisseau.
- #define [METEORITE_SIZE](#) 32
Taille d'un météorite.
- #define [FINISH_LINE_HEIGHT](#) 10
Hauteur de la ligne d'arrivée.
- #define [MOVING_STEP](#) 10
Pas de déplacement horizontal du vaisseau.
- #define [INITIAL_SPEED](#) 2
Vitesse initiale de déplacement vertical des éléments du jeu.

Typedefs

- typedef struct [sprite_s](#) [sprite_t](#)
Type qui correspond aux données du sprite.
- typedef struct [textures_s](#) [textures_t](#)
Type qui correspond aux textures du jeu.
- typedef struct [world_s](#) [world_t](#)
Type qui correspond aux données du monde.

Functions

- void [apply_sprite](#) (SDL_Renderer *renderer, SDL_Texture *texture, [sprite_t](#) *sprite)
La fonction initialise la position du sprite.
- void [init_sprite](#) ([sprite_t](#) *sprite, int x, int y, int w, int h)
La fonction initialise la position du sprite.
- void [print_sprite](#) ([sprite_t](#) sprite)
La fonction affiche la position du sprite.
- void [init_data](#) ([world_t](#) *world)
La fonction initialise les données du monde du jeu.
- void [clean_data](#) ([world_t](#) *world)
La fonction nettoie les données du monde.
- int [is_game_over](#) ([world_t](#) *world)
La fonction indique si le jeu est fini en fonction des données du monde.
- void [update_data](#) ([world_t](#) *world)
La fonction met à jour les données en tenant compte de la physique du monde.
- void [handle_events](#) (SDL_Event *event, [world_t](#) *world)
La fonction gère les événements ayant eu lieu et qui n'ont pas encore été traités.
- void [clean_textures](#) ([textures_t](#) *textures)
La fonction nettoie les textures.
- void [init_textures](#) (SDL_Renderer *renderer, [textures_t](#) *textures)
La fonction initialise les textures nécessaires à l'affichage graphique du jeu.
- void [apply_background](#) (SDL_Renderer *renderer, SDL_Texture *texture)
La fonction applique la texture du fond sur le renderer lié à l'écran de jeu.
- void [apply_meteorite](#) (SDL_Renderer *renderer, [world_t](#) *world, [textures_t](#) *textures, int hauteur, int largeur)
La fonction applique la texture des météorites sur le mur.
- void [refresh_graphics](#) (SDL_Renderer *renderer, [world_t](#) *world, [textures_t](#) *textures)
La fonction rafraichit l'écran en fonction de l'état des données du monde.
- void [clean](#) (SDL_Window *window, SDL_Renderer *renderer, [textures_t](#) *textures, [world_t](#) *world)
fonction qui nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des textures, nettoyage des données
- void [init](#) (SDL_Window **window, SDL_Renderer **renderer, [textures_t](#) *textures, [world_t](#) *world)
fonction qui initialise le jeu: initialisation de la partie graphique (SDL), chargement des textures, initialisation des données
- int [main](#) (int argc, char *args[])
programme principal qui implémente la boucle du jeu

4.1.1 Detailed Description

Programme principal initial du niveau 1.

Author

Mathieu Constant

Version

1.0

Date

18 mars 2021

4.1.2 Function Documentation

4.1.2.1 apply_background()

```
void apply_background (
    SDL_Renderer * renderer,
    SDL_Texture * texture )
```

La fonction applique la texture du fond sur le renderer lié à l'écran de jeu.

Parameters

<i>renderer</i>	le renderer
<i>texture</i>	la texture liée au fond

4.1.2.2 apply_meteorite()

```
void apply_meteorite (
    SDL_Renderer * renderer,
    world_t * world,
    textures_t * textures,
    int hauteur,
    int largeur )
```

La fonction applique la texture des météorites sur le mur.

Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>textures</i>	les textures

4.1.2.3 apply_sprite()

```
void apply_sprite (
    SDL_Renderer * renderer,
    SDL_Texture * texture,
    sprite_t * sprite )
```

La fonction initialise la position du sprite.

Parameters

<i>sprite</i>	vaisseau
<i>x</i>	abscisse de la position du sprite
<i>y</i>	ordonnée de la position du sprite
<i>w</i>	largeur de la position du sprite
<i>h</i>	hauteur de la position du sprite

4.1.2.4 clean()

```
void clean (
    SDL_Window * window,
    SDL_Renderer * renderer,
    textures_t * textures,
    world_t * world )
```

fonction qui nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des textures, nettoyage des données

Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>textures</i>	les textures
<i>world</i>	le monde

4.1.2.5 clean_data()

```
void clean_data (
    world_t * world )
```

La fonction nettoie les données du monde.

Parameters

<i>world</i>	les données du monde
--------------	----------------------

4.1.2.6 clean_textures()

```
void clean_textures (
    textures_t * textures )
```

La fonction nettoie les textures.

Parameters

<i>textures</i>	les textures
-----------------	--------------

4.1.2.7 handle_events()

```
void handle_events (
    SDL_Event * event,
    world_t * world )
```

La fonction gère les évènements ayant eu lieu et qui n'ont pas encore été traités.

Parameters

<i>event</i>	paramètre qui contient les événements
<i>world</i>	les données du monde

4.1.2.8 init()

```
void init (
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    textures_t * textures,
    world_t * world )
```

fonction qui initialise le jeu: initialisation de la partie graphique (SDL), chargement des textures, initialisation des données

Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>textures</i>	les textures
<i>world</i>	le monde

4.1.2.9 init_data()

```
void init_data (
    world_t * world )
```

La fonction initialise les données du monde du jeu.

Parameters

<i>world</i>	les données du monde
--------------	----------------------

4.1.2.10 init_sprite()

```
void init_sprite (
    sprite_t * sprite,
    int x,
    int y,
    int w,
    int h )
```

La fonction initialise la position du sprite.

Parameters

<i>sprite</i>	vaisseau
<i>x</i>	abscisse de la position du sprite
<i>y</i>	ordonnée de la position du sprite
<i>w</i>	largeur de la position du sprite
<i>h</i>	hauteur de la position du sprite

4.1.2.11 init_textures()

```
void init_textures (
    SDL_Renderer * renderer,
    textures_t * textures )
```

La fonction initialise les textures nécessaires à l'affichage graphique du jeu.

Parameters

<i>screen</i>	la surface correspondant à l'écran de jeu
<i>textures</i>	les textures du jeu

4.1.2.12 is_game_over()

```
int is_game_over (
    world_t * world )
```

La fonction indique si le jeu est fini en fonction des données du monde.

Parameters

<i>world</i>	les données du monde
--------------	----------------------

Returns

1 si le jeu est fini, 0 sinon

4.1.2.13 print_sprite()

```
void print_sprite (
    sprite_t sprite )
```

La fonction affiche la position du sprite.

Parameters

<i>sprite</i>	vaisseau
---------------	----------

4.1.2.14 refresh_graphics()

```
void refresh_graphics (
    SDL_Renderer * renderer,
    world_t * world,
    textures_t * textures )
```

La fonction rafraichit l'écran en fonction de l'état des données du monde.

Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>textures</i>	les textures

4.1.2.15 update_data()

```
void update_data (
    world_t * world )
```

La fonction met à jour les données en tenant compte de la physique du monde.

Parameters

<i>les</i>	données du monde
------------	------------------

4.2 sdl2-light.c File Reference

sur-couche de SDL2 pour simplifier son utilisation pour le projet

```
#include "sdl2-light.h"
#include <stdio.h>
#include <stdlib.h>
```

Functions

- int [init_sdl](#) (SDL_Window **window, SDL_Renderer **renderer, int width, int height)
La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.
- SDL_Texture * [load_image](#) (const char path[], SDL_Renderer *renderer)
La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.
- void [apply_texture](#) (SDL_Texture *texture, SDL_Renderer *renderer, int x, int y)
La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.
- void [clean_texture](#) (SDL_Texture *texture)
La fonction nettoie une texture en mémoire.
- void [clear_renderer](#) (SDL_Renderer *renderer)
La fonction vide le contenu graphique du renderer lié à l'écran de jeu.
- void [update_screen](#) (SDL_Renderer *renderer)
La fonction met à jour l'écran avec le contenu du renderer.
- void [pause](#) (int time)
La fonction met le programme en pause pendant un laps de temps.
- void [clean_sdl](#) (SDL_Renderer *renderer, SDL_Window *window)
La fonction nettoie le renderer et la fenêtre du jeu en mémoire.

4.2.1 Detailed Description

sur-couche de SDL2 pour simplifier son utilisation pour le projet

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

4.2.2 Function Documentation

4.2.2.1 apply_texture()

```
void apply_texture (
    SDL_Texture * texture,
    SDL_Renderer * renderer,
    int x,
    int y )
```

La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.

Parameters

<i>texture</i>	la texture que l'on va appliquer
<i>renderer</i>	le renderer qui va recevoir la texture
<i>x</i>	l'abscisse sur le renderer de l'endroit où est appliquée texture (point en haut à gauche de la surface)
<i>y</i>	l'ordonnée sur le renderer de l'endroit où est appliquée texture (point en haut à gauche de la surface)

4.2.2.2 clean_sdl()

```
void clean_sdl (
    SDL_Renderer * renderer,
    SDL_Window * window )
```

La fonction nettoie le renderer et la fenêtre du jeu en mémoire.

Parameters

<i>renderer</i>	le renderer à nettoyer
<i>window</i>	la fenêtre à nettoyer

4.2.2.3 clean_texture()

```
void clean_texture (
    SDL_Texture * texture )
```

La fonction nettoie une texture en mémoire.

Parameters

<i>texture</i>	la texture à nettoyer
----------------	-----------------------

4.2.2.4 clear_renderer()

```
void clear_renderer (
    SDL_Renderer * renderer )
```

La fonction vide le contenu graphique du renderer lié à l'écran de jeu.

Parameters

<i>renderer</i>	le renderer de l'écran
-----------------	------------------------

4.2.2.5 init_sdl()

```
int init_sdl (
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    int width,
    int height )
```

La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.

Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>width</i>	largeur de l'écran de jeu
<i>height</i>	hauteur de l'écran de jeu

Returns

-1 en cas d'erreur, 0 sinon

4.2.2.6 load_image()

```
SDL_Texture* load_image (
    const char path[],
    SDL_Renderer * renderer )
```

La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.

Parameters

<i>path</i>	est le chemin du fichier image. Le fichier doit être obligatoirement du BMP.
<i>renderer</i>	le renderer

Returns

la surface SDL contenant l'image avec la couleur RGB (255,0,255) rendue transparente. Elle renvoie NULL si le chargement a échoué (ex. le fichier path n'existe pas)

4.2.2.7 pause()

```
void pause (
    int time )
```

La fonction met le programme en pause pendant un laps de temps.

Parameters

<i>time</i>	ce laps de temps en milliseconde
-------------	----------------------------------

4.2.2.8 update_screen()

```
void update_screen (
    SDL_Renderer * renderer )
```

La fonction met à jour l'écran avec le contenu du renderer.

Parameters

<i>renderer</i>	le renderer de l'écran
-----------------	------------------------

4.3 sdl2-light.h File Reference

en-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet

```
#include "include/SDL.h"
```

Functions

- void [clean_sdl](#) (SDL_Renderer *renderer, SDL_Window *window)
La fonction nettoie le renderer et la fenêtre du jeu en mémoire.
- SDL_Texture * [load_image](#) (const char path[], SDL_Renderer *renderer)
La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.
- int [init_sdl](#) (SDL_Window **window, SDL_Renderer **renderer, int width, int height)
La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.
- void [clean_texture](#) (SDL_Texture *texture)
La fonction nettoie une texture en mémoire.
- void [apply_texture](#) (SDL_Texture *texture, SDL_Renderer *renderer, int x, int y)
La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.
- void [clear_renderer](#) (SDL_Renderer *renderer)
La fonction vide le contenu graphique du renderer lié à l'écran de jeu.
- void [update_screen](#) (SDL_Renderer *renderer)
La fonction met à jour l'écran avec le contenu du renderer.
- void [pause](#) (int time)
La fonction met le programme en pause pendant un laps de temps.

4.3.1 Detailed Description

en-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

4.3.2 Function Documentation

4.3.2.1 [apply_texture\(\)](#)

```
void apply_texture (
    SDL_Texture * texture,
    SDL_Renderer * renderer,
    int x,
    int y )
```

La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.

Parameters

<i>texture</i>	la texture que l'on va appliquer
<i>render</i>	le renderer qui va recevoir la texture
<i>x</i>	l'abscisse sur le renderer de l'endroit où est appliquée texture (point en haut à gauche de la surface)
<i>y</i>	l'ordonnée sur le renderer de l'endroit où est appliquée texture (point en haut à gauche de la surface)

4.3.2.2 clean_sdl()

```
void clean_sdl (
    SDL_Renderer * renderer,
    SDL_Window * window )
```

La fonction nettoie le renderer et la fenêtre du jeu en mémoire.

Parameters

<i>renderer</i>	le renderer à nettoyer
<i>window</i>	la fenêtre à nettoyer

4.3.2.3 clean_texture()

```
void clean_texture (
    SDL_Texture * texture )
```

La fonction nettoie une texture en mémoire.

Parameters

<i>texture</i>	la texture à nettoyer
----------------	-----------------------

4.3.2.4 clear_renderer()

```
void clear_renderer (
    SDL_Renderer * renderer )
```

La fonction vide le contenu graphique du renderer lié à l'écran de jeu.

Parameters

<i>renderer</i>	le renderer de l'écran
-----------------	------------------------

4.3.2.5 init_sdl()

```
int init_sdl (
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    int width,
    int height )
```

La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.

Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>width</i>	largeur de l'écran de jeu
<i>height</i>	hauteur de l'écran de jeu

Returns

-1 en cas d'erreur, 0 sinon

4.3.2.6 load_image()

```
SDL_Texture* load_image (
    const char path[],
    SDL_Renderer * renderer )
```

La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.

Parameters

<i>path</i>	est le chemin du fichier image. Le fichier doit être obligatoirement du BMP.
<i>renderer</i>	le renderer

Returns

la surface SDL contenant l'image avec la couleur RGB (255,0,255) rendue transparente. Elle renvoie NULL si le chargement a échoué (ex. le fichier path n'existe pas)

4.3.2.7 pause()

```
void pause (
    int time )
```

La fonction met le programme en pause pendant un laps de temps.

Parameters

<i>time</i>	ce laps de temps en milliseconde
-------------	----------------------------------

4.3.2.8 update_screen()

```
void update_screen (
    SDL_Renderer * renderer )
```

La fonction met à jour l'écran avec le contenu du renderer.

Parameters

<i>renderer</i>	le renderer de l'écran
-----------------	------------------------

Index

- apply_background
 - main.c, [11](#)
- apply_meteorite
 - main.c, [11](#)
- apply_sprite
 - main.c, [12](#)
- apply_texture
 - sdl2-light.c, [17](#)
 - sdl2-light.h, [20](#)
- background
 - textures_s, [6](#)
- clean
 - main.c, [12](#)
- clean_data
 - main.c, [12](#)
- clean_sdl
 - sdl2-light.c, [17](#)
 - sdl2-light.h, [21](#)
- clean_texture
 - sdl2-light.c, [17](#)
 - sdl2-light.h, [21](#)
- clean_textures
 - main.c, [13](#)
- clear_renderer
 - sdl2-light.c, [18](#)
 - sdl2-light.h, [21](#)
- gameover
 - world_s, [7](#)
- h
 - sprite_s, [5](#)
- handle_events
 - main.c, [13](#)
- init
 - main.c, [13](#)
- init_data
 - main.c, [14](#)
- init_sdl
 - sdl2-light.c, [18](#)
 - sdl2-light.h, [22](#)
- init_sprite
 - main.c, [14](#)
- init_textures
 - main.c, [14](#)
- is_game_over
 - main.c, [15](#)
- ligne_arrive
 - textures_s, [6](#)
 - world_s, [8](#)
- load_image
 - sdl2-light.c, [18](#)
 - sdl2-light.h, [22](#)
- main.c, [9](#)
 - apply_background, [11](#)
 - apply_meteorite, [11](#)
 - apply_sprite, [12](#)
 - clean, [12](#)
 - clean_data, [12](#)
 - clean_textures, [13](#)
 - handle_events, [13](#)
 - init, [13](#)
 - init_data, [14](#)
 - init_sprite, [14](#)
 - init_textures, [14](#)
 - is_game_over, [15](#)
 - print_sprite, [15](#)
 - refresh_graphics, [15](#)
 - update_data, [15](#)
- meteorite
 - textures_s, [7](#)
- mur
 - world_s, [8](#)
- pause
 - sdl2-light.c, [19](#)
 - sdl2-light.h, [22](#)
- print_sprite
 - main.c, [15](#)
- refresh_graphics
 - main.c, [15](#)
- sdl2-light.c, [16](#)
 - apply_texture, [17](#)
 - clean_sdl, [17](#)
 - clean_texture, [17](#)
 - clear_renderer, [18](#)
 - init_sdl, [18](#)
 - load_image, [18](#)
 - pause, [19](#)
 - update_screen, [19](#)
- sdl2-light.h, [20](#)
 - apply_texture, [20](#)
 - clean_sdl, [21](#)
 - clean_texture, [21](#)

- clear_renderer, [21](#)
 - init_sdl, [22](#)
 - load_image, [22](#)
 - pause, [22](#)
 - update_screen, [23](#)
- sprite_s, [5](#)
 - h, [5](#)
 - w, [5](#)
 - x, [5](#)
 - y, [6](#)
- textures_s, [6](#)
 - background, [6](#)
 - ligne_arrive, [6](#)
 - meteorite, [7](#)
 - vaisseau, [7](#)
- update_data
 - main.c, [15](#)
- update_screen
 - sdl2-light.c, [19](#)
 - sdl2-light.h, [23](#)
- vaisseau
 - textures_s, [7](#)
 - world_s, [8](#)
- w
 - sprite_s, [5](#)
- world_s, [7](#)
 - gameover, [7](#)
 - ligne_arrive, [8](#)
 - mur, [8](#)
 - vaisseau, [8](#)
- x
 - sprite_s, [5](#)
- y
 - sprite_s, [6](#)