

## My Project

Generated by Doxygen 1.8.18



<b>1 Règles du jeu</b>	<b>1</b>
1.1 Comment jouer ?	1
1.2 Le but du jeu	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 sprite_s Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 h	7
4.1.2.2 w	7
4.1.2.3 x	8
4.1.2.4 y	8
4.2 textures_s Struct Reference	8
4.2.1 Detailed Description	8
4.2.2 Member Data Documentation	8
4.2.2.1 arrival	8
4.2.2.2 background	9
4.2.2.3 font	9
4.2.2.4 meteorite	9
4.2.2.5 vaisseau	9
4.3 world_s Struct Reference	9
4.3.1 Detailed Description	10
4.3.2 Member Data Documentation	10
4.3.2.1 arrival	10
4.3.2.2 gameover	10
4.3.2.3 make_disappear	10
4.3.2.4 mur	10
4.3.2.5 vaisseau	10
4.3.2.6 vy	10
<b>5 File Documentation</b>	<b>11</b>
5.1 game_event.c File Reference	11
5.2 game_event.h File Reference	11
5.2.1 Detailed Description	12
5.2.2 Function Documentation	12
5.2.2.1 handle_sprites_collision()	12
5.2.2.2 init_data()	12
5.2.2.3 init_sprite()	13

5.2.2.4 init_sprite_meteore()	13
5.2.2.5 init_walls()	14
5.2.2.6 is_game_over()	14
5.2.2.7 out_of_screen()	14
5.2.2.8 sprites_collide()	14
5.2.2.9 update_data()	16
5.2.2.10 update_walls()	16
5.3 graphic.c File Reference	16
5.3.1 Detailed Description	17
5.3.2 Function Documentation	17
5.3.2.1 apply_background()	17
5.3.2.2 apply_sprite()	18
5.3.2.3 apply_wall()	18
5.3.2.4 apply_walls()	19
5.3.2.5 clean()	19
5.3.2.6 clean_data()	19
5.3.2.7 clean_textures()	20
5.3.2.8 init()	20
5.3.2.9 init_textures()	20
5.3.2.10 refresh_graphics()	21
5.4 handle_event.c File Reference	21
5.4.1 Detailed Description	21
5.4.2 Function Documentation	22
5.4.2.1 handle_events()	22
5.5 handle_event.h File Reference	22
5.5.1 Detailed Description	22
5.5.2 Function Documentation	23
5.5.2.1 handle_events()	23
5.6 main.c File Reference	23
5.6.1 Detailed Description	23
5.6.2 Function Documentation	24
5.6.2.1 main()	24
5.7 param.h File Reference	24
5.7.1 Detailed Description	25
5.8 sdl2-light.c File Reference	26
5.8.1 Detailed Description	26
5.8.2 Function Documentation	26
5.8.2.1 apply_texture()	26
5.8.2.2 clean_sdl()	27
5.8.2.3 clean_texture()	27
5.8.2.4 clear_renderer()	27
5.8.2.5 init_sdl()	28

5.8.2.6 load_image()	28
5.8.2.7 pause()	28
5.8.2.8 update_screen()	29
5.9 sdl2-light.h File Reference	29
5.9.1 Detailed Description	30
5.9.2 Function Documentation	30
5.9.2.1 apply_texture()	30
5.9.2.2 clean_sdl()	30
5.9.2.3 clean_texture()	31
5.9.2.4 clear_renderer()	31
5.9.2.5 init_sdl()	31
5.9.2.6 load_image()	32
5.9.2.7 pause()	32
5.9.2.8 update_screen()	32
5.10 tests.c File Reference	33
5.10.1 Detailed Description	33
5.10.2 Function Documentation	34
5.10.2.1 main()	34
5.10.2.2 print_sprite()	34
5.10.2.3 test_handle_sprites_collision_param()	34
5.10.2.4 test_init_sprite_param()	35
5.10.2.5 test_out_of_screen_param()	35
5.10.2.6 test_sprites_collide_param()	35
<b>Index</b>	<b>39</b>



# Chapter 1

## Règles du jeu

### 1.1 Comment jouer ?

- Déplacements
  - Touche **Z** : Ralentir le jeu
  - Touche **S** : Accélérer le jeu
  - Touche **Q** : Se déplacer à gauche
  - Touche **D** : Se déplacer à droite
- Sortir du jeu
  - Touche **Echap**

### 1.2 Le but du jeu

L'objectif principal de ce jeu est d'atteindre la ligne d'arrivée tout en évitant les météorites qui vous barrent le passage.





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">sprite_s</a>	Représentation d'une texture du jeu . . . . .	7
<a href="#">textures_s</a>	Représentation pour stocker les textures nécessaires à l'affichage graphique . . . . .	8
<a href="#">world_s</a>	Représentation du monde du jeu . . . . .	9



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">game_event.c</a>	Module de gestions des événements . . . . .	11
<a href="#">game_event.h</a>	Entête du module de gestions des événements . . . . .	11
<a href="#">graphic.c</a>	Module gérant l'affichage SDL2 . . . . .	16
<b>graphic.h</b>	. . . . .	??
<a href="#">handle_event.c</a>	Fichier .c du module handle_event . . . . .	21
<a href="#">handle_event.h</a>	Fichier .h du module handle_event . . . . .	22
<a href="#">main.c</a>	Programme principal initial du niveau 1 . . . . .	23
<b>main.h</b>	. . . . .	??
<a href="#">param.h</a>	Fichier qui contient les different constante librerie et structure . . . . .	24
<a href="#">sdl2-light.c</a>	Sur-couche de SDL2 pour simplifier son utilisation pour le projet . . . . .	26
<a href="#">sdl2-light.h</a>	En-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet . . . . .	29
<b>sdl2-ttf-light.h</b>	. . . . .	??
<a href="#">tests.c</a>	Fichier de test pour les fonction de game_event . . . . .	33



## Chapter 4

# Class Documentation

### 4.1 sprite\_s Struct Reference

Représentation d'une texture du jeu.

```
#include <param.h>
```

#### Public Attributes

- int [x](#)
- int [y](#)
- int [w](#)
- int [h](#)

#### 4.1.1 Detailed Description

Représentation d'une texture du jeu.

#### 4.1.2 Member Data Documentation

##### 4.1.2.1 h

```
int sprite_s::h
```

Hauteur du sprite.

##### 4.1.2.2 w

```
int sprite_s::w
```

Largeur du sprite.

#### 4.1.2.3 x

```
int sprite_s::x
```

Position du sprite sur x.

#### 4.1.2.4 y

```
int sprite_s::y
```

Position du sprite sur y.

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

- [param.h](#)

## 4.2 textures\_s Struct Reference

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

```
#include <param.h>
```

### Public Attributes

- SDL\_Texture \* [background](#)
- SDL\_Texture \* [vaisseau](#)
- SDL\_Texture \* [arrival](#)
- SDL\_Texture \* [meteorite](#)
- TTF\_Font \* [font](#)

### 4.2.1 Detailed Description

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

### 4.2.2 Member Data Documentation

#### 4.2.2.1 arrival

```
SDL_Texture* textures_s::arrival
```

Texture liée à l'image de la ligne d'arrivée.

#### 4.2.2.2 background

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

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

#### 4.2.2.3 font

```
TTF_Font* textures_s::font
```

Texture liée à la police utilisée.

#### 4.2.2.4 meteorite

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

Texture liée à l'image d'un météorite.

#### 4.2.2.5 vaisseau

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

Texture liée à l'image du vaisseau.

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

- [param.h](#)

## 4.3 world\_s Struct Reference

Représentation du monde du jeu.

```
#include <param.h>
```

Collaboration diagram for world\_s:

### Public Attributes

- int [gameover](#)
- [sprite\\_t](#) [vaisseau](#)
- [sprite\\_t](#) [arrival](#)
- int [vy](#)
- [sprite\\_t](#) [mur](#) [[MAX\\_METEORITE\\_WALL\\_NUMBER](#)]
- int [make\\_disappear](#)

### 4.3.1 Detailed Description

Représentation du monde du jeu.

### 4.3.2 Member Data Documentation

#### 4.3.2.1 arrival

```
sprite_t world_s::arrival
```

Information de la ligne d'arrivée.

#### 4.3.2.2 gameover

```
int world_s::gameover
```

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

#### 4.3.2.3 make\_disappear

```
int world_s::make_disappear
```

Informe si le vaisseau doit être visible ou pas

#### 4.3.2.4 mur

```
sprite_t world_s::mur[MAX_METEORITE_WALL_NUMBER]
```

Informations sur un mur d'astéroïdes.

#### 4.3.2.5 vaisseau

```
sprite_t world_s::vaisseau
```

Information du vaisseau.

#### 4.3.2.6 vy

```
int world_s::vy
```

Vitesse de déplacement verticale.

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

- [param.h](#)



## Chapter 5

# File Documentation

### 5.1 game\_event.c File Reference

Module de gestions des événements.

```
#include "game_event.h"
```

Include dependency graph for game\_event.c:

### 5.2 game\_event.h File Reference

Entête du module de gestions des événements.

```
#include "param.h"
```

```
#include "sdl2-light.h"
```

Include dependency graph for game\_event.h: This graph shows which files directly or indirectly include this file:

### Functions

- 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 [init\\_sprite](#) ([sprite\\_t](#) \*sprite, int x, int y, int w, int h)  
*La fonction initialise les données d'un sprite selon les valeurs entrées.*
- void [init\\_sprite\\_meteore](#) ([sprite\\_t](#) \*sprite, int x, int y, int w, int h, int screen\_nbr)  
*La fonction initialise les données d'un sprite de meteore selon les valeurs entrées.*
- void [init\\_data](#) ([world\\_t](#) \*world)  
*La fonction initialise les données du monde du jeu.*
- void [out\\_of\\_screen](#) ([world\\_t](#) \*world)  
*Fonction qui detecte si le vaisseau est hors de l'écran.*
- int [sprites\\_collide](#) ([sprite\\_t](#) \*sp1, [sprite\\_t](#) \*sp2)  
*Fonction qui detecte si deux sprite sont en collision.*
- void [handle\\_sprites\\_collision](#) ([world\\_t](#) \*world, [sprite\\_t](#) \*sp1, [sprite\\_t](#) \*sp2, int \*make\_disappear)  
*Fonction qui change la vitesse du monde en cas de collision.*
- void [init\\_walls](#) ([world\\_t](#) \*world)  
*Fonction qui change initialise les murs.*
- void [update\\_walls](#) ([world\\_t](#) \*world)  
*Fonction qui update la position de tout les murs.*

## 5.2.1 Detailed Description

Entête du module de gestions des événements.

### Author

Victor Dallé

### Version

1.0

### Date

1 avril 2021

## 5.2.2 Function Documentation

### 5.2.2.1 handle\_sprites\_collision()

```
void handle_sprites_collision (
    world_t * world,
    sprite_t * sp1,
    sprite_t * sp2,
    int * make_disappear )
```

Fonction qui change la vitesse du monde en cas de collision.

#### Parameters

<i>world</i>	Structure des données du monde.
<i>sp1</i>	premier sprite
<i>sp2</i>	second prite
<i>make_disappear</i>	détermine la visibilité du premie sprite de la fonction

### 5.2.2.2 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.
--------------	-----------------------

## 5.2.2.3 init\_sprite()

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

La fonction initialise les données d'un sprite selon les valeurs entrées.

## Parameters

<i>sprite</i>	Pointeur vers sprite_t pour l'initialisation des données.
<i>x</i>	Coordonnée x du sprite.
<i>y</i>	Coordonnée y du sprite.
<i>w</i>	Largeur du sprite.
<i>h</i>	Hauteur du sprite.

## 5.2.2.4 init\_sprite\_meteore()

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

La fonction initialise les données d'un sprite de meteore selon les valeurs entrées.

## Parameters

<i>sprite</i>	Pointeur vers sprite_t pour l'initialisation des données.
<i>x</i>	Coordonnée x du sprite.
<i>y</i>	Coordonnée y du sprite.
<i>w</i>	Largeur du sprite.
<i>h</i>	Hauteur du sprite.
<i>screen_nbr</i>	L'ecran ou le meteore doit être placer

#### 5.2.2.5 init\_walls()

```
void init_walls (
    world_t * world )
```

Fonction qui initialise les murs.

##### Parameters

<i>world</i>	Structure des données du monde.
--------------	---------------------------------

#### 5.2.2.6 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.

#### 5.2.2.7 out\_of\_screen()

```
void out_of_screen (
    world_t * world )
```

Fonction qui detecte si le vaisseau est hors de l'écran.

##### Parameters

<i>world</i>	Structure des données du monde.
--------------	---------------------------------

#### 5.2.2.8 sprites\_collide()

```
int sprites_collide (
    sprite_t * sp1,
    sprite_t * sp2 )
```

Fonction qui detecte si deux sprite sont en collision.

**Parameters**

<i>sp1</i>	premier sprite
<i>sp2</i>	second prite

**Returns**

1 si collision est 0 si aucun collision.

**5.2.2.9 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>world</i>	Les données du monde.
--------------	-----------------------

**5.2.2.10 update\_walls()**

```
void update_walls (
    world_t * world )
```

Fonction qui update la position de tout les murs.

**Parameters**

<i>world</i>	Structure des données du monde.
--------------	---------------------------------

## 5.3 graphic.c File Reference

Module gérant l'affichage SDL2.

```
#include "graphic.h"
Include dependency graph for graphic.c:
```

**Functions**

- 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\\_sprite](#) (SDL\_Renderer \*renderer, SDL\_Texture \*texture, [sprite\\_t](#) \*sprite, int make\_disappear)
- La fonction applique un sprite au renderer.*
- 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\\_wall](#) ([textures\\_t](#) \*textures, SDL\_Renderer \*renderer, [world\\_t](#) \*world, int x, int y, int height, int width)
- La fonction applique la texture des meteorite sur le renderer.*
- 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\\_data](#) ([world\\_t](#) \*world)
- La fonction nettoie les données du monde.*
- void [clean](#) (SDL\_Window \*window, SDL\_Renderer \*renderer, [textures\\_t](#) \*textures, [world\\_t](#) \*world)
- La fonction 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)
- La fonction initialise le jeu: initialisation de la partie graphique (SDL), chargement des textures, initialisation des données.*
- void [apply\\_walls](#) (SDL\_Renderer \*\*renderer, [textures\\_t](#) \*textures, [world\\_t](#) \*world)
- La fonction applique la texture des murs.*

### 5.3.1 Detailed Description

Module gérant l'affichage SDL2.

Entête du module gérant l'affichage SDL2.

Author

Victor Dallé

Version

1.0

Date

1 avril 2021

### 5.3.2 Function Documentation

#### 5.3.2.1 [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.

### 5.3.2.2 apply\_sprite()

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

La fonction applique un sprite au renderer.

## Parameters

<i>renderer</i>	Renderer vers lequel on envoie les textures et les sprites.
<i>texture</i>	Texture envoyée vers le renderer.
<i>sprite</i>	Sprite envoyé vers le renderer.
<i>make_disappear</i>	Déside si le sprite et afficher ou pas.

### 5.3.2.3 apply\_wall()

```
void apply_wall (
    textures_t * textures,
    SDL_Renderer * renderer,
    world_t * world,
    int x,
    int y,
    int height,
    int width )
```

La fonction applique la texture des meteorite sur le renderer.

## Parameters

<i>textures</i>	Les textures.
<i>renderer</i>	Le renderer lié à l'écran de jeu.
<i>world</i>	Les données du monde.
<i>x</i>	La position du mur sur l'axe des abscisses.
<i>y</i>	La position du mur sur l'axe des ordonnées.
<i>height</i>	La longueur du mur.
<i>width</i>	La largeur du mur.



#### 5.3.2.4 apply\_walls()

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

La fonction applique la texture des murs.

##### Parameters

<i>renderer</i>	Le renderer.
<i>textures</i>	Les textures.
<i>world</i>	Le monde.

#### 5.3.2.5 clean()

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

La fonction 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.

#### 5.3.2.6 clean\_data()

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

La fonction nettoie les données du monde.

##### Parameters

<i>world</i>	Les données du monde.
--------------	-----------------------

### 5.3.2.7 clean\_textures()

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

La fonction nettoie les textures.

#### Parameters

<i>textures</i>	Les textures.
-----------------	---------------

### 5.3.2.8 init()

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

La fonction 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.

### 5.3.2.9 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.

### 5.3.2.10 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.

## 5.4 handle\_event.c File Reference

Fichier .c du module handle\_event.

```
#include "handle_event.h"
```

Include dependency graph for handle\_event.c:

### Functions

- 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.*

### 5.4.1 Detailed Description

Fichier .c du module handle\_event.

#### Author

Victor Dallé

#### Version

0.1

#### Date

2021-04-12

#### Copyright

Copyright (c) 2021

## 5.4.2 Function Documentation

### 5.4.2.1 `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.

## 5.5 `handle_event.h` File Reference

Fichier .h du module `handle_event`.

```
#include <SDL2/SDL.h>
#include "param.h"
#include "game_event.h"
```

Include dependency graph for `handle_event.h`: This graph shows which files directly or indirectly include this file:

### Functions

- 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.*

### 5.5.1 Detailed Description

Fichier .h du module `handle_event`.

#### Author

Victor Dallé

#### Version

0.1

#### Date

2021-04-12

#### Copyright

Copyright (c) 2021

## 5.5.2 Function Documentation

### 5.5.2.1 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.

## 5.6 main.c File Reference

Programme principal initial du niveau 1.

```
#include "main.h"
```

Include dependency graph for main.c:

### Functions

- int `main` (int argc, char \*argv[])  
*Programme principal qui implémente la boucle du jeu.*

### 5.6.1 Detailed Description

Programme principal initial du niveau 1.

Entête du main.

#### Author

Mathieu Constant

#### Version

1.0

#### Date

18 mars 2021

**Author**

Victor Dallé

**Version**

1.0

**Date**

1 avril 2021

## 5.6.2 Function Documentation

### 5.6.2.1 main()

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

Programme principal qui implémente la boucle du jeu.

**Parameters**

<i>argc</i>	Taille du tableau argv.
<i>argv</i>	Pointeur vers un tableau de char de taille argc.

**Returns**

0, si il n'y a pas eu d'erreurs.

## 5.7 param.h File Reference

Fichier qui contient les different constante librerie et structure.

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
```

Include dependency graph for param.h: This graph shows which files directly or indirectly include this file:

**Classes**

- struct [textures\\_s](#)  
*Représentation pour stocker les textures nécessaires à l'affichage graphique.*
- struct [sprite\\_s](#)  
*Représentation d'une texture du jeu.*
- 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.*
- #define [MAX\\_METEORITE\\_WALL\\_NUMBER](#) 20  
*Nombre de mur de météorite.*
- #define [METEORITE\\_WALL\\_NUMBER](#) 16  
*Nombre de mur de météorite.*

## Typedefs

- typedef struct [textures\\_s](#) [textures\\_t](#)  
*Type qui correspond aux textures du jeu.*
- typedef struct [sprite\\_s](#) [sprite\\_t](#)  
*Type qui correspond à une texture.*
- typedef struct [world\\_s](#) [world\\_t](#)  
*Type qui correspond aux données du monde.*

### 5.7.1 Detailed Description

Fichier qui contient les different constante librerie et structure.

#### Author

Periney Yann

#### Version

1

#### Date

1 avril 2021

## 5.8 sdl2-light.c File Reference

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

```
#include "sdl2-light.h"
```

Include dependency graph for sdl2-light.c:

### 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.*

### 5.8.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

### 5.8.2 Function Documentation

#### 5.8.2.1 [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)

### 5.8.2.2 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

### 5.8.2.3 clean\_texture()

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

La fonction nettoie une texture en mémoire.

## Parameters

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

### 5.8.2.4 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
-----------------	------------------------

### 5.8.2.5 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

### 5.8.2.6 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)

### 5.8.2.7 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
-------------	----------------------------------

#### 5.8.2.8 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
-----------------	------------------------

## 5.9 sdl2-light.h File Reference

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

```
#include "param.h"
#include <SDL2/SDL.h>
```

Include dependency graph for sdl2-light.h: This graph shows which files directly or indirectly include this file:

### 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.*

### 5.9.1 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

### 5.9.2 Function Documentation

#### 5.9.2.1 `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)

#### 5.9.2.2 `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

### 5.9.2.3 clean\_texture()

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

La fonction nettoie une texture en mémoire.

## Parameters

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

### 5.9.2.4 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
-----------------	------------------------

### 5.9.2.5 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

**5.9.2.6 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)

**5.9.2.7 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
-------------	----------------------------------

**5.9.2.8 update\_screen()**

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

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

## Parameters

<code>render</code>	le renderer de l'écran
---------------------	------------------------

## 5.10 tests.c File Reference

Fichier de test pour les fonction de game\_event.

```
#include "param.h"
#include "game_event.h"
Include dependency graph for tests.c:
```

### Functions

- void `print_sprite` (`sprite_t` \*sprite)  
*Fonction qui affiche les coordonnées d'un sprite.*
- void `test_init_sprite_param` (`sprite_t` sprite, int x, int y, int w, int h)  
*Fonction de test\_param pour init\_sprite.*
- void `test_init_sprite` ()
- void `test_out_of_screen_param` (`world_t` \*world)  
*Fonction de test pour la sortie de l'écran du vaisseau.*
- void `test_out_of_screen` ()
- void `test_sprites_collide_param` (`sprite_t` sprite1, `sprite_t` sprite2)  
*Fonction de test pour la collision de 2 sprites.*
- void `test_sprites_collide` ()
- void `test_handle_sprites_collision_param` (`world_t` world, `sprite_t` spr1, `sprite_t` spr2, int disp)  
*Fonction de test effectuant des modifs sur data world lors d'une collision.*
- void `test_handle_sprites_collision` ()
- void `test_init_walls_param` (`world_t` \*world)
- void `test_init_walls` ()
- void `test_update_walls_param` (`world_t` \*world)
- void `test_update_walls` ()
- int `main` (int argc, char \*argv[])  
*Main pour le programme de test pour le module `game_event.h`.*

### 5.10.1 Detailed Description

Fichier de test pour les fonction de game\_event.

## Author

Periney Yann Victor Dallé

## Version

1

## Date

1 avril 2021

## 5.10.2 Function Documentation

### 5.10.2.1 main()

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

Main pour le programme de test pour le module [game\\_event.h](#).

#### Parameters

<i>argc</i>	Taille du tableau argv.
<i>argv</i>	Pointeur vers un tableau de char de taille argc.

#### Returns

0, si il n'y a pas eu d'erreurs.

### 5.10.2.2 print\_sprite()

```
void print_sprite (
    sprite\_t * sprite )
```

Fonction qui affiche les coordonnées d'un sprite.

#### Parameters

<i>sprite</i>	Sprite cible du test.
---------------	-----------------------

### 5.10.2.3 test\_handle\_sprites\_collision\_param()

```
void test_handle_sprites_collision_param (
    world\_t world,
    sprite\_t spr1,
    sprite\_t spr2,
    int disp )
```

Fonction de test effectuant des modifs sur data world lors d'une collision.



## Parameters

<i>world</i>	Données du monde.
<i>spr1</i>	Premier sprite.
<i>spr2</i>	Deuxième sprite.
<i>disp</i>	disparition (bool) du vaisseau.

## 5.10.2.4 test\_init\_sprite\_param()

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

Fonction de test\_param pour init\_sprite.

## Parameters

<i>sprite</i>	Le sprite qui sera initialisé et testé.
<i>x</i>	Position de l'abscisse.
<i>y</i>	Position de l'ordonnée.
<i>w</i>	Largeur du sprite.
<i>h</i>	Hauteur du sprite.

## 5.10.2.5 test\_out\_of\_screen\_param()

```
void test_out_of_screen_param (
    world_t * world )
```

Fonction de test pour la sortie de l'écran du vaisseau.

## Parameters

<i>world</i>	Données du monde.
--------------	-------------------

## 5.10.2.6 test\_sprites\_collide\_param()

```
void test_sprites_collide_param (
    sprite_t sprite1,
    sprite_t sprite2 )
```

Fonction de test pour la collision de 2 sprites.

## Parameters

<i>sprite1</i>	Premier sprite.
<i>sprite2</i>	Deuxième sprite.



# Index

- apply\_background
  - graphic.c, 17
- apply\_sprite
  - graphic.c, 18
- apply\_texture
  - sdl2-light.c, 26
  - sdl2-light.h, 30
- apply\_wall
  - graphic.c, 18
- apply\_walls
  - graphic.c, 19
- arrival
  - textures\_s, 8
  - world\_s, 10
- background
  - textures\_s, 8
- clean
  - graphic.c, 19
- clean\_data
  - graphic.c, 19
- clean\_sdl
  - sdl2-light.c, 27
  - sdl2-light.h, 30
- clean\_texture
  - sdl2-light.c, 27
  - sdl2-light.h, 31
- clean\_textures
  - graphic.c, 20
- clear\_renderer
  - sdl2-light.c, 27
  - sdl2-light.h, 31
- font
  - textures\_s, 9
- game\_event.c, 11
- game\_event.h, 11
  - handle\_sprites\_collision, 12
  - init\_data, 12
  - init\_sprite, 13
  - init\_sprite\_meteor, 13
  - init\_walls, 13
  - is\_game\_over, 14
  - out\_of\_screen, 14
  - sprites\_collide, 14
  - update\_data, 16
  - update\_walls, 16
- gameover
  - world\_s, 10
- graphic.c, 16
  - apply\_background, 17
  - apply\_sprite, 18
  - apply\_wall, 18
  - apply\_walls, 19
  - clean, 19
  - clean\_data, 19
  - clean\_textures, 20
  - init, 20
  - init\_textures, 20
  - refresh\_graphics, 20
- h
  - sprite\_s, 7
- handle\_event.c, 21
  - handle\_events, 22
- handle\_event.h, 22
  - handle\_events, 23
- handle\_events
  - handle\_event.c, 22
  - handle\_event.h, 23
- handle\_sprites\_collision
  - game\_event.h, 12
- init
  - graphic.c, 20
- init\_data
  - game\_event.h, 12
- init\_sdl
  - sdl2-light.c, 28
  - sdl2-light.h, 31
- init\_sprite
  - game\_event.h, 13
- init\_sprite\_meteor
  - game\_event.h, 13
- init\_textures
  - graphic.c, 20
- init\_walls
  - game\_event.h, 13
- is\_game\_over
  - game\_event.h, 14
- load\_image
  - sdl2-light.c, 28
  - sdl2-light.h, 32
- main
  - main.c, 24
  - tests.c, 34

- main.c, [23](#)
  - main, [24](#)
- make\_disappear
  - world\_s, [10](#)
- meteorite
  - textures\_s, [9](#)
- mur
  - world\_s, [10](#)
- out\_of\_screen
  - game\_event.h, [14](#)
- param.h, [24](#)
- pause
  - sdl2-light.c, [28](#)
  - sdl2-light.h, [32](#)
- print\_sprite
  - tests.c, [34](#)
- refresh\_graphics
  - graphic.c, [20](#)
- sdl2-light.c, [26](#)
  - apply\_texture, [26](#)
  - clean\_sdl, [27](#)
  - clean\_texture, [27](#)
  - clear\_renderer, [27](#)
  - init\_sdl, [28](#)
  - load\_image, [28](#)
  - pause, [28](#)
  - update\_screen, [29](#)
- sdl2-light.h, [29](#)
  - apply\_texture, [30](#)
  - clean\_sdl, [30](#)
  - clean\_texture, [31](#)
  - clear\_renderer, [31](#)
  - init\_sdl, [31](#)
  - load\_image, [32](#)
  - pause, [32](#)
  - update\_screen, [32](#)
- sprite\_s, [7](#)
  - h, [7](#)
  - w, [7](#)
  - x, [7](#)
  - y, [8](#)
- sprites\_collide
  - game\_event.h, [14](#)
- test\_handle\_sprites\_collision\_param
  - tests.c, [34](#)
- test\_init\_sprite\_param
  - tests.c, [35](#)
- test\_out\_of\_screen\_param
  - tests.c, [35](#)
- test\_sprites\_collide\_param
  - tests.c, [35](#)
- tests.c, [33](#)
  - main, [34](#)
  - print\_sprite, [34](#)
  - test\_handle\_sprites\_collision\_param, [34](#)
  - test\_init\_sprite\_param, [35](#)
  - test\_out\_of\_screen\_param, [35](#)
  - test\_sprites\_collide\_param, [35](#)
- textures\_s, [8](#)
  - arrival, [8](#)
  - background, [8](#)
  - font, [9](#)
  - meteorite, [9](#)
  - vaisseau, [9](#)
- update\_data
  - game\_event.h, [16](#)
- update\_screen
  - sdl2-light.c, [29](#)
  - sdl2-light.h, [32](#)
- update\_walls
  - game\_event.h, [16](#)
- vaisseau
  - textures\_s, [9](#)
  - world\_s, [10](#)
- vy
  - world\_s, [10](#)
- w
  - sprite\_s, [7](#)
- world\_s, [9](#)
  - arrival, [10](#)
  - gameover, [10](#)
  - make\_disappear, [10](#)
  - mur, [10](#)
  - vaisseau, [10](#)
  - vy, [10](#)
- x
  - sprite\_s, [7](#)
- y
  - sprite\_s, [8](#)