

My Project

Generated by Doxygen 1.8.18

1 Règles du jeu	1
1.1 Le but du jeu	1
1.2 Comment jouer ?	1
1.3 Fin du jeu	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 menu_s Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 currentmenu	7
4.1.2.2 currentoption	8
4.1.2.3 menunumber	8
4.1.2.4 menuover	8
4.1.2.5 optionnumber	8
4.1.2.6 quitte	8
4.1.2.7 time	8
4.2 program_s Struct Reference	8
4.2.1 Detailed Description	9
4.2.2 Member Data Documentation	9
4.2.2.1 mode	9
4.2.2.2 programover	9
4.2.2.3 restart	9
4.3 sprite_s Struct Reference	9
4.3.1 Detailed Description	10
4.3.2 Member Data Documentation	10
4.3.2.1 h	10
4.3.2.2 w	10
4.3.2.3 x	10
4.3.2.4 y	10
4.4 textures_s Struct Reference	11
4.4.1 Detailed Description	11
4.4.2 Member Data Documentation	11
4.4.2.1 arrival	11
4.4.2.2 background	11
4.4.2.3 fin	11
4.4.2.4 finb	12
4.4.2.5 font	12

4.4.2.6 menu1_select	12
4.4.2.7 menu_background	12
4.4.2.8 menu_replay	12
4.4.2.9 meteorite	12
4.4.2.10 rules	12
4.4.2.11 vaisseau	13
4.5 world_s Struct Reference	13
4.5.1 Detailed Description	13
4.5.2 Member Data Documentation	13
4.5.2.1 arrival	13
4.5.2.2 gameover	14
4.5.2.3 levelstart	14
4.5.2.4 make_disappear	14
4.5.2.5 mur	14
4.5.2.6 vaisseau	14
4.5.2.7 vy	14
5 File Documentation	15
5.1 graphic.c File Reference	15
5.2 graphic.h File Reference	15
5.2.1 Detailed Description	16
5.2.2 Function Documentation	16
5.2.2.1 apply_background()	16
5.2.2.2 apply_sprite()	17
5.2.2.3 apply_wall()	17
5.2.2.4 apply_walls()	18
5.2.2.5 clean()	18
5.2.2.6 clean_data()	18
5.2.2.7 clean_textures()	19
5.2.2.8 init()	19
5.2.2.9 init_textures()	19
5.2.2.10 level_start()	20
5.2.2.11 print_end()	20
5.2.2.12 print_end_b()	20
5.2.2.13 refresh_graphics()	21
5.3 handle_event.c File Reference	21
5.3.1 Detailed Description	21
5.3.2 Function Documentation	22
5.3.2.1 handle_events()	22
5.3.2.2 handle_events_menu()	22
5.3.2.3 menu_selection()	22
5.4 handle_event.h File Reference	23

5.4.1 Detailed Description	23
5.4.2 Function Documentation	23
5.4.2.1 handle_events()	23
5.4.2.2 handle_events_menu()	24
5.4.2.3 menu_selection()	24
5.5 main.c File Reference	24
5.5.1 Detailed Description	25
5.5.2 Function Documentation	25
5.5.2.1 main()	25
5.6 main.h File Reference	26
5.6.1 Detailed Description	26
5.7 menu_data.c File Reference	26
5.7.1 Detailed Description	27
5.7.2 Function Documentation	27
5.7.2.1 is_menu_over()	27
5.7.2.2 is_menu_quitte()	27
5.8 menu_data.h File Reference	28
5.8.1 Detailed Description	28
5.8.2 Function Documentation	28
5.8.2.1 is_menu_over()	28
5.8.2.2 is_menu_quitte()	29
5.9 menu_graphic.h File Reference	29
5.9.1 Detailed Description	30
5.9.2 Function Documentation	30
5.9.2.1 apply_menu()	30
5.9.2.2 apply_select()	30
5.9.2.3 refresh_menu_graphics()	31
5.10 param.h File Reference	31
5.10.1 Detailed Description	32
5.11 sdl2-light.c File Reference	33
5.11.1 Detailed Description	33
5.11.2 Function Documentation	33
5.11.2.1 apply_texture()	33
5.11.2.2 clean_sdl()	34
5.11.2.3 clean_texture()	34
5.11.2.4 clear_renderer()	34
5.11.2.5 init_sdl()	35
5.11.2.6 load_image()	35
5.11.2.7 pause()	35
5.11.2.8 update_screen()	36
5.12 sdl2-light.h File Reference	36
5.12.1 Detailed Description	37

5.12.2 Function Documentation	37
5.12.2.1 apply_texture()	37
5.12.2.2 clean_sdl()	37
5.12.2.3 clean_texture()	38
5.12.2.4 clear_renderer()	38
5.12.2.5 init_sdl()	38
5.12.2.6 load_image()	39
5.12.2.7 pause()	39
5.12.2.8 update_screen()	39
5.13 sdl2-ttf-light.c File Reference	40
5.13.1 Detailed Description	40
5.13.2 Function Documentation	40
5.13.2.1 apply_text()	41
5.13.2.2 clean_font()	41
5.13.2.3 load_font()	41
5.14 sdl2-ttf-light.h File Reference	42
5.14.1 Detailed Description	42
5.14.2 Function Documentation	42
5.14.2.1 apply_text()	43
5.14.2.2 clean_font()	43
5.14.2.3 load_font()	43
5.15 tests.c File Reference	44
5.15.1 Detailed Description	44
5.15.2 Function Documentation	45
5.15.2.1 main()	45
5.15.2.2 print_sprite()	45
5.15.2.3 test_handle_sprites_collision_param()	45
5.15.2.4 test_init_sprite_param()	46
5.15.2.5 test_out_of_screen_param()	46
5.15.2.6 test_sprites_collide_param()	46
Index	49

Chapter 1

Règles du jeu

1.1 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. Pour les éviter, vous devez vous déplacer avec les touches ci-dessous.

1.2 Comment jouer ?

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

1.3 Fin du jeu

Si vous avez eu le talent de finir le jeu, vous pouvez *saisir votre nom* dans le **terminal** pour graver votre performance dans l'octet.

Chapter 2

Class Index

2.1 Class List

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

menu_s	Représentation des données du menu	7
program_s	Représentation des données du programme	8
sprite_s	Représentation d'une texture du jeu	9
textures_s	Représentation pour stocker les textures nécessaires à l'affichage graphique	11
world_s	Représentation des données du monde	13

Chapter 3

File Index

3.1 File List

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

game_data.h	??
graphic.c	
Module gérant la partie graphique du jeu	15
graphic.h	
En-tête du module gérant la partie graphique du jeu	15
handle_event.c	
Module gérant les entrées utilisateur	21
handle_event.h	
En-tête du module gérant les entrées utilisateur	23
leaderboard.h	??
main.c	
Programme principal	24
main.h	
En-tête du programme principal	26
menu_data.c	
Module de gestion des évènements du menu du jeu	26
menu_data.h	
En-tête du module de gestion des évènements du menu du jeu	28
menu_graphic.h	
Module de gestion des évènements du jeu	29
param.h	
Fichier en-tête avec toutes les constantes et les structures	31
sdl2-light.c	
Sur-couche de SDL2 pour simplifier son utilisation pour le projet	33
sdl2-light.h	
En-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet	36
sdl2-ttf-light.c	
Module de gestion du texte de SDL2	40
sdl2-ttf-light.h	
En-tête du module de gestion du texte de SDL2	42
tests.c	
Fichier de test pour les fonctions du module game_event	44

Chapter 4

Class Documentation

4.1 menu_s Struct Reference

Représentation des données du menu.

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

Public Attributes

- int [menuover](#)
- int [menunumber](#)
- int [currentmenu](#)
- int [optionnumber](#)
- int [currentoption](#)
- int [time](#)
- int [quitte](#)

4.1.1 Detailed Description

Représentation des données du menu.

4.1.2 Member Data Documentation

4.1.2.1 currentmenu

```
int menu_s::currentmenu
```

Le menu actuellement sélectionné.

4.1.2.2 currentoption

```
int menu_s::currentoption
```

Option actuellement sélectionnée.

4.1.2.3 menunumber

```
int menu_s::menunumber
```

Nombre de menu.

4.1.2.4 menuover

```
int menu_s::menuover
```

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

4.1.2.5 optionnumber

```
int menu_s::optionnumber
```

Nombre d'option sur le menu.

4.1.2.6 quitte

```
int menu_s::quitte
```

Informe si l'on affiche l'ecran de fin.

4.1.2.7 time

```
int menu_s::time
```

Sauvegarde le temps d'execution du menu.

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

- [param.h](#)

4.2 program_s Struct Reference

Réprésentation des données du programme.

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

Public Attributes

- int [programover](#)
- int [restart](#)
- int [mode](#)

4.2.1 Detailed Description

Réprésentation des données du programme.

4.2.2 Member Data Documentation

4.2.2.1 mode

```
int program_s::mode
```

Champ indiquant le mode de jeux.

4.2.2.2 programover

```
int program_s::programover
```

Champ indiquant la fin du programme.

4.2.2.3 restart

```
int program_s::restart
```

Champ indiquant le redémarrage du jeux.

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

- [param.h](#)

4.3 sprite_s Struct Reference

Représentation d'une texture du jeu.

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

Public Attributes

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

4.3.1 Detailed Description

Représentation d'une texture du jeu.

4.3.2 Member Data Documentation

4.3.2.1 [h](#)

```
int sprite_s::h
```

Hauteur du sprite.

4.3.2.2 [w](#)

```
int sprite_s::w
```

Largeur du sprite.

4.3.2.3 [x](#)

```
int sprite_s::x
```

Position du sprite sur x.

4.3.2.4 [y](#)

```
int sprite_s::y
```

Position du sprite sur y.

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

- [param.h](#)

4.4 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](#)
- SDL_Texture * [menu_background](#)
- SDL_Texture * [rules](#)
- SDL_Texture * [menu1_select](#)
- SDL_Texture * [menu_replay](#)
- SDL_Texture * [fin](#)
- SDL_Texture * [finb](#)

4.4.1 Detailed Description

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

4.4.2 Member Data Documentation

4.4.2.1 arrival

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

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

4.4.2.2 background

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

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

4.4.2.3 fin

```
SDL_Texture* textures_s::fin
```

Texture liée à la fin du jeu. (si on a gagné)

4.4.2.4 finb

```
SDL_Texture* textures_s::finb
```

Texture liée à la fin du jeu (si on a perdu)

4.4.2.5 font

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

Texture liée à la police utilisée.

4.4.2.6 menu1_select

```
SDL_Texture* textures_s::menu1_select
```

Texture liée à la sélection pour le menu 1.

4.4.2.7 menu_background

```
SDL_Texture* textures_s::menu_background
```

Texture liée au background du menu.

4.4.2.8 menu_replay

```
SDL_Texture* textures_s::menu_replay
```

Texture liée au replay du jeu.

4.4.2.9 meteorite

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

Texture liée à l'image d'un météorite.

4.4.2.10 rules

```
SDL_Texture* textures_s::rules
```

Texture liée aux règles du jeu.

4.4.2.11 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](#)

4.5 world_s Struct Reference

Représentation des données du monde.

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

Collaboration diagram for world_s:

Public Attributes

- int [gameover](#)
- int [levelstart](#)
- [sprite_t](#) [vaisseau](#)
- [sprite_t](#) [arrival](#)
- int [vy](#)
- [sprite_t](#) [mur](#) [[MAX_METEORITE_WALL_NUMBER](#)]
- int [make_disappear](#)
- int [levelnbr](#)
- int [level](#)

4.5.1 Detailed Description

Représentation des données du monde.

4.5.2 Member Data Documentation

4.5.2.1 arrival

```
sprite\_t world_s::arrival
```

Information de la ligne d'arrivée.

4.5.2.2 gameover

```
int world_s::gameover
```

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

4.5.2.3 levelstart

```
int world_s::levelstart
```

Indique le début d'un niveaux.

4.5.2.4 make_disappear

```
int world_s::make_disappear
```

Informe si le vaisseau doit être visible ou pas

4.5.2.5 mur

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

Informations sur un mur d'astéroïdes.

4.5.2.6 vaisseau

```
sprite_t world_s::vaisseau
```

Information du vaisseau.

4.5.2.7 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 `graphic.c` File Reference

Module gérant la partie graphique du jeu.

```
#include "graphic.h"
```

Include dependency graph for `graphic.c`:

5.2 `graphic.h` File Reference

En-tête du module gérant la partie graphique du jeu.

```
#include "param.h"
#include "sdl2-light.h"
#include "game_data.h"
#include "sdl2-ttf-light.h"
#include "leaderboard.h"
```

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

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 render.
- 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 render.
- void `refresh_graphics` (SDL_Renderer *renderer, `world_t` *world, `textures_t` *textures, `menu_t` *menu)
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, `menu_t` *menu, `program_t` *program)

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.

- void `print_end` (SDL_Renderer *renderer, `textures_t` *textures)

Fonction qui applique un background à la fin du jeu si on a gagné

- void `print_end_b` (SDL_Renderer *renderer, `textures_t` *textures)

Fonction qui applique un background à la fin du jeu si on a perdu.

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

La fonction qui commence un niveaux.

5.2.1 Detailed Description

En-tête du module gérant la partie graphique du jeu.

Author

Victor Dallé / Yann Periney

Version

0.1

Date

2021-04-01

Copyright

Copyright (c) 2021

5.2.2 Function Documentation

5.2.2.1 `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.

5.2.2.2 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.

5.2.2.3 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.

5.2.2.4 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.

5.2.2.5 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.

5.2.2.6 clean_data()

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

La fonction nettoie les données du monde.

Parameters

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

5.2.2.7 clean_textures()

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

La fonction nettoie les textures.

Parameters

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

5.2.2.8 init()

```
void init (
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    textures_t * textures,
    world_t * world,
    menu_t * menu,
    program_t * program )
```

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.
<i>menu</i>	Le menu.
<i>program</i>	Les données du programme

5.2.2.9 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.

5.2.2.10 level_start()

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

La fonction qui commence un niveaux.

Parameters

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

5.2.2.11 print_end()

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

Fonction qui applique un background à la fin du jeu si on a gagné

Parameters

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

5.2.2.12 print_end_b()

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

Fonction qui applique un background à la fin du jeu si on a perdu.

Parameters

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

5.2.2.13 refresh_graphics()

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

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.
<i>menu</i>	Utile pour gérer le timer

5.3 handle_event.c File Reference

Module gérant les entrées utilisateur.

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

Include dependency graph for handle_event.c:

Functions

- void [menu_selection](#) ([menu_t](#) *menu, [program_t](#) *program)
La fonction qui gère les évènement selon le menu et les option choissi.
- void [handle_events_menu](#) ([SDL_Event](#) *event, [world_t](#) *world, [menu_t](#) *menu, [program_t](#) *program)
La fonction gère les évènements ayant eu lieu dans le menu et qui n'ont pas encore été traités.
- void [handle_events](#) ([SDL_Event](#) *event, [world_t](#) *world, [menu_t](#) *menu, [program_t](#) *program)
La fonction gère les évènements ayant eu lieu et qui n'ont pas encore été traités.

5.3.1 Detailed Description

Module gérant les entrées utilisateur.

Author

Victor Dallé / Yann Periney

Version

0.1

Date

2021-04-12

Copyright

Copyright (c) 2021

5.3.2 Function Documentation

5.3.2.1 `handle_events()`

```
void handle_events (
    SDL_Event * event,
    world_t * world,
    menu_t * menu,
    program_t * program )
```

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.
<i>menu</i>	Les données du menu.
<i>program</i>	Les données du program.

5.3.2.2 `handle_events_menu()`

```
void handle_events_menu (
    SDL_Event * event,
    world_t * world,
    menu_t * menu,
    program_t * program )
```

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

Parameters

<i>event</i>	Paramètre qui contient les événements.
<i>world</i>	Les données du monde.
<i>menu</i>	Les données du menu.
<i>program</i>	Les données du program.

5.3.2.3 `menu_selection()`

```
void menu_selection (
    menu_t * menu,
    program_t * program )
```

La fonction qui gère les évènement selon le menu et les option choissi.

Parameters

<i>menu</i>	Les données du menu.
<i>program</i>	Les données du program.

5.4 handle_event.h File Reference

En-tête du module gérant les entrées utilisateur.

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

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

Functions

- void [menu_selection](#) ([menu_t](#) *menu, [program_t](#) *program)
La fonction qui gère les évènement selon le menu et les option choissi.
- void [handle_events_menu](#) (SDL_Event *event, [world_t](#) *world, [menu_t](#) *menu, [program_t](#) *program)
La fonction gère les évènements ayant eu lieu dans le menu et qui n'ont pas encore été traités.
- void [handle_events](#) (SDL_Event *event, [world_t](#) *world, [menu_t](#) *menu, [program_t](#) *program)
La fonction gère les évènements ayant eu lieu et qui n'ont pas encore été traités.

5.4.1 Detailed Description

En-tête du module gérant les entrées utilisateur.

Author

Victor Dallé / Yann Periney

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,
    menu\_t * menu,
    program\_t * program )
```

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.
<i>menu</i>	Les données du menu.
<i>program</i>	Les données du program.

5.4.2.2 handle_events_menu()

```
void handle_events_menu (
    SDL_Event * event,
    world_t * world,
    menu_t * menu,
    program_t * program )
```

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

Parameters

<i>event</i>	Paramètre qui contient les événements.
<i>world</i>	Les données du monde.
<i>menu</i>	Les données du menu.
<i>program</i>	Les données du program.

5.4.2.3 menu_selection()

```
void menu_selection (
    menu_t * menu,
    program_t * program )
```

La fonction qui gère les évènement selon le menu et les option choisi.

Parameters

<i>menu</i>	Les données du menu.
<i>program</i>	Les données du program.

5.5 main.c File Reference

Programme principal.

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

Include dependency graph for main.c:

Functions

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

Programme principal qui implémente la boucle du jeu.

5.5.1 Detailed Description

Programme principal.

Author

Mathieu Constant / Yann Periney / Victor Dallé

Version

0.1

Date

2021-03-18

Copyright

Copyright (c) 2021

5.5.2 Function Documentation

5.5.2.1 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

int 0 s'il n'y a pas eu d'erreurs.

5.6 main.h File Reference

En-tête du programme principal.

```
#include <stdio.h>
#include <stdlib.h>
#include <SDL2/SDL.h>
#include "sdl2-light.h"
#include "param.h"
#include "menu_data.h"
#include "menu_graphic.h"
#include "game_data.h"
#include "graphic.h"
#include "handle_event.h"
#include "leaderboard.h"
```

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

5.6.1 Detailed Description

En-tête du programme principal.

Author

Victor Dallé / Yann Periney

Version

0.1

Date

2021-03-18

Copyright

Copyright (c) 2021

5.7 menu_data.c File Reference

Module de gestion des évènements du menu du jeu.

```
#include "menu_data.h"
```

Include dependency graph for menu_data.c:

Functions

- `int is_menu_over (menu_t *menu)`
La fonction indique si le menu est fini.
- `int is_menu_quitte (menu_t *menu)`
La fonction indique si on a quitter par menu.

5.7.1 Detailed Description

Module de gestion des évènements du menu du jeu.

Author

Victor Dallé / Yann Periney

Version

0.1

Date

2021-04-13

Copyright

Copyright (c) 2021

5.7.2 Function Documentation

5.7.2.1 is_menu_over()

```
int is_menu_over (
    menu_t * menu )
```

La fonction indique si le menu est fini.

Parameters

<i>menu</i>	Les données du menu.
-------------	----------------------

Returns

1 si le jeu est fini, 0 sinon.

5.7.2.2 is_menu_quitte()

```
int is_menu_quitte (
    menu_t * menu )
```

La fonction indique si on a quitter par menu.

Parameters

<i>menu</i>	Les données du menu.
-------------	----------------------

Returns

1 si le jeu est fini, 0 sinon.

5.8 menu_data.h File Reference

En-tête du module de gestion des évènements du menu du jeu.

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

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

Functions

- int [is_menu_over](#) ([menu_t](#) *menu)
La fonction indique si le menu est fini.
- int [is_menu_quitte](#) ([menu_t](#) *menu)
La fonction indique si on a quitter par menu.

5.8.1 Detailed Description

En-tête du module de gestion des évènements du menu du jeu.

Author

Victor Dallé / Yann Periney

Version

0.1

Date

2021-04-13

Copyright

Copyright (c) 2021

5.8.2 Function Documentation

5.8.2.1 is_menu_over()

```
int is_menu_over (
    menu\_t * menu )
```

La fonction indique si le menu est fini.

Parameters

<i>menu</i>	Les données du menu.
-------------	----------------------

Returns

1 si le jeu est fini, 0 sinon.

5.8.2.2 is_menu_quitte()

```
int is_menu_quitte (
    menu_t * menu )
```

La fonction indique si on a quitter par menu.

Parameters

<i>menu</i>	Les données du menu.
-------------	----------------------

Returns

1 si le jeu est fini, 0 sinon.

5.9 menu_graphic.h File Reference

Module de gestion des évènements du jeu.

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

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

Functions

- void [refresh_menu_graphics](#) (SDL_Renderer *renderer, [menu_t](#) *menu, [textures_t](#) *textures)
Fonction qui applique la texture du menu sur le renderer lié à l'écran de jeu.
- void [apply_menu](#) (SDL_Renderer *renderer, [menu_t](#) *menu, [textures_t](#) *textures)
Fonction qui applique la texture du menu.
- void [apply_select](#) (SDL_Renderer *renderer, [menu_t](#) *menu, [textures_t](#) *textures)
Fonction qui applique la texture du selecteur au menu.

5.9.1 Detailed Description

Module de gestion des évènements du jeu.

Author

Victor Dallé / Yann Periney

Version

0.1

Date

2021-04-13

Copyright

Copyright (c) 2021

5.9.2 Function Documentation

5.9.2.1 apply_menu()

```
void apply_menu (
    SDL_Renderer * renderer,
    menu_t * menu,
    textures_t * textures )
```

Fonction qui applique la texture du menu.

Parameters

<i>renderer</i>	Le renderer.
<i>menu</i>	Paramètre du menu.
<i>texture</i>	La texture liée au fond.

5.9.2.2 apply_select()

```
void apply_select (
    SDL_Renderer * renderer,
    menu_t * menu,
    textures_t * textures )
```

Fonction qui applique la texture du selecteur au menu.

Parameters

<i>renderer</i>	Le renderer.
<i>menu</i>	Paramètre du menu.
<i>texture</i>	La texture liée au fond.

5.9.2.3 refresh_menu_graphics()

```
void refresh_menu_graphics (
    SDL_Renderer * renderer,
    menu_t * menu,
    textures_t * textures )
```

Fonction qui applique la texture du menu sur le renderer lié à l'écran de jeu.

Parameters

<i>renderer</i>	Le renderer.
<i>menu</i>	Paramètre du menu.
<i>texture</i>	La texture liée au fond.

5.10 param.h File Reference

Fichier en-tête avec toutes les constantes et les structures.

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <time.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 des données du monde.
- struct [menu_s](#)
Représentation des données du menu.
- struct [program_s](#)
Représentation des données du programme.

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` 16
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.
- typedef struct `menu_s menu_t`
Type qui correspond aux données du menu.
- typedef struct `program_s program_t`
Type qui correspond aux données du programme.

5.10.1 Detailed Description

Fichier en-tête avec toutes les constantes et les structures.

Author

Yann Periney / Victor Dallé

Version

0.1

Date

2021-04-01

Copyright

Copyright (c) 2021

5.11 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.

5.11.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

5.11.2 Function Documentation

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

5.11.2.2 clean_sdl()

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

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

Parameters

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

5.11.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

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

5.11.2.4 clear_render()

```
void clear_render (
    SDL_Renderer * render )
```

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

Parameters

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

5.11.2.5 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

5.11.2.6 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)

5.11.2.7 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
-------------	----------------------------------

5.11.2.8 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
-----------------	------------------------

5.12 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.

5.12.1 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

5.12.2 Function Documentation

5.12.2.1 `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)

5.12.2.2 `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

5.12.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

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

5.12.2.4 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
-----------------	------------------------

5.12.2.5 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

5.12.2.6 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)

5.12.2.7 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
-------------	----------------------------------

5.12.2.8 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
-----------------	------------------------

5.13 sdl2-ttf-light.c File Reference

Module de gestion du texte de SDL2.

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

Include dependency graph for sdl2-ttf-light.c:

Functions

- void [init_ttf](#) ()
La fonction initialise l'environnement TTF.
- TTF_Font * [load_font](#) (const char *path, int font_size)
La fonction charge une police.
- void [apply_text](#) (SDL_Renderer *renderer, int x, int y, int w, int h, const char *text, TTF_Font *font)
La fonction applique un texte dans une certaine police sur le renderer à une certaine position et avec une certaine dimension.
- void [clean_font](#) (TTF_Font *font)
La fonction nettoie une police en mémoire.

5.13.1 Detailed Description

Module de gestion du texte de SDL2.

Author

Mathieu Constant

Version

0.1

Date

2021-04-12

Copyright

Copyright (c) 2021

5.13.2 Function Documentation

5.13.2.1 apply_text()

```
void apply_text (
    SDL_Renderer * renderer,
    int x,
    int y,
    int w,
    int h,
    const char * text,
    TTF_Font * font )
```

La fonction applique un texte dans une certaine police sur le renderer à une certaine position et avec une certaine dimension.

Parameters

<i>renderer</i>	le renderer
<i>x</i>	abscisse du coin en haut à gauche du texte
<i>y</i>	son abscisse
<i>w</i>	la largeur du message
<i>h</i>	sa hauteur
<i>text</i>	le texte à afficher
<i>font</i>	la police

5.13.2.2 clean_font()

```
void clean_font (
    TTF_Font * font )
```

La fonction nettoie une police en mémoire.

Parameters

<i>font</i>	la police
-------------	-----------

5.13.2.3 load_font()

```
TTF_Font* load_font (
    const char * path,
    int font_size )
```

La fonction charge une police.

Parameters

<i>path</i>	le chemin du fichier correspondant à la police
<i>font_size</i>	la taille de la police

Returns

la police chargée

5.14 sdl2-ttf-light.h File Reference

En-tête du module de gestion du texte de SDL2.

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

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

Functions

- void [init_ttf](#) ()
La fonction initialise l'environnement TTF.
- TTF_Font * [load_font](#) (const char *path, int font_size)
La fonction charge une police.
- void [apply_text](#) (SDL_Renderer *renderer, int x, int y, int w, int h, const char *text, TTF_Font *font)
La fonction applique un texte dans une certaine police sur le renderer à une certaine position et avec une certaine dimension.
- void [clean_font](#) (TTF_Font *font)
La fonction nettoie une police en mémoire.

5.14.1 Detailed Description

En-tête du module de gestion du texte de SDL2.

Author

Mathieu Constant

Version

0.1

Date

2021-04-12

Copyright

Copyright (c) 2021

5.14.2 Function Documentation

5.14.2.1 apply_text()

```
void apply_text (
    SDL_Renderer * renderer,
    int x,
    int y,
    int w,
    int h,
    const char * text,
    TTF_Font * font )
```

La fonction applique un texte dans une certaine police sur le renderer à une certaine position et avec une certaine dimension.

Parameters

<i>renderer</i>	le renderer
<i>x</i>	abscisse du coin en haut à gauche du texte
<i>y</i>	son abscisse
<i>w</i>	la largeur du message
<i>h</i>	sa hauteur
<i>text</i>	le texte à afficher
<i>font</i>	la police

5.14.2.2 clean_font()

```
void clean_font (
    TTF_Font * font )
```

La fonction nettoie une police en mémoire.

Parameters

<i>font</i>	la police
-------------	-----------

5.14.2.3 load_font()

```
TTF_Font* load_font (
    const char * path,
    int font_size )
```

La fonction charge une police.

Parameters

<i>path</i>	le chemin du fichier correspondant à la police
<i>font_size</i>	la taille de la police

Returns

la police chargée

5.15 tests.c File Reference

Fichier de test pour les fonctions du module game_event.

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

5.15.1 Detailed Description

Fichier de test pour les fonctions du module game_event.

Author

Yann Periney / Victor Dallé

Version

0.1

Date

2021-03-01

Copyright

Copyright (c) 2021

5.15.2 Function Documentation

5.15.2.1 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.

5.15.2.2 print_sprite()

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

Fonction qui affiche les coordonnées d'un sprite.

Parameters

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

5.15.2.3 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.

5.15.2.4 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.

5.15.2.5 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.
--------------	-------------------

5.15.2.6 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.h, [16](#)
- apply_menu
 - menu_graphic.h, [30](#)
- apply_select
 - menu_graphic.h, [30](#)
- apply_sprite
 - graphic.h, [17](#)
- apply_text
 - sdl2-ttf-light.c, [40](#)
 - sdl2-ttf-light.h, [42](#)
- apply_texture
 - sdl2-light.c, [33](#)
 - sdl2-light.h, [37](#)
- apply_wall
 - graphic.h, [17](#)
- apply_walls
 - graphic.h, [18](#)
- arrival
 - textures_s, [11](#)
 - world_s, [13](#)
- background
 - textures_s, [11](#)
- clean
 - graphic.h, [18](#)
- clean_data
 - graphic.h, [18](#)
- clean_font
 - sdl2-ttf-light.c, [41](#)
 - sdl2-ttf-light.h, [43](#)
- clean_sdl
 - sdl2-light.c, [34](#)
 - sdl2-light.h, [37](#)
- clean_texture
 - sdl2-light.c, [34](#)
 - sdl2-light.h, [38](#)
- clean_textures
 - graphic.h, [19](#)
- clear_renderer
 - sdl2-light.c, [34](#)
 - sdl2-light.h, [38](#)
- currentmenu
 - menu_s, [7](#)
- currentoption
 - menu_s, [7](#)
- fin
 - textures_s, [11](#)
- finb
 - textures_s, [11](#)
- font
 - textures_s, [12](#)
- gameover
 - world_s, [13](#)
- graphic.c, [15](#)
- graphic.h, [15](#)
 - apply_background, [16](#)
 - apply_sprite, [17](#)
 - apply_wall, [17](#)
 - apply_walls, [18](#)
 - clean, [18](#)
 - clean_data, [18](#)
 - clean_textures, [19](#)
 - init, [19](#)
 - init_textures, [19](#)
 - level_start, [20](#)
 - print_end, [20](#)
 - print_end_b, [20](#)
 - refresh_graphics, [21](#)
- h
 - sprite_s, [10](#)
- handle_event.c, [21](#)
 - handle_events, [22](#)
 - handle_events_menu, [22](#)
 - menu_selection, [22](#)
- handle_event.h, [23](#)
 - handle_events, [23](#)
 - handle_events_menu, [24](#)
 - menu_selection, [24](#)
- handle_events
 - handle_event.c, [22](#)
 - handle_event.h, [23](#)
- handle_events_menu
 - handle_event.c, [22](#)
 - handle_event.h, [24](#)
- init
 - graphic.h, [19](#)
- init_sdl
 - sdl2-light.c, [35](#)
 - sdl2-light.h, [38](#)
- init_textures
 - graphic.h, [19](#)
- is_menu_over
 - menu_data.c, [27](#)
 - menu_data.h, [28](#)

- is_menu_quitte
 - menu_data.c, [27](#)
 - menu_data.h, [29](#)
- level_start
 - graphic.h, [20](#)
- levelstart
 - world_s, [14](#)
- load_font
 - sdl2-ttf-light.c, [41](#)
 - sdl2-ttf-light.h, [43](#)
- load_image
 - sdl2-light.c, [35](#)
 - sdl2-light.h, [39](#)
- main
 - main.c, [25](#)
 - tests.c, [45](#)
- main.c, [24](#)
 - main, [25](#)
- main.h, [26](#)
- make_disappear
 - world_s, [14](#)
- menu1_select
 - textures_s, [12](#)
- menu_background
 - textures_s, [12](#)
- menu_data.c, [26](#)
 - is_menu_over, [27](#)
 - is_menu_quitte, [27](#)
- menu_data.h, [28](#)
 - is_menu_over, [28](#)
 - is_menu_quitte, [29](#)
- menu_graphic.h, [29](#)
 - apply_menu, [30](#)
 - apply_select, [30](#)
 - refresh_menu_graphics, [31](#)
- menu_replay
 - textures_s, [12](#)
- menu_s, [7](#)
 - currentmenu, [7](#)
 - currentoption, [7](#)
 - menunumber, [8](#)
 - menuover, [8](#)
 - optionnumber, [8](#)
 - quitte, [8](#)
 - time, [8](#)
- menu_selection
 - handle_event.c, [22](#)
 - handle_event.h, [24](#)
- menunumber
 - menu_s, [8](#)
- menuover
 - menu_s, [8](#)
- meteorite
 - textures_s, [12](#)
- mode
 - program_s, [9](#)
- mur
 - world_s, [14](#)
- optionnumber
 - menu_s, [8](#)
- param.h, [31](#)
- pause
 - sdl2-light.c, [35](#)
 - sdl2-light.h, [39](#)
- print_end
 - graphic.h, [20](#)
- print_end_b
 - graphic.h, [20](#)
- print_sprite
 - tests.c, [45](#)
- program_s, [8](#)
 - mode, [9](#)
 - programover, [9](#)
 - restart, [9](#)
- programover
 - program_s, [9](#)
- quitte
 - menu_s, [8](#)
- refresh_graphics
 - graphic.h, [21](#)
- refresh_menu_graphics
 - menu_graphic.h, [31](#)
- restart
 - program_s, [9](#)
- rules
 - textures_s, [12](#)
- sdl2-light.c, [33](#)
 - apply_texture, [33](#)
 - clean_sdl, [34](#)
 - clean_texture, [34](#)
 - clear_renderer, [34](#)
 - init_sdl, [35](#)
 - load_image, [35](#)
 - pause, [35](#)
 - update_screen, [36](#)
- sdl2-light.h, [36](#)
 - apply_texture, [37](#)
 - clean_sdl, [37](#)
 - clean_texture, [38](#)
 - clear_renderer, [38](#)
 - init_sdl, [38](#)
 - load_image, [39](#)
 - pause, [39](#)
 - update_screen, [39](#)
- sdl2-ttf-light.c, [40](#)
 - apply_text, [40](#)
 - clean_font, [41](#)
 - load_font, [41](#)
- sdl2-ttf-light.h, [42](#)
 - apply_text, [42](#)
 - clean_font, [43](#)

- load_font, [43](#)
- sprite_s, [9](#)
 - h, [10](#)
 - w, [10](#)
 - x, [10](#)
 - y, [10](#)
- test_handle_sprites_collision_param
 - tests.c, [45](#)
- test_init_sprite_param
 - tests.c, [46](#)
- test_out_of_screen_param
 - tests.c, [46](#)
- test_sprites_collide_param
 - tests.c, [46](#)
- tests.c, [44](#)
 - main, [45](#)
 - print_sprite, [45](#)
 - test_handle_sprites_collision_param, [45](#)
 - test_init_sprite_param, [46](#)
 - test_out_of_screen_param, [46](#)
 - test_sprites_collide_param, [46](#)
- textures_s, [11](#)
 - arrival, [11](#)
 - background, [11](#)
 - fin, [11](#)
 - finb, [11](#)
 - font, [12](#)
 - menu1_select, [12](#)
 - menu_background, [12](#)
 - menu_replay, [12](#)
 - meteorite, [12](#)
 - rules, [12](#)
 - vaisseau, [12](#)
- time
 - menu_s, [8](#)
- update_screen
 - sdl2-light.c, [36](#)
 - sdl2-light.h, [39](#)
- vaisseau
 - textures_s, [12](#)
 - world_s, [14](#)
- vy
 - world_s, [14](#)
- w
 - sprite_s, [10](#)
- world_s, [13](#)
 - arrival, [13](#)
 - gameover, [13](#)
 - levelstart, [14](#)
 - make_disappear, [14](#)
 - mur, [14](#)
 - vaisseau, [14](#)
 - vy, [14](#)
- x
 - sprite_s, [10](#)
- y
 - sprite_s, [10](#)