My Project

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 sprite_s Struct Reference

Représentation des parametres du sprite du jeu.

`#include <data.h>`

Public Attributes

- int **x**
- int **y**
- int **h**
- int **w**

3.1.1 Detailed Description

Représentation des parametres du sprite du jeu.

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

- data.h

3.2 textures_s Struct Reference

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

`#include <graphics.h>`

Public Attributes

- SDL_Texture * background
- SDL_Texture * **sprite**
- SDL_Texture * **finish_line**
- SDL_Texture * **meteorites**

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.

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

- graphics.h

3.3 world_s Struct Reference

Représentation du monde du jeu.

```
#include <data.h>
```

Collaboration diagram for world_s:

Public Attributes

- sprite_t **vaisseau**
- sprite_t **finish_line**
- sprite_t **mur**
- int gameover
- int **vy**

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

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

- data.h

Chapter 4

File Documentation

4.1 data.c File Reference

Module d'initialisation.

```
#include "constantes.h"
#include "data.h"
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for data.c:

4.2 data.h File Reference

Module d'initialisation.

This graph shows which files directly or indirectly include this file:

Classes

- struct sprite_s

  Représentation des parametres du sprite du jeu.
- struct world_s

  Représentation du monde du jeu.

Typedefs

- typedef struct sprite_s sprite_t

  Type qui correspond au sprite.
- typedef struct world_s world_t

  Type qui correspond aux données du monde.

Functions

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

  La fonction initialise les parametres nécessaires à l'affichage graphique 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.
- void update_data (world_t *world)

  La fonction met à jour les données en tenant compte de la physique du monde.
- int is_game_over (world_t *world)

  La fonction indique si le jeu est fini en fonction des données du monde.
- void depasse (sprite_t *sprite)

  La fonction teste si le sprite est sorti du cadre de jeu, et replace le sprite si c'est le cas.
- void handle_sprites_collision (world_t *world, sprite_t *sp1, sprite_t *sp2, int make_disappear)

  La fonction appelle la fonction de detection de collisions entre les sprites sp1 et sp2. Si ils sont en collision, la vitesse du monde devient 0.
- void print_sprite (sprite_t *sprite)

  La fonction affiche les coordonnees et la taille du sprite.
- int sprites_collide (sprite_t *sp1, sprite_t *sp2)

  La fonction de detection de collisions entre les sprites sp1 et sp2.

4.2.1 Detailed Description

Module d'initialisation.

Author

Ludovic Tagnon / Matheo Serrier

Version

3.0

Date

7 avril 2021

4.2.2 Function Documentation

4.2.2.1 clean_data()

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

La fonction nettoie les données du monde.

Parameters

world	les données du monde

4.2.2.2 depasse()

```
void depasse (
            sprite_t * sprite )
```

La fonction teste si le sprite est sorti du cadre de jeu, et replace le sprite si c'est le cas.

Parameters

sprite	Le sprite

4.2.2.3 handle_sprites_collision()

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

La fonction appelle la fonction de detection de collisions entre les sprites sp1 et sp2. Si ils sont en collision, la vitesse du monde devient 0.

Parameters

world	les données du monde
sp1	Le 1er sprite
sp2	Le 2eme sprite
make_disappear	la condition qui fait disparraitre le vaisseau si il est en collision

4.2.2.4 init_data()

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

La fonction initialise les données du monde du jeu.

Parameters

world	les données du monde

4.2.2.5 init_sprite()

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

La fonction initialise les parametres nécessaires à l'affichage graphique du sprite.

Parameters

sprite	Le sprite
x,y	la position du sprite
w,h	taille du sprite

4.2.2.6 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

world	les données du monde

Returns

1 si le jeu est fini, 0 sinon

4.2.2.7 print_sprite()

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

La fonction affiche les coordonnees et la taille du sprite.

Parameters

sprite	Le sprite

4.2.2.8 sprites_collide()

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

La fonction de detection de collisions entre les sprites sp1 et sp2.

Parameters

sp1	Le 1er sprite
sp2	Le 2eme sprite

4.2.2.9 update_data()

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

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

Parameters

les	données du monde

4.3 graphics.c File Reference

Module d'affichage des elements du jeu.

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

Include dependency graph for graphics.c:

Functions

- void init_textures (SDL_Renderer ∗renderer, textures_t ∗textures)

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

- void clean_textures (textures_t ∗textures)

  La fonction nettoie les textures.
- 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_sprite (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

  La fonction applique la texture du sprite sur le renderer.
- void handle_wall (SDL_Renderer ∗renderer, textures_t ∗textures, world_t ∗world, sprite_t temp[world->mur.w/METEORITE_SIZE][world->mur.h/METEORITE_SIZE])

  La fonction copie les coordonnees intiales de la 1ere meteorite et ajuste les coordonnees des autres (chaque case du tableau) avant de les afficher.
- 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 handle_events (SDL_Event ∗event, world_t ∗world)

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

4.3.1 Detailed Description

Module d'affichage des elements du jeu.

Author

Ludovic Tagnon / Matheo Serrier

Version

3.0

Date

7 avril 2021

4.3.2 Function Documentation

4.3.2.1 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

renderer	le renderer
texture	la texture liée au fond

4.3.2.2 apply_sprite()

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

La fonction applique la texture du sprite sur le renderer.

Parameters

renderer	le renderer
texture	la texture liée au fond
sprite	le sprite a afficher

4.3.2.3 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

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.3.2.4 clean_textures()

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

La fonction nettoie les textures.

Parameters

textures	les textures

4.3.2.5 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

event	paramètre qui contient les événements
world	les données du monde

4.3.2.6 handle_wall()

```
void handle_wall (
            SDL_Renderer * renderer,
            textures_t * textures,
            world_t * world,
            sprite_t temp[world->mur.w/METEORITE_SIZE][world->mur.h/METEORITE_SIZE] )
```

La fonction copie les coordonnees intiales de la 1ere meteorite et ajuste les coordonnees des autres (chaque case du tableau) avant de les afficher.

Parameters

renderer	le renderer
world	les données du monde
textures	les textures
sprite[w][h]	les meteorites/sprites a afficher

4.3.2.7 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

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.3.2.8 init_textures()

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

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

Parameters

screen	la surface correspondant à l'écran de jeu
textures	les textures du jeu

4.3.2.9 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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
textures	les textures

4.4 graphics.h File Reference

Module d'affichage des elements du jeu.

```
#include "data.h"
```

Include dependency graph for graphics.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.

Typedefs

- typedef struct textures_s textures_t

  Type qui correspond aux textures du jeu.

Functions

- void init_textures (SDL_Renderer ∗renderer, textures_t ∗textures)

  La fonction initialise les textures nécessaires à l'affichage graphique du jeu.
- 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
- void clean_textures (textures_t ∗textures)

  La fonction nettoie les textures.
- 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_sprite (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

  La fonction applique la texture du sprite sur le renderer.
- void handle_wall (SDL_Renderer ∗renderer, textures_t ∗textures, world_t ∗world, sprite_t temp[world->mur.w/METEORITE_SIZE][world->mur.h/METEORITE_SIZE])

  La fonction copie les coordonnees intiales de la 1ere meteorite et ajuste les coordonnees des autres (chaque case du tableau) avant de les afficher.
- 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 handle_events (SDL_Event ∗event, world_t ∗world)

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

4.4.1 Detailed Description

Module d'affichage des elements du jeu.

Author

Ludovic Tagnon / Matheo Serrier

Version

3.0

Date

7 avril 2021

4.4.2 Function Documentation

4.4.2.1 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

renderer	le renderer
texture	la texture liée au fond

4.4.2.2 apply_sprite()

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

La fonction applique la texture du sprite sur le renderer.

Parameters

renderer	le renderer
texture	la texture liée au fond
sprite	le sprite a afficher

4.4.2.3 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

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.4.2.4 clean_textures()

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

La fonction nettoie les textures.

Parameters

textures	les textures

4.4.2.5 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

event	paramètre qui contient les événements
world	les données du monde

4.4.2.6 handle_wall()

```
void handle_wall (
            SDL_Renderer * renderer,
            textures_t * textures,
            world_t * world,
            sprite_t temp[world->mur.w/METEORITE_SIZE][world->mur.h/METEORITE_SIZE] )
```

La fonction copie les coordonnees intiales de la 1ere meteorite et ajuste les coordonnees des autres (chaque case du tableau) avant de les afficher.

Parameters

renderer	le renderer
world	les données du monde
textures	les textures
sprite[w][h]	les meteorites/sprites a afficher

4.4.2.7 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

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.4.2.8 init_textures()

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

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

Parameters

screen	la surface correspondant à l'écran de jeu
textures	les textures du jeu

4.4.2.9 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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
textures	les textures

4.5 main.c File Reference

Programme initial du niveau 1.

```
#include "sdl2-light.h"
#include "constantes.h"
#include "data.h"
#include "graphics.h"
```

Include dependency graph for main.c:

Functions

- int main (int argc, char *args[ ])

  programme principal qui implémente la boucle du jeu

4.5.1 Detailed Description

Programme initial du niveau 1.

Author

Ludovic Tagnon / Matheo Serrier

Version

3.0

Date

6 avril 2021

4.6 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>
```

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.

4.6.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

4.6.2 Function Documentation

4.6.2.1 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

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

4.6.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

4.6.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

4.6.2.4 clear_renderer()

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

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

Parameters

renderer	le renderer de l'écran

4.6.2.5 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

window	la fenêtre du jeu
renderer	le renderer
width	largeur de l'écran de jeu
height	hauteur de l'écran de jeu

Returns

-1 en cas d'erreur, 0 sinon

4.6.2.6 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

path	est le chemin du fichier image. Le fichier doit être obligatoirement du BMP.
renderer	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)

4.6.2.7 pause()

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

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

Parameters

time	ce laps de temps en milliseconde

4.6.2.8 update_screen()

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

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

Parameters

renderer	le renderer de l'écran

4.7 sdl2-light.h File Reference

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

```
#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.

4.7.1 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

4.7.2 Function Documentation

4.7.2.1 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

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

4.7.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

4.7.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

4.7.2.4 clear_renderer()

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

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

Parameters

renderer	le renderer de l'écran

4.7.2.5 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

window	la fenêtre du jeu
renderer	le renderer
width	largeur de l'écran de jeu
height	hauteur de l'écran de jeu

Returns

-1 en cas d'erreur, 0 sinon

4.7.2.6 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

path	est le chemin du fichier image. Le fichier doit être obligatoirement du BMP.
renderer	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)

4.7.2.7 pause()

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

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

Parameters

time	ce laps de temps en milliseconde

4.7.2.8 update_screen()

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

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

Parameters

renderer	le renderer de l'écran

4.8 tests.c File Reference

Module de test.

```
#include <stdlib.h>
#include <stdio.h>
#include "data.h"
#include "constantes.h"
```

Include dependency graph for tests.c:

Functions

- void **test_init_sprite_param** (sprite_t ∗s)
- void **test_depasse_param** (sprite_t ∗s)
- void **test_sprites_collide_param** (sprite_t ∗sp1, sprite_t ∗sp2)
- void **test_handle_sprites_collision_param** (world_t ∗world, sprite_t ∗sp1, sprite_t ∗sp2)
- void **test_init_sprite** ()
- void **test_depasse** ()
- void **test_sprites_collide** ()
- void **test_handle_sprites_collision** ()
- int **main** ()

4.8.1 Detailed Description

Module de test.

Author

Ludovic Tagnon / Matheo Serrier

Version

3.0

Date

7 avril 2021

Index