

My Project

Generated by Doxygen 1.8.17

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 sprite_s Struct Reference	5
3.1.1 Detailed Description	5
3.2 textures_s Struct Reference	5
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 background	6
3.3 world_s Struct Reference	6
3.3.1 Detailed Description	6
3.3.2 Member Data Documentation	6
3.3.2.1 gameover	6
4 File Documentation	7
4.1 data.c File Reference	7
4.2 data.h File Reference	7
4.2.1 Detailed Description	8
4.2.2 Function Documentation	8
4.2.2.1 clean_data()	8
4.2.2.2 depasse()	9
4.2.2.3 handle_sprites_collision()	9
4.2.2.4 init_data()	9
4.2.2.5 init_sprite()	10
4.2.2.6 is_game_over()	10
4.2.2.7 print_sprite()	10
4.2.2.8 sprites_collide()	11
4.2.2.9 update_data()	11
4.3 graphics.c File Reference	11
4.3.1 Detailed Description	12
4.3.2 Function Documentation	12
4.3.2.1 apply_background()	12
4.3.2.2 apply_sprite()	13
4.3.2.3 clean()	13
4.3.2.4 clean_textures()	13
4.3.2.5 handle_events()	14
4.3.2.6 handle_wall()	14
4.3.2.7 init()	14
4.3.2.8 init_textures()	16

4.3.2.9 refresh_graphics()	16
4.4 graphics.h File Reference	16
4.4.1 Detailed Description	17
4.4.2 Function Documentation	18
4.4.2.1 apply_background()	18
4.4.2.2 apply_sprite()	18
4.4.2.3 clean()	18
4.4.2.4 clean_textures()	19
4.4.2.5 handle_events()	19
4.4.2.6 handle_wall()	19
4.4.2.7 init()	20
4.4.2.8 init_textures()	20
4.4.2.9 refresh_graphics()	20
4.5 main.c File Reference	21
4.5.1 Detailed Description	21
4.6 sdl2-light.c File Reference	21
4.6.1 Detailed Description	22
4.6.2 Function Documentation	22
4.6.2.1 apply_texture()	22
4.6.2.2 clean_sdl()	23
4.6.2.3 clean_texture()	23
4.6.2.4 clear_renderer()	23
4.6.2.5 init_sdl()	24
4.6.2.6 load_image()	24
4.6.2.7 pause()	24
4.6.2.8 update_screen()	25
4.7 sdl2-light.h File Reference	25
4.7.1 Detailed Description	26
4.7.2 Function Documentation	26
4.7.2.1 apply_texture()	26
4.7.2.2 clean_sdl()	26
4.7.2.3 clean_texture()	27
4.7.2.4 clear_renderer()	27
4.7.2.5 init_sdl()	27
4.7.2.6 load_image()	28
4.7.2.7 pause()	28
4.7.2.8 update_screen()	28
4.8 tests.c File Reference	29
4.8.1 Detailed Description	29

Chapter 1

Class Index

1.1 Class List

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

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

Chapter 2

File Index

2.1 File List

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

constantes.h	...	??
data.c	Module d'initialisation	7
data.h	Module d'initialisation	7
graphics.c	Module d'affichage des elements du jeu	11
graphics.h	Module d'affichage des elements du jeu	16
main.c	Programme initial du niveau 1	21
sdl2-light.c	Sur-couche de SDL2 pour simplifier son utilisation pour le projet	21
sdl2-light.h	En-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet	25
tests.c	Module de test	29

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

<i>world</i>	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

<i>sprite</i>	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

<i>world</i>	les données du monde
<i>sp1</i>	Le 1er sprite
<i>sp2</i>	Le 2eme sprite
<i>make_disappear</i>	la condition qui fait disparaitre 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

<i>world</i>	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

<i>sprite</i>	Le sprite
<i>x,y</i>	la position du sprite
<i>w,h</i>	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

<i>world</i>	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

<i>sprite</i>	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

<i>sp1</i>	Le 1er sprite
<i>sp2</i>	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

<i>/es</i>	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 initiales 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

<i>renderer</i>	le renderer
<i>texture</i>	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

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

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

4.3.2.4 clean_textures()

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

La fonction nettoie les textures.

Parameters

<i>textures</i>	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

<i>event</i>	paramètre qui contient les événements
<i>world</i>	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 coordonnées initiales de la 1ère météorite et ajuste les coordonnées des autres (chaque case du tableau) avant de les afficher.

Parameters

<i>renderer</i>	le renderer
<i>world</i>	les données du monde
<i>textures</i>	les textures
<i>sprite[w][h]</i>	les météorites/sprites à 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

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

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

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>textures</i>	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 coordonnées initiales de la 1ere meteorite et ajuste les coordonnées 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 événements ayant eu lieu et qui n'ont pas encore été traités.

4.4.1 Detailed Description

Module d'affichage des éléments 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

<i>renderer</i>	le renderer
<i>texture</i>	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

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

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

4.4.2.4 clean_textures()

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

La fonction nettoie les textures.

Parameters

<i>textures</i>	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

<i>event</i>	paramètre qui contient les événements
<i>world</i>	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 coordonnées initiales de la 1ère météorite et ajuste les coordonnées des autres (chaque case du tableau) avant de les afficher.

Parameters

<i>renderer</i>	le renderer
<i>world</i>	les données du monde
<i>textures</i>	les textures
<i>sprite[w][h]</i>	les météorites/sprites à 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

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

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

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

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

4.6.2.2 clean_sdl()

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

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

Parameters

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

4.6.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

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

4.6.2.4 clear_renderer()

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

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

Parameters

<i>renderer</i>	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

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

Returns

-1 en cas d'erreur, 0 sinon

4.6.2.6 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

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

Returns

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

4.6.2.7 pause()

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

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

Parameters

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

4.6.2.8 update_screen()

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

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

Parameters

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

4.7 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

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

4.7.2.2 `clean_sdl()`

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

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

Parameters

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

4.7.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

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

4.7.2.4 clear_renderer()

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

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

Parameters

<i>renderer</i>	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

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

Returns

-1 en cas d'erreur, 0 sinon

4.7.2.6 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

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

Returns

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

4.7.2.7 pause()

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

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

Parameters

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

4.7.2.8 update_screen()

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

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

Parameters

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

4.8 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 / Mattheo Serrier

Version

3.0

Date

7 avril 2021

Index

- apply_background
 - graphics.c, 12
 - graphics.h, 18
- apply_sprite
 - graphics.c, 13
 - graphics.h, 18
- apply_texture
 - sdl2-light.c, 22
 - sdl2-light.h, 26
- background
 - textures_s, 6
- clean
 - graphics.c, 13
 - graphics.h, 18
- clean_data
 - data.h, 8
- clean_sdl
 - sdl2-light.c, 23
 - sdl2-light.h, 26
- clean_texture
 - sdl2-light.c, 23
 - sdl2-light.h, 27
- clean_textures
 - graphics.c, 13
 - graphics.h, 19
- clear_renderer
 - sdl2-light.c, 23
 - sdl2-light.h, 27
- data.c, 7
- data.h, 7
 - clean_data, 8
 - depasse, 9
 - handle_sprites_collision, 9
 - init_data, 9
 - init_sprite, 10
 - is_game_over, 10
 - print_sprite, 10
 - sprites_collide, 11
 - update_data, 11
- depasse
 - data.h, 9
- gameover
 - world_s, 6
- graphics.c, 11
 - apply_background, 12
 - apply_sprite, 13
 - clean, 13
 - clean_textures, 13
 - handle_events, 14
 - handle_wall, 14
 - init, 14
 - init_textures, 16
 - refresh_graphics, 16
- graphics.h, 16
 - apply_background, 18
 - apply_sprite, 18
 - clean, 18
 - clean_textures, 19
 - handle_events, 19
 - handle_wall, 19
 - init, 20
 - init_textures, 20
 - refresh_graphics, 20
- handle_events
 - graphics.c, 14
 - graphics.h, 19
- handle_sprites_collision
 - data.h, 9
- handle_wall
 - graphics.c, 14
 - graphics.h, 19
- init
 - graphics.c, 14
 - graphics.h, 20
- init_data
 - data.h, 9
- init_sdl
 - sdl2-light.c, 24
 - sdl2-light.h, 27
- init_sprite
 - data.h, 10
- init_textures
 - graphics.c, 16
 - graphics.h, 20
- is_game_over
 - data.h, 10
- load_image
 - sdl2-light.c, 24
 - sdl2-light.h, 28
- main.c, 21
- pause
 - sdl2-light.c, 24

- sdl2-light.h, [28](#)
- print_sprite
 - data.h, [10](#)
- refresh_graphics
 - graphics.c, [16](#)
 - graphics.h, [20](#)
- sdl2-light.c, [21](#)
 - apply_texture, [22](#)
 - clean_sdl, [23](#)
 - clean_texture, [23](#)
 - clear_renderer, [23](#)
 - init_sdl, [24](#)
 - load_image, [24](#)
 - pause, [24](#)
 - update_screen, [25](#)
- sdl2-light.h, [25](#)
 - apply_texture, [26](#)
 - clean_sdl, [26](#)
 - clean_texture, [27](#)
 - clear_renderer, [27](#)
 - init_sdl, [27](#)
 - load_image, [28](#)
 - pause, [28](#)
 - update_screen, [28](#)
- sprite_s, [5](#)
- sprites_collide
 - data.h, [11](#)
- tests.c, [29](#)
- textures_s, [5](#)
 - background, [6](#)
- update_data
 - data.h, [11](#)
- update_screen
 - sdl2-light.c, [25](#)
 - sdl2-light.h, [28](#)
- world_s, [6](#)
 - gameover, [6](#)