My Project

Generated by Doxygen 1.8.18

Chapter 1

Règles du jeu

1.1 Comment jouer ?

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

1.2 Le but du jeu

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 sprite_s Struct Reference

Représentation d'une texture du jeu.

`#include <param.h>`

Public Attributes

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

4.1.1 Detailed Description

Représentation d'une texture du jeu.

4.1.2 Member Data Documentation

4.1.2.1 h

`int sprite_s::h`

Hauteur du sprite.

4.1.2.2 w

`int sprite_s::w`

Largeur du sprite.

4.1.2.3 x

```
int sprite_s::x
```

Position du sprite sur x.

4.1.2.4 y

```
int sprite_s::y
```

Position du sprite sur y.

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

- param.h

4.2 textures_s Struct Reference

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

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

Public Attributes

- SDL_Texture * background
- SDL_Texture * vaisseau
- SDL_Texture * arrival
- SDL_Texture * meteorite
- TTF_Font * font

4.2.1 Detailed Description

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

4.2.2 Member Data Documentation

4.2.2.1 arrival

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

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

4.2.2.2 background

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

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

4.2.2.3 font

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

Texture liée à la police utilisée.

4.2.2.4 meteorite

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

Texture liée à l'image d'un météorite.

4.2.2.5 vaisseau

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

Texture liée à l'image du vaisseau.

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

- param.h

4.3 world_s Struct Reference

Représentation du monde du jeu.

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

Collaboration diagram for world_s:

Public Attributes

- int gameover
- sprite_t vaisseau
- sprite_t arrival
- int vy
- sprite_t mur [MAX_METEORITE_WALL_NUMBER]
- int make_disappear

4.3.1 Detailed Description

Représentation du monde du jeu.

4.3.2 Member Data Documentation

4.3.2.1 arrival

```
sprite_t world_s::arrival
```

Information de la ligne d'arrivée.

4.3.2.2 gameover

```
int world_s::gameover
```

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

4.3.2.3 make_disappear

```
int world_s::make_disappear
```

Informe si le vaisseau doit être visible ou pas

4.3.2.4 mur

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

Informations sur un mur d'astéroides.

4.3.2.5 vaisseau

```
sprite_t world_s::vaisseau
```

Information du vaisseau.

4.3.2.6 vy

```
int world_s::vy
```

Vitesse de déplacement verticale.

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

- param.h

Chapter 5

File Documentation

5.1 game_event.c File Reference

Module de gestions des événements.

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

5.2 game_event.h File Reference

Entête du module de gestions des événements.

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

Functions

- int is_game_over (world_t *world)

  La fonction indique si le jeu est fini en fonction des données du monde.
- void update_data (world_t *world)

  La fonction met à jour les données en tenant compte de la physique du monde.
- void init_sprite (sprite_t *sprite, int x, int y, int w, int h)

  La fonction initialise les données d'un sprite selon les valeurs entrées.
- void init_sprite_meteore (sprite_t *sprite, int x, int y, int w, int h, int screen_nbr)

  La fonction initialise les données d'un sprite de meteore selon les valeurs entrées.
- void init_data (world_t *world)

  La fonction initialise les données du monde du jeu.
- void out_of_screen (world_t *world)

  Fonction qui detecte si le vaisseau est hors de l'écran.
- int sprites_collide (sprite_t *sp1, sprite_t *sp2)

  Fonction qui detecte si deux sprite sont en collision.
- void handle_sprites_collision (world_t *world, sprite_t *sp1, sprite_t *sp2, int *make_disappear)

  Fonction qui change la vitesse du monde en cas de collision.
- void init_walls (world_t *world)

  Fonction qui change initalise les murs.
- void update_walls (world_t *world)

  Fonction qui update la position de tout les murs.

5.2.1 Detailed Description

Entête du module de gestions des événements.

Author

Victor Dallé

Version

1.0

Date

1 avril 2021

5.2.2 Function Documentation

5.2.2.1 handle_sprites_collision()

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

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

Parameters

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

5.2.2.2 init_data()

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

La fonction initialise les données du monde du jeu.

Parameters

world	Les données du monde.

5.2.2.3 init_sprite()

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

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

Parameters

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

5.2.2.4 init_sprite_meteore()

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

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

Parameters

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

5.2.2.5 init_walls()

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

Fonction qui change initalise les murs.

Parameters

world	Structure des données du monde.

5.2.2.6 is_game_over()

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

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

Parameters

world	Les données du monde.

Returns

1 si le jeu est fini, 0 sinon.

5.2.2.7 out_of_screen()

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

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

Parameters

world	Structure des données du monde.

5.2.2.8 sprites_collide()

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

Fonction qui detecte si deux sprite sont en collision.

Parameters

sp1	premier sprite
sp2	second prite

Returns

1 si collision est 0 si aucun collision.

5.2.2.9 update_data()

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

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

Parameters

world	Les données du monde.

5.2.2.10 update_walls()

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

Fonction qui update la position de tout les murs.

Parameters

world	Structure des données du monde.

5.3 graphic.c File Reference

Module gérant l'affichage SDL2.

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

Functions

- void clean_textures (textures_t *textures)

La fonction nettoie les textures.

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

  La fonction initialise les textures nécessaires à l'affichage graphique du jeu.
- void apply_sprite (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite, int make_disappear)

  La fonction applique un sprite au renderer.
- void apply_background (SDL_Renderer ∗renderer, SDL_Texture ∗texture)

  La fonction applique la texture du fond sur le renderer lié à l'écran de jeu.
- void apply_wall (textures_t ∗textures, SDL_Renderer ∗renderer, world_t ∗world, int x, int y, int height, int width)

  La fonction applique la texture des meteorite sur le renderer.
- void refresh_graphics (SDL_Renderer ∗renderer, world_t ∗world, textures_t ∗textures)

  La fonction rafraichit l'écran en fonction de l'état des données du monde.
- void clean_data (world_t ∗world)

  La fonction nettoie les données du monde.
- void clean (SDL_Window ∗window, SDL_Renderer ∗renderer, textures_t ∗textures, world_t ∗world)

  La fonction nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des textures, nettoyage des données.
- void init (SDL_Window ∗∗window, SDL_Renderer ∗∗renderer, textures_t ∗textures, world_t ∗world)

  La fonction initialise le jeu: initialisation de la partie graphique (SDL), chargement des textures, initialisation des données.
- void apply_walls (SDL_Renderer ∗∗renderer, textures_t ∗textures, world_t ∗world)

  La fonction applique la texture des murs.

5.3.1 Detailed Description

Module gérant l'affichage SDL2.

Entête du module gérant l'affichage SDL2.

Author

Victor Dallé

Version

1.0

Date

1 avril 2021

5.3.2 Function Documentation

5.3.2.1 apply_background()

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

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

Parameters

renderer	Le renderer.
texture	La texture liée au fond.

5.3.2.2 apply_sprite()

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

La fonction applique un sprite au renderer.

Parameters

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

5.3.2.3 apply_wall()

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

La fonction applique la texture des meteorite sur le renderer.

Parameters

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

5.3.2.4 apply_walls()

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

La fonction applique la texture des murs.

Parameters

renderer	Le renderer.
textures	Les textures.
world	Le monde.

5.3.2.5 clean()

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

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

Parameters

window	La fenêtre du jeu.
renderer	Le renderer.
textures	Les textures.
world	Le monde.

5.3.2.6 clean_data()

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

La fonction nettoie les données du monde.

Parameters

world	Les données du monde.

5.3.2.7 clean_textures()

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

La fonction nettoie les textures.

Parameters

textures	Les textures.

5.3.2.8 init()

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

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

Parameters

window	La fenêtre du jeu.
renderer	Le renderer.
textures	Les textures.
world	Le monde.

5.3.2.9 init_textures()

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

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

Parameters

screen	La surface correspondant à l'écran de jeu.
textures	Les textures du jeu.

5.3.2.10 refresh_graphics()

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

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

Parameters

renderer	Le renderer lié à l'écran de jeu.
world	Les données du monde.
textures	Les textures.

5.4 handle_event.c File Reference

Fichier .c du module handle_event.

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

Functions

- void handle_events (SDL_Event ∗event, world_t ∗world)

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

5.4.1 Detailed Description

Fichier .c du module handle_event.

Author

Victor Dallé

Version

0.1

Date

2021-04-12

Copyright

5.4.2 Function Documentation

5.4.2.1 handle_events()

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

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

Parameters

event	Paramètre qui contient les événements.
world	Les données du monde.

5.5 handle_event.h File Reference

Fichier .h du module handle_event.

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

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

Functions

- void handle_events (SDL_Event ∗event, world_t ∗world)

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

5.5.1 Detailed Description

Fichier .h du module handle_event.

Author

Victor Dallé

Version

0.1

Date

2021-04-12

Copyright

5.5.2 Function Documentation

5.5.2.1 handle_events()

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

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

Parameters

event	Paramètre qui contient les événements.
world	Les données du monde.

5.6 main.c File Reference

Programme principal initial du niveau 1.

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

Functions

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

5.6.1 Detailed Description

Programme principal initial du niveau 1.

Entête du main.

Author

Mathieu Constant

Version

1.0

Date

18 mars 2021

Author

Victor Dallé

Version

1.0

Date

1 avril 2021

5.6.2 Function Documentation

5.6.2.1 main()

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

Programme principal qui implémente la boucle du jeu.

Parameters

argc	Taille du tableau argv.
argv	Pointeur vers un tableau de char de taille argc.

Returns

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

5.7 param.h File Reference

Fichier qui contient les different constante librerie et structure.

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

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

Classes

- struct textures_s

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

  Représentation d'une texture du jeu.
- struct world_s

  Représentation du monde du jeu.

Macros

- #define SCREEN_WIDTH 300

  Largeur de l'écran de jeu.
- #define SCREEN_HEIGHT 480

  Hauteur de l'écran de jeu.
- #define SHIP_SIZE 32

  Taille d'un vaisseau.
- #define METEORITE_SIZE 32

  Taille d'un météorite.
- #define FINISH_LINE_HEIGHT 10

  Hauteur de la ligne d'arrivée.
- #define MOVING_STEP 10

  Pas de déplacement horizontal du vaisseau.
- #define INITIAL_SPEED 2

  Vitesse initiale de déplacement vertical des éléments du jeu.
- #define MAX_METEORITE_WALL_NUMBER 20

  Nombre de mur de météorite.
- #define METEORITE_WALL_NUMBER 16

  Nombre de mur de météorite.

Typedefs

- typedef struct textures_s textures_t

  Type qui correspond aux textures du jeu.
- typedef struct sprite_s sprite_t

  Type qui correspond à une texture.
- typedef struct world_s world_t

  Type qui correspond aux données du monde.

5.7.1 Detailed Description

Fichier qui contient les different constante librerie et structure.

Author

Periney Yann

Version

1

Date

1 avril 2021

5.8 sdl2-light.c File Reference

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

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

Include dependency graph for sdl2-light.c:

Functions

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

  La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.
- SDL_Texture * load_image (const char path[], SDL_Renderer *renderer)

  La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.
- void apply_texture (SDL_Texture *texture, SDL_Renderer *renderer, int x, int y)

  La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.
- void clean_texture (SDL_Texture *texture)

  La fonction nettoie une texture en mémoire.
- void clear_renderer (SDL_Renderer *renderer)

  La fonction vide le contenu graphique du renderer lié à l'écran de jeu.
- void update_screen (SDL_Renderer *renderer)

  La fonction met à jour l'écran avec le contenu du renderer.
- void pause (int time)

  La fonction met le programme en pause pendant un laps de temps.
- void clean_sdl (SDL_Renderer *renderer, SDL_Window *window)

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

5.8.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

5.8.2 Function Documentation

5.8.2.1 apply_texture()

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

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

Parameters

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)

5.8.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

5.8.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

5.8.2.4 clear_renderer()

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

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

Parameters

renderer	le renderer de l'écran

5.8.2.5 init_sdl()

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

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

Parameters

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

5.8.2.6 load_image()

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

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

Parameters

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)

5.8.2.7 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

5.8.2.8 update_screen()

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

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

Parameters

renderer	le renderer de l'écran

5.9 sdl2-light.h File Reference

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

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

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

Functions

- void clean_sdl (SDL_Renderer ∗renderer, SDL_Window ∗window)

  La fonction nettoie le renderer et la fenêtre du jeu en mémoire.
- SDL_Texture ∗ load_image (const char path[ ], SDL_Renderer ∗renderer)

  La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.
- int init_sdl (SDL_Window ∗∗window, SDL_Renderer ∗∗renderer, int width, int height)

  La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.
- void clean_texture (SDL_Texture ∗texture)

  La fonction nettoie une texture en mémoire.
- void apply_texture (SDL_Texture ∗texture, SDL_Renderer ∗renderer, int x, int y)

  La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.
- void clear_renderer (SDL_Renderer ∗renderer)

  La fonction vide le contenu graphique du renderer lié à l'écran de jeu.
- void update_screen (SDL_Renderer ∗renderer)

  La fonction met à jour l'écran avec le contenu du renderer.
- void pause (int time)

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

5.9.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

5.9.2 Function Documentation

5.9.2.1 apply_texture()

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

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

Parameters

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)

5.9.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

5.9.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

5.9.2.4 clear_renderer()

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

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

Parameters

renderer	le renderer de l'écran

5.9.2.5 init_sdl()

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

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

Parameters

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

5.9.2.6 load_image()

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

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

Parameters

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)

5.9.2.7 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

5.9.2.8 update_screen()

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

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

Parameters

renderer	le renderer de l'écran

5.10 tests.c File Reference

Fichier de test pour les fonction de game_event.

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

Include dependency graph for tests.c:

Functions

- void print_sprite (sprite_t ∗sprite)

  Fonction qui affiche les coordonnées d'un sprite.
- void test_init_sprite_param (sprite_t sprite, int x, int y, int w, int h)

  Fonction de test_param pour init_sprite.
- void **test_init_sprite** ()
- void test_out_of_screen_param (world_t ∗world)

  Fonction de test pour la sortie de l'écran du vaisseau.
- void **test_out_of_screen** ()
- void test_sprites_collide_param (sprite_t sprite1, sprite_t sprite2)

  Fonction de test pour la collision de 2 sprites.
- void **test_sprites_collide** ()
- void test_handle_sprites_collision_param (world_t world, sprite_t spr1, sprite_t spr2, int disp)

  Fonction de test effectuant des modifs sur data world lors d'une collision.
- void **test_handle_sprites_collision** ()
- void **test_init_walls_param** (world_t ∗world)
- void **test_init_walls** ()
- void **test_update_walls_param** (world_t ∗world)
- void **test_update_walls** ()
- int main (int argc, char ∗argv[ ])

  Main pour le programme de test pour le module game_event.h.

5.10.1 Detailed Description

Fichier de test pour les fonction de game_event.

Author

Periney Yann Victor Dallé

Version

1

Date

1 avril 2021

5.10.2 Function Documentation

5.10.2.1 main()

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

Main pour le programme de test pour le module game_event.h.

Parameters

argc	Taille du tableau argv.
argv	Pointeur vers un tableau de char de taille argc.

Returns

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

5.10.2.2 print_sprite()

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

Fonction qui affiche les coordonnées d'un sprite.

Parameters

sprite	Sprite cible du test.

5.10.2.3 test_handle_sprites_collision_param()

```
void test_handle_sprites_collision_param (
            world_t world,
            sprite_t spr1,
            sprite_t spr2,
            int disp )
```

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

Parameters

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

5.10.2.4 test_init_sprite_param()

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

Fonction de test_param pour init_sprite.

Parameters

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

5.10.2.5 test_out_of_screen_param()

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

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

Parameters

world	Données du monde.

5.10.2.6 test_sprites_collide_param()

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

Fonction de test pour la collision de 2 sprites.

Parameters

sprite1	Premier sprite.
sprite2	Deuxième sprite.

Index