My Project

Generated by Doxygen 1.8.13

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 ressources_s Struct Reference 5

3.1.1 Member Data Documentation 5

3.1.1.1 background 5

3.1.1.2 boss 5

3.1.1.3 enemy_blue 6

3.1.1.4 enemy_green 6

3.1.1.5 enemy_orange 6

3.1.1.6 enemy_purple 6

3.1.1.7 enemy_red 6

3.1.1.8 font 6

3.1.1.9 missile 6

3.1.1.10 missile_boss 6

3.1.1.11 missile_orange 7

3.1.1.12 missile_purple 7

3.1.1.13 ship 7

3.2 sprite_s Struct Reference 7

3.2.1 Member Data Documentation 7

3.2.1.1 h . . . 7
3.2.1.2 hp . . . 8
3.2.1.3 is_visible . . . 8
3.2.1.4 v . . . 8
3.2.1.5 w . . . 8
3.2.1.6 x . . . 8
3.2.1.7 y . . . 8
3.3 sprite_t Struct Reference . . . 8
3.3.1 Detailed Description . . . 9
3.4 texture_t Struct Reference . . . 9
3.4.1 Detailed Description . . . 9
3.5 world_s Struct Reference . . . 9
3.5.1 Member Data Documentation . . . 10
3.5.1.1 blue . . . 10
3.5.1.2 boss . . . 10
3.5.1.3 boss_speed_x . . . 10
3.5.1.4 boss_speed_y . . . 10
3.5.1.5 compteur . . . 10
3.5.1.6 etat_partie . . . 10
3.5.1.7 gameover . . . 10
3.5.1.8 green . . . 11
3.5.1.9 missile . . . 11
3.5.1.10 missile_boss . . . 11
3.5.1.11 missile_orange . . . 11
3.5.1.12 missile_purple . . . 11
3.5.1.13 orange . . . 11
3.5.1.14 purple . . . 11
3.5.1.15 red . . . 11
3.5.1.16 score . . . 12
3.5.1.17 ship . . . 12
3.5.1.18 timer . . . 12
3.5.1.19 vague . . . 12
3.6 world_t Struct Reference . . . 12
3.6.1 Detailed Description . . . 12

4 File Documentation 13

4.1 biblio.h File Reference . . . 13

4.2 event.c File Reference . . . 13

4.2.1 Detailed Description . . . 13

4.2.2 Function Documentation . . . 14

4.2.2.1 handle_events() . . . 14

4.3 event.h File Reference . . . 14

4.3.1 Detailed Description . . . 14

4.3.2 Function Documentation . . . 14

4.3.2.1 handle_events() . . . 14

4.4 main.c File Reference . . . 15

4.4.1 Detailed Description . . . 15

4.5 renderer.c File Reference . . . 15

4.5.1 Detailed Description . . . 16

4.5.2 Function Documentation . . . 16

4.5.2.1 apply_sprite() . . . 16

4.5.2.2 apply_sprite_boss() . . . 17

4.5.2.3 apply_sprite_missile() . . . 17

4.5.2.4 clean() . . . 17

4.5.2.5 init() . . . 18

4.5.2.6 refresh() . . . 18

4.5.2.7 refresh_graphics() . . . 18

4.6 renderer.h File Reference . . . 19

4.6.1 Detailed Description . . . 19

4.6.2 Function Documentation . . . 19

4.6.2.1 apply_sprite() . . . 20

4.6.2.2 apply_sprite_boss() . . . 20

4.6.2.3 apply_sprite_missile() . . . 20

4.6.2.4 clean() . . . 21

4.6.2.5 init() . . . 21

4.6.2.6 refresh() . . . 21
4.6.2.7 refresh_graphics() . . . 22
4.7 ressources.c File Reference . . . 22
4.7.1 Detailed Description . . . 22
4.7.2 Function Documentation . . . 23
4.7.2.1 apply_background() . . . 23
4.7.2.2 clean_ressources() . . . 23
4.7.2.3 init_ressources() . . . 23
4.8 ressources.h File Reference . . . 24
4.8.1 Detailed Description . . . 24
4.8.2 Function Documentation . . . 24
4.8.2.1 apply_background() . . . 24
4.8.2.2 clean_ressources() . . . 25
4.8.2.3 init_ressources() . . . 25
4.9 sdl2-light.c File Reference . . . 25
4.9.1 Detailed Description . . . 26
4.9.2 Function Documentation . . . 26
4.9.2.1 apply_texture() . . . 26
4.9.2.2 clean_sdl() . . . 27
4.9.2.3 clean_texture() . . . 27
4.9.2.4 clear_renderer() . . . 27
4.9.2.5 init_sdl() . . . 27
4.9.2.6 load_image() . . . 28
4.9.2.7 pause() . . . 28
4.9.2.8 update_screen() . . . 29
4.10 sprite.c File Reference . . . 29
4.10.1 Detailed Description . . . 29
4.10.2 Function Documentation . . . 29
4.10.2.1 init_sprite() . . . 30
4.10.2.2 print_sprite() . . . 30

4.10.2.3 set_invisible() . . . 30
4.10.2.4 set_visible() . . . 31
4.11 sprite.h File Reference . . . 31
4.11.1 Detailed Description . . . 31
4.11.2 Function Documentation . . . 32
4.11.2.1 init_sprite() . . . 32
4.11.2.2 print_sprite() . . . 32
4.11.2.3 set_invisible() . . . 32
4.11.2.4 set_visible() . . . 33
4.12 tests.c File Reference . . . 33
4.12.1 Detailed Description . . . 34
4.13 world.c File Reference . . . 34
4.13.1 Detailed Description . . . 35
4.13.2 Function Documentation . . . 35
4.13.2.1 clean_data_1() . . . 35
4.13.2.2 clean_data_2() . . . 36
4.13.2.3 compute_game() . . . 36
4.13.2.4 depassement_enemy_horizontal() . . . 36
4.13.2.5 depassement_enemy_vertical() . . . 37
4.13.2.6 depassement_missile() . . . 37
4.13.2.7 depassement_missile_boss() . . . 37
4.13.2.8 depassement_missile_enemy() . . . 37
4.13.2.9 generate_number() . . . 38
4.13.2.10 handle_sprite_collide() . . . 38
4.13.2.11 init_blue() . . . 39
4.13.2.12 init_data_1() . . . 39
4.13.2.13 init_data_2() . . . 39
4.13.2.14 init_green() . . . 39
4.13.2.15 init_missile() . . . 40
4.13.2.16 init_missile_boss() . . . 40

4.13.2.17 init_missile_enemy() . . . 40
4.13.2.18 init_orange() . . . 41
4.13.2.19 init_purple() . . . 41
4.13.2.20 init_red() . . . 41
4.13.2.21 is_game_over() . . . 41
4.13.2.22 max() . . . 42
4.13.2.23 sprite_collide() . . . 42
4.13.2.24 update_boss() . . . 43
4.13.2.25 update_data() . . . 43
4.13.2.26 update_enemy_horizontal() . . . 43
4.13.2.27 update_enemy_vertical() . . . 43
4.13.2.28 update_missile() . . . 44
4.13.2.29 update_missile_boss() . . . 44
4.13.2.30 update_missile_enemy() . . . 44
4.14 world.h File Reference . . . 45
4.14.1 Detailed Description . . . 46
4.14.2 Function Documentation . . . 47
4.14.2.1 clean_data_1() . . . 47
4.14.2.2 clean_data_2() . . . 47
4.14.2.3 compute_game() . . . 47
4.14.2.4 depassement_enemy_horizontal() . . . 47
4.14.2.5 depassement_enemy_vertical() . . . 48
4.14.2.6 depassement_missile() . . . 48
4.14.2.7 depassement_missile_boss() . . . 48
4.14.2.8 depassement_missile_enemy() . . . 49
4.14.2.9 depassement_orange() . . . 49
4.14.2.10 generate_number() . . . 49
4.14.2.11 handle_sprite_collide() . . . 50
4.14.2.12 init_blue() . . . 50
4.14.2.13 init_data_1() . . . 50

4.14.2.14 init_data_2() . . . 51
4.14.2.15 init_green() . . . 51
4.14.2.16 init_missile() . . . 51
4.14.2.17 init_missile_boss() . . . 51
4.14.2.18 init_missile_enemy() . . . 52
4.14.2.19 init_orange() . . . 52
4.14.2.20 init_purple() . . . 52
4.14.2.21 init_red() . . . 53
4.14.2.22 is_game_over() . . . 53
4.14.2.23 max() . . . 53
4.14.2.24 sprite_collide() . . . 54
4.14.2.25 update_boss() . . . 54
4.14.2.26 update_data() . . . 54
4.14.2.27 update_enemy_horizontal() . . . 55
4.14.2.28 update_enemy_vertical() . . . 55
4.14.2.29 update_missile() . . . 55
4.14.2.30 update_missile_boss() . . . 56
4.14.2.31 update_missile_enemy() . . . 56

Index **57**

Chapter 1

Class Index

1.1 Class List

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

ressources_s 5
sprite_s 7
sprite_t
    Représentation d'un sprite 8
texture_t
    Représentation pour stocker les ressources nécessaires à l'affichage graphique 9
world_s 9
world_t
    Représentation du monde du jeu 12

Chapter 2

File Index

2.1 File List

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

biblio.h
Le module des include généraux et des define . . . 13
event.c
Fonctions des interactions du joueur avec le jeu . . . 13
event.h
Bibliothèque de event . . . 14
main.c
Programme principal du jeu . . . 15
renderer.c
Fonctions de l'affichage . . . 15
renderer.h
Bibliothèque de renderer . . . 19
ressources.c
Gestion des ressources . . . 22
ressources.h
Bibliothèque de ressources . . . 24
sdl2-light.c
Sur-couche de SDL2 pour simplifier son utilisation pour le projet . . . 25
sdl2-light.h . . . **??**
sdl2-ttf-light.h . . . **??**
sprite.c
Gestion des sprites . . . 29
sprite.h
Bibliothèque de sprite . . . 31
tests.c
Programme de tests . . . 33
world.c
Gestion du monde . . . 34
world.h
Bibliothèque de world . . . 45

Chapter 3

Class Documentation

3.1 ressources_s Struct Reference

Public Attributes

- SDL_Texture * background
- SDL_Texture * ship
- SDL_Texture * boss
- SDL_Texture * enemy_blue
- SDL_Texture * enemy_red
- SDL_Texture * enemy_green
- SDL_Texture * enemy_purple
- SDL_Texture * enemy_orange
- SDL_Texture * missile
- SDL_Texture * missile_purple
- SDL_Texture * missile_orange
- SDL_Texture * missile_boss
- TTF_Font * font

3.1.1 Member Data Documentation

3.1.1.1 background

```
SDL_Texture* ressources_s::background
```

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

3.1.1.2 boss

```
SDL_Texture* ressources_s::boss
```

Texture liée à l'image du boss.

3.1.1.3 enemy_blue

```
SDL_Texture* ressources_s::enemy_blue
```

Texture liée à l'image de l'ennemi blue.

3.1.1.4 enemy_green

```
SDL_Texture* ressources_s::enemy_green
```

Texture liée à l'image de l'ennemi green.

3.1.1.5 enemy_orange

```
SDL_Texture* ressources_s::enemy_orange
```

Texture liée à l'image de l'ennemi orange.

3.1.1.6 enemy_purple

```
SDL_Texture* ressources_s::enemy_purple
```

Texture liée à l'image de l'ennemi purple.

3.1.1.7 enemy_red

```
SDL_Texture* ressources_s::enemy_red
```

Texture liée à l'image de l'ennemi red.

3.1.1.8 font

```
TTF_Font* ressources_s::font
```

Texture de la police.

3.1.1.9 missile

```
SDL_Texture* ressources_s::missile
```

Texture liée à l'image d'un missile du joueur.

3.1.1.10 missile_boss

```
SDL_Texture* ressources_s::missile_boss
```

Texture liée à l'image d'un missile du boss.

3.1.1.11 missile_orange

```
SDL_Texture* ressources_s::missile_orange
```

Texture liée à l'image d'un missile ennemi orange.

3.1.1.12 missile_purple

```
SDL_Texture* ressources_s::missile_purple
```

Texture liée à l'image d'un missile ennemi purple.

3.1.1.13 ship

```
SDL_Texture* ressources_s::ship
```

Texture liée à l'image du vaisseau.

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

- ressources.h

3.2 sprite_s Struct Reference

Public Attributes

- int x
- int y
- int h
- int w
- int v
- int is_visible
- int hp

3.2.1 Member Data Documentation

3.2.1.1 h

```
int sprite_s::h
```

Hauteur du sprite.

3.2.1.2 hp

```
int sprite_s::hp
```

Points de vie du sprite.

3.2.1.3 is_visible

```
int sprite_s::is_visible
```

Visibilité du sprite.

3.2.1.4 v

```
int sprite_s::v
```

Vitesse de déplacement verticale.

3.2.1.5 w

```
int sprite_s::w
```

Largeur du sprite.

3.2.1.6 x

```
int sprite_s::x
```

Abscisses par rapport au centre de l'image.

3.2.1.7 y

```
int sprite_s::y
```

Ordionnée par rapport au centre de l'image.

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

- sprite.h

3.3 sprite_t Struct Reference

Représentation d'un sprite.

```
#include <sprite.h>
```

3.3.1 Detailed Description

Représentation d'un sprite.

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

- sprite.h

3.4 texture_t Struct Reference

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

`#include <ressources.h>`

3.4.1 Detailed Description

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

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

- ressources.h

3.5 world_s Struct Reference

Collaboration diagram for world_s:

Public Attributes

- sprite_t * ship
- sprite_t * boss
- sprite_t * blue [TOTAL_BLUE]
- sprite_t * red [TOTAL_RED]
- sprite_t * green [TOTAL_GREEN]
- sprite_t * purple [TOTAL_PURPLE]
- sprite_t * orange [TOTAL_ORANGE]
- sprite_t * missile [NB_MISSILES]
- sprite_t * missile_purple [TOTAL_PURPLE]
- sprite_t * missile_orange [TOTAL_ORANGE]
- sprite_t * missile_boss [BOSS_MISSILES]
- int gameover
- unsigned int compteur
- int etat_partie
- int score
- int timer
- int vague
- int boss_speed_x
- int boss_speed_y

3.5.1 Member Data Documentation

3.5.1.1 blue

```
sprite_t* world_s::blue[TOTAL_BLUE]
```

Tableau d'ennemis blue.

3.5.1.2 boss

```
sprite_t* world_s::boss
```

Sprite du boss.

3.5.1.3 boss_speed_x

```
int world_s::boss_speed_x
```

La vitesse horizontale du boss.

3.5.1.4 boss_speed_y

```
int world_s::boss_speed_y
```

La vitesse verticale du boss.

3.5.1.5 compteur

```
unsigned int world_s::compteur
```

Compteur de vaisseaux sortis.

3.5.1.6 etat_partie

```
int world_s::etat_partie
```

État de la partie.

3.5.1.7 gameover

```
int world_s::gameover
```

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

3.5.1.8 green

```
sprite_t* world_s::green[TOTAL_GREEN]
```

Tableau d'ennemis green.

3.5.1.9 missile

```
sprite_t* world_s::missile[NB_MISSILES]
```

Sprites des missiles.

3.5.1.10 missile_boss

```
sprite_t* world_s::missile_boss[BOSS_MISSILES]
```

Sprites des missiles du boss.

3.5.1.11 missile_orange

```
sprite_t* world_s::missile_orange[TOTAL_ORANGE]
```

Sprites des missiles ennemis orange.

3.5.1.12 missile_purple

```
sprite_t* world_s::missile_purple[TOTAL_PURPLE]
```

Sprites des missiles ennemis purple.

3.5.1.13 orange

```
sprite_t* world_s::orange[TOTAL_ORANGE]
```

Tableau d'ennemis orange.

3.5.1.14 purple

```
sprite_t* world_s::purple[TOTAL_PURPLE]
```

Tableau d'ennemis purple.

3.5.1.15 red

```
sprite_t* world_s::red[TOTAL_RED]
```

Tableau d'ennemis red.

3.5.1.16 score

```
int world_s::score
```

Score.

3.5.1.17 ship

```
sprite_t* world_s::ship
```

Sprite du vaisseau.

3.5.1.18 timer

```
int world_s::timer
```

Le timer.

3.5.1.19 vague

```
int world_s::vague
```

La vague.

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

- world.h

3.6 world_t Struct Reference

Représentation du monde du jeu.

```
#include <world.h>
```

3.6.1 Detailed Description

Représentation du monde du jeu.

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

- world.h

Chapter 4

File Documentation

4.1 biblio.h File Reference

Le module des include généraux et des define.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include "sdl2-light.h"
#include "sdl2-ttf-light.h"
#include "math.h"
```

Include dependency graph for biblio.h:

4.2 event.c File Reference

Fonctions des interactions du joueur avec le jeu.

```
#include "event.h"
```

Include dependency graph for event.c:

Functions

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

  La fonction gère les interactions de l'utilisateur.

4.2.1 Detailed Description

Fonctions des interactions du joueur avec le jeu.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.2.2 Function Documentation

4.2.2.1 handle_events()

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

La fonction gère les interactions de l'utilisateur.

Parameters

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

4.3 event.h File Reference

Bibliothèque de event.

```
#include "world.h"
#include "biblio.h"
```

Include dependency graph for 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 interactions de l'utilisateur.

4.3.1 Detailed Description

Bibliothèque de event.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.3.2 Function Documentation

4.3.2.1 handle_events()

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

La fonction gère les interactions de l'utilisateur.

Parameters

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

4.4 main.c File Reference

Programme principal du jeu.

```
#include "biblio.h"
#include "event.h"
#include "sprite.h"
#include "ressources.h"
#include "world.h"
#include "renderer.h"
```

Include dependency graph for main.c:

Functions

- int main (int argc, char ∗args[ ])

  Programme principal qui implémente la boucle du jeu.

4.4.1 Detailed Description

Programme principal du jeu.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.5 renderer.c File Reference

Fonctions de l'affichage.

```
#include "renderer.h"
```

Include dependency graph for renderer.c:

Functions

- void refresh_graphics (SDL_Renderer ∗renderer, world_t ∗world, ressources_t ∗ressources)

  La fonction rafraîchit l'écran en fonction de l'état des données du monde.
- void refresh (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗tab[ ], int total)

  La fonction rafraîchit l'écran pour les missiles.
- void clean (SDL_Window ∗window, SDL_Renderer ∗renderer, ressources_t ∗ressources, world_t ∗world)

  La fonction nettoie la partie graphique (SDL), nettoie des ressources et des données.
- void init (SDL_Window ∗∗window, SDL_Renderer ∗∗renderer, ressources_t ∗ressources, world_t ∗world)

  La fonction initialise la partie graphique (SDL), charge des ressources et initialise des données.
- void apply_sprite (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

  La fonction applique la texture du vaisseau sur le renderer lié à l'écran de jeu.
- void apply_sprite_boss (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

  La fonction applique la texture du boss sur le renderer lié à l'écran de jeu.
- void apply_sprite_missile (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

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

4.5.1 Detailed Description

Fonctions de l'affichage.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.5.2 Function Documentation

4.5.2.1 apply_sprite()

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

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.
sprite	Le sprite.

4.5.2.2 apply_sprite_boss()

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

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.
sprite	Le sprite.

4.5.2.3 apply_sprite_missile()

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

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.
sprite	Le sprite.

4.5.2.4 clean()

```
void clean (
            SDL_Window * window,
            SDL_Renderer * renderer,
            ressources_t * ressources,
            world_t * world )
```

La fonction nettoie la partie graphique (SDL), nettoie des ressources et des données.

Parameters

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

4.5.2.5 init()

```
void init (
            SDL_Window ** window,
            SDL_Renderer ** renderer,
            ressources_t * ressources,
            world_t * world )
```

La fonction initialise la partie graphique (SDL), charge des ressources et initialise des données.

Parameters

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

4.5.2.6 refresh()

```
void refresh (
            SDL_Renderer * renderer,
            SDL_Texture * texture,
            sprite_t * tab[],
            int total )
```

La fonction rafraîchit l'écran pour les missiles.

Parameters

renderer	La surface de l'écran de jeu.
texture	La texture.
tab	Le tableau.
total	Le total.

4.5.2.7 refresh_graphics()

```
void refresh_graphics (
            SDL_Renderer * renderer,
            world_t * world,
            ressources_t * ressources )
```

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

Parameters

renderer	La surface de l'écran de jeu.
world	Les données du monde.
ressources	Les ressources.

4.6 renderer.h File Reference

Bibliothèque de renderer.

```
#include "biblio.h"
#include "world.h"
#include "ressources.h"
```

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

Functions

- void refresh_graphics (SDL_Renderer ∗renderer, world_t ∗world, ressources_t ∗ressources)

  La fonction rafraîchit l'écran en fonction de l'état des données du monde.
- void refresh (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗tab[ ], int total)

  La fonction rafraîchit l'écran pour les missiles.
- void clean (SDL_Window ∗window, SDL_Renderer ∗renderer, ressources_t ∗ressources, world_t ∗world)

  La fonction nettoie la partie graphique (SDL), nettoie des ressources et des données.
- void init (SDL_Window ∗∗window, SDL_Renderer ∗∗renderer, ressources_t ∗ressources, world_t ∗world)

  La fonction initialise la partie graphique (SDL), charge des ressources et initialise des données.
- void apply_sprite (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

  La fonction applique la texture du vaisseau sur le renderer lié à l'écran de jeu.
- void apply_sprite_boss (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

  La fonction applique la texture du boss sur le renderer lié à l'écran de jeu.
- void apply_sprite_missile (SDL_Renderer ∗renderer, SDL_Texture ∗texture, sprite_t ∗sprite)

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

4.6.1 Detailed Description

Bibliothèque de renderer.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.6.2 Function Documentation

4.6.2.1 apply_sprite()

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

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.
sprite	Le sprite.

4.6.2.2 apply_sprite_boss()

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

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.
sprite	Le sprite.

4.6.2.3 apply_sprite_missile()

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

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.
sprite	Le sprite.

4.6.2.4 clean()

```
void clean (
            SDL_Window * window,
            SDL_Renderer * renderer,
            ressources_t * ressources,
            world_t * world )
```

La fonction nettoie la partie graphique (SDL), nettoie des ressources et des données.

Parameters

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

4.6.2.5 init()

```
void init (
            SDL_Window ** window,
            SDL_Renderer ** renderer,
            ressources_t * ressources,
            world_t * world )
```

La fonction initialise la partie graphique (SDL), charge des ressources et initialise des données.

Parameters

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

4.6.2.6 refresh()

```
void refresh (
            SDL_Renderer * renderer,
            SDL_Texture * texture,
            sprite_t * tab[],
            int total )
```

La fonction rafraîchit l'écran pour les missiles.

Parameters

renderer	La surface de l'écran de jeu.
texture	La texture.
tab	Le tableau.
total	Le total.

4.6.2.7 refresh_graphics()

```
void refresh_graphics (
            SDL_Renderer * renderer,
            world_t * world,
            ressources_t * ressources )
```

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

Parameters

renderer	La surface de l'écran de jeu.
world	Les données du monde.
ressources	Les ressources.

4.7 ressources.c File Reference

Gestion des ressources.

```
#include "ressources.h"
```

Include dependency graph for ressources.c:

Functions

- void clean_ressources (ressources_t *ressources)
  La fonction nettoie les ressources.
- void init_ressources (SDL_Renderer *renderer, ressources_t *ressources)
  La fonction initialise les textures.
- void apply_background (SDL_Renderer *renderer, ressources_t *ressources)
  La fonction applique la texture du fond sur le renderer lié à l'écran de jeu.

4.7.1 Detailed Description

Gestion des ressources.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.7.2 Function Documentation

4.7.2.1 apply_background()

```
void apply_background (
            SDL_Renderer * renderer,
            ressources_t * ressources )
```

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.

4.7.2.2 clean_ressources()

```
void clean_ressources (
            ressources_t * ressources )
```

La fonction nettoie les ressources.

Parameters

ressources	Les ressources.

4.7.2.3 init_ressources()

```
void init_ressources (
            SDL_Renderer * renderer,
            ressources_t * ressources )
```

La fonction initialise les textures.

Parameters

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

4.8 ressources.h File Reference

Bibliothèque de ressources.

```
#include "biblio.h"
```

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

Classes

- struct ressources_s

Typedefs

- typedef struct ressources_s **ressources_t**

Functions

- void clean_ressources (ressources_t *ressources)

  La fonction nettoie les ressources.
- void init_ressources (SDL_Renderer *renderer, ressources_t *ressources)

  La fonction initialise les textures.
- void apply_background (SDL_Renderer *renderer, ressources_t *ressources)

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

4.8.1 Detailed Description

Bibliothèque de ressources.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.8.2 Function Documentation

4.8.2.1 apply_background()

```
void apply_background (
            SDL_Renderer * renderer,
            ressources_t * ressources )
```

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

Parameters

renderer	Le renderer.
ressources	Les ressources du jeu.

4.8.2.2 clean_ressources()

```
void clean_ressources (
            ressources_t * ressources )
```

La fonction nettoie les ressources.

Parameters

ressources	Les ressources.

4.8.2.3 init_ressources()

```
void init_ressources (
            SDL_Renderer * renderer,
            ressources_t * ressources )
```

La fonction initialise les textures.

Parameters

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

4.9 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.

- 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.

4.9.1 Detailed Description

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

Author

Mathieu Constant

Version

0.1

Date

10 mars 2020

4.9.2 Function Documentation

4.9.2.1 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)

4.9.2.2 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

4.9.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

4.9.2.4 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

4.9.2.5 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

4.9.2.6 load_image()

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

La fonction charge une image et renvoie la texture correspondante.

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. Elle renvoie NULL si le chargement a échoué (ex. le fichier path n'existe pas)

4.9.2.7 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

4.9.2.8 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

4.10 sprite.c File Reference

Gestion des sprites.

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

Functions

- void init_sprite (sprite_t *sprite, int x, int y, int w, int h, int v, int visible, int hp)
  La fonction initialise le sprite.
- void print_sprite (sprite_t *sprite)
  La fonction affiche le sprite.
- void set_visible (sprite_t *sprite)
  La fonction définit le sprite comme visible.
- void set_invisible (sprite_t *sprite)
  La fonction définit le sprite comme invisible.

4.10.1 Detailed Description

Gestion des sprites.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.10.2 Function Documentation

4.10.2.1 init_sprite()

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

La fonction initialise le sprite.

Parameters

x	La position en abcisse.
y	La position en ordonnée.
w	La largeur.
h	La hauteur.
v	La vitesse verticale.
visible	La visibilité.
hp	Les points de vie.

4.10.2.2 print_sprite()

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

La fonction affiche le sprite.

Parameters

sprite	Le sprite.

4.10.2.3 set_invisible()

```
void set_invisible (
            sprite_t * sprite )
```

La fonction définit le sprite comme invisible.

Parameters

sprite	Le sprite.

4.10.2.4 set_visible()

```
void set_visible (
            sprite_t * sprite )
```

La fonction définit le sprite comme visible.

Parameters

sprite	Le sprite.

4.11 sprite.h File Reference

Bibliothèque de sprite.

```
#include "biblio.h"
```

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

Classes

- struct sprite_s

Typedefs

- typedef struct sprite_s **sprite_t**

Functions

- void init_sprite (sprite_t ∗sprite, int x, int y, int w, int h, int v, int visible, int hp)
  La fonction initialise le sprite.
- void print_sprite (sprite_t ∗sprite)
  La fonction affiche le sprite.
- void set_invisible (sprite_t ∗sprite)
  La fonction définit le sprite comme invisible.
- void set_visible (sprite_t ∗sprite)
  La fonction définit le sprite comme visible.

4.11.1 Detailed Description

Bibliothèque de sprite.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.11.2 Function Documentation

4.11.2.1 init_sprite()

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

La fonction initialise le sprite.

Parameters

x	La position en abcisse.
y	La position en ordonnée.
w	La largeur.
h	La hauteur.
v	La vitesse verticale.
visible	La visibilité.
hp	Les points de vie.

4.11.2.2 print_sprite()

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

La fonction affiche le sprite.

Parameters

sprite	Le sprite.

4.11.2.3 set_invisible()

```
void set_invisible (
            sprite_t * sprite )
```

La fonction définit le sprite comme invisible.

Parameters

sprite	Le sprite.

4.11.2.4 set_visible()

```
void set_visible (
            sprite_t * sprite )
```

La fonction définit le sprite comme visible.

Parameters

sprite	Le sprite.

4.12 tests.c File Reference

Programme de tests.

```
#include "biblio.h"
#include "event.h"
#include "sprite.h"
#include "ressources.h"
#include "world.h"
#include "renderer.h"
```

Include dependency graph for tests.c:

Functions

- void **test_init_sprite_param** (sprite_t *sprite, int x, int y, int w, int h, int v, int visible)
- void **test_init_sprite** ()
- void **test_visible_param** (sprite_t *sprite)
- void **test_visible** ()
- void **test_init_blue_param** (sprite_t *blue[ ])
- void **test_init_blue** ()
- void **test_init_missile_param** (world_t *world)
- void **test_init_missile** ()
- void **test_depassement_blue_param** (world_t *world)
- void **test_depassement_blue** ()
- void **test_depassement_missile_param** (sprite_t *missile[ ])
- void **test_depassement_missile** ()
- void **test_handle_sprite_collide_param** (sprite_t *sp1, sprite_t *sp2, world_t *world)
- void **test_handle_sprite_collide** ()
- int **main** (int argc, char *args[ ])

4.12.1 Detailed Description

Programme de tests.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.13 world.c File Reference

Gestion du monde.

```
#include "world.h"
```

Include dependency graph for world.c:

Functions

- void init_data_1 (world_t *world)

  La fonction initialise les données du monde du jeu.
- void init_data_2 (sprite_t *tab[ ], int total)

  La fonction initialise les données des tableaux du monde du jeu.
- void clean_data_1 (world_t *world)

  La fonction nettoie les données du monde.
- void clean_data_2 (sprite_t *tab[ ], int total)

  La fonction nettoie les données du monde.
- int is_game_over (world_t *world)

  La fonction indique si le jeu est fini en fonction des données du monde.
- int generate_number (int a, int b)

  Generation d'un nombre entier compris entre a et b (ce dernier non inclus).
- int max (int a, int b)

  Maximum entre a et b.
- void init_blue (sprite_t *blue[ ])

  La fonction initialise les données des ennemis blue.
- void init_red (sprite_t *red[ ])

  La fonction initialise les données des ennemis red.
- void init_green (sprite_t *green[ ])

  La fonction initialise les données des ennemis green.
- void init_purple (sprite_t *purple[ ])

  La fonction initialise les données des ennemis purple.
- void init_orange (sprite_t *orange[ ])

  La fonction initialise les données des ennemis orange.
- void init_missile (world_t *world)

  La fonction initialise les données des missiles du joueur.
- void init_missile_enemy (world_t *world, sprite_t *enemy[ ], sprite_t *missile[ ], int total)

  La fonction initialise les données des missiles des ennemis.

- void init_missile_boss (world_t *world)
  La fonction initialise les données des missiles du boss.
- void depassement_enemy_vertical (world_t *world, sprite_t *enemy[ ], int total)
  La fonction incrémente le compteur et supprime les ennemis qui dépassent la bordure inférieure.
- void depassement_enemy_horizontal (world_t *world, sprite_t *enemy[ ], int total, int limite)
  La fonction indrémente le compteur et supprime les ennemis violets qui dépassent la bordure inférieure.
- void depassement_missile (sprite_t *missile[ ])
  La fonction supprime le missile s'il dépasse le haut de l'écran de jeu.
- void depassement_missile_enemy (sprite_t *missile[ ], int total)
  La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.
- void depassement_missile_boss (world_t *world)
  La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.
- int sprite_collide (sprite_t *sp2, sprite_t *sp1, world_t *world)
  La fonction vérifie s'il y a collision entre deux sprites.
- void handle_sprite_collide (sprite_t *sp1, sprite_t *sp2, world_t *world)
  La fonction traite l'évènement de collision en rendant invisible les sprites et les arrêtant.
- void update_enemy_vertical (world_t *world, sprite_t *enemy[ ], int total)
  La fonction met à jour les données des ennemis à déplacement vertical.
- void update_enemy_horizontal (world_t *world, sprite_t *enemy[ ], int total, int limite)
  La fonction met à jour les données des ennemis à déplacement horizontal.
- void update_boss (world_t *world)
  La fonction met à jour les données du boss.
- void update_missile (world_t *world)
  La fonction met à jour les données du missile.
- void update_missile_enemy (world_t *world, sprite_t *enemy[ ], sprite_t *missile[ ], int total)
  La fonction met à jour les données du missile de l'ennemi.
- void update_missile_boss (world_t *world)
  La fonction met à jour les données du missile du boss.
- void compute_game (world_t *world)
  La fonction donne l'état de la partie.
- void update_data (world_t *world)
  La fonction met à jour les données en tenant compte de la physique du monde.

4.13.1 Detailed Description

Gestion du monde.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.13.2 Function Documentation

4.13.2.1 clean_data_1()

```
void clean_data_1 (
            world_t * world )
```

La fonction nettoie les données du monde.

Parameters

world	Les données du monde.

4.13.2.2 clean_data_2()

```
void clean_data_2 (
            sprite_t * tab[],
            int total )
```

La fonction nettoie les données du monde.

Parameters

tab	Les données des tableaux du monde.
total	Nombre total.

4.13.2.3 compute_game()

```
void compute_game (
            world_t * world )
```

La fonction donne l'état de la partie.

Parameters

world	Les données du monde.

4.13.2.4 depassement_enemy_horizontal()

```
void depassement_enemy_horizontal (
            world_t * world,
            sprite_t * enemy[],
            int total,
            int limite )
```

La fonction indrémente le compteur et supprime les ennemis violets qui dépassent la bordure inférieure.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.
limite	Limite de l'écran à ne pas dépasser.

4.13.2.5 depassement_enemy_vertical()

```
void depassement_enemy_vertical (
            world_t * world,
            sprite_t * enemy[],
            int total )
```

La fonction incrémente le compteur et supprime les ennemis qui dépassent la bordure inférieure.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.

4.13.2.6 depassement_missile()

```
void depassement_missile (
            sprite_t * missile[] )
```

La fonction supprime le missile s'il dépasse le haut de l'écran de jeu.

Parameters

missile	Les missiles du joueur.

4.13.2.7 depassement_missile_boss()

```
void depassement_missile_boss (
            world_t * world )
```

La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.

Parameters

world	Les données du monde.

4.13.2.8 depassement_missile_enemy()

```
void depassement_missile_enemy (
```

```
            sprite_t * missile[],
            int total )
```

La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.

Parameters

missile	Les missiles.
total	Le total.

4.13.2.9 generate_number()

```
int generate_number (
            int a,
            int b )
```

Generation d'un nombre entier compris entre a et b (ce dernier non inclus).

Parameters

a	Borne inférieure.
b	Borne supérieure.

Returns

Un nombre généré aléatoirement entre *a* et *b*.

4.13.2.10 handle_sprite_collide()

```
void handle_sprite_collide (
            sprite_t * sp1,
            sprite_t * sp2,
            world_t * world )
```

La fonction traite l'évènement de collision en rendant invisible les sprites et les arrêtant.

Parameters

sp1	L'un des sprites.
sp2	L'autre sprite.
world	Les données du monde.

4.13.2.11 init_blue()

```
void init_blue (
            sprite_t * blue[] )
```

La fonction initialise les données des ennemis blue.

Parameters

blue	Le tableau d'ennemis blue.

4.13.2.12 init_data_1()

```
void init_data_1 (
            world_t * world )
```

La fonction initialise les données du monde du jeu.

Parameters

world	Les données du monde.

4.13.2.13 init_data_2()

```
void init_data_2 (
            sprite_t * tab[],
            int total )
```

La fonction initialise les données des tableaux du monde du jeu.

Parameters

tab	Les données des tableaux du monde.
total	Nombre total.

4.13.2.14 init_green()

```
void init_green (
            sprite_t * green[] )
```

La fonction initialise les données des ennemis green.

Parameters

green	Le tableau d'ennemis green.

4.13.2.15 init_missile()

```
void init_missile (
            world_t * world )
```

La fonction initialise les données des missiles du joueur.

Parameters

world	Les données du monde.

4.13.2.16 init_missile_boss()

```
void init_missile_boss (
            world_t * world )
```

La fonction initialise les données des missiles du boss.

Parameters

world	Les données du monde.

4.13.2.17 init_missile_enemy()

```
void init_missile_enemy (
            world_t * world,
            sprite_t * enemy[],
            sprite_t * missile[],
            int total )
```

La fonction initialise les données des missiles des ennemis.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
enemy	Tableau des sprites des missiles.
total	Nombre d'ennemis total.

4.13.2.18 init_orange()

```
void init_orange (
            sprite_t * orange[] )
```

La fonction initialise les données des ennemis orange.

Parameters

orange	Le tableau d'ennemis orange.

4.13.2.19 init_purple()

```
void init_purple (
            sprite_t * purple[] )
```

La fonction initialise les données des ennemis purple.

Parameters

purple	Le tableau d'ennemis purple.

4.13.2.20 init_red()

```
void init_red (
            sprite_t * red[] )
```

La fonction initialise les données des ennemis red.

Parameters

red	Le tableau d'ennemis red.

4.13.2.21 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.

4.13.2.22 max()

```
int max (
            int a,
            int b )
```

Maximum entre a et b.

Parameters

a	Entier a.
b	Entier b.

Returns

Le maximum entre *a* et *b*.

4.13.2.23 sprite_collide()

```
int sprite_collide (
            sprite_t * sp2,
            sprite_t * sp1,
            world_t * world )
```

La fonction vérifie s'il y a collision entre deux sprites.

Parameters

sp2	L'un des sprites.
sp1	L'autre sprite.
world	Les données du monde.

Returns

1 si les sprites se collisionnent, 0 sinon.

4.13.2.24 update_boss()

```
void update_boss (
            world_t * world )
```

La fonction met à jour les données du boss.

Parameters

world	Les données du monde.

4.13.2.25 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.

4.13.2.26 update_enemy_horizontal()

```
void update_enemy_horizontal (
            world_t * world,
            sprite_t * enemy[],
            int total,
            int limite )
```

La fonction met à jour les données des ennemis à déplacement horizontal.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.
limite	Limite de l'écran à ne pas dépasser.

4.13.2.27 update_enemy_vertical()

```
void update_enemy_vertical (
            world_t * world,
```

```
            sprite_t * enemy[],
            int total )
```

La fonction met à jour les données des ennemis à déplacement vertical.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.

4.13.2.28 update_missile()

```
void update_missile (
            world_t * world )
```

La fonction met à jour les données du missile.

Parameters

world	Les données du monde.

4.13.2.29 update_missile_boss()

```
void update_missile_boss (
            world_t * world )
```

La fonction met à jour les données du missile du boss.

Parameters

world	Les données du monde.

4.13.2.30 update_missile_enemy()

```
void update_missile_enemy (
            world_t * world,
            sprite_t * enemy[],
            sprite_t * missile[],
            int total )
```

La fonction met à jour les données du missile de l'ennemi.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
missile	Tableau des missiles d'ennemis.
total	Nombre d'ennemis total.

4.14 world.h File Reference

Bibliothèque de world.

```
#include "biblio.h"
#include "sprite.h"
```

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

Classes

- struct world_s

Typedefs

- typedef struct world_s **world_t**

Functions

- void init_data_1 (world_t *world)

  La fonction initialise les données du monde du jeu.
- void init_data_2 (sprite_t *tab[ ], int total)

  La fonction initialise les données des tableaux du monde du jeu.
- void clean_data_1 (world_t *world)

  La fonction nettoie les données du monde.
- void clean_data_2 (sprite_t *tab[ ], int total)

  La fonction nettoie les données du monde.
- int is_game_over (world_t *world)

  La fonction indique si le jeu est fini en fonction des données du monde.
- int generate_number (int a, int b)

  Generation d'un nombre entier compris entre a et b (ce dernier non inclus).
- int max (int a, int b)

  Maximum entre a et b.
- void init_blue (sprite_t *blue[ ])

  La fonction initialise les données des ennemis blue.
- void init_red (sprite_t *red[ ])

  La fonction initialise les données des ennemis red.
- void init_green (sprite_t *green[ ])

  La fonction initialise les données des ennemis green.
- void init_purple (sprite_t *purple[ ])

  La fonction initialise les données des ennemis purple.

- void init_orange (sprite_t *orange[ ])

  La fonction initialise les données des ennemis orange.
- void init_missile (world_t *world)

  La fonction initialise les données des missiles du joueur.
- void init_missile_enemy (world_t *world, sprite_t *enemy[ ], sprite_t *missile[ ], int total)

  La fonction initialise les données des missiles des ennemis.
- void init_missile_boss (world_t *world)

  La fonction initialise les données des missiles du boss.
- void depassement_enemy_vertical (world_t *world, sprite_t *enemy[ ], int total)

  La fonction incrémente le compteur et supprime les ennemis qui dépassent la bordure inférieure.
- void depassement_enemy_horizontal (world_t *world, sprite_t *enemy[ ], int total, int limite)

  La fonction indrémente le compteur et supprime les ennemis violets qui dépassent la bordure inférieure.
- void depassement_orange (world_t *world)

  La fonction incrémente le compteur et supprime les ennemis oranges qui dépassent la bordure inférieure.
- void depassement_missile (sprite_t *missile[ ])

  La fonction supprime le missile s'il dépasse le haut de l'écran de jeu.
- void depassement_missile_enemy (sprite_t *missile[ ], int total)

  La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.
- void depassement_missile_boss (world_t *world)

  La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.
- int sprite_collide (sprite_t *sp2, sprite_t *sp1, world_t *world)

  La fonction vérifie s'il y a collision entre deux sprites.
- void handle_sprite_collide (sprite_t *sp1, sprite_t *sp2, world_t *world)

  La fonction traite l'évènement de collision en rendant invisible les sprites et les arrêtant.
- void update_enemy_vertical (world_t *world, sprite_t *enemy[ ], int total)

  La fonction met à jour les données des ennemis à déplacement vertical.
- void update_enemy_horizontal (world_t *world, sprite_t *enemy[ ], int total, int limite)

  La fonction met à jour les données des ennemis à déplacement horizontal.
- void update_boss (world_t *world)

  La fonction met à jour les données du boss.
- void update_missile (world_t *world)

  La fonction met à jour les données du missile.
- void update_missile_enemy (world_t *world, sprite_t *enemy[ ], sprite_t *missile[ ], int total)

  La fonction met à jour les données du missile de l'ennemi.
- void update_missile_boss (world_t *world)

  La fonction met à jour les données du missile du boss.
- void compute_game (world_t *world)

  La fonction donne l'état de la partie.
- void update_data (world_t *world)

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

4.14.1 Detailed Description

Bibliothèque de world.

Author

MOSELLE Marie-Luc & MATHIEU STEINBACH Hugo

Date

13 mai 2020

4.14.2 Function Documentation

4.14.2.1 clean_data_1()

```
void clean_data_1 (
            world_t * world )
```

La fonction nettoie les données du monde.

Parameters

world	Les données du monde.

4.14.2.2 clean_data_2()

```
void clean_data_2 (
            sprite_t * tab[],
            int total )
```

La fonction nettoie les données du monde.

Parameters

tab	Les données des tableaux du monde.
total	Nombre total.

4.14.2.3 compute_game()

```
void compute_game (
            world_t * world )
```

La fonction donne l'état de la partie.

Parameters

world	Les données du monde.

4.14.2.4 depassement_enemy_horizontal()

```
void depassement_enemy_horizontal (
```

```
            world_t * world,
            sprite_t * enemy[],
            int total,
            int limite )
```

La fonction indrémente le compteur et supprime les ennemis violets qui dépassent la bordure inférieure.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.
limite	Limite de l'écran à ne pas dépasser.

4.14.2.5 depassement_enemy_vertical()

```
void depassement_enemy_vertical (
            world_t * world,
            sprite_t * enemy[],
            int total )
```

La fonction incrémente le compteur et supprime les ennemis qui dépassent la bordure inférieure.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.

4.14.2.6 depassement_missile()

```
void depassement_missile (
            sprite_t * missile[] )
```

La fonction supprime le missile s'il dépasse le haut de l'écran de jeu.

Parameters

missile	Les missiles du joueur.

4.14.2.7 depassement_missile_boss()

```
void depassement_missile_boss (
            world_t * world )
```

La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.

Parameters

world	Les données du monde.

4.14.2.8 depassement_missile_enemy()

```
void depassement_missile_enemy (
            sprite_t * missile[],
            int total )
```

La fonction supprime le missile s'il dépasse le bas de l'écran de jeu.

Parameters

missile	Les missiles.
total	Le total.

4.14.2.9 depassement_orange()

```
void depassement_orange (
            world_t * world )
```

La fonction incrémente le compteur et supprime les ennemis oranges qui dépassent la bordure inférieure.

Parameters

world	Les données du monde.

4.14.2.10 generate_number()

```
int generate_number (
            int a,
            int b )
```

Generation d'un nombre entier compris entre a et b (ce dernier non inclus).

Parameters

a	Borne inférieure.
b	Borne supérieure.

Returns

Un nombre généré aléatoirement entre *a* et *b*.

4.14.2.11 handle_sprite_collide()

```
void handle_sprite_collide (
            sprite_t * sp1,
            sprite_t * sp2,
            world_t * world )
```

La fonction traite l'évènement de collision en rendant invisible les sprites et les arrêtant.

Parameters

sp1	L'un des sprites.
sp2	L'autre sprite.
world	Les données du monde.

4.14.2.12 init_blue()

```
void init_blue (
            sprite_t * blue[] )
```

La fonction initialise les données des ennemis blue.

Parameters

blue	Le tableau d'ennemis blue.

4.14.2.13 init_data_1()

```
void init_data_1 (
            world_t * world )
```

La fonction initialise les données du monde du jeu.

Parameters

world	Les données du monde.

4.14.2.14 init_data_2()

```
void init_data_2 (
            sprite_t * tab[],
            int total )
```

La fonction initialise les données des tableaux du monde du jeu.

Parameters

tab	Les données des tableaux du monde.
total	Nombre total.

4.14.2.15 init_green()

```
void init_green (
            sprite_t * green[] )
```

La fonction initialise les données des ennemis green.

Parameters

green	Le tableau d'ennemis green.

4.14.2.16 init_missile()

```
void init_missile (
            world_t * world )
```

La fonction initialise les données des missiles du joueur.

Parameters

world	Les données du monde.

4.14.2.17 init_missile_boss()

```
void init_missile_boss (
            world_t * world )
```

La fonction initialise les données des missiles du boss.

Parameters

world	Les données du monde.

4.14.2.18 init_missile_enemy()

```
void init_missile_enemy (
            world_t * world,
            sprite_t * enemy[],
            sprite_t * missile[],
            int total )
```

La fonction initialise les données des missiles des ennemis.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
enemy	Tableau des sprites des missiles.
total	Nombre d'ennemis total.

4.14.2.19 init_orange()

```
void init_orange (
            sprite_t * orange[] )
```

La fonction initialise les données des ennemis orange.

Parameters

orange	Le tableau d'ennemis orange.

4.14.2.20 init_purple()

```
void init_purple (
            sprite_t * purple[] )
```

La fonction initialise les données des ennemis purple.

Parameters

purple	Le tableau d'ennemis purple.

4.14.2.21 init_red()

```
void init_red (
            sprite_t * red[] )
```

La fonction initialise les données des ennemis red.

Parameters

red	Le tableau d'ennemis red.

4.14.2.22 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.

4.14.2.23 max()

```
int max (
            int a,
            int b )
```

Maximum entre a et b.

Parameters

a	Entier a.
b	Entier b.

Returns

Le maximum entre *a* et *b*.

4.14.2.24 sprite_collide()

```
int sprite_collide (
            sprite_t * sp2,
            sprite_t * sp1,
            world_t * world )
```

La fonction vérifie s'il y a collision entre deux sprites.

Parameters

sp2	L'un des sprites.
sp1	L'autre sprite.
world	Les données du monde.

Returns

1 si les sprites se collisionnent, 0 sinon.

4.14.2.25 update_boss()

```
void update_boss (
            world_t * world )
```

La fonction met à jour les données du boss.

Parameters

world	Les données du monde.

4.14.2.26 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.

4.14.2.27 update_enemy_horizontal()

```
void update_enemy_horizontal (
            world_t * world,
            sprite_t * enemy[],
            int total,
            int limite )
```

La fonction met à jour les données des ennemis à déplacement horizontal.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.
limite	Limite de l'écran à ne pas dépasser.

4.14.2.28 update_enemy_vertical()

```
void update_enemy_vertical (
            world_t * world,
            sprite_t * enemy[],
            int total )
```

La fonction met à jour les données des ennemis à déplacement vertical.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
total	Nombre d'ennemis total.

4.14.2.29 update_missile()

```
void update_missile (
            world_t * world )
```

La fonction met à jour les données du missile.

Parameters

world	Les données du monde.

4.14.2.30 update_missile_boss()

```
void update_missile_boss (
            world_t * world )
```

La fonction met à jour les données du missile du boss.

Parameters

world	Les données du monde.

4.14.2.31 update_missile_enemy()

```
void update_missile_enemy (
            world_t * world,
            sprite_t * enemy[],
            sprite_t * missile[],
            int total )
```

La fonction met à jour les données du missile de l'ennemi.

Parameters

world	Les données du monde.
enemy	Tableau des sprites d'ennemis.
missile	Tableau des missiles d'ennemis.
total	Nombre d'ennemis total.

Index

apply_background
    ressources.c, 23
    ressources.h, 24
apply_sprite
    renderer.c, 16
    renderer.h, 19
apply_sprite_boss
    renderer.c, 17
    renderer.h, 20
apply_sprite_missile
    renderer.c, 17
    renderer.h, 20
apply_texture
    sdl2-light.c, 26

background
    ressources_s, 5
biblio.h, 13
blue
    world_s, 10
boss
    ressources_s, 5
    world_s, 10
boss_speed_x
    world_s, 10
boss_speed_y
    world_s, 10

clean
    renderer.c, 17
    renderer.h, 21
clean_data_1
    world.c, 35
    world.h, 47
clean_data_2
    world.c, 36
    world.h, 47
clean_ressources
    ressources.c, 23
    ressources.h, 25
clean_sdl
    sdl2-light.c, 27
clean_texture
    sdl2-light.c, 27
clear_renderer
    sdl2-light.c, 27
compteur
    world_s, 10
compute_game
    world.c, 36
    world.h, 47

depassement_enemy_horizontal
    world.c, 36
    world.h, 47
depassement_enemy_vertical
    world.c, 37
    world.h, 48
depassement_missile
    world.c, 37
    world.h, 48
depassement_missile_boss
    world.c, 37
    world.h, 48
depassement_missile_enemy
    world.c, 37
    world.h, 49
depassement_orange
    world.h, 49

enemy_blue
    ressources_s, 5
enemy_green
    ressources_s, 6
enemy_orange
    ressources_s, 6
enemy_purple
    ressources_s, 6
enemy_red
    ressources_s, 6
etat_partie
    world_s, 10
event.c, 13
    handle_events, 14
event.h, 14
    handle_events, 14

font
    ressources_s, 6

gameover
    world_s, 10
generate_number
    world.c, 38
    world.h, 49
green
    world_s, 10

h
    sprite_s, 7
handle_events

event.c, 14
event.h, 14
handle_sprite_collide
world.c, 38
world.h, 50
hp
sprite_s, 7

init
renderer.c, 18
renderer.h, 21
init_blue
world.c, 38
world.h, 50
init_data_1
world.c, 39
world.h, 50
init_data_2
world.c, 39
world.h, 50
init_green
world.c, 39
world.h, 51
init_missile
world.c, 40
world.h, 51
init_missile_boss
world.c, 40
world.h, 51
init_missile_enemy
world.c, 40
world.h, 52
init_orange
world.c, 41
world.h, 52
init_purple
world.c, 41
world.h, 52
init_red
world.c, 41
world.h, 53
init_ressources
ressources.c, 23
ressources.h, 25
init_sdl
sdl2-light.c, 27
init_sprite
sprite.c, 29
sprite.h, 32
is_game_over
world.c, 41
world.h, 53
is_visible
sprite_s, 8

load_image
sdl2-light.c, 28

main.c, 15
max
world.c, 42
world.h, 53
missile
ressources_s, 6
world_s, 11
missile_boss
ressources_s, 6
world_s, 11
missile_orange
ressources_s, 6
world_s, 11
missile_purple
ressources_s, 7
world_s, 11

orange
world_s, 11

pause
sdl2-light.c, 28
print_sprite
sprite.c, 30
sprite.h, 32
purple
world_s, 11

red
world_s, 11
refresh
renderer.c, 18
renderer.h, 21
refresh_graphics
renderer.c, 18
renderer.h, 22
renderer.c, 15
apply_sprite, 16
apply_sprite_boss, 17
apply_sprite_missile, 17
clean, 17
init, 18
refresh, 18
refresh_graphics, 18
renderer.h, 19
apply_sprite, 19
apply_sprite_boss, 20
apply_sprite_missile, 20
clean, 21
init, 21
refresh, 21
refresh_graphics, 22
ressources.c, 22
apply_background, 23
clean_ressources, 23
init_ressources, 23
ressources.h, 24
apply_background, 24
clean_ressources, 25
init_ressources, 25

ressources_s, 5
    background, 5
    boss, 5
    enemy_blue, 5
    enemy_green, 6
    enemy_orange, 6
    enemy_purple, 6
    enemy_red, 6
    font, 6
    missile, 6
    missile_boss, 6
    missile_orange, 6
    missile_purple, 7
    ship, 7

score
    world_s, 11
sdl2-light.c, 25
    apply_texture, 26
    clean_sdl, 27
    clean_texture, 27
    clear_renderer, 27
    init_sdl, 27
    load_image, 28
    pause, 28
    update_screen, 29
set_invisible
    sprite.c, 30
    sprite.h, 32
set_visible
    sprite.c, 31
    sprite.h, 33
ship
    ressources_s, 7
    world_s, 12
sprite.c, 29
    init_sprite, 29
    print_sprite, 30
    set_invisible, 30
    set_visible, 31
sprite.h, 31
    init_sprite, 32
    print_sprite, 32
    set_invisible, 32
    set_visible, 33
sprite_collide
    world.c, 42
    world.h, 54
sprite_s, 7
    h, 7
    hp, 7
    is_visible, 8
    v, 8
    w, 8
    x, 8
    y, 8
sprite_t, 8

tests.c, 33
texture_t, 9
timer
    world_s, 12

update_boss
    world.c, 42
    world.h, 54
update_data
    world.c, 43
    world.h, 54
update_enemy_horizontal
    world.c, 43
    world.h, 54
update_enemy_vertical
    world.c, 43
    world.h, 55
update_missile
    world.c, 44
    world.h, 55
update_missile_boss
    world.c, 44
    world.h, 55
update_missile_enemy
    world.c, 44
    world.h, 56
update_screen
    sdl2-light.c, 29

v
    sprite_s, 8
vague
    world_s, 12

w
    sprite_s, 8
world.c, 34
    clean_data_1, 35
    clean_data_2, 36
    compute_game, 36
    depassement_enemy_horizontal, 36
    depassement_enemy_vertical, 37
    depassement_missile, 37
    depassement_missile_boss, 37
    depassement_missile_enemy, 37
    generate_number, 38
    handle_sprite_collide, 38
    init_blue, 38
    init_data_1, 39
    init_data_2, 39
    init_green, 39
    init_missile, 40
    init_missile_boss, 40
    init_missile_enemy, 40
    init_orange, 41
    init_purple, 41
    init_red, 41
    is_game_over, 41
    max, 42
    sprite_collide, 42

update_boss, 42
update_data, 43
update_enemy_horizontal, 43
update_enemy_vertical, 43
update_missile, 44
update_missile_boss, 44
update_missile_enemy, 44
world.h, 45
clean_data_1, 47
clean_data_2, 47
compute_game, 47
depassement_enemy_horizontal, 47
depassement_enemy_vertical, 48
depassement_missile, 48
depassement_missile_boss, 48
depassement_missile_enemy, 49
depassement_orange, 49
generate_number, 49
handle_sprite_collide, 50
init_blue, 50
init_data_1, 50
init_data_2, 50
init_green, 51
init_missile, 51
init_missile_boss, 51
init_missile_enemy, 52
init_orange, 52
init_purple, 52
init_red, 53
is_game_over, 53
max, 53
sprite_collide, 54
update_boss, 54
update_data, 54
update_enemy_horizontal, 54
update_enemy_vertical, 55
update_missile, 55
update_missile_boss, 55
update_missile_enemy, 56
world_s, 9
blue, 10
boss, 10
boss_speed_x, 10
boss_speed_y, 10
compteur, 10
etat_partie, 10
gameover, 10
green, 10
missile, 11
missile_boss, 11
missile_orange, 11
missile_purple, 11
orange, 11
purple, 11
red, 11
score, 11
ship, 12
timer, 12
vague, 12
world_t, 12

x
sprite_s, 8

y
sprite_s, 8