

## My Project

Generated by Doxygen 1.8.18



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 gameinfo_s Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 close	5
3.1.2.2 enter	6
3.1.2.3 finishTime	6
3.1.2.4 gamemode	6
3.1.2.5 select	6
3.1.2.6 startTime	6
3.2 playerinfo_s Struct Reference	6
3.2.1 Detailed Description	7
3.2.2 Member Data Documentation	7
3.2.2.1 bestTime	7
3.2.2.2 hasShip	7
3.2.2.3 lastTime	7
3.2.2.4 name	7
3.2.2.5 selectedShip	7
3.2.2.6 stars	7
3.3 resources_s Struct Reference	8
3.3.1 Detailed Description	8
3.3.2 Member Data Documentation	8
3.3.2.1 background	8
3.3.2.2 background_menu	8
3.3.2.3 ennemy_ship	9
3.3.2.4 finish_line	9
3.3.2.5 font	9
3.3.2.6 font_menu	9
3.3.2.7 meteorite	9
3.3.2.8 missile	9
3.3.2.9 return_menu	9
3.3.2.10 select	9
3.3.2.11 selected	10
3.3.2.12 shield	10
3.3.2.13 ship	10
3.3.2.14 ship1	10
3.3.2.15 ship2	10

3.3.2.16 ship3 . . . . .	10
3.3.2.17 ship4 . . . . .	10
3.3.2.18 star . . . . .	10
3.3.2.19 time . . . . .	11
3.4 sprite_s Struct Reference . . . . .	11
3.4.1 Detailed Description . . . . .	11
3.4.2 Member Data Documentation . . . . .	11
3.4.2.1 h . . . . .	11
3.4.2.2 w . . . . .	11
3.4.2.3 x . . . . .	12
3.4.2.4 y . . . . .	12
3.5 world_s Struct Reference . . . . .	12
3.5.1 Detailed Description . . . . .	12
3.5.2 Member Data Documentation . . . . .	13
3.5.2.1 bonus . . . . .	13
3.5.2.2 bonus_type . . . . .	13
3.5.2.3 collision . . . . .	13
3.5.2.4 collision_finish_line . . . . .	13
3.5.2.5 ennemy . . . . .	13
3.5.2.6 finish_line . . . . .	13
3.5.2.7 gameover . . . . .	13
3.5.2.8 hasShield . . . . .	14
3.5.2.9 is_bonus_available . . . . .	14
3.5.2.10 is_ennemy_dead . . . . .	14
3.5.2.11 is_missile_free . . . . .	14
3.5.2.12 meteorite . . . . .	14
3.5.2.13 missile . . . . .	14
3.5.2.14 ship . . . . .	14
<b>4 File Documentation</b> . . . . .	<b>15</b>
4.1 definition.h File Reference . . . . .	15
4.2 graphique.c File Reference . . . . .	15
4.2.1 Detailed Description . . . . .	16
4.2.2 Function Documentation . . . . .	16
4.2.2.1 apply_background() . . . . .	16
4.2.2.2 apply_bonus() . . . . .	17
4.2.2.3 apply_enemies() . . . . .	17
4.2.2.4 apply_missiles() . . . . .	17
4.2.2.5 apply_sprite() . . . . .	18
4.2.2.6 apply_texts() . . . . .	18
4.2.2.7 apply_walls() . . . . .	18
4.2.2.8 clean_resources() . . . . .	19

4.2.2.9 init_resources()	19
4.2.2.10 refresh_graphics()	19
4.3 graphique.h File Reference	20
4.3.1 Detailed Description	21
4.3.2 Function Documentation	21
4.3.2.1 apply_background()	21
4.3.2.2 apply_bonus()	21
4.3.2.3 apply_enemies()	22
4.3.2.4 apply_missiles()	22
4.3.2.5 apply_sprite()	22
4.3.2.6 apply_texts()	23
4.3.2.7 apply_walls()	23
4.3.2.8 clean_resources()	23
4.3.2.9 init_resources()	24
4.3.2.10 refresh_graphics()	24
4.4 logique.c File Reference	24
4.4.1 Detailed Description	25
4.4.2 Function Documentation	26
4.4.2.1 abs()	26
4.4.2.2 depassement_d()	26
4.4.2.3 depassement_g()	26
4.4.2.4 handle_sprites_collision()	27
4.4.2.5 init_bonus()	27
4.4.2.6 init_data()	27
4.4.2.7 init_enemies()	28
4.4.2.8 init_missiles()	28
4.4.2.9 init_sprite()	28
4.4.2.10 init_walls()	29
4.4.2.11 is_game_over()	29
4.4.2.12 print_sprite()	29
4.4.2.13 retrieve_index()	29
4.4.2.14 shoot()	30
4.4.2.15 sprites_collide()	30
4.4.2.16 update_data()	31
4.4.2.17 update_sprites()	31
4.5 logique.h File Reference	31
4.5.1 Detailed Description	33
4.5.2 Function Documentation	33
4.5.2.1 abs()	33
4.5.2.2 depassement_d()	33
4.5.2.3 depassement_g()	34
4.5.2.4 handle_sprites_collision()	34

4.5.2.5	init_bonus()	34
4.5.2.6	init_data()	35
4.5.2.7	init_enemies()	35
4.5.2.8	init_missiles()	35
4.5.2.9	init_sprite()	35
4.5.2.10	init_walls()	36
4.5.2.11	is_game_over()	36
4.5.2.12	print_sprite()	37
4.5.2.13	retrieve_index()	37
4.5.2.14	shoot()	37
4.5.2.15	sprites_collide()	38
4.5.2.16	update_data()	38
4.5.2.17	update_sprites()	38
4.6	main.c File Reference	39
4.6.1	Detailed Description	39
4.7	menu.c File Reference	39
4.7.1	Detailed Description	40
4.7.2	Function Documentation	40
4.7.2.1	apply_menu()	40
4.7.2.2	buy_select_ship()	41
4.7.2.3	menu_credits()	41
4.7.2.4	menu_infos()	41
4.7.2.5	menu_main()	42
4.7.2.6	menu_shop()	42
4.7.2.7	print_select()	43
4.7.2.8	print_stars()	43
4.8	menu.h File Reference	44
4.8.1	Detailed Description	44
4.8.2	Function Documentation	44
4.8.2.1	apply_menu()	45
4.8.2.2	buy_select_ship()	45
4.8.2.3	menu_credits()	45
4.8.2.4	menu_infos()	46
4.8.2.5	menu_main()	46
4.8.2.6	menu_shop()	46
4.8.2.7	print_select()	47
4.8.2.8	print_stars()	47
4.9	sdl2-light.c File Reference	48
4.9.1	Detailed Description	48
4.9.2	Function Documentation	48
4.9.2.1	apply_texture()	49
4.9.2.2	clean_sdl()	49

4.9.2.3 clean_texture()	49
4.9.2.4 clear_renderer()	50
4.9.2.5 init_sdl()	50
4.9.2.6 load_image()	50
4.9.2.7 pause()	51
4.9.2.8 update_screen()	51
4.10 sdl2-light.h File Reference	51
4.10.1 Detailed Description	52
4.10.2 Function Documentation	52
4.10.2.1 apply_texture()	52
4.10.2.2 clean_sdl()	53
4.10.2.3 clean_texture()	53
4.10.2.4 clear_renderer()	53
4.10.2.5 init_sdl()	53
4.10.2.6 load_image()	54
4.10.2.7 pause()	54
4.10.2.8 update_screen()	55
4.11 sdl2-ttf-light.c File Reference	55
4.11.1 Detailed Description	55
4.11.2 Function Documentation	56
4.11.2.1 apply_text()	56
4.11.2.2 clean_font()	56
4.11.2.3 load_font()	57
4.12 sdl2-ttf-light.h File Reference	57
4.12.1 Detailed Description	57
4.12.2 Function Documentation	58
4.12.2.1 apply_text()	58
4.12.2.2 clean_font()	58
4.12.2.3 load_font()	59
4.13 tests.c File Reference	59
4.13.1 Detailed Description	60
4.14 transition.c File Reference	60
4.14.1 Detailed Description	61
4.14.2 Function Documentation	61
4.14.2.1 clean()	61
4.14.2.2 concat_array_playername()	61
4.14.2.3 formatted_charArray_to_uint()	62
4.14.2.4 handle_events()	62
4.14.2.5 init()	62
4.14.2.6 init_player()	63
4.14.2.7 save_info()	63
4.14.2.8 save_score()	63

4.15 transition.h File Reference . . . . .	64
4.15.1 Detailed Description . . . . .	65
4.15.2 Function Documentation . . . . .	65
4.15.2.1 clean() . . . . .	65
4.15.2.2 concat_array_playername() . . . . .	65
4.15.2.3 formatted_charArray_to_uint() . . . . .	66
4.15.2.4 handle_events() . . . . .	66
4.15.2.5 init() . . . . .	66
4.15.2.6 init_player() . . . . .	67
4.15.2.7 save_info() . . . . .	67
4.15.2.8 save_score() . . . . .	67
<b>Index</b>	<b>69</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">gameinfo_s</a>	Stockage des informations du jeu en cours (extension) . . . . .	5
<a href="#">playerinfo_s</a>	Stockage des informations du joueur(extension) . . . . .	6
<a href="#">resources_s</a>	Stockage des ressources nécessaires à l'affichage graphique . . . . .	8
<a href="#">sprite_s</a>	Représentation d'un sprite du jeu . . . . .	11
<a href="#">world_s</a>	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:

<a href="#">definition.h</a>	Header regroupant toutes les constantes utiles au programme . . . . .	15
<a href="#">graphique.c</a>	Module gérant la partie graphique du jeu . . . . .	15
<a href="#">graphique.h</a>	Header du module graphique . . . . .	20
<a href="#">logique.c</a>	Module gérant la partie logique du jeu . . . . .	24
<a href="#">logique.h</a>	Header du module logique . . . . .	31
<a href="#">main.c</a>	Programme principal initial du niveau 3 + extensions . . . . .	39
<a href="#">menu.c</a>	Module gérant le fonctionnement du menu . . . . .	39
<a href="#">menu.h</a>	Header regroupant toutes les fonctions appelant des fonctions graphique et logique . . . . .	44
<a href="#">sdl2-light.c</a>	Sur-couche de SDL2 pour simplifier son utilisation pour le projet (modifié) . . . . .	48
<a href="#">sdl2-light.h</a>	Sur-couche de SDL2 pour simplifier son utilisation pour le projet (modifié) . . . . .	51
<a href="#">sdl2-ttf-light.c</a>	Sur-couche de SDL2_ttf pour simplifier son utilisation pour le projet (modifié) . . . . .	55
<a href="#">sdl2-ttf-light.h</a>	Sur-couche de SDL2_ttf pour simplifier son utilisation pour le projet (modifié) . . . . .	57
<a href="#">tests.c</a>	Programme testant la partie logique du jeu . . . . .	59
<a href="#">transition.c</a>	Module gérant la transition entre les modules différents modules . . . . .	60
<a href="#">transition.h</a>	Header regroupant toutes les fonctions appelant des fonctions graphique et logique . . . . .	64



## Chapter 3

# Class Documentation

### 3.1 gameinfo\_s Struct Reference

Stockage des informations du jeu en cours (extension)

```
#include <logique.h>
```

#### Public Attributes

- unsigned int [startTime](#)
- unsigned int [finishTime](#)
- unsigned int [gamemode](#)
- unsigned int [close](#)
- unsigned int [select](#)
- unsigned int [enter](#)

#### 3.1.1 Detailed Description

Stockage des informations du jeu en cours (extension)

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 close

```
unsigned int gameinfo_s::close
```

Gère la demande de fermeture du jeu

### 3.1.2.2 enter

```
unsigned int gameinfo_s::enter
```

Gère l'entrée dans une section du menu

### 3.1.2.3 finishTime

```
unsigned int gameinfo_s::finishTime
```

Temps de fin de partie

### 3.1.2.4 gamemode

```
unsigned int gameinfo_s::gamemode
```

Mode du jeu en cours

### 3.1.2.5 select

```
unsigned int gameinfo_s::select
```

Gère la sélection du menu

### 3.1.2.6 startTime

```
unsigned int gameinfo_s::startTime
```

Temps du début de partie

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

- [logique.h](#)

## 3.2 playerinfo\_s Struct Reference

Stockage des informations du joueur(extension)

```
#include <logique.h>
```

### Public Attributes

- char [name](#) [50]
- unsigned int [stars](#)
- unsigned int [bestTime](#)
- unsigned int [lastTime](#)
- unsigned int [hasShip](#) [4]
- unsigned int [selectedShip](#)

### 3.2.1 Detailed Description

Stockage des informations du joueur(extension)

### 3.2.2 Member Data Documentation

#### 3.2.2.1 bestTime

```
unsigned int playerinfo_s::bestTime
```

Meilleur temps du joueur

#### 3.2.2.2 hasShip

```
unsigned int playerinfo_s::hasShip[4]
```

Indique si le joueur possède déjà la texture du vaisseau i

#### 3.2.2.3 lastTime

```
unsigned int playerinfo_s::lastTime
```

Dernier temps effectué par le joueur

#### 3.2.2.4 name

```
char playerinfo_s::name[50]
```

Nom du joueur

#### 3.2.2.5 selectedShip

```
unsigned int playerinfo_s::selectedShip
```

Indique la texture du vaisseau à utiliser en jeu

#### 3.2.2.6 stars

```
unsigned int playerinfo_s::stars
```

Nombre d'étoiles du joueur

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

- [logique.h](#)

### 3.3 resources\_s Struct Reference

Stockage des ressources nécessaires à l'affichage graphique.

```
#include <graphique.h>
```

#### Public Attributes

- SDL\_Texture \* [background\\_menu](#)
- SDL\_Texture \* [background](#)
- SDL\_Texture \* [ship](#)
- SDL\_Texture \* [ship1](#)
- SDL\_Texture \* [ship2](#)
- SDL\_Texture \* [ship3](#)
- SDL\_Texture \* [ship4](#)
- SDL\_Texture \* [finish\\_line](#)
- SDL\_Texture \* [meteorite](#)
- TTF\_Font \* [font](#)
- TTF\_Font \* [font\\_menu](#)
- SDL\_Texture \* [star](#)
- SDL\_Texture \* [return\\_menu](#)
- SDL\_Texture \* [select](#)
- SDL\_Texture \* [selected](#)
- SDL\_Texture \* [shield](#)
- SDL\_Texture \* [time](#)
- SDL\_Texture \* [missile](#)
- SDL\_Texture \* [enemy\\_ship](#)

#### 3.3.1 Detailed Description

Stockage des ressources nécessaires à l'affichage graphique.

#### 3.3.2 Member Data Documentation

##### 3.3.2.1 background

```
SDL_Texture* resources_s::background
```

ressource liée à l'image du fond de l'écran de jeu.

##### 3.3.2.2 background\_menu

```
SDL_Texture* resources_s::background_menu
```

ressource liée à l'image du fond du menu.



### 3.3.2.3 enemy\_ship

```
SDL_Texture* resources_s::enemy_ship
```

ressource liée à l'image d'un vaisseau ennemi

### 3.3.2.4 finish\_line

```
SDL_Texture* resources_s::finish_line
```

ressource liée à la ligne d'arrivée

### 3.3.2.5 font

```
TTF_Font* resources_s::font
```

police d'affichage du jeu

### 3.3.2.6 font\_menu

```
TTF_Font* resources_s::font_menu
```

police d'affichage du menu

### 3.3.2.7 meteorite

```
SDL_Texture* resources_s::meteorite
```

ressource liée à une météorite

### 3.3.2.8 missile

```
SDL_Texture* resources_s::missile
```

ressource liée à l'image d'un missile

### 3.3.2.9 return\_menu

```
SDL_Texture* resources_s::return_menu
```

ressource liée au retour dans le menu

### 3.3.2.10 select

```
SDL_Texture* resources_s::select
```

ressource liée à la sélection dans le menu

#### 3.3.2.11 selected

```
SDL_Texture* resources_s::selected
```

ressource liée au vaisseau selectionné dans le shop

#### 3.3.2.12 shield

```
SDL_Texture* resources_s::shield
```

ressource liée à l'image d'un bouclier

#### 3.3.2.13 ship

```
SDL_Texture* resources_s::ship
```

ressource liée à l'image du vaisseau de base.

#### 3.3.2.14 ship1

```
SDL_Texture* resources_s::ship1
```

ressource liée à l'image du vaisseau 1.

#### 3.3.2.15 ship2

```
SDL_Texture* resources_s::ship2
```

ressource liée à l'image du vaisseau 2.

#### 3.3.2.16 ship3

```
SDL_Texture* resources_s::ship3
```

ressource liée à l'image du vaisseau 3.

#### 3.3.2.17 ship4

```
SDL_Texture* resources_s::ship4
```

ressource liée à l'image du vaisseau 4.

#### 3.3.2.18 star

```
SDL_Texture* resources_s::star
```

ressource liée à une étoile

### 3.3.2.19 `time`

```
SDL_Texture* resources_s::time
```

ressource liée à l'image d'un symbole de temps

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

- [graphique.h](#)

## 3.4 `sprite_s` Struct Reference

Représentation d'un sprite du jeu.

```
#include <logique.h>
```

### Public Attributes

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

### 3.4.1 Detailed Description

Représentation d'un sprite du jeu.

### 3.4.2 Member Data Documentation

#### 3.4.2.1 `h`

```
int sprite_s::h
```

Champ indiquant la hauteur

#### 3.4.2.2 `w`

```
int sprite_s::w
```

Champ indiquant la largeur

### 3.4.2.3 x

```
int sprite_s::x
```

Champ indiquant l'abscisse de la position

### 3.4.2.4 y

```
int sprite_s::y
```

Champ indiquant l'ordonnée de la position

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

- [logique.h](#)

## 3.5 world\_s Struct Reference

Représentation du monde du jeu.

```
#include <logique.h>
```

Collaboration diagram for world\_s:

### Public Attributes

- [sprite\\_t](#) ship
- [sprite\\_t](#) finish\_line
- int **vy**
- int [gameover](#)
- int [collision](#)
- int [collision\\_finish\\_line](#)
- [sprite\\_t](#) meteorite [NB\_WALLS]
- [sprite\\_t](#) bonus [NB\_BONUS]
- unsigned int [bonus\\_type](#) [NB\_BONUS]
- unsigned int [is\\_bonus\\_available](#) [NB\_BONUS]
- unsigned char [hasShield](#)
- [sprite\\_t](#) missile [MISSILE\_MAX]
- unsigned char [is\\_missile\\_free](#) [MISSILE\_MAX]
- [sprite\\_t](#) ennemy [ENNEMIES\_MAX]
- unsigned char [is\\_ennemy\\_dead](#) [MISSILE\_MAX]

### 3.5.1 Detailed Description

Représentation du monde du jeu.

## 3.5.2 Member Data Documentation

### 3.5.2.1 bonus

```
sprite_t world_s::bonus[NB_BONUS]
```

Champ représentant un tableau de sprite de type bonus

### 3.5.2.2 bonus\_type

```
unsigned int world_s::bonus_type[NB_BONUS]
```

Champ représentant un tableau indiquant le type du bonus associé au même indice

### 3.5.2.3 collision

```
int world_s::collision
```

Champ indiquant si le vaisseau est rentré en colision avec un mur ou un ennemi

### 3.5.2.4 collision\_finish\_line

```
int world_s::collision_finish_line
```

Champ indiquant si le vaisseau est rentré en colision avec la ligne d'arrivée

### 3.5.2.5 enemy

```
sprite_t world_s::enemy[ENNEMIES_MAX]
```

Champ représentant un tableau de sprite de type ennemi

### 3.5.2.6 finish\_line

```
sprite_t world_s::finish_line
```

Champ représentant la ligne d'arrivée

### 3.5.2.7 gameover

```
int world_s::gameover
```

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

### 3.5.2.8 hasShield

```
unsigned char world_s::hasShield
```

Champ si le joueur possède ou non un bouclier

### 3.5.2.9 is\_bonus\_available

```
unsigned int world_s::is_bonus_available[NB_BONUS]
```

Champ représentant un tableau indiquant si le bonus associé au même indice est disponible

### 3.5.2.10 is\_enemy\_dead

```
unsigned char world_s::is_enemy_dead[MISSILE_MAX]
```

Champ représentant un tableau indiquant si l'ennemi associé au même indice est mort ou vivant

### 3.5.2.11 is\_missile\_free

```
unsigned char world_s::is_missile_free[MISSILE_MAX]
```

Champ représentant un tableau indiquant si le missile associé au même indice est disponible

### 3.5.2.12 meteorite

```
sprite_t world_s::meteorite[NB_WALLS]
```

Champ représentant un tableau de murs de météorites

### 3.5.2.13 missile

```
sprite_t world_s::missile[MISSILE_MAX]
```

Champ représentant un tableau de sprite de type missile

### 3.5.2.14 ship

```
sprite_t world_s::ship
```

Champ représentant le sprite vaisseau

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

- [logique.h](#)

## Chapter 4

# File Documentation

### 4.1 definition.h File Reference

Header regroupant toutes les constantes utiles au programme.

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

### 4.2 graphique.c File Reference

Module gérant la partie graphique du jeu.

```
#include "sdl2-light.h"
#include "sdl2-ttf-light.h"
#include "transition.h"
#include "graphique.h"
#include "logique.h"
#include "definition.h"
#include "menu.h"
#include <stdlib.h>
#include <stdio.h>
```

Include dependency graph for graphique.c:

### Functions

- void `clean_resources` (`resources_t` \*resources)  
*La fonction nettoie les ressources.*
- void `init_resources` (SDL\_Renderer \*renderer, `resources_t` \*resources)  
*La fonction initialise les ressources nécessaires à l'affichage graphique du jeu.*
- void `apply_sprite` (SDL\_Renderer \*renderer, SDL\_Texture \*resource, `sprite_t` \*sprite, unsigned int make\_↔ disappear)  
*La fonction positionne la texture à la position du sprite.*
- void `apply_background` (SDL\_Renderer \*renderer, SDL\_Texture \*resource)  
*La fonction applique la ressource du fond.*
- void `apply_walls` (SDL\_Renderer \*renderer, `world_t` \*world, `resources_t` \*resources)  
*La fonction applique la ressource des couloirs de météorites.*

- void `apply_missiles` (SDL\_Renderer \*renderer, `world_t` \*world, `resources_t` \*resources)  
*La fonction applique la ressource aux différents missiles.*
- void `apply_enemies` (SDL\_Renderer \*renderer, `world_t` \*world, `resources_t` \*resources)  
*La fonction applique la ressource aux différents ennemis.*
- void `apply_bonus` (SDL\_Renderer \*renderer, `world_t` \*world, `resources_t` \*resources)  
*La fonction applique la ressource aux différents bonus.*
- void `apply_texts` (SDL\_Renderer \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game)  
*La fonction affiche les textes à l'écran et gère le temps.*
- void `refresh_graphics` (SDL\_Renderer \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game, `playerinfo_t` \*player)  
*La fonction rafraichit l'écran en fonction de l'état des données du monde.*

## 4.2.1 Detailed Description

Module gérant la partie graphique du jeu.

### Author

LESNIAK Louis & SLIMANI Kamelia

### Version

3.0 + extensions

### Date

23 mai 2021

## 4.2.2 Function Documentation

### 4.2.2.1 `apply_background()`

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

La fonction applique la ressource du fond.

#### Parameters

<code>renderer</code>	le renderer SDL
<code>resource</code>	la ressource liée au fond



#### 4.2.2.2 apply\_bonus()

```
void apply_bonus (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource aux différents bonus.

##### Parameters

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

#### 4.2.2.3 apply\_enemies()

```
void apply_enemies (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource aux différents ennemis.

##### Parameters

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

#### 4.2.2.4 apply\_missiles()

```
void apply_missiles (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource aux différents missiles.

##### Parameters

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

#### 4.2.2.5 apply\_sprite()

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

La fonction positionne la texture à la position du sprite.

##### Parameters

<i>renderer</i>	le renderer SDL
<i>resource</i>	les ressources du jeu
<i>sprite</i>	sprite à afficher
<i>make_disappear</i>	affiche ou non la texture

#### 4.2.2.6 apply\_texts()

```
void apply_texts (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game )
```

La fonction affiche les textes à l'écran et gère le temps.

##### Parameters

<i>renderer</i>	le renderer SDL
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu

#### 4.2.2.7 apply\_walls()

```
void apply_walls (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource des couloirs de météorites.

## Parameters

<i>renderer</i>	le renderer SDL
<i>world</i>	les données du monde
<i>resources</i>	les ressources

**4.2.2.8 clean\_resources()**

```
void clean_resources (
    resources_t * resources )
```

La fonction nettoie les ressources.

## Parameters

<i>resources</i>	les ressources
------------------	----------------

**4.2.2.9 init\_resources()**

```
void init_resources (
    SDL_Renderer * renderer,
    resources_t * resources )
```

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

## Parameters

<i>renderer</i>	le renderer SDL
<i>resources</i>	les ressources du jeu

**4.2.2.10 refresh\_graphics()**

```
void refresh_graphics (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game,
    playerinfo_t * player )
```

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

## Parameters

<i>renderer</i>	renderer SDL
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	données du jeu
<i>player</i>	données du joueur

## 4.3 graphique.h File Reference

Header du module graphique.

```
#include "logique.h"
```

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

### Classes

- struct [resources\\_s](#)  
*Stockage des ressources nécessaires à l'affichage graphique.*

### Typedefs

- typedef struct [resources\\_s](#) [resources\\_t](#)  
*Type qui correspond aux ressources du jeu.*

### Functions

- void [clean\\_resources](#) ([resources\\_t](#) \*resources)  
*La fonction nettoie les ressources.*
- void [init\\_resources](#) (SDL\_Renderer \*renderer, [resources\\_t](#) \*resources)  
*La fonction initialise les ressources nécessaires à l'affichage graphique du jeu.*
- void [apply\\_sprite](#) (SDL\_Renderer \*renderer, SDL\_Texture \*resource, [sprite\\_t](#) \*sprite, unsigned int make\_↵ disappear)  
*La fonction positionne la texture à la position du sprite.*
- void [apply\\_background](#) (SDL\_Renderer \*renderer, SDL\_Texture \*resource)  
*La fonction applique la ressource du fond.*
- void [apply\\_walls](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources)  
*La fonction applique la ressource des couloirs de météorites.*
- void [apply\\_missiles](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources)  
*La fonction applique la ressource aux différents missiles.*
- void [apply\\_enemies](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources)  
*La fonction applique la ressource aux différents ennemis.*
- void [apply\\_bonus](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources)  
*La fonction applique la ressource aux différents bonus.*
- void [apply\\_texts](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game)  
*La fonction affiche les textes à l'écran et gère le temps.*
- void [refresh\\_graphics](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction rafraichit l'écran en fonction de l'état des données du monde.*

### 4.3.1 Detailed Description

Header du module graphique.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.3.2 Function Documentation

#### 4.3.2.1 apply\_background()

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

La fonction applique la ressource du fond.

##### Parameters

<i>renderer</i>	le renderer SDL
<i>resource</i>	la ressource liée au fond

#### 4.3.2.2 apply\_bonus()

```
void apply_bonus (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource aux différents bonus.

##### Parameters

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

#### 4.3.2.3 apply\_enemies()

```
void apply_enemies (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource aux différents ennemis.

##### Parameters

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

#### 4.3.2.4 apply\_missiles()

```
void apply_missiles (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource aux différents missiles.

##### Parameters

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

#### 4.3.2.5 apply\_sprite()

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

La fonction positionne la texture à la position du sprite.

##### Parameters

<i>renderer</i>	le renderer SDL
<i>resource</i>	les ressources du jeu
<i>sprite</i>	sprite à afficher
<i>make_disappear</i>	affiche ou non la texture

#### 4.3.2.6 apply\_texts()

```
void apply_texts (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game )
```

La fonction affiche les textes à l'écran et gère le temps.

##### Parameters

<i>renderer</i>	le renderer SDL
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu

#### 4.3.2.7 apply\_walls()

```
void apply_walls (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources )
```

La fonction applique la ressource des couloirs de météorites.

##### Parameters

<i>renderer</i>	le renderer SDL
<i>world</i>	les données du monde
<i>resources</i>	les ressources

#### 4.3.2.8 clean\_resources()

```
void clean_resources (
    resources_t * resources )
```

La fonction nettoie les ressources.

##### Parameters

<i>resources</i>	les ressources
------------------	----------------

#### 4.3.2.9 init\_resources()

```
void init_resources (
    SDL_Renderer * renderer,
    resources_t * resources )
```

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

##### Parameters

<i>renderer</i>	le renderer SDL
<i>resources</i>	les ressources du jeu

#### 4.3.2.10 refresh\_graphics()

```
void refresh_graphics (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game,
    playerinfo_t * player )
```

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

##### Parameters

<i>renderer</i>	renderer SDL
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	données du jeu
<i>player</i>	données du joueur

## 4.4 logique.c File Reference

Module gérant la partie logique du jeu.

```
#include "logique.h"
#include "definition.h"
#include <stdio.h>
#include <stdlib.h>
Include dependency graph for logique.c:
```



## Functions

- void `init_sprite` (`sprite_t` \*sprite, int x, int y, int w, int h)  
*La fonction initialise la position du sprite.*
- void `init_walls` (`world_t` \*world)  
*La fonction qui génère aléatoirement la position des murs de météorites.*
- void `init_bonus` (`world_t` \*world)  
*La fonction qui génère aléatoirement la position des bonus.*
- void `init_enemies` (`world_t` \*world)  
*La fonction qui génère aléatoirement la position des ennemis entre les murs.*
- void `init_missiles` (`world_t` \*world)  
*La fonction qui rend disponible tous les missiles.*
- void `init_data` (`world_t` \*world)  
*La fonction initialise les données du monde du jeu.*
- void `print_sprite` (`sprite_t` sprite)  
*La fonction affiche la position du sprite.*
- int `is_game_over` (`world_t` \*world, `gameinfo_t` \*game)  
*La fonction indique si le jeu est fini en fonction des données du monde.*
- void `update_data` (`world_t` \*world, `gameinfo_t` \*game, `playerinfo_t` \*player)  
*La fonction met à jour les données en tenant compte de la physique du monde.*
- void `update_sprites` (`world_t` \*world, `playerinfo_t` \*player, `gameinfo_t` \*game)  
*La fonction met à jour les données des sprites (sauf le vaisseau)*
- void `depassement_g` (`sprite_t` \*sprite)  
*fonction qui verifie que le sprite ne depaase pas la limite gauche du jeu*
- void `depassement_d` (`sprite_t` \*sprite)  
*fonction qui verifie que le sprite ne depaase pas la limite droite du jeu*
- void `shoot` (`world_t` \*world)  
*fonction qui initialise un missile devant le vaisseau si il y en a un qui est disponible*
- int `abs` (int a)  
*fonction qui retourne la valeur absolue d'un entier*
- int `sprites_collide` (`sprite_t` \*sp1, `sprite_t` \*sp2)  
*fonction qui verifie que 2 sprite ne soit pas en collision*
- unsigned int `retrieve_index` (`sprite_t` \*sp1, `sprite_t` \*Array, unsigned int limit)  
*La fonction retrouve l'index d'un élément d'une array à partir de l'adresse de son élément.*
- void `handle_sprites_collision` (`sprite_t` \*sp1, `sprite_t` \*sp2, `world_t` \*world, `playerinfo_t` \*player, `gameinfo_t` \*game, int sprite\_type)  
*fonction qui verifie que 2 sprite ne soit pas en collision*

### 4.4.1 Detailed Description

Module gérant la partie logique du jeu.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

## 4.4.2 Function Documentation

### 4.4.2.1 abs()

```
int abs (  
    int a )
```

fonction qui retourne la valeur absolue d'un entier

#### Parameters

<i>a</i>	int
----------	-----

#### Returns

$|a|$  : -a si  $a < 0$  ou a si  $a > 0$

### 4.4.2.2 depassement\_d()

```
void depassement_d (  
    sprite_t * sprite )
```

fonction qui verifie que le sprite ne depaase pas la limite droite du jeu

#### Parameters

<i>sprite</i>	vaisseau
---------------	----------

### 4.4.2.3 depassement\_g()

```
void depassement_g (  
    sprite_t * sprite )
```

fonction qui verifie que le sprite ne depaase pas la limite gauche du jeu

#### Parameters

<i>sprite</i>	vaisseau
---------------	----------

#### 4.4.2.4 handle\_sprites\_collision()

```
void handle_sprites_collision (
    sprite_t * sp1,
    sprite_t * sp2,
    world_t * world,
    playerinfo_t * player,
    gameinfo_t * game,
    int make_disappear )
```

fonction qui verifie que 2 sprite ne soit pas en collision

##### Parameters

<i>sp1</i>	vaisseau
<i>sp2</i>	sprite (mur ou ligne arrivé)
<i>world</i>	monde
<i>player</i>	les données du joueur
<i>game</i>	les données du jeu
<i>make_disappear</i>	sprite visible/invisible

#### 4.4.2.5 init\_bonus()

```
void init_bonus (
    world_t * world )
```

La fonction qui génère aléatoirement la position des bonus.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.4.2.6 init\_data()

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

La fonction initialise les données du monde du jeu.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.4.2.7 init\_enemies()

```
void init_enemies (
    world_t * world )
```

La fonction qui génère aléatoirement la position des ennemis entre les murs.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.4.2.8 init\_missiles()

```
void init_missiles (
    world_t * world )
```

La fonction qui rend disponible tous les missiles.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.4.2.9 init\_sprite()

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

La fonction initialise la position du sprite.

##### Parameters

<i>sprite</i>	vaisseau
<i>x</i>	abscisse de la position du sprite
<i>y</i>	ordonnée de la position du sprite
<i>w</i>	largeur de la position du sprite
<i>h</i>	hauteur de la position du sprite

#### 4.4.2.10 init\_walls()

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

La fonction qui génère aléatoirement la position des murs de météorites.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.4.2.11 is\_game\_over()

```
int is_game_over (
    world_t * world,
    gameinfo_t * game )
```

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

##### Parameters

<i>world</i>	les données du monde
<i>game</i>	les données du jeu

##### Returns

1 si le jeu est fini, 0 sinon

#### 4.4.2.12 print\_sprite()

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

La fonction affiche la position du sprite.

##### Parameters

<i>sprite</i>	vaisseau
---------------	----------

#### 4.4.2.13 retrieve\_index()

```
unsigned int retrieve_index (
    sprite_t * spl,
```

```
sprite_t * Array,  
unsigned int limit )
```

La fonction retrouve l'index d'un élément d'une array à partir de l'adresse de son élément.

#### Parameters

<i>sp1</i>	adresse d'un sprite
<i>Array</i>	adresse d'un sprite Array
<i>limit</i>	nombre d'index de l'array pour ne pas faire de dépassements

#### Returns

l'index (unsigned int)

#### 4.4.2.14 shoot()

```
void shoot (  
    world_t * world )
```

fonction qui initialise un missile devant le vaisseau si il y en a un qui est disponible

#### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.4.2.15 sprites\_collide()

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

fonction qui verifie que 2 sprite ne soit pas en collision

#### Parameters

<i>sp1</i>	vaisseau
<i>sp2</i>	sprite (mur ou ligne arrivé)

#### Returns

1 si collision 0 sinon

#### 4.4.2.16 update\_data()

```
void update_data (
    world_t * world,
    gameinfo_t * game,
    playerinfo_t * player )
```

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

##### Parameters

<i>world</i>	les données du monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.4.2.17 update\_sprites()

```
void update_sprites (
    world_t * world,
    playerinfo_t * player,
    gameinfo_t * game )
```

La fonction met à jour les données des sprites (sauf le vaisseau)

##### Parameters

<i>world</i>	les données du monde
<i>player</i>	les données du joueur
<i>game</i>	les données du jeu

## 4.5 logique.h File Reference

Header du module logique.

```
#include "definition.h"
```

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

### Classes

- struct [sprite\\_s](#)  
*Représentation d'un sprite du jeu.*
- struct [gameinfo\\_s](#)  
*Stockage des informations du jeu en cours (extension)*
- struct [playerinfo\\_s](#)  
*Stockage des informations du joueur(extension)*
- struct [world\\_s](#)  
*Représentation du monde du jeu.*

## Typedefs

- typedef struct [sprite\\_s](#) [sprite\\_t](#)  
*Type qui correspond aux données du sprite.*
- typedef struct [gameinfo\\_s](#) [gameinfo\\_t](#)  
*Type qui correspond aux données du jeu.*
- typedef struct [playerinfo\\_s](#) [playerinfo\\_t](#)  
*Type qui correspond aux données du joueur.*
- typedef struct [world\\_s](#) [world\\_t](#)  
*Type qui correspond aux données du monde.*

## Functions

- void [init\\_sprite](#) ([sprite\\_t](#) \*sprite, int x, int y, int w, int h)  
*La fonction initialise la position du sprite.*
- void [init\\_walls](#) ([world\\_t](#) \*world)  
*La fonction qui génère aléatoirement la position des murs de météorites.*
- void [init\\_bonus](#) ([world\\_t](#) \*world)  
*La fonction qui génère aléatoirement la position des bonus.*
- void [init\\_enemies](#) ([world\\_t](#) \*world)  
*La fonction qui génère aléatoirement la position des ennemis entre les murs.*
- void [init\\_missiles](#) ([world\\_t](#) \*world)  
*La fonction qui rend disponible tous les missiles.*
- void [init\\_data](#) ([world\\_t](#) \*world)  
*La fonction initialise les données du monde du jeu.*
- void [print\\_sprite](#) ([sprite\\_t](#) sprite)  
*La fonction affiche la position du sprite.*
- int [is\\_game\\_over](#) ([world\\_t](#) \*world, [gameinfo\\_t](#) \*game)  
*La fonction indique si le jeu est fini en fonction des données du monde.*
- void [update\\_data](#) ([world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction met à jour les données en tenant compte de la physique du monde.*
- void [update\\_sprites](#) ([world\\_t](#) \*world, [playerinfo\\_t](#) \*player, [gameinfo\\_t](#) \*game)  
*La fonction met à jour les données des sprites (sauf le vaisseau)*
- void [depassement\\_g](#) ([sprite\\_t](#) \*sprite)  
*fonction qui verifie que le sprite ne depaase pas la limite gauche du jeu*
- void [depassement\\_d](#) ([sprite\\_t](#) \*sprite)  
*fonction qui verifie que le sprite ne depaase pas la limite droite du jeu*
- void [shoot](#) ([world\\_t](#) \*world)  
*fonction qui initialise un missile devant le vaisseau si il y en a un qui est disponible*
- int [abs](#) (int a)  
*fonction qui retourne la valeur absolue d'un entier*
- int [sprites\\_collide](#) ([sprite\\_t](#) \*sp1, [sprite\\_t](#) \*sp2)  
*fonction qui verifie que 2 sprite ne soit pas en collision*
- unsigned int [retreive\\_index](#) ([sprite\\_t](#) \*sp1, [sprite\\_t](#) \*Array, unsigned int limit)  
*La fonction retrouve l'index d'un élément d'une array à partir de l'adresse de son élément.*
- void [handle\\_sprites\\_collision](#) ([sprite\\_t](#) \*sp1, [sprite\\_t](#) \*sp2, [world\\_t](#) \*world, [playerinfo\\_t](#) \*player, [gameinfo\\_t](#) \*game, int make\_disappear)  
*fonction qui verifie que 2 sprite ne soit pas en collision*



### 4.5.1 Detailed Description

Header du module logique.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.5.2 Function Documentation

#### 4.5.2.1 abs()

```
int abs (  
    int a )
```

fonction qui retourne la valeur absolue d'un entier

#### Parameters

<i>a</i>	int
----------	-----

#### Returns

$|a|$  : -a si  $a < 0$  ou a si  $a > 0$

#### 4.5.2.2 depassement\_d()

```
void depassement_d (  
    sprite_t * sprite )
```

fonction qui verifie que le sprite ne depaase pas la limite droite du jeu

#### Parameters

<i>sprite</i>	vaisseau
---------------	----------

#### 4.5.2.3 depassement\_g()

```
void depassement_g (
    sprite_t * sprite )
```

fonction qui verifie que le sprite ne depaase pas la limite gauche du jeu

##### Parameters

<i>sprite</i>	vaisseau
---------------	----------

#### 4.5.2.4 handle\_sprites\_collision()

```
void handle_sprites_collision (
    sprite_t * sp1,
    sprite_t * sp2,
    world_t * world,
    playerinfo_t * player,
    gameinfo_t * game,
    int make_disappear )
```

fonction qui verifie que 2 sprite ne soit pas en collision

##### Parameters

<i>sp1</i>	vaisseau
<i>sp2</i>	sprite (mur ou ligne arrivé)
<i>world</i>	monde
<i>player</i>	les données du joueur
<i>game</i>	les données du jeu
<i>make_disappear</i>	sprite visible/invisible

#### 4.5.2.5 init\_bonus()

```
void init_bonus (
    world_t * world )
```

La fonction qui génère aléatoirement la position des bonus.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.5.2.6 init\_data()

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

La fonction initialise les données du monde du jeu.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.5.2.7 init\_enemies()

```
void init_enemies (
    world_t * world )
```

La fonction qui génère aléatoirement la position des ennemis entre les murs.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.5.2.8 init\_missiles()

```
void init_missiles (
    world_t * world )
```

La fonction qui rend disponible tous les missiles.

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.5.2.9 init\_sprite()

```
void init_sprite (
    sprite_t * sprite,
    int x,
```

```
int y,  
int w,  
int h )
```

La fonction initialise la position du sprite.

#### Parameters

<i>sprite</i>	vaisseau
<i>x</i>	abscisse de la position du sprite
<i>y</i>	ordonnée de la position du sprite
<i>w</i>	largeur de la position du sprite
<i>h</i>	hauteur de la position du sprite

#### 4.5.2.10 init\_walls()

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

La fonction qui génère aléatoirement la position des murs de météorites.

#### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.5.2.11 is\_game\_over()

```
int is_game_over (  
    world_t * world,  
    gameinfo_t * game )
```

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

#### Parameters

<i>world</i>	les données du monde
<i>game</i>	les données du jeu

#### Returns

1 si le jeu est fini, 0 sinon

#### 4.5.2.12 print\_sprite()

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

La fonction affiche la position du sprite.

##### Parameters

<i>sprite</i>	vaisseau
---------------	----------

#### 4.5.2.13 retrieve\_index()

```
unsigned int retrieve_index (
    sprite_t * sp1,
    sprite_t * Array,
    unsigned int limit )
```

La fonction retrouve l'index d'un élément d'une array à partir de l'adresse de son élément.

##### Parameters

<i>sp1</i>	adresse d'un sprite
<i>Array</i>	adresse d'un sprite Array
<i>limit</i>	nombre d'index de l'array pour ne pas faire de dépassements

##### Returns

l'index (unsigned int)

#### 4.5.2.14 shoot()

```
void shoot (
    world_t * world )
```

fonction qui initialise un missile devant le vaisseau si il y en a un qui est disponible

##### Parameters

<i>world</i>	les données du monde
--------------	----------------------

#### 4.5.2.15 sprites\_collide()

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

fonction qui verifie que 2 sprite ne soit pas en collision

##### Parameters

<i>sp1</i>	vaisseau
<i>sp2</i>	sprite (mur ou ligne arrivé)

##### Returns

1 si collision 0 sinon

#### 4.5.2.16 update\_data()

```
void update_data (
    world_t * world,
    gameinfo_t * game,
    playerinfo_t * player )
```

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

##### Parameters

<i>world</i>	les données du monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.5.2.17 update\_sprites()

```
void update_sprites (
    world_t * world,
    playerinfo_t * player,
    gameinfo_t * game )
```

La fonction met à jour les données des sprites (sauf le vaisseau)

##### Parameters

<i>world</i>	les données du monde
<i>player</i>	les données du joueur
<i>game</i>	les données du jeu

## 4.6 main.c File Reference

Programme principal initial du niveau 3 + extensions.

```
#include "sdl2-light.h"
#include "sdl2-ttf-light.h"
#include "logique.h"
#include "graphique.h"
#include "transition.h"
Include dependency graph for main.c:
```

### Functions

- `int main (int argc, char *args[])`  
*programme principal qui implémente la boucle du jeu*

### 4.6.1 Detailed Description

Programme principal initial du niveau 3 + extensions.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

## 4.7 menu.c File Reference

Module gérant le fonctionnement du menu.

```
#include "sdl2-light.h"
#include "sdl2-ttf-light.h"
#include "transition.h"
#include "logique.h"
#include "graphique.h"
#include "definition.h"
#include "menu.h"
#include <stdlib.h>
#include <stdio.h>
Include dependency graph for menu.c:
```

## Functions

- void `buy_select_ship` (`playerinfo_t` \*player, unsigned int ShipNumber)  
*La fonction qui gère l'achat d'une texture de vaisseau.*
- void `print_stars` (`SDL_Renderer` \*renderer, `resources_t` \*resources, `playerinfo_t` \*player, int x)  
*La fonction qui gère l'affichage du nombre d'étoiles du joueur dans le menu.*
- void `print_select` (`SDL_Renderer` \*renderer, `resources_t` \*resources, `gameinfo_t` \*game, int x, int y)  
*La fonction qui gère l'affichage de la sélection dans le menu.*
- void `menu_main` (`SDL_Renderer` \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game, `playerinfo_t` \*player)  
*La fonction affiche le menu principal.*
- void `menu_shop` (`SDL_Renderer` \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game, `playerinfo_t` \*player)  
*La fonction affiche le magasin.*
- void `menu_infos` (`SDL_Renderer` \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game)  
*La fonction affiche les règles du jeu.*
- void `menu_credits` (`SDL_Renderer` \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game)  
*La fonction affiche les crédits du jeu.*
- void `apply_menu` (`SDL_Renderer` \*renderer, `world_t` \*world, `resources_t` \*resources, `gameinfo_t` \*game, `playerinfo_t` \*player)  
*La fonction gère l'appel des fonctions relatives au menu.*

### 4.7.1 Detailed Description

Module gérant le fonctionnement du menu.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.7.2 Function Documentation

#### 4.7.2.1 `apply_menu()`

```
void apply_menu (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game,
    playerinfo_t * player )
```

La fonction gère l'appel des fonctions relatives au menu.



## Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.7.2.2 buy\_select\_ship()

```
void buy_select_ship (
    playerinfo_t * player,
    unsigned int ShipNumber )
```

La fonction qui gère l'achat d'une texture de vaisseau.

## Parameters

<i>player</i>	les données du joueur
<i>ShipNumber</i>	le numéro du vaisseau à acheter

#### 4.7.2.3 menu\_credits()

```
void menu_credits (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game )
```

La fonction affiche les crédits du jeu.

## Parameters

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

#### 4.7.2.4 menu\_infos()

```
void menu_infos (
    SDL_Renderer * renderer,
```

```
world_t * world,  
resources_t * resources,  
gameinfo_t * game )
```

La fonction affiche les règles du jeu.

#### Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.7.2.5 menu\_main()

```
void menu_main (  
    SDL_Renderer * renderer,  
    world_t * world,  
    resources_t * resources,  
    gameinfo_t * game,  
    playerinfo_t * player )
```

La fonction affiche le menu principal.

#### Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.7.2.6 menu\_shop()

```
void menu_shop (  
    SDL_Renderer * renderer,  
    world_t * world,  
    resources_t * resources,  
    gameinfo_t * game,  
    playerinfo_t * player )
```

La fonction affiche le magasin.

#### Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
-----------------	----------------------------------

## Parameters

<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu

#### 4.7.2.7 print\_select()

```
void print_select (
    SDL_Renderer * renderer,
    resources_t * resources,
    gameinfo_t * game,
    int x,
    int y )
```

La fonction qui gère l'affichage de la sélection dans le menu.

## Parameters

<i>renderer</i>	le renderer SDL
<i>resources</i>	les ressources de l'application
<i>game</i>	les données du jeu
<i>x</i>	position x du début de l'affichage
<i>y</i>	position y du début de l'affichage

#### 4.7.2.8 print\_stars()

```
void print_stars (
    SDL_Renderer * renderer,
    resources_t * resources,
    playerinfo_t * player,
    int x )
```

La fonction qui gère l'affichage du nombre d'étoiles du joueur dans le menu.

## Parameters

<i>renderer</i>	le renderer SDL
<i>resources</i>	les ressources de l'application
<i>player</i>	les données du joueur
<i>x</i>	position du début de l'affichage

## 4.8 menu.h File Reference

Header regroupant toutes les fonctions appelant des fonctions graphique et logique.

```
#include "logique.h"
#include "graphique.h"
```

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

### Functions

- void [buy\\_select\\_ship](#) ([playerinfo\\_t](#) \*player, unsigned int ShipNumber)  
*La fonction qui gère l'achat d'une texture de vaisseau.*
- void [print\\_stars](#) (SDL\_Renderer \*renderer, [resources\\_t](#) \*resources, [playerinfo\\_t](#) \*player, int x)  
*La fonction qui gère l'affichage du nombre d'étoiles du joueur dans le menu.*
- void [print\\_select](#) (SDL\_Renderer \*renderer, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game, int x, int y)  
*La fonction qui gère l'affichage de la sélection dans le menu.*
- void [menu\\_main](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction affiche le menu principal.*
- void [menu\\_shop](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction affiche le magasin.*
- void [menu\\_infos](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game)  
*La fonction affiche les règles du jeu.*
- void [menu\\_credits](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game)  
*La fonction affiche les crédits du jeu.*
- void [apply\\_menu](#) (SDL\_Renderer \*renderer, [world\\_t](#) \*world, [resources\\_t](#) \*resources, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction gère l'appel des fonctions relatives au menu.*

### 4.8.1 Detailed Description

Header regroupant toutes les fonctions appelant des fonctions graphique et logique.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.8.2 Function Documentation

#### 4.8.2.1 apply\_menu()

```
void apply_menu (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game,
    playerinfo_t * player )
```

La fonction gère l'appel des fonctions relatives au menu.

##### Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.8.2.2 buy\_select\_ship()

```
void buy_select_ship (
    playerinfo_t * player,
    unsigned int ShipNumber )
```

La fonction qui gère l'achat d'une texture de vaisseau.

##### Parameters

<i>player</i>	les données du joueur
<i>ShipNumber</i>	le numéro du vaisseau à acheter

#### 4.8.2.3 menu\_credits()

```
void menu_credits (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game )
```

La fonction affiche les crédits du jeu.

##### Parameters

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

#### 4.8.2.4 menu\_infos()

```
void menu_infos (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game )
```

La fonction affiche les règles du jeu.

##### Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.8.2.5 menu\_main()

```
void menu_main (
    SDL_Renderer * renderer,
    world_t * world,
    resources_t * resources,
    gameinfo_t * game,
    playerinfo_t * player )
```

La fonction affiche le menu principal.

##### Parameters

<i>renderer</i>	le renderer lié à l'écran de jeu
<i>world</i>	les données du monde
<i>resources</i>	les ressources
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.8.2.6 menu\_shop()

```
void menu_shop (
    SDL_Renderer * renderer,
    world_t * world,
```

```
resources_t * resources,  
gameinfo_t * game,  
playerinfo_t * player )
```

La fonction affiche le magasin.

#### Parameters

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

#### 4.8.2.7 print\_select()

```
void print_select (  
    SDL_Renderer * renderer,  
    resources_t * resources,  
    gameinfo_t * game,  
    int x,  
    int y )
```

La fonction qui gère l'affichage de la sélection dans le menu.

#### Parameters

<i>renderer</i>	le renderer SDL
<i>resources</i>	les ressources de l'application
<i>game</i>	les données du jeu
<i>x</i>	position x du début de l'affichage
<i>y</i>	position y du début de l'affichage

#### 4.8.2.8 print\_stars()

```
void print_stars (  
    SDL_Renderer * renderer,  
    resources_t * resources,  
    playerinfo_t * player,  
    int x )
```

La fonction qui gère l'affichage du nombre d'étoiles du joueur dans le menu.

#### Parameters

<i>renderer</i>	le renderer SDL
<i>resources</i>	les ressources de l'application
<i>player</i>	les données du joueur
<i>x</i>	position du début de l'affichage

## 4.9 sdl2-light.c File Reference

sur-couche de SDL2 pour simplifier son utilisation pour le projet (modifié)

```
#include "sdl2-light.h"
#include <stdio.h>
#include <stdlib.h>
Include dependency graph for sdl2-light.c:
```

### Functions

- int [init\\_sdl](#) (SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, int width, int height)  
*La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.*
- SDL\_Texture \* [load\\_image](#) (const char path[], SDL\_Renderer \*renderer)  
*La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.*
- void [apply\\_texture](#) (SDL\_Texture \*texture, SDL\_Renderer \*renderer, int x, int y)  
*La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.*
- void [clean\\_texture](#) (SDL\_Texture \*texture)  
*La fonction nettoie une texture en mémoire.*
- void [clear\\_renderer](#) (SDL\_Renderer \*renderer)  
*La fonction vide le contenu graphique du renderer lié à l'écran de jeu.*
- void [update\\_screen](#) (SDL\_Renderer \*renderer)  
*La fonction met à jour l'écran avec le contenu du renderer.*
- void [pause](#) (int time)  
*La fonction met le programme en pause pendant un laps de temps.*
- void [clean\\_sdl](#) (SDL\_Renderer \*renderer, SDL\_Window \*window)  
*La fonction nettoie le renderer et la fenêtre du jeu en mémoire.*

### 4.9.1 Detailed Description

sur-couche de SDL2 pour simplifier son utilisation pour le projet (modifié)

#### Author

LESNIAK Louis & SLIMANI Kamelia (modification du document de CONSTANT Mathieu)

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 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

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

#### 4.9.2.2 clean\_sdl()

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

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

##### Parameters

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

#### 4.9.2.3 clean\_texture()

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

La fonction nettoie une texture en mémoire.

##### Parameters

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

#### 4.9.2.4 clear\_renderer()

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

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

##### Parameters

<i>renderer</i>	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

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

##### Returns

-1 en cas d'erreur, 0 sinon

#### 4.9.2.6 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

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

**Returns**

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

**4.9.2.7 pause()**

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

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

**Parameters**

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

**4.9.2.8 update\_screen()**

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

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

**Parameters**

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

**4.10 sdl2-light.h File Reference**

sur-couche de SDL2 pour simplifier son utilisation pour le projet (modifié)

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

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

**Functions**

- void [clean\\_sdl](#) (SDL\_Renderer \*renderer, SDL\_Window \*window)  
*La fonction nettoie le renderer et la fenêtre du jeu en mémoire.*
- SDL\_Texture \* [load\\_image](#) (const char path[], SDL\_Renderer \*renderer)  
*La fonction charge une image et renvoie la texture correspondante où la couleur RGB (255, 0, 255) est rendue transparente.*
- int [init\\_sdl](#) (SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, int width, int height)  
*La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.*

- void `clean_texture` (SDL\_Texture \*texture)  
*La fonction nettoie une texture en mémoire.*
- void `apply_texture` (SDL\_Texture \*texture, SDL\_Renderer \*renderer, int x, int y)  
*La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.*
- void `clear_renderer` (SDL\_Renderer \*renderer)  
*La fonction vide le contenu graphique du renderer lié à l'écran de jeu.*
- void `update_screen` (SDL\_Renderer \*renderer)  
*La fonction met à jour l'écran avec le contenu du renderer.*
- void `pause` (int time)  
*La fonction met le programme en pause pendant un laps de temps.*

### 4.10.1 Detailed Description

sur-couche de SDL2 pour simplifier son utilisation pour le projet (modifié)

#### Author

LESNIAK Louis & SLIMANI Kamelia (modification du document de CONSTANT Mathieu)

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.10.2 Function Documentation

#### 4.10.2.1 `apply_texture()`

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

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

#### Parameters

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

#### 4.10.2.2 clean\_sdl()

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

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

##### Parameters

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

#### 4.10.2.3 clean\_texture()

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

La fonction nettoie une texture en mémoire.

##### Parameters

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

#### 4.10.2.4 clear\_renderer()

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

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

##### Parameters

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

#### 4.10.2.5 init\_sdl()

```
int init_sdl (
    SDL_Window ** window,
```

```
SDL_Renderer ** renderer,  
int width,  
int height )
```

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

#### Parameters

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

#### Returns

-1 en cas d'erreur, 0 sinon

#### 4.10.2.6 load\_image()

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

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

#### Parameters

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

#### Returns

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

#### 4.10.2.7 pause()

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

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

#### Parameters

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

#### 4.10.2.8 update\_screen()

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

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

##### Parameters

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

## 4.11 sdl2-ttf-light.c File Reference

sur-couche de SDL2\_ttf pour simplifier son utilisation pour le projet (modifié)

```
#include "sdl2-ttf-light.h"
Include dependency graph for sdl2-ttf-light.c:
```

### Functions

- void [init\\_ttf](#) ()  
*La fonction initialise l'environnement TTF.*
- TTF\_Font \* [load\\_font](#) (const char \*path, int font\_size)  
*La fonction charge une police.*
- void [apply\\_text](#) (SDL\_Renderer \*renderer, int x, int y, int w, int h, const char \*text, TTF\_Font \*font, int red, int green, int blue)  
*La fonction applique un texte dans une certaine police sur le renderer à une certaine position et avec une certaine dimension.*
- void [clean\\_font](#) (TTF\_Font \*font)  
*La fonction nettoie une police en mémoire.*

#### 4.11.1 Detailed Description

sur-couche de SDL2\_ttf pour simplifier son utilisation pour le projet (modifié)

##### Author

LESNIAK Louis & SLIMANI Kamelia (modification du document de CONSTANT Mathieu)

##### Version

3.0 + extensions

##### Date

23 mai 2021

## 4.11.2 Function Documentation

### 4.11.2.1 `apply_text()`

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

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

#### Parameters

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

### 4.11.2.2 `clean_font()`

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

La fonction nettoie une police en mémoire.

#### Parameters

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



### 4.11.2.3 load\_font()

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

La fonction charge une police.

#### Parameters

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

#### Returns

la police chargée

## 4.12 sdl2-ttf-light.h File Reference

sur-couche de SDL2\_ttf pour simplifier son utilisation pour le projet (modifié)

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

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

### Functions

- void [init\\_ttf](#) ()  
*La fonction initialise l'environnement TTF.*
- TTF\_Font \* [load\\_font](#) (const char \*path, int font\_size)  
*La fonction charge une police.*
- void [apply\\_text](#) (SDL\_Renderer \*renderer, int x, int y, int w, int h, const char \*text, TTF\_Font \*font, int red, int green, int blue)  
*La fonction applique un texte dans une certaine police sur le renderer à une certaine position et avec une certaine dimension.*
- void [clean\\_font](#) (TTF\_Font \*font)  
*La fonction nettoie une police en mémoire.*

### 4.12.1 Detailed Description

sur-couche de SDL2\_ttf pour simplifier son utilisation pour le projet (modifié)

#### Author

LESNIAK Louis & SLIMANI Kamelia (modification du document de CONSTANT Mathieu)

#### Version

3.0 + extensions

#### Date

23 mai 2021

## 4.12.2 Function Documentation

### 4.12.2.1 `apply_text()`

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

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

#### Parameters

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

### 4.12.2.2 `clean_font()`

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

La fonction nettoie une police en mémoire.

#### Parameters

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

### 4.12.2.3 load\_font()

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

La fonction charge une police.

#### Parameters

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

#### Returns

la police chargée

## 4.13 tests.c File Reference

Programme testant la partie logique du jeu.

```
#include "definition.h"
#include "logique.h"
#include <stdio.h>
#include <stdlib.h>
Include dependency graph for tests.c:
```

### Functions

- void **test\_init\_sprite\_param** ([sprite\\_t](#) \*sprite, int x, int y, int w, int h)
- void **test\_init\_sprite** ()
- void **test\_depassement\_g\_param** ([sprite\\_t](#) \*sprite)
- void **test\_depassement\_g** ()
- void **test\_depassement\_d\_param** ([sprite\\_t](#) \*sprite)
- void **test\_depassement\_d** ()
- void **test\_sprites\_collide\_param** ([sprite\\_t](#) \*sprite1, [sprite\\_t](#) \*sprite2)
- void **test\_sprites\_collide** ()
- void **test\_init\_walls\_param** ([world\\_t](#) \*world)
- void **test\_init\_walls** ()
- void **test\_init\_bonus\_param** ([world\\_t](#) \*world)
- void **test\_init\_bonus** ()
- void **test\_init\_enemies\_param** ([world\\_t](#) \*world)
- void **test\_init\_enemies** ()
- void **test\_init\_missiles\_param** ([world\\_t](#) \*world)
- void **test\_init\_missiles** ()
- void **test\_shoot\_param** ([world\\_t](#) \*world)
- void **test\_shoot** ()
- void **test\_update\_sprites\_param** ([world\\_t](#) \*world, [playerinfo\\_t](#) \*player, [gameinfo\\_t](#) \*game)
- void **test\_update\_sprites** ()
- void **test\_handle\_sprites\_collision\_param** ([sprite\\_t](#) sp1, [sprite\\_t](#) sp2, [world\\_t](#) world, [playerinfo\\_t](#) player, [gameinfo\\_t](#) game, int sprite\_type)
- void **test\_handle\_sprites\_collision** ()
- int **main** ()

### 4.13.1 Detailed Description

Programme testant la partie logique du jeu.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

## 4.14 transition.c File Reference

Module gérant la transition entre les modules différents modules.

```
#include "sdl2-light.h"
#include "sdl2-ttf-light.h"
#include "transition.h"
#include "logique.h"
#include "graphique.h"
#include "definition.h"
#include "menu.h"
#include <stdio.h>
```

Include dependency graph for transition.c:

### Functions

- void [clean](#) (SDL\_Window \*window, SDL\_Renderer \*render, [resources\\_t](#) \*resources, [world\\_t](#) \*world, [gameinfo\\_t](#) \*game)  
*fonction qui nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des ressources, nettoyage des données*
- void [init](#) (SDL\_Window \*\*window, SDL\_Renderer \*\*render, [resources\\_t](#) \*resources, [world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*fonction qui initialise le jeu: initialisation de la partie graphique (SDL), chargement des ressources, initialisation des données (du jeu, du monde et du joueur)*
- void [init\\_player](#) ([playerinfo\\_t](#) \*player)  
*La fonction initialise les données joueurs (extension)*
- void [print\\_credits](#) ()  
*Affiche dans la console le fichier texte des crédits.*
- void [handle\\_events](#) (SDL\_Event \*event, [world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction gère les événements ayant eu lieu et qui n'ont pas encore été traités.*
- char \* [concat\\_array\\_playername](#) ([playerinfo\\_t](#) \*player, char \*array, unsigned int arraySize)  
*La fonction concatène une chaîne de caractères avec le nom du joueur.*
- unsigned int [formatted\\_charArray\\_to\\_uint](#) (char \*Array)  
*La fonction transforme un tableau de char (formaté pour) en int.*
- void [save\\_score](#) ([world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction demande à l'utilisateur son nom et créer un fichier sauvegardant le score. (extension)*
- void [save\\_info](#) ([playerinfo\\_t](#) \*player)  
*La fonction sauvegarde les données du joueur (extension)*

### 4.14.1 Detailed Description

Module gérant la transition entre les modules différents modules.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.14.2 Function Documentation

#### 4.14.2.1 clean()

```
void clean (
    SDL_Window * window,
    SDL_Renderer * renderer,
    resources_t * resources,
    world_t * world,
    gameinfo_t * game )
```

fonction qui nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des ressources, nettoyage des données

#### Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>resources</i>	les ressources
<i>world</i>	le monde
<i>game</i>	données du jeu

#### 4.14.2.2 concat\_array\_playername()

```
char* concat_array_playername (
    playerinfo_t * player,
    char * array,
    unsigned int arraySize )
```

La fonction concatène une chaîne de caractères avec le nom du joueur.

## Parameters

<i>player</i>	les données du joueur
<i>array</i>	chaîne de caractères
<i>arraySize</i>	taille de array

## Returns

pointeur vers la chaîne de caractères produite

**4.14.2.3 formatted\_charArray\_to\_uint()**

```
unsigned int formatted_charArray_to_uint (
    char * Array )
```

La fonction transforme un tableau de char (formaté pour) en int.

## Parameters

<i>Array</i>	tableau de char formaté SSXXXXS où les X sont des chiffres et les S représentent le formatage réalisé
--------------	---

**4.14.2.4 handle\_events()**

```
void handle_events (
    SDL_Event * event,
    world_t * world,
    gameinfo_t * game,
    playerinfo_t * player )
```

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

## Parameters

<i>event</i>	paramètre qui contient les événements
<i>world</i>	les données du monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

**4.14.2.5 init()**

```
void init (
    SDL_Window ** window,
```

```
SDL_Renderer ** renderer,  
resources_t * resources,  
world_t * world,  
gameinfo_t * game,  
playerinfo_t * player )
```

fonction qui initialise le jeu: initialisation de la partie graphique (SDL), chargement des ressources, initialisation des données (du jeu, du monde et du joueur)

#### Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>resources</i>	les ressources
<i>world</i>	le monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.14.2.6 init\_player()

```
void init_player (  
    playerinfo_t * player )
```

La fonction initialise les données joueurs (extension)

#### Parameters

<i>player</i>	les données du joueur
---------------	-----------------------

#### 4.14.2.7 save\_info()

```
void save_info (  
    playerinfo_t * player )
```

La fonction sauvegarde les données du joueur (extension)

#### Parameters

<i>player</i>	les données du joueur
---------------	-----------------------

#### 4.14.2.8 save\_score()

```
void save_score (  

```

```
world_t * world,
gameinfo_t * game,
playerinfo_t * player )
```

La fonction demande à l'utilisateur son nom et créer un fichier sauvegardant le score. (extension)

#### Parameters

<i>world</i>	les données du monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

## 4.15 transition.h File Reference

Header regroupant toutes les fonctions appelant des fonctions graphique et logique.

```
#include "logique.h"
#include "graphique.h"
```

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

### Functions

- void [clean](#) (SDL\_Window \*window, SDL\_Renderer \*renderer, [resources\\_t](#) \*resources, [world\\_t](#) \*world, [gameinfo\\_t](#) \*game)  
*fonction qui nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des ressources, nettoyage des données*
- void [init](#) (SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, [resources\\_t](#) \*resources, [world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*fonction qui initialise le jeu: initialisation de la partie graphique (SDL), chargement des ressources, initialisation des données (du jeu, du monde et du joueur)*
- void [init\\_player](#) ([playerinfo\\_t](#) \*player)  
*La fonction initialise les données joueurs (extension)*
- void [print\\_credits](#) ()  
*Affiche dans la console le fichier texte des crédits.*
- void [handle\\_events](#) (SDL\_Event \*event, [world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction gère les événements ayant eu lieu et qui n'ont pas encore été traités.*
- char \* [concat\\_array\\_playername](#) ([playerinfo\\_t](#) \*player, char \*array, unsigned int arraySize)  
*La fonction concatène une chaîne de caractères avec le nom du joueur.*
- unsigned int [formatted\\_charArray\\_to\\_uint](#) (char \*Array)  
*La fonction transforme un tableau de char (formaté pour) en int.*
- void [save\\_score](#) ([world\\_t](#) \*world, [gameinfo\\_t](#) \*game, [playerinfo\\_t](#) \*player)  
*La fonction demande à l'utilisateur son nom et créer un fichier sauvegardant le score. (extension)*
- void [save\\_info](#) ([playerinfo\\_t](#) \*player)  
*La fonction sauvegarde les données du joueur (extension)*



### 4.15.1 Detailed Description

Header regroupant toutes les fonctions appelant des fonctions graphique et logique.

#### Author

LESNIAK Louis & SLIMANI Kamelia

#### Version

3.0 + extensions

#### Date

23 mai 2021

### 4.15.2 Function Documentation

#### 4.15.2.1 clean()

```
void clean (
    SDL_Window * window,
    SDL_Renderer * renderer,
    resources_t * resources,
    world_t * world,
    gameinfo_t * game )
```

fonction qui nettoie le jeu: nettoyage de la partie graphique (SDL), nettoyage des ressources, nettoyage des données

#### Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>resources</i>	les ressources
<i>world</i>	le monde
<i>game</i>	données du jeu

#### 4.15.2.2 concat\_array\_playername()

```
char* concat_array_playername (
    playerinfo_t * player,
    char * array,
    unsigned int arraySize )
```

La fonction concatène une chaîne de caractères avec le nom du joueur.

## Parameters

<i>player</i>	les données du joueur
<i>array</i>	chaîne de caractères
<i>arraySize</i>	taille de array

## Returns

pointeur vers la chaîne de caractères produite

#### 4.15.2.3 formatted\_charArray\_to\_uint()

```
unsigned int formatted_charArray_to_uint (
    char * Array )
```

La fonction transforme un tableau de char (formaté pour) en int.

## Parameters

<i>Array</i>	tableau de char formaté SSXXXXS où les X sont des chiffres et les S représentent le formatage réalisé
--------------	---

#### 4.15.2.4 handle\_events()

```
void handle_events (
    SDL_Event * event,
    world_t * world,
    gameinfo_t * game,
    playerinfo_t * player )
```

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

## Parameters

<i>event</i>	paramètre qui contient les événements
<i>world</i>	les données du monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.15.2.5 init()

```
void init (
    SDL_Window ** window,
```

```
SDL_Renderer ** renderer,  
resources_t * resources,  
world_t * world,  
gameinfo_t * game,  
playerinfo_t * player )
```

fonction qui initialise le jeu: initialisation de la partie graphique (SDL), chargement des ressources, initialisation des données (du jeu, du monde et du joueur)

#### Parameters

<i>window</i>	la fenêtre du jeu
<i>renderer</i>	le renderer
<i>resources</i>	les ressources
<i>world</i>	le monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

#### 4.15.2.6 init\_player()

```
void init_player (  
    playerinfo_t * player )
```

La fonction initialise les données joueurs (extension)

#### Parameters

<i>player</i>	les données du joueur
---------------	-----------------------

#### 4.15.2.7 save\_info()

```
void save_info (  
    playerinfo_t * player )
```

La fonction sauvegarde les données du joueur (extension)

#### Parameters

<i>player</i>	les données du joueur
---------------	-----------------------

#### 4.15.2.8 save\_score()

```
void save_score (  

```

```
world_t * world,  
gameinfo_t * game,  
playerinfo_t * player )
```

La fonction demande à l'utilisateur son nom et créer un fichier sauvegardant le score. (extension)

#### Parameters

<i>world</i>	les données du monde
<i>game</i>	les données du jeu
<i>player</i>	les données du joueur

# Index

- abs
  - logique.c, 26
  - logique.h, 33
- apply\_background
  - graphique.c, 16
  - graphique.h, 21
- apply\_bonus
  - graphique.c, 16
  - graphique.h, 21
- apply\_ennemies
  - graphique.c, 17
  - graphique.h, 22
- apply\_menu
  - menu.c, 40
  - menu.h, 44
- apply\_missiles
  - graphique.c, 17
  - graphique.h, 22
- apply\_sprite
  - graphique.c, 18
  - graphique.h, 22
- apply\_text
  - sdl2-ttf-light.c, 56
  - sdl2-ttf-light.h, 58
- apply\_texts
  - graphique.c, 18
  - graphique.h, 23
- apply\_texture
  - sdl2-light.c, 48
  - sdl2-light.h, 52
- apply\_walls
  - graphique.c, 18
  - graphique.h, 23
- background
  - resources\_s, 8
- background\_menu
  - resources\_s, 8
- bestTime
  - playerinfo\_s, 7
- bonus
  - world\_s, 13
- bonus\_type
  - world\_s, 13
- buy\_select\_ship
  - menu.c, 41
  - menu.h, 45
- clean
  - transition.c, 61
- transition.h, 65
- clean\_font
  - sdl2-ttf-light.c, 56
  - sdl2-ttf-light.h, 58
- clean\_resources
  - graphique.c, 19
  - graphique.h, 23
- clean\_sdl
  - sdl2-light.c, 49
  - sdl2-light.h, 53
- clean\_texture
  - sdl2-light.c, 49
  - sdl2-light.h, 53
- clear\_renderer
  - sdl2-light.c, 49
  - sdl2-light.h, 53
- close
  - gameinfo\_s, 5
- collision
  - world\_s, 13
- collision\_finish\_line
  - world\_s, 13
- concat\_array\_playername
  - transition.c, 61
  - transition.h, 65
- definition.h, 15
- depassement\_d
  - logique.c, 26
  - logique.h, 33
- depassement\_g
  - logique.c, 26
  - logique.h, 34
- enemy
  - world\_s, 13
- enemy\_ship
  - resources\_s, 8
- enter
  - gameinfo\_s, 5
- finish\_line
  - resources\_s, 9
  - world\_s, 13
- finishTime
  - gameinfo\_s, 6
- font
  - resources\_s, 9
- font\_menu
  - resources\_s, 9

- formatted\_charArray\_to\_uint
  - transition.c, 62
  - transition.h, 66
- gameinfo\_s, 5
  - close, 5
  - enter, 5
  - finishTime, 6
  - gamemode, 6
  - select, 6
  - startTime, 6
- gamemode
  - gameinfo\_s, 6
- gameover
  - world\_s, 13
- graphique.c, 15
  - apply\_background, 16
  - apply\_bonus, 16
  - apply\_enemies, 17
  - apply\_missiles, 17
  - apply\_sprite, 18
  - apply\_texts, 18
  - apply\_walls, 18
  - clean\_resources, 19
  - init\_resources, 19
  - refresh\_graphics, 19
- graphique.h, 20
  - apply\_background, 21
  - apply\_bonus, 21
  - apply\_enemies, 22
  - apply\_missiles, 22
  - apply\_sprite, 22
  - apply\_texts, 23
  - apply\_walls, 23
  - clean\_resources, 23
  - init\_resources, 24
  - refresh\_graphics, 24
- h
  - sprite\_s, 11
- handle\_events
  - transition.c, 62
  - transition.h, 66
- handle\_sprites\_collision
  - logique.c, 26
  - logique.h, 34
- hasShield
  - world\_s, 13
- hasShip
  - playerinfo\_s, 7
- init
  - transition.c, 62
  - transition.h, 66
- init\_bonus
  - logique.c, 27
  - logique.h, 34
- init\_data
  - logique.c, 27
  - logique.h, 35
- init\_enemies
  - logique.c, 27
  - logique.h, 35
- init\_missiles
  - logique.c, 28
  - logique.h, 35
- init\_player
  - transition.c, 63
  - transition.h, 67
- init\_resources
  - graphique.c, 19
  - graphique.h, 24
- init\_sdl
  - sdl2-light.c, 50
  - sdl2-light.h, 53
- init\_sprite
  - logique.c, 28
  - logique.h, 35
- init\_walls
  - logique.c, 28
  - logique.h, 36
- is\_bonus\_available
  - world\_s, 14
- is\_enemy\_dead
  - world\_s, 14
- is\_game\_over
  - logique.c, 29
  - logique.h, 36
- is\_missile\_free
  - world\_s, 14
- lastTime
  - playerinfo\_s, 7
- load\_font
  - sdl2-ttf-light.c, 56
  - sdl2-ttf-light.h, 58
- load\_image
  - sdl2-light.c, 50
  - sdl2-light.h, 54
- logique.c, 24
  - abs, 26
  - depassement\_d, 26
  - depassement\_g, 26
  - handle\_sprites\_collision, 26
  - init\_bonus, 27
  - init\_data, 27
  - init\_enemies, 27
  - init\_missiles, 28
  - init\_sprite, 28
  - init\_walls, 28
  - is\_game\_over, 29
  - print\_sprite, 29
  - retrieve\_index, 29
  - shoot, 30
  - sprites\_collide, 30
  - update\_data, 30
  - update\_sprites, 31
- logique.h, 31

- abs, [33](#)
- depassement\_d, [33](#)
- depassement\_g, [34](#)
- handle\_sprites\_collision, [34](#)
- init\_bonus, [34](#)
- init\_data, [35](#)
- init\_enemies, [35](#)
- init\_missiles, [35](#)
- init\_sprite, [35](#)
- init\_walls, [36](#)
- is\_game\_over, [36](#)
- print\_sprite, [36](#)
- retriev\_index, [37](#)
- shoot, [37](#)
- sprites\_collide, [37](#)
- update\_data, [38](#)
- update\_sprites, [38](#)
- main.c, [39](#)
- menu.c, [39](#)
  - apply\_menu, [40](#)
  - buy\_select\_ship, [41](#)
  - menu\_credits, [41](#)
  - menu\_infos, [41](#)
  - menu\_main, [42](#)
  - menu\_shop, [42](#)
  - print\_select, [43](#)
  - print\_stars, [43](#)
- menu.h, [44](#)
  - apply\_menu, [44](#)
  - buy\_select\_ship, [45](#)
  - menu\_credits, [45](#)
  - menu\_infos, [46](#)
  - menu\_main, [46](#)
  - menu\_shop, [46](#)
  - print\_select, [47](#)
  - print\_stars, [47](#)
- menu\_credits
  - menu.c, [41](#)
  - menu.h, [45](#)
- menu\_infos
  - menu.c, [41](#)
  - menu.h, [46](#)
- menu\_main
  - menu.c, [42](#)
  - menu.h, [46](#)
- menu\_shop
  - menu.c, [42](#)
  - menu.h, [46](#)
- meteorite
  - resources\_s, [9](#)
  - world\_s, [14](#)
- missile
  - resources\_s, [9](#)
  - world\_s, [14](#)
- name
  - playerinfo\_s, [7](#)
- pause
  - sdl2-light.c, [51](#)
  - sdl2-light.h, [54](#)
- playerinfo\_s, [6](#)
  - bestTime, [7](#)
  - hasShip, [7](#)
  - lastTime, [7](#)
  - name, [7](#)
  - selectedShip, [7](#)
  - stars, [7](#)
- print\_select
  - menu.c, [43](#)
  - menu.h, [47](#)
- print\_sprite
  - logique.c, [29](#)
  - logique.h, [36](#)
- print\_stars
  - menu.c, [43](#)
  - menu.h, [47](#)
- refresh\_graphics
  - graphique.c, [19](#)
  - graphique.h, [24](#)
- resources\_s, [8](#)
  - background, [8](#)
  - background\_menu, [8](#)
  - ennemy\_ship, [8](#)
  - finish\_line, [9](#)
  - font, [9](#)
  - font\_menu, [9](#)
  - meteorite, [9](#)
  - missile, [9](#)
  - return\_menu, [9](#)
  - select, [9](#)
  - selected, [9](#)
  - shield, [10](#)
  - ship, [10](#)
  - ship1, [10](#)
  - ship2, [10](#)
  - ship3, [10](#)
  - ship4, [10](#)
  - star, [10](#)
  - time, [10](#)
- retriev\_index
  - logique.c, [29](#)
  - logique.h, [37](#)
- return\_menu
  - resources\_s, [9](#)
- save\_info
  - transition.c, [63](#)
  - transition.h, [67](#)
- save\_score
  - transition.c, [63](#)
  - transition.h, [67](#)
- sdl2-light.c, [48](#)
  - apply\_texture, [48](#)
  - clean\_sdl, [49](#)
  - clean\_texture, [49](#)

- clear\_renderer, 49
- init\_sdl, 50
- load\_image, 50
- pause, 51
- update\_screen, 51
- sdl2-light.h, 51
  - apply\_texture, 52
  - clean\_sdl, 53
  - clean\_texture, 53
  - clear\_renderer, 53
  - init\_sdl, 53
  - load\_image, 54
  - pause, 54
  - update\_screen, 55
- sdl2-ttf-light.c, 55
  - apply\_text, 56
  - clean\_font, 56
  - load\_font, 56
- sdl2-ttf-light.h, 57
  - apply\_text, 58
  - clean\_font, 58
  - load\_font, 58
- select
  - gameinfo\_s, 6
  - resources\_s, 9
- selected
  - resources\_s, 9
- selectedShip
  - playerinfo\_s, 7
- shield
  - resources\_s, 10
- ship
  - resources\_s, 10
  - world\_s, 14
- ship1
  - resources\_s, 10
- ship2
  - resources\_s, 10
- ship3
  - resources\_s, 10
- ship4
  - resources\_s, 10
- shoot
  - logique.c, 30
  - logique.h, 37
- sprite\_s, 11
  - h, 11
  - w, 11
  - x, 11
  - y, 12
- sprites\_collide
  - logique.c, 30
  - logique.h, 37
- star
  - resources\_s, 10
- stars
  - playerinfo\_s, 7
- startTime
  - gameinfo\_s, 6
- tests.c, 59
- time
  - resources\_s, 10
- transition.c, 60
  - clean, 61
  - concat\_array\_playername, 61
  - formatted\_charArray\_to\_uint, 62
  - handle\_events, 62
  - init, 62
  - init\_player, 63
  - save\_info, 63
  - save\_score, 63
- transition.h, 64
  - clean, 65
  - concat\_array\_playername, 65
  - formatted\_charArray\_to\_uint, 66
  - handle\_events, 66
  - init, 66
  - init\_player, 67
  - save\_info, 67
  - save\_score, 67
- update\_data
  - logique.c, 30
  - logique.h, 38
- update\_screen
  - sdl2-light.c, 51
  - sdl2-light.h, 55
- update\_sprites
  - logique.c, 31
  - logique.h, 38
- w
  - sprite\_s, 11
- world\_s, 12
  - bonus, 13
  - bonus\_type, 13
  - collision, 13
  - collision\_finish\_line, 13
  - ennemy, 13
  - finish\_line, 13
  - gameover, 13
  - hasShield, 13
  - is\_bonus\_available, 14
  - is\_ennemy\_dead, 14
  - is\_missile\_free, 14
  - meteorite, 14
  - missile, 14
  - ship, 14
- x
  - sprite\_s, 11
- y
  - sprite\_s, 12