My Project

Generated by Doxygen 1.8.18

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 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 * ennemy_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 ennemy_ship

```
SDL_Texture* resources_s::ennemy_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 ennemy

```
sprite_t world_s::ennemy[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_ennemy_dead

```
unsigned char world_s::is_ennemy_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_ennemies (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

renderer	le renderer SDL
resource	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

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

4.2.2.3 apply_ennemies()

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

La fonction applique la ressource aux différents ennemis.

Parameters

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

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

renderer	le renderer SDL
resource	les ressources du jeu
sprite	sprite à afficher
make_disappear	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

renderer	le renderer SDL
world	les données du monde
resources	les ressources
game	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

renderer	le renderer SDL
world	les données du monde
resources	les ressources

4.2.2.8 clean_resources()

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

La fonction nettoie les ressources.

Parameters

resources	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

renderer	le renderer SDL
resources	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

renderer	renderer SDL
world	les données du monde
resources	les ressources
game	données du jeu
player	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_ennemies (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

renderer	le renderer SDL
resource	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

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

4.3.2.3 apply_ennemies()

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

La fonction applique la ressource aux différents ennemis.

Parameters

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

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

renderer	le renderer SDL
resource	les ressources du jeu
sprite	sprite à afficher
make_disappear	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

renderer	le renderer SDL
world	les données du monde
resources	les ressources
game	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

renderer	le renderer SDL
world	les données du monde
resources	les ressources

4.3.2.8 clean_resources()

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

La fonction nettoie les ressources.

Parameters

resources	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

renderer	le renderer SDL
resources	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

renderer	renderer SDL
world	les données du monde
resources	les ressources
game	données du jeu
player	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_ennemies (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 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

a	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

sprite	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

sprite	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

sp1	vaisseau
sp2	sprite (mur ou ligne arrivé)
world	monde
player	les données du joueur
game	les données du jeu
make_disappear	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

world	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

world	les données du monde

4.4.2.7 init_ennemies()

```
void init_ennemies (
            world_t * world )
```

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

Parameters

world	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

world	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

sprite	vaisseau
x	abscisse de la position du sprite
y	ordonnée de la position du sprite
w	largeur de la position du sprite
h	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

world	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

world	les données du monde
game	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

sprite	vaisseau

4.4.2.13 retreive_index()

```
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.

Parameters

sp1	adresse d'un sprite
Array	adresse d'un sprite Array
limit	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

world	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

sp1	vaisseau
sp2	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

world	les données du monde
game	les données du jeu
player	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

world	les données du monde
player	les données du joueur
game	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_ennemies (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

a	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

sprite	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

sprite	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

sp1	vaisseau
sp2	sprite (mur ou ligne arrivé)
world	monde
player	les données du joueur
game	les données du jeu
make_disappear	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

world	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

world	les données du monde

4.5.2.7 init_ennemies()

```
void init_ennemies (
            world_t * world )
```

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

Parameters

world	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

world	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

sprite	vaisseau
x	abscisse de la position du sprite
y	ordonnée de la position du sprite
w	largeur de la position du sprite
h	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

world	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

world	les données du monde
game	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

sprite	vaisseau

4.5.2.13 retreive_index()

```
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.

Parameters

sp1	adresse d'un sprite
Array	adresse d'un sprite Array
limit	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

world	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

sp1	vaisseau
sp2	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

world	les données du monde
game	les données du jeu
player	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

world	les données du monde
player	les données du joueur
game	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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
resources	les resources
game	les données du jeu
player	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

player	les données du joueur
ShipNumber	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

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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
resources	les resources
game	les données du jeu
player	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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
resources	les resources
game	les données du jeu
player	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

renderer	le renderer lié à l'écran de jeu

Parameters

world	les données du monde
resources	les resources
game	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

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

renderer	le renderer SDL
resources	les ressources de l'application
player	les données du joueur
x	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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
resources	les resources
game	les données du jeu
player	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

player	les données du joueur
ShipNumber	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

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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
resources	les resources
game	les données du jeu
player	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

renderer	le renderer lié à l'écran de jeu
world	les données du monde
resources	les resources
game	les données du jeu
player	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

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

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

renderer	le renderer SDL
resources	les ressources de l'application
player	les données du joueur
x	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

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 où la couleur RGB (255, 0, 255) est rendue transparente.

Parameters

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

Returns

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

4.9.2.7 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 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

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)

4.10.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

4.10.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	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

renderer	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

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

4.10.2.6 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

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

Returns

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

4.10.2.7 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

4.10.2.8 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

4.11 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

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

4.11.2.2 clean_font()

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

La fonction nettoie une police en mémoire.

Parameters

font	la police

4.11.2.3 load_font()

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

La fonction charge une police.

Parameters

path	le chemin du fichier correpondant à la police
font_size	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

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

4.12.2.2 clean_font()

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

La fonction nettoie une police en mémoire.

Parameters

font	la police

4.12.2.3 load_font()

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

La fonction charge une police.

Parameters

path	le chemin du fichier correpondant à la police
font_size	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_ennemies_param** (world_t *world)
- void **test_init_ennemies** ()
- 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 *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 resources, 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 évènements 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 chaine 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

window	la fenêtre du jeu
renderer	le renderer
resources	les ressources
world	le monde
game	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 chaine de caractères avec le nom du joueur.

Parameters

player	les données du joueur
array	chaine de caractères
arraySize	taille de array

Returns

pointeur vers la chaine 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

Array	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 évènements ayant eu lieu et qui n'ont pas encore été traités.

Parameters

event	paramètre qui contient les événements
world	les données du monde
game	les données du jeu
player	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 resources, initialisation des données (du jeu, du monde et du joueur)

Parameters

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

4.14.2.6 init_player()

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

La fonction initialise les données joueurs (extension)

Parameters

player	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

player	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

world	les données du monde
game	les données du jeu
player	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 resources, 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 évènements 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 chaine 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

window	la fenêtre du jeu
renderer	le renderer
resources	les ressources
world	le monde
game	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 chaine de caractères avec le nom du joueur.

Parameters

player	les données du joueur
array	chaine de caractères
arraySize	taille de array

Returns

pointeur vers la chaine 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

Array	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 évènements ayant eu lieu et qui n'ont pas encore été traités.

Parameters

event	paramètre qui contient les événements
world	les données du monde
game	les données du jeu
player	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 resources, initialisation des données (du jeu, du monde et du joueur)

Parameters

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

4.15.2.6 init_player()

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

La fonction initialise les données joueurs (extension)

Parameters

player	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

player	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

world	les données du monde
game	les données du jeu
player	les données du joueur

Index