My Project

Generated by Doxygen 1.9.1

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 sprite_s Struct Reference

Représentation d'un sprite du jeu.

`#include <logique.h>`

Public Attributes

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

3.1.1 Detailed Description

Représentation d'un sprite du jeu.

3.1.2 Member Data Documentation

3.1.2.1 h

`int sprite_s::h`

Champ indiquant la hauteur

3.1.2.2 w

`int sprite_s::w`

Champ indiquant la largeur

3.1.2.3 x

```
int sprite_s::x
```

Champ indiquant l'abscisse de la position

3.1.2.4 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

3.2 textures_s Struct Reference

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

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

Public Attributes

- SDL_Texture * background
- SDL_Texture * vaisseau
- SDL_Texture * ligne_arrive
- SDL_Texture * meteorite

3.2.1 Detailed Description

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

3.2.2 Member Data Documentation

3.2.2.1 background

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

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

3.2.2.2 ligne_arrive

```
SDL_Texture* textures_s::ligne_arrive
```

Texture liée à la ligne d'arrivée

3.2.2.3 meteorite

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

Texture liée à une météorite

3.2.2.4 vaisseau

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

Texture liée à l'image du vaisseau.

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

- graphique.h

3.3 world_s Struct Reference

Représentation du monde du jeu.

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

Public Attributes

- sprite_t vaisseau
- sprite_t ligne_arrive
- sprite_t mur
- int **vy**
- int gameover
- int collision_mur

3.3.1 Detailed Description

Représentation du monde du jeu.

3.3.2 Member Data Documentation

3.3.2.1 collision_mur

```
int world_s::collision_mur
```

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

3.3.2.2 gameover

```
int world_s::gameover
```

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

3.3.2.3 ligne_arrive

```
sprite_t world_s::ligne_arrive
```

Champ représentant la ligne d'arrivée

3.3.2.4 mur

```
sprite_t world_s::mur
```

Champ représentant un mur de météorites

3.3.2.5 vaisseau

```
sprite_t world_s::vaisseau
```

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.

Macros

- #define SCREEN_WIDTH 300

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

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

  Taille d'un vaisseau.
- #define METEORITE_SIZE 32

  Taille d'un météorite.
- #define FINISH_LINE_HEIGHT 10

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

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

  Vitesse initiale de déplacement vertical des éléments du jeu.

4.1.1 Detailed Description

Header regroupant toutes les constantes utiles au programme.

Author

LESNIAK Louis & SLIMANI Kamelia

Version

2.0

Date

14 avril 2021

4.2 graphique.c File Reference

Module gérant la partie graphique du jeu.

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

Functions

- void clean_textures (textures_t *textures)

  La fonction nettoie les textures.
- void init_textures (SDL_Renderer *renderer, textures_t *textures)

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

  La fonction initialise la position du sprite.
- void apply_background (SDL_Renderer *renderer, SDL_Texture *texture)

  La fonction applique la texture du fond sur le renderer lié à l'écran de jeu.
- void apply_meteorite (SDL_Renderer *renderer, world_t *world, textures_t *textures)

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

  La fonction rafraichit l'écran en fonction de l'état des données du monde.
- void clean (SDL_Window *window, SDL_Renderer *renderer, textures_t *textures, world_t *world)

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

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

4.2.1 Detailed Description

Module gérant la partie graphique du jeu.

Author

LESNIAK Louis & SLIMANI Kamelia

Version

2.0

Date

14 avril 2021

4.2.2 Function Documentation

4.2.2.1 apply_background()

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

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

Parameters

renderer	le renderer
texture	la texture liée au fond
sprite	sprite
make_disappear	permet de faire disparaitre ou non le sprite

4.2.2.2 apply_meteorite()

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

La fonction applique la texture des météorites sur le mur.

Parameters

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

4.2.2.3 apply_sprite()

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

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

4.2.2.4 clean()

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

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

Parameters

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.2.2.5 clean_textures()

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

La fonction nettoie les textures.

Parameters

textures	les textures

4.2.2.6 init()

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

```
            textures_t * textures,
            world_t * world )
```

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

Parameters

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.2.2.7 init_textures()

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

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

Parameters

screen	la surface correspondant à l'écran de jeu
textures	les textures du jeu

4.2.2.8 refresh_graphics()

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

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

Parameters

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

4.3 graphique.h File Reference

Header du module graphique.

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

Classes

- struct textures_s

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

Typedefs

- typedef struct textures_s textures_t

  Type qui correspond aux textures du jeu.

Functions

- void clean_textures (textures_t *textures)

  La fonction nettoie les textures.
- void init_textures (SDL_Renderer *renderer, textures_t *textures)

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

  La fonction initialise la position du sprite.
- void apply_background (SDL_Renderer *renderer, SDL_Texture *texture)

  La fonction applique la texture du fond sur le renderer lié à l'écran de jeu.
- void apply_meteorite (SDL_Renderer *renderer, world_t *world, textures_t *textures)

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

  La fonction rafraichit l'écran en fonction de l'état des données du monde.
- void clean (SDL_Window *window, SDL_Renderer *renderer, textures_t *textures, world_t *world)

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

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

4.3.1 Detailed Description

Header du module graphique.

Author

LESNIAK Louis & SLIMANI Kamelia

Version

2.0

Date

14 avril 2021

4.3.2 Function Documentation

4.3.2.1 apply_background()

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

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

Parameters

renderer	le renderer
texture	la texture liée au fond
sprite	sprite
make_disappear	permet de faire disparaitre ou non le sprite

4.3.2.2 apply_meteorite()

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

La fonction applique la texture des météorites sur le mur.

Parameters

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

4.3.2.3 apply_sprite()

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

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

4.3.2.4 clean()

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

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

Parameters

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.3.2.5 clean_textures()

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

La fonction nettoie les textures.

Parameters

textures	les textures

4.3.2.6 init()

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

```
            textures_t * textures,
            world_t * world )
```

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

Parameters

window	la fenêtre du jeu
renderer	le renderer
textures	les textures
world	le monde

4.3.2.7 init_textures()

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

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

Parameters

screen	la surface correspondant à l'écran de jeu
textures	les textures du jeu

4.3.2.8 refresh_graphics()

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

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

Parameters

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

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>
```

Functions

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

  La fonction initialise la position du sprite.
- 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.
- void clean_data (world_t *world)

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

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

  La fonction met à jour les données en tenant compte de la physique du monde.
- 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
- 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
- void handle_sprites_collision (sprite_t *sp1, sprite_t *sp2, world_t *world, int make_disappear)

  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

2.0

Date

14 avril 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 clean_data()

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

La fonction nettoie les données du monde.

Parameters

world	les données du monde

4.4.2.3 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

4.4.2.4 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

4.4.2.5 handle_sprites_collision()

```
void handle_sprites_collision (
            sprite_t * sp1,
            sprite_t * sp2,
            world_t * world,
            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
make_disappear	sprite visible/invisible

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_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

4.4.2.8 is_game_over()

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

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

Parameters

world	les données du monde

Returns

1 si le jeu est fini, 0 sinon

4.4.2.9 print_sprite()

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

La fonction affiche la position du sprite.

Parameters

sprite	vaisseau

4.4.2.10 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)

4.4.2.11 update_data()

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

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

Parameters

les	données du monde

4.5 logique.h File Reference

Header du module logique.

Classes

- struct sprite_s

  Représentation d'un sprite du jeu.
- struct world_s

  Représentation du monde du jeu.

Typedefs

- typedef struct sprite_s sprite_t

  Type qui correspond aux données du sprite.
- typedef struct world_s world_t

  Type qui correspond aux données du monde.

Functions

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

  La fonction initialise la position du sprite.
- 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.
- void clean_data (world_t *world)

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

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

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

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

  fonction qui verifie que 2 sprite ne soit pas en collision
- void handle_sprites_collision (sprite_t *sp1, sprite_t *sp2, world_t *world, int make_disappear)

  fonction qui verifie que 2 sprite ne soit pas en collision
- int abs (int a)

  fonction qui retourne la valeur absolue d'un entier

4.5.1 Detailed Description

Header du module logique.

Author

LESNIAK Louis & SLIMANI Kamelia

Version

2.0

Date

14 avril 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 clean_data()

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

La fonction nettoie les données du monde.

Parameters

world	les données du monde

4.5.2.3 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

4.5.2.4 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

4.5.2.5 handle_sprites_collision()

```
void handle_sprites_collision (
            sprite_t * sp1,
            sprite_t * sp2,
            world_t * world,
            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
make_disappear	sprite visible/invisible

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_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

4.5.2.8 is_game_over()

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

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

Parameters

world	les données du monde

Returns

1 si le jeu est fini, 0 sinon

4.5.2.9 print_sprite()

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

La fonction affiche la position du sprite.

Parameters

sprite	vaisseau

4.5.2.10 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)

4.5.2.11 update_data()

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

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

Parameters

les	données du monde

4.6 main.c File Reference

Programme principal initial du niveau 2.

```
#include "sdl2-light.h"
#include "logique.h"
#include "graphique.h"
#include "definition.h"
```

Functions

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

  La fonction gère les évènements ayant eu lieu et qui n'ont pas encore été traités.
- int main (int argc, char ∗args[ ])

  programme principal qui implémente la boucle du jeu

4.6.1 Detailed Description

Programme principal initial du niveau 2.

Author

LESNIAK Louis & SLIMANI Kamelia

Version

2.0

Date

14 avril 2021

4.6.2 Function Documentation

4.6.2.1 handle_events()

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

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

Parameters

event	paramètre qui contient les événements
world	les données du monde

4.7 sdl2-light.c File Reference

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

```
#include "sdl2-light.h"
#include <stdio.h>
#include <stdlib.h>
```

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.

4.7.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

4.7.2 Function Documentation

4.7.2.1 apply_texture()

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

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

Parameters

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)

4.7.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

4.7.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

4.7.2.4 clear_renderer()

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

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

Parameters

renderer	le renderer de l'écran

4.7.2.5 init_sdl()

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

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

Parameters

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

4.7.2.6 load_image()

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

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

Parameters

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)

4.7.2.7 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

4.7.2.8 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

4.8 sdl2-light.h File Reference

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

```
#include "include/SDL.h"
```

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.

4.8.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

4.8.2 Function Documentation

4.8.2.1 apply_texture()

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

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

Parameters

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

4.8.2.2 clean_sdl()

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

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

Parameters

renderer	le renderer à nettoyer
window	la fenêtre à nettoyer

4.8.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

texture	la texture à nettoyer

4.8.2.4 clear_renderer()

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

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

Parameters

renderer	le renderer de l'écran

4.8.2.5 init_sdl()

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

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

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

Parameters

window	la fenêtre du jeu
renderer	le renderer
width	largeur de l'écran de jeu
height	hauteur de l'écran de jeu

Returns

-1 en cas d'erreur, 0 sinon

4.8.2.6 load_image()

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

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

Parameters

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

Returns

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

4.8.2.7 pause()

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

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

Parameters

time	ce laps de temps en milliseconde

4.8.2.8 update_screen()

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

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

Parameters

renderer	le renderer de l'écran

4.9 tests.c File Reference

Programme testant la partie logique du jeu.

```
#include "definition.h"
#include "logique.h"
#include <stdio.h>
```

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_handle_sprites_collision_param** (sprite_t *sp1, sprite_t *sp2, world_t *world)
- void **test_handle_sprites_collision** ()
- int **main** ()

4.9.1 Detailed Description

Programme testant la partie logique du jeu.

Author

LESNIAK Louis & SLIMANI Kamelia

Version

2.0

Date

14 avril 2021

Index