

My Project

Generated by Doxygen 1.8.13

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	textures_s Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Data Documentation	5
3.1.2.1	personnage	5
3.2	world_s Struct Reference	5
3.2.1	Detailed Description	6
3.2.2	Member Data Documentation	6
3.2.2.1	gameover	6
4	File Documentation	7
4.1	main.c File Reference	7
4.2	sdl2-light.c File Reference	7
4.2.1	Detailed Description	8
4.2.2	Function Documentation	8
4.2.2.1	apply_texture()	8
4.2.2.2	clean_sdl()	8
4.2.2.3	clean_texture()	9
4.2.2.4	clear_renderer()	9
4.2.2.5	init_sdl()	9
4.2.2.6	load_image()	10
4.2.2.7	pause()	10
4.2.2.8	update_screen()	10
	Index	11

Chapter 1

Class Index

1.1 Class List

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

textures_s	Représentation pour stocker les textures nécessaires à l'affichage graphique	5
world_s	Représentation du monde du jeu	5

Chapter 2

File Index

2.1 File List

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

main.c	Programme principal initial du niveau 0	7
sdl2-light.c	Sur-couche de SDL2 pour simplifier son utilisation pour le projet	7
sdl2-light.h	??

Chapter 3

Class Documentation

3.1 textures_s Struct Reference

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

Public Attributes

- SDL_Texture * **background**
- SDL_Texture * [personnage](#)

3.1.1 Detailed Description

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

3.1.2 Member Data Documentation

3.1.2.1 personnage

```
SDL_Texture* textures_s::personnage
```

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

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

- [main.c](#)

3.2 world_s Struct Reference

Représentation du monde du jeu.

Public Attributes

- int [gameover](#)

3.2.1 Detailed Description

Représentation du monde du jeu.

3.2.2 Member Data Documentation

3.2.2.1 gameover

```
int world_s::gameover
```

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

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

- [main.c](#)

Chapter 4

File Documentation

4.1 main.c File Reference

Programme principal initial du niveau 0.

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

Include dependency graph for main.c:

4.2 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>
```

Include dependency graph for sdl2-light.c:

Functions

- int [init_sdl](#) (SDL_Window **window, SDL_Renderer **renderer, int width, int height)
La fonction initialise la SDL. Elle crée la fenêtre du jeu ainsi que le renderer.
- SDL_Texture * [load_image](#) (const char path[], SDL_Renderer *renderer)
La fonction charge une image et renvoie la texture correspondante.
- void [apply_texture](#) (SDL_Texture *texture, SDL_Renderer *renderer, int x, int y)
La fonction permet d'appliquer une texture sur le renderer à une position donnée. La hauteur et la largeur est la même que celle de la texture.
- void [clean_texture](#) (SDL_Texture *texture)
La fonction nettoie une texture en mémoire.
- void [clear_renderer](#) (SDL_Renderer *renderer)
La fonction vide le contenu graphique du renderer lié à l'écran de jeu.
- void [update_screen](#) (SDL_Renderer *renderer)
La fonction met à jour l'écran avec le contenu du renderer.
- void [pause](#) (int time)
La fonction met le programme en pause pendant un laps de temps.
- void [clean_sdl](#) (SDL_Renderer *renderer, SDL_Window *window)
La fonction nettoie le renderer et la fenêtre du jeu en mémoire.

4.2.1 Detailed Description

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

Author

Mathieu Constant

Version

0.1

Date

10 mars 2020

4.2.2 Function Documentation

4.2.2.1 `apply_texture()`

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

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

Parameters

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

4.2.2.2 `clean_sdl()`

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

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

Parameters

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

4.2.2.3 `clean_texture()`

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

La fonction nettoie une texture en mémoire.

Parameters

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

4.2.2.4 `clear_renderer()`

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

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

Parameters

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

4.2.2.5 `init_sdl()`

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

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

Parameters

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

Returns

-1 en cas d'erreur, 0 sinon

4.2.2.6 load_image()

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

La fonction charge une image et renvoie la texture correspondante.

Parameters

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

Returns

la surface SDL contenant l'image. Elle renvoie NULL si le chargement a échoué (ex. le fichier *path* n'existe pas)

4.2.2.7 pause()

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

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

Parameters

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

4.2.2.8 update_screen()

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

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

Parameters

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

Index

- apply_texture
 - sdl2-light.c, [8](#)
- clean_sdl
 - sdl2-light.c, [8](#)
- clean_texture
 - sdl2-light.c, [9](#)
- clear_renderer
 - sdl2-light.c, [9](#)
- gameover
 - world_s, [6](#)
- init_sdl
 - sdl2-light.c, [9](#)
- load_image
 - sdl2-light.c, [10](#)
- main.c, [7](#)
- pause
 - sdl2-light.c, [10](#)
- personnage
 - textures_s, [5](#)
- sdl2-light.c, [7](#)
 - apply_texture, [8](#)
 - clean_sdl, [8](#)
 - clean_texture, [9](#)
 - clear_renderer, [9](#)
 - init_sdl, [9](#)
 - load_image, [10](#)
 - pause, [10](#)
 - update_screen, [10](#)
- textures_s, [5](#)
 - personnage, [5](#)
- update_screen
 - sdl2-light.c, [10](#)
- world_s, [5](#)
 - gameover, [6](#)