

My Project

Generated by Doxygen 1.8.17

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 background	5
3.1.2.2 sprite	5
3.2 world_s Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 gameover	6
3.2.2.2 x	6
3.2.2.3 y	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
4.3 sdl2-light.h File Reference	11
4.3.1 Detailed Description	11
4.3.2 Function Documentation	12
4.3.2.1 apply_texture()	12
4.3.2.2 clean_sdl()	12
4.3.2.3 clean_texture()	12
4.3.2.4 clear_renderer()	13
4.3.2.5 init_sdl()	13
4.3.2.6 load_image()	13
4.3.2.7 pause()	14
4.3.2.8 update_screen()	14

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	6

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	En-tête du module correspondant à une sur-couche de SDL2 pour simplifier son utilisation pour le projet	11

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 *` [sprite](#)

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 background

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

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

3.1.2.2 sprite

```
SDL_Texture* textures_s::sprite
```

Texture liée au personnage.

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](#)
- int [x](#)
- int [y](#)

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

3.2.2.2 x

```
int world_s::x
```

Coordonnée x du personnage.

3.2.2.3 y

```
int world_s::y
```

Coordonnée y du personnage.

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 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.

4.2.1 Detailed Description

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

Author

Mathieu Constant

Version

0.2

Date

10 mars 2021

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

Parameters

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

Returns

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

4.2.2.7 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

<code>render</code>	le renderer de l'écran
---------------------	------------------------

4.3 sdl2-light.h File Reference

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

```
#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 render 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 render.
- 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 render 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 render lié à l'écran de jeu.
- void `update_screen` (SDL_Renderer *renderer)
La fonction met à jour l'écran avec le contenu du render.
- void `pause` (int time)
La fonction met le programme en pause pendant un laps de temps.

4.3.1 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

4.3.2 Function Documentation

4.3.2.1 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)

4.3.2.2 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

4.3.2.3 clean_texture()

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

La fonction nettoie une texture en mémoire.

Parameters

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

4.3.2.4 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
-----------------	------------------------

4.3.2.5 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

4.3.2.6 load_image()

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

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

Parameters

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

Returns

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

4.3.2.7 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
-------------	----------------------------------

4.3.2.8 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](#)
 - sdl2-light.h, [12](#)
- background
 - textures_s, [5](#)
- clean_sdl
 - sdl2-light.c, [8](#)
 - sdl2-light.h, [12](#)
- clean_texture
 - sdl2-light.c, [9](#)
 - sdl2-light.h, [12](#)
- clear_renderer
 - sdl2-light.c, [9](#)
 - sdl2-light.h, [13](#)
- gameover
 - world_s, [6](#)
- init_sdl
 - sdl2-light.c, [9](#)
 - sdl2-light.h, [13](#)
- load_image
 - sdl2-light.c, [10](#)
 - sdl2-light.h, [13](#)
- main.c, [7](#)
- pause
 - sdl2-light.c, [10](#)
 - sdl2-light.h, [14](#)
- 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](#)
- sdl2-light.h, [11](#)
 - apply_texture, [12](#)
 - clean_sdl, [12](#)
 - clean_texture, [12](#)
 - clear_renderer, [13](#)
 - init_sdl, [13](#)
 - load_image, [13](#)
 - pause, [14](#)
 - update_screen, [14](#)
- sprite
 - textures_s, [5](#)
- textures_s, [5](#)
 - background, [5](#)
 - sprite, [5](#)
- update_screen
 - sdl2-light.c, [10](#)
 - sdl2-light.h, [14](#)
- world_s, [6](#)
 - gameover, [6](#)
 - x, [6](#)
 - y, [6](#)
- x
 - world_s, [6](#)
- y
 - world_s, [6](#)