Sokoban

Généré par Doxygen 1.13.1

1 Main Test 1

2 Index des structures de données 3

2.1 Structures de données . . . 3

3 Index des fichiers 5

3.1 Liste des fichiers . . . 5

4 Documentation des structures de données 7

4.1 Référence de la structure essential_sdl . . . 7

4.2 Référence de la structure Score . . . 7

4.3 Référence de la structure Vecteur . . . 7

5 Documentation des fichiers 9

5.1 display.h . . . 9

5.2 function.h . . . 9

5.3 read.h . . . 10

5.4 Référence du fichier display.c . . . 10

5.4.1 Description détaillée . . . 11

5.4.2 Documentation des fonctions . . . 11

5.4.2.1 backgroundDisplay() . . . 11

5.5 Référence du fichier function.c . . . 12

5.5.1 Description détaillée . . . 13

5.5.2 Documentation des fonctions . . . 13

5.5.2.1 blockBox() . . . 13

5.5.2.2 canIGoDirection() . . . 13

5.5.2.3 creatArea2D() . . . 14

5.5.2.4 free2D() . . . 14

5.5.2.5 inEditorLoop() . . . 15

5.5.2.6 inGameLoop() . . . 15

5.5.2.7 islose() . . . 16

5.5.2.8 isWin() . . . 16

5.5.2.9 move() . . . 17

5.5.2.10 nullScore() . . . 17

5.5.2.11 plusVect() . . . 17

5.5.2.12 timeToText() . . . 18

5.5.2.13 titleScreen() . . . 18

5.5.2.14 winOrLoseLoop() . . . 18

5.6 Référence du fichier main.c . . . 19

5.6.1 Description détaillée . . . 19

5.7 Référence du fichier read.c . . . 19

5.7.1 Description détaillée . . . 20

5.7.2 Documentation des fonctions . . . 20

5.7.2.1 countCustomMaps() . . . 20

5.7.2.2 fileToTab2D() 20
5.7.2.3 generatorMenu() 20
5.7.2.4 save_grid_to_file() 21

Chapter 1

Main Test

ewuewiewhf wef ewuewiewhf ewuewiewhfewf wefe:Lw

dewfewfewfew f ewuewiewhfewf fw ewuewiewhfewfewf f wf

Chapter 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

essential_sdl 7
Score 7
Vecteur 7

Chapter 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

display.h 9
function.h 9
read.h 10
display.c 10
function.c 12
main.c 19
read.c 19

Chapter 4

Documentation des structures de données

4.1 Référence de la structure essential_sdl

Champs de données

- SDL_Window ∗ **window**
- SDL_Renderer ∗ **renderer**
- unsigned int **size_window**
- unsigned int **size_box**
- unsigned int **size_menu**

La documentation de cette structure a été générée à partir du fichier suivant :

- function.h

4.2 Référence de la structure Score

Champs de données

- time_t **before**
- time_t **after**
- unsigned int **move_player**
- unsigned int **move_box**

La documentation de cette structure a été générée à partir du fichier suivant :

- function.h

4.3 Référence de la structure Vecteur

Champs de données

- int **x**
- int **y**

La documentation de cette structure a été générée à partir du fichier suivant :

- function.h

Chapter 5

Documentation des fichiers

5.1 display.h

```
#ifndef DISPLAY_H
#define DISPLAY_H

#include "../include/function.h"
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>



extern unsigned int seed;
void screenDisplay (char **tab, int x, int y);
int getMaxSize (dis display_user);
void displayImage (SDL_Renderer *renderer, SDL_Texture *texture, vect pos,
                   int size);
void initSDL (dis *display_user);
void displayTextSDL(dis *display_user,char *text, vect coor, vect size, int font_size);
void screenDisplayGameSDL (char **tab,vect dim_tab, dis *display_user, vect *player_pos, int fov, vect direction);
void backgroundDisplay(dis *display_user,int bg);
void playAudio(int sfx);

#endif // !DISPLAY_H
```

5.2 function.h

```
#ifndef FONCTION_H
#define FONCTION_H

#include <SDL2/SDL.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_video.h>
#include <stdbool.h>
#include <stdlib.h>
#include <time.h>

#define EMPTY 1
#define WALL 0
#define BOX 2
#define TARGET 3
#define BOX_ON_TARGET 4
#define PLAYER 5
#define PLAYER_ON_TARGET 6
#define BUTTON 7
#define PLAYER_ON_BUTTON 8
#define BUTTON_CUSTOM 9

typedef struct Vecteur
{
    int x;
    int y;
} vect;

typedef struct Score
```

```
{
    time_t before;
    time_t after;
    unsigned int move_player;
    unsigned int move_box;
} score;

typedef struct essential_sdl
{
    SDL_Window *window;
    SDL_Renderer *renderer;
    unsigned int size_window;
    unsigned int size_box;
    unsigned int size_menu;
} dis;

char **creatArea2D (const int x, const int y);
void free2D (char **tab, int x);
bool isWin (char **tab2d, vect *targets, int nbr_targets);
bool islose (char **tab2d, const int N);
bool blockBox (char **tab2d, vect box_coor);
vect plusVect (vect one, vect two);

int inGameLoop (char **tab2d, vect *dim_tab, vect *playerPos, vect *targets,
                 int nbr_targets, dis *display_user, score *score_user, bool menu);
char *timeToText(time_t time);
void nullScore(score *player_score);
void winOrLoseLoop(dis *display_user,score *score_user, bool win);
unsigned int scoreCalculator(score *score_user, bool win);
unsigned int scoreCalculatorCyp(score *score_user, bool win);
int inEditorLoop (char **tab2d, vect *dim_tab, vect *playerPos, vect *targets,
                int nbr_targets, dis *display_user,score *score_user, int num_fichier);
int titleScreen (dis *display_user);
#endif // FONCTION_H
```

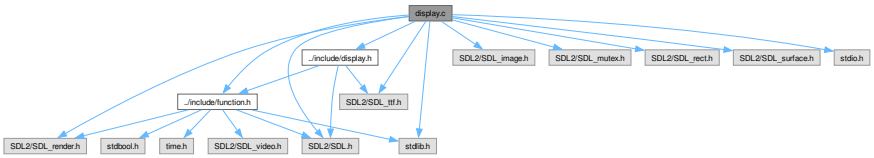
5.3 read.h

```
#ifndef READ_H
#define READ_H
#include "function.h"
vect *fileToTab2D (const char *name_file, char **tab,
                   const unsigned N, vect *player, int *nbr_targets);
int countCustomMaps(char *name_directory);
char ** generatorMenu(char *name_directory, vect *size, vect *pos_player);
void save_grid_to_file(int filename, char **tab2D);
#endif // !READ_H
```

5.4 Référence du fichier display.c

```
#include "../include/display.h"
#include "../include/function.h"
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mutex.h>
#include <SDL2/SDL_mixer.h>
#include <SDL2/SDL_rect.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_surface.h>
#include <SDL2/SDL_ttf.h>
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de display.c:



Fonctions

- void **screenDisplay** (char ∗∗tab, int x, int y)

  La fonction permet d'afficher simplement le plateau de jeu dans le terminal.
- void **screenDisplayGameSDL** (char ∗∗tab, **vect** dim_tab, **dis** ∗display_user, **vect** ∗player_pos, int fov, **vect** direction)

  La fonction affiche a l'aide de SDL la zone de jeu.
- int **getMaxSize** (**dis** display_user)

  Taille de l'ecrant carre en fonction de l'ecrant du joueur avec une marge.
- void **initSDL** (**dis** ∗display_user)

  Initialise SDL.
- void **displayImage** (SDL_Renderer ∗renderer, SDL_Texture ∗texture, **vect** pos, int size)

  Cette fonction affiche l'image dans la fenetre de l'utilisateur.
- void **displayTextSDL** (**dis** ∗display_user, char ∗text, **vect** coor, **vect** size, int font_size)

  Cette fonction affiche du texte dans la fenetre de l'utilisateur.
- void **backgroundDisplay** (**dis** ∗display_user, int bg)

  Afficher l'arriere plan.
- void **playAudio** (int sfx)

  Cette fonction permet de joueur des effet sonor.

5.4.1 Description détaillée

Fichier contient tout les fonctions pour l'affichage.

5.4.2 Documentation des fonctions

5.4.2.1 backgroundDisplay()

```
void backgroundDisplay (
            dis * display_user,
           int bg)
```

Afficher l'arriere plan.

Paramètres

display_user	Tout les information du display de l'utilisateur utile.
bg	quel back ground afficher.

Voici le graphe d'appel pour cette fonction :

5.5 Référence du fichier function.c

```
#include "../include/function.h"
#include "../include/display.h"
#include "../include/read.h"
#include <SDL2/SDL_events.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_keycode.h>
#include <SDL2/SDL_rect.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_scancode.h>
#include <SDL2/SDL_timer.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Graphe des dépendances par inclusion de function.c:

Fonctions

- char ** **creatArea2D** (const int x, const int y)

  Cette fonction permet de creer une liste 2D.
- void **free2D** (char **tab, int x)

  Cette fontion permet de liberer l'espace tu tableau 2D de char.
- char **canIGoDirection** (char valueOfNCase, char valueOfNPlusOneCase)

  La fontction permet de savoir si le joueur peut ce deplacer dans une direction.
- int **move** (char **tab, **vect** *playerPos, **vect** direction, **score** *score_user)

  Cette fonction effectue les deplacements du joueur et des boites en fonction de la situation.
- int **inGameLoop** (char **tab2d, **vect** *dim_tab, **vect** *playerPos, **vect** *targets, int nbr_targets, **dis** *display_user, **score** *score_user, bool menu)

  La fonction permet de faire la boucle de jeu et le menu.
- int **inEditorLoop** (char **tab2d, **vect** *dim_tab, **vect** *playerPos, **vect** *targets, int nbr_targets, **dis** *display_user, **score** *score_user, int num_fichier)

  La fonction permet de faire la boucle de l'éditeur.
- int **titleScreen** (**dis** *display_user)

  La fonction permet d' afficher le title screen.
- bool **isWin** (char **tab2d, **vect** *targets, int nbr_targets)

  Cette fonction verrifie si la partie est gagnante.
- bool **islose** (char **tab2d, const int N)

  La fonction renvois si la partie est perdante.
- **vect** **plusVect** (**vect** one, **vect** two)

  La fonction fait une addition de vecteur, (x1+x2, y1+y2).
- int **lengthVect** (**vect** vector)

  Renvois la longeur Manhattan.
- bool **blockBox** (char **tab2d, **vect** box_coor)

  La fonction permet de savoir si une boite est dans une situation ou le joueur ne poura pas la debloqué.
- char * **timeToText** (time_t time)

  Cette fonction renvois transforme le forma time en texte. (min:sec)
- void **nullScore** (**score** *player_score)

  Mets à 0 le score.
- void **winOrLoseLoop** (**dis** *display_user, **score** *score_user, bool win)

  Fonction de loop pour la win ou la lose.
- unsigned int **scoreCalculator** (**score** *score_user, bool win)
- unsigned int **scoreCalculatorCyp** (**score** *score_user, bool win)

5.5.1 Description détaillée

Ce fichier contient toute les fonction utile pour le jeu. Sauf pour l'affichage.

5.5.2 Documentation des fonctions

5.5.2.1 blockBox()

```
bool blockBox (
            char ** tab2d,
             vect box_coor)
```

La fonction permet de savoir si une boite est dans une situation ou le joueur ne poura pas la debloqué.

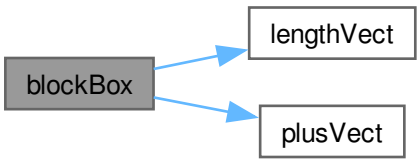
Paramètres

tab2d	Le tableau 2D carre du plateau de jeu.
box_coor	Les corrdonnée de la boite que la fonction test.

Renvoie

True si la la boite est bloquer, sinon false.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

5.5.2.2 canIGoDirection()

```
char canIGoDirection (
            char valueOfNCase,
            char valueOfNPlusOneCase)
```

La fontction permet de savoir si le joueur peut ce deplacer dans une direction.

Paramètres

valueOfNCase	La valeur de la case dans la direction que le joueur veut aller.
valueOfNPlusOneCase	La valeur de la case dans la direction que le joueur veut aller mais une fois de plus.

Renvoie

0 Si c'est un mur devant le joueur, 1 si c'est vide devant le joueur, 2 si c'est une boite mais qu'on peux la pousser ,3 si le joueur pousse une boite sur un point d'interer, 4 si le joueur bouge sur un point d'interer, 5 si le joueur peux pousser une boite mais le joueur se place sur un point d'interer et 6 si le joueur pousse une boite sur un point d'interer et que le joueur est aussi sur un point d'interer. 7 si valueOfNCase est un bouton.

Voici le graphe des appelants de cette fonction :

5.5.2.3 creatArea2D()

```
char ** creatArea2D (
            const int x,
            const int y)
```

Cette fonction permet de creer une liste 2D.

Paramètres

x	Nombre de ligne.
y	Nombre de colonne.

Renvoie

Le pointeur du tableau 2D de char (1 octet).

5.5.2.4 free2D()

```
void free2D (
            char ** tab,
            int x)
```

Cette fontion permet de liberer l'espace tu tableau 2D de char.

Paramètres

tab	Le tableau 2D.
x	Le nombre de ligne.

Renvoie

Void.

5.5.2.5 inEditorLoop()

```
int inEditorLoop (
            char ** tab2d,
             vect * dim_tab,
             vect * playerPos,
             vect * targets,
            int nbr_targets,
             dis * display_user,
             score * score_user,
            int num_fichier)
```

La fonction permet de faire la boucle de l'éditeur.

Paramètres

tab2d	Le tableau 2d carre.
N	LE nombre d'element de tab2d.
playerPos	La position de depart du joueur.
targets	Le tableau de toutes les positions des points d'interer de la maps.
int	nbr_targets Le nombre de point d'interer.
display_user	Tout les information SDL pour afficher le jeu.
score_user	Toute les données nécessaire pour calculer le score fini du joueur.
num_fichier	Simplement le numéro de la map actuel.

Renvoie

renvoie -2 ce qui signifie l'editeur

Voici le graphe d'appel pour cette fonction :

5.5.2.6 inGameLoop()

```
int inGameLoop (
            char ** tab2d,
             vect * dim_tab,
             vect * playerPos,
             vect * targets,
            int nbr_targets,
             dis * display_user,
             score * score_user,
            bool menu)
```

La fonction permet de faire la boucle de jeu et le menu.

Paramètres

tab2d	Le tableau 2d carre.
N	LE nombre d'element de tab2d.
playerPos	La position de depart du joueur.
targets	Le tableau de toutes les positions des points d'interer de la maps.
int	nbr_targets Le nombre de point d'interer.
display_user	Tout les information SDL pour afficher le jeu.
score_user	Toute les données nécessaire pour calculer le score fini du joueur.
menu	True si c'est la loop du menu.

Renvoie

L'indice de la map si c'est un menu, sinon renvoie -1

Voici le graphe d'appel pour cette fonction :

5.5.2.7 islose()

```
bool islose (
            char ** tab2d,
            const int N)
```

La fonction renvois si la partie est perdante.

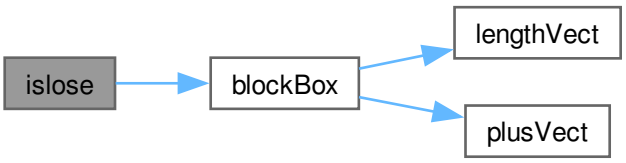
Paramètres

tab2d	Le tableau 2d carre du plateau de jeu.
N	Le nombre d'éléments dans le tab2d. (zone de jeu est carré)

Renvoie

True si c'est perdu, false si c'est pas perdu a cette instant.

Voici le graphe d'appel pour cette fonction :



5.5.2.8 isWin()

```
bool isWin (
            char ** tab2d,
             vect * targets,
            int nbr_targets)
```

Cette fonction verrifie si la partie est gagnante.

Paramètres

tab2d	Le tableau 2D du jeu.
targets	Le tableau de toute les positions des points d'interer.
nbr_targets	Le nombre de points d'interer.

Renvoie

True si le joueur a remplis tout les points d'interer, false si ce n'est pas le cas .

5.5.2.9 move()

```
int move (
            char ** tab,
             vect * playerPos,
             vect direction,
             score * score_user)
```

Cette fonction effectue les deplacements du joueur et des boites en fonction de la situation.

Paramètres

tab	Le tableau 2D du plateau de jeu.
playerpos	La position actuel du joueur.
direction	La direction que le joueur veut effectuer.
score_user	Toutes les données nécessaire pour calculer le score fini du joueur.

Renvoie

int return 1 si le joueur n'a pas bouger.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

5.5.2.10 nullScore()

```
void nullScore (
             score * player_score)
```

Mets à 0 le score.

Paramètres

player_score	Le score a mettre à 0.

Renvoie

void

5.5.2.11 plusVect()

```
vect plusVect (
             vect one,
             vect two)
```

La fonction fait une addition de vecteur, (x1+x2, y1+y2).

Paramètres

one	Premier vecteur.
two	Deuzieme vecteur.

Renvoie

vect Un vecteur de l'addition de one et two.

Voici le graphe des appelants de cette fonction :

5.5.2.12 timeToText()

```
char * timeToText (
            time_t time)
```

Cette fonction renvois transforme le forma time en texte. (min:sec)

Paramètres

time	Le temps qu'on veux convertire.

Renvoie

char Le string du texte.

Voici le graphe des appelants de cette fonction :

5.5.2.13 titleScreen()

```
int titleScreen (
            dis * display_user)
```

La fonction permet d' afficher le title screen.

Paramètres

display_user	Tout les information SDL pour afficher le jeu.

Renvoie

renvoie -3 ce qui signifie le title screen

5.5.2.14 winOrLoseLoop()

```
void winOrLoseLoop (
            dis * display_user,
            score * score_user,
            bool win)
```

Fonction de loop pour la win ou la lose.

Paramètres

display_user	Tout les information du display de l'utilisateur utile.
win	Si on veut un affichage de victoire ou non.

5.6 Référence du fichier main.c

```
#include "../include/display.h"
#include "../include/function.h"
#include "../include/read.h"
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_mixer.h>
#include <time.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
```

Graphe des dépendances par inclusion de main.c:

Macros

- #define **SIZE_PLAY** 20
- #define **SIZE_MENU** 200

Fonctions

- int **main** ()

Variables

- unsigned int **seed** = 0

5.6.1 Description détaillée

Le main permet de stocker et de lancer les fonctions permetant de lancer le jeu.

5.7 Référence du fichier read.c

```
#include "../include/function.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Graphe des dépendances par inclusion de read.c:

Fonctions

- **vect** ∗ **fileToTab2D** (const char ∗name_file, char ∗∗tab, const unsigned N, **vect** ∗player, int ∗nbr_targets)

  La fonction permet de stocker la zone de jeu en fonction de la lecture d'un fichier.
- int **countCustomMaps** (char ∗name_directory)

  La fonction permet de compter le nombre de maps custom dans le fichier des maps.
- char ∗∗ **generatorMenu** (char ∗name_directory, **vect** ∗size, **vect** ∗pos_player)

  La fonction permet de creer la zone de jeu du menu en fonction du nombre de maps custom.
- void **save_grid_to_file** (int filename, char ∗∗tab2D)

  La fonction permet de retranscrire un niveau creer du tableau vers le fichier.txt.

5.7.1 Description détaillée

Ce fichier est le programe qui lit d'autre fichier, nottament les maps.

5.7.2 Documentation des fonctions

5.7.2.1 countCustomMaps()

```
int countCustomMaps (
            char * name_directory)
```

La fonction permet de compter le nombre de maps custom dans le fichier des maps.

Paramètres

name_directory	Le nom du dossier contenant les maps. return Le nombre de maps custom.

Voici le graphe des appelants de cette fonction :

5.7.2.2 fileToTab2D()

```
vect * fileToTab2D (
            const char * name_file,
            char ** tab,
            const unsigned N,
             vect * player,
            int * nbr_targets)
```

La fonction permet de stocker la zone de jeu en fonction de la lecture d'un fichier.

Paramètres

name_file	Le nom du fichier a ouvrir.
tab	Le tableau 2D carre du plateau de jeu a remplir.
N	La taille de tab.
player	Les coordonnée du joueur que le programe vas trouvé.
nbr_targets	Le nombre de points d'interer trouver.

Renvoie

Vect La fonction renvoie le tableau des coordonnée des points d'interer.

5.7.2.3 generatorMenu()

```
char ** generatorMenu (
            char * name_directory,
             vect * size,
             vect * pos_player)
```

La fonction permet de creer la zone de jeu du menu en fonction du nombre de maps custom.

Paramètres

name_directory	Le nom du dossier contenant les maps.
size	La taille de la zone de jeu.
pos_player	La position du joueur dans le menu.

Voici le graphe d'appel pour cette fonction :

5.7.2.4 save_grid_to_file()

```
void save_grid_to_file (
            int filename,
            char ** tab2D)
```

La fonction permet de retranscrire un niveau creer du tableau vers le fichier.txt.

Paramètres

filename	Le nom du fichier a remplir.
tab2d	Le tableau 2d carre.