Sokoban

Généré par Doxygen 1.12.0

Chapitre 1

Index des structures de données

1.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 3

Documentation des structures de données

3.1 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

3.2 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

3.3 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

Chapitre 4

Documentation des fichiers

4.1 display.h

```
00001 #ifndef DISPLAY_H
00002 #define DISPLAY_H
00003
00004 #include "../include/function.h"
00005 #include <SDL2/SDL.h>
00006 #include <SDL2/SDL_ttf.h>
00007
00008 void screenDisplay (char **tab, int size);
00009 int getMaxSize (dis display_user);
00010 void displayImage (SDL_Renderer *renderer, SDL_Texture *texture, vect pos,
00011                    int size);
00012 void initSDL (dis *display_user);
00013 void screenDisplayGameSDL (char **tab, dis *display_user);
00014 void displayTextSDL(dis *display_user,char *text, vect coor, vect size, int font_size);
00015
00016
00017 #endif // !DISPLAY_H
```

4.2 function.h

```
00001 #ifndef FONCTION_H
00002 #define FONCTION_H
00003
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_render.h>
00006 #include <SDL2/SDL_video.h>
00007 #include <stdbool.h>
00008 #include <stdlib.h>
00009 #include <time.h>
00010
00011 #define EMPTY 0
00012 #define WALL 1
00013 #define BOX 2
00014 #define TARGET 3
00015 #define BOX_ON_TARGET 4
00016 #define PLAYER 5
00017 #define PLAYER_ON_TARGET 6
00018
00019 typedef struct Vecteur
00020 {
00021     int x;
00022     int y;
00023 } vect;
00024
00025 typedef struct Score
00026 {
00027     time_t before;
00028     time_t after;
00029     unsigned int move_player;
00030     unsigned int move_box;
00031 } score;
00032
00033 typedef struct essential_sdl
00034 {
```

```
00035     SDL_Window *window;
00036     SDL_Renderer *renderer;
00037     unsigned int size_window;
00038     unsigned int size_box;
00039     unsigned int size_menu;
00040 } dis;
00041
00042 char **creatArea2D (const unsigned int N);
00043 void free2D (char **tab, int N);
00044 void screenDisplay (char **tab, int size);
00045 bool isWin (char **tab2d, vect *targets, int nbr_targets);
00046 bool islose (char **tab2d, const int N);
00047 bool blockBox (char **tab2d, vect box_coor);
00048 vect plusVect (vect one, vect two);
00049 void inGameLoop (char **tab2d, int N, vect *playerPos,
00050                  vect *targets, int nbr_targets, dis *display_user, score *score_user);
00051 char *timeToText(time_t time);
00052
00053 #endif // FONCTION_H
```

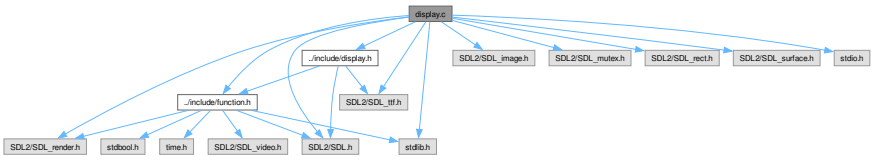
4.3 read.h

```
00001 #ifndef READ_H
00002 #define READ_H
00003 #include "function.h"
00004 vect *fileToTab2D (const char *name_file, char **tab,
00005                    const unsigned N, vect *player, int *nbr_targets);
00006 #endif // !READ_H
```

4.4 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_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 size)

La fonction permet d'afficher simplement le plateau de jeu dans le terminal.

— void **screenDisplayGameSDL** (char **tab, **dis** *display_user)

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.

4.4.1 Documentation des fonctions

4.4.1.1 displayImage()

```
void displayImage (
            SDL_Renderer * renderer,
            SDL_Texture * texture,
             vect pos,
            int size)
```

Cette fonction affiche l'image dans la fenetre de l'utilisateur.

Paramètres

renderer	Le renderer de l'utilisateur.
texture	La texture de l'image à appliquer.
pos	La position de l'image à afficher.
size	La taile de l'image.

Renvoie

Void

Voici le graphe des appelants de cette fonction :

4.4.1.2 displayTextSDL()

```
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.

Paramètres

display_user	Tous les éléments SDL de l'utilisateur.
text	Le string à afficher.
coor	Les coordonnée du texte.
size	La taille du texte.
font_size	La taille de la font.

Renvoie

Void

4.4.1.3 getMaxSize()

```
int getMaxSize (
              dis display_user)
```

Taille de l'ecrant carre en fonction de l'ecrant du joueur avec une marge.

Paramètres

display_user	Qui serra modifier pour stoquer les information.

Renvoie

La taille max pour la fenetre de l'utilisateur.

4.4.1.4 initSDL()

```
void initSDL (
              dis * display_user)
```

Initialise SDL.

Paramètres

display_user	Stockage d'éléments SDL.

Renvoie

Void

4.4.1.5 screenDisplay()

```
void screenDisplay (
             char ** tab,
             int size)
```

La fonction permet d'afficher simplement le plateau de jeu dans le terminal.

Paramètres

tab	Le tableau 2d carre du plateau.
size	La taille du plateau.

Renvoie

Void

4.4.1.6 screenDisplayGameSDL()

```
void screenDisplayGameSDL (
            char ** tab,
             dis * display_user)
```

La fonction affiche a l'aide de SDL la zone de jeu.

Paramètres

tab	Le tableau 2d de la zone de jeu.
display_user	La structure qui possede tous ce qu'il faut pour l'affichage SDL

Renvoie

Void

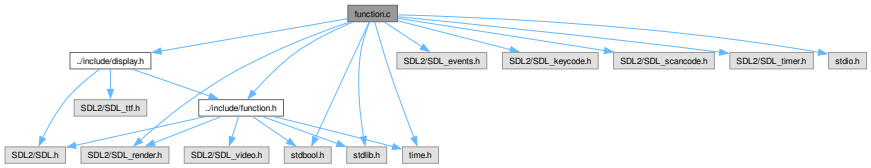
Voici le graphe d'appel pour cette fonction :

4.5 Référence du fichier function.c

```
#include "../include/function.h"
#include "../include/display.h"
#include <SDL2/SDL_events.h>
#include <SDL2/SDL_keycode.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 unsigned int N)

Cette fonction permet de creer une liste 2D carre.

— void **free2D** (char **tab, int N)

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.

— void **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.

— void **inGameLoop** (char **tab2d, int N, **vect** *playerPos, **vect** *targets, int nbr_targets, **dis** *display_user, **score** *score_user)

La fonction permet de faire la boucle de jeu.

— 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)

4.5.1 Description détaillée

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

4.5.2 Documentation des fonctions

4.5.2.1 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.

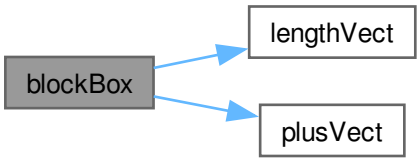
Paramètres

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

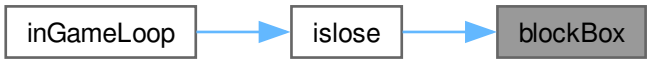
Renvoie

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 :



4.5.2.2 canIGoDirection()

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

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

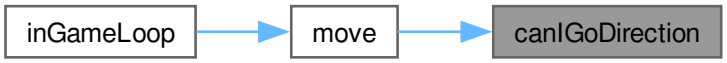
Paramètres

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 joeuur pousse une boite sur un point d'interer et que le joueu est aussi sur un point d'interer.

Voici le graphe des appelants de cette fonction :



4.5.2.3 creatArea2D()

```
char ** creatArea2D (
            const unsigned int N)
```

Cette fonction permet de creer une liste 2D carre.

Paramètres

N	La valeur N est le nombre d'élément dans le tableau.

Renvoie

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

4.5.2.4 free2D()

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

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

Paramètres

tab	Le tableau 2D.
N	Le nombre d'éléments.

Renvoie

Void.

4.5.2.5 inGameLoop()

```
void inGameLoop (
             char ** tab2d,
             int N,
              vect * playerPos,
              vect * targets,
             int nbr_targets,
              dis * display_user,
              score * score_user)
```

La fonction permet de faire la boucle de jeu.

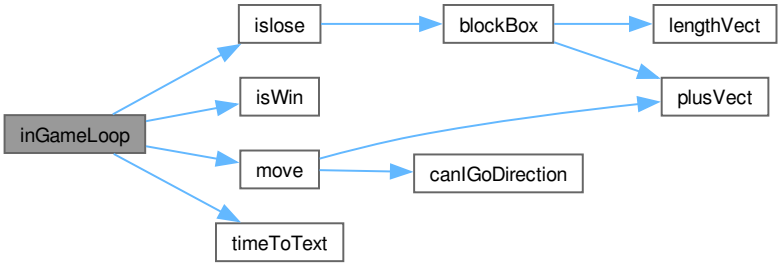
Paramètres

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.

Renvoie

Void

Voici le graphe d'appel pour cette fonction :



4.5.2.6 islose()

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

La fonction renvois si la partie est perdante.

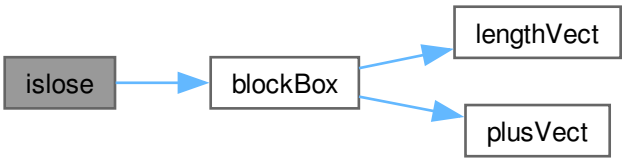
Paramètres

tab2d	Le tableau 2d carre du plateau de jeu.
N	Le nombre d'éléments dans le tab2d.

Renvoie

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

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.5.2.7 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 .

Voici le graphe des appelants de cette fonction :

4.5.2.8 move()

```
void 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.

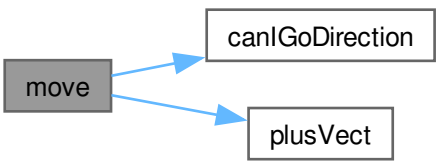
Paramètres

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

Void

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.5.2.9 plusVect()

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

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

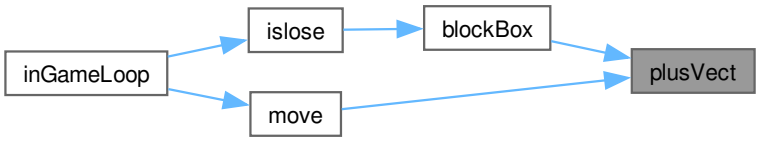
Paramètres

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 :



4.5.2.10 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

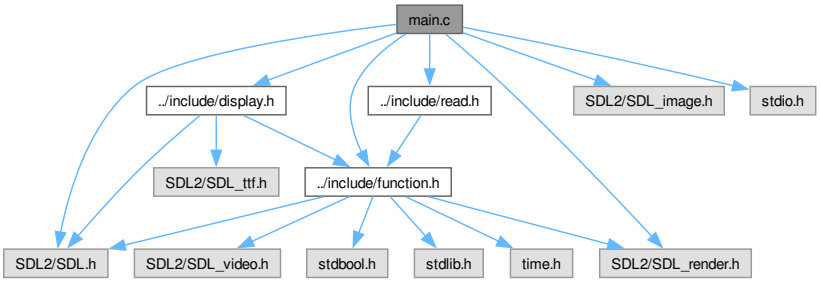
char Le string du texte.

Voici le graphe des appelants de cette fonction :

4.6 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 <stdio.h>
```

Graphe des dépendances par inclusion de main.c:



Macros

- #define **SIZE_PLAY** 19
- #define **SIZE_MENU** 200;

Fonctions

— int **main** ()

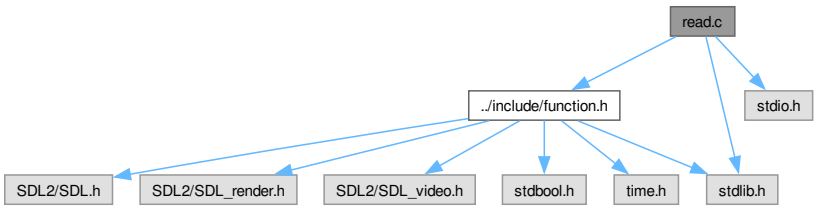
4.6.1 Description détaillée

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

4.7 Référence du fichier read.c

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

4.7.1 Description détaillée

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

4.7.2 Documentation des fonctions

4.7.2.1 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.

Index