

Sokoban

Généré par Doxygen 1.13.1



|                                                       |          |
|-------------------------------------------------------|----------|
| <b>1 Rapport</b>                                      | <b>1</b> |
| 1.1 Décision de structure . . . . .                   | 1        |
| 1.2 Comment lancé le jeu ? . . . . .                  | 1        |
| <b>2 Index des structures de données</b>              | <b>3</b> |
| 2.1 Structures de données . . . . .                   | 3        |
| <b>3 Index des fichiers</b>                           | <b>5</b> |
| 3.1 Liste des fichiers . . . . .                      | 5        |
| <b>4 Documentation des structures de données</b>      | <b>7</b> |
| 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        |
| <b>5 Documentation des fichiers</b>                   | <b>9</b> |
| 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

## Rapport

Nous avons choisi d'utiliser doxygen pour faire une documentation de notre code. Nous vous rendons donc notre rapport dans cette documentation, donc la majorité de ce pdf est en faite la documentation de notre code.

### 1.1 Décision de structure

Pour commencer on a pris comme structure pour la zone de jeu un tableau 2d de `char`. Nous avons pris des `char` car on sais qu'on a pas besoin de plus de possibilité de nombre que un octet. On a fais des `define` qui permet de definir quel nombre correspond à quel objet dans le jeux (joueur , caisse, mur, ...).

Nous avons ensuite créer une structure `coor` qui est une structure representant des coordonnées x et y. Cela est pratique pour naviger dans le tableau 2d par exemple.

On a fais une structure de score qui prend comme variable ce qu'il y a besoin pour calculer le score a la fin de la partie. Il y a le temps avant la partie et apres (pour obtenir le temps passer dans la partie), ensuite on a le nombre de déplacement fais par le joueur ainsi que les déplacement fais par les caisses.

Pour finir on a fais une structure `dis` qui est enfaite la structure possédant tout ce qu'il faut pour faire l'affichage SDL. En effet on a la `window`, le `renderer`, on a la taille de la `window`, la taille des boites ainsi que la taille du menu.

### 1.2 Comment lancé le jeu ?

Il vous faut installer SDL2, SDL2 Mixer, SDL2 Image, SDL2 ttf, make, gcc.

```
make all
./sokoban
```



## Chapter 2

# Index des structures de données

### 2.1 Structures de données

Liste des structures de données avec une brève description :

|                      |   |
|----------------------|---|
| <b>essential_sdl</b> | 7 |
| <b>Score</b>         | 7 |
| <b>Vecteur</b>       | 7 |



## Chapter 3

# Index des fichiers

### 3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

|                   |    |
|-------------------|----|
| <b>display.h</b>  | 9  |
| <b>function.h</b> | 9  |
| <b>read.h</b>     | 10 |
| <b>display.c</b>  | 10 |
| <b>function.c</b> | 12 |
| <b>main.c</b>     | 19 |
| <b>read.c</b>     | 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

```
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
00009
00010 extern unsigned int seed;
00011 void screenDisplay (char **tab, int x, int y);
00012 int getMaxSize (dis display_user);
00013 void displayImage (SDL_Renderer *renderer, SDL_Texture *texture, vect pos,
00014                  vect size);
00015 void initSDL (dis *display_user);
00016 void displayTextSDL(dis *display_user,char *text, vect coor, vect size, int font_size);
00017 void screenDisplayGameSDL (char **tab,vect dim_tab, dis *display_user, vect *player_pos, int fov, vect
    direction);
00018 void backgroundDisplay(dis *display_user,int bg);
00019 void playAudio(int sfx);
00020
00021 #endif // !DISPLAY_H
```

### 5.2 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 1
00012 #define WALL 0
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 #define BUTTON 7
00019 #define PLAYER_ON_BUTTON 8
00020 #define BUTTON_CUSTOM 9
00021
00022 typedef struct Vecteur
00023 {
00024     int x;
00025     int y;
00026 } vect;
00027
00028 typedef struct Score
```



## 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'ecrnat carre en fonction de l'ecrnat du joueur avec une marge.*
- void **initSDL** ( **dis** \*display\_user)  
*Initialise SDL.*
- void **displayImage** (SDL\_Renderer \*renderer, SDL\_Texture \*texture, **vect** pos, **vect** 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

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <i>display_user</i> | Tout les information du display de l'utilisateur utile. |
| <i>bg</i>           | 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 fonction permet de liberer l'espace du tableau 2D de char.*
- char **canGoDirection** (char valueOfNCase, char valueOfNPlusOneCase)  
*La fonction permet de savoir si le joueur peut se déplacer dans une direction.*
- int **move** (char \*\*tab, **vect** \*playerPos, **vect** direction, **score** \*score\_user)  
*Cette fonction effectue les déplacements du joueur et des boîtes 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, **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 titre screen.*
- bool **isWin** (char \*\*tab2d, **vect** \*targets, int nbr\_targets)  
*Cette fonction vérifie si la partie est gagnante.*
- bool **islose** (char \*\*tab2d, const int N)  
*La fonction renvoie 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)  
*Renvoie la longueur Manhattan.*
- bool **blockBox** (char \*\*tab2d, **vect** box\_coor)  
*La fonction permet de savoir si une boîte est dans une situation où le joueur ne pourra pas la débloquent.*
- char \* **timeToText** (time\_t time)  
*Cette fonction renvoie transforme le format 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)

### 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 pourra pas la debloqué.

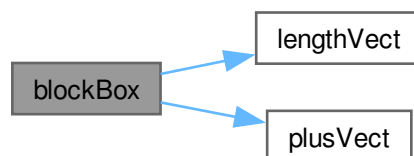
##### Paramètres

|                 |                                                  |
|-----------------|--------------------------------------------------|
| <i>tab2d</i>    | Le tableau 2D carre du plateau de jeu.           |
| <i>box_coor</i> | 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**

|                            |                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------|
| <i>valueOfNCase</i>        | La valeur de la case dans la direction que le joueur veut aller.                       |
| <i>valueOfNPlusOneCase</i> | 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 peut 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 peut 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**

|          |                    |
|----------|--------------------|
| <i>x</i> | Nombre de ligne.   |
| <i>y</i> | 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 fonction permet de liberer l'espace du tableau 2D de char.

**Paramètres**

|            |                     |
|------------|---------------------|
| <i>tab</i> | Le tableau 2D.      |
| <i>x</i>   | Le nombre de ligne. |

**Renvoie**

Void.

### 5.5.2.5 inEditorLoop()

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

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

#### Paramètres

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

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

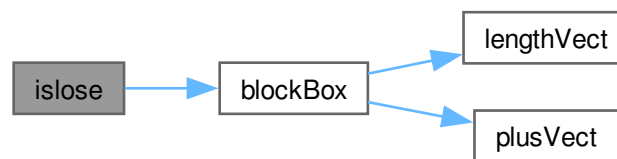
**Paramètres**

|              |                                                             |
|--------------|-------------------------------------------------------------|
| <i>tab2d</i> | Le tableau 2d carre du plateau de jeu.                      |
| <i>N</i>     | 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**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <i>tab2d</i>       | Le tableau 2D du jeu.                                   |
| <i>targets</i>     | Le tableau de toute les positions des points d'interer. |
| <i>nbr_targets</i> | 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 déplacements du joueur et des boites en fonction de la situation.

#### Paramètres

|                   |                                                                      |
|-------------------|----------------------------------------------------------------------|
| <i>tab</i>        | Le tableau 2D du plateau de jeu.                                     |
| <i>playerpos</i>  | La position actuel du joueur.                                        |
| <i>direction</i>  | La direction que le joueur veut effectuer.                           |
| <i>score_user</i> | 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

|                     |                        |
|---------------------|------------------------|
| <i>player_score</i> | 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

|            |                   |
|------------|-------------------|
| <i>one</i> | Premier vecteur.  |
| <i>two</i> | 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 renvoie transforme le forma time en texte. (min:sec)

##### Paramètres

|             |                                |
|-------------|--------------------------------|
| <i>time</i> | Le temps qu'on veut convertir. |
|-------------|--------------------------------|

##### 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

|                     |                                                |
|---------------------|------------------------------------------------|
| <i>display_user</i> | 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

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <i>display_user</i> | Tout les information du display de l'utilisateur utile. |
| <i>win</i>          | 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 permettant 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 créer 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 créer du tableau vers le fichier.txt.*

### 5.7.1 Description détaillée

Ce fichier est le programme qui lit d'autre fichier, notamment 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

|                       |                                                                        |
|-----------------------|------------------------------------------------------------------------|
| <i>name_directory</i> | 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

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <i>name_file</i>   | Le nom du fichier a ouvrir.                            |
| <i>tab</i>         | Le tableau 2D carre du plateau de jeu a remplir.       |
| <i>N</i>           | La taille de tab.                                      |
| <i>player</i>      | Les coordonnées du joueur que le programme vas trouvé. |
| <i>nbr_targets</i> | Le nombre de points d'interer trouver.                 |

##### Renvoie

Vect La fonction renvoie le tableau des coordonnées 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

|                       |                                       |
|-----------------------|---------------------------------------|
| <i>name_directory</i> | Le nom du dossier contenant les maps. |
| <i>size</i>           | La taille de la zone de jeu.          |
| <i>pos_player</i>     | 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 créer du tableau vers le fichier.txt.

## Paramètres

|                 |                              |
|-----------------|------------------------------|
| <i>filename</i> | Le nom du fichier a remplir. |
| <i>tab2d</i>    | Le tableau 2d carre.         |

