

Tetris

1.1

Généré par Doxygen 1.8.1.2

Lundi Mai 25 2015 04 :49 :13

Table des matières

1	Index des structures de données	1
1.1	Structures de données	1
2	Index des fichiers	3
2.1	Liste des fichiers	3
3	Documentation des structures de données	5
3.1	Référence de la structure Bouton	5
3.1.1	Description détaillée	5
3.1.2	Documentation des champs	5
3.1.2.1	brille	5
3.1.2.2	on_off	5
3.1.2.3	text	5
3.2	Référence de la structure Game	6
3.2.1	Description détaillée	6
3.2.2	Documentation des champs	6
3.2.2.1	bag	6
3.2.2.2	bool	6
3.2.2.3	current	6
3.2.2.4	game_speed	6
3.2.2.5	game_speed_base	7
3.2.2.6	hold	7
3.2.2.7	level	7
3.2.2.8	lineCleared	7
3.2.2.9	next	7
3.2.2.10	score	7
3.2.2.11	tab	7
3.2.2.12	turn	7
3.3	Référence de la structure param	7
3.3.1	Description détaillée	8
3.3.2	Documentation des champs	8
3.3.2.1	game_speed	8

3.3.2.2	hold	8
3.3.2.3	level	8
3.3.2.4	music_n	8
3.3.2.5	next	8
3.3.2.6	pause	8
3.3.2.7	theme	8
3.4	Référence de la structure tetri	8
3.4.1	Description détaillée	9
3.4.2	Documentation des champs	9
3.4.2.1	hold_possible	9
3.4.2.2	rotation	9
3.4.2.3	shape	9
3.4.2.4	visual	9
3.4.2.5	x_pos	9
3.4.2.6	y_pos	9
4	Documentation des fichiers	11
4.1	Référence du fichier def.h	11
4.1.1	Description détaillée	12
4.1.2	Documentation des macros	12
4.1.2.1	BUTTON_X_ALIGN	12
4.1.2.2	BUTTON_X_SIZE	12
4.1.2.3	BUTTON_X_TEXT	12
4.1.2.4	BUTTON_Y_ALIGN	12
4.1.2.5	BUTTON_Y_GAP	12
4.1.2.6	BUTTON_Y_SIZE	12
4.1.2.7	BUTTON_Y_TEXT	12
4.1.2.8	COEF_DIFFUCLTY	12
4.1.2.9	GAME_SIZE_X	12
4.1.2.10	GAME_SIZE_Y	12
4.1.2.11	HIDEN_Y_CASE	12
4.1.2.12	KEY_LAG	12
4.1.2.13	POS_X_NEXT	12
4.1.2.14	POS_X_PLA	12
4.1.2.15	POS_X_SCO	12
4.1.2.16	POS_Y_NEXT	12
4.1.2.17	POS_Y_PLA	12
4.1.2.18	POS_Y_SCO	12
4.1.2.19	TET_SIZE	12
4.1.2.20	WINDOW_SIZE_X	12

4.1.2.21	WINDOW_SIZE_Y	13
4.1.3	Documentation des définitions de type	13
4.1.3.1	Bouton	13
4.1.3.2	Game	13
4.1.3.3	param	13
4.1.3.4	tetri	13
4.2	Référence du fichier game.c	13
4.2.1	Description détaillée	13
4.2.2	Documentation des fonctions	14
4.2.2.1	clear_line	14
4.2.2.2	hold	14
4.2.2.3	initialiseGame	14
4.2.2.4	initialiseTab	14
4.2.2.5	is_collison	14
4.2.2.6	is_complete_line	14
4.2.2.7	is_ground	14
4.2.2.8	legalPosition	15
4.2.2.9	levelUp	15
4.2.2.10	make_tet	15
4.2.2.11	nextTetri	15
4.2.2.12	print_tet	15
4.2.2.13	randomBag	16
4.2.2.14	reached_top	16
4.2.2.15	rotate_tet	16
4.2.3	Documentation des variables	16
4.2.3.1	tetrinos	16
4.3	Référence du fichier game.h	16
4.3.1	Documentation des fonctions	17
4.3.1.1	clear_line	17
4.3.1.2	hold	17
4.3.1.3	initialiseGame	17
4.3.1.4	initialiseTab	17
4.3.1.5	is_collison	17
4.3.1.6	is_complete_line	17
4.3.1.7	is_ground	18
4.3.1.8	legalPosition	18
4.3.1.9	levelUp	18
4.3.1.10	make_tet	18
4.3.1.11	nextTetri	18
4.3.1.12	print_tet	19

4.3.1.13	randomBag	19
4.3.1.14	reached_top	19
4.3.1.15	rotate_tet	19
4.4	Référence du fichier main.c	19
4.4.1	Documentation des fonctions	19
4.4.1.1	main	19
4.5	Référence du fichier menu.c	19
4.5.1	Description détaillée	20
4.5.2	Documentation des fonctions	20
4.5.2.1	game	20
4.5.2.2	highscore_menu	20
4.5.2.3	main_menu	20
4.5.2.4	nameinput_menu	20
4.5.2.5	option_menu	20
4.5.2.6	option_menu_general	20
4.5.2.7	option_menu_vfx_sfx	21
4.5.2.8	print_highscore	21
4.5.2.9	read_highscore	21
4.5.2.10	reset_highscore	21
4.5.2.11	validate_highscore	21
4.6	Référence du fichier menu.h	21
4.6.1	Description détaillée	22
4.6.2	Documentation des fonctions	22
4.6.2.1	game	22
4.6.2.2	highscore_menu	22
4.6.2.3	main_menu	22
4.6.2.4	nameinput_menu	22
4.6.2.5	option_menu_general	22
4.6.2.6	print_highscore	22
4.6.2.7	read_highscore	22
4.6.2.8	reset_highscore	22
4.6.2.9	validate_highscore	22
4.7	Référence du fichier SFX.c	23
4.7.1	Description détaillée	23
4.7.2	Documentation des fonctions	23
4.7.2.1	music	23
4.7.2.2	stop_music	23
4.8	Référence du fichier SFX.h	23
4.8.1	Description détaillée	23
4.8.2	Documentation des fonctions	23

4.8.2.1	music	23
4.8.2.2	stop_music	24
4.9	Référence du fichier visual.c	24
4.9.1	Description détaillée	24
4.9.2	Documentation des fonctions	24
4.9.2.1	blit_tab	24
4.9.2.2	blit_tet	24
4.9.2.3	pause	24
4.9.2.4	write_text	24
4.10	Référence du fichier visual.h	25
4.10.1	Description détaillée	25
4.10.2	Documentation des fonctions	25
4.10.2.1	blit_tab	25
4.10.2.2	blit_tet	26
4.10.2.3	pause	26
4.10.2.4	write_text	26

Chapitre 1

Index des structures de données

1.1 Structures de données

Liste des structures de données avec une brève description :

Bouton	Structure associé au bouton	5
Game	Structure associé à une partie en cours	6
param	Structure definissant les paramtres l du Jeu	7
tetri	Structure stockant un tetrino	8

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

def.h	Fichier effectuant les différentes définitions	11
game.c	Fichier regroupant les fonctions du jeu	13
game.h	16
main.c	19
menu.c	Fichier regroupant les différents menus	19
menu.h	Fichier regroupant les différents menus	21
SFX.c	23
SFX.h	23
visual.c	Fichier regroupant les fonctions graphiques	24
visual.h	Fichier regroupant les fonctions graphiques	25

Chapitre 3

Documentation des structures de données

3.1 Référence de la structure Bouton

Structure associé au bouton.

```
#include <def.h>
```

Champs de données

- int [brille](#)
Boolean indiquant si le bonton en surbrillance.
- char [text](#) [30]
Texte.
- int [on_off](#)
On ou off.

3.1.1 Description détaillée

Structure associé au bouton.

3.1.2 Documentation des champs

3.1.2.1 int [brille](#)

Boolean indiquant si le bonton en surbrillance.

3.1.2.2 int [on_off](#)

On ou off.

3.1.2.3 char [text](#)[30]

Texte.

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

- [def.h](#)

3.2 Référence de la structure Game

Structure associé à une partie en cours.

```
#include <def.h>
```

Champs de données

- int `tab` [`GAME_SIZE_X` * `GAME_SIZE_Y`]
Plateau de jeu.
- `tetri` * `current`
Tetrino en mouvement.
- `tetri` * `next`
Tetrino suivant.
- `tetri` * `hold`
Tetrino retenu.
- float `game_speed`
Vitesse actuelle du jeu.
- float `game_speed_base`
Vitesse de base du jeu.
- int `level`
Niveau.
- int `lineCleared`
Ligne détruite au niveau en cours.
- int `bag` [14]
pieces suivant
- int `bool`
Partie du sac de pieces suivantes en cours de parcours.
- int `score`
Score.
- int `turn`
N° Du tour de jeu.

3.2.1 Description détaillée

Structure associé à une partie en cours.

3.2.2 Documentation des champs

3.2.2.1 int bag[14]

pieces suivant

3.2.2.2 int bool

Partie du sac de pieces suivantes en cours de parcours.

3.2.2.3 tetri* current

Tetrino en mouvement.

3.2.2.4 float game_speed

Vitesse actuelle du jeu.

3.2.2.5 float game_speed_base

Vitesse de base du jeu.

3.2.2.6 tetri* hold

Tetrino retenu.

3.2.2.7 int level

Niveau.

3.2.2.8 int lineCleared

Ligne détruite au niveau en cours.

3.2.2.9 tetri* next

Tetrino suivant.

3.2.2.10 int score

Score.

3.2.2.11 int tab[GAME_SIZE_X * GAME_SIZE_Y]

Plateau de jeu.

3.2.2.12 int turn

N° Du tour de jeu.

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

– [def.h](#)

3.3 Référence de la structure param

Structure définissant les paramètres du Jeu.

```
#include <def.h>
```

Champs de données

- float [game_speed](#)
Définit la vitesse de base du jeu.
- int [hold](#)
Autorise le hold.
- int [pause](#)
Autorise la mise en pause.
- int [level](#)
Sélection du niveau initial.
- int [next](#)
Autorise la prévisualisation de la pièce suivante.

- int `music_n`
Fond sonore.
- int `theme`
Fond écran.

3.3.1 Description détaillée

Structure définissant les paramètres I du Jeu.

3.3.2 Documentation des champs

3.3.2.1 float `game_speed`

Définis la vitesse de base du jeu.

3.3.2.2 int `hold`

Autorise le hold.

3.3.2.3 int `level`

Sélection du niveau initial.

3.3.2.4 int `music_n`

Fond sonore.

3.3.2.5 int `next`

Autorise la prévisualisation de la pièce suivante.

3.3.2.6 int `pause`

Autorise la mise en pause.

3.3.2.7 int `theme`

Fond écran.

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

- `def.h`

3.4 Référence de la structure `tetri`

Structure stockant un tetrino.

```
#include <def.h>
```


Champs de données

- int [visual](#) [4][4]
Tableau d'entien stokant le visuel du tetrino.
- int [shape](#)
entier stockant l'identifiant du tetrino
- int [rotation](#)
entier stockant l'angle de rotation du tetrino
- int [x_pos](#)
entier stickant la postition en x du tetrino
- int [y_pos](#)
entier stickant la postition en y du tetrino
- int [hold_possible](#)
boolean indiquant la possibilite ou non de hold la piece

3.4.1 Description détaillée

Structure stockant un tetrino.

3.4.2 Documentation des champs

3.4.2.1 int hold_possible

boolean indiquant la possibilite ou non de hold la piece

3.4.2.2 int rotation

entier stockant l'angle de rotation du tetrino

3.4.2.3 int shape

entier stockant l'identifiant du tetrino

3.4.2.4 int visual[4][4]

Tableau d'entien stokant le visuel du tetrino.

3.4.2.5 int x_pos

entier stickant la postition en x du tetrino

3.4.2.6 int y_pos

entier stickant la postition en y du tetrino

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

- [def.h](#)

Chapitre 4

Documentation des fichiers

4.1 Référence du fichier def.h

Fichier effectuant les différentes définitions.

```
#include <SDL/SDL.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
```

Structures de données

- struct `tetri`
Structure stockant un tetrino.
- struct `Game`
Structure associée à une partie en cours.
- struct `Bouton`
Structure associée au bouton.
- struct `param`
Structure définissant les paramètres du Jeu.

Macros

- #define `WINDOW_SIZE_X` 600
- #define `WINDOW_SIZE_Y` 600
- #define `BUTTON_X_ALIGN` 100
- #define `BUTTON_Y_ALIGN` 30
- #define `BUTTON_Y_TEXT` 50
- #define `BUTTON_X_TEXT` 0
- #define `BUTTON_Y_GAP` 150
- #define `BUTTON_X_SIZE` 450
- #define `BUTTON_Y_SIZE` 100
- #define `COEF_DIFFICULTY` 0.9
- #define `GAME_SIZE_X` 10
- #define `GAME_SIZE_Y` 21
- #define `TET_SIZE` 20
- #define `HIDDEN_Y_CASE` 2
- #define `POS_X_PLA` 200
- #define `POS_Y_PLA` 50
- #define `POS_X_SCO` 200
- #define `POS_Y_SCO` 50
- #define `POS_X_NEXT` 200
- #define `POS_Y_NEXT` 50
- #define `KEY_LAG` 100

Définitions de type

- typedef struct [tetri](#) [tetri](#)
- typedef struct [Game](#) [Game](#)
- typedef struct [Bouton](#) [Bouton](#)
- typedef struct [param](#) [param](#)

4.1.1 Description détaillée

Fichier effectuant les différentes définitions. Fichier effectuant les différentes définitions

4.1.2 Documentation des macros

4.1.2.1 `#define BUTTON_X_ALIGN 100`

4.1.2.2 `#define BUTTON_X_SIZE 450`

4.1.2.3 `#define BUTTON_X_TEXT 0`

4.1.2.4 `#define BUTTON_Y_ALIGN 30`

4.1.2.5 `#define BUTTON_Y_GAP 150`

4.1.2.6 `#define BUTTON_Y_SIZE 100`

4.1.2.7 `#define BUTTON_Y_TEXT 50`

4.1.2.8 `#define COEF_DIFFUCLTY 0.9`

4.1.2.9 `#define GAME_SIZE_X 10`

4.1.2.10 `#define GAME_SIZE_Y 21`

4.1.2.11 `#define HIDE_Y_CASE 2`

4.1.2.12 `#define KEY_LAG 100`

4.1.2.13 `#define POS_X_NEXT 200`

4.1.2.14 `#define POS_X_PLA 200`

4.1.2.15 `#define POS_X_SCO 200`

4.1.2.16 `#define POS_Y_NEXT 50`

4.1.2.17 `#define POS_Y_PLA 50`

4.1.2.18 `#define POS_Y_SCO 50`

4.1.2.19 `#define TET_SIZE 20`

4.1.2.20 `#define WINDOW_SIZE_X 600`

*Position X des boutons

4.1.2.21 #define WINDOW_SIZE_Y 600

*Position Y des boutons

4.1.3 Documentation des définitions de type

4.1.3.1 typedef struct Bouton Bouton

4.1.3.2 typedef struct Game Game

4.1.3.3 typedef struct param param

4.1.3.4 typedef struct tetri tetri

4.2 Référence du fichier game.c

Fichier regroupant les fonctions du jeu.

```
#include "def.h"
```

Fonctions

- void **rotate_tet** (tetri *tet)
Rotation dans le sens horraire du tetrino tet.
- void **randomBag** (Game *play)
Genere aleatoirement suivant le SRS systeme le sac de piece de jeu.
- int **is_collison** (int *play_tab, tetri *tet)
- int **is_ground** (int *play_tab, tetri *tet)
Verifie si il y eu contacte avec le bas du tablea.
- int **is_complete_line** (int *play_tab, int *result_tab)
Verifie si des ligne sont pleines, si c'est le cas ecrit 1 dans la case correspondante de result_tab et retourne le nombre de lignes completes.
- int **reached_top** (int *play_tab)
- void **clear_line** (int *play_tab, int ln_num)
Vide la ligne correspondante à ln_num.
- tetri * **make_tet** (int shape, int posx, int posy)
Genere un tetrino de forme shape à la position posx posy.
- void **print_tet** (tetri *tet, int *tab)
Incruste le tetrino dans le plateau de jeu.
- void **initialiseTab** (int *tab)
Initialise un tableau à deux dimension de la taille du plateau avec des 0.
- int **legalPosition** (Game *play, int posx, int posy, int rot)
Verifie qu'une position est legale.
- void **nextTetri** (Game *game)
Passe à la piece suivante.
- void **levelUp** (Game *game)
Effectue les operations lie au changement de niveau.
- void **hold** (Game *game)
Effectue les operations lie au hold d'une piece.
- void **initialiseGame** (Game *current_play)
Initialise le jeu.

Variables

- const int **tetrinos** [7][4][4]

4.2.1 Description détaillée

Fichier regroupant les fonctions du jeu. Fichier regroupant les fonctions du jeu

4.2.2 Documentation des fonctions

4.2.2.1 void clear_line (int * *play_tab*, int *ln_num*)

Vide la ligne correspondante à *ln_num*.

Paramètres

in	<i>play_tab</i>	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	<i>ln_num</i>	Numero de la ligne a vide

4.2.2.2 void hold (Game * *game*)

Effectue les operations lie au hold d'une piece.

Paramètres

in	<i>Game</i>	pointeur vers le jeu en cours
----	-------------	-------------------------------

4.2.2.3 void initialiseGame (Game * *game*)

Initialise le jeu.

Paramètres

in	<i>current_play</i>	pointeur vers le tableau du jeu en cours
----	---------------------	--

4.2.2.4 void initialiseTab (int * *tab*)

Initialise un tableau à deux dimension de la taille du plateau avec des 0.

Paramètres

in	<i>tab</i>	pointeur vers le tableau
----	------------	--------------------------

4.2.2.5 int is_collision (int * *play_tab*, tetri * *tet*)

4.2.2.6 int is_complete_line (int * *play_tab*, int * *result_tab*)

Verifie si des ligne sont pleines, si c'est le cas ecrit 1 dans la case correspondante de *result_tab* et retourne le nombre de lignes completes.

Paramètres

in	<i>play_tab</i>	Pointeur sur un objet de type tableau d'entier à deux dimension.
out	<i>result_tab</i>	Pointeur sur un objet de type tableau à une dimension de taille egale à la hauteur du plateau

Renvoie

retourne le nombre de ligne pleine

4.2.2.7 int is_ground (int * *play_tab*, tetri * *tet*)

Verifie si il y eu contacte avec le bas du tablea.

Paramètres

in	<i>play_tab</i>	pointeur vers le tableau du jeu en cours
in	<i>tet</i>	pointeur vers le tetrino sur lequel effectué le teste

Renvoie

Retourn 1 si il y a contact avec le bas du tableau 0 sinon

4.2.2.8 int legalPosition (Game * play, int posX, int posY, int rot)

Verifie qu'une position est legale.

Paramètres

in	<i>play</i>	pointeur vers une structure Game du jeu en cours
in	<i>posx</i>	position horizontale desire
in	<i>posy</i>	position verticale desire
in	<i>rot</i>	rotation desire

Renvoie

Retourn 1 si la position est legale 0 sinon

4.2.2.9 void levelUp (Game * game)

Effectue les operations lie au changement de niveau.

Paramètres

in	Game	pointeur vers le jeu en cours
----	----------------------	-------------------------------

4.2.2.10 tetri * make_tet (int shape, int posX, int posY)

Genere un tetrino de forme shape à la position posX posy.

Paramètres

in	<i>shape</i>	forme du tetrino
in	<i>posx</i>	position horizontale d'origine du tetrino
in	<i>posy</i>	position verticale d'origine du tetrino

Renvoie

retourne un pointeur vers un objet de type tetrino du tertino generer

4.2.2.11 void nextTetri (Game * game)

Passe à la piece suivante.

Paramètres

in	Game	pointeur vers le jeu en cours
----	----------------------	-------------------------------

4.2.2.12 void print_tet (tetri * tet, int * tab)

Incruste le tetrino dans le plateau de jeu.

Paramètres

in	tab	pointeur vers le tableau du jeu en cours
in	tet	pointeur vers le tetrino a incruste

4.2.2.13 void randomBag (Game * play)

Genere aleatoirement suivant le SRS systeme le sac de piece de jeu.

Paramètres

in	play	pointeur vers le jeu en cours
----	------	-------------------------------

4.2.2.14 int reached_top (int * play_tab)

4.2.2.15 void rotate_tet (tetri * tet)

Rotation dans le sens horraire du tetrino tet.

Paramètres

in	tet	Pointeur sur un objet de type tetrino
----	-----	---------------------------------------

4.2.3 Documentation des variables

4.2.3.1 const int tetrinos[7][4][4][4]

4.3 Référence du fichier game.h

```
#include "def.h"
```

Fonctions

- int [is_collison](#) (int *play_tab, [tetri](#) *tet)
- int [reached_top](#) (int *play_tab, [tetri](#) *tet)
Verifie si une piece a atteint le sommet de l'ecran.
- void [rotate_tet](#) ([tetri](#) *tet)
Rotation dans le sens horraire du tetrino tet.
- int [is_complete_line](#) (int *play_tab, int *result_tab)
Verifie si des ligne sont pleines, si c'est le cas ecrit 1 dans la case correspondante de result_tab et retourne le nombre de lignes completes.
- void [clear_line](#) (int *play_tab, int ln_num)
Vide la ligne correspondante à ln_num.
- [tetri](#) * [make_tet](#) (int shape, int posx, int posy)
Genere un tetrino de forme shape à la position posx posy.
- void [initialiseTab](#) (int *tab)
Initialise un tableau à deux dimension de la taille du plateau avec des 0.
- int [legalPosition](#) ([Game](#) *play, int posx, int posy, int rot)
Verifie qu'une position est legale.
- int [is_ground](#) (int *play_tab, [tetri](#) *tet)
Verifie si il y eu contacte avec le bas du tablea.
- void [print_tet](#) ([tetri](#) *tet, int *tab)
Incruste le tetrino dans le plateau de jeu.
- void [randomBag](#) ([Game](#) *play)
Genere aleatoirement suivant le SRS systeme le sac de piece de jeu.
- void [levelUp](#) ([Game](#) *game)
Effectue les operations lie au changement de niveau.
- void [hold](#) ([Game](#) *game)
Effectue les operations lie au hold d'une piece.

- void `nextTetri (Game *game)`
Passe à la piece suivante.
- void `initialiseGame (Game *current_play)`
Initialise le jeu.

4.3.1 Documentation des fonctions

4.3.1.1 void `clear_line (int * play_tab, int ln_num)`

Vide la ligne correspondante à `ln_num`.

Paramètres

<code>in</code>	<code>play_tab</code>	Pointeur sur un objet de type tableau d'entier à deux dimension.
<code>in</code>	<code>ln_num</code>	Numero de la ligne a vide

4.3.1.2 void `hold (Game * game)`

Effectue les operations lie au hold d'une piece.

Paramètres

<code>in</code>	<code>Game</code>	pointeur vers le jeu en cours
-----------------	-------------------	-------------------------------

4.3.1.3 void `initialiseGame (Game * current_play)`

Initialise le jeu.

Paramètres

<code>in</code>	<code>current_play</code>	pointeur vers le tableau du jeu en cours
-----------------	---------------------------	--

4.3.1.4 void `initialiseTab (int * tab)`

Initialise un tableau à deux dimension de la taille du plateau avec des 0.

Paramètres

<code>in</code>	<code>tab</code>	pointeur vers le tableau
-----------------	------------------	--------------------------

4.3.1.5 int `is_collison (int * play_tab, tetri * tet)`

4.3.1.6 int `is_complete_line (int * play_tab, int * result_tab)`

Verifie si des ligne sont pleines, si c'est le cas ecrit 1 dans la case correspondante de `result_tab` et retourne le nombre de lignes completes.

Paramètres

<code>in</code>	<code>play_tab</code>	Pointeur sur un objet de type tableau d'entier à deux dimension.
<code>out</code>	<code>result_tab</code>	Pointeur sur un objet de type tableau à une dimension de taille egale à la hauteur du plateau

Renvoie

retourne le nombre de ligne pleine

4.3.1.7 int is_ground (int * *play_tab*, tetri * *tet*)

Verifie si il y eu contacte avec le bas du tablea.

Paramètres

in	<i>play_tab</i>	pointeur vers le tableau du jeu en cours
in	<i>tet</i>	pointeur vers le tetrino sur lequel effectué le teste

Renvoie

Retourn 1 si il y a contact avec le bas du tableau 0 sinon

4.3.1.8 int legalPosition (Game * *play*, int *posx*, int *posy*, int *rot*)

Verifie qu'une position est legale.

Paramètres

in	<i>play</i>	pointeur vers une structure Game du jeu en cours
in	<i>posx</i>	position horizontale desire
in	<i>posy</i>	position verticale desire
in	<i>rot</i>	rotation desire

Renvoie

Retourn 1 si la position est legale 0 sinon

4.3.1.9 void levelUp (Game * *game*)

Effectue les operations lie au changement de niveau.

Paramètres

in	Game	pointeur vers le jeu en cours
----	----------------------	-------------------------------

4.3.1.10 tetri* make_tet (int *shape*, int *posx*, int *posy*)

Genere un tetrino de forme shape à la position posx posy.

Paramètres

in	<i>shape</i>	forme du tetrino
in	<i>posx</i>	position horizontale d'origine du tetrino
in	<i>posy</i>	position verticale d'origine du tetrino

Renvoie

retourne un pointeur vers un objet de type tetrino du tertino generer

4.3.1.11 void nextTetri (Game * *game*)

Passe à la piece suivante.

Paramètres

in	Game	pointeur vers le jeu en cours
----	----------------------	-------------------------------

4.3.1.12 void print_tet (tetri * tet, int * tab)

Incruste le tetrino dans le plateau de jeu.

Paramètres

in	tab	pointeur vers le tableau du jeu en cours
in	tet	pointeur vers le tetrino a incruste

4.3.1.13 void randomBag (Game * play)

Genere aleatoirement suivant le SRS systeme le sac de piece de jeu.

Paramètres

in	play	pointeur vers le jeu en cours
----	------	-------------------------------

4.3.1.14 int reached_top (int * play_tab, tetri * tet)

Verifie si une piece a atteint le sommet de l'ecran.

Paramètres

in	play_tab	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	tet	Pointeur sur un objet de type tetrino

Renvoie

1 si il y a eu une colision 0 sinon

4.3.1.15 void rotate_tet (tetri * tet)

Rotation dans le sens horraire du tetrino tet.

Paramètres

in	tet	Pointeur sur un objet de type tetrino
----	-----	---------------------------------------

4.4 Référence du fichier main.c

```
#include "menu.h"
```

Fonctions

– int [main](#) (int argc, char *argv[])

4.4.1 Documentation des fonctions

4.4.1.1 int main (int argc, char * argv[])

4.5 Référence du fichier menu.c

Fichier regroupant les differents menu.

```
#include "game.h"
#include "SFX.h"
#include "menu.h"
#include "visual.h"
```

Fonctions

- void `option_menu_general` (SDL_Surface *ecran, param *settings)
Affiche le menu d'option general : Permet de regler les options de jeux.
- void `option_menu_vfx_sfx` (SDL_Surface *ecran, param *settings)
- void `option_menu` (SDL_Surface *ecran, param *settings)
- int `validate_highscore` (char *name, int score)
Calcule la cle associé à un couple nom/score.
- int `reset_highscore` ()
Reinitialise les highscores.
- int `read_highscore` (char *text, int *score)
Lit les highscores du fichier highscore et remplit le tableau text avec les nom, et score avec les score.
- void `print_highscore` (char *name, int score)
Ecris le highscore dans le fichier a highscore.
- void `highscore_menu` (SDL_Surface *ecran)
Affiche les highscore.
- void `nameinput_menu` (SDL_Surface *ecran, char *name)
Affiche le menu de saisie du nom du joueur name doit etre un tableau de 10 char, le nom doit faire moins de 5 char.
- int `game` (param *settings, SDL_Surface *ecran)
Lance une partie sur ecran, avec les parametres settings.
- int `main_menu` ()
Gere le menu principal.

4.5.1 Description détaillée

Fichier regroupant les differents menu. Fichier regroupant les differents menu

4.5.2 Documentation des fonctions

4.5.2.1 int game (param * settings, SDL_Surface * ecran)

Lance une partie sur ecran, avec les parametres settings.

4.5.2.2 void highscore_menu (SDL_Surface * ecran)

Affiche les highscore.

4.5.2.3 main_menu ()

Gere le menu principal.

4.5.2.4 void nameinput_menu (SDL_Surface * ecran, char * name)

Affiche le menu de saisie du nom du joueur name doit etre un tableau de 10 char, le nom doit faire moins de 5 char.

4.5.2.5 void option_menu (SDL_Surface * ecran, param * settings)

4.5.2.6 void option_menu_general (SDL_Surface * ecran, param * settings)

Affiche le menu d'option general : Permet de regler les options de jeux.

Paramètres

in	<i>ecran</i>	Pointeur sur un objet de type SDL_Surface : Ecran d'affichage principale
in	<i>settings</i>	Reglage du jeu

4.5.2.7 void option_menu_vfx_sfx (SDL_Surface * *ecran*, param * *settings*)

4.5.2.8 void print_highscore (char * *name*, int *score*)

Ecris le highscore dans le fichier a highscore.

Paramètres

in	<i>name</i>	Pointeur sur un tableau de char de taille 5 representant le nom
in	<i>score</i>	score a inscrire

4.5.2.9 int read_highscore (char * *text*, int * *score*)

Lit les highscores du fichier highscore et remplit le tableau text avec les nom, et score avec les score.

4.5.2.10 int reset_highscore ()

Reinitialise les highscores.

4.5.2.11 validate_highscore (char * *name*, int *score*)

Calcule la cle associé à un couple nom/score.

4.6 Référence du fichier menu.h

Fichier regroupant les differents menu.

```
#include "def.h"
```

Fonctions

- void [option_menu_general](#) (SDL_Surface **ecran*, param **settings*)
Affiche le menu d'option general : Permet de regler les options de jeux.
- void [highscore_menu](#) (SDL_Surface **ecran*)
Affiche les highscore.
- void [print_highscore](#) (char **name*, int *score*)
Ecris le highscore dans le fichier a highscore.
- int [read_highscore](#) (char **text*, int **score*)
Lit les highscores du fichier highscore et remplit le tableau text avec les nom, et score avec les score.
- int [reset_highscore](#) ()
Reinitialise les highscores.
- int [validate_highscore](#) (char **name*, int *score*)
Calcule la cle associé à un couple nom/score.
- void [nameinput_menu](#) (SDL_Surface **ecran*, char **name*)
Affiche le menu de saisie du nom du joueur name doit etre un tableau de 10 char, le nom doit faire moins de 5 char.
- int [game](#) (param **settings*, SDL_Surface **ecran*)
Lance une partie sur ecran, avec les parametres settings.
- int [main_menu](#) ()
Gere le menu principal.

4.6.1 Description détaillée

Fichier regroupant les differents menu. Fichier regroupant les differents menu

4.6.2 Documentation des fonctions

4.6.2.1 `int game (param * settings, SDL_Surface * ecran)`

Lance une partie sur ecran, avec les parametres settings.

4.6.2.2 `void highscore_menu (SDL_Surface * ecran)`

Affiche les highscore.

4.6.2.3 `int main_menu ()`

Gere le menu principal.

4.6.2.4 `void nameinput_menu (SDL_Surface * ecran, char * name)`

Affiche le menu de saisie du nom du joueur name doit etre un tableau de 10 char, le nom doit faire moins de 5 char.

4.6.2.5 `void option_menu_general (SDL_Surface * ecran, param * settings)`

Affiche le menu d'option general : Permet de regler les options de jeux.

Paramètres

<code>in</code>	<code>ecran</code>	Pointeur sur un objet de type <code>SDL_Surface</code> : Ecran d'affichage principale
<code>in</code>	<code>settings</code>	Reglage du jeu

4.6.2.6 `void print_highscore (char * name, int score)`

Ecris le highscore dans le fichier a highscore.

Paramètres

<code>in</code>	<code>name</code>	Pointeur sur un tableau de char de taille 5 representant le nom
<code>in</code>	<code>score</code>	score a inscrire

4.6.2.7 `int read_highscore (char * text, int * score)`

Lit les highscores du fichier highscore et remplit le tableau text avec les nom, et score avec les score.

4.6.2.8 `int reset_highscore ()`

Reinitialise les highscores.

4.6.2.9 `int validate_highscore (char * name, int score)`

Calcule la cle associé à un couple nom/score.

4.7 Référence du fichier SFX.c

```
#include "def.h"
```

Fonctions

- void [music](#) (FMOD_SYSTEM **system, FMOD_SOUND **musique, [param](#) *settings)
Lance la musique indiquer dans la structure settings.
- void [stop_music](#) (FMOD_SYSTEM *system, FMOD_SOUND *musique)

4.7.1 Description détaillée

Fichier regroupant les fonctions de lecture audio.

4.7.2 Documentation des fonctions

4.7.2.1 [music](#) (FMOD_SYSTEM ** *system*, FMOD_SOUND ** *musique*, [param](#) * *settings*)

Lance la musique indiquer dans la structure settings.

Arrete la musique.

Paramètres

in	<i>settings</i>	pointeur vers une structure param, stock entre autre le morceau a jouer
in	<i>systeme</i>	pointeur vers pointeur de FMOD_SYSTEM
in	<i>musique</i>	pointeur vers pointeur de SOUND
in	<i>systeme</i>	pointeur vers pointeur de FMOD_SYSTEM
in	<i>musique</i>	pointeur vers pointeur de SOUND

4.7.2.2 void [stop_music](#) (FMOD_SYSTEM * *system*, FMOD_SOUND * *musique*)

4.8 Référence du fichier SFX.h

```
#include "def.h"
```

Fonctions

- void [music](#) (FMOD_SYSTEM **system, FMOD_SOUND **musique, [param](#) *settings)
Lance la musique indiquer dans la structure settings.
- void [stop_music](#) (FMOD_SYSTEM *system, FMOD_SOUND *musique)

4.8.1 Description détaillée

Fichier regroupant les fonctions de lecture audio.

4.8.2 Documentation des fonctions

4.8.2.1 void [music](#) (FMOD_SYSTEM ** *system*, FMOD_SOUND ** *musique*, [param](#) * *settings*)

Lance la musique indiquer dans la structure settings.

Arrete la musique.

Paramètres

in	<i>settings</i>	pointeur vers une structure param, stock entre autre le morceau a jouer
in	<i>systeme</i>	pointeur vers pointeur de FMOD_SYSTEM
in	<i>musique</i>	pointeur vers pointeur de SOUND
in	<i>systeme</i>	pointeur vers pointeur de FMOD_SYSTEM
in	<i>musique</i>	pointeur vers pointeur de SOUND

4.8.2.2 void stop_music (FMOD_SYSTEM * system, FMOD_SOUND * musique)

4.9 Référence du fichier visual.c

Fichier regroupant les fonctions graphiques.

```
#include "visual.h"
```

Fonctions

- void [pause](#) ()
Met le jeu en pause.
- void [blit_tab](#) (SDL_Surface **tet, SDL_Surface *screen, int *tab)
affiche le tableau de jeu
- SDL_Surface * [write_text](#) (char *text, int color_R, int color_G, int color_B, int size)
Genere une surface avec contenant un texte.
- void [blit_tet](#) (tetri *tetr, SDL_Surface **tet, SDL_Surface *screen, int pos_x, int pos_y)

4.9.1 Description détaillée

Fichier regroupant les fonctions graphiques. Fichier regroupant les fonctions graphiques

4.9.2 Documentation des fonctions

4.9.2.1 void blit_tab (SDL_Surface ** tet, SDL_Surface * screen, int * tab)

affiche le tableau de jeu

Paramètres

in	<i>tab</i>	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	<i>tet</i>	Pointeur sur tableau de SDL_Surfaces, representant les textures elementaires des tetrino
in	<i>screen</i>	Poiteur sur une SDL_Surface, ecran princial

4.9.2.2 void blit_tet (tetri * tetr, SDL_Surface ** tet, SDL_Surface * screen, int pos_x, int pos_y)

4.9.2.3 void pause ()

Met le jeu en pause.

4.9.2.4 SDL_Surface * write_text (char * text, int color_R, int color_G, int color_B, int size)

Genere une surface avec contenant un texte.

affiche a l'ecran un tetrino

Paramètres

in	<i>text</i>	Pointeur sur string, repretant la chaine de caractere
in	<i>color_R</i>	valeur de la couleur rouge
in	<i>color_V</i>	valeur de la couleur vert
in	<i>color_B</i>	valeur de la couleur bleu
in	<i>size</i>	taille du texte

Renvoie

un pointeur vers la surface generer

Paramètres

in	<i>tet</i>	texture des tetrino
in	<i>ecran</i>	Surface de l'ecran
in	<i>pos_x</i>	decalage selon l'axe x
in	<i>pos_y</i>	decalage selon l'axe y
in	<i>tetr</i>	pointeur vers la structure tetri

4.10 Référence du fichier visual.h

Fichier regroupant les fonctions graphiques.

```
#include "def.h"
```

Fonctions

- void **pause** ()
Met le jeu en pause.
- void **blit_tab** (SDL_Surface **tet, SDL_Surface *screen, int *tab)
affiche le tableau de jeu
- SDL_Surface * **write_text** (char *text, int color_R, int color_G, int color_B, int size)
Genere une surface avec contenant un texte.
- void **blit_tet** (tetri *tetr, SDL_Surface **tet, SDL_Surface *screen, int pos_x, int pos_y)

4.10.1 Description détaillée

Fichier regroupant les fonctions graphiques. Fichier regroupant les fonctions graphiques

4.10.2 Documentation des fonctions

4.10.2.1 void blit_tab (SDL_Surface ** tet, SDL_Surface * screen, int * tab)

affiche le tableau de jeu

Paramètres

in	<i>tab</i>	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	<i>tet</i>	Pointeur sur tableau de SDL_Surfaces, representant les textures elementaires des tetrino
in	<i>screen</i>	Poitteur sur une SDL_Surface, ecran pricipal

4.10.2.2 void blit_tet (tetri * *tetr*, SDL_Surface ** *tet*, SDL_Surface * *screen*, int *pos_x*, int *pos_y*)

4.10.2.3 void pause ()

Met le jeu en pause.

4.10.2.4 SDL_Surface* write_text (char * *text*, int *color_R*, int *color_G*, int *color_B*, int *size*)

Genere une surface avec contenant un texte.

affiche a l'ecran un tetrino

Paramètres

in	<i>text</i>	Pointeur sur string, representant la chaine de caractere
in	<i>color_R</i>	valeur de la couleur rouge
in	<i>color_V</i>	valeur de la couleur vert
in	<i>color_B</i>	valeur de la couleur bleu
in	<i>size</i>	taille du texte

Renvoie

un pointeur vers la surface generer

Paramètres

in	<i>tet</i>	texture des tetrino
in	<i>ecran</i>	Surface de l'ecran
in	<i>pos_x</i>	decalage selon l'axe x
in	<i>pos_y</i>	decalage selon l'axe y
in	<i>tetr</i>	pointeur vers la structure tetri

Index

BUTTON_X_ALIGN
def.h, [12](#)
BUTTON_X_SIZE
def.h, [12](#)
BUTTON_X_TEXT
def.h, [12](#)
BUTTON_Y_ALIGN
def.h, [12](#)
BUTTON_Y_GAP
def.h, [12](#)
BUTTON_Y_SIZE
def.h, [12](#)
BUTTON_Y_TEXT
def.h, [12](#)
bag
Game, [6](#)
blit_tab
visual.c, [24](#)
visual.h, [25](#)
blit_tet
visual.c, [24](#)
visual.h, [25](#)
bool
Game, [6](#)
Bouton, [5](#)
brille, [5](#)
def.h, [13](#)
on_off, [5](#)
text, [5](#)
brille
Bouton, [5](#)
COEF_DIFFUCLTY
def.h, [12](#)
clear_line
game.c, [14](#)
game.h, [17](#)
current
Game, [6](#)
def.h, [11](#)
BUTTON_X_ALIGN, [12](#)
BUTTON_X_SIZE, [12](#)
BUTTON_X_TEXT, [12](#)
BUTTON_Y_ALIGN, [12](#)
BUTTON_Y_GAP, [12](#)
BUTTON_Y_SIZE, [12](#)
BUTTON_Y_TEXT, [12](#)
Bouton, [13](#)
COEF_DIFFUCLTY, [12](#)
GAME_SIZE_X, [12](#)
GAME_SIZE_Y, [12](#)
Game, [13](#)
HIDEN_Y_CASE, [12](#)
KEY_LAG, [12](#)
POS_X_NEXT, [12](#)
POS_X_PLA, [12](#)
POS_X_SCO, [12](#)
POS_Y_NEXT, [12](#)
POS_Y_PLA, [12](#)
POS_Y_SCO, [12](#)
param, [13](#)
TET_SIZE, [12](#)
tetri, [13](#)
WINDOW_SIZE_X, [12](#)
WINDOW_SIZE_Y, [12](#)
GAME_SIZE_X
def.h, [12](#)
GAME_SIZE_Y
def.h, [12](#)
Game, [6](#)
bag, [6](#)
bool, [6](#)
current, [6](#)
def.h, [13](#)
game_speed, [6](#)
game_speed_base, [6](#)
hold, [7](#)
level, [7](#)
lineCleared, [7](#)
next, [7](#)
score, [7](#)
tab, [7](#)
turn, [7](#)
game
menu.c, [20](#)
menu.h, [22](#)
game.c, [13](#)
clear_line, [14](#)
hold, [14](#)
initialiseGame, [14](#)
initialiseTab, [14](#)
is_collison, [14](#)
is_complete_line, [14](#)
is_ground, [14](#)
legalPosition, [15](#)
levelUp, [15](#)
make_tet, [15](#)
nextTetri, [15](#)

- print_tet, 15
- randomBag, 16
- reached_top, 16
- rotate_tet, 16
- tetrinos, 16
- game.h, 16
 - clear_line, 17
 - hold, 17
 - initialiseGame, 17
 - initialiseTab, 17
 - is_collison, 17
 - is_complete_line, 17
 - is_ground, 17
 - legalPosition, 18
 - levelUp, 18
 - make_tet, 18
 - nextTetri, 18
 - print_tet, 18
 - randomBag, 19
 - reached_top, 19
 - rotate_tet, 19
- game_speed
 - Game, 6
 - param, 8
- game_speed_base
 - Game, 6
- HIDEN_Y_CASE
 - def.h, 12
- highscore_menu
 - menu.c, 20
 - menu.h, 22
- hold
 - Game, 7
 - game.c, 14
 - game.h, 17
 - param, 8
- hold_possible
 - tetri, 9
- initialiseGame
 - game.c, 14
 - game.h, 17
- initialiseTab
 - game.c, 14
 - game.h, 17
- is_collison
 - game.c, 14
 - game.h, 17
- is_complete_line
 - game.c, 14
 - game.h, 17
- is_ground
 - game.c, 14
 - game.h, 17
- KEY_LAG
 - def.h, 12
- legalPosition
 - game.c, 15
 - game.h, 18
- level
 - Game, 7
 - param, 8
- levelUp
 - game.c, 15
 - game.h, 18
- lineCleared
 - Game, 7
- main
 - main.c, 19
- main.c, 19
 - main, 19
- main_menu
 - menu.c, 20
 - menu.h, 22
- make_tet
 - game.c, 15
 - game.h, 18
- menu.c, 19
 - game, 20
 - highscore_menu, 20
 - main_menu, 20
 - nameinput_menu, 20
 - option_menu, 20
 - option_menu_general, 20
 - option_menu_vfx_sfx, 21
 - print_highscore, 21
 - read_highscore, 21
 - reset_highscore, 21
 - validate_highscore, 21
- menu.h, 21
 - game, 22
 - highscore_menu, 22
 - main_menu, 22
 - nameinput_menu, 22
 - option_menu_general, 22
 - print_highscore, 22
 - read_highscore, 22
 - reset_highscore, 22
 - validate_highscore, 22
- music
 - SFX.c, 23
 - SFX.h, 23
- music_n
 - param, 8
- nameinput_menu
 - menu.c, 20
 - menu.h, 22
- next
 - Game, 7
 - param, 8
- nextTetri
 - game.c, 15
 - game.h, 18

- on_off
 - Bouton, 5
- option_menu
 - menu.c, 20
- option_menu_general
 - menu.c, 20
 - menu.h, 22
- option_menu_vfx_sfx
 - menu.c, 21
- POS_X_NEXT
 - def.h, 12
- POS_X_PLA
 - def.h, 12
- POS_X_SCO
 - def.h, 12
- POS_Y_NEXT
 - def.h, 12
- POS_Y_PLA
 - def.h, 12
- POS_Y_SCO
 - def.h, 12
- param, 7
 - def.h, 13
 - game_speed, 8
 - hold, 8
 - level, 8
 - music_n, 8
 - next, 8
 - pause, 8
 - theme, 8
- pause
 - param, 8
 - visual.c, 24
 - visual.h, 26
- print_highscore
 - menu.c, 21
 - menu.h, 22
- print_tet
 - game.c, 15
 - game.h, 18
- randomBag
 - game.c, 16
 - game.h, 19
- reached_top
 - game.c, 16
 - game.h, 19
- read_highscore
 - menu.c, 21
 - menu.h, 22
- reset_highscore
 - menu.c, 21
 - menu.h, 22
- rotate_tet
 - game.c, 16
 - game.h, 19
- rotation
 - tetri, 9
- SFX.c, 23
 - music, 23
 - stop_music, 23
- SFX.h, 23
 - music, 23
 - stop_music, 24
- score
 - Game, 7
- shape
 - tetri, 9
- stop_music
 - SFX.c, 23
 - SFX.h, 24
- TET_SIZE
 - def.h, 12
- tab
 - Game, 7
- tetri, 8
 - def.h, 13
 - hold_possible, 9
 - rotation, 9
 - shape, 9
 - visual, 9
 - x_pos, 9
 - y_pos, 9
- tetrinos
 - game.c, 16
- text
 - Bouton, 5
- theme
 - param, 8
- turn
 - Game, 7
- validate_highscore
 - menu.c, 21
 - menu.h, 22
- visual
 - tetri, 9
- visual.c, 24
 - blit_tab, 24
 - blit_tet, 24
 - pause, 24
 - write_text, 24
- visual.h, 25
 - blit_tab, 25
 - blit_tet, 25
 - pause, 26
 - write_text, 26
- WINDOW_SIZE_X
 - def.h, 12
- WINDOW_SIZE_Y
 - def.h, 12
- write_text
 - visual.c, 24
 - visual.h, 26

x_pos

tetri, [9](#)

y_pos

tetri, [9](#)