Tetris

1.1

Généré par Doxygen 1.8.1.2

Lundi Mai 25 2015 04 :49 :13

Table des matières

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 avec une brève description :

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 detruite 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 detruite 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 definissant les paramtres l du Jeu.

`#include <def.h>`

Champs de données

– float game_speed
  Definis la vitesse de base du jeu.
– int hold
  Autorise le hold.
– int pause
  Autorise la mise en pause.
– int level
  Selection du niveau initial.
– int next
  Autorise la previsualisation de la piece suivante.

– int music_n
  Fond sonor.
– int theme
  Fond ecran.

3.3.1 Description détaillée

Structure definissant les paramtres l du Jeu.

3.3.2 Documentation des champs

3.3.2.1 float game_speed

Definis la vitesse de base du jeu.

3.3.2.2 int hold

Autorise le hold.

3.3.2.3 int level

Selection du niveau initial.

3.3.2.4 int music_n

Fond sonor.

3.3.2.5 int next

Autorise la previsualisation de la piece suivante.

3.3.2.6 int pause

Autorise la mise en pause.

3.3.2.7 int theme

Fond ecran.

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 effectuants les differentes definitions.

```
#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 associé à une partie en cours.
- struct Bouton
  Structure associé au bouton.
- struct param
  Structure definissant les paramtres l 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_DIFFUCLTY 0.9
- #define GAME_SIZE_X 10
- #define GAME_SIZE_Y 21
- #define TET_SIZE 20
- #define HIDEN_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 effectuants les differentes definitions. Fichier effectuants les differentes definitions

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 HIDEN_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 correspondate 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]

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 correspondate à ln_num.

Paramètres

in	*play_tab*	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	*ln_num*	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	*Game*	pointeur vers le jeu en cours

4.2.2.3 void initialiseGame (Game * *game*)

Initialise le jeu.

Paramètres

in	*current_play*	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	*tab*	pointeur vers le tableau

4.2.2.5 int is_collison (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	*play_tab*	Pointeur sur un objet de type tableau d'entier à deux dimension.
out	*result_tab*	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	*play_tab*	pointeur vers le tableau du jeu en cours
in	*tet*	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	*play*	pointeur vers une structure Game du jeu en cours
in	*posx*	position horizontale desire
in	*posy*	position verticale desire
in	*rot*	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	*shape*	forme du tetrino
in	*posx*	position horizontale d'origine du tetrino
in	*posy*	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)
  Verifdie 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 correspondate 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 correspondate à ln_num.

Paramètres

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

4.3.1.2 void hold (Game * *game*)

Effectue les operations lie au hold d'une piece.

Paramètres

in	*Game*	pointeur vers le jeu en cours

4.3.1.3 void initialiseGame (Game * *current_play*)

Initialise le jeu.

Paramètres

in	*current_play*	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

in	*tab*	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

in	*play_tab*	Pointeur sur un objet de type tableau d'entier à deux dimension.
out	*result_tab*	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	*play_tab*	pointeur vers le tableau du jeu en cours
in	*tet*	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	*play*	pointeur vers une structure Game du jeu en cours
in	*posx*	position horizontale desire
in	*posy*	position verticale desire
in	*rot*	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	*shape*	forme du tetrino
in	*posx*	position horizontale d'origine du tetrino
in	*posy*	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*)

Verifdie 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 remplie 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	*ecran*	Pointeur sur un objet de type SDL_Surface : Ecran d'affichage principale
in	*settings*	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	*name*	Pointeur sur un tableau de char de taille 5 representant le nom
in	*score*	score a inscrire

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

Lit les highscores du fichier highscore et remplie 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 remplie 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

in	*ecran*	Pointeur sur un objet de type SDL_Surface : Ecran d'affichage principale
in	*settings*	Reglage du jeu

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

Ecris le highscore dans le fichier a highscore.

Paramètres

in	*name*	Pointeur sur un tableau de char de taille 5 representant le nom
in	*score*	score a inscrire

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

Lit les highscores du fichier highscore et remplie 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	*settings*	pointeur vers une structure param, stock entre autre le morceau a jouer
in	*systeme*	pointeur vers pointeur de FMOD_SYSTEM
in	*musique*	pointeur vers pointeur de SOUND
in	*systeme*	pointeur vers pointeur de FMOD_SYSTEM
in	*musique*	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	*settings*	pointeur vers une structure param, stock entre autre le morceau a jouer
in	*systeme*	pointeur vers pointeur de FMOD_SYSTEM
in	*musique*	pointeur vers pointeur de SOUND
in	*systeme*	pointeur vers pointeur de FMOD_SYSTEM
in	*musique*	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	*tab*	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	*tet*	Pointeur sur tableau de SDL_Surfaces, representant les textures elementaires des tetrino
in	*screen*	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	*text*	Pointeur sur string, reprentant la chaine de caractere
in	*color_R*	valeur de la couleur rouge
in	*color_V*	valeur de la couleur vert
in	*color_B*	valeur de la couleur bleu
in	*size*	taille du texte

Renvoie

un pointeur vers la surface generer

Paramètres

in	*tet*	texture des tetrino
in	*ecran*	Surface de l'ecran
in	*pos_x*	decalage selon l'axe x
in	*pos_y*	decalage selon l'axe y
in	*tetr*	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	*tab*	Pointeur sur un objet de type tableau d'entier à deux dimension.
in	*tet*	Pointeur sur tableau de SDL_Surfaces, representant les textures elementaires des tetrino
in	*screen*	Poiteur sur une SDL_Surface, ecran princial

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	*text*	Pointeur sur string, reprentant la chaine de caractere
in	*color_R*	valeur de la couleur rouge
in	*color_V*	valeur de la couleur vert
in	*color_B*	valeur de la couleur bleu
in	*size*	taille du texte

Renvoie

un pointeur vers la surface generer

Paramètres

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

Index