

## Metatravers

Généré par Doxygen 1.9.8



<b>1 Index des classes</b>	<b>1</b>
1.1 Liste des classes	1
<b>2 Index des fichiers</b>	<b>3</b>
2.1 Liste des fichiers	3
<b>3 Documentation des classes</b>	<b>5</b>
3.1 Référence de la structure barreDeSon	5
3.1.1 Description détaillée	5
3.1.2 Documentation des données membres	6
3.1.2.1 barre	6
3.1.2.2 curseur	6
3.1.2.3 volume	6
3.1.2.4 volume_precedent	6
3.2 Référence de la structure itemMenu	6
3.2.1 Description détaillée	7
3.2.2 Documentation des données membres	7
3.2.2.1 rectangle	7
3.2.2.2 texte	7
3.3 Référence de la structure niveaux	7
3.3.1 Description détaillée	8
3.3.2 Documentation des données membres	8
3.3.2.1 niveau_fini	8
3.3.2.2 numero_collectible	8
3.3.2.3 texture_image_collectible	8
<b>4 Documentation des fichiers</b>	<b>9</b>
4.1 Référence du fichier fichiers_c/fonctions_arrivee_niveaux_2_3.c	9
4.1.1 Description détaillée	10
4.1.2 Documentation des fonctions	10
4.1.2.1 arrivee_niveaux_2_3()	10
4.1.2.2 explications()	12
4.1.2.3 mise_a_jour_rendu_arrivee_niveaux_2_3()	13
4.1.2.4 salon_arrivee_niveaux_2_3()	13
4.2 fonctions_arrivee_niveaux_2_3.c	14
4.3 Référence du fichier fichiers_c/fonctions_carte.c	29
4.3.1 Description détaillée	31
4.3.2 Documentation des fonctions	31
4.3.2.1 carte()	31
4.3.2.2 deplacement_personnage_carte()	32
4.3.2.3 initialisation_objets_carte()	33
4.3.2.4 mise_a_jour_rendu_carte()	33
4.4 fonctions_carte.c	34

4.5 Référence du fichier fichiers_c/fonctions_generales.c	50
4.5.1 Description détaillée	51
4.5.2 Documentation des fonctions	51
4.5.2.1 affichage_texte()	51
4.5.2.2 chargement_image()	52
4.5.2.3 clic_case()	52
4.5.2.4 clic_plein_ecran()	52
4.5.2.5 creer_fenetre_rendu()	53
4.5.2.6 demande_quitter_niveau()	53
4.5.2.7 demande_sauvegarde()	54
4.5.2.8 deplacement_personnage()	55
4.5.2.9 detruire_fenetre_rendu()	55
4.5.2.10 detruire_objets()	55
4.5.2.11 erreur()	57
4.5.2.12 initialisation_objets()	57
4.5.2.13 redimensionnement_fenetre()	58
4.5.2.14 sauvegarder_partie()	59
4.5.2.15 verification_sauvegarde()	59
4.6 fonctions_generales.c	60
4.7 Référence du fichier fichiers_c/fonctions_introduction.c	68
4.7.1 Description détaillée	69
4.7.2 Documentation des fonctions	69
4.7.2.1 introduction()	69
4.7.2.2 mise_a_jour_rendu_introduction()	70
4.8 fonctions_introduction.c	71
4.9 Référence du fichier fichiers_c/fonctions_menu_principal.c	74
4.9.1 Description détaillée	74
4.9.2 Documentation des fonctions	75
4.9.2.1 initialisation_objets_menu_principal()	75
4.9.2.2 menu_principal()	76
4.9.2.3 mise_a_jour_rendu_menu_principal()	77
4.10 fonctions_menu_principal.c	78
4.11 Référence du fichier fichiers_c/fonctions_niveau_1.c	82
4.11.1 Description détaillée	83
4.11.2 Documentation des fonctions	83
4.11.2.1 chargement_niveau_1()	83
4.11.2.2 initialisation_objets_niveau_1()	84
4.11.2.3 mise_a_jour_rendu_niveau_1()	84
4.11.2.4 niveau_1()	85
4.12 fonctions_niveau_1.c	86
4.13 Référence du fichier fichiers_c/fonctions_niveau_2.c	96
4.13.1 Documentation des fonctions	97

4.13.1.1 initialisation_objets_niveau_2()	97
4.13.1.2 mini_jeu_1_niveau_2()	98
4.13.1.3 mini_jeu_2_niveau_2()	99
4.13.1.4 mini_jeux_niveau_2()	99
4.13.1.5 mise_a_jour_bordures_niveau_2()	100
4.13.1.6 mise_a_jour_mini_jeu_1_niveau_2()	101
4.13.1.7 mise_a_jour_mini_jeu_2_niveau_2()	102
4.13.1.8 verification_chemin()	102
4.14 fonctions_niveau_2.c	103
4.15 Référence du fichier fichiers_c/fonctions_niveau_3.c	117
4.15.1 Documentation des fonctions	118
4.15.1.1 initialisation_objets_niveau_3()	118
4.15.1.2 mini_jeu_2_niveau_3()	119
4.15.1.3 mini_jeux_niveau_3()	119
4.15.1.4 mise_a_jour_bordures_niveau_3()	120
4.15.1.5 mise_a_jour_mini_jeu_1_niveau_3()	121
4.15.1.6 mise_a_jour_mini_jeu_2_niveau_3()	121
4.15.1.7 piece_proche_position_correcte()	122
4.15.1.8 rectangle_piece_aleatoire()	122
4.15.1.9 traitement_touches()	123
4.15.1.10 verification_puzzle_fini()	123
4.16 fonctions_niveau_3.c	123
4.17 Référence du fichier fichiers_c/fonctions_niveau_4.c	139
4.17.1 Description détaillée	140
4.17.2 Documentation des fonctions	140
4.17.2.1 etage_1()	140
4.17.2.2 etage_2()	141
4.17.2.3 etage_3()	142
4.17.2.4 etage_4()	142
4.17.2.5 etage_5()	143
4.17.2.6 initialisation_objets_niveau_4()	144
4.17.2.7 mise_a_jour_rendu_niveau_4()	144
4.17.2.8 niveau_4()	145
4.18 fonctions_niveau_4.c	146
4.19 Référence du fichier fichiers_c/fonctions_nouvelle_partie.c	159
4.19.1 Description détaillée	160
4.19.2 Documentation des fonctions	160
4.19.2.1 initialisation_objets_nouvelle_partie()	160
4.19.2.2 mise_a_jour_rendu_nouvelle_partie()	161
4.19.2.3 nouvelle_partie()	161
4.20 fonctions_nouvelle_partie.c	162
4.21 Référence du fichier fichiers_c/fonctions_options.c	168

4.21.1 Description détaillée	169
4.21.2 Documentation des fonctions	169
4.21.2.1 initialisation_objets_options()	169
4.21.2.2 mise_a_jour_barre_de_son()	170
4.21.2.3 mise_a_jour_rendu_options()	170
4.21.2.4 mise_a_jour_touches()	172
4.21.2.5 options()	172
4.22 fonctions_options.c	173
4.23 Référence du fichier fichiers_c/metatravers.c	181
4.23.1 Description détaillée	182
4.23.2 Documentation des fonctions	182
4.23.2.1 main()	182
4.24 metatravers.c	183
4.25 Référence du fichier fichiers_h/fonctions_arrivee_niveaux_2_3.h	202
4.25.1 Documentation des fonctions	204
4.25.1.1 arrivee_niveaux_2_3()	204
4.25.1.2 explications()	206
4.25.1.3 mise_a_jour_rendu_arrivee_niveaux_2_3()	206
4.25.1.4 salon_arrivee_niveaux_2_3()	207
4.26 fonctions_arrivee_niveaux_2_3.h	207
4.27 Référence du fichier fichiers_h/fonctions_carte.h	208
4.27.1 Documentation des fonctions	210
4.27.1.1 carte()	210
4.27.1.2 deplacement_personnage_carte()	211
4.27.1.3 initialisation_objets_carte()	212
4.27.1.4 mise_a_jour_rendu_carte()	212
4.28 fonctions_carte.h	213
4.29 Référence du fichier fichiers_h/fonctions_generales.h	214
4.29.1 Documentation des définitions de type	216
4.29.1.1 direction_t	216
4.29.1.2 modes_t	216
4.29.1.3 option_t	216
4.29.1.4 page_t	217
4.29.1.5 personnage_t	217
4.29.1.6 position_t	217
4.29.2 Documentation du type de l'énumération	217
4.29.2.1 direction_s	217
4.29.2.2 modes_s	217
4.29.2.3 option_s	217
4.29.2.4 page_s	218
4.29.2.5 personnage_s	218
4.29.2.6 position_s	218

4.29.3 Documentation des fonctions	219
4.29.3.1 affichage_texte()	219
4.29.3.2 chargement_image()	219
4.29.3.3 clic_case()	219
4.29.3.4 clic_plein_ecran()	220
4.29.3.5 creer_fenetre_rendu()	220
4.29.3.6 demande_quitter_niveau()	221
4.29.3.7 demande_sauvegarde()	221
4.29.3.8 deplacement_personnage()	222
4.29.3.9 detruire_fenetre_rendu()	222
4.29.3.10 detruire_objets()	223
4.29.3.11 erreur()	224
4.29.3.12 initialisation_objets()	224
4.29.3.13 redimensionnement_fenetre()	225
4.29.3.14 sauvegarder_partie()	225
4.29.3.15 verification_sauvegarde()	225
4.30 fonctions_generales.h	226
4.31 Référence du fichier fichiers_h/fonctions_introduction.h	228
4.31.1 Documentation des fonctions	229
4.31.1.1 introduction()	229
4.31.1.2 mise_a_jour_rendu_introduction()	230
4.32 fonctions_introduction.h	230
4.33 Référence du fichier fichiers_h/fonctions_menu_principal.h	231
4.33.1 Documentation des fonctions	232
4.33.1.1 initialisation_objets_menu_principal()	232
4.33.1.2 menu_principal()	232
4.33.1.3 mise_a_jour_rendu_menu_principal()	234
4.34 fonctions_menu_principal.h	234
4.35 Référence du fichier fichiers_h/fonctions_niveau_1.h	235
4.35.1 Documentation des fonctions	237
4.35.1.1 chargement_niveau_1()	237
4.35.1.2 initialisation_objets_niveau_1()	237
4.35.1.3 mise_a_jour_rendu_niveau_1()	238
4.35.1.4 niveau_1()	238
4.36 fonctions_niveau_1.h	240
4.37 Référence du fichier fichiers_h/fonctions_niveau_2.h	240
4.37.1 Documentation des fonctions	242
4.37.1.1 initialisation_objets_niveau_2()	242
4.37.1.2 mini_jeu_1_niveau_2()	243
4.37.1.3 mini_jeu_2_niveau_2()	243
4.37.1.4 mini_jeux_niveau_2()	244
4.37.1.5 mise_a_jour_bordures_niveau_2()	245

4.37.1.6 mise_a_jour_mini_jeu_1_niveau_2()	246
4.37.1.7 mise_a_jour_mini_jeu_2_niveau_2()	247
4.37.1.8 salon_arrivee_niveaux_2_3()	247
4.37.1.9 verification_chemin()	247
4.38 fonctions_niveau_2.h	248
4.39 Référence du fichier fichiers_h/fonctions_niveau_3.h	249
4.39.1 Documentation des fonctions	251
4.39.1.1 initialisation_objets_niveau_3()	251
4.39.1.2 mini_jeu_2_niveau_3()	252
4.39.1.3 mini_jeux_niveau_3()	252
4.39.1.4 mise_a_jour_bordures_niveau_3()	254
4.39.1.5 mise_a_jour_mini_jeu_1_niveau_3()	254
4.39.1.6 mise_a_jour_mini_jeu_2_niveau_3()	255
4.39.1.7 piece_proche_position_correcte()	255
4.39.1.8 rectangle_piece_aleatoire()	255
4.39.1.9 salon_arrivee_niveaux_2_3()	256
4.39.1.10 traitement_touches()	256
4.39.1.11 verification_puzzle_fini()	257
4.40 fonctions_niveau_3.h	257
4.41 Référence du fichier fichiers_h/fonctions_niveau_4.h	258
4.41.1 Documentation des fonctions	260
4.41.1.1 etage_1()	260
4.41.1.2 etage_2()	261
4.41.1.3 etage_3()	261
4.41.1.4 etage_4()	262
4.41.1.5 etage_5()	262
4.41.1.6 initialisation_objets_niveau_4()	263
4.41.1.7 mise_a_jour_rendu_niveau_4()	263
4.41.1.8 niveau_4()	264
4.42 fonctions_niveau_4.h	265
4.43 Référence du fichier fichiers_h/fonctions_nouvelle_partie.h	266
4.43.1 Documentation des fonctions	268
4.43.1.1 initialisation_objets_nouvelle_partie()	268
4.43.1.2 mise_a_jour_rendu_nouvelle_partie()	268
4.43.1.3 nouvelle_partie()	269
4.44 fonctions_nouvelle_partie.h	270
4.45 Référence du fichier fichiers_h/fonctions_options.h	270
4.45.1 Documentation des fonctions	272
4.45.1.1 initialisation_objets_options()	272
4.45.1.2 mise_a_jour_barre_de_son()	272
4.45.1.3 mise_a_jour_rendu_options()	273
4.45.1.4 mise_a_jour_touches()	274



---

4.45.1.5 options()	274
4.46 fonctions_options.h	275
4.47 Référence du fichier test unitaire/test_pseudo.c	276
4.47.1 Documentation des macros	277
4.47.1.1 BUFFER_SIZE	277
4.47.2 Documentation des fonctions	277
4.47.2.1 main()	277
4.48 test_pseudo.c	277
<b>Index</b>	<b>279</b>



# Chapitre 1

## Index des classes

### 1.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

<a href="#">barreDeSon</a>	.....	5
<a href="#">itemMenu</a>	.....	6
<a href="#">niveaux</a>	.....	7



# Chapitre 2

## Index des fichiers

### 2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

<a href="#">fichiers_c/fonctions_arrivee_niveaux_2_3.c</a>	
Fichier contenant les fonctions s'occupant du hub du niveau 2 et 3	9
<a href="#">fichiers_c/fonctions_carte.c</a>	
Fichier contenant toutes les fonctions gérant la carte principal	29
<a href="#">fichiers_c/fonctions_generales.c</a>	
Fichier avec les fichiers utilisé régulièrement	50
<a href="#">fichiers_c/fonctions_introduction.c</a>	
Fichier avec les fonctions présentant l'introduction du jeu	68
<a href="#">fichiers_c/fonctions_menu_principal.c</a>	
Fichier contenant les fonctions pour le menu principal	
74	
<a href="#">fichiers_c/fonctions_niveau_1.c</a>	
Fichier contenant les fonctions servant à la gestion du niveau 1	
82	
<a href="#">fichiers_c/fonctions_niveau_2.c</a>	96
<a href="#">fichiers_c/fonctions_niveau_3.c</a>	117
<a href="#">fichiers_c/fonctions_niveau_4.c</a>	
Fichier contenant les fonctions servant à la gestion du niveau 2	
139	
<a href="#">fichiers_c/fonctions_nouvelle_partie.c</a>	
Fichier contenant les fonctions servant à la fenêtre d'une nouvelle partie	159
<a href="#">fichiers_c/fonctions_options.c</a>	
Fichier qui réunit les fonctions s'occupant de la fenêtre des options du jeu	168
<a href="#">fichiers_c/metatravers.c</a>	
Fichier qui réunit les différents modules pour le bon fonctionnement du programme	181
<a href="#">fichiers_h/fonctions_arrivee_niveaux_2_3.h</a>	202
<a href="#">fichiers_h/fonctions_carte.h</a>	208
<a href="#">fichiers_h/fonctions_generales.h</a>	214
<a href="#">fichiers_h/fonctions_introduction.h</a>	228
<a href="#">fichiers_h/fonctions_menu_principal.h</a>	231
<a href="#">fichiers_h/fonctions_niveau_1.h</a>	235
<a href="#">fichiers_h/fonctions_niveau_2.h</a>	240
<a href="#">fichiers_h/fonctions_niveau_3.h</a>	249
<a href="#">fichiers_h/fonctions_niveau_4.h</a>	258
<a href="#">fichiers_h/fonctions_nouvelle_partie.h</a>	266
<a href="#">fichiers_h/fonctions_options.h</a>	270
<a href="#">test_unitaire/test_pseudo.c</a>	276



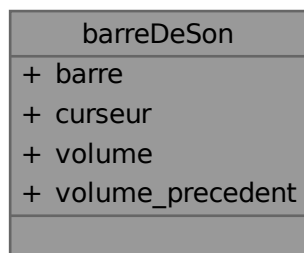
## Chapitre 3

# Documentation des classes

### 3.1 Référence de la structure barreDeSon

```
#include <fonctions_generales.h>
```

Graphe de collaboration de barreDeSon:



#### Attributs publics

- `SDL_Rect` [barre](#)
- `SDL_Rect` [curseur](#)
- `float` [volume](#)
- `float` [volume\\_precedent](#)

#### 3.1.1 Description détaillée

Définition à la ligne [33](#) du fichier [fonctions\\_generales.h](#).

### 3.1.2 Documentation des données membres

#### 3.1.2.1 barre

```
SDL_Rect barreDeSon::barre
```

Définition à la ligne 34 du fichier [fonctions\\_generales.h](#).

#### 3.1.2.2 curseur

```
SDL_Rect barreDeSon::curseur
```

Définition à la ligne 35 du fichier [fonctions\\_generales.h](#).

#### 3.1.2.3 volume

```
float barreDeSon::volume
```

Définition à la ligne 36 du fichier [fonctions\\_generales.h](#).

#### 3.1.2.4 volume\_precedent

```
float barreDeSon::volume_precedent
```

Définition à la ligne 37 du fichier [fonctions\\_generales.h](#).

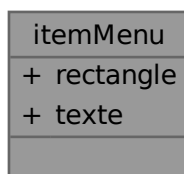
La documentation de cette structure a été générée à partir du fichier suivant :

— [fichiers\\_h/fonctions\\_generales.h](#)

## 3.2 Référence de la structure itemMenu

```
#include <fonctions_generales.h>
```

Graphe de collaboration de itemMenu:





#### Attributs publics

- SDL\_Rect [rectangle](#)
- char [texte](#) [100]

### 3.2.1 Description détaillée

Définition à la ligne [27](#) du fichier [fonctions\\_generales.h](#).

### 3.2.2 Documentation des données membres

#### 3.2.2.1 rectangle

```
SDL_Rect itemMenu::rectangle
```

Définition à la ligne [28](#) du fichier [fonctions\\_generales.h](#).

#### 3.2.2.2 texte

```
char itemMenu::texte[100]
```

Définition à la ligne [29](#) du fichier [fonctions\\_generales.h](#).

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

- [fichiers\\_h/fonctions\\_generales.h](#)

## 3.3 Référence de la structure niveaux

```
#include <fonctions_generales.h>
```

Graphe de collaboration de niveaux:

niveaux
+ niveau_fini
+ texture_image_collectible
+ numero_collectible

### Attributs publics

- int [niveau\\_fini](#)
- SDL\_Texture \* [texture\\_image\\_collectible](#)
- int [numero\\_collectible](#) [3]

### 3.3.1 Description détaillée

Définition à la ligne [41](#) du fichier [fonctions\\_generales.h](#).

### 3.3.2 Documentation des données membres

#### 3.3.2.1 niveau\_fini

```
int niveaux::niveau_fini
```

Définition à la ligne [42](#) du fichier [fonctions\\_generales.h](#).

#### 3.3.2.2 numero\_collectible

```
int niveaux::numero_collectible[3]
```

Définition à la ligne [44](#) du fichier [fonctions\\_generales.h](#).

#### 3.3.2.3 texture\_image\_collectible

```
SDL_Texture* niveaux::texture_image_collectible
```

Définition à la ligne [43](#) du fichier [fonctions\\_generales.h](#).

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

- [fichiers\\_h/fonctions\\_generales.h](#)

## Chapitre 4

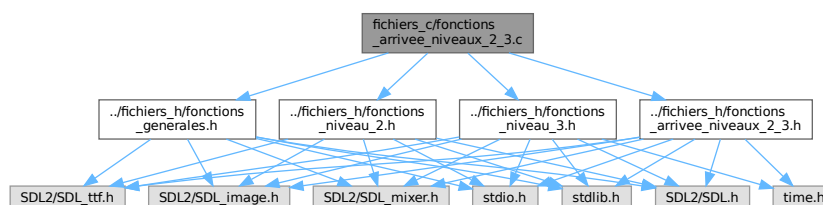
# Documentation des fichiers

### 4.1 Référence du fichier fichiers\_c/fonctions\_arrivee\_niveaux\_2\_3.c

Fichier contenant les fonctions s'occupant du hub du niveau 2 et 3.

```
#include <../fichiers_h/fonctions_generales.h>
#include <../fichiers_h/fonctions_niveau_2.h>
#include <../fichiers_h/fonctions_niveau_3.h>
#include <../fichiers_h/fonctions_arrivee_niveaux_2_3.h>
```

Graphe des dépendances par inclusion de fonctions\_arrivee\_niveaux\_2\_3.c:



### Fonctions

- void [salon\\_arrivee\\_niveaux\\_2\\_3](#) (int \*position\_x, int \*position\_y, int tile\_map[18][32], [page\\_t](#) page\_active)  
*Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.*
- void [mise\\_a\\_jour\\_rendu\\_arrivee\\_niveaux\\_2\\_3](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int position\_x, int position\_y, int tile\_map[18][32], [niveaux](#) \*avancee\_niveaux, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile, [page\\_t](#) page\_active)
- void [explications](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_explications, SDL\_Keycode touche\_interagir, SDL\_Keycode touche\_sauter\_monter, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleur, [itemMenu](#) \*itemsExplications, int largeur, int hauteur, int numero\_mini\_jeu)

— void [arrivee\\_niveaux\\_2\\_3](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_↵  
 bool \*programme\_lance, int \*mini\_jeu, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, SDL\_Texture  
 \*\*texture, SDL\_Surface \*\*surface, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_↵  
 plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_↵  
 \_personnage, int \*mini\_jeu\_termine, int \*mini\_jeu\_1\_termine, int \*mini\_jeu\_2\_termine, SDL\_Texture  
 \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Texture \*\*texture\_image\_monstre\_↵  
 terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_↵  
 Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_interagir, SDL\_Texture \*\*texture\_image\_porte,  
[niveaux](#) \*avancee\_niveaux, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre,  
 SDL\_Texture \*\*texture\_image\_dossier, SDL\_Texture \*\*barre\_windows\_1, SDL\_Texture \*\*barre\_windows\_↵  
 \_2, SDL\_Texture \*\*barre\_windows\_3, SDL\_Texture \*\*barre\_windows\_4, int tile\_map[18][32], SDL\_Rect  
 \*rectangle\_tile, int \*mouvement\_monstre, [modes\\_t](#) \*modeActif, int \*mode\_difficile, [itemMenu](#) \*items\_↵  
 DemandeQuitter, int tailleDemande, SDL\_Color couleurNoire, int tile\_map\_mini\_jeu\_niveau\_2[19][27],  
 SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Rect \*rectangle\_demande, time\_t \*timestamp,  
 SDL\_Texture \*\*texture\_image\_perso\_gagnant, int \*avancer, int \*reculer, int \*sauter, int \*position\_↵  
 \_avant\_saut, int \*saut, int \*tombe, int \*position\_x\_initiale, int \*position\_y\_initiale, int \*position\_x, int  
 \*position\_y, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, [page\\_t](#) \*page\_active, SDL\_↵  
 Texture \*\*texture\_image\_mur\_mini\_jeu, int collectibles\_intermediaires[3], [itemMenu](#) \*itemsExplications,  
 SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture  
 \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_↵  
 \_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_↵  
 \_image\_pipe\_courant, SDL\_Texture \*\*texture\_image\_mur\_termine, int \*valide, SDL\_Rect rectangle\_↵  
 piece[45], int piece\_bloquee[45], SDL\_Rect rectangle\_emplacement\_piece[45], int \*piece\_selectionnee,  
 int \*decalage\_x, int \*decalage\_y, SDL\_Texture \*\*texture\_image\_puzzle, Mix\_Music \*\*musique, SDL\_↵  
 \_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int tile\_map\_mini\_jeu\_niveau\_3[24][32], int  
 \*descendre, int \*interagir, int \*bloc\_x, int \*bloc\_y, SDL\_Texture \*\*texture\_image\_sol\_labyrinthe, SDL\_↵  
 \_Texture \*\*texture\_image\_bordure\_labyrinthe, SDL\_Texture \*\*texture\_image\_fin\_labyrinthe, SDL\_Color  
 couleurTitre, [itemMenu](#) \*itemsDemandeSauvegarde, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo,  
[personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int  
 \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

### 4.1.1 Description détaillée

Fichier contenant les fonctions s'occupant du hub du niveau 2 et 3.

Définition dans le fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

### 4.1.2 Documentation des fonctions

#### 4.1.2.1 arrivee\_niveaux\_2\_3()

```
void arrivee_niveaux_2_3 (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    int * mini_jeu,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    SDL_Texture ** texture,
    SDL_Surface ** surface,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_personnage,
```

```
SDL_Rect * rectangle_personnage,
int * mini_jeu_termine,
int * mini_jeu_1_termine,
int * mini_jeu_2_termine,
SDL_Texture ** texture_image_fond,
SDL_Texture ** texture_image_sol,
SDL_Texture ** texture_image_monstre_terrestre,
SDL_Texture ** texture_image_monstre_volant,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_interagir,
SDL_Texture ** texture_image_porte,
niveau * avancee_niveaux,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
SDL_Texture ** texture_image_dossier,
SDL_Texture ** barre_windows_1,
SDL_Texture ** barre_windows_2,
SDL_Texture ** barre_windows_3,
SDL_Texture ** barre_windows_4,
int tile_map[18][32],
SDL_Rect * rectangle_tile,
int * mouvement_monstre,
modes_t * modeActif,
int * mode_difficile,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
SDL_Color couleurNoire,
int tile_map_mini_jeu_niveau_2[19][27],
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Rect * rectangle_demande,
time_t * timestamp,
SDL_Texture ** texture_image_perso_gagnant,
int * avancer,
int * reculer,
int * sauter,
int * position_avant_saut,
int * saut,
int * tombe,
int * position_x_initiale,
int * position_y_initiale,
int * position_x,
int * position_y,
int * largeur,
int * hauteur,
int * largeur_tile,
int * hauteur_tile,
page_t * page_active,
SDL_Texture ** texture_image_mur_mini_jeu,
int collectibles_intermediaires[3],
itemMenu * itemsExplications,
SDL_Texture ** texture_image_pipe_vertical,
SDL_Texture ** texture_image_pipe_horizontal,
SDL_Texture ** texture_image_pipe_haut_droit,
SDL_Texture ** texture_image_pipe_bas_droit,
SDL_Texture ** texture_image_pipe_bas_gauche,
SDL_Texture ** texture_image_pipe_haut_gauche,
```

```

SDL_Texture ** texture_image_pipe_courant,
SDL_Texture ** texture_image_mur_termine,
int * valide,
SDL_Rect rectangle_piece[45],
int piece_bloquee[45],
SDL_Rect rectangle_emplacement_piece[45],
int * piece_selectionnee,
int * decalage_x,
int * decalage_y,
SDL_Texture ** texture_image_puzzle,
Mix_Music ** musique,
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
int tile_map_mini_jeu_niveau_3[24][32],
int * descendre,
int * interagir,
int * bloc_x,
int * bloc_y,
SDL_Texture ** texture_image_sol_labyrinthe,
SDL_Texture ** texture_image_bordure_labyrinthe,
SDL_Texture ** texture_image_fin_labyrinthe,
SDL_Color couleurTitre,
itemMenu * itemsDemandeSauvegarde,
barreDeSon * barre_de_son,
itemMenu * pseudo,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 482 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

#### 4.1.2.2 explications()

```

void explications (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_explications,
    SDL_Keycode touche_interagir,
    SDL_Keycode touche_sauter_monter,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleur,
    itemMenu * itemsExplications,
    int largeur,
    int hauteur,
    int numero_mini_jeu )

```

Définition à la ligne 269 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

## 4.1.2.3 mise\_a\_jour\_rendu\_arrivee\_niveaux\_2\_3()

```

void mise_a_jour_rendu_arrivee_niveaux_2_3 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Texture ** texture_image_dossier,
    SDL_Texture ** barre_windows_1,
    SDL_Texture ** barre_windows_2,
    SDL_Texture ** barre_windows_3,
    SDL_Texture ** barre_windows_4,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int position_x,
    int position_y,
    int tile_map[18][32],
    niveaux * avancee_niveaux,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    int largeur,
    int hauteur,
    int largeur_tile,
    int hauteur_tile,
    page_t page_active )

```

Définition à la ligne 125 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

## 4.1.2.4 salon\_arrivee\_niveaux\_2\_3()

```

void salon_arrivee_niveaux_2_3 (
    int * position_x,
    int * position_y,
    int tile_map[18][32],
    page_t page_active )

```

Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.

## Paramètres

<i>position_x</i>	pointeur sur la position du personnage sur l'horizontal du tilemap
<i>position_y</i>	pointeur sur la position du perosnnage sur la verticale du tilemap
<i>tile_map</i>	Matrice représentant la map ou se trouve le personnage
<i>page_active</i>	Enumération représentant sur quel page on se trouve

Définition à la ligne 21 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).





```
00063     else if(page_active == NIVEAU_3) {
00064
00065         /* Définition du salon pour le niveau 3 */
00066         int initialisation_tile_map_2[18][32] = {
00067             {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2},
00068             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00069             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00070             {2, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0},
00071             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00072             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00073             {2, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
00074             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00075             {2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1},
00076             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00077             {2, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
00078             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00079             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00080             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00081             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0},
00082             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00083             {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00084             {13, 1, 14, 1, 15, 1, 16, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
00085         };
00086
00087         /* Copie de l'arrivée dans le niveau 2 ou 3 */
00088         for (y = 0; y < 18; y++)
00089             for (x = 0; x < 32; x++)
00090                 tile_map[y][x] = initialisation_tile_map_2[y][x];
00091     }
00092 }
00093
00094
00095
00096 /**
00097  * \fn void mise_a_jour_rendu_arrivee_niveaux_2_3(SDL_Renderer **renderer, SDL_Texture
00098  **texture_image_fond, SDL_Texture **texture_image_sol, SDL_Rect *rectangle_plein_ecran, SDL_Texture
00099  **texture_image_plein_ecran, SDL_Texture **texture_image_fin_premiers_niveaux, SDL_Texture **texture,
00100  SDL_Rect *rectangle_tile, SDL_Texture **texture_image_dossier, SDL_Texture **barre_windows_1,
00101  SDL_Texture **barre_windows_2, SDL_Texture **barre_windows_3, SDL_Texture **barre_windows_4,
00102  SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int position_x, int
00103  position_y, int tile_map[18][32], niveaux avancee_niveaux, int largeur, int hauteur, int
00104  largeur_tile, int hauteur_tile, page_t page_active)
00105  * \brief Fonction qui permet de mettre à jour le rendu du salon en arrivant dans le niveau 2 ou 3
00106  * \param renderer Pointeur vers le renderer SDL.
00107  * \param texture_image_fond Texture de l'image de fond.
00108  * \param texture_image_sol Texture de l'image du sol.
00109  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00110  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00111  * \param texture_image_fin_premiers_niveaux Texture de l'image de fin des premiers niveaux.
00112  * \param texture Texture SDL.
00113  * \param rectangle_tile Rectangle du personnage SDL.
00114  * \param position_x Position en x.
00115  * \param position_y Position en y.
00116  * \param tile_map Carte de tuiles.
00117  * \param avancee_niveaux Structure de progression des niveaux.
00118  * \param largeur Largeur.
00119  * \param hauteur Hauteur.
00120  * \param largeur_tile Largeur de la tuile.
00121  * \param hauteur_tile Hauteur de la tuile.
00122  * \param page_active Page active.
00123  * \see erreur
00124  */
```

```

00125 void mise_a_jour_rendu_arrivee_niveaux_2_3(SDL_Renderer **renderer, SDL_Texture **texture_image_fond,
00126 SDL_Texture **texture_image_sol,
00127 SDL_Rect *rectangle_plein_ecran, SDL_Texture
00128 **texture_image_plein_ecran, SDL_Texture **texture_image_fin_premiers_niveaux,
00129 SDL_Texture **texture, SDL_Rect *rectangle_tile,
00130 SDL_Texture **texture_image_dossier,
00131 SDL_Texture **barre_windows_1, SDL_Texture
00132 **barre_windows_2, SDL_Texture **barre_windows_3,
00133 SDL_Texture **barre_windows_4, SDL_Texture
00134 **texture_image_personnage, SDL_Rect *rectangle_personnage,
00135 int position_x, int position_y, int tile_map[18][32],
00136 niveaux *avancee_niveaux, SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix,
00137 int largeur, int hauteur, int largeur_tile, int
00138 hauteur_tile, page_t page_active) {
00139     int x, y;
00140     /* Efface le rendu */
00141     if(SDL_RenderClear((*renderer)) != 0)
00142         erreur("Effacement rendu échoué");
00143     /* Copie la texture de l'image de fond du salon */
00144     if(SDL_RenderCopy((*renderer), (*texture_image_fond), NULL, NULL) != 0)
00145         erreur("Copie de la texture");
00146     /* Affiche tout le salon en fonction des valeurs */
00147     for (y = 0; y < hauteur / hauteur_tile; y++) {
00148         for (x = 0; x < largeur / largeur_tile; x++) {
00149             /* Définition de la position de la tuile */
00150             rectangle_tile->x = x * largeur_tile;
00151             rectangle_tile->y = y * hauteur_tile;
00152             rectangle_tile->w = largeur_tile;
00153             rectangle_tile->h = hauteur_tile;
00154             if((tile_map[y][x] == 1))
00155                 (*texture) = (*texture_image_sol);
00156             else
00157                 (*texture) = NULL;
00158             if((*texture))
00159                 SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile);
00160             if(page_active == NIVEAU_2) {
00161                 if((tile_map[y][x] == 5) && (y == 6) && (avancee_niveaux[1].numero_collectible[0] ==
00162 0))
00163                     if(SDL_RenderCopy((*renderer), avancee_niveaux[1].texture_image_collectible, NULL,
00164 rectangle_tile) != 0)
00165                         erreur("Copie de la texture");
00166                 if((tile_map[y][x] == 5) && (y == 8) && (avancee_niveaux[1].numero_collectible[2] ==
00167 0))
00168                     if(SDL_RenderCopy((*renderer), avancee_niveaux[1].texture_image_collectible, NULL,
00169 rectangle_tile) != 0)
00170                         erreur("Copie de la texture");
00171             }
00172             else if(page_active == NIVEAU_3) {
00173                 if((tile_map[y][x] == 5) && (y == 5) && (avancee_niveaux[2].numero_collectible[0] ==
00174 0))
00175                     if(SDL_RenderCopy((*renderer), avancee_niveaux[2].texture_image_collectible, NULL,
00176 rectangle_tile) != 0)
00177                         erreur("Copie de la texture");
00178                 if((tile_map[y][x] == 5) && (y == 13) && (avancee_niveaux[2].numero_collectible[2] ==
00179 0))
00180                     if(SDL_RenderCopy((*renderer), avancee_niveaux[2].texture_image_collectible, NULL,
00181 rectangle_tile) != 0)
00182                         erreur("Copie de la texture");
00183             }
00184             if(tile_map[y][x] == 6) {
00185                 rectangle_tile->x = x * largeur_tile;
00186                 rectangle_tile->y = y * hauteur_tile;
00187                 rectangle_tile->w = largeur_tile * 3;
00188                 rectangle_tile->h = hauteur_tile * 3;
00189                 if(SDL_RenderCopy((*renderer), (*texture_image_dossier), NULL, rectangle_tile) != 0)
00190                     erreur("Copie de la texture");
00191             }
00192             if((tile_map[y][x] == 7) && ((position_x != 16) || (position_y != 16)))

```

```

00197         if(SDL_RenderCopy((*renderer), (*texture_image_fin_premiers_niveaux), NULL,
rectangle_tile) != 0)
00198             erreur("Copie de la texture");
00199     }
00200 }
00201
00202 /* Cas pour le début de la barre windows dans le niveau 3 */
00203 if(page_active == NIVEAU_3) {
00204
00205     /* Affiche tout le salon en fonction des valeurs */
00206     for (y = 0; y < hauteur / hauteur_tile; y++) {
00207
00208         for (x = 0; x < largeur / largeur_tile; x++) {
00209
00210             rectangle_tile->x = x * largeur_tile;
00211             rectangle_tile->y = y * hauteur_tile;
00212             rectangle_tile->w = largeur_tile * 2;
00213             rectangle_tile->h = hauteur_tile;
00214
00215             if(tile_map[y][x] == 13)
00216                 if(SDL_RenderCopy((*renderer), (*barre_windows_1), NULL, rectangle_tile) != 0)
00217                     erreur("Copie de la texture");
00218
00219             if(tile_map[y][x] == 14)
00220                 if(SDL_RenderCopy((*renderer), (*barre_windows_2), NULL, rectangle_tile) != 0)
00221                     erreur("Copie de la texture");
00222
00223             if(tile_map[y][x] == 15)
00224                 if(SDL_RenderCopy((*renderer), (*barre_windows_3), NULL, rectangle_tile) != 0)
00225                     erreur("Copie de la texture");
00226
00227             if(tile_map[y][x] == 16)
00228                 if(SDL_RenderCopy((*renderer), (*barre_windows_4), NULL, rectangle_tile) != 0)
00229                     erreur("Copie de la texture");
00230         }
00231     }
00232 }
00233
00234 /* Copie la texture de l'image du personnage */
00235
00236 rectangle_personnage->x = position_x * largeur_tile;
00237 rectangle_personnage->y = position_y * hauteur_tile;
00238 rectangle_personnage->w = largeur_tile;
00239 rectangle_personnage->h = hauteur_tile;
00240
00241 if(SDL_RenderCopy((*renderer), (*texture_image_personnage), NULL, rectangle_personnage) != 0)
00242     erreur("Copie de la texture");
00243
00244 /* Copie la texture de l'image de plein écran */
00245
00246 rectangle_plein_ecran->x = largeur_tile * 31;
00247 rectangle_plein_ecran->y = 0;
00248 rectangle_plein_ecran->w = largeur_tile;
00249 rectangle_plein_ecran->h = hauteur_tile;
00250
00251 if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00252     erreur("Copie de la texture");
00253
00254 /* Copie la texture de l'image de la croix */
00255
00256 rectangle_croix->x = 0;
00257 rectangle_croix->y = 0;
00258 rectangle_croix->w = largeur_tile;
00259 rectangle_croix->h = hauteur_tile;
00260
00261 if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00262     erreur("Copie de la texture");
00263
00264 /* Affiche le rendu */
00265 SDL_RenderPresent((*renderer));
00266 }
00267
00268 /* Fonction qui permet d'afficher des explications pour chaque mini-jeux */
00269 void explications(SDL_Renderer **renderer, SDL_Rect *rectangle_explications, SDL_Keycode
touche_interagir, SDL_Keycode touche_sauter_monter,
00270                 SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, SDL_Color
couleur,
00271                 itemMenu *itemsExplications, int largeur, int hauteur, int numero_mini_jeu) {
00272
00273     /* Utilisation de la fusion pour un rendu avec transparence */
00274     SDL_SetRenderDrawBlendMode((*renderer), SDL_BLENDMODE_BLEND);
00275
00276     /* Affichage du rectangle des explications */
00277
00278     rectangle_explications->x = 0;
00279     rectangle_explications->y = 0;
00280     rectangle_explications->w = largeur;

```

```

00281     rectangle_explications->h = hauteur;
00282
00283     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 200);
00284     SDL_RenderFillRect((*renderer), rectangle_explications);
00285
00286     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 0);
00287
00288     /* Initialisation du texte pour la première ligne des explications */
00289
00290     if(numero_mini_jeu == 1)
00291         sprintf(itemsExplications[0].texte, "Il semblerait que la structure de refroidissement de la
machine ait un soucis.");
00292
00293     else if(numero_mini_jeu == 2)
00294         sprintf(itemsExplications[0].texte, "                                Des virus ont envahi l'ordinateur
!!!                                ");
00295
00296     else if(numero_mini_jeu == 3)
00297         sprintf(itemsExplications[0].texte, "                                Un composant n'est plus fonctionnel car un appareil
vient de chuter.                                ");
00298
00299     else if(numero_mini_jeu == 4)
00300         sprintf(itemsExplications[0].texte, "                                Une information confidentielle n'a pas pu atteindre la
fin du transfert.                                ");
00301
00302     /* Affichage du rectangle de la première phrase des explications */
00303
00304     itemsExplications[0].rectangle.x = largeur / 15;
00305     itemsExplications[0].rectangle.y = hauteur / 4 + hauteur / 20;
00306     itemsExplications[0].rectangle.w = largeur - largeur / 15 * 2;
00307     itemsExplications[0].rectangle.h = hauteur / 10;
00308
00309     affichage_texte(renderer, surface, texture_texte, &(itemsExplications[0]),
00310                     police, couleur);
00311
00312     /* Initialisation du texte pour la seconde ligne des explications */
00313
00314     if(numero_mini_jeu == 1)
00315         sprintf(itemsExplications[0].texte, "                                Interagissez (touche %s) avec les tuyaux
et                                ", SDL_GetKeyName(touche_interagir));
00316
00317     else if(numero_mini_jeu == 2)
00318         sprintf(itemsExplications[0].texte, "                                Eliminez les tous en sautant dessus
(touche %s).                                ", SDL_GetKeyName(touche_sauter_monter));
00319
00320     else if(numero_mini_jeu == 3)
00321         sprintf(itemsExplications[0].texte, "                                Reconstituez-le en bougeant les divers morceaux
avec la souris                                ");
00322
00323     else if(numero_mini_jeu == 4)
00324         sprintf(itemsExplications[0].texte, "                                Ramenez-la au plus vite vers la fin de ce
labyrinthe                                ");
00325
00326     /* Affichage du rectangle de la troisième phrase des explications */
00327
00328     itemsExplications[0].rectangle.y = hauteur / 4 + hauteur / 20 + hauteur / 10;
00329
00330     affichage_texte(renderer, surface, texture_texte, &(itemsExplications[0]),
00331                     police, couleur);
00332
00333     itemsExplications[0].rectangle.y = hauteur / 4 + hauteur / 20 + hauteur / 10 * 2;
00334
00335     /* Initialisation du texte pour la seconde ligne des explications */
00336
00337     if(numero_mini_jeu == 1)
00338         sprintf(itemsExplications[0].texte, "                                activez la valve afin de remettre la structure
en marche.                                ");
00339
00340     else if(numero_mini_jeu == 3)
00341         sprintf(itemsExplications[0].texte, "                                pour le faire fonctionner de
nouveau.                                ");
00342
00343     else if(numero_mini_jeu == 4)
00344         sprintf(itemsExplications[0].texte, "                                en poussant et en tirant le bloc (rester appuyer sur
la touche %s)                                ", SDL_GetKeyName(touche_interagir));
00345
00346     /* Affichage du rectangle de la troisième phrase des explications */
00347     if(numero_mini_jeu != 2)
00348         affichage_texte(renderer, surface, texture_texte, &(itemsExplications[0]),
00349                         police, couleur);
00350
00351     itemsExplications[1].rectangle.x = largeur / 3 * 2;
00352     itemsExplications[1].rectangle.y = hauteur - hauteur / 4 - hauteur / 20;
00353     itemsExplications[1].rectangle.w = largeur / 7;
00354     itemsExplications[1].rectangle.h = hauteur / 10;
00355
00356     /* Initialisation du texte pour sortir de l'explication */

```

```

00357
00358     sprintf(itemsExplications[1].texte, " C'est parti ! ");
00359
00360     SDL_SetRenderDrawColor((*render), 200, 200, 200, 255);
00361     SDL_RenderDrawRect((*render), &(itemsExplications[1].rectangle));
00362
00363     /* Affichage du texte et du rectangle pour sortir de l'explication */
00364
00365     SDL_SetRenderDrawColor((*render), 0, 0, 0, 0);
00366
00367     affichage_texte(render, surface, texture_texte, &(itemsExplications[1]),
00368                     police, couleur);
00369
00370     /* Affiche le rendu */
00371     SDL_RenderPresent((*render));
00372 }
00373
00374
00375 /**
00376  * \fn void arrivee_niveaux_2_3(SDL_Event *event, SDL_Window **window, SDL_Renderer **render, int
  *mini_jeu, SDL_Texture **texture_image_fin_premiers_niveaux, SDL_Texture **texture, SDL_Surface
  **surface, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
  *plein_ecran, SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int
  *mini_jeu_termine, int *mini_jeu_1_termine, int *mini_jeu_2_termine, SDL_Texture **texture_image_fond,
  SDL_Texture **texture_image_sol, SDL_Texture **texture_image_monstre_terrestre, SDL_Texture
  **texture_image_monstre_volant, SDL_Keycode *touche_aller_a_droite, SDL_Keycode
  *touche_aller_a_gauche, SDL_Keycode *touche_interagir, SDL_Texture **texture_image_porte, niveaux
  *avancee_niveaux, SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Texture
  **texture_image_dossier, SDL_Texture **barre_windows_1, SDL_Texture **barre_windows_2, SDL_Texture
  **barre_windows_3, SDL_Texture **barre_windows_4, int tile_map[18][32], SDL_Rect *rectangle_tile, int
  *mouvement_monstre, modes_t *modeActif, int *mode_difficile, itemMenu *itemsDemandeQuitter, int
  tailleDemandeQuitter, SDL_Color couleurNoire, int tile_map_mini_jeu_niveau_2[19][27], SDL_Texture
  **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande_quitter, time_t *timestamp,
  SDL_Texture **texture_image_perso_gagnant, int *avancer, int *reculer, int *sauter, int
  *position_avant_saut, int *saut, int *tombe, int *position_x_initiale, int *position_y_initiale, int
  *position_x, int *position_y, int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile,
  page_t *page_active, SDL_Texture **texture_image_mur_mini_jeu, int collectibles_intermediaires[3],
  SDL_Texture **texture_image_pipe_vertical, SDL_Texture **texture_image_pipe_horizontal, SDL_Texture
  **texture_image_pipe_haut_droit, SDL_Texture **texture_image_pipe_bas_droit, SDL_Texture
  **texture_image_pipe_bas_gauche, SDL_Texture **texture_image_pipe_haut_gauche, SDL_Texture
  **texture_image_pipe_courant, SDL_Texture **texture_image_mur_termine, int *valide, SDL_Rect
  rectangle_piece[45], int piece_bloquee[45], SDL_Rect rectangle_emplacement_piece[45], int
  *piece_selectionnee, int *decalage_x, int *decalage_y, SDL_Texture **texture_image_puzzle, Mix_Music
  **musique, int tile_map_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int *bloc_x, int
  *bloc_y, SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture **texture_image_bordure_labyrinthe,
  SDL_Texture **texture_image_fin_labyrinthe)
00377  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le salon en
  arrivant dans le niveau 2 ou 3
00378  *
00379  * Cette fonction prend en charge la gestion de l'arrivée aux niveaux 2 et 3 du jeu.
00380  *
00381  * \param event Pointeur vers l'événement SDL.
00382  * \param window Pointeur vers la fenêtre SDL.
00383  * \param renderer Pointeur vers le renderer SDL.
00384  * \param mini_jeu Pointeur vers le type de mini-jeu en cours.
00385  * \param texture_image_fin_premiers_niveaux Texture de l'image de fin des premiers niveaux.
00386  * \param texture Texture SDL.
00387  * \param surface Surface SDL.
00388  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00389  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00390  * \param plein_ecran Booléen pour le plein écran.
00391  * \param texture_image_personnage Texture de l'image du personnage.
00392  * \param rectangle_personnage Rectangle du personnage SDL.
00393  * \param mini_jeu_termine Booléen pour le mini-jeu terminé.
00394  * \param mini_jeu_1_termine Booléen pour le mini-jeu 1 terminé.
00395  * \param mini_jeu_2_termine Booléen pour le mini-jeu 2 terminé.
00396  * \param texture_image_fond Texture de l'image de fond.
00397  * \param texture_image_sol Texture de l'image du sol.
00398  * \param texture_image_monstre_terrestre Texture de l'image du monstre terrestre.
00399  * \param texture_image_monstre_volant Texture de l'image du monstre volant.
00400  * \param touche_aller_a_droite Touche pour aller à droite.
00401  * \param touche_aller_a_gauche Touche pour aller à gauche.
00402  * \param touche_interagir Touche pour interagir.
00403  * \param texture_image_porte Texture de l'image de porte.
00404  * \param avancee_niveaux Structure de progression des niveaux.
00405  * \param touche_sauter_monter Touche pour sauter/monter.
00406  * \param touche_descendre Touche pour descendre.
00407  * \param texture_image_dossier Texture de l'image du dossier.
00408  * \param barre_windows_1 Texture de la barre Windows 1.
00409  * \param barre_windows_2 Texture de la barre Windows 2.
00410  * \param barre_windows_3 Texture de la barre Windows 3.
00411  * \param barre_windows_4 Texture de la barre Windows 4.
00412  * \param tile_map Carte de tuiles.
00413  * \param rectangle_tile Rectangle de la tuile SDL.
00414  * \param mouvement_monstre Mouvement du monstre.
00415  * \param modeActif Mode actif du jeu.
00416  * \param mode_difficile Niveau de difficulté.

```

```

00417 * \param itemsDemandeQuitter Tableau d'items pour la demande de quitter.
00418 * \param tailleDemandeQuitter Taille du tableau d'items pour la demande de quitter.
00419 * \param couleurNoire Couleur noire SDL.
00420 * \param tile_map_mini_jeu_niveau_2 Carte de tuiles pour le mini-jeu niveau 2.
00421 * \param texture_texte Texture du texte SDL.
00422 * \param police Police de caractères TTF.
00423 * \param rectangle_demande_quitter Rectangle de la demande de quitter.
00424 * \param timestamp Timestamp.
00425 * \param texture_image_perso_gagnant Texture de l'image du personnage gagnant.
00426 * \param avancer Avancer.
00427 * \param reculer Reculer.
00428 * \param sauter Sauter.
00429 * \param position_avant_saut Position avant le saut.
00430 * \param saut Saut.
00431 * \param tombe Tomber.
00432 * \param position_x_initiale Position initiale en x.
00433 * \param position_y_initiale Position initiale en y.
00434 * \param position_x Position en x.
00435 * \param position_y Position en y.
00436 * \param largeur Largeur.
00437 * \param hauteur Hauteur.
00438 * \param largeur_tile Largeur de la tuile.
00439 * \param hauteur_tile Hauteur de la tuile.
00440 * \param page_active Page active.
00441 * \param texture_image_mur_mini_jeu Texture de l'image du mur pour le mini-jeu.
00442 * \param collectibles_intermediaires Collectibles intermédiaires.
00443 * \param texture_image_pipe_vertical Texture de l'image du tuyau vertical.
00444 * \param texture_image_pipe_horizontal Texture de l'image du tuyau horizontal.
00445 * \param texture_image_pipe_haut_droit Texture de l'image du tuyau haut droit.
00446 * \param texture_image_pipe_bas_droit Texture de l'image du tuyau bas droit.
00447 * \param texture_image_pipe_bas_gauche Texture de l'image du tuyau bas gauche.
00448 * \param texture_image_pipe_haut_gauche Texture de l'image du tuyau haut gauche.
00449 * \param texture_image_pipe_courant Texture de l'image du tuyau courant.
00450 * \param texture_image_mur_termine Texture de l'image du mur terminé.
00451 * \param valide Valide.
00452 * \param rectangle_piece Rectangle des pièces.
00453 * \param piece_bloquee Pièce bloquée.
00454 * \param rectangle_emplacement_piece Rectangle de l'emplacement de la pièce.
00455 * \param piece_selectionnee Pièce sélectionnée.
00456 * \param decalage_x Décalage en x.
00457 * \param decalage_y Décalage en y.
00458 * \param texture_image_puzzle Texture de l'image du puzzle.
00459 * \param musique Musique du jeu.
00460 * \param tile_map_mini_jeu_niveau_3 Carte de tuiles pour le mini-jeu niveau 3.
00461 * \param descendre Descendre.
00462 * \param interagir Interagir.
00463 * \param bloc_x Bloc en x.
00464 * \param bloc_y Bloc en y.
00465 * \param texture_image_sol_labyrinthe Texture de l'image du sol pour le labyrinthe.
00466 * \param texture_image_bordure_labyrinthe Texture de l'image de la bordure pour le labyrinthe.
00467 * \param texture_image_fin_labyrinthe Texture de l'image de fin pour le labyrinthe.
00468 * \see erreur
00469 * \see redimensionnement_fenetre
00470 * \see salon_arrivee_niveaux_2_3
00471 * \see demande_quitter_niveau
00472 * \see mini_jeux_niveau_2
00473 * \see mini_jeu_1_niveau_2
00474 * \see mini_jeu_2_niveau_2
00475 * \see clic_plein_ecran
00476 * \see clic_case
00477 * \see deplacement_personnage
00478 * \see rectangle_piece_aleatoire
00479 * \see mini_jeux_niveau_3
00480 * \see mise_a_jour_rendu_arrivee_niveaux_2_3
00481 */
00482 void arrivee_niveaux_2_3(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance, int *mini_jeu, SDL_Texture **texture_image_fin_premiers_niveaux,
00483 SDL_Texture **texture, SDL_Surface **surface, SDL_Rect
*rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool *plein_ecran,
00484 SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int
*mini_jeu_termine, int *mini_jeu_1_termine, int *mini_jeu_2_termine,
00485 SDL_Texture **texture_image_fond, SDL_Texture **texture_image_sol,
SDL_Texture **texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant,
00486 SDL_Keycode *touche_interagir, SDL_Texture **texture_image_porte, niveaux *avancee_niveaux,
SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Texture
00487 **texture_image_dossier,
SDL_Texture **barre_windows_1, SDL_Texture **barre_windows_2, SDL_Texture
00488 **barre_windows_3,
SDL_Texture **barre_windows_4, int tile_map[18][32], SDL_Rect
00489 *rectangle_tile, int *mouvement_monstre, modes_t *modeActif, int *mode_difficile,
itemMenu *itemsDemandeQuitter, int tailleDemande, SDL_Color couleurNoire, int
00490 tile_map_mini_jeu_niveau_2[19][27],
SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande,
00491 time_t *timestamp, SDL_Texture **texture_image_perso_gagnant,
int *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut,
00492 int *tombe,

```

```

00493             int *position_x_initiale, int *position_y_initiale, int *position_x, int
00494 *position_y,
00495             int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page_t
00496 *page_active,
00497             SDL_Texture **texture_image_mur_mini_jeu, int collectibles_intermediaires[3],
00498             SDL_Texture **texture_image_pipe_vertical, SDL_Texture
00499 **texture_image_pipe_horizontal,
00500             SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
00501 **texture_image_pipe_bas_droit,
00502             SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
00503 **texture_image_pipe_haut_gauche,
00504             SDL_Texture **texture_image_pipe_courant, SDL_Texture
00505 **texture_image_mur_termine, int *valide,
00506             SDL_Rect rectangle_piece[45], int piece_bloquee[45], SDL_Rect
00507 rectangle_emplacement_piece[45],
00508             int *piece_selectionnee, int *decalage_x, int *decalage_y, SDL_Texture
00509 **texture_image_puzzle, Mix_Music **musique, SDL_Texture **texture_image_croix, SDL_Rect
00510 *rectangle_croix,
00511             int tile_map_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int
00512 *bloc_x, int *bloc_y,
00513             SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture
00514 **texture_image_bordure_labyrinthe,
00515             SDL_Texture **texture_image_fin_labyrinthe, SDL_Color couleurTitre,
00516             itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
00517             personnage_t *personnageActif, position_t *positionActive, int tailleNiveaux,
00518             time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
00519 avancee_succes_intermediaires[10]) {
00520     SDL_Event event_temporaire;
00521     SDL_bool clic_effectue = SDL_FALSE;
00522     Mix_Chunk *effet_sonore = NULL;
00523     int x, y, i;
00524
00525     /* Vérification si le joueur n'est pas dans un mini-jeu ou si le niveau est actif */
00526     if (((!(*mini_jeu) == 1)) && ((*page_active) == NIVEAU_2)) ||
00527         ((!(*mini_jeu) && ((*page_active) == NIVEAU_3))) {
00528         /* Boucle de gestion d'événement */
00529         while (SDL_PollEvent(&event)) {
00530             switch (event->type) {
00531                 /* Gestion de l'événement de redimensionnement de la fenêtre */
00532                 case SDL_WINDOWEVENT:
00533                     redimensionnement_fenetre((event), largeur, hauteur);
00534
00535                     (*largeur_tile) = (*largeur) / 32;
00536                     (*hauteur_tile) = (*hauteur) / 18;
00537
00538                     break;
00539
00540                 /* Si une touche au clavier est pressée */
00541                 case SDL_KEYDOWN:
00542                     /* On met les valeurs à 1 pour dire qu'on a appuyer sur la touche correspondante
00543
00544                     /* Cela permet l'appuie de plusieurs touches en même temps */
00545                     if (event->key.keysym.sym == (*touche_sauter_monter))
00546                         (*sauter) = 1;
00547
00548                     if (event->key.keysym.sym == (*touche_aller_a_droite))
00549                         (*avancer) = 1;
00550
00551                     if (event->key.keysym.sym == (*touche_aller_a_gauche))
00552                         (*reculer) = 1;
00553
00554                     if (event->key.keysym.sym == (*touche_interagir)) {
00555                         /* Cas où vous retournez sur la carte */
00556                         if (((*mini_jeu_termine) && ((*position_x) == 20) && ((*position_y) == 16)) {
00557                             /* Effet sonore quand on passe dans une porte */
00558                             if ((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/portes.wav")) ==
00559                                 NULL)
00560                                 erreur("Chargement de l'effet sonore");
00561
00562                             Mix_PlayChannel(1, effet_sonore, 0);
00563
00564                             /* Musique du salon */
00565                             if ((*musique) = Mix_LoadMUS("./sons/musiques/salon.mp3")) == NULL)
00566                                 erreur("Chargement de la musique");
00567
00568                             Mix_PlayMusic((*musique), -1);

```

```

00565             (*mini_jeu) = 0;
00566             (*mini_jeu_2_termine) = 1;
00567
00568             salon_arrivee_niveaux_2_3(position_x, position_y, tile_map,
(*page_active));
00569
00570             tile_map[2][27] = 0;
00571
00572             if((*mini_jeu_1_termine)) {
00573                 tile_map[4][2] = 0;
00574                 tile_map[6][3] = 5;
00575             }
00576         }
00577     }
00578
00579     break;
00580
00581     /* Si une touche au clavier est relâchée */
00582     case SDL_KEYUP:
00583
00584         /* On met les valeurs à 0 pour dire qu'on a relâchée la touche correspondante */
00585         if(event->key.keysym.sym == (*touche_sauter_monter))
00586             (*sauter) = 0;
00587
00588         if(event->key.keysym.sym == (*touche_aller_a_droite))
00589             (*avancer) = 0;
00590
00591         if(event->key.keysym.sym == (*touche_aller_a_gauche))
00592             (*reculer) = 0;
00593
00594     break;
00595
00596     /* Option plein écran */
00597     case SDL_MOUSEBUTTONDOWN:
00598
00599         if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window)) {
00600             redimensionnement_fenetre((*event), largeur, hauteur);
00601
00602             (*largeur_tile) = (*largeur) / 32;
00603             (*hauteur_tile) = (*hauteur) / 18;
00604         }
00605
00606         /* Demande au joueur s'il veut quitter le niveau */
00607         if(clic_case((*event), (*rectangle_croix))) {
00608             SDL_SetWindowResizable((*window), SDL_FALSE);
00609
00610             demande_quitter_niveau(renderer, rectangle_demande,
00611                                     surface, texture_texte, police, couleurNoire,
00612                                     itemsDemandeQuitter, tailleDemande, (*largeur),
00613                                     (*hauteur));
00614
00615             while (!clic_effectue) {
00616                 while (SDL_PollEvent(&event_temporaire)) {
00617                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00618                         if(clic_case(event_temporaire, itemsDemandeQuitter[1].rectangle))
00619                         {
00620                             if((*page_active) == NIVEAU_2)
00621                                 for(i = 0; i < 3; i++)
00622                                     avancee_niveaux[1].numero_collectible[i] =
00623                                     collectibles_intermediaires[i];
00624
00625                             else if((*page_active) == NIVEAU_3)
00626                                 for(i = 0; i < 3; i++)
00627                                     avancee_niveaux[2].numero_collectible[i] =
00628                                     collectibles_intermediaires[i];
00629
00630                             for(i = 0; i < 10; i++)
00631                                 avancee_succes[i] = avancee_succes_intermediaires[i];
00632
00633                             (*page_active) = CARTE;
00634                             clic_effectue = SDL_TRUE;
00635                         }
00636                     }
00637                 }
00638                 else if(clic_case(event_temporaire,
00639                                     itemsDemandeQuitter[2].rectangle))
00640                     clic_effectue = SDL_TRUE;
00641             }
00642         }
00643     }
00644 }
00645

```



```

00646
00647         SDL_SetWindowResizable((*window), SDL_TRUE);
00648     }
00649
00650     break;
00651
00652     /* Quitter le programme en demandant s'il faut sauvergar la partie */
00653     case SDL_QUIT:
00654
00655         SDL_SetWindowResizable((*window), SDL_FALSE);
00656
00657         demande_sauvegarde(renderer, rectangle_demande,
00658                             surface, texture_texte, police, couleurNoire,
00659                             itemsDemandeSauvegarde, tailleDemande, (*largeur),
00660                             (*hauteur));
00661
00662         while (!cllic_effectue) {
00663             while (SDL_PollEvent(&event_temporaire)) {
00664                 if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00665                     if(cllic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {
00666                         if((*page_active) == NIVEAU_2)
00667                             for(i = 0; i < 3; i++)
00668                                 avancee_niveaux[1].numero_collectible[i] =
00669                                     collectibles_intermediaires[i];
00670                         else if((*page_active) == NIVEAU_3)
00671                             for(i = 0; i < 3; i++)
00672                                 avancee_niveaux[2].numero_collectible[i] =
00673                                     collectibles_intermediaires[i];
00674                         sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
00675                                             touche_descendre, touche_interagir,
00676                                             (*modeActif), (*personnageActif),
00677                                             (*positionActive),
00678                                             temps_debut_partie, (*compteur_mort), avancee_succes);
00679
00680                         (*programme_lance) = SDL_FALSE;
00681                         clic_effectue = SDL_TRUE;
00682                     }
00683                     else if(cllic_case(event_temporaire,
00684                                         itemsDemandeSauvegarde[2].rectangle)) {
00685                         (*programme_lance) = SDL_FALSE;
00686                         clic_effectue = SDL_TRUE;
00687                     }
00688                     else if(!cllic_case(event_temporaire, (*rectangle_demande)))
00689                         clic_effectue = SDL_TRUE;
00690                 }
00691             }
00692         }
00693
00694         SDL_SetWindowResizable((*window), SDL_TRUE);
00695
00696         break;
00697
00698     default:
00699         break;
00700 }
00701
00702 /* Déplacement du personnage */
00703 deplacement_personnage(saut, tombe, position_x, position_y, position_avant_saut,
00704                         (*sauter), (*avancer), (*reculer), tile_map, (*personnageActif));
00705
00706 /* Cas où le joueur est dans le second mini jeu */
00707 if((*mini_jeu) == 2) && ((*page_active) == NIVEAU_2) {
00708     /* Cas où le personnage tue un monstre */
00709     if((tile_map[(*position_y) + 1][(*position_x)] == 8) || (tile_map[(*position_y) +
00710 1][(*position_x)] == 9)) {
00711         /* Effet sonore de la mort d'un monstre */
00712         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_monstre.wav")) == NULL)
00713             erreur("Chargement de l'effet sonore");
00714
00715         Mix_PlayChannel(1, effet_sonore, 0);
00716
00717         tile_map[(*position_y) + 1][(*position_x)] = 0;
00718
00719     }
00720 }

```

```

00724         (*tombe) = 0;
00725         (*saut) = 1;
00726
00727         for (y = 0; y < 18; y++)
00728             for (x = 0; x < 32; x++)
00729                 if((tile_map[y][x] == 8) || (tile_map[y][x] == 9))
00730                     (*mini_jeu_termine)++;
00731
00732         if((*modeActif) == MODE_HARD) {
00733             if(!(*mini_jeu_termine) && (!(*mode_difficile))) {
00734                 (*mode_difficile) = 1;
00735                 mini_jeu_2_niveau_2(position_x, position_y, position_x_initiale,
00736 position_y_initiale, tile_map, (*mode_difficile));
00737                 (*mini_jeu_termine) = 0;
00738             }
00739
00740             else if(!(*mini_jeu_termine) && (*mode_difficile))
00741                 (*mini_jeu_termine) = 1;
00742
00743             else
00744                 (*mini_jeu_termine) = 0;
00745         }
00746
00747         else if((*modeActif) == MODE_NORMAL) {
00748             if(!(*mini_jeu_termine))
00749                 (*mini_jeu_termine) = 1;
00750
00751             else
00752                 (*mini_jeu_termine) = 0;
00753         }
00754     }
00755
00756     /* Cas où le personnage meurt par des monstres */
00757     if((tile_map[(*position_y)][(*position_x)] == 8) ||
00758 (tile_map[(*position_y)][(*position_x)] == 9)) {
00759         (*compteur_mort)++;
00760
00761         if((*personnageActif) == PERSONNAGE_1) {
00762             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_masculin.wav")) ==
00763 NULL)
00764                 erreur("Chargement de l'effet sonore");
00765
00766             else if((*personnageActif) == PERSONNAGE_2) {
00767                 if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_feminin.wav")) == NULL)
00768                     erreur("Chargement de l'effet sonore");
00769
00770                 Mix_PlayChannel(1, effet_sonore, 0);
00771
00772                 (*saut) = 0;
00773                 (*tombe) = 0;
00774
00775                 (*mouvement_monstre) = 0;
00776
00777                 (*mode_difficile) = 0;
00778
00779                 mini_jeu_2_niveau_2(position_x, position_y, position_x_initiale, position_y_initiale,
00780 tile_map, (*mode_difficile));
00781             }
00782
00783             /* Déplacement des monstres */
00784
00785             else if(!((time(NULL) - 1) % 4) && !((time(NULL) - 1) % 4))
00786                 (*mouvement_monstre) = 1;
00787
00788             if((*timestamp) < time(NULL) && (*mouvement_monstre)) {
00789                 (*timestamp) = time(NULL);
00790
00791                 for (y = 0; y < 18; y++)
00792                     for (x = 0; x < 32; x++)
00793                         if(tile_map[y][x] == 8) {
00794                             if(!(((*timestamp) % 4) || (!(((*timestamp) - 1) % 4))) {
00795                                 tile_map[y][x] = 0;
00796                                 tile_map[y][x + 1] = 8;
00797                                 x++;
00798                             }
00799
00800                             else if(!(((*timestamp) % 2) || (!(((*timestamp) + 1) % 4))) {

```

```

00807
00808         tile_map[y][x] = 0;
00809         tile_map[y][x - 1] = 8;
00810     }
00811 }
00812
00813     else if(tile_map[y][x] == 9) {
00814
00815         if(!((*timestamp) % 4) || (!((*timestamp) - 1) % 4)) {
00816
00817             tile_map[y][x] = 0;
00818             tile_map[y][x + 1] = 9;
00819             x++;
00820         }
00821
00822         else if(!((*timestamp) % 2) || (!((*timestamp) + 1) % 4)) {
00823
00824             tile_map[y][x] = 0;
00825             tile_map[y][x - 1] = 9;
00826         }
00827     }
00828 }
00829 }
00830 }
00831
00832 /* Si le niveau actif est le niveau 2 */
00833 if((*page_active) == NIVEAU_2) {
00834
00835     /* Cas où le joueur rentre dans le premier dossier */
00836     if(((position_x) >= 2) && ((position_x <= 4)) && ((position_y) >= 4) && ((position_y <=
6)) && !(mini_jeu) && !(mini_jeu_1_termine)) {
00837
00838         /* Musique des mini-jeux */
00839         if(((*musique) = Mix_LoadMUS("./sons/musiques/mini_jeux.mp3")) == NULL)
00840             erreur("Chargement de la musique");
00841
00842         Mix_PlayMusic((*musique), -1);
00843
00844         (*largeur_tile) = (*largeur) / 27;
00845         (*hauteur_tile) = (*hauteur) / 19;
00846
00847         (*mini_jeu) = 1;
00848
00849         (*valide) = 0;
00850
00851         mini_jeu_1_niveau_2(position_x, position_y, tile_map_mini_jeu_niveau_2);
00852
00853         explications(renderer, rectangle_demande, (*touche_interagir), (*touche_sauter_monter),
00854             surface, texture_texte, police, couleurTitre,
00855             itemsExplications, (*largeur), (*hauteur), 1);
00856
00857         SDL_SetWindowResizable((*window), SDL_FALSE);
00858
00859         while(!clic_effectue) {
00860
00861             while(SDL_PollEvent(&event_temporaire))
00862
00863                 if(event_temporaire.type == SDL_MOUSEBUTTONDOWN)
00864
00865                     if(clic_case(event_temporaire, itemsExplications[1].rectangle))
00866                         clic_effectue = SDL_TRUE;
00867         }
00868
00869         SDL_SetWindowResizable((*window), SDL_TRUE);
00870
00871         clic_effectue = SDL_FALSE;
00872     }
00873
00874     /* Cas où le joueur rentre dans le second dossier */
00875     else if(((position_x) >= 27) && ((position_x <= 29)) && ((position_y) >= 2) &&
((position_y <= 4)) && !(mini_jeu) && !(mini_jeu_2_termine)) {
00876
00877         /* Musique des mini-jeux */
00878         if(((*musique) = Mix_LoadMUS("./sons/musiques/mini_jeux.mp3")) == NULL)
00879             erreur("Chargement de la musique");
00880
00881         Mix_PlayMusic((*musique), -1);
00882
00883         (*mini_jeu) = 2;
00884
00885         (*sauter) = 0;
00886         (*avancer) = 0;
00887         (*reculer) = 0;
00888         (*tombe) = 0;
00889         (*saut) = 0;
00890
00891         mini_jeu_2_niveau_2(position_x, position_y, position_x_initiale, position_y_initiale,

```

```

    tile_map, (*mode_difficile));
00892
00893     explications(renderer, rectangle_demande, (*touche_interagir), (*touche_sauter_monter),
00894                 surface, texture_texte, police, couleurTitre,
00895                 itemsExplications, (*largeur), (*hauteur), 2);
00896
00897     SDL_SetWindowResizable((*window), SDL_FALSE);
00898
00899     while(!clic_effectue) {
00900         while(SDL_PollEvent(&event_temporaire))
00901             if(event_temporaire.type == SDL_MOUSEBUTTONDOWN)
00902                 if(clic_case(event_temporaire, itemsExplications[1].rectangle))
00903                     clic_effectue = SDL_TRUE;
00904     }
00905
00906     SDL_SetWindowResizable((*window), SDL_TRUE);
00907
00908     clic_effectue = SDL_FALSE;
00909 }
00910
00911 if((*mini_jeu_1_termine) && (*mini_jeu_2_termine)) {
00912     tile_map[8][18] = 5;
00913     tile_map[16][16] = 7;
00914 }
00915
00916 /* Cas où le joueur récupère un collectible dans le niveau 2 */
00917
00918 if((!(*mini_jeu)) && (tile_map[(position_y)][(position_x)] == 5) && ((position_y) == 6) &&
00919 (!avancee_niveaux[1].numero_collectible[0])) {
00920
00921     /* Effet sonore quand on ramasse un collectible */
00922     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00923         erreur("Chargement de l'effet sonore");
00924
00925     Mix_PlayChannel(1, effet_sonore, 0);
00926
00927     avancee_niveaux[1].numero_collectible[0] = 1;
00928 }
00929
00930 if(((*mini_jeu) == 2) && (tile_map[(position_y)][(position_x)] == 5) &&
00931 (!avancee_niveaux[1].numero_collectible[1])) {
00932
00933     /* Effet sonore quand on ramasse un collectible */
00934     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00935         erreur("Chargement de l'effet sonore");
00936
00937     Mix_PlayChannel(1, effet_sonore, 0);
00938
00939     avancee_niveaux[1].numero_collectible[1] = 1;
00940 }
00941
00942 if((!(*mini_jeu)) && (tile_map[(position_y)][(position_x)] == 5) && ((position_y) == 8) &&
00943 (!avancee_niveaux[1].numero_collectible[2])) {
00944
00945     /* Effet sonore quand on ramasse un collectible */
00946     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00947         erreur("Chargement de l'effet sonore");
00948
00949     Mix_PlayChannel(1, effet_sonore, 0);
00950
00951     avancee_niveaux[1].numero_collectible[2] = 1;
00952 }
00953
00954 /* Cas où le joueur est dans un mini jeu */
00955 if((*mini_jeu))
00956     /* Mise à jour du rendu du mini jeu */
00957     mini_jeux_niveau_2(event, renderer, window, programme_lance, texture_image_fond,
00958 texture_image_sol,
00959 rectangle_plein_ecran, texture_image_plein_ecran, plein_ecran,
00960 texture_image_porte, avancee_niveaux,
00961 texture, rectangle_tile, mini_jeu, mini_jeu_1_termine,
00962 mini_jeu_2_termine,
00963 texture_image_personnage, rectangle_personnage, (*mini_jeu_termine),
00964 position_x, position_y, tile_map, tile_map_mini_jeu_niveau_2,
00965 texture_image_monstre_terrestre, texture_image_monstre_volant,
00966 largeur, hauteur, largeur_tile, hauteur_tile, texture_image_croix,
00967 rectangle_croix,
00968 texture_image_mur_mini_jeu, touche_aller_a_droite,
00969 touche_aller_a_gauche, touche_interagir,
00970 touche_sauter_monter, touche_descendre, valide,
00971 texture_image_pipe_vertical, texture_image_pipe_horizontal,
00972 texture_image_pipe_haut_droit, texture_image_pipe_bas_droit,
00973 texture_image_pipe_bas_gauche, texture_image_pipe_haut_gauche,
00974 texture_image_pipe_courant, rectangle_demande,

```

```

00969         surface, texture_texte, police, couleurNoire,
00970         itemsDemandeQuitter, tailleDemande, collectibles_intermediaires,
00971         texture_image_mur_termine, page_active, musique,
00972         avancer, reculer, sauter, saut, tombe,
00973         itemsDemandeSauvegarde, barre_de_son, pseudo,
00974         modeActif, personnageActif, positionActive, tailleNiveaux,
00975         temps_debut_partie, compteur_mort, avancee_succes,
    avancee_succes_intermediaires);
00976     }
00977
00978     /* Si le niveau actif est le niveau 3 */
00979     else if((*page_active) == NIVEAU_3) {
00980
00981         /* Cas où le joueur rentre dans le premier dossier */
00982         if(((position_x) >= 5) && ((position_x <= 7)) && ((position_y) >= 3) && ((position_y <=
5)) && (!(mini_jeu)) && (!(mini_jeu_1_termine))) {
00983
00984             /* Musique des mini-jeux */
00985             if(((*musique) = Mix_LoadMUS("./sons/musiques/mini_jeux.mp3")) == NULL)
00986                 erreur("Chargement de la musique");
00987
00988             Mix_PlayMusic((*musique), -1);
00989
00990             (*mini_jeu) = 1;
00991
00992             (*piece_selectionnee) = -1;
00993
00994             (*decalage_x) = 0;
00995             (*decalage_y) = 0;
00996
00997             for(i = 0; i < 45; i++)
00998                 piece_bloquee[i] = 0;
00999
01000             explications(renderer, rectangle_demande, (*touche_interagir), (*touche_sauter_monter),
01001                 surface, texture_texte, police, couleurTitre,
01002                 itemsExplications, (*largeur), (*hauteur), 3);
01003
01004             SDL_SetWindowResizable((*window), SDL_FALSE);
01005
01006             while(!clic_effectue) {
01007
01008                 while(SDL_PollEvent(&event_temporaire))
01009
01010                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN)
01011
01012                         if(clic_case(event_temporaire, itemsExplications[1].rectangle))
01013                             clic_effectue = SDL_TRUE;
01014             }
01015
01016             clic_effectue = SDL_FALSE;
01017
01018             if((*plein_ecran))
01019                 SDL_SetWindowFullscreen((*window), 0);
01020
01021             SDL_SetWindowSize((*window), 1600, 520);
01022             SDL_SetWindowPosition((*window), SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED);
01023
01024             /* Initialisation de srand pour la génération de pièces aléatoires*/
01025             srand(time(NULL));
01026
01027             /* Calcul des rectangles pour chaque pièce du puzzle et des emplacements corrects */
01028             for (y = 0; y < 5; y++)
01029
01030                 for (x = 0; x < 9; x++) {
01031
01032                     rectangle_piece[y * 9 + x].w = 1600 / 9;
01033                     rectangle_piece[y * 9 + x].h = 520 / 5;
01034                     rectangle_piece[y * 9 + x].x = x * 1600 / 9;
01035                     rectangle_piece[y * 9 + x].y = y * 520 / 5;
01036
01037                     rectangle_emplacement_piece[y * 9 + x].x = x * 1600 / 9;
01038                     rectangle_emplacement_piece[y * 9 + x].y = y * 520 / 5;
01039                     rectangle_emplacement_piece[y * 9 + x].w = 1600 / 9;
01040                     rectangle_emplacement_piece[y * 9 + x].h = 520 / 5;
01041                 }
01042
01043             for (i = 0; i < 45; i++)
01044                 rectangle_piece[i] = rectangle_piece_aleatoire(1600, 520);
01045
01046             SDL_SetWindowSize((*window), 1600, 520);
01047             SDL_SetWindowPosition((*window), SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED);
01048         }
01049
01050         /* Cas où le joueur rentre dans le second dossier */
01051         else if(((position_x) >= 24) && ((position_x <= 26)) && ((position_y) >= 3) &&
((position_y <= 5)) && !(mini_jeu) && !(mini_jeu_2_termine))) {
01052

```

```

01053      /* Musique des mini-jeux */
01054      if ((*musique) = Mix_LoadMUS("./sons/musiques/mini_jeux.mp3")) == NULL)
01055          erreur("Chargement de la musique");
01056
01057      Mix_PlayMusic((*musique), -1);
01058
01059      (*mini_jeu) = 2;
01060
01061      (*descendre) = 0;
01062      (*interagir) = 0;
01063      (*sauter) = 0;
01064      (*reculer) = 0;
01065      (*avancer) = 0;
01066
01067      (*largeur_tile) = (*largeur) / 32;
01068      (*hauteur_tile) = (*hauteur) / 24;
01069
01070      mini_jeu_2_niveau_3(position_x, position_y, bloc_x, bloc_y, tile_map_mini_jeu_niveau_3);
01071
01072      explications(renderer, rectangle_demande, (*touche_interagir), (*touche_sauter_monter),
01073                  surface, texture_texte, police, couleurTitre,
01074                  itemsExplications, (*largeur), (*hauteur), 4);
01075
01076      SDL_SetWindowResizable((*window), SDL_FALSE);
01077
01078      while (!cllic_effectue) {
01079          while (SDL_PollEvent(&event_temporaire))
01080              if (event_temporaire.type == SDL_MOUSEBUTTONDOWN)
01081                  if (cllic_case(event_temporaire, itemsExplications[1].rectangle))
01082                      clic_effectue = SDL_TRUE;
01083              }
01084
01085      SDL_SetWindowResizable((*window), SDL_TRUE);
01086
01087      clic_effectue = SDL_FALSE;
01088  }
01089
01090  if ((*mini_jeu_1_termine) && (*mini_jeu_2_termine)) {
01091      tile_map[13][25] = 5;
01092      tile_map[16][16] = 7;
01093  }
01094
01095  /* Cas où le joueur récupère un collectible dans le niveau 3 */
01096
01097  if (!(*mini_jeu) && (tile_map[(*position_y)][(*position_x)] == 5) && ((*position_y) == 5) &&
01098      (!avancee_niveaux[2].numero_collectible[0])) {
01099
01100      /* Effet sonore quand on ramasse un collectible */
01101      if ((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
01102          erreur("Chargement de l'effet sonore");
01103
01104      Mix_PlayChannel(1, effet_sonore, 0);
01105
01106      avancee_niveaux[2].numero_collectible[0] = 1;
01107  }
01108
01109  if (!(*mini_jeu) && (tile_map[(*position_y)][(*position_x)] == 5) && ((*position_y) == 13) &&
01110      (!avancee_niveaux[2].numero_collectible[2])) {
01111
01112      /* Effet sonore quand on ramasse un collectible */
01113      if ((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
01114          erreur("Chargement de l'effet sonore");
01115
01116      Mix_PlayChannel(1, effet_sonore, 0);
01117
01118      avancee_niveaux[2].numero_collectible[2] = 1;
01119  }
01120
01121  /* Cas où le joueur est dans un mini jeu */
01122  if ((*mini_jeu))
01123      /* Mise à jour du rendu du mini jeu */
01124      mini_jeux_niveau_3(event, renderer, window, programme_lance,
01125                          rectangle_plein_ecran, texture_image_plein_ecran, plein_ecran,
01126                          avancee_niveaux, tile_map, texture_image_croix, rectangle_croix,
01127                          mini_jeu_1_termine, mini_jeu_2_termine,
01128                          position_x, position_y, texture,
01129                          largeur, hauteur, rectangle_demande,
01130                          surface, texture_texte, police, couleurNoire,
01131                          itemsDemandeQuitter, tailleDemande, collectibles_intermediaires,
01132                          page_active, rectangle_tile, largeur_tile, hauteur_tile,
01133                          avancer, reculer, sauter, saut, tombe,
01134                          rectangle_piece, piece_bloquee, rectangle_emplacement_piece,
01135                          piece_selectionnee,
01136                          decalage_x, decalage_y, texture_image_puzzle,

```

```

01137         tile_map_mini_jeu_niveau_3, descendre, interagir, bloc_x, bloc_y,
01138         texture_image_sol_labyrinthe, texture_image_bordure_labyrinthe,
01139         texture_image_fin_labyrinthe, musique,
01140         texture_image_personnage, rectangle_personnage,
01141         texture_image_mur_termine, texture_image_mur_mini_jeu,
01142         touche_aller_a_droite, touche_aller_a_gauche, touche_interagir,
01143         touche_sauter_monter, touche_descendre, modeActif,
01144         itemsDemandeSauvegarde, barre_de_son, pseudo,
01145         personnageActif, positionActive, tailleNiveaux,
01146         temps_debut_partie, compteur_mort, avancee_succes,
    avancee_succes_intermediaires);
01147     }
01148
01149     /* Cas où vous avez fini le niveau */
01150     if(!(*mini_jeu) && (tile_map[(position_y)][(position_x)] == 7)) {
01151
01152         /* Effet sonore quand on finit un niveau */
01153         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/fin_niveaux.wav")) == NULL)
01154             erreur("Chargement de l'effet sonore");
01155
01156         Mix_PlayChannel(1, effet_sonore, 0);
01157
01158         /* Mise à jour du rendu du salon */
01159         mise_a_jour_rendu_arrivee_niveaux_2_3(renderer, texture_image_fond, texture_image_sol,
01160         rectangle_plein_ecran, texture_image_plein_ecran,
01161         texture_image_fin_premiers_niveaux,
01162         texture, rectangle_tile, texture_image_dossier,
01163         barre_windows_1, barre_windows_2, barre_windows_3,
01164         rectangle_personnage,
01165         barre_windows_4, texture_image_perso_gagnant,
01166         (*position_x), (*position_y), tile_map,
01167         avancee_niveaux, texture_image_croix, rectangle_croix,
01168         (*largeur), (*hauteur), (*largeur_tile),
01169         (*hauteur_tile), (*page_active));
01170
01171         SDL_Delay(1000);
01172
01173         if((*page_active) == NIVEAU_2)
01174             avancee_niveaux[1].niveau_fini = 1;
01175
01176         else if((*page_active) == NIVEAU_3)
01177             avancee_niveaux[2].niveau_fini = 1;
01178
01179         (*page_active) = CARTE;
01180     }
01181     /* Cas où le joueur n'est pas dans un mini jeu */
01182     if(!(*mini_jeu))
01183         /* Mise à jour du rendu du salon */
01184         mise_a_jour_rendu_arrivee_niveaux_2_3(renderer, texture_image_fond, texture_image_sol,
01185         rectangle_plein_ecran, texture_image_plein_ecran,
01186         texture_image_fin_premiers_niveaux,
01187         texture, rectangle_tile, texture_image_dossier,
01188         barre_windows_1, barre_windows_2, barre_windows_3,
01189         barre_windows_4, texture_image_personnage,
01190         rectangle_personnage,
01191         (*position_x), (*position_y), tile_map, avancee_niveaux,
01192         texture_image_croix, rectangle_croix,
01193         (*largeur), (*hauteur), (*largeur_tile),
01194         (*hauteur_tile), (*page_active));
01195 }

```

## 4.3 Référence du fichier fichiers\_c/fonctions\_carte.c

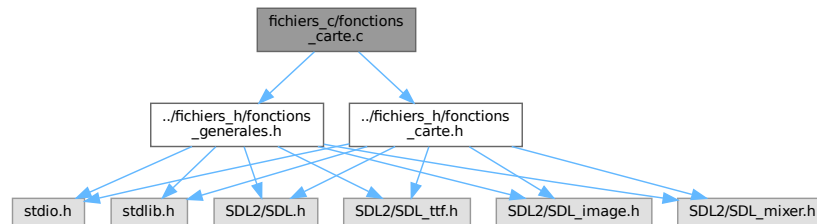
Fichier contenant toutes les fonctions gérant la carte principal.

```

#include <../fichiers_h/fonctions_generales.h>
#include <../fichiers_h/fonctions_carte.h>

```

Graphe des dépendances par inclusion de fonctions\_carte.c:



## Fonctions

- void [initialisation\\_objets\\_carte](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_carte, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_1, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_2, SDL\_Texture \*\*texture\_image\_perso\_1\_haut\_1, SDL\_Texture \*\*texture\_image\_perso\_1\_haut\_2, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_gauche\_1, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_gauche\_2, SDL\_Texture \*\*texture\_image\_perso\_1\_haut, SDL\_Texture \*\*texture\_image\_perso\_1\_droite, SDL\_Texture \*\*texture\_image\_perso\_1\_gauche, SDL\_Texture \*\*texture\_image\_perso\_1\_pose, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_1, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_2, SDL\_Texture \*\*texture\_image\_perso\_2\_haut\_1, SDL\_Texture \*\*texture\_image\_perso\_2\_haut\_2, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_gauche\_1, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_gauche\_2, SDL\_Texture \*\*texture\_image\_perso\_2\_haut, SDL\_Texture \*\*texture\_image\_perso\_2\_droite, SDL\_Texture \*\*texture\_image\_perso\_2\_gauche, SDL\_Texture \*\*texture\_image\_perso\_2\_pose, [itemMenu](#) \*itemsNiveaux, SDL\_Texture \*\*texture\_image\_retour\_menu, [itemMenu](#) \*itemsSucces, SDL\_Texture \*\*textures\_images\_succes)
- void [mise\\_a\\_jour\\_rendu\\_carte](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_carte, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, SDL\_Rect \*rectangle\_perso, SDL\_Texture \*\*texture\_image\_perso, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Rect \*rectangle\_succes, [position\\_t](#) positionActive, SDL\_Color couleurNoire, SDL\_Rect \*rectangle\_retour\_menu, SDL\_Texture \*\*texture\_image\_retour\_menu, [itemMenu](#) \*itemsNiveaux, int tailleNiveaux, int largeur, int hauteur, [niveaux](#) \*avancee\_niveaux)
- void [deplacement\\_personnage\\_carte](#) (SDL\_Renderer \*\*renderer, SDL\_Window \*\*window, SDL\_Texture \*\*texture\_image\_carte, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, SDL\_Rect \*rectangle\_perso, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Rect \*rectangle\_succes, [position\\_t](#) \*positionActive, SDL\_Color couleurNoire, SDL\_Rect \*rectangle\_retour\_menu, SDL\_Texture \*\*texture\_image\_retour\_menu, [itemMenu](#) \*itemsNiveaux, int tailleNiveaux, int largeur, int hauteur, int valeur\_maximale, [direction\\_t](#) direction, [niveaux](#) \*avancee\_niveaux)
- void [carte](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_carte, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, SDL\_Rect \*rectangle\_retour\_menu, SDL\_Texture \*\*texture\_image\_retour\_menu, SDL\_Texture \*\*texture\_image\_perso\_bas\_1, SDL\_Texture \*\*texture\_image\_perso\_bas\_2, SDL\_Texture \*\*texture\_image\_perso\_haut\_1, SDL\_Texture \*\*texture\_image\_perso\_haut\_2, SDL\_Texture \*\*texture\_image\_perso\_bas\_gauche\_1, SDL\_Texture \*\*texture\_image\_perso\_bas\_gauche\_2, SDL\_Texture \*\*texture\_image\_perso\_haut, SDL\_Texture \*\*texture\_image\_perso\_droite, SDL\_Texture \*\*texture\_image\_perso\_gauche, SDL\_Texture \*\*texture\_image\_perso\_pose, SDL\_Texture \*\*texture\_image\_perso, SDL\_Rect \*rectangle\_perso, [niveaux](#) \*avancee\_niveaux, int niveau\_fini[4], int collectibles[12], [position\\_t](#) \*position\_intermediaire, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Rect \*rectangle\_succes, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, [direction\\_t](#) \*direction, int \*touche\_pressee, SDL\_Rect \*rectangle\_demande\_sauvegarde, [itemMenu](#) \*itemsDemandeSauvegarde, int tailleDemandeSauvegarde, [position\\_t](#) \*positionActive, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [modes\\_t](#)



```
*modeActif, personnage\_t *personnageActif, SDL_Color couleurNoire, SDL_Keycode *touche_aller_↵
a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter, SDL_Keycode
*touche_descendre, SDL_Keycode *touche_interagir, itemMenu *itemsNiveaux, int tailleNiveaux, int
*largeur, int *hauteur, page\_t *page_active, itemMenu *itemsSucces, SDL_Texture **textures_images_↵
_succes, time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int avancee_succes_↵
intermediaires[10])
```

### 4.3.1 Description détaillée

Fichier contenant toutes les fonctions gérant la carte principal.

Définition dans le fichier [fonctions\\_carte.c](#).

### 4.3.2 Documentation des fonctions

#### 4.3.2.1 carte()

```
void carte (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_carte,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    SDL_Rect * rectangle_retour_menu,
    SDL_Texture ** texture_image_retour_menu,
    SDL_Texture ** texture_image_perso_bas_1,
    SDL_Texture ** texture_image_perso_bas_2,
    SDL_Texture ** texture_image_perso_haut_1,
    SDL_Texture ** texture_image_perso_haut_2,
    SDL_Texture ** texture_image_perso_bas_gauche_1,
    SDL_Texture ** texture_image_perso_bas_gauche_2,
    SDL_Texture ** texture_image_perso_haut,
    SDL_Texture ** texture_image_perso_droite,
    SDL_Texture ** texture_image_perso_gauche,
    SDL_Texture ** texture_image_perso_pose,
    SDL_Texture ** texture_image_perso,
    SDL_Rect * rectangle_perso,
    niveaux * avancee_niveaux,
    int niveau_fini[4],
    int collectibles[12],
    position\_t * position_intermediaire,
    SDL_Texture ** texture_image_fin_dernier_niveau,
    SDL_Rect * rectangle_succes,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    direction\_t * direction,
    int * touche_pressee,
    SDL_Rect * rectangle_demande_sauvegarde,
```

```

itemMenu * itemsDemandeSauvegarde,
int tailleDemandeSauvegarde,
position_t * positionActive,
barreDeSon * barre_de_son,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
SDL_Color couleurNoire,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
SDL_Keycode * touche_interagir,
itemMenu * itemsNiveaux,
int tailleNiveaux,
int * largeur,
int * hauteur,
page_t * page_active,
itemMenu * itemsSucces,
SDL_Texture ** textures_images_succes,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 539 du fichier [fonctions\\_carte.c](#).

#### 4.3.2.2 `deplacement_personnage_carte()`

```

void deplacement_personnage_carte (
    SDL_Renderer ** renderer,
    SDL_Window ** window,
    SDL_Texture ** texture_image_carte,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    SDL_Rect * rectangle_perso,
    SDL_Texture ** texture_image_perso_1,
    SDL_Texture ** texture_image_perso_2,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Texture ** texture_image_fin_dernier_niveau,
    SDL_Rect * rectangle_succes,
    position_t * positionActive,
    SDL_Color couleurNoire,
    SDL_Rect * rectangle_retour_menu,
    SDL_Texture ** texture_image_retour_menu,
    itemMenu * itemsNiveaux,
    int tailleNiveaux,
    int largeur,
    int hauteur,
    int valeur_maximale,
    direction_t direction,
    niveaux * avancee_niveaux )

```

Définition à la ligne 376 du fichier [fonctions\\_carte.c](#).

## 4.3.2.3 initialisation\_objets\_carte()

```

void initialisation_objets_carte (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_carte,
    SDL_Texture ** texture_image_perso_1_bas_1,
    SDL_Texture ** texture_image_perso_1_bas_2,
    SDL_Texture ** texture_image_perso_1_haut_1,
    SDL_Texture ** texture_image_perso_1_haut_2,
    SDL_Texture ** texture_image_perso_1_bas_gauche_1,
    SDL_Texture ** texture_image_perso_1_bas_gauche_2,
    SDL_Texture ** texture_image_perso_1_haut,
    SDL_Texture ** texture_image_perso_1_droite,
    SDL_Texture ** texture_image_perso_1_gauche,
    SDL_Texture ** texture_image_perso_1_pose,
    SDL_Texture ** texture_image_perso_2_bas_1,
    SDL_Texture ** texture_image_perso_2_bas_2,
    SDL_Texture ** texture_image_perso_2_haut_1,
    SDL_Texture ** texture_image_perso_2_haut_2,
    SDL_Texture ** texture_image_perso_2_bas_gauche_1,
    SDL_Texture ** texture_image_perso_2_bas_gauche_2,
    SDL_Texture ** texture_image_perso_2_haut,
    SDL_Texture ** texture_image_perso_2_droite,
    SDL_Texture ** texture_image_perso_2_gauche,
    SDL_Texture ** texture_image_perso_2_pose,
    itemMenu * itemsNiveaux,
    SDL_Texture ** texture_image_retour_menu,
    itemMenu * itemsSucces,
    SDL_Texture ** textures_images_succes )

```

Définition à la ligne 39 du fichier [fonctions\\_carte.c](#).

## 4.3.2.4 mise\_a\_jour\_rendu\_carte()

```

void mise_a_jour_rendu_carte (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_carte,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    SDL_Rect * rectangle_perso,
    SDL_Texture ** texture_image_perso,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Texture ** texture_image_fin_dernier_niveau,
    SDL_Rect * rectangle_succes,
    position_t positionActive,
    SDL_Color couleurNoire,
    SDL_Rect * rectangle_retour_menu,
    SDL_Texture ** texture_image_retour_menu,
    itemMenu * itemsNiveaux,
    int tailleNiveaux,
    int largeur,

```

```

    int hauteur,
    niveaux * avancee_niveaux )

```

Définition à la ligne 141 du fichier [fonctions\\_carte.c](#).

## 4.4 fonctions\_carte.c

Aller à la documentation de ce fichier.

```

00001 /**
00002  * \file fonctions_carte.c
00003  * \brief Fichier contenant toutes les fonctions gérant la carte principal
00004  */
00005
00006 #include <../fichiers_h/fonctions_generales.h>
00007 #include <../fichiers_h/fonctions_carte.h>
00008
00009 /**
00010  * \fn void initialisation_objets_carte(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_carte, SDL_Texture **texture_image_perso_1_bas_1, SDL_Texture
**texture_image_perso_1_bas_2, SDL_Texture **texture_image_perso_1_haut_1, SDL_Texture
**texture_image_perso_1_haut_2, SDL_Texture **texture_image_perso_1_bas_gauche_1, SDL_Texture
**texture_image_perso_1_bas_gauche_2, SDL_Texture **texture_image_perso_1_haut, SDL_Texture
**texture_image_perso_1_droite, SDL_Texture **texture_image_perso_1_gauche, SDL_Texture
**texture_image_perso_1_pose, SDL_Texture **texture_image_perso_2_bas_1, SDL_Texture
**texture_image_perso_2_bas_2, SDL_Texture **texture_image_perso_2_haut_1, SDL_Texture
**texture_image_perso_2_haut_2, SDL_Texture **texture_image_perso_2_bas_gauche_1, SDL_Texture
**texture_image_perso_2_bas_gauche_2, SDL_Texture **texture_image_perso_2_haut, SDL_Texture
**texture_image_perso_2_droite, SDL_Texture **texture_image_perso_2_gauche, SDL_Texture
**texture_image_perso_2_pose, itemMenu *itemsNiveaux, SDL_Texture **texture_image_retour_menu)
00011  * \brief Fonction qui permet d'initialiser les différents objets de la carte.
00012  * \param renderer Pointeur vers le renderer SDL.
00013  * \param surface Surface SDL.
00014  * \param texture_image_carte Texture de l'image de la carte.
00015  * \param texture_image_perso_1_bas_1 Texture de l'image du personnage 1 en bas 1.
00016  * \param texture_image_perso_1_bas_2 Texture de l'image du personnage 1 en bas 2.
00017  * \param texture_image_perso_1_haut_1 Texture de l'image du personnage 1 en haut 1.
00018  * \param texture_image_perso_1_haut_2 Texture de l'image du personnage 1 en haut 2.
00019  * \param texture_image_perso_1_bas_gauche_1 Texture de l'image du personnage 1 en bas gauche 1.
00020  * \param texture_image_perso_1_bas_gauche_2 Texture de l'image du personnage 1 en bas gauche 2.
00021  * \param texture_image_perso_1_haut Texture de l'image du personnage 1 en haut.
00022  * \param texture_image_perso_1_droite Texture de l'image du personnage 1 à droite.
00023  * \param texture_image_perso_1_gauche Texture de l'image du personnage 1 à gauche.
00024  * \param texture_image_perso_1_pose Texture de l'image du personnage 1 en pose.
00025  * \param texture_image_perso_2_bas_1 Texture de l'image du personnage 2 en bas 1.
00026  * \param texture_image_perso_2_bas_2 Texture de l'image du personnage 2 en bas 2.
00027  * \param texture_image_perso_2_haut_1 Texture de l'image du personnage 2 en haut 1.
00028  * \param texture_image_perso_2_haut_2 Texture de l'image du personnage 2 en haut 2.
00029  * \param texture_image_perso_2_bas_gauche_1 Texture de l'image du personnage 2 en bas gauche 1.
00030  * \param texture_image_perso_2_bas_gauche_2 Texture de l'image du personnage 2 en bas gauche 2.
00031  * \param texture_image_perso_2_haut Texture de l'image du personnage 2 en haut.
00032  * \param texture_image_perso_2_droite Texture de l'image du personnage 2 à droite.
00033  * \param texture_image_perso_2_gauche Texture de l'image du personnage 2 à gauche.
00034  * \param texture_image_perso_2_pose Texture de l'image du personnage 2 en pose.
00035  * \param itemsNiveaux Tableau d'items pour les niveaux.
00036  * \param texture_image_retour_menu Texture de l'image du bouton retour au menu.
00037  * \see chargement_image
00038  */
00039 void initialisation_objets_carte(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_carte,
00040                                SDL_Texture **texture_image_perso_1_bas_1, SDL_Texture
**texture_image_perso_1_bas_2,
00041                                SDL_Texture **texture_image_perso_1_haut_1, SDL_Texture
**texture_image_perso_1_haut_2,
00042                                SDL_Texture **texture_image_perso_1_bas_gauche_1, SDL_Texture
**texture_image_perso_1_bas_gauche_2,
00043                                SDL_Texture **texture_image_perso_1_haut, SDL_Texture
**texture_image_perso_1_droite,
00044                                SDL_Texture **texture_image_perso_1_gauche, SDL_Texture
**texture_image_perso_1_pose,
00045                                SDL_Texture **texture_image_perso_2_bas_1, SDL_Texture
**texture_image_perso_2_bas_2,
00046                                SDL_Texture **texture_image_perso_2_haut_1, SDL_Texture
**texture_image_perso_2_haut_2,
00047                                SDL_Texture **texture_image_perso_2_bas_gauche_1, SDL_Texture
**texture_image_perso_2_bas_gauche_2,
00048                                SDL_Texture **texture_image_perso_2_haut, SDL_Texture
**texture_image_perso_2_droite,
00049                                SDL_Texture **texture_image_perso_2_gauche, SDL_Texture
**texture_image_perso_2_pose,
00050                                itemMenu *itemsNiveaux, SDL_Texture **texture_image_retour_menu,

```

```

00051             itemMenu *itemsSucces, SDL_Texture **textures_images_succes) {
00052
00053     /* Initialisation de l'image de fond de la carte */
00054     chargement_image(renderer, surface, texture_image_carte, "./images/carte.jpg");
00055
00056     /* Initialisation des différentes images des personnages */
00057
00058     chargement_image(renderer, surface, texture_image_perso_1_bas_1,
00059     "./images/personnages/personnage_masculin_bas_1.png");
00059     chargement_image(renderer, surface, texture_image_perso_1_bas_2,
00060     "./images/personnages/personnage_masculin_bas_2.png");
00060     chargement_image(renderer, surface, texture_image_perso_1_haut_1,
00061     "./images/personnages/personnage_masculin_haut_1.png");
00061     chargement_image(renderer, surface, texture_image_perso_1_haut_2,
00062     "./images/personnages/personnage_masculin_haut_2.png");
00062     chargement_image(renderer, surface, texture_image_perso_1_bas_gauche_1,
00063     "./images/personnages/personnage_masculin_bas_gauche_1.png");
00063     chargement_image(renderer, surface, texture_image_perso_1_bas_gauche_2,
00064     "./images/personnages/personnage_masculin_bas_gauche_2.png");
00064     chargement_image(renderer, surface, texture_image_perso_1_haut,
00065     "./images/personnages/personnage_masculin_haut.png");
00065     chargement_image(renderer, surface, texture_image_perso_1_droite,
00066     "./images/personnages/personnage_masculin_droite.png");
00066     chargement_image(renderer, surface, texture_image_perso_1_gauche,
00067     "./images/personnages/personnage_masculin_gauche.png");
00067     chargement_image(renderer, surface, texture_image_perso_1_pose,
00068     "./images/personnages/personnage_masculin_pose.png");
00068     chargement_image(renderer, surface, texture_image_perso_2_bas_1,
00069     "./images/personnages/personnage_feminin_bas_1.png");
00069     chargement_image(renderer, surface, texture_image_perso_2_bas_2,
00070     "./images/personnages/personnage_feminin_bas_2.png");
00070     chargement_image(renderer, surface, texture_image_perso_2_haut_1,
00071     "./images/personnages/personnage_feminin_haut_1.png");
00071     chargement_image(renderer, surface, texture_image_perso_2_haut_2,
00072     "./images/personnages/personnage_feminin_haut_2.png");
00072     chargement_image(renderer, surface, texture_image_perso_2_bas_gauche_1,
00073     "./images/personnages/personnage_feminin_bas_gauche_1.png");
00073     chargement_image(renderer, surface, texture_image_perso_2_bas_gauche_2,
00074     "./images/personnages/personnage_feminin_bas_gauche_2.png");
00074     chargement_image(renderer, surface, texture_image_perso_2_haut,
00075     "./images/personnages/personnage_feminin_haut.png");
00075     chargement_image(renderer, surface, texture_image_perso_2_droite,
00076     "./images/personnages/personnage_feminin_droite.png");
00076     chargement_image(renderer, surface, texture_image_perso_2_gauche,
00077     "./images/personnages/personnage_feminin_gauche.png");
00077     chargement_image(renderer, surface, texture_image_perso_2_pose,
00078     "./images/personnages/personnage_feminin_pose.png");
00078
00079     /* Initialisation de l'image de retour au menu principal */
00080     chargement_image(renderer, surface, texture_image_retour_menu, "./images/menu.png");
00081
00082     /* Initialisation des images des succès */
00083     chargement_image(renderer, surface, &(textures_images_succes[0]),
00084     "./images/succes/succes_non_accorde.png");
00084     chargement_image(renderer, surface, &(textures_images_succes[1]), "./images/succes/succes_1.png");
00085     chargement_image(renderer, surface, &(textures_images_succes[2]), "./images/succes/succes_2.png");
00086     chargement_image(renderer, surface, &(textures_images_succes[3]), "./images/succes/succes_3.png");
00087     chargement_image(renderer, surface, &(textures_images_succes[4]), "./images/succes/succes_4.png");
00088     chargement_image(renderer, surface, &(textures_images_succes[5]), "./images/succes/succes_5.png");
00089     chargement_image(renderer, surface, &(textures_images_succes[6]), "./images/succes/succes_6.png");
00090     chargement_image(renderer, surface, &(textures_images_succes[7]), "./images/succes/succes_7.png");
00091     chargement_image(renderer, surface, &(textures_images_succes[8]), "./images/succes/succes_8.png");
00092     chargement_image(renderer, surface, &(textures_images_succes[9]), "./images/succes/succes_9.png");
00093     chargement_image(renderer, surface, &(textures_images_succes[10]),
00094     "./images/succes/succes_10.png");
00094
00095     /* Initialisation du texte dans les items de la carte */
00096     sprintf(itemsNiveaux[0].texte, " Le Commencement ");
00097     sprintf(itemsNiveaux[1].texte, " Bienvenue dans Linux ");
00098     sprintf(itemsNiveaux[2].texte, " Windows XP ");
00099     sprintf(itemsNiveaux[3].texte, " La Tour Infernale ");
00100
00101     /* Initialisation du texte dans les items des succès */
00102     sprintf(itemsSucces[0].texte, " Succes ");
00103     sprintf(itemsSucces[1].texte, " C'est bien ! : Finir le jeu
00104 ");
00104     sprintf(itemsSucces[2].texte, " Collectionneur : Avoir tous les collectibles du jeu
00105 ");
00105     sprintf(itemsSucces[3].texte, " T'as un train ? : Finir le jeu en moins de 10 minutes
00106 ");
00106     sprintf(itemsSucces[4].texte, " La vie est dure : Finir le jeu en mode difficile
00107 ");
00107     sprintf(itemsSucces[5].texte, " Pas de vermines ici ! : Tuer tous les monstres du jeu
00108 ");
00108     sprintf(itemsSucces[6].texte, " Tu jouais ta vie ? : Finir le jeu sans mourir
00109 ");
00109     sprintf(itemsSucces[7].texte, " Maitre du jeu : Finir le jeu en mode difficile et sans mourir

```

```

    ");
00110     sprintf(itemsSucces[8].texte, " Tricheur va ! : Activer le code de triche
    ");
00111     sprintf(itemsSucces[9].texte, " Tu transpires ? : Finir le jeu en mode difficile, en moins de 10
    minutes et sans mourir ");
00112     sprintf(itemsSucces[10].texte, " Dieu du jeu : Obtenir tous les succes (sans compter \"Tricheur va
    !\")");
00113     sprintf(itemsSucces[11].texte, " Fermer ");
00114 }
00115
00116 /** \fn void mise_a_jour_rendu_carte(SDL_Renderer **renderer, SDL_Texture **texture_image_carte,
    SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_Rect *rectangle_options,
    SDL_Texture **texture_image_options, SDL_Rect *rectangle_perso, SDL_Texture **texture_image_perso,
    SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, position_t positionActive,
    SDL_Color couleurNoire, SDL_Rect *rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
    itemMenu *itemsNiveaux, int tailleNiveaux, int largeur, int hauteur, niveaux *avancee_niveaux)
00117  * \brief Fonction qui met à jour le rendu de la carte après redimension de la fenêtre
00118  * \param renderer Pointeur vers le renderer SDL.
00119  * \param texture_image_carte Texture de l'image de la carte.
00120  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00121  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00122  * \param rectangle_options Rectangle des options SDL.
00123  * \param texture_image_options Texture de l'image des options.
00124  * \param rectangle_perso Rectangle du personnage SDL.
00125  * \param texture_image_perso Texture de l'image du personnage.
00126  * \param surface Surface SDL.
00127  * \param texture_texte Texture du texte SDL.
00128  * \param police Police de caractères TTF.
00129  * \param positionActive Position active.
00130  * \param couleurNoire Couleur noire SDL.
00131  * \param rectangle_retour_menu Rectangle du bouton retour au menu SDL.
00132  * \param texture_image_retour_menu Texture de l'image du bouton retour au menu.
00133  * \param itemsNiveaux Tableau d'items pour les niveaux.
00134  * \param tailleNiveaux Taille du tableau d'items pour les niveaux.
00135  * \param largeur Largeur.
00136  * \param hauteur Hauteur.
00137  * \param avancee_niveaux Structure de progression des niveaux.
00138  * \see erreur
00139  * \see affichage_texte
00140 */
00141 void mise_a_jour_rendu_carte(SDL_Renderer **renderer, SDL_Texture **texture_image_carte,
00142     SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
00143     SDL_Rect *rectangle_options, SDL_Texture **texture_image_options,
00144     SDL_Rect *rectangle_perso, SDL_Texture **texture_image_perso,
00145     SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
    SDL_Texture **texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes,
    position_t positionActive, SDL_Color couleurNoire, SDL_Rect
00146     *rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
    itemMenu *itemsNiveaux, int tailleNiveaux, int largeur, int hauteur,
00147     niveaux *avancee_niveaux) {
00148     int i;
00149
00150     /* Efface le rendu */
00151     if(SDL_RenderClear((*renderer)) != 0)
00152         erreur("Effacement rendu échoué");
00153
00154     /* Utilisation de la fusion pour un rendu avec transparence */
00155     SDL_SetRenderDrawBlendMode((*renderer), SDL_BLENDMODE_BLEND);
00156
00157     /* Copie la texture de l'image de fond de la carte */
00158     if(SDL_RenderCopy((*renderer), (*texture_image_carte), NULL, NULL) != 0)
00159         erreur("Copie de la texture");
00160
00161     /* Copie la texture de l'image de retour au menu principal */
00162     rectangle_retour_menu->x = largeur / 53;
00163     rectangle_retour_menu->y = hauteur / 30;
00164     rectangle_retour_menu->w = largeur / 21;
00165     rectangle_retour_menu->h = hauteur / 12;
00166
00167     if(SDL_RenderCopy((*renderer), (*texture_image_retour_menu), NULL, rectangle_retour_menu) != 0)
00168         erreur("Copie de la texture");
00169
00170     /* Copie la texture de l'image de plein écran */
00171     rectangle_plein_ecran->x = largeur - largeur / 21 - largeur / 53;
00172     rectangle_plein_ecran->y = hauteur / 30;
00173     rectangle_plein_ecran->w = largeur / 21;
00174     rectangle_plein_ecran->h = hauteur / 12;
00175
00176     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00177         erreur("Copie de la texture");
00178
00179     /* Copie la texture de l'image des options */
00180     rectangle_options->x = largeur - largeur / 21 - largeur / 53;

```

```

00185     rectangle_options->y = hauteur - hauteur / 12 - hauteur / 30;
00186     rectangle_options->w = largeur / 21;
00187     rectangle_options->h = hauteur / 12;
00188
00189     if(SDL_RenderCopy((*renderer), (*texture_image_options), NULL, rectangle_options) != 0)
00190         erreur("Copie de la texture");
00191
00192     /* Copie la texture de l'image des succès */
00193
00194     if(avancee_niveaux[3].niveau_fini) {
00195
00196         rectangle_succes->x = largeur / 53;
00197         rectangle_succes->y = hauteur - hauteur / 12 - hauteur / 30;
00198         rectangle_succes->w = largeur / 21;
00199         rectangle_succes->h = hauteur / 12;
00200
00201         if(SDL_RenderCopy((*renderer), (*texture_image_fin_dernier_niveau), NULL, rectangle_succes) !=
00202 0)
00203             erreur("Copie de la texture");
00204     }
00205     else {
00206         rectangle_succes->x = 0;
00207         rectangle_succes->y = 0;
00208         rectangle_succes->w = 0;
00209         rectangle_succes->h = 0;
00210     }
00211
00212     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 150);
00213
00214     /* Dessine les éléments de la carte et dessine le personnage sur la carte */
00215
00216     for (i = 0; i < tailleNiveaux; i++) {
00217
00218         itemsNiveaux[i].rectangle.w = largeur / 7;
00219         itemsNiveaux[i].rectangle.h = hauteur / 17;
00220     }
00221
00222     if(positionActive == NIVEAU1) {
00223
00224         rectangle_perso->x = largeur / 4 + largeur / 75;
00225         rectangle_perso->y = hauteur / 2 + hauteur / 20 + hauteur / 17;
00226         rectangle_perso->w = largeur / 10;
00227         rectangle_perso->h = hauteur / 17;
00228
00229         SDL_RenderFillRect((*renderer), rectangle_perso);
00230
00231         rectangle_perso->w = largeur / 30;
00232
00233         /* Compte le nombre de collectible récupéré */
00234         for(i = 0; i < 3; i++)
00235             if(avancee_niveaux[0].numero_collectible[i]) {
00236
00237                 rectangle_perso->x = largeur / 4 + largeur / 75 + i * (rectangle_perso->w);
00238                 if(SDL_RenderCopy((*renderer), avancee_niveaux[0].texture_image_collectible, NULL,
rectangle_perso) != 0)
00239                     erreur("Copie de la texture");
00240             }
00241
00242         itemsNiveaux[0].rectangle.x = largeur / 4 + largeur / 75;
00243         itemsNiveaux[0].rectangle.y = hauteur / 2 + hauteur / 20;
00244
00245         rectangle_perso->x = largeur / 3 - largeur / 70;
00246         rectangle_perso->y = hauteur / 3 * 2 + hauteur / 50;
00247     }
00248
00249     else if(positionActive == NIVEAU2) {
00250
00251         rectangle_perso->x = largeur / 2 - largeur / 18;
00252         rectangle_perso->y = hauteur / 3 * 2 - hauteur / 30 + hauteur / 17;
00253         rectangle_perso->w = largeur / 10;
00254         rectangle_perso->h = hauteur / 17;
00255
00256         SDL_RenderFillRect((*renderer), rectangle_perso);
00257
00258         rectangle_perso->w = largeur / 30;
00259
00260         /* Compte le nombre de collectible récupéré */
00261         for(i = 0; i < 3; i++)
00262             if(avancee_niveaux[1].numero_collectible[i]) {
00263
00264                 rectangle_perso->x = largeur / 2 - largeur / 18 + i * (rectangle_perso->w);
00265                 if(SDL_RenderCopy((*renderer), avancee_niveaux[1].texture_image_collectible, NULL,
rectangle_perso) != 0)
00266                     erreur("Copie de la texture");
00267             }
00268

```

```

00269         itemsNiveaux[1].rectangle.x = largeur / 2 - largeur / 18;
00270         itemsNiveaux[1].rectangle.y = hauteur / 3 * 2 - hauteur / 30;
00271
00272         rectangle_perso->x = largeur / 2 - largeur / 500;
00273         rectangle_perso->y = hauteur / 2 + hauteur / 50;
00274     }
00275
00276     else if(positionActive == NIVEAU3) {
00277
00278         rectangle_perso->x = largeur / 2 - largeur / 20;
00279         rectangle_perso->y = hauteur / 3 - hauteur / 27 + hauteur / 17;
00280         rectangle_perso->w = largeur / 10;
00281         rectangle_perso->h = hauteur / 17;
00282
00283         SDL_RenderFillRect((*renderer), rectangle_perso);
00284
00285         rectangle_perso->w = largeur / 30;
00286
00287         /* Compte le nombre de collectible récupéré */
00288         for(i = 0; i < 3; i++)
00289             if(avancee_niveaux[2].numero_collectible[i]) {
00290
00291                 rectangle_perso->x = largeur / 2 - largeur / 20 + i * (rectangle_perso->w);
00292
00293                 if(SDL_RenderCopy((*renderer), avancee_niveaux[2].texture_image_collectible, NULL,
rectangle_perso) != 0)
00294                     erreur("Copie de la texture");
00295             }
00296
00297         itemsNiveaux[2].rectangle.x = largeur / 2 - largeur / 20;
00298         itemsNiveaux[2].rectangle.y = hauteur / 3 - hauteur / 27;
00299
00300         rectangle_perso->x = largeur / 2 + largeur / 400;
00301         rectangle_perso->y = hauteur / 2 - hauteur / 14;
00302     }
00303
00304     else if(positionActive == NIVEAU4){
00305
00306         rectangle_perso->x = largeur / 3 * 2 - largeur / 9;
00307         rectangle_perso->y = hauteur / 4 - hauteur / 18 + hauteur / 17;
00308         rectangle_perso->w = largeur / 10;
00309         rectangle_perso->h = hauteur / 17;
00310
00311         SDL_RenderFillRect((*renderer), rectangle_perso);
00312
00313         rectangle_perso->w = largeur / 30;
00314
00315         /* Compte le nombre de collectible récupéré */
00316         for(i = 0; i < 3; i++)
00317             if(avancee_niveaux[3].numero_collectible[i]) {
00318
00319                 rectangle_perso->x = largeur / 3 * 2 - largeur / 9 + i * (rectangle_perso->w);
00320                 if(SDL_RenderCopy((*renderer), avancee_niveaux[3].texture_image_collectible, NULL,
rectangle_perso) != 0)
00321                     erreur("Copie de la texture");
00322             }
00323
00324
00325         itemsNiveaux[3].rectangle.x = largeur / 3 * 2 - largeur / 9;
00326         itemsNiveaux[3].rectangle.y = hauteur / 4 - hauteur / 18;
00327
00328         rectangle_perso->x = largeur / 3 * 2 - largeur / 19;
00329         rectangle_perso->y = hauteur / 4 + hauteur / 13;
00330     }
00331
00332     /* Le personnage est sur un niveau */
00333     if(positionActive != NIVEAU0)
00334         affichage_texte(renderer, surface, texture_texte, &(itemsNiveaux[positionActive - 1]),
police, couleurNoire);
00335
00336     rectangle_perso->w = largeur / 30;
00337     rectangle_perso->h = hauteur / 13;
00338
00339     if(SDL_RenderCopy((*renderer), (*texture_image_perso), NULL, rectangle_perso) != 0)
00340         erreur("Copie de la texture");
00341
00342     /* Affiche le rendu */
00343     SDL_RenderPresent((*renderer));
00344 }
00345
00346 /**
00347 * \fn void deplacement_personnage_carte(SDL_Renderer **renderer, SDL_Window **window, SDL_Texture
**texture_image_carte, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
SDL_Rect *rectangle_options, SDL_Texture **texture_image_options, SDL_Rect *rectangle_perso,
SDL_Texture **texture_image_perso_1, SDL_Texture **texture_image_perso_2, SDL_Surface **surface,
SDL_Texture **texture_texte, TTF_Font **police, position_t *positionActive, SDL_Color couleurNoire,
SDL_Rect *rectangle_retour_menu, SDL_Texture **texture_image_retour_menu, itemMenu *itemsNiveaux, int

```



```

    tailleNiveaux, int largeur, int hauteur, int valeur_maximale, direction_t direction, niveaux
    *avancee_niveaux)
00349 * \brief Fonction qui permet de deplacer le personnage sur la carte
00350 * \param renderer Pointeur vers le renderer SDL.
00351 * \param window Pointeur vers la fenetre SDL.
00352 * \param texture_image_carte Texture de l'image de la carte.
00353 * \param rectangle_plein_ecran Rectangle plein écran SDL.
00354 * \param texture_image_plein_ecran Texture de l'image en plein écran.
00355 * \param rectangle_options Rectangle des options SDL.
00356 * \param texture_image_options Texture de l'image des options.
00357 * \param rectangle_perso Rectangle du personnage SDL.
00358 * \param texture_image_perso_1 Texture de l'image du personnage 1.
00359 * \param texture_image_perso_2 Texture de l'image du personnage 2.
00360 * \param surface Surface SDL.
00361 * \param texture_texte Texture du texte SDL.
00362 * \param police Police de caractères TTF.
00363 * \param positionActive Position active sur la carte.
00364 * \param couleurNoire Couleur noire SDL.
00365 * \param rectangle_retour_menu Rectangle du bouton retour au menu SDL.
00366 * \param texture_image_retour_menu Texture de l'image du bouton retour au menu.
00367 * \param itemsNiveaux Tableau d'items pour les niveaux.
00368 * \param tailleNiveaux Taille du tableau d'items pour les niveaux.
00369 * \param largeur Largeur de la carte.
00370 * \param hauteur Hauteur de la carte.
00371 * \param valeur_maximale Valeur maximale pour la direction.
00372 * \param direction Direction du déplacement.
00373 * \param avancee_niveaux Structure de progression des niveaux.
00374 * \see mise_a_jour_rendu_carte
00375 */
00376 void deplacement_personnage_carte(SDL_Renderer **renderer, SDL_Window **window, SDL_Texture
**texture_image_carte,
00377                                SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00378                                SDL_Rect *rectangle_options, SDL_Texture **texture_image_options,
00379                                SDL_Rect *rectangle_perso, SDL_Texture **texture_image_perso_1,
SDL_Texture **texture_image_perso_2,
00380                                SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font
**police, SDL_Texture **texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes,
00381                                position_t *positionActive, SDL_Color couleurNoire, SDL_Rect
*rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
00382                                itemMenu *itemsNiveaux, int tailleNiveaux, int largeur, int hauteur,
00383                                int valeur_maximale, direction_t direction, niveaux
*avancee_niveaux) {
00384     int i;
00385
00386     SDL_SetWindowResizable((*window), SDL_FALSE);
00387
00388     (*positionActive) = NIVEAU0;
00389
00390     /* Cas pour aller vers le haut ou vers le haut à gauche */
00391     if((direction == HAUT) || (direction == HAUT_DROITE)) {
00392         for(i = 0; rectangle_perso->y > valeur_maximale; i++) {
00393             if((direction == HAUT) || (direction == HAUT_DROITE))
00394                 rectangle_perso->y -= hauteur / 100;
00395             else if((direction == BAS) || (direction == BAS_GAUCHE))
00396                 rectangle_perso->y += hauteur / 100;
00397             if(direction == HAUT_DROITE)
00398                 rectangle_perso->x += largeur / 100;
00399             else if(direction == BAS_GAUCHE)
00400                 rectangle_perso->x -= largeur / 100;
00401
00402             if(i % 2)
00403                 mise_a_jour_rendu_carte(renderer, texture_image_carte,
rectangle_plein_ecran, texture_image_plein_ecran,
00411                 rectangle_options, texture_image_options,
rectangle_perso, texture_image_perso_1,
00413                 surface, texture_texte, police,
texture_image_fin_dernier_niveau, rectangle_succes,
00415                 (*positionActive), couleurNoire, rectangle_retour_menu,
texture_image_retour_menu,
00416                 itemsNiveaux, tailleNiveaux, largeur, hauteur,
avancee_niveaux);
00417             else
00418                 mise_a_jour_rendu_carte(renderer, texture_image_carte,
rectangle_plein_ecran, texture_image_plein_ecran,
00421                 rectangle_options, texture_image_options,
rectangle_perso, texture_image_perso_2,
00423                 surface, texture_texte, police,
texture_image_fin_dernier_niveau, rectangle_succes,

```

```

00424                                     (*positionActive), couleurNoire, rectangle_retour_menu,
texture_image_retour_menu,
00425                                     itemsNiveaux, tailleNiveaux, largeur, hauteur,
avancee_niveaux);
00426
00427         SDL_Delay(75);
00428     }
00429 }
00430
00431 /* Cas pour aller vers le bas ou vers le bas à gauche */
00432 else if((direction == BAS) || (direction == BAS_GAUCHE)) {
00433
00434     for(i = 0; rectangle_perso->y < valeur_maximale; i++) {
00435
00436         if((direction == HAUT) || (direction == HAUT_DROITE))
00437             rectangle_perso->y -= hauteur / 100;
00438
00439         else if((direction == BAS) || (direction == BAS_GAUCHE))
00440             rectangle_perso->y += hauteur / 100;
00441
00442         if(direction == HAUT_DROITE)
00443             rectangle_perso->x += largeur / 100;
00444
00445         else if(direction == BAS_GAUCHE)
00446             rectangle_perso->x -= largeur / 100;
00447
00448
00449         if(i % 2)
00450             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00451                                     rectangle_plein_ecran, texture_image_plein_ecran,
00452                                     rectangle_options, texture_image_options,
00453                                     rectangle_perso, texture_image_perso_1,
00454                                     surface, texture_texte, police,
texture_image_fin_dernier_niveau, rectangle_succes,
00455                                     (*positionActive), couleurNoire, rectangle_retour_menu,
texture_image_retour_menu,
00456                                     itemsNiveaux, tailleNiveaux, largeur, hauteur,
avancee_niveaux);
00457
00458         else
00459             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00460                                     rectangle_plein_ecran, texture_image_plein_ecran,
00461                                     rectangle_options, texture_image_options,
00462                                     rectangle_perso, texture_image_perso_2,
00463                                     surface, texture_texte, police,
texture_image_fin_dernier_niveau, rectangle_succes,
00464                                     (*positionActive), couleurNoire, rectangle_retour_menu,
texture_image_retour_menu,
00465                                     itemsNiveaux, tailleNiveaux, largeur, hauteur,
avancee_niveaux);
00466
00467         SDL_Delay(75);
00468     }
00469 }
00470
00471     SDL_SetWindowResizable((*window), SDL_TRUE);
00472 }
00473
00474 /**
00475  * \fn void carte(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance, SDL_Texture **texture_image_carte, SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran, SDL_bool *plein_ecran, SDL_Rect *rectangle_options, SDL_Texture
**texture_image_options, SDL_Rect *rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
SDL_Texture **texture_image_perso_bas_1, SDL_Texture **texture_image_perso_bas_2, SDL_Texture
**texture_image_perso_haut_1, SDL_Texture **texture_image_perso_haut_2, SDL_Texture
**texture_image_perso_bas_gauche_1, SDL_Texture **texture_image_perso_bas_gauche_2, SDL_Texture
**texture_image_perso_haut, SDL_Texture **texture_image_perso_droite, SDL_Texture
**texture_image_perso_gauche, SDL_Texture **texture_image_perso_pose, SDL_Texture
**texture_image_perso, SDL_Rect *rectangle_perso, niveaux *avancee_niveaux, int niveau_fini[4], int
collectibles[12], position_t *position_intermediaire, SDL_Surface **surface, SDL_Texture
**texture_texte, TTF_Font **police, direction_t *direction, int *touche_pressee, SDL_Rect
*rectangle_demande_sauvegarde, itemMenu *itemsDemandeSauvegarde, int tailleDemandeSauvegarde,
position_t *positionActive, barreDeSon *barre_de_son, itemMenu *pseudo, modes_t *modeActif,
personnage_t *personnageActif, SDL_Color couleurNoire, SDL_Keycode *touche_aller_a_droite, SDL_Keycode
*touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Keycode
*touche_interagir, itemMenu *itemsNiveaux, int tailleNiveaux, int *largeur, int *hauteur, page_t
*page_active)
00476  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent sur la carte
00477
00478
00479  * \param event Pointeur vers l'événement SDL.
00480  * \param window Pointeur vers la fenêtre SDL.
00481  * \param renderer Pointeur vers le renderer SDL.
00482  * \param programme_lance Booléen indiquant si le programme est en cours d'exécution.
00483  * \param texture_image_carte Texture de l'image de la carte.
00484  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00485  * \param texture_image_plein_ecran Texture de l'image en plein écran.

```

```

00486 * \param plein_ecran Booléen pour le plein écran.
00487 * \param rectangle_options Rectangle des options SDL.
00488 * \param texture_image_options Texture de l'image des options.
00489 * \param rectangle_retour_menu Rectangle du bouton retour au menu SDL.
00490 * \param texture_image_retour_menu Texture de l'image du bouton retour au menu.
00491 * \param texture_image_perso_bas_1 Texture de l'image du personnage en bas 1.
00492 * \param texture_image_perso_bas_2 Texture de l'image du personnage en bas 2.
00493 * \param texture_image_perso_haut_1 Texture de l'image du personnage en haut 1.
00494 * \param texture_image_perso_haut_2 Texture de l'image du personnage en haut 2.
00495 * \param texture_image_perso_bas_gauche_1 Texture de l'image du personnage en bas gauche 1.
00496 * \param texture_image_perso_bas_gauche_2 Texture de l'image du personnage en bas gauche 2.
00497 * \param texture_image_perso_haut Texture de l'image du personnage en haut.
00498 * \param texture_image_perso_droite Texture de l'image du personnage à droite.
00499 * \param texture_image_perso_gauche Texture de l'image du personnage à gauche.
00500 * \param texture_image_perso_pose Texture de l'image du personnage en pose.
00501 * \param texture_image_perso Texture de l'image du personnage.
00502 * \param rectangle_perso Rectangle du personnage SDL.
00503 * \param avancee_niveaux Structure de progression des niveaux.
00504 * \param niveau_fini Tableau indiquant si chaque niveau est terminé.
00505 * \param collectibles Tableau des collectibles.
00506 * \param position_intermediaire Position intermédiaire du personnage.
00507 * \param surface Surface SDL.
00508 * \param texture_texte Texture du texte SDL.
00509 * \param police Police de caractères TTF.
00510 * \param direction Direction du personnage.
00511 * \param touche_pressee Indicateur de touche pressée.
00512 * \param rectangle_demande_sauvegarde Rectangle de la demande de sauvegarde SDL.
00513 * \param itemsDemandeSauvegarde Tableau d'items pour la demande de sauvegarde.
00514 * \param tailleDemandeSauvegarde Taille du tableau d'items pour la demande de sauvegarde.
00515 * \param positionActive Position active.
00516 * \param barre_de_son Barre de son SDL.
00517 * \param pseudo Pseudo du joueur SDL.
00518 * \param modeActif Mode actif du jeu.
00519 * \param personnageActif Personnage actif.
00520 * \param couleurNoire Couleur noire SDL.
00521 * \param touche_aller_a_droite Touche pour aller à droite.
00522 * \param touche_aller_a_gauche Touche pour aller à gauche.
00523 * \param touche_sauter_monter Touche pour sauter/monter.
00524 * \param touche_descendre Touche pour descendre.
00525 * \param touche_interagir Touche pour interagir.
00526 * \param itemsNiveaux Tableau d'items pour les niveaux.
00527 * \param tailleNiveaux Taille du tableau d'items pour les niveaux.
00528 * \param largeur Largeur de la carte.
00529 * \param hauteur Hauteur de la carte.
00530 * \param page_active Page active.
00531 * \see mise_a_jour_rendu_carte
00532 * \see redimensionnement_fenetre
00533 * \see deplacement_personnage_carte
00534 * \see erreur
00535 * \see demande_sauvegarde
00536 * \see sauvegarder_partie
00537 * \see clic_case
00538 */
00539 void carte(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool *programme_lance,
SDL_Texture **texture_image_carte,
00540 SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
*plein_ecran,
00541 SDL_Rect *rectangle_options, SDL_Texture **texture_image_options, SDL_Rect
*rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
00542 SDL_Texture **texture_image_perso_bas_1, SDL_Texture **texture_image_perso_bas_2,
00543 SDL_Texture **texture_image_perso_haut_1, SDL_Texture **texture_image_perso_haut_2,
00544 SDL_Texture **texture_image_perso_bas_gauche_1, SDL_Texture
**texture_image_perso_bas_gauche_2,
00545 SDL_Texture **texture_image_perso_haut, SDL_Texture **texture_image_perso_droite,
00546 SDL_Texture **texture_image_perso_gauche, SDL_Texture **texture_image_perso_pose,
00547 SDL_Texture **texture_image_perso, SDL_Rect *rectangle_perso, niveaux *avancee_niveaux,
00548 int niveau_fini[4], int collectibles[12], position_t *position_intermediaire, SDL_Texture
**texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes,
00549 SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, direction_t
*direction, int *touche_pressee,
00550 SDL_Rect *rectangle_demande_sauvegarde, itemMenu *itemsDemandeSauvegarde, int
tailleDemandeSauvegarde,
00551 position_t *positionActive, barreDeSon *barre_de_son, itemMenu *pseudo, modes_t *modeActif,
personnage_t *personnageActif,
00552 SDL_Color couleurNoire, SDL_Keycode *touche_aller_a_droite, SDL_Keycode
*touche_aller_a_gauche,
00553 SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Keycode
*touche_interagir,
00554 itemMenu *itemsNiveaux, int tailleNiveaux, int *largeur, int *hauteur, page_t *page_active,
00555 itemMenu *itemsSucces, SDL_Texture **textures_images_succes,
00556 time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
avancee_succes_intermediaires[10]) {
00557
00558     SDL_Event event_temporaire;
00559     SDL_bool clic_effectue = SDL_FALSE;
00560
00561     Mix_Chunk *effet_sonore = NULL;

```

```

00562
00563     int i, j;
00564
00565     for(i = 0; i < 4; i++)
00566         for(j = 0; j < 3; j++)
00567
00568         if(avancee_niveaux[i].numero_collectible[j]) {
00569
00570             avancee_succes[1] = 1;
00571         }
00572
00573         else {
00574
00575             avancee_succes[1] = 0;
00576             j = 3;
00577             i = 4;
00578         }
00579
00580     if((avancee_succes[0]) && (avancee_succes[1]) &&
00581        (avancee_succes[2]) && (avancee_succes[3]) &&
00582        (avancee_succes[4]) && (avancee_succes[5]) &&
00583        (avancee_succes[6]) && (avancee_succes[8]))
00584        avancee_succes[9] = 1;
00585
00586     /* Mise à jour du rendu */
00587     switch((*direction)) {
00588
00589         case HAUT:
00590             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00591                                     rectangle_plein_ecran, texture_image_plein_ecran,
00592                                     rectangle_options, texture_image_options,
00593                                     rectangle_perso, texture_image_perso_haut,
00594                                     surface, texture_texte, police, texture_image_fin_dernier_niveau,
00595                                     (*positionActive), couleurNoire, rectangle_retour_menu,
00596                                     itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
00597                                     avancee_niveaux);
00598             break;
00599
00600         case BAS:
00601             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00602                                     rectangle_plein_ecran, texture_image_plein_ecran,
00603                                     rectangle_options, texture_image_options,
00604                                     rectangle_perso, texture_image_perso,
00605                                     surface, texture_texte, police, texture_image_fin_dernier_niveau,
00606                                     (*positionActive), couleurNoire, rectangle_retour_menu,
00607                                     itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
00608                                     avancee_niveaux);
00609             break;
00610
00611         case DROITE:
00612             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00613                                     rectangle_plein_ecran, texture_image_plein_ecran,
00614                                     rectangle_options, texture_image_options,
00615                                     rectangle_perso, texture_image_perso_droite,
00616                                     surface, texture_texte, police, texture_image_fin_dernier_niveau,
00617                                     (*positionActive), couleurNoire, rectangle_retour_menu,
00618                                     itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
00619                                     avancee_niveaux);
00620             break;
00621
00622         default:
00623             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00624                                     rectangle_plein_ecran, texture_image_plein_ecran,
00625                                     rectangle_options, texture_image_options,
00626                                     rectangle_perso, texture_image_perso_gauche,
00627                                     surface, texture_texte, police, texture_image_fin_dernier_niveau,
00628                                     (*positionActive), couleurNoire, rectangle_retour_menu,
00629                                     itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
00630                                     avancee_niveaux);
00631     }
00632     while(SDL_PollEvent(event)) {
00633         switch(event->type) {
00634
00635             /* Gestion de l'événement de redimensionnement de la fenêtre */
00636

```

```

00637         case SDL_WINDOWEVENT:
00638             redimensionnement_fenetre((event), largeur, hauteur);
00639
00640             break;
00641
00642             /* Lecture de la touche pressée et de la position du personnage pour savoir où aller
*/
00643         case SDL_KEYDOWN :
00644
00645             /* Aller du niveau 1 au niveau 2 */
00646             if ((*positionActive == NIVEAU1) && (!(*touche_pressee)) &&
(avancee_niveaux[0].niveau_fini) &&
00647             ((event->key.keysym.sym == (*touche_aller_a_droite)) || (event->key.keysym.sym
== (*touche_sauter_monter)))) {
00648
00649                 (*touche_pressee) = 1;
00650
00651                 deplacement_personnage_carte(renderer, window, texture_image_carte,
rectangle_plein_ecran, texture_image_plein_ecran,
00652                 rectangle_options, texture_image_options,
rectangle_perso, texture_image_perso_haut_1,
00653                 texture_image_perso_haut_2,
surface, texture_texte, police,
00654                 texture_image_fin_dernier_niveau, rectangle_succes,
positionActive, couleurNoire,
00655                 rectangle_retour_menu, texture_image_retour_menu,
itemsNiveaux, tailleNiveaux, (*largeur),
00656                 (*hauteur),
(((*hauteur) / 2 + (*hauteur) / 50) + (*hauteur)
00657                 / 75), HAUT_DROITE, avancee_niveaux);
00658
00659                 (*positionActive) = NIVEAU2;
00660                 (*direction) = BAS;
00661
00662             }
00663
00664             /* Entrer dans le niveau 1 */
00665             else if ((*positionActive == NIVEAU1) && (!(*touche_pressee)) &&
(event->key.keysym.sym == (*touche_interagir))) {
00666
00667                 /* Effet sonore quand on rentre dans un niveau */
00668                 if (effet_sonore = Mix_LoadWAV("./sons/effets_sonores/rentree_niveaux.wav"))
00669                 == NULL)
00670                     erreur("Chargement de l'effet sonore");
00671
00672                 Mix_PlayChannel(1, effet_sonore, 0);
00673
00674                 (*touche_pressee) = 1;
00675
00676                 mise_a_jour_rendu_carte(renderer, texture_image_carte,
rectangle_plein_ecran, texture_image_plein_ecran,
00677                 rectangle_options, texture_image_options,
rectangle_perso, texture_image_perso_pose,
00678                 surface, texture_texte, police,
texture_image_fin_dernier_niveau, rectangle_succes,
00679                 (*positionActive), couleurNoire,
rectangle_retour_menu, texture_image_retour_menu,
00680                 itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
avancee_niveaux);
00681
00682                 SDL_Delay(1000);
00683
00684                 (*page_active) = NIVEAU1;
00685             }
00686
00687             /* Aller du niveau 2 au niveau 3 */
00688             else if ((*positionActive == NIVEAU2) && (!(*touche_pressee)) &&
(avancee_niveaux[1].niveau_fini) &&
00689             (event->key.keysym.sym == (*touche_sauter_monter)))) {
00690
00691                 (*touche_pressee) = 1;
00692
00693                 deplacement_personnage_carte(renderer, window, texture_image_carte,
rectangle_plein_ecran, texture_image_plein_ecran,
00694                 rectangle_options, texture_image_options,
rectangle_perso, texture_image_perso_haut_1,
00695                 texture_image_perso_haut_2,
surface, texture_texte, police,
00696                 texture_image_fin_dernier_niveau, rectangle_succes,
positionActive, couleurNoire,
00697                 rectangle_retour_menu, texture_image_retour_menu,
itemsNiveaux, tailleNiveaux, (*largeur),
00698                 (*hauteur),
(((*hauteur) / 2 - (*hauteur) / 14) + (*hauteur)
00699                 / 75), HAUT, avancee_niveaux);
00700
00701                 (*positionActive) = NIVEAU3;

```

```

00706             (*direction) = BAS;
00707         }
00708
00709         /* Aller du niveau 2 au niveau 1 */
00710         else if ((*positionActive == NIVEAU2) && (!(*touche_pressee)) &&
00711             ((event->key.keysym.sym == (*touche_aller_a_gauche)) ||
00712             (event->key.keysym.sym == (*touche_descendre)))) {
00713             (*touche_pressee) = 1;
00714
00715             deplacement_personnage_carte(renderer, window, texture_image_carte,
00716                 rectangle_plein_ecran, texture_image_plein_ecran,
00717                 rectangle_options, texture_image_options,
00718                 rectangle_perso,
00719                 texture_image_perso_bas_gauche_1, texture_image_perso_bas_gauche_2,
00720                 surface, texture_texte, police,
00721                 texture_image_fin_dernier_niveau, rectangle_succes,
00722                 positionActive, couleurNoire,
00723                 rectangle_retour_menu, texture_image_retour_menu,
00724                 itemsNiveaux, tailleNiveaux, (*largeur),
00725                 (*hauteur),
00726                 (((*hauteur) / 3 * 2 + (*hauteur) / 50) -
00727                 (*largeur) / 75), BAS_GAUCHE, avancee_niveaux);
00728
00729             (*positionActive) = NIVEAU1;
00730             (*direction) = BAS;
00731         }
00732
00733         /* Entrer dans le niveau 2 */
00734         else if ((*positionActive == NIVEAU2) && (!(*touche_pressee)) &&
00735             (event->key.keysym.sym == (*touche_interagir))) {
00736
00737             /* Effet sonore quand on rentre dans un niveau */
00738             if ((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/rentree_niveaux.wav"))
00739 == NULL)
00740                 erreur("Chargement de l'effet sonore");
00741
00742             Mix_PlayChannel(1, effet_sonore, 0);
00743
00744             (*touche_pressee) = 1;
00745
00746             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00747                 rectangle_plein_ecran, texture_image_plein_ecran,
00748                 rectangle_options, texture_image_options,
00749                 rectangle_perso, texture_image_perso_pose,
00750                 surface, texture_texte, police,
00751                 texture_image_fin_dernier_niveau, rectangle_succes,
00752                 (*positionActive), couleurNoire,
00753                 rectangle_retour_menu, texture_image_retour_menu,
00754                 itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
00755                 avancee_niveaux);
00756
00757             SDL_Delay(1000);
00758
00759             (*page_active) = NIVEAU_2;
00760         }
00761
00762         /* Aller du niveau 3 au niveau 4 */
00763         else if ((*positionActive == NIVEAU3) && (!(*touche_pressee)) &&
00764             (avancee_niveaux[2].niveau_fini) &&
00765             ((event->key.keysym.sym == (*touche_aller_a_droite)) ||
00766             (event->key.keysym.sym == (*touche_sauter_monter)))) {
00767             (*touche_pressee) = 1;
00768
00769             deplacement_personnage_carte(renderer, window, texture_image_carte,
00770                 rectangle_plein_ecran, texture_image_plein_ecran,
00771                 rectangle_options, texture_image_options,
00772                 rectangle_perso, texture_image_perso_haut_1,
00773                 texture_image_perso_haut_2,
00774                 surface, texture_texte, police,
00775                 texture_image_fin_dernier_niveau, rectangle_succes,
00776                 positionActive, couleurNoire,
00777                 rectangle_retour_menu, texture_image_retour_menu,
00778                 itemsNiveaux, tailleNiveaux, (*largeur),
00779                 (*hauteur),
00780                 (((*hauteur) / 4 + (*hauteur) / 13) + (*hauteur)
00781                 / 75), HAUT_DROITE, avancee_niveaux);
00782
00783             (*positionActive) = NIVEAU4;
00784             (*direction) = BAS;
00785         }
00786
00787         /* Aller du niveau 3 au niveau 2 */
00788         else if ((*positionActive == NIVEAU3) && (!(*touche_pressee)) &&
00789             (event->key.keysym.sym == (*touche_descendre))) {

```

```

00776             (*touche_pressee) = 1;
00777
00778             deplacement_personnage_carte(renderer, window, texture_image_carte,
00779                                         rectangle_plein_ecran, texture_image_plein_ecran,
00780                                         rectangle_options, texture_image_options,
00781                                         rectangle_perso, texture_image_perso_bas_1,
00782                                         texture_image_perso_bas_2,
00783                                         surface, texture_texte, police,
00784                                         texture_image_fin_dernier_niveau, rectangle_succes,
00785                                         positionActive, couleurNoire,
00786                                         rectangle_retour_menu, texture_image_retour_menu,
00787                                         itemsNiveaux, tailleNiveaux, (*largeur),
00788                                         (*hauteur),
00789                                         (((*hauteur) / 2 + (*hauteur) / 50) - (*hauteur)
00790                                         / 75), BAS, avancee_niveaux);
00791
00792             (*positionActive) = NIVEAU2;
00793             (*direction) = BAS;
00794         }
00795
00796         /* Entrer dans le niveau 3 */
00797         else if ((*positionActive == NIVEAU3) && (!(*touche_pressee)) &&
00798                 (event->key.keysym.sym == (*touche_interagir))) {
00799
00800             /* Effet sonore quand on rentre dans un niveau */
00801             if ((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/rentree_niveaux.wav"))
00802                 == NULL)
00803                 erreur("Chargement de l'effet sonore");
00804
00805             Mix_PlayChannel(1, effet_sonore, 0);
00806
00807             (*touche_pressee) = 1;
00808
00809             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00810                                     rectangle_plein_ecran, texture_image_plein_ecran,
00811                                     rectangle_options, texture_image_options,
00812                                     rectangle_perso, texture_image_perso_pose,
00813                                     surface, texture_texte, police,
00814                                     texture_image_fin_dernier_niveau, rectangle_succes,
00815                                     (*positionActive), couleurNoire,
00816                                     rectangle_retour_menu, texture_image_retour_menu,
00817                                     itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
00818                                     avancee_niveaux);
00819
00820             SDL_Delay(1000);
00821
00822             (*page_active) = NIVEAU3;
00823         }
00824
00825         /* Aller du niveau 4 au niveau 3 */
00826         else if ((*positionActive == NIVEAU4) && (!(*touche_pressee)) &&
00827                 ((event->key.keysym.sym == (*touche_aller_a_gauche)) || (event->key.keysym.sym
00828                 == (*touche_descendre)))) {
00829
00830             (*touche_pressee) = 1;
00831
00832             deplacement_personnage_carte(renderer, window, texture_image_carte,
00833                                         rectangle_plein_ecran, texture_image_plein_ecran,
00834                                         rectangle_options, texture_image_options,
00835                                         rectangle_perso,
00836                                         texture_image_perso_bas_gauche_1, texture_image_perso_bas_gauche_2,
00837                                         surface, texture_texte, police,
00838                                         texture_image_fin_dernier_niveau, rectangle_succes,
00839                                         positionActive, couleurNoire,
00840                                         rectangle_retour_menu, texture_image_retour_menu,
00841                                         itemsNiveaux, tailleNiveaux, (*largeur),
00842                                         (*hauteur),
00843                                         (((*hauteur) / 2 - (*hauteur) / 14) - (*largeur)
00844                                         / 75), BAS_GAUCHE, avancee_niveaux);
00845
00846             (*positionActive) = NIVEAU3;
00847             (*direction) = BAS;
00848         }
00849
00850         /* Entrer dans le niveau 4 */
00851         else if ((*positionActive == NIVEAU4) && (!(*touche_pressee)) &&
00852                 (event->key.keysym.sym == (*touche_interagir))) {
00853
00854             /* Effet sonore quand on rentre dans un niveau */
00855             if ((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/rentree_niveau_4.wav"))
00856                 == NULL)
00857                 erreur("Chargement de l'effet sonore");
00858
00859             Mix_PlayChannel(1, effet_sonore, 0);
00860
00861             (*touche_pressee) = 1;

```

```

00847             mise_a_jour_rendu_carte(renderer, texture_image_carte,
00848                                     rectangle_plein_ecran, texture_image_plein_ecran,
00849                                     rectangle_options, texture_image_options,
00850                                     rectangle_perso, texture_image_perso_pose,
00851                                     surface, texture_texte, police,
texture_image_fin_dernier_niveau, rectangle_succes,
00852                                     (*positionActive), couleurNoire,
rectangle_retour_menu, texture_image_retour_menu,
00853                                     itemsNiveaux, tailleNiveaux, (*largeur), (*hauteur),
avancee_niveaux);
00854
00855             SDL_Delay(1000);
00856
00857             (*page_active) = NIVEAU_4;
00858         }
00859
00860         /* Pivoter vers le haut si on ne peut pas monter */
00861         else if(((((*positionActive) == NIVEAU1) && (!avancee_niveaux[0].niveau_fini)) ||
00862                 (((*positionActive) == NIVEAU2) && (!avancee_niveaux[1].niveau_fini)) ||
00863                 (((*positionActive) == NIVEAU3) && (!avancee_niveaux[2].niveau_fini)) ||
00864                 (((*positionActive) == NIVEAU4) && (!(*touche_pressee))) &&
00865                 (event->key.keysym.sym == (*touche_sauter_monter)))) {
00866
00867             /* Effet sonore quand on ne peut pas aller dans une direction */
00868             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collision_mur.wav")) ==
NULL)
00869                 erreur("Chargement de l'effet sonore");
00870
00871             Mix_PlayChannel(1, effet_sonore, 0);
00872
00873             (*touche_pressee) = 1;
00874
00875             (*direction) = HAUT;
00876         }
00877
00878         /* Pivoter vers le bas si on ne peut pas descendre */
00879         else if(((((*positionActive) == NIVEAU1) && (!(*touche_pressee))) &&
00880                 (event->key.keysym.sym == (*touche_descendre)))) {
00881
00882             /* Effet sonore quand on ne peut pas aller dans une direction */
00883             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collision_mur.wav")) ==
NULL)
00884                 erreur("Chargement de l'effet sonore");
00885
00886             Mix_PlayChannel(1, effet_sonore, 0);
00887
00888             (*touche_pressee) = 1;
00889
00890             (*direction) = BAS;
00891         }
00892
00893         /* Pivoter vers la droite si on ne peut pas aller vers la droite */
00894         else if(((((*positionActive) == NIVEAU2) || ((*positionActive) == NIVEAU4)) ||
00895                 (((*positionActive) == NIVEAU1) && (!avancee_niveaux[0].niveau_fini)) ||
00896                 (((*positionActive) == NIVEAU3) && (!avancee_niveaux[2].niveau_fini))) &&
00897                 (!(*touche_pressee)) && (event->key.keysym.sym ==
(*touche_aller_a_droite))) {
00898
00899             /* Effet sonore quand on ne peut pas aller dans une direction */
00900             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collision_mur.wav")) ==
NULL)
00901                 erreur("Chargement de l'effet sonore");
00902
00903             Mix_PlayChannel(1, effet_sonore, 0);
00904
00905             (*touche_pressee) = 1;
00906
00907             (*direction) = DROITE;
00908         }
00909
00910         /* Pivoter vers la gauche si on ne peut pas aller vers la gauche */
00911         else if(((((*positionActive) == NIVEAU1) || ((*positionActive) == NIVEAU3))) &&
00912                 (!(*touche_pressee)) && (event->key.keysym.sym ==
(*touche_aller_a_gauche))) {
00913
00914             /* Effet sonore quand on ne peut pas aller dans une direction */
00915             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collision_mur.wav")) ==
NULL)
00916                 erreur("Chargement de l'effet sonore");
00917
00918             Mix_PlayChannel(1, effet_sonore, 0);
00919
00920             (*touche_pressee) = 1;
00921
00922             (*direction) = GAUCHE;
00923         }
00924

```



```

00925         break;
00926
00927     /* Options plein écran, options, retour au menu principal et succès */
00928     case SDL_MOUSEBUTTONDOWN:
00929
00930         if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window))
00931             redimensionnement_fenetre((*event), largeur, hauteur);
00932
00933         if(clic_case((*event), (*rectangle_options)))
00934             (*page_active) = OPTIONS;
00935
00936         if(clic_case((*event), (*rectangle_retour_menu))) {
00937             SDL_SetWindowResizable((*window), SDL_FALSE);
00938
00939             demande_sauvegarde(renderer, rectangle_demande_sauvegarde,
00940                               surface, texture_texte, police, couleurNoire,
00941                               itemsDemandeSauvegarde, tailleDemandeSauvegarde,
00942                               (*largeur), (*hauteur));
00943
00944             while (!clic_effectue) {
00945                 while (SDL_PollEvent(&event_temporaire)) {
00946                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00947                         if(clic_case(event_temporaire,
00948                                     itemsDemandeSauvegarde[1].rectangle)) {
00949                             sauvegarder_partie(touche_aller_a_droite,
00950                                                 touche_aller_a_gauche, touche_sauter_monter,
00951                                                 touche_descendre, touche_interagir,
00952                                                 barre_de_son, pseudo,
00953                                                 (*modeActif), (*personnageActif),
00954                                                 (*positionActive),
00955                                                 avancee_niveaux, tailleNiveaux,
00956                                                 temps_debut_partie, (*compteur_mort), avancee_succes);
00957                             (*page_active) = MENU_PRINCIPAL;
00958                             clic_effectue = SDL_TRUE;
00959                         }
00960                         else if(clic_case(event_temporaire,
00961                                     itemsDemandeSauvegarde[2].rectangle)) {
00962                             for(i = 0; i < 4; i++) {
00963                                 avancee_niveaux[i].niveau_fini = niveau_fini[i];
00964                                 for(j = 0; j < 3; j++)
00965                                     avancee_niveaux[i].numero_collectible[j] =
00966                                         collectibles[i + j];
00967                                 (*positionActive) = (*position_intermediaire);
00968                             }
00969                             for(i = 0; i < 10; i++)
00970                                 avancee_succes[i] = avancee_succes_intermediaires[i];
00971                             (*page_active) = MENU_PRINCIPAL;
00972                             clic_effectue = SDL_TRUE;
00973                         }
00974                         else if(!clic_case(event_temporaire,
00975                                     (*rectangle_demande_sauvegarde)))
00976                             clic_effectue = SDL_TRUE;
00977                     }
00978                 }
00979             }
00980             SDL_SetWindowResizable((*window), SDL_TRUE);
00981         }
00982     }
00983 }
00984
00985 /* Page des succès */
00986 if(clic_case((*event), (*rectangle_succes))) {
00987     SDL_SetWindowResizable((*window), SDL_FALSE);
00988     while (!clic_effectue) {
00989         SDL_Event event_temporaire_bis;
00990         SDL_bool clic_effectue_bis = SDL_FALSE;
00991
00992         SDL_SetRenderDrawColor((*renderer), 240, 240, 240, 255);
00993
00994         /* Efface le rendu */
00995         if(SDL_RenderClear((*renderer)) != 0)

```

```

01003         erreur("Effacement rendu échoué");
01004
01005         /* Initialise le rectangle pour les images des succès */
01006
01007         rectangle_succes->x = (*largeur) / 50;
01008         rectangle_succes->w = (*largeur) / 25;
01009         rectangle_succes->h = (*hauteur) / 14;
01010
01011         /* Affichage des différentes images */
01012         for(i = 0; i < 10; i++) {
01013
01014             rectangle_succes->y = (*hauteur) / 14 + i * ((*hauteur) / 70 +
01015 (*hauteur) / 14);
01016
01017             if(avancee_succes[i]) {
01018                 if(SDL_RenderCopy((*render), textures_images_succes[i + 1],
01019 NULL, rectangle_succes) != 0)
01020                     erreur("Copie de la texture");
01021             }
01022             else {
01023                 if(SDL_RenderCopy((*render), textures_images_succes[0], NULL,
01024 rectangle_succes) != 0)
01025                     erreur("Copie de la texture");
01026             }
01027         }
01028
01029         /* Affiche le titre de la page */
01030
01031         itemsSucces[0].rectangle.x = (*largeur) / 8 * 3;
01032         itemsSucces[0].rectangle.y = (*hauteur) / 11 - (*hauteur) / 12;
01033         itemsSucces[0].rectangle.w = (*largeur) / 4;
01034         itemsSucces[0].rectangle.h = (*hauteur) / 15;
01035
01036         affichage_texte(render, surface, texture_texte, &(itemsSucces[0]),
01037             police, couleurNoire);
01038
01039         /* Affichage les différentes phrases pour chaque succès */
01040         for(i = 0; i < 10; i++) {
01041
01042             itemsSucces[i + 1].rectangle.x = rectangle_succes->x * 2 +
01043 rectangle_succes->w;
01044             itemsSucces[i + 1].rectangle.y = (*hauteur) / 14 + i * ((*hauteur) /
01045 70 + (*hauteur) / 14);
01046             itemsSucces[i + 1].rectangle.w = (*largeur) - itemsSucces[i +
01047 1].rectangle.x - rectangle_succes->x;
01048             itemsSucces[i + 1].rectangle.h = (*hauteur) / 14;
01049
01050             affichage_texte(render, surface, texture_texte, &(itemsSucces[i +
01051 1]),
01052                 police, couleurNoire);
01053         }
01054
01055         /* Initialise le rectangle pour le bouton de sorti de la page des succès
01056 */
01057
01058         itemsSucces[11].rectangle.x = (*largeur) / 3 * 2;
01059         itemsSucces[11].rectangle.y = (*hauteur) - (*hauteur) / 11;
01060         itemsSucces[11].rectangle.w = (*largeur) / 7;
01061         itemsSucces[11].rectangle.h = (*hauteur) / 12;
01062
01063         SDL_SetRenderDrawColor((*render), 0, 0, 0, 255);
01064         SDL_RenderDrawRect((*render), &(itemsSucces[11].rectangle));
01065
01066         SDL_SetRenderDrawColor((*render), 255, 255, 255, 0);
01067
01068         affichage_texte(render, surface, texture_texte, &(itemsSucces[11]),
01069             police, couleurNoire);
01070
01071         /* Affiche le rendu */
01072         SDL_RenderPresent((*render));
01073
01074         while (SDL_PollEvent(&event_temporaire))
01075
01076             /* Cas où on quitte la page des succès */
01077             if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
01078                 if(clic_case(event_temporaire, itemsSucces[11].rectangle))
01079                     clic_effectue = SDL_TRUE;
01080             }
01081
01082             /* Quitter le programme en demandant s'il faut sauvegarder la partie */
01083             else if(event_temporaire.type == SDL_QUIT) {
01084                 demande_sauvegarde(render, rectangle_demande_sauvegarde,

```

```

01082                                     surface, texture_texte, police, couleurNoire,
01083                                     itemsDemandeSauvegarde,
01084     tailleDemandeSauvegarde, (*largeur), (*hauteur));
01085                                     while (!clic_effectue_bis) {
01086                                     while (SDL_PollEvent(&event_temporaire_bis)) {
01087                                         if(event_temporaire_bis.type == SDL_MOUSEBUTTONDOWN) {
01088                                             if(clic_case(event_temporaire_bis,
01089 itemsDemandeSauvegarde[1].rectangle)) {
01090
01091                                     sauvegarder_partie(touche_aller_a_droite,
01092 touche_aller_a_gauche, touche_sauter_monter,
01093                                     touche_descendre,
01094 touche_interagir, barre_de_son, pseudo,
01095                                     (*modeActif), (*personnageActif),
01096 avancee_niveaux, tailleNiveaux,
01097 temps_debut_partie, (*compteur_mort), avancee_succes);
01098                                     (*programme_lance) = SDL_FALSE;
01099                                     clic_effectue_bis = SDL_TRUE;
01100                                     clic_effectue = SDL_TRUE;
01101                                     }
01102                                     else if(clic_case(event_temporaire_bis,
01103 itemsDemandeSauvegarde[2].rectangle)) {
01104                                     (*programme_lance) = SDL_FALSE;
01105                                     clic_effectue_bis = SDL_TRUE;
01106                                     clic_effectue = SDL_TRUE;
01107                                     }
01108                                     else if(!clic_case(event_temporaire_bis,
01109 (*rectangle_demande_sauvegarde)))
01110                                     clic_effectue_bis = SDL_TRUE;
01111                                     }
01112                                     }
01113                                     }
01114                                     }
01115                                     }
01116                                     SDL_SetWindowResizable((*window), SDL_TRUE);
01117                                     }
01118                                     }
01119                                     break;
01120                                     }
01121                                     /* Quitter le programme en demandant s'il faut sauvegarder la partie */
01122                                     case SDL_QUIT:
01123                                     SDL_SetWindowResizable((*window), SDL_FALSE);
01124                                     demande_sauvegarde(renderer, rectangle_demande_sauvegarde,
01125                                     surface, texture_texte, police, couleurNoire,
01126                                     itemsDemandeSauvegarde, tailleDemandeSauvegarde, (*largeur),
01127                                     (*hauteur));
01128                                     }
01129                                     while (!clic_effectue) {
01130                                     while (SDL_PollEvent(&event_temporaire)) {
01131                                         if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
01132                                             if(clic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {
01133                                                 sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
01134 touche_sauter_monter,
01135                                     touche_descendre, touche_interagir,
01136 barre_de_son, pseudo,
01137                                     (*modeActif), (*personnageActif),
01138 avancee_niveaux, tailleNiveaux,
01139 temps_debut_partie, (*compteur_mort), avancee_succes);
01140                                     (*programme_lance) = SDL_FALSE;
01141                                     clic_effectue = SDL_TRUE;
01142                                     }
01143                                     else if(clic_case(event_temporaire,
01144 itemsDemandeSauvegarde[2].rectangle)) {
01145                                     (*programme_lance) = SDL_FALSE;
01146                                     clic_effectue = SDL_TRUE;
01147                                     }
01148                                     else if(!clic_case(event_temporaire, (*rectangle_demande_sauvegarde)))
01149                                     clic_effectue = SDL_TRUE;
01150                                     }
01151                                     }
01152                                     }
01153                                     }
01154                                     }

```

```

01155         }
01156     }
01157 }
01158
01159     SDL_SetWindowResizable((*window), SDL_TRUE);
01160
01161     break;
01162
01163     default:
01164         break;
01165 }
01166 }
01167
01168 }

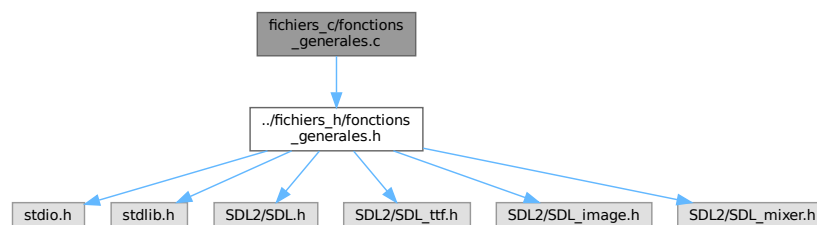
```

## 4.5 Référence du fichier fichiers\_c/fonctions\_generales.c

Fichier avec les fichiers utilisé régulièrement.

```
#include <../fichiers_h/fonctions_generales.h>
```

Graphe des dépendances par inclusion de fonctions\_generales.c:



## Fonctions

- void [erreur](#) (const char \*message)  
*Affiche l'erreur en cas de problème et ferme la SDL.*
- void [chargement\\_image](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture, char \*chemin)  
*Fonction qui permet de charger une image.*
- void [affichage\\_texte](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture, [itemMenu](#) \*item, TTF\_Font \*\*police, SDL\_Color couleur)
- void [creer\\_fenetre\\_rendu](#) (SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, int largeur, int hauteur)  
*Fonction qui permet de créer une fenêtre et le rendu.*
- void [initialisation\\_objets](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture ↵ image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image ↵ options, SDL\_Texture \*\*texture\_image\_passer, [itemMenu](#) \*itemsDemandeSauvegarde, [itemMenu](#) \*items ↵ DemandeQuitter, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, SDL\_Texture \*\*texture\_image ↵ \_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image ↵ perso\_1\_gagnant, SDL\_Texture \*\*texture\_image\_perso\_2\_gagnant, [niveaux](#) \*avancee\_niveaux, TTF\_Font \*\*police, SDL\_Texture \*\*texture\_image\_croix)
- void [demande\\_sauvegarde](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_demande\_sauvegarde, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleur, [itemMenu](#) \*itemsDemandeSauvegarde, int tailleDemandeSauvegarde, int largeur, int hauteur)  
*fenêtre se chargeant de demander à l'utilisateur si il souhaite sauvegarder*
- void [demande\\_quitter\\_niveau](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_demande\_quitter, SDL ↵ \_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleur, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemandeQuitter, int largeur, int hauteur)

- Fonction qui permet de demander à l'utilisateur de quitter le niveau.*
- void [redimensionnement\\_fenetre](#) (SDL\_Event event, int \*largeur, int \*hauteur)  
*Fonction qui permet de récupérer les nouvelles dimensions de la fenêtre pour redimensionner cette dernière et les différents objets.*
  - int [verification\\_sauvegarde](#) ()  
*Vérifie si une sauvegarde existe.*
  - void [sauvegarder\\_partie](#) (SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_↵, interagir, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [modes\\_t](#) modeActif, [personnage\\_t](#) personnage↵, [position\\_t](#) positionActive, [niveaux](#) \*avancee\_niveaux, int tailleNiveaux, time\_t temps\_debut\_partie, int compteur\_mort, int avancee\_succes[10])
  - int [clik\\_case](#) (SDL\_Event event, SDL\_Rect rectangle)  
*Fonction qui permet de renvoyer vrai quand on clique sur un rectangle, faux sinon.*
  - int [clik\\_plein\\_ecran](#) (SDL\_Event event, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_↵ Window \*\*window)  
*Fonction qui permet de mettre la fenêtre en plein écran quand on clique sur le bouton plein écran.*
  - void [deplacement\\_personnage](#) (int \*saut, int \*tombe, int \*position\_x, int \*position\_y, int \*position\_avant\_↵ saut, int sauter, int avancer, int reculer, int tile\_map[18][32], [personnage\\_t](#) personnageActif)
  - void [destruire\\_objets](#) (TTF\_Font \*\*police, Mix\_Music \*\*musique, SDL\_Texture \*\*texture1, SDL\_Texture \*\*texture2, SDL\_Texture \*\*texture3, SDL\_Texture \*\*texture4, SDL\_Texture \*\*texture5, SDL\_Texture \*\*texture6, SDL\_Texture \*\*texture7, SDL\_Texture \*\*texture8, SDL\_Texture \*\*texture9, SDL\_Texture \*\*texture10, SDL\_Texture \*\*texture11, SDL\_Texture \*\*texture12, SDL\_Texture \*\*texture13, SDL\_Texture \*\*texture14, SDL\_Texture \*\*texture15, SDL\_Texture \*\*texture16, SDL\_Texture \*\*texture17, SDL\_Texture \*\*texture18, SDL\_Texture \*\*texture19, SDL\_Texture \*\*texture20, SDL\_Texture \*\*texture21, SDL\_Texture \*\*texture22, SDL\_Texture \*\*texture23, SDL\_Texture \*\*texture24, SDL\_Texture \*\*texture25, SDL\_Texture \*\*texture26, SDL\_Texture \*\*texture27, SDL\_Texture \*\*texture28, SDL\_Texture \*\*texture29, SDL\_Texture \*\*texture30, SDL\_Texture \*\*texture31, SDL\_Texture \*\*texture32, SDL\_Texture \*\*texture33, SDL\_Texture \*\*texture34, SDL\_Texture \*\*texture35, SDL\_Texture \*\*texture36, SDL\_Texture \*\*texture37, SDL\_Texture \*\*texture38, SDL\_Texture \*\*texture39, SDL\_Texture \*\*texture40, SDL\_Texture \*\*texture41, SDL\_Texture \*\*texture42, SDL\_Texture \*\*texture43, SDL\_Texture \*\*texture44, SDL\_Texture \*\*texture45, SDL\_Texture \*\*texture46, SDL\_Texture \*\*texture47, SDL\_Texture \*\*texture48, SDL\_Texture \*\*texture49, SDL\_Texture \*\*texture50, SDL\_Texture \*\*texture51, SDL\_Texture \*\*texture52, SDL\_Texture \*\*texture53, SDL\_Texture \*\*texture54, SDL\_Texture \*\*texture55, SDL\_Texture \*\*texture56, SDL\_Texture \*\*texture57, SDL\_Texture \*\*texture58, SDL\_Texture \*\*texture59, SDL\_Texture \*\*texture60, SDL\_Texture \*\*texture61, SDL\_Texture \*\*texture62, SDL\_Texture \*\*texture63, SDL\_Texture \*\*texture64, SDL\_Texture \*\*texture65, SDL\_Texture \*\*texture66, SDL\_Texture \*\*texture67, SDL\_Texture \*\*texture68, SDL\_Texture \*\*texture69, SDL\_Texture \*\*texture70, SDL\_Texture \*\*texture71, SDL\_Texture \*\*texture72, SDL\_Texture \*\*textures\_images\_succes)
  - void [destruire\\_fenetre\\_rendu](#) (SDL\_Renderer \*\*render, SDL\_Window \*\*window)  
*Fonction qui permet de détruire le rendu et la fenêtre.*

## 4.5.1 Description détaillée

Fichier avec les fichiers utilisé régulièrement.

Définition dans le fichier [fonctions\\_generales.c](#).

## 4.5.2 Documentation des fonctions

### 4.5.2.1 affichage\_texte()

```
void affichage_texte (
    SDL_Renderer ** render,
    SDL_Surface ** surface,
    SDL_Texture ** texture,
    itemMenu * item,
    TTF_Font ** police,
    SDL_Color couleur )
```

Définition à la ligne 55 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.2 chargement\_image()

```
chargement_image (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture,
    char * chemin )
```

Fonction qui permet de charger une image.

##### Paramètres

<i>renderer</i>	rendu sur lequel posé l'image
<i>surface</i>	Surface à utiliser pour récupérer l'image
<i>texture</i>	Texture à crée
<i>chemin</i>	Pointeur sur caractère représentant le chemin d'accès du fichier

##### Voir également

[erreur](#)

Définition à la ligne [30](#) du fichier [fonctions\\_generales.c](#).

#### 4.5.2.3 clic\_case()

```
int clic_case (
    SDL_Event event,
    SDL_Rect rectangle )
```

Fonction qui permet de renvoyer vrai quand on clique sur un rectangle, faux sinon.

##### Paramètres

<i>event</i>	Evenement SDL
<i>rectangle</i>	Rectangle qui a été cliqué ou non

##### Renvoie

booléen représentant si le clic s'est fait dans le rectangle (1 si c'est le cas sinon 0)

Définition à la ligne [414](#) du fichier [fonctions\\_generales.c](#).

#### 4.5.2.4 clic\_plein\_ecran()

```
int clic_plein_ecran (
    SDL_Event event,
    SDL_Rect * rectangle_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Window ** window )
```

Fonction qui permet de mettre la fenêtre en plein écran quand on clique sur le bouton plein écran.

## Paramètres

<i>event</i>	Evenement SDL
<i>rectangle_plein_ecran</i>	Rectangle ou se situe le bouton pour afficher le plein écran ou le retirer
<i>plein_ecran</i>	booléen qui dit si il est en mode plein écran
<i>window</i>	fenêtre à changer en pleine écran ou non

## Renvoie

le changement d'état sous la forme d'un booléen

Définition à la ligne 445 du fichier [fonctions\\_generales.c](#).

## 4.5.2.5 creer\_fenetre\_rendu()

```
void creer_fenetre_rendu (
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    int largeur,
    int hauteur )
```

Fonction qui permet de créer une fenêtre et le rendu.

## Paramètres

<i>window</i>	fenêtre à créer
<i>renderer</i>	Rendu de la fenêtre à créer
<i>largeur</i>	largeur de la fenêtre souhaité
<i>hauteur</i>	hauteur de la fenêtre souhaité

## Voir également

[erreur](#)

Définition à la ligne 78 du fichier [fonctions\\_generales.c](#).

## 4.5.2.6 demande\_quitter\_niveau()

```
void demande_quitter_niveau (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_demande_quitter,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleur,
    itemMenu * itemsDemandeQuitter,
    int tailleDemandeQuitter,
    int largeur,
    int hauteur )
```

Fonction qui permet de demander à l'utilisateur de quitter le niveau.

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>rectangle_demande_quitter</i>	Rectangle de la demande de quitter le niveau SDL.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleur</i>	Couleur du texte.
<i>itemsDemandeQuitter</i>	Tableau d'items pour la demande de quitter le niveau.
<i>tailleDemandeQuitter</i>	Taille du tableau d'items pour la demande de quitter le niveau.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

## Voir également

[affichage\\_texte](#)

Définition à la ligne [252](#) du fichier [fonctions\\_generales.c](#).

#### 4.5.2.7 demande\_sauvegarde()

```
void demande_sauvegarde (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_demande_sauvegarde,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleur,
    itemMenu * itemsDemandeSauvegarde,
    int tailleDemandeSauvegarde,
    int largeur,
    int hauteur )
```

fenêtre se chargeant de demander à l'utilisateur si il souhaite sauvegarder

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>rectangle_demande_sauvegarde</i>	Rectangle de la demande de sauvegarde SDL.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleur</i>	Couleur du texte.
<i>itemsDemandeSauvegarde</i>	Tableau d'items pour la demande de sauvegarde.
<i>tailleDemandeSauvegarde</i>	Taille du tableau d'items pour la demande de sauvegarde.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.



Voir également

[affichage\\_texte](#)

Définition à la ligne 187 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.8 `deplacement_personnage()`

```
void deplacement_personnage (
    int * saut,
    int * tombe,
    int * position_x,
    int * position_y,
    int * position_avant_saut,
    int sauter,
    int avancer,
    int reculer,
    int tile_map[18][32],
    personnage_t personnageActif )
```

Définition à la ligne 491 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.9 `detruire_fenetre_rendu()`

```
void detruire_fenetre_rendu (
    SDL_Renderer ** renderer,
    SDL_Window ** window )
```

Fonction qui permet de détruire le rendu et la fenêtre.

##### Paramètres

<i>renderer</i>	Rendu à détruire
<i>window</i>	fenêtre à détruire

Définition à la ligne 690 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.10 `detruire_objets()`

```
void detruire_objets (
    TTF_Font ** police,
    Mix_Music ** musique,
    SDL_Texture ** texture1,
    SDL_Texture ** texture2,
    SDL_Texture ** texture3,
    SDL_Texture ** texture4,
    SDL_Texture ** texture5,
    SDL_Texture ** texture6,
    SDL_Texture ** texture7,
    SDL_Texture ** texture8,
    SDL_Texture ** texture9,
```

```
SDL_Texture ** texture10,  
SDL_Texture ** texture11,  
SDL_Texture ** texture12,  
SDL_Texture ** texture13,  
SDL_Texture ** texture14,  
SDL_Texture ** texture15,  
SDL_Texture ** texture16,  
SDL_Texture ** texture17,  
SDL_Texture ** texture18,  
SDL_Texture ** texture19,  
SDL_Texture ** texture20,  
SDL_Texture ** texture21,  
SDL_Texture ** texture22,  
SDL_Texture ** texture23,  
SDL_Texture ** texture24,  
SDL_Texture ** texture25,  
SDL_Texture ** texture26,  
SDL_Texture ** texture27,  
SDL_Texture ** texture28,  
SDL_Texture ** texture29,  
SDL_Texture ** texture30,  
SDL_Texture ** texture31,  
SDL_Texture ** texture32,  
SDL_Texture ** texture33,  
SDL_Texture ** texture34,  
SDL_Texture ** texture35,  
SDL_Texture ** texture36,  
SDL_Texture ** texture37,  
SDL_Texture ** texture38,  
SDL_Texture ** texture39,  
SDL_Texture ** texture40,  
SDL_Texture ** texture41,  
SDL_Texture ** texture42,  
SDL_Texture ** texture43,  
SDL_Texture ** texture44,  
SDL_Texture ** texture45,  
SDL_Texture ** texture46,  
SDL_Texture ** texture47,  
SDL_Texture ** texture48,  
SDL_Texture ** texture49,  
SDL_Texture ** texture50,  
SDL_Texture ** texture51,  
SDL_Texture ** texture52,  
SDL_Texture ** texture53,  
SDL_Texture ** texture54,  
SDL_Texture ** texture55,  
SDL_Texture ** texture56,  
SDL_Texture ** texture57,  
SDL_Texture ** texture58,  
SDL_Texture ** texture59,  
SDL_Texture ** texture60,  
SDL_Texture ** texture61,  
SDL_Texture ** texture62,  
SDL_Texture ** texture63,  
SDL_Texture ** texture64,  
SDL_Texture ** texture65,  
SDL_Texture ** texture66,  
SDL_Texture ** texture67,
```

```
SDL_Texture ** texture68,  
SDL_Texture ** texture69,  
SDL_Texture ** texture70,  
SDL_Texture ** texture71,  
SDL_Texture ** texture72,  
SDL_Texture ** textures_images_succes )
```

Définition à la ligne 599 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.11 erreur()

```
erreur (  
    const char * message )
```

Affiche l'erreur en cas de problème et ferme la SDL.

##### Paramètres

<i>message</i>	Un pointeur sur caractère en lecture représentant le message d'erreur
----------------	---

##### Renvoie

Arrêt du programme en Echec

Définition à la ligne 14 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.12 initialisation\_objets()

```
void initialisation_objets (  
    SDL_Renderer ** renderer,  
    SDL_Surface ** surface,  
    SDL_Texture ** texture_image_plein_ecran,  
    SDL_Texture ** texture_image_retour_en_arriere,  
    SDL_Texture ** texture_image_options,  
    SDL_Texture ** texture_image_passer,  
    itemMenu * itemsDemandeSauvegarde,  
    itemMenu * itemsDemandeQuitter,  
    SDL_Texture ** texture_image_fin_premiers_niveaux,  
    SDL_Texture ** texture_image_monstre_terrestre,  
    SDL_Texture ** texture_image_monstre_volant,  
    SDL_Texture ** texture_image_perso_1_gagnant,  
    SDL_Texture ** texture_image_perso_2_gagnant,  
    niveaux * avancee_niveaux,  
    TTF_Font ** police,  
    SDL_Texture ** texture_image_croix )
```

Définition à la ligne 117 du fichier [fonctions\\_generales.c](#).

#### 4.5.2.13 redimensionnement\_fenetre()

```
void redimensionnement_fenetre (
    SDL_Event event,
    int * largeur,
    int * hauteur )
```

Fonction qui permet de récupérer les nouvelles dimensions de la fenêtre pour redimensionner cette dernière et les différents objets.

## Paramètres

<i>event</i>	Evenement SDL
<i>largeur</i>	largeur de la fenêtre a redimensionné
<i>hauteur</i>	hauteur de la fenêtre a redimensionné

Définition à la ligne 309 du fichier [fonctions\\_generales.c](#).

**4.5.2.14 sauvegarder\_partie()**

```
void sauvegarder_partie (
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_sauter_monter,
    SDL_Keycode * touche_descendre,
    SDL_Keycode * touche_interagir,
    barreDeSon * barre_de_son,
    itemMenu * pseudo,
    modes_t modeActif,
    personnage_t personnageActif,
    position_t positionActive,
    niveaux * avancee_niveaux,
    int tailleNiveaux,
    time_t temps_debut_partie,
    int compteur_mort,
    int avancee_succes[10] )
```

Définition à la ligne 359 du fichier [fonctions\\_generales.c](#).

**4.5.2.15 verification\_sauvegarde()**

```
int verification_sauvegarde ( )
```

Vérifie si une sauvegarde existe.

## Renvoie

booléen représentant si il y a une sauvegarde ou non (1 si existant, sinon 0)

## Voir également

[erreur](#)

Définition à la ligne 324 du fichier [fonctions\\_generales.c](#).

## 4.6 fonctions\_generales.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_generales.c
00003  * \brief Fichier avec les fonctions utilisées régulièrement
00004  */
00005
00006 #include <../fichiers_h/fonctions_generales.h>
00007
00008 /**
00009  * \fn erreur(const char *message)
00010  * \brief Affiche l'erreur en cas de problème et ferme la SDL
00011  * \param message Un pointeur sur caractère en lecture représentant le message d'erreur
00012  * \return Arrêt du programme en Echec
00013  */
00014 void erreur(const char *message) {
00015     SDL_Log("ERREUR : %s > %s\n", message, SDL_GetError());
00016     SDL_Quit();
00017     exit(EXIT_FAILURE);
00018 }
00019
00020 /**
00021  * \fn chargement_image(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, char
00022  *chemin)
00023  * \brief Fonction qui permet de charger une image
00024  * \param renderer rendu sur lequel posé l'image
00025  * \param surface Surface à utiliser pour récupérer l'image
00026  * \param texture Texture à créer
00027  * \param chemin Pointeur sur caractère représentant le chemin d'accès du fichier
00028  * \see erreur
00029  */
00030 void chargement_image(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, char
*chemin) {
00031     (*surface) = IMG_Load(chemin);
00032
00033     if((*surface) == NULL)
00034         erreur("Chargement de l'image");
00035
00036     (*texture) = SDL_CreateTextureFromSurface((*renderer), (*surface));
00037
00038     if((*texture) == NULL)
00039         erreur("Création de la texture");
00040
00041     SDL_FreeSurface((*surface));
00042 }
00043
00044 /**
00045  * \fn affichage_texte(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, itemMenu
*item, TTF_Font **police, SDL_Color couleur)
00046  * \brief Affiche du texte sur la fenêtre
00047  * \param renderer Rendu de la fenêtre
00048  * \param surface Surface ou appliquer la texture
00049  * \param texture Texture à afficher
00050  * \param item Texte à afficher
00051  * \param police police d'écriture
00052  * \param couleur couleur
00053  */
00054 void affichage_texte(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, itemMenu
*item,
00055     TTF_Font **police, SDL_Color couleur) {
00056     SDL_RenderFillRect((*renderer), &(item->rectangle));
00057
00058     (*surface) = TTF_RenderText_Solid((*police), item->text, couleur);
00059     (*texture) = SDL_CreateTextureFromSurface((*renderer), (*surface));
00060
00061     SDL_RenderCopy((*renderer), (*texture), NULL, &(item->rectangle));
00062
00063     SDL_FreeSurface((*surface));
00064     SDL_DestroyTexture((*texture));
00065 }
00066
00067 /**
00068  * \fn void creer_fenetre_rendu(SDL_Window **window, SDL_Renderer **renderer, int largeur, int hauteur)
00069  */
00070 void creer_fenetre_rendu(SDL_Window **window, SDL_Renderer **renderer, int largeur, int hauteur)
00071 {
00072     *window = SDL_CreateWindow("SDL", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, largeur,
hauteur, SDL_WINDOW_OPENGL);
00073     *renderer = SDL_CreateRenderer(*window, -1, SDL_RENDERER_ACCELERATED);
00074 }

```

```

00078 void creer_fenetre_rendu(SDL_Window **window, SDL_Renderer **renderer, int largeur, int hauteur) {
00079
00080     /* Création de la fenêtre */
00081     (*window) = SDL_CreateWindow("MétaTravers", SDL_WINDOWPOS_CENTERED,
00082                                   SDL_WINDOWPOS_CENTERED,
00083                                   largeur, hauteur,
00084                                   SDL_WINDOW_RESIZABLE | SDL_WINDOW_SHOWN);
00085
00086     if((*window) == NULL)
00087         erreur("Création fenêtre échouée");
00088
00089     /* Création du rendu */
00090     (*renderer) = SDL_CreateRenderer((*window), -1, SDL_RENDERER_ACCELERATED |
00091                                     SDL_RENDERER_PRESENTVSYNC);
00092
00093     if((*renderer) == NULL)
00094         erreur("Création rendu échoué");
00095 }
00096 /**
00097  * \fn void initialisation_objets(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00098  **texture_image_plein_ecran, SDL_Texture **texture_image_retour_en_arriere, SDL_Texture
00099  **texture_image_options, SDL_Texture **texture_image_passer, itemMenu *itemsDemandeSauvegarde,
00100  itemMenu *itemsDemandeQuitter, SDL_Texture **texture_image_fin_premiers_niveaux, SDL_Texture
00101  **texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant, SDL_Texture
00102  **texture_image_perso_1_gagnant, SDL_Texture **texture_image_perso_2_gagnant, niveaux
00103  *avancee_niveaux, TTF_Font **police)
00104  * \brief Fonctions qui permet d'initialiser les objets globaux
00105  * \param renderer Pointeur vers le renderer SDL.
00106  * \param surface Surface SDL.
00107  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00108  * \param texture_image_retour_en_arriere Texture de l'image du bouton retour en arrière.
00109  * \param texture_image_options Texture de l'image du bouton options.
00110  * \param texture_image_passer Texture de l'image du bouton passer.
00111  * \param itemsDemandeSauvegarde Tableau d'items pour la demande de sauvegarde.
00112  * \param itemsDemandeQuitter Tableau d'items pour la demande de quitter.
00113  * \param texture_image_fin_premiers_niveaux Texture de l'image de fin des premiers niveaux.
00114  * \param texture_image_monstre_terrestre Texture de l'image du monstre terrestre.
00115  * \param texture_image_monstre_volant Texture de l'image du monstre volant.
00116  * \param texture_image_perso_1_gagnant Texture de l'image du personnage 1 gagnant.
00117  * \param texture_image_perso_2_gagnant Texture de l'image du personnage 2 gagnant.
00118  * \param avancee_niveaux Structure de progression des niveaux.
00119  * \param police Police de caractères TTF.
00120  * \see chargement_image
00121  * \see erreur
00122  */
00123 void initialisation_objets(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00124 **texture_image_plein_ecran,
00125 SDL_Texture **texture_image_retour_en_arriere, SDL_Texture
00126 **texture_image_options,
00127 SDL_Texture **texture_image_passer, itemMenu *itemsDemandeSauvegarde,
00128 itemMenu *itemsDemandeQuitter,
00129 SDL_Texture **texture_image_fin_premiers_niveaux, SDL_Texture
00130 **texture_image_monstre_terrestre,
00131 SDL_Texture **texture_image_monstre_volant, SDL_Texture
00132 **texture_image_perso_1_gagnant,
00133 SDL_Texture **texture_image_perso_2_gagnant,
00134 niveaux *avancee_niveaux, TTF_Font **police, SDL_Texture
00135 **texture_image_croix) {
00136
00137     /* Initialisation de l'image du plein écran du menu */
00138     chargement_image(renderer, surface, texture_image_plein_ecran, "./images/plein_ecran_blanc.png");
00139
00140     /* Initialisation de l'image du retour en arrière */
00141     chargement_image(renderer, surface, texture_image_retour_en_arriere,
00142                     "./images/retour_en_arriere.png");
00143
00144     /* Initialisation de l'image des options du menu */
00145     chargement_image(renderer, surface, texture_image_options, "./images/options_blanc.png");
00146
00147     /* Initialisation de l'image du passer du menu */
00148     chargement_image(renderer, surface, texture_image_passer, "./images/passer.png");
00149
00150     /* Initialisation de l'image de la croix */
00151     chargement_image(renderer, surface, texture_image_croix, "./images/croix.png");
00152
00153     /* Initialisation de la police */
00154     if((*police) = TTF_OpenFont("./polices/04B_11__.TTF", 20)) == NULL)
00155         erreur("Chargement de la police");
00156
00157     /* Initialisation des images pour les collectibles */
00158     chargement_image(renderer, surface, &(avancee_niveaux[0].texture_image_collectible),
00159                     "./images/niveau_1/collectible_niveau_1.png");
00160     chargement_image(renderer, surface, &(avancee_niveaux[1].texture_image_collectible),
00161                     "./images/niveau_2/collectible_niveau_2.png");
00162     chargement_image(renderer, surface, &(avancee_niveaux[2].texture_image_collectible),
00163                     "./images/niveau_3/collectible_niveau_3.png");

```

```

00148     chargement_image(renderer, surface, &(avancee_niveaux[3].texture_image_collectible),
00149     "./images/niveau_4/collectible_niveau_4.png");
00149
00150     /* Initialisation de l'image de la fin des premiers niveaux */
00151     chargement_image(renderer, surface, texture_image_fin_premiers_niveaux,
00152     "./images/fin_premiers_niveaux.png");
00152
00153     /* Initialisation des images des monstres */
00154     chargement_image(renderer, surface, texture_image_monstre_terrestre,
00155     "./images/monstre_terrestre.png");
00155     chargement_image(renderer, surface, texture_image_monstre_volant, "./images/monstre_volant.png");
00156
00157     /* Initialisation des images des poses gagnantes des personnages */
00158     chargement_image(renderer, surface, texture_image_perso_1_gagnant,
00159     "./images/personnages/personnage_masculin_gagnant.png");
00159     chargement_image(renderer, surface, texture_image_perso_2_gagnant,
00160     "./images/personnages/personnage_feminin_gagnant.png");
00160
00161     /* Initialisation du texte dans les items de la demande de sauvegarde */
00162     sprintf(itemsDemandeSauvegarde[0].texte, " Voulez-vous sauvegarder la partie avant de quitter ?
00163 ");
00163     sprintf(itemsDemandeSauvegarde[1].texte, " Oui ");
00164     sprintf(itemsDemandeSauvegarde[2].texte, " Non ");
00165
00166     /* Initialisation du texte dans les items de la demande de quitter le niveau */
00167     sprintf(itemsDemandeQuitter[0].texte, " Voulez-vous quittez le niveau ? ");
00168     sprintf(itemsDemandeQuitter[1].texte, " Oui ");
00169     sprintf(itemsDemandeQuitter[2].texte, " Non ");
00170 }
00171
00172 /**
00173  * \fn void demande_sauvegarde(SDL_Renderer **renderer, SDL_Rect *rectangle_demande_sauvegarde,
00174  * SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, SDL_Color couleur, itemMenu
00175  * itemsDemandeSauvegarde, int tailleDemandeSauvegarde, int largeur, int hauteur)
00176  * \brief fenêtre se chargeant de demander à l'utilisateur si il souhaite sauvegarder
00177  * \param renderer Pointeur vers le renderer SDL.
00178  * \param rectangle_demande_sauvegarde Rectangle de la demande de sauvegarde SDL.
00179  * \param surface Surface SDL.
00180  * \param texture_texte Texture du texte SDL.
00181  * \param police Police de caractères TTF.
00182  * \param couleur Couleur du texte.
00183  * \param itemsDemandeSauvegarde Tableau d'items pour la demande de sauvegarde.
00184  * \param tailleDemandeSauvegarde Taille du tableau d'items pour la demande de sauvegarde.
00185  * \param largeur Largeur de l'écran.
00186  * \param hauteur Hauteur de l'écran.
00187  * \see affichage_texte
00188  */
00187 void demande_sauvegarde(SDL_Renderer **renderer, SDL_Rect *rectangle_demande_sauvegarde,
00188     SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00189     SDL_Color couleur,
00190     itemMenu *itemsDemandeSauvegarde, int tailleDemandeSauvegarde, int largeur,
00191     int hauteur) {
00192
00193     int i;
00194
00195     /* Affichage du rectangle de la demande de sauvegarde */
00196     rectangle_demande_sauvegarde->x = largeur / 6;
00197     rectangle_demande_sauvegarde->y = hauteur / 6;
00198     rectangle_demande_sauvegarde->w = largeur / 3 * 2;
00199     rectangle_demande_sauvegarde->h = hauteur / 3 * 2;
00200
00201     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00202     SDL_RenderFillRect((*renderer), rectangle_demande_sauvegarde);
00203
00204     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00205     SDL_RenderDrawRect((*renderer), rectangle_demande_sauvegarde);
00206
00207     /* Affichage du rectangle de la question de la demande de sauvegarde */
00208     itemsDemandeSauvegarde[0].rectangle.x = largeur / 6 + largeur / 100;
00209     itemsDemandeSauvegarde[0].rectangle.y = hauteur / 4 + hauteur / 20;
00210     itemsDemandeSauvegarde[0].rectangle.w = largeur / 3 * 2 - largeur / 50;
00211     itemsDemandeSauvegarde[0].rectangle.h = hauteur / 10;
00212
00213     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00214
00215     affichage_texte(renderer, surface, texture_texte, &(itemsDemandeSauvegarde[0]),
00216     police, couleur);
00217
00218     /* Affichage des rectangles des réponses de la demande de sauvegarde */
00219     for(i = 1; i < tailleDemandeSauvegarde; i++) {
00220         itemsDemandeSauvegarde[i].rectangle.x = largeur / 3 + (i-1) * largeur / 3 - (i-1) * largeur /
00221         10;
00222         itemsDemandeSauvegarde[i].rectangle.y = hauteur - hauteur / 4 - hauteur / 20 - hauteur / 10;
00223         itemsDemandeSauvegarde[i].rectangle.w = largeur / 10;
00224         itemsDemandeSauvegarde[i].rectangle.h = hauteur / 10;
00225     }

```



```

00224     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00225
00226     affichage_texte(renderer, surface, texture_texte, &(itemsDemandeSauvegarde[i]),
00227                     police, couleur);
00228
00229     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00230     SDL_RenderDrawRect((*renderer), &(itemsDemandeSauvegarde[i].rectangle));
00231 }
00232
00233 /* Affiche le rendu */
00234 SDL_RenderPresent((*renderer));
00235 }
00236
00237 /**
00238  * \fn void demande_quitter_niveau(SDL_Renderer **renderer, SDL_Rect *rectangle_demande_quitter,
00239  * SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, SDL_Color couleur, itemMenu
00240  * itemsDemandeQuitter, int tailleDemandeQuitter, int largeur, int hauteur)
00241  * \brief Fonction qui permet de demander à l'utilisateur de quitter le niveau
00242  * \param renderer Pointeur vers le renderer SDL.
00243  * \param rectangle_demande_quitter Rectangle de la demande de quitter le niveau SDL.
00244  * \param surface Surface SDL.
00245  * \param texture_texte Texture du texte SDL.
00246  * \param police Police de caractères TTF.
00247  * \param couleur Couleur du texte.
00248  * \param itemsDemandeQuitter Tableau d'items pour la demande de quitter le niveau.
00249  * \param tailleDemandeQuitter Taille du tableau d'items pour la demande de quitter le niveau.
00250  * \param largeur Largeur de l'écran.
00251  * \param hauteur Hauteur de l'écran.
00252  * \see affichage_texte
00253  */
00254 void demande_quitter_niveau(SDL_Renderer **renderer, SDL_Rect *rectangle_demande_quitter,
00255                             SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00256                             SDL_Color couleur,
00257                             itemMenu *itemsDemandeQuitter, int tailleDemandeQuitter, int largeur, int
00258                             hauteur) {
00259     int i;
00260
00261     /* Affichage du rectangle de la demande de quitter le niveau */
00262     rectangle_demande_quitter->x = largeur / 6;
00263     rectangle_demande_quitter->y = hauteur / 6;
00264     rectangle_demande_quitter->w = largeur / 3 * 2;
00265     rectangle_demande_quitter->h = hauteur / 3 * 2;
00266
00267     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00268     SDL_RenderFillRect((*renderer), rectangle_demande_quitter);
00269
00270     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00271     SDL_RenderDrawRect((*renderer), rectangle_demande_quitter);
00272
00273     /* Affichage du rectangle de la question de la demande de quitter le niveau */
00274     itemsDemandeQuitter[0].rectangle.x = largeur / 6 + largeur / 100;
00275     itemsDemandeQuitter[0].rectangle.y = hauteur / 4 + hauteur / 20;
00276     itemsDemandeQuitter[0].rectangle.w = largeur / 3 * 2 - largeur / 50;
00277     itemsDemandeQuitter[0].rectangle.h = hauteur / 10;
00278
00279     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00280
00281     affichage_texte(renderer, surface, texture_texte, &(itemsDemandeQuitter[0]),
00282                     police, couleur);
00283
00284     /* Affichage des rectangles des réponses de la demande de quitter le niveau */
00285     for(i = 1; i < tailleDemandeQuitter; i++) {
00286         itemsDemandeQuitter[i].rectangle.x = largeur / 3 + (i-1) * largeur / 3 - (i-1) * largeur / 10;
00287         itemsDemandeQuitter[i].rectangle.y = hauteur - hauteur / 4 - hauteur / 20 - hauteur / 10;
00288         itemsDemandeQuitter[i].rectangle.w = largeur / 10;
00289         itemsDemandeQuitter[i].rectangle.h = hauteur / 10;
00290
00291         SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00292
00293         affichage_texte(renderer, surface, texture_texte, &(itemsDemandeQuitter[i]),
00294                         police, couleur);
00295
00296         SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00297         SDL_RenderDrawRect((*renderer), &(itemsDemandeQuitter[i].rectangle));
00298     }
00299
00300     /* Affiche le rendu */
00301     SDL_RenderPresent((*renderer));
00302 }
00303
00304 /**
00305  * \fn void redimensionnement_fenetre(SDL_Event event, int *largeur, int *hauteur)
00306  * \brief Fonction qui permet de récupérer les nouvelles dimensions de la fenêtre pour redimensionner
00307  * cette dernière et les différents objets
00308  * \param event Evenement SDL

```

```

00306 * \param largeur largeur de la fenetre a redimensionné
00307 * \param hauteur hauteur de la fenetre a redimensionné
00308 */
00309 void redimensionnement_fenetre(SDL_Event event, int *largeur, int *hauteur) {
00310
00311     if(event.window.event == SDL_WINDOWEVENT_RESIZED) {
00312         (*largeur) = event.window.data1;
00313         (*hauteur) = event.window.data2;
00314     }
00315 }
00316 }
00317
00318 /**
00319 * \fn int verification_sauvegarde()
00320 * \brief Vérifie si une sauvegarde existe
00321 * \return booléen représentant si il y a une sauvegarde ou non (1 si existant, sinon 0)
00322 * \see erreur
00323 */
00324 int verification_sauvegarde() {
00325
00326     FILE *fichier_sauvegarde;
00327
00328     /* Ouverture du fichier en mode lecture */
00329     fichier_sauvegarde = fopen("./sauvegardes/sauvegarde.txt", "r");
00330
00331     /* Vérifie si le fichier existe */
00332     if (fichier_sauvegarde == NULL)
00333         return 0;
00334
00335     /* Fermeture du fichier */
00336     if (fclose(fichier_sauvegarde) != 0)
00337         erreur("Fermeture du fichier");
00338
00339     return 1;
00340 }
00341
00342 /**
00343 * \fn void sauvegarder_partie(SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
    SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir,
    barreDeSon *barre_de_son, itemMenu *pseudo, modes_t modeActif, personnage_t personnageActif,
    position_t positionActive, niveaux *avancee_niveaux, int tailleNiveaux)
00344 * \brief Fonction qui permet de sauvegarder la partie dans un fichier
00345 * \param touche_aller_a_droite Touche pour aller à droite.
00346 * \param touche_aller_a_gauche Touche pour aller à gauche.
00347 * \param touche_sauter_monter Touche pour sauter/monter.
00348 * \param touche_descendre Touche pour descendre.
00349 * \param touche_interagir Touche pour interagir.
00350 * \param barre_de_son Réglages de la barre de son.
00351 * \param pseudo Pseudo du joueur.
00352 * \param modeActif Mode actif du jeu.
00353 * \param personnageActif Personnage actif.
00354 * \param positionActive Position active.
00355 * \param avancee_niveaux Structure de progression des niveaux.
00356 * \param tailleNiveaux Taille du tableau d'items pour les niveaux.
00357 * \see erreur
00358 */
00359 void sauvegarder_partie(SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
    SDL_Keycode *touche_sauter_monter,
00360     SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, barreDeSon
    *barre_de_son, itemMenu *pseudo,
00361     modes_t modeActif, personnage_t personnageActif, position_t positionActive,
00362     niveaux *avancee_niveaux, int tailleNiveaux, time_t temps_debut_partie, int
    compteur_mort, int avancee_succes[10]) {
00363
00364     FILE *fichier_sauvegarde;
00365
00366     int i;
00367
00368     /* Ouverture du fichier en mode écriture */
00369     fichier_sauvegarde = fopen("./sauvegardes/sauvegarde.txt", "w");
00370
00371     fprintf(fichier_sauvegarde, "%f\n", barre_de_son[0].volume);
00372     fprintf(fichier_sauvegarde, "%f\n", barre_de_son[1].volume);
00373     fprintf(fichier_sauvegarde, "%s\n", SDL_GetKeyName((*touche_aller_a_droite)));
00374     fprintf(fichier_sauvegarde, "%s\n", SDL_GetKeyName((*touche_aller_a_gauche)));
00375     fprintf(fichier_sauvegarde, "%s\n", SDL_GetKeyName((*touche_sauter_monter)));
00376     fprintf(fichier_sauvegarde, "%s\n", SDL_GetKeyName((*touche_descendre)));
00377     fprintf(fichier_sauvegarde, "%s\n", SDL_GetKeyName((*touche_interagir)));
00378     fprintf(fichier_sauvegarde, "%s\n", pseudo->texte);
00379
00380     fprintf(fichier_sauvegarde, "%d\n", personnageActif);
00381
00382     fprintf(fichier_sauvegarde, "%d\n", modeActif);
00383
00384     fprintf(fichier_sauvegarde, "%d\n", positionActive);
00385
00386     for(i = 0; i < tailleNiveaux; i++) {

```

```

00387
00388     fprintf(fichier_sauvegarde, "%d %d %d %d\n", avancee_niveaux[i].niveau_fini,
00389             avancee_niveaux[i].numero_collectible[0],
00390             avancee_niveaux[i].numero_collectible[1],
00391             avancee_niveaux[i].numero_collectible[2]);
00392 }
00393
00394 fprintf(fichier_sauvegarde, "%ld\n", temps_debut_partie);
00395
00396 fprintf(fichier_sauvegarde, "%d\n", compteur_mort);
00397
00398 for(i = 0; i < 10; i++)
00399     fprintf(fichier_sauvegarde, "%d ", avancee_succes[i]);
00400
00401 /* Fermeture du fichier */
00402 if (fclose(fichier_sauvegarde) != 0)
00403     erreur("Fermeture du fichier");
00404 }
00405
00406 /**
00407  * \fn int clic_case(SDL_Event event, SDL_Rect rectangle)
00408  * \brief Fonction qui permet de renvoyer vrai quand on clique sur un rectangle, faux sinon
00409  * \param event Evenement SDL
00410  * \param rectangle Rectangle qui a été cliqué ou non
00411  * \return booléen représentant si le clic s'est fait dans le rectangle (1 si c'est le cas sinon 0)
00412  *
00413  */
00414 int clic_case(SDL_Event event, SDL_Rect rectangle) {
00415
00416     Mix_Chunk *effet_sonore = NULL;
00417
00418     if((event.button.x >= rectangle.x) &&
00419        (event.button.x <= rectangle.x + rectangle.w) &&
00420        (event.button.y >= rectangle.y) &&
00421        (event.button.y <= rectangle.y + rectangle.h) &&
00422        (event.button.button == SDL_BUTTON_LEFT)) {
00423
00424         /* Effet sonore quand on clique sur un bouton */
00425         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/clic_bouton.wav")) == NULL)
00426             erreur("Chargement de l'effet sonore");
00427
00428         Mix_PlayChannel(1, effet_sonore, 0);
00429
00430         return 1;
00431     }
00432
00433     return 0;
00434 }
00435
00436 /**
00437  * \fn int clic_plein_ecran(SDL_Event event, SDL_Rect *rectangle_plein_ecran, SDL_bool *plein_ecran,
00438  * SDL_Window **window)
00439  * \brief Fonction qui permet de mettre la fenêtre en plein écran quand on clique sur le bouton plein
00440  * écran
00441  * \param event Evenement SDL
00442  * \param rectangle_plein_ecran Rectangle ou se situe le bouton pour afficher le plein écran ou le
00443  * retirer
00444  * \param plein_ecran booléen qui dit si il est en mode plein écran
00445  * \param window fenêtre à changer en pleine écran ou non
00446  * \return le changement d'état sous la forme d'un booléen
00447  */
00448 int clic_plein_ecran(SDL_Event event, SDL_Rect *rectangle_plein_ecran, SDL_bool *plein_ecran,
00449                      SDL_Window **window) {
00450
00451     Mix_Chunk *effet_sonore = NULL;
00452
00453     if ((event.button.x >= rectangle_plein_ecran->x) &&
00454         (event.button.x <= rectangle_plein_ecran->x + rectangle_plein_ecran->w) &&
00455         (event.button.y >= rectangle_plein_ecran->y) &&
00456         (event.button.y <= rectangle_plein_ecran->y + rectangle_plein_ecran->h) &&
00457         (event.button.button == SDL_BUTTON_LEFT)) {
00458
00459         /* Effet sonore quand on clique sur un bouton */
00460         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/clic_bouton.wav")) == NULL)
00461             erreur("Chargement de l'effet sonore");
00462
00463         Mix_PlayChannel(1, effet_sonore, 0);
00464
00465         if ((*plein_ecran)) {
00466
00467             SDL_SetWindowFullscreen((*window), 0);
00468             (*plein_ecran) = SDL_FALSE;
00469         }
00470
00471         else {
00472
00473             SDL_SetWindowFullscreen((*window), SDL_WINDOW_FULLSCREEN_DESKTOP);
00474             (*plein_ecran) = SDL_TRUE;
00475         }
00476     }

```

```

00470     }
00471
00472     return 1;
00473 }
00474
00475     return 0;
00476 }
00477
00478 /**
00479  * \fn void deplacement_personnage(int *saut, int *tombe, int *position_x, int *position_y, int
00480  * \brief Fonction qui permet de déplacer le personnage dans les différents niveaux
00481  * \param saut Indicateur de saut.
00482  * \param tombe Indicateur de chute.
00483  * \param position_x Position en x du personnage.
00484  * \param position_y Position en y du personnage.
00485  * \param position_avant_saut Position avant le saut.
00486  * \param sauter Indicateur de tentative de saut.
00487  * \param avancer Indicateur d'avancement.
00488  * \param reculer Indicateur de recul.
00489  * \param tile_map Carte du niveau.
00490  */
00491 void deplacement_personnage(int *saut, int *tombe, int *position_x, int *position_y, int
00492 *position_avant_saut,
00493                             int sauter, int avancer, int reculer, int tile_map[18][32], personnage_t
00494 personnageActif) {
00495     Mix_Chunk *effet_sonore = NULL;
00496
00497     /* Cas où la touche pour sauter est pressée */
00498     if((!(*saut)) && (!(*tombe)) && (sauter)) {
00499         if(personnageActif == PERSONNAGE_1) {
00500             /* Effet sonore quand le premier personnage saute */
00501             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/saut_masculin.wav")) == NULL)
00502                 erreur("Chargement de l'effet sonore");
00503         }
00504
00505         else if(personnageActif == PERSONNAGE_2) {
00506             /* Effet sonore quand le second personnage saute */
00507             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/saut_feminin.wav")) == NULL)
00508                 erreur("Chargement de l'effet sonore");
00509         }
00510
00511         Mix_PlayChannel(1, effet_sonore, 0);
00512         (*position_avant_saut) = (*position_y);
00513         (*saut) = 1;
00514     }
00515
00516     /* Cas où la touche pour aller à gauche est pressée */
00517     if(((!(tile_map[(*position_y)][(*position_x) - 1])) ||
00518         ((tile_map[(*position_y)][(*position_x) - 1] >= 3) &&
00519         (tile_map[(*position_y)][(*position_x) - 1] <= 9)) ||
00520         (tile_map[(*position_y)][(*position_x) - 1] == 11) ||
00521         (tile_map[(*position_y)][(*position_x) - 1] == 12)) && (reculer)) {
00522         (*position_x)--;
00523
00524         if((!(*saut)) && (!(*tombe))) {
00525             (*tombe) = 1;
00526             SDL_Delay(75);
00527         }
00528     }
00529
00530     /* Cas où la touche pour aller à droite est pressée */
00531     if(((!(tile_map[(*position_y)][(*position_x) + 1])) ||
00532         ((tile_map[(*position_y)][(*position_x) + 1] >= 3) &&
00533         (tile_map[(*position_y)][(*position_x) + 1] <= 9)) ||
00534         (tile_map[(*position_y)][(*position_x) + 1] == 11) ||
00535         (tile_map[(*position_y)][(*position_x) + 1] == 12)) && (avancer)) {
00536         (*position_x)++;
00537
00538         if((!(*saut)) && (!(*tombe))) {
00539             (*tombe) = 1;
00540             SDL_Delay(75);
00541         }
00542     }
00543
00544     /* Cas où le personnage est entrain de sauter */
00545     if((*saut)) {
00546         if ((*position_y) < (*position_avant_saut)-2) {

```

```

00554
00555         (*tombe) = 1;
00556         (*saut) = 0;
00557     }
00558
00559     else if((tile_map[(*position_y) - 1][( *position_x)] == 0) ||
00560             (tile_map[( *position_y) - 1][( *position_x)] == 5) ||
00561             (tile_map[( *position_y) - 1][( *position_x)] == 6) ||
00562             (tile_map[( *position_y) - 1][( *position_x)] == 9)) {
00563
00564         (*position_y) -= 1;
00565         SDL_Delay(75);
00566     }
00567
00568     else {
00569
00570         (*tombe) = 1;
00571         (*saut) = 0;
00572     }
00573 }
00574
00575 /* Cas où le personnage est entrain de tomber */
00576 if((*tombe)) {
00577
00578     if ((tile_map[( *position_y) + 1][( *position_x)] == 1) || (tile_map[( *position_y) +
00579 1][( *position_x)] == 10) ||
00580         (tile_map[( *position_y) + 1][( *position_x)] == 13) || (tile_map[( *position_y) +
00581 1][( *position_x)] == 14) ||
00582         (tile_map[( *position_y) + 1][( *position_x)] == 15) || (tile_map[( *position_y) +
00583 1][( *position_x)] == 16)) {
00584
00585         (*tombe) = 0;
00586     }
00587
00588     else {
00589
00590         (*position_y) += 1;
00591         SDL_Delay(75);
00592     }
00593 }
00594 /**
00595  * \fn
00596  * \brief Fonction qui permet de détruire les objets initialisés
00597  * \param police Police d'écriture qui sera désallouer
00598  * \param texture_1_to_71 les 71 textures à désallouer
00599  */
00599 void detruire_objets(TTF_Font **police, Mix_Music **musique, SDL_Texture **texture1, SDL_Texture
**texture2,
00600                     SDL_Texture **texture3, SDL_Texture **texture4, SDL_Texture **texture5,
SDL_Texture **texture6,
00601                     SDL_Texture **texture7, SDL_Texture **texture8,
SDL_Texture **texture9, SDL_Texture **texture10, SDL_Texture **texture11,
00602                     SDL_Texture **texture12, SDL_Texture **texture13, SDL_Texture **texture14,
SDL_Texture **texture15, SDL_Texture **texture16,
00603                     SDL_Texture **texture17, SDL_Texture **texture18,
SDL_Texture **texture19, SDL_Texture **texture20,
00604                     SDL_Texture **texture21, SDL_Texture **texture22,
SDL_Texture **texture23, SDL_Texture **texture24,
00605                     SDL_Texture **texture25, SDL_Texture **texture26,
SDL_Texture **texture27, SDL_Texture **texture28,
00606                     SDL_Texture **texture29, SDL_Texture **texture30,
SDL_Texture **texture31, SDL_Texture **texture32,
00607                     SDL_Texture **texture33, SDL_Texture **texture34,
SDL_Texture **texture35, SDL_Texture **texture36, SDL_Texture **texture37,
00608                     SDL_Texture **texture38, SDL_Texture **texture39, SDL_Texture **texture40,
SDL_Texture **texture41, SDL_Texture **texture42,
00609                     SDL_Texture **texture43, SDL_Texture **texture44,
SDL_Texture **texture45, SDL_Texture **texture46,
00610                     SDL_Texture **texture47, SDL_Texture **texture48,
SDL_Texture **texture49, SDL_Texture **texture50,
00611                     SDL_Texture **texture51, SDL_Texture **texture52,
SDL_Texture **texture53, SDL_Texture **texture54,
00612                     SDL_Texture **texture55, SDL_Texture **texture56,
SDL_Texture **texture57, SDL_Texture **texture58,
00613                     SDL_Texture **texture59, SDL_Texture **texture60,
SDL_Texture **texture61, SDL_Texture **texture62,
00614                     SDL_Texture **texture63, SDL_Texture **texture64,
SDL_Texture **texture65, SDL_Texture **texture66,
00615                     SDL_Texture **texture67, SDL_Texture **texture68,
SDL_Texture **texture69, SDL_Texture **texture70,
00616                     SDL_Texture **texture71, SDL_Texture **texture72,
SDL_Texture **textures_images_succes) {
00617
00618     int i;
00619 }
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635

```

```

00636     /* Destructions des textures */
00637     SDL_DestroyTexture((*texture1)); SDL_DestroyTexture((*texture2));
00638     SDL_DestroyTexture((*texture3)); SDL_DestroyTexture((*texture4));
00639     SDL_DestroyTexture((*texture5)); SDL_DestroyTexture((*texture6));
00640     SDL_DestroyTexture((*texture7)); SDL_DestroyTexture((*texture8));
00641     SDL_DestroyTexture((*texture9)); SDL_DestroyTexture((*texture10));
00642     SDL_DestroyTexture((*texture11)); SDL_DestroyTexture((*texture12));
00643     SDL_DestroyTexture((*texture13)); SDL_DestroyTexture((*texture14));
00644     SDL_DestroyTexture((*texture15)); SDL_DestroyTexture((*texture16));
00645     SDL_DestroyTexture((*texture17)); SDL_DestroyTexture((*texture18));
00646     SDL_DestroyTexture((*texture19)); SDL_DestroyTexture((*texture20));
00647     SDL_DestroyTexture((*texture21)); SDL_DestroyTexture((*texture22));
00648     SDL_DestroyTexture((*texture23)); SDL_DestroyTexture((*texture24));
00649     SDL_DestroyTexture((*texture25)); SDL_DestroyTexture((*texture26));
00650     SDL_DestroyTexture((*texture27)); SDL_DestroyTexture((*texture28));
00651     SDL_DestroyTexture((*texture29)); SDL_DestroyTexture((*texture30));
00652     SDL_DestroyTexture((*texture31)); SDL_DestroyTexture((*texture32));
00653     SDL_DestroyTexture((*texture33)); SDL_DestroyTexture((*texture34));
00654     SDL_DestroyTexture((*texture35)); SDL_DestroyTexture((*texture36));
00655     SDL_DestroyTexture((*texture37)); SDL_DestroyTexture((*texture38));
00656     SDL_DestroyTexture((*texture39)); SDL_DestroyTexture((*texture40));
00657     SDL_DestroyTexture((*texture41)); SDL_DestroyTexture((*texture42));
00658     SDL_DestroyTexture((*texture43)); SDL_DestroyTexture((*texture44));
00659     SDL_DestroyTexture((*texture45)); SDL_DestroyTexture((*texture46));
00660     SDL_DestroyTexture((*texture47)); SDL_DestroyTexture((*texture48));
00661     SDL_DestroyTexture((*texture49)); SDL_DestroyTexture((*texture50));
00662     SDL_DestroyTexture((*texture51)); SDL_DestroyTexture((*texture52));
00663     SDL_DestroyTexture((*texture53)); SDL_DestroyTexture((*texture54));
00664     SDL_DestroyTexture((*texture55)); SDL_DestroyTexture((*texture56));
00665     SDL_DestroyTexture((*texture57)); SDL_DestroyTexture((*texture58));
00666     SDL_DestroyTexture((*texture59)); SDL_DestroyTexture((*texture60));
00667     SDL_DestroyTexture((*texture61)); SDL_DestroyTexture((*texture62));
00668     SDL_DestroyTexture((*texture63)); SDL_DestroyTexture((*texture64));
00669     SDL_DestroyTexture((*texture65)); SDL_DestroyTexture((*texture66));
00670     SDL_DestroyTexture((*texture67)); SDL_DestroyTexture((*texture68));
00671     SDL_DestroyTexture((*texture69)); SDL_DestroyTexture((*texture70));
00672     SDL_DestroyTexture((*texture71)); SDL_DestroyTexture((*texture72));
00673
00674     for(i = 0; i < 11; i++)
00675         SDL_DestroyTexture(textures_images_succes[i]);
00676
00677     /* Destruction de la police */
00678     TTF_CloseFont((*police));
00679
00680     /* Destruction de la musique */
00681     Mix_FreeMusic((*musique));
00682 }
00683
00684 /**
00685  * \fn void detruire_fenetre_rendu(SDL_Renderer **renderer, SDL_Window **window)
00686  * \brief Fonction qui permet de détruire le rendu et la fenêtre
00687  * \param renderer Rendu à détruire
00688  * \param widnow fenêtre à détruire
00689  */
00690 void detruire_fenetre_rendu(SDL_Renderer **renderer, SDL_Window **window) {
00691     SDL_DestroyRenderer((*renderer));
00692     SDL_DestroyWindow((*window));
00693 }
00694 }

```

## 4.7 Référence du fichier fichiers\_c/fonctions\_introduction.c

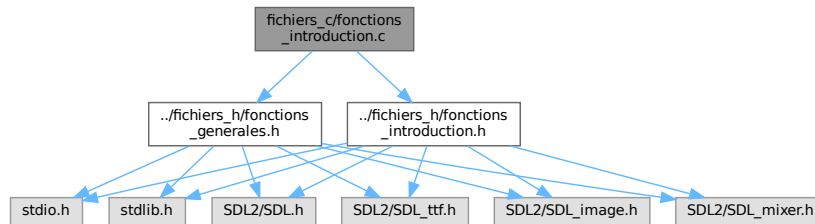
Fichier avec les fonctions présentant l'introduction du jeu.

```

#include <../fichiers_h/fonctions_generales.h>
#include <../fichiers_h/fonctions_introduction.h>

```

Graphe des dépendances par inclusion de fonctions\_introduction.c:



## Fonctions

- void [mise\\_a\\_jour\\_rendu\\_introduction](#) (SDL\_Renderer \*\*renderer, int indice, char \*ligne, SDL\_Rect \*rectangle\_passer, SDL\_Texture \*\*texture\_image\_passer, SDL\_Rect \*rectangle\_texte\_introduction, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurBlanche, int largeur, int hauteur)  
*Fonction qui met à jour le rendu de l'introduction.*
- void [introduction](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_passer, SDL\_Texture \*\*texture\_image\_passer, SDL\_Rect \*rectangle\_texte\_introduction, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, [personnage\\_t](#) \*personnageActif, SDL\_Color couleurBlanche, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active)  
*Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans l'introduction.*

### 4.7.1 Description détaillée

Fichier avec les fonctions présentant l'introduction du jeu.

Définition dans le fichier [fonctions\\_introduction.c](#).

### 4.7.2 Documentation des fonctions

#### 4.7.2.1 introduction()

```

void introduction (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_passer,
    SDL_Texture ** texture_image_passer,
    SDL_Rect * rectangle_texte_introduction,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    personnage\_t * personnageActif,
    SDL_Color couleurBlanche,
    int * largeur,
    int * hauteur,
    page\_t * page_active )
  
```

Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans l'introduction.

## Paramètres

<i>event</i>	Événement SDL.
<i>window</i>	Pointeur vers la fenêtre SDL.
<i>render</i>	Pointeur vers le renderer SDL.
<i>programme_lance</i>	Indicateur de lancement du programme.
<i>rectangle_passer</i>	Rectangle pour le bouton "Passer".
<i>texture_image_passer</i>	Texture de l'image du bouton "Passer".
<i>rectangle_texte_introduction</i>	Rectangle pour le texte d'introduction.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>personnageActif</i>	Personnage actif du joueur.
<i>couleurBlanche</i>	Couleur blanche SDL.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.
<i>page_active</i>	Page active du jeu.

## Voir également

[erreur](#)  
[redimensionnement\\_fenetre](#)  
[clic\\_case](#)  
[mise\\_a\\_jour\\_rendu\\_introduction](#)

Définition à la ligne 103 du fichier [fonctions\\_introduction.c](#).

## 4.7.2.2 mise\_a\_jour\_rendu\_introduction()

```

void mise_a_jour_rendu_introduction (
    SDL_Renderer ** renderer,
    int indice,
    char * ligne,
    SDL_Rect * rectangle_passer,
    SDL_Texture ** texture_image_passer,
    SDL_Rect * rectangle_texte_introduction,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurBlanche,
    int largeur,
    int hauteur )

```

Fonction qui met à jour le rendu de l'introduction.

## Paramètres

<i>render</i>	Pointeur vers le renderer SDL.
<i>indice</i>	Indice de la ligne de texte.
<i>ligne</i>	Ligne de texte à afficher.
<i>rectangle_passer</i>	Rectangle pour le bouton "Passer".
<i>texture_image_passer</i>	Texture de l'image du bouton "Passer".



## Paramètres

<i>rectangle_texte_introduction</i>	Rectangle pour le texte d'introduction.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleurBlanche</i>	Couleur blanche SDL.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

## Voir également

[erreur](#)

Définition à la ligne 25 du fichier [fonctions\\_introduction.c](#).

## 4.8 fonctions\_introduction.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_introduction.c
00003  * \brief Fichier avec les fonctions présentant l'introduction du jeu
00004  */
00005 #include <../fichiers_h/fonctions_generales.h>
00006 #include <../fichiers_h/fonctions_introduction.h>
00007
00008 /**
00009  * \fn void mise_a_jour_rendu_introduction(SDL_Renderer **renderer, int indice, char *ligne, SDL_Rect
00010  *rectangle_passer, SDL_Texture **texture_image_passer, SDL_Rect *rectangle_texte_introduction,
00011  *SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, SDL_Color couleurBlanche, int
00012  *largeur, int hauteur)
00013  * \brief Fonction qui met à jour le rendu de l'introduction
00014  * \param renderer Pointeur vers le renderer SDL.
00015  * \param indice Indice de la ligne de texte.
00016  * \param ligne Ligne de texte à afficher.
00017  * \param rectangle_passer Rectangle pour le bouton "Passer".
00018  * \param texture_image_passer Texture de l'image du bouton "Passer".
00019  * \param rectangle_texte_introduction Rectangle pour le texte d'introduction.
00020  * \param surface Surface SDL.
00021  * \param texture_texte Texture du texte SDL.
00022  * \param police Police de caractères TTF.
00023  * \param couleurBlanche Couleur blanche SDL.
00024  * \param largeur Largeur de l'écran.
00025  * \param hauteur Hauteur de l'écran.
00026  * \see erreur
00027  */
00028 void mise_a_jour_rendu_introduction(SDL_Renderer **renderer, int indice, char *ligne,
00029                                     SDL_Rect *rectangle_passer, SDL_Texture **texture_image_passer,
00030                                     SDL_Rect *rectangle_texte_introduction, SDL_Surface **surface,
00031                                     SDL_Texture **texture_texte,
00032                                     TTF_Font **police, SDL_Color couleurBlanche,
00033                                     int largeur, int hauteur) {
00034
00035     /* Création d'une sous-chaine du texte jusqu'à la lettre actuelle */
00036     char buffer[indice + 2];
00037     strncpy(buffer, ligne, indice);
00038
00039     /* Ajout de la lettre actuelle */
00040     buffer[indice] = ligne[indice];
00041
00042     /* Ajout d'un caractère '\0' pour terminer la chaîne */
00043     buffer[indice + 1] = '\0';
00044
00045     /* Rendu du texte actuel sur la surface texte */
00046     (*surface) = TTF_RenderUTF8_Blended((*police), buffer, couleurBlanche);
00047
00048     /* Création de la texture texture_texte depuis la surface texte */
00049     (*texture_texte) = SDL_CreateTextureFromSurface((*renderer), (*surface));
00050
00051     /* Récupération des dimensions du texte */
00052     int largeur_texte, hauteur_texte;

```

```

00049     SDL_QueryTexture((*texture_texte), NULL, NULL, &largeur_texte, &hauteur_texte);
00050
00051     /* Positionnement du texte au centre */
00052     rectangle_texte_introduction->x = largeur / 2 - largeur_texte / 2;
00053     rectangle_texte_introduction->y = hauteur / 2 - hauteur_texte ;
00054     rectangle_texte_introduction->w = largeur_texte;
00055     rectangle_texte_introduction->h = hauteur_texte;
00056
00057     /* Affichage de la texture texture_texte */
00058     SDL_RenderCopy((*renderer), (*texture_texte), NULL, rectangle_texte_introduction);
00059
00060     SDL_FreeSurface((*surface));
00061     SDL_DestroyTexture((*texture_texte));
00062
00063     /* Copie la texture de l'image du passer */
00064     rectangle_passer->x = largeur - largeur / 21- largeur / 53;
00065     rectangle_passer->y = hauteur / 30;
00066     rectangle_passer->w = largeur / 21;
00067     rectangle_passer->h = hauteur / 12;
00068
00069     if(SDL_RenderCopy((*renderer), (*texture_image_passer), NULL, rectangle_passer) != 0)
00070         erreur("Copie de la texture");
00071
00072     /* Fond noir */
00073     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 0);
00074
00075     /* Affiche le rendu */
00076     SDL_RenderPresent((*renderer));
00077 }
00078
00079 /**
00080  * \fn void introduction(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
00081  * \param programme_lance, SDL_Rect *rectangle_passer, SDL_Texture **texture_image_passer, SDL_Rect
00082  * \param rectangle_texte_introduction, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00083  * \param personnage_t *personnageActif, SDL_Color couleurBlanche, int *largeur, int *hauteur, page_t
00084  * \param page_active)
00085  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans l'introduction
00086
00087  * \param event Événement SDL.
00088  * \param window Pointeur vers la fenêtre SDL.
00089  * \param renderer Pointeur vers le renderer SDL.
00090  * \param programme_lance Indicateur de lancement du programme.
00091  * \param rectangle_passer Rectangle pour le bouton "Passer".
00092  * \param texture_image_passer Texture de l'image du bouton "Passer".
00093  * \param rectangle_texte_introduction Rectangle pour le texte d'introduction.
00094  * \param surface Surface SDL.
00095  * \param texture_texte Texture du texte SDL.
00096  * \param police Police de caractères TTF.
00097  * \param personnageActif Personnage actif du joueur.
00098  * \param couleurBlanche Couleur blanche SDL.
00099  * \param largeur Largeur de l'écran.
00100  * \param hauteur Hauteur de l'écran.
00101  * \param page_active Page active du jeu.
00102  * \see erreur
00103  * \see redimensionnement_fenetre
00104  * \see clic_case
00105  * \see mise_a_jour_rendu_introduction
00106 */
00107 void introduction(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
00108 *programme_lance,
00109                 SDL_Rect *rectangle_passer, SDL_Texture **texture_image_passer,
00110                 SDL_Rect *rectangle_texte_introduction, SDL_Surface **surface, SDL_Texture
00111 **texture_texte, TTF_Font **police,
00112                 personnage_t *personnageActif, SDL_Color couleurBlanche,
00113                 int *largeur, int *hauteur, page_t *page_active) {
00114
00115     /* Ouverture du fichier contenant l'introduction */
00116     FILE *fichier = NULL;
00117
00118     if((*personnageActif) == PERSONNAGE_1) {
00119         fichier = fopen("./textes/introduction_masculin.txt", "r");
00120         if (!fichier)
00121             erreur("Ouverture du fichier introduction_masculin.txt");
00122     }
00123     else {
00124         fichier = fopen("./textes/introduction_feminin.txt", "r");
00125         if (!fichier)
00126             erreur("Ouverture du fichier introduction_feminin.txt");
00127     }
00128
00129     /* Initialisation d'une chaîne de caractères */
00130     char * ligne = malloc(sizeof(char) * 125);
00131     int indice;

```

```

00129
00130     /* Lecture de chaque ligne du fichier */
00131     while((fgets(ligne, sizeof(char) * 125, fichier) != NULL) && ((*page_active) == INTRODUCTION) &&
00132           ((*programme_lance) == SDL_TRUE)) {
00133
00134         /* Boucle pour afficher le texte lettre par lettre */
00135         for (indice = 0; (indice < ((int)strlen(ligne))) && ((*page_active) == INTRODUCTION) &&
00136              ((*programme_lance) == SDL_TRUE); indice++) {
00137
00138             /* Effacement du rendu précédent */
00139             SDL_RenderClear((*renderer));
00140
00141             while(SDL_PollEvent(event)) {
00142
00143                 switch(event->type) {
00144
00145                     case SDL_WINDOWEVENT:
00146
00147                         /* Gestion de l'événement de redimensionnement de la fenêtre */
00148                         redimensionnement_fenetre((*event), largeur, hauteur);
00149
00150                         /* Actualisation de la taille de la police */
00151                         (*police) = TTF_OpenFont("./polices/02587_ARIALMT.ttf", (*largeur) / 50);
00152
00153                         break;
00154
00155                     case SDL_MOUSEBUTTONDOWN:
00156
00157                         /* Appuyer sur le clic gauche de la souris */
00158                         if(event->button.button == SDL_BUTTON_LEFT)
00159                             /* Fin de la ligne */
00160                             indice = strlen(ligne);
00161
00162                         /* Bouton pour passer l'introduction */
00163                         if(clic_case((*event), *rectangle_passer))
00164                             (*page_active) = NIVEAU_1;
00165
00166                         break;
00167
00168                     /* Quitter le programme */
00169                     case SDL_QUIT:
00170                         (*programme_lance) = SDL_FALSE;
00171                         break;
00172
00173                     default:
00174                         break;
00175                 }
00176             }
00177
00178             if((*page_active) == INTRODUCTION) && ((*programme_lance) == SDL_TRUE)) {
00179
00180                 /* Mise à jour du rendu */
00181                 mise_a_jour_rendu_introduction(renderer, indice, ligne,
00182                                                 rectangle_passer, texture_image_passer,
00183                                                 rectangle_texte_introduction, surface, texture_texte,
00184                                                 police, couleurBlanche,
00185                                                 (*largeur), (*hauteur));
00186
00187                 /* Délai entre chaque lettre */
00188                 SDL_Delay(50);
00189             }
00190
00191             if((*page_active) == INTRODUCTION) && ((*programme_lance) == SDL_TRUE)) {
00192
00193                 SDL_SetWindowResizable((*window), SDL_FALSE);
00194
00195                 Mix_PauseMusic();
00196
00197                 /* Délai d'attente avant de passer à la ligne suivante */
00198                 SDL_Delay(2000);
00199
00200                 Mix_ResumeMusic();
00201
00202                 SDL_SetWindowResizable((*window), SDL_TRUE);
00203             }
00204
00205             /* Libération de la mémoire allouée pour la ligne et fermeture du fichier */
00206             free(ligne);
00207             fclose(fichier);
00208 }

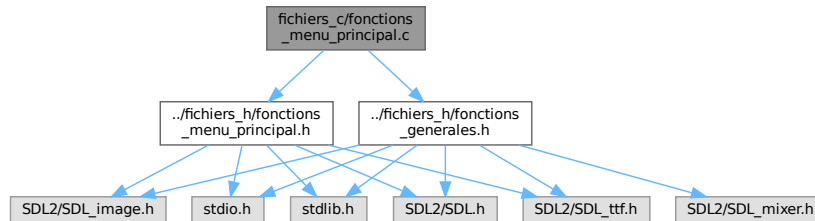
```

## 4.9 Référence du fichier fichiers\_c/fonctions\_menu\_principal.c

Fichier contenant les fonctions pour le menu principal

```
#include <../fichiers_h/fonctions_generales.h>
#include <../fichiers_h/fonctions_menu_principal.h>
```

Graphes des dépendances par inclusion de fonctions\_menu\_principal.c:



### Fonctions

- void [initialisation\\_objets\\_menu\\_principal](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_menu, [itemMenu](#) \*titre, [itemMenu](#) \*itemsMenu, int tailleMenu)  
*Fonction qui permet d'initialiser les différents objets du menu principal.*
- void [mise\\_a\\_jour\\_rendu\\_menu\\_principal](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_menu, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurTitre, SDL\_Color couleurNoire, int selection\_menu, [itemMenu](#) \*itemsMenu, int tailleMenu, int largeur, int hauteur)  
*Fonction qui met à jour le rendu du menu principal.*
- void [menu\\_principal](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_menu, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurTitre, SDL\_Color couleurNoire, int code\_de\_triche[3], int \*selection\_menu, [itemMenu](#) \*itemsMenu, int tailleMenu, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active)  
*Fonction qui permet de gérer toutes les possibilités qui sont possibles dans le menu principal.*

### 4.9.1 Description détaillée

Fichier contenant les fonctions pour le menu principal

Définition dans le fichier [fonctions\\_menu\\_principal.c](#).

## 4.9.2 Documentation des fonctions

### 4.9.2.1 initialisation\_objets\_menu\_principal()

```
void initialisation_objets_menu_principal (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_menu,
    itemMenu * titre,
    itemMenu * itemsMenu,
    int tailleMenu )
```

Fonction qui permet d'initialiser les différents objets du menu principal.

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Surface SDL.
<i>texture_image_menu</i>	Texture de l'image du menu principal.
<i>titre</i>	Titre du menu.
<i>itemsMenu</i>	Tableau d'items pour le menu principal.
<i>tailleMenu</i>	Taille du tableau d'items pour le menu principal.

## Voir également

[chargement\\_image](#)

Définition à la ligne 21 du fichier [fonctions\\_menu\\_principal.c](#).

## 4.9.2.2 menu\_principal()

```
void menu_principal (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_menu,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    itemMenu * titre,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurTitre,
    SDL_Color couleurNoire,
    int code_de_triche[3],
    int * selection_menu,
    itemMenu * itemsMenu,
    int tailleMenu,
    int * largeur,
    int * hauteur,
    page_t * page_active )
```

Fonction qui permet de gérer toutes les possibilités qui sont possible dans le menu principal.

## Paramètres

<i>event</i>	Événement SDL.
<i>window</i>	Pointeur vers la fenêtre SDL.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>programme_lance</i>	Indicateur de lancement du programme.
<i>texture_image_menu</i>	Texture de l'image du menu principal.
<i>rectangle_plein_ecran</i>	Rectangle plein écran SDL.
<i>texture_image_plein_ecran</i>	Texture de l'image en plein écran.
<i>plein_ecran</i>	Indicateur de mode plein écran.
<i>titre</i>	Titre du menu principal.

## Paramètres

<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleurTitre</i>	Couleur du titre.
<i>couleurNoire</i>	Couleur noire SDL.
<i>code_de_triche</i>	Tableau de codes de triche.
<i>selection_menu</i>	Sélection actuelle du menu.
<i>itemsMenu</i>	Tableau d'items pour le menu principal.
<i>tailleMenu</i>	Taille du tableau d'items pour le menu principal.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.
<i>page_active</i>	Page active du menu.

## Voir également

[redimensionnement\\_fenetre](#)  
[clic\\_case](#)  
[clic\\_plein\\_ecran](#)  
[mise\\_a\\_jour\\_rendu\\_menu\\_principal](#)

Définition à la ligne 214 du fichier [fonctions\\_menu\\_principal.c](#).

## 4.9.2.3 mise\_a\_jour\_rendu\_menu\_principal()

```
void mise_a_jour_rendu_menu_principal (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_menu,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    itemMenu * titre,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurTitre,
    SDL_Color couleurNoire,
    int selection_menu,
    itemMenu * itemsMenu,
    int tailleMenu,
    int largeur,
    int hauteur )
```

Fonction qui met à jour le rendu du menu principal.

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>texture_image_menu</i>	Texture de l'image du menu principal.
<i>rectangle_plein_ecran</i>	Rectangle plein écran SDL.
<i>texture_image_plein_ecran</i>	Texture de l'image en plein écran.
<i>titre</i>	Titre du menu principal.

## Paramètres

<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleurTitre</i>	Couleur du titre.
<i>couleurNoire</i>	Couleur noire SDL.
<i>selection_menu</i>	Sélection actuelle du menu.
<i>itemsMenu</i>	Tableau d'items pour le menu principal.
<i>tailleMenu</i>	Taille du tableau d'items pour le menu principal.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

## Voir également

[erreur](#)  
[affichage\\_texte](#)

Définition à la ligne 84 du fichier [fonctions\\_menu\\_principal.c](#).

## 4.10 fonctions\_menu\_principal.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_menu_principal.c
00003  * \brief Fichier contenant les fonctions pour le menu principal
00004  */
00005
00006 #include <../fichiers_h/fonctions_generales.h>
00007 #include <../fichiers_h/fonctions_menu_principal.h>
00008
00009 /**
00010  * \fn void initialisation_objets_menu_principal(SDL_Renderer **renderer, SDL_Surface **surface,
00011  * \brief Fonction qui permet d'initialiser les différents objets du menu principal
00012  * \param renderer Pointeur vers le renderer SDL.
00013  * \param surface Surface SDL.
00014  * \param texture_image_menu Texture de l'image du menu principal.
00015  * \param titre Titre du menu.
00016  * \param itemsMenu Tableau d'items pour le menu principal.
00017  * \param tailleMenu Taille du tableau d'items pour le menu principal.
00018  * \see chargement_image
00019  */
00020 */
00021 void initialisation_objets_menu_principal(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00022 **texture_image_menu,
00023                                     itemMenu *titre, itemMenu *itemsMenu, int tailleMenu) {
00024     int i;
00025
00026     /* Initialisation de l'image de fond du menu */
00027     chargement_image(renderer, surface, texture_image_menu, "./images/ecran_accueil.png");
00028
00029     /* Initialisation du titre du menu */
00030     sprintf(titre->texte, " MetaTravers ");
00031
00032     /* Initialisation du texte dans les items du menu */
00033
00034     if(tailleMenu == 2) {
00035         for(i = 0; i < tailleMenu; i++) {
00036             if(!i)
00037                 sprintf(itemsMenu[i].texte, " Nouvelle Partie ");
00038             else
00039                 sprintf(itemsMenu[i].texte, " Options ");
00040         }
00041     }
00042 }
00043

```



```

00044     }
00045
00046     else {
00047
00048         for(i = 0; i < tailleMenu; i++) {
00049
00050             if(!i)
00051                 sprintf(itemsMenu[i].texte, " Continuer ");
00052
00053             else if(i == 1)
00054                 sprintf(itemsMenu[i].texte, " Nouvelle Partie ");
00055
00056             else
00057                 sprintf(itemsMenu[i].texte, " Options ");
00058         }
00059     }
00060 }
00061
00062 /**
00063  * \fn void mise_a_jour_rendu_menu_principal(SDL_Renderer **renderer, SDL_Texture
00064  **texture_image_menu, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
00065  itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, SDL_Color
00066  couleurTitre, SDL_Color couleurNoire, int selection_menu, itemMenu *itemsMenu, int tailleMenu, int
00067  largeur, int hauteur)
00068  * \brief Fonction qui met à jour le rendu du menu principal
00069  * \param renderer Pointeur vers le renderer SDL.
00070  * \param texture_image_menu Texture de l'image du menu principal.
00071  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00072  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00073  * \param titre Titre du menu principal.
00074  * \param surface Surface SDL.
00075  * \param texture_texte Texture du texte SDL.
00076  * \param police Police de caractères TTF.
00077  * \param couleurTitre Couleur du titre.
00078  * \param couleurNoire Couleur noire SDL.
00079  * \param selection_menu Sélection actuelle du menu.
00080  * \param itemsMenu Tableau d'items pour le menu principal.
00081  * \param tailleMenu Taille du tableau d'items pour le menu principal.
00082  * \param largeur Largeur de l'écran.
00083  * \param hauteur Hauteur de l'écran.
00084  * \see erreur
00085  * \see affichage_texte
00086  */
00087 void mise_a_jour_rendu_menu_principal(SDL_Renderer **renderer, SDL_Texture **texture_image_menu,
00088 SDL_Rect *rectangle_plein_ecran, SDL_Texture
00089 **texture_image_plein_ecran,
00090 itemMenu *titre, SDL_Surface **surface, SDL_Texture
00091 **texture_texte, TTF_Font **police,
00092 SDL_Color couleurTitre, SDL_Color couleurNoire, int
00093 selection_menu,
00094 itemMenu *itemsMenu, int tailleMenu, int largeur, int hauteur) {
00095
00096     int i;
00097
00098     /* Efface le rendu */
00099     if(SDL_RenderClear((*renderer)) != 0)
00100         erreur("Effacement rendu échoué");
00101
00102     /* Utilisation de la fusion pour un rendu avec transparence */
00103     SDL_SetRenderDrawBlendMode((*renderer), SDL_BLENDMODE_BLEND);
00104
00105     /* Copie la texture de l'image de fond du menu */
00106     if(SDL_RenderCopy((*renderer), (*texture_image_menu), NULL, NULL) != 0)
00107         erreur("Copie de la texture");
00108
00109     /* Copie la texture de l'image de plein écran */
00110
00111     rectangle_plein_ecran->x = largeur - largeur / 21 - largeur / 53;
00112     rectangle_plein_ecran->y = hauteur / 30;
00113     rectangle_plein_ecran->w = largeur / 21;
00114     rectangle_plein_ecran->h = hauteur / 12;
00115
00116     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00117         erreur("Copie de la texture");
00118
00119     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 175);
00120
00121     /* Dessine le titre du menu */
00122
00123     titre->rectangle.x = largeur / 2 - largeur / 3;
00124     titre->rectangle.y = hauteur / 20;
00125     titre->rectangle.w = largeur / 2 + largeur / 5;
00126     titre->rectangle.h = hauteur / 5;
00127
00128     affichage_texte(renderer, surface, texture_texte, titre,
00129 police, couleurTitre);

```

```

00124
00125     /* Dessine les éléments du menu */
00126
00127     if(tailleMenu == 2) {
00128
00129         for (i = 0; i < tailleMenu; i++) {
00130
00131             if(selection_menu == (i + 1)) {
00132
00133                 itemsMenu[i].rectangle.x = largeur / 2 - largeur / 6 - largeur / 200;
00134                 itemsMenu[i].rectangle.y = hauteur / 3 + (i+1) * hauteur / 13 + i * hauteur / 11 -
hauteur / 90;
00135                 itemsMenu[i].rectangle.w = largeur / 3 + largeur / 100;
00136                 itemsMenu[i].rectangle.h = hauteur / 10 + hauteur / 60 + hauteur / 250;
00137
00138                 SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00139                 SDL_RenderFillRect((*renderer), &(itemsMenu[i].rectangle));
00140             }
00141
00142             SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00143
00144             itemsMenu[i].rectangle.x = largeur / 2 - largeur / 6;
00145             itemsMenu[i].rectangle.y = hauteur / 3 + (i+1) * hauteur / 13 + i * hauteur / 11;
00146             itemsMenu[i].rectangle.w = largeur / 3;
00147             itemsMenu[i].rectangle.h = hauteur / 10;
00148
00149             affichage_texte(renderer, surface, texture_texte, &(itemsMenu[i]),
00150                             police, couleurNoire);
00151         }
00152     }
00153     else {
00154
00155         for (i = 0; i < tailleMenu; i++) {
00156
00157             if(selection_menu == (i + 1)) {
00158
00159                 itemsMenu[i].rectangle.x = largeur / 2 - largeur / 6 - largeur / 200;
00160                 itemsMenu[i].rectangle.y = hauteur / 3 + i * hauteur / 6 - hauteur / 90;
00161                 itemsMenu[i].rectangle.w = largeur / 3 + largeur / 100;
00162                 itemsMenu[i].rectangle.h = hauteur / 10 + hauteur / 60 + hauteur / 250;
00163
00164                 SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00165                 SDL_RenderFillRect((*renderer), &(itemsMenu[i].rectangle));
00166             }
00167
00168             SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00169
00170             itemsMenu[i].rectangle.x = largeur / 2 - largeur / 6;
00171             itemsMenu[i].rectangle.y = hauteur / 3 + i * hauteur / 6;
00172             itemsMenu[i].rectangle.w = largeur / 3;
00173             itemsMenu[i].rectangle.h = hauteur / 10;
00174
00175             affichage_texte(renderer, surface, texture_texte, &(itemsMenu[i]),
00176                             police, couleurNoire);
00177         }
00178     }
00179 }
00180
00181 /* Affiche le rendu */
00182 SDL_RenderPresent((*renderer));
00183 }
00184
00185 /**
00186  * \fn void menu_principal(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance, SDL_Texture **texture_image_menu, SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran, SDL_bool *plein_ecran, itemMenu *titre, SDL_Surface **surface,
SDL_Texture **texture_texte, TTF_Font **police, SDL_Color couleurTitre, SDL_Color couleurNoire, int
code_de_triche[3], int *selection_menu, itemMenu *itemsMenu, int tailleMenu, int *largeur, int
*hauteur, page_t *page_active)
00187  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le menu
principal
00188  * \param event Événement SDL.
00189  * \param window Pointeur vers la fenêtre SDL.
00190  * \param renderer Pointeur vers le renderer SDL.
00191  * \param programme_lance Indicateur de lancement du programme.
00192  * \param texture_image_menu Texture de l'image du menu principal.
00193  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00194  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00195  * \param plein_ecran Indicateur de mode plein écran.
00196  * \param titre Titre du menu principal.
00197  * \param surface Surface SDL.
00198  * \param texture_texte Texture du texte SDL.
00199  * \param police Police de caractères TTF.
00200  * \param couleurTitre Couleur du titre.
00201  * \param couleurNoire Couleur noire SDL.
00202  * \param code_de_triche Tableau de codes de triche.
00203  * \param selection_menu Sélection actuelle du menu.

```

```

00204 * \param itemsMenu Tableau d'items pour le menu principal.
00205 * \param tailleMenu Taille du tableau d'items pour le menu principal.
00206 * \param largeur Largeur de l'écran.
00207 * \param hauteur Hauteur de l'écran.
00208 * \param page_active Page active du menu.
00209 * \see redimensionnement_fenetre
00210 * \see clic_case
00211 * \see clic_plein_ecran
00212 * \see mise_a_jour_rendu_menu_principal
00213 */
00214 void menu_principal(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance, SDL_Texture **texture_image_menu,
00215 SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
*plein_ecran,
00216 itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font
**police,
00217 SDL_Color couleurTitre, SDL_Color couleurNoire, int code_de_triche[3], int
*selection_menu,
00218 itemMenu *itemsMenu, int tailleMenu, int *largeur, int *hauteur, page_t
*page_active) {
00219     while(SDL_PollEvent(event)) {
00220         while(SDL_PollEvent(event)) {
00221             switch(event->type) {
00222                 /* Gestion de l'événement de redimensionnement de la fenêtre */
00223                 case SDL_WINDOWEVENT:
00224                     redimensionnement_fenetre((*event), largeur, hauteur);
00225                     break;
00226                 /* Gestion de l'événement de la position de la souris sur les différents items */
00227                 case SDL_MOUSEMOTION:
00228                     if((event->motion.x >= itemsMenu[0].rectangle.x) && (event->motion.x <=
(itemsMenu[0].rectangle.x + itemsMenu[0].rectangle.w)) &&
00229                     (event->motion.y >= itemsMenu[0].rectangle.y) && (event->motion.y <=
(itemsMenu[0].rectangle.y + itemsMenu[0].rectangle.h)))
00230                     (*selection_menu) = 1;
00231                     else if((event->motion.x >= itemsMenu[1].rectangle.x) && (event->motion.x <=
(itemsMenu[1].rectangle.x + itemsMenu[1].rectangle.w)) &&
00232                     (event->motion.y >= itemsMenu[1].rectangle.y) && (event->motion.y <=
(itemsMenu[1].rectangle.y + itemsMenu[1].rectangle.h)))
00233                     (*selection_menu) = 2;
00234                     else if((event->motion.x >= itemsMenu[2].rectangle.x) && (event->motion.x <=
(itemsMenu[2].rectangle.x + itemsMenu[2].rectangle.w)) &&
00235                     (event->motion.y >= itemsMenu[2].rectangle.y) && (event->motion.y <=
(itemsMenu[2].rectangle.y + itemsMenu[2].rectangle.h)))
00236                     (*selection_menu) = 3;
00237                     else
00238                         (*selection_menu) = 0;
00239                     break;
00240                 /* Gestion de l'événement du clic sur les différents items */
00241                 case SDL_MOUSEBUTTONDOWN:
00242                     if(tailleMenu == 2) {
00243                         if(clic_case((*event), itemsMenu[0].rectangle))
00244                             (*page_active) = NOUVELLE_PARTIE;
00245                         else if(clic_case((*event), itemsMenu[1].rectangle))
00246                             (*page_active) = OPTIONS;
00247                     }
00248                     else {
00249                         if(clic_case((*event), itemsMenu[0].rectangle))
00250                             (*page_active) = CARTE;
00251                         else if(clic_case((*event), itemsMenu[1].rectangle))
00252                             (*page_active) = NOUVELLE_PARTIE;
00253                         else if(clic_case((*event), itemsMenu[2].rectangle))
00254                             (*page_active) = OPTIONS;
00255                     }
00256                     /* Option plein écran */
00257                     if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window))
00258                         redimensionnement_fenetre((*event), largeur, hauteur);
00259                     break;

```

```

00280
00281      /* Gestion de l'événement du code de triche */
00282      case SDL_KEYDOWN:
00283
00284          if((event->key.keysym.sym == SDLK_g) && (code_de_triche[1]))
00285              code_de_triche[0] = 1;
00286
00287          if(event->key.keysym.sym == SDLK_b)
00288              code_de_triche[1] = 1;
00289
00290          if((event->key.keysym.sym == SDLK_n) && (code_de_triche[0]))
00291              code_de_triche[2] = 1;
00292
00293          break;
00294
00295      /* Quitter le programme */
00296      case SDL_QUIT:
00297          (*programme_lance) = SDL_FALSE;
00298          break;
00299
00300      default:
00301          break;
00302    }
00303 }
00304
00305 /* Mise à jour du rendu */
00306 mise_a_jour_rendu_menu_principal(renderer, texture_image_menu,
00307     rectangle_plein_ecran, texture_image_plein_ecran,
00308     titre, surface, texture_texte, police,
00309     couleurTitre, couleurNoire, (*selection_menu),
00310     itemsMenu, tailleMenu, (*largeur), (*hauteur));
00311 }

```

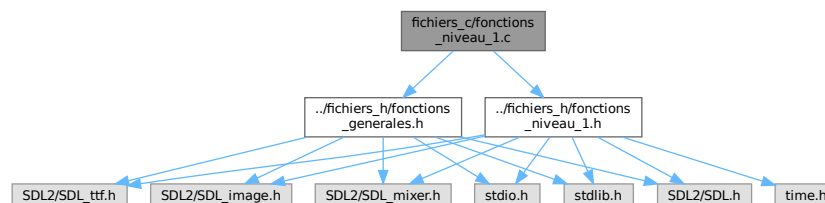
## 4.11 Référence du fichier fichiers\_c/fonctions\_niveau\_1.c

Fichier contenant les fonctions servant à la gestion du niveau 1

```
#include <../fichiers_h/fonctions_generales.h>
```

```
#include <../fichiers_h/fonctions_niveau_1.h>
```

Graphe des dépendances par inclusion de fonctions\_niveau\_1.c:



### Fonctions

- void **initialisation\_objets\_niveau\_1** (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_sol\_surface\_niveau\_1, SDL\_Texture \*\*texture\_image\_sol\_profondeur\_niveau\_1, SDL\_Texture \*\*texture\_image\_fond\_niveau\_1, SDL\_Texture \*\*texture\_image\_nuage\_1, SDL\_Texture \*\*texture\_image\_nuage\_2)

*Fonction qui permet d'initialiser les différents objets du niveau 4.*

- void **chargement\_niveau\_1** (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map\_niveau\_1[18][110])

*Fonction qui permet de créer l'étage 1.*

- void [mise\\_a\\_jour\\_rendu\\_niveau\\_1](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_sol\_surface, SDL\_Texture \*\*texture\_image\_sol\_profondeur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, int position\_x, int position\_y, int tile\_map[18][32], int secret, SDL\_Texture \*\*texture\_image\_nuage\_1, SDL\_Texture \*\*texture\_image\_nuage\_2, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix)
- void [niveau\\_1](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image\_sol\_surface, SDL\_Texture \*\*texture\_image\_sol\_profondeur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, int \*mouvement\_monstre, SDL\_Surface \*\*surface, int collectibles\_intermediaires[3], time\_t \*timestamp, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, int \*decalage, int \*secret\_1, int \*secret\_2, int tile\_map[18][32], int tile\_map\_niveau\_1[18][110], SDL\_Rect \*rectangle\_tile, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemande, SDL\_Texture \*\*texture\_image\_perso\_gagnant, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Rect \*rectangle\_demande, SDL\_Texture \*\*texture\_image\_nuage\_1, SDL\_Texture \*\*texture\_image\_nuage\_2, SDL\_Color couleurNoire, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, int \*avancer, int \*reculer, int \*sauter, int \*position\_avant\_saut, int \*saut, int \*tombe, int \*position\_x\_initiale, int \*position\_y\_initiale, int \*position\_x, int \*position\_y, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, [page\\_t](#) \*page\_active, [itemMenu](#) \*itemsDemandeSauvegarde, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_interagir, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [modes\\_t](#) \*modeActif, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

### 4.11.1 Description détaillée

Fichier contenant les fonctions servant à la gestion du niveau 1

Définition dans le fichier [fonctions\\_niveau\\_1.c](#).

### 4.11.2 Documentation des fonctions

#### 4.11.2.1 chargement\_niveau\_1()

```
void chargement_niveau_1 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map_niveau_1[18][110] )
```

Fonction qui permet de créer l'étage 1.

#### Paramètres

<i>position_x</i>	Position du personnage à l'apparition sur la verticale
<i>position_y</i>	Position du personnage à l'apparition sur l'horizontale
<i>position_x_initiale</i>	Position initiale verticale du niveau en cas de mort du personnage
<i>position_y_initiale</i>	Position initiale horizontale du niveau en cas de mort du personnage
<i>tile_map_niveau_1</i>	Map du niveau 1

Définition à la ligne 43 du fichier [fonctions\\_niveau\\_1.c](#).

#### 4.11.2.2 initialisation\_objets\_niveau\_1()

```
void initialisation_objets_niveau_1 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_sol_surface_niveau_1,
    SDL_Texture ** texture_image_sol_profondeur_niveau_1,
    SDL_Texture ** texture_image_fond_niveau_1,
    SDL_Texture ** texture_image_nuage_1,
    SDL_Texture ** texture_image_nuage_2 )
```

Fonction qui permet d'initialiser les différents objets du niveau 4.

##### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Surface SDL.
<i>texture_image_sol_surface_niveau_1</i>	Texture de l'image du sol de surface du niveau 1.
<i>texture_image_sol_profondeur_niveau_1</i>	Texture de l'image du sol de profondeur du niveau 1.
<i>texture_image_fond_niveau_1</i>	Texture de l'image de fond du niveau 1.
<i>texture_image_nuage_1</i>	Texture de l'image du nuage 1.
<i>texture_image_nuage_2</i>	Texture de l'image du nuage 2.

##### Voir également

[chargement\\_image](#)

Définition à la ligne 20 du fichier [fonctions\\_niveau\\_1.c](#).

#### 4.11.2.3 mise\_a\_jour\_rendu\_niveau\_1()

```
void mise_a_jour_rendu_niveau_1 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_sol_surface,
    SDL_Texture ** texture_image_sol_profondeur,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    SDL_Texture ** texture_image_pique,
    niveaux * avancee_niveaux,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    int position_x,
    int position_y,
```

```

int tile_map[18][32],
int secret,
SDL_Texture ** texture_image_nuage_1,
SDL_Texture ** texture_image_nuage_2,
int largeur,
int hauteur,
int largeur_tile,
int hauteur_tile,
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix )

```

Définition à la ligne 114 du fichier [fonctions\\_niveau\\_1.c](#).

#### 4.11.2.4 niveau\_1()

```

void niveau_1 (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    SDL_Texture ** texture_image_sol_surface,
    SDL_Texture ** texture_image_sol_profondeur,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_pique,
    niveaux * avancee_niveaux,
    int * mouvement_monstre,
    SDL_Surface ** surface,
    int collectibles_intermediaires[3],
    time_t * timestamp,
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_sauter_monter,
    int * decalage,
    int * secret_1,
    int * secret_2,
    int tile_map[18][32],
    int tile_map_niveau_1[18][110],
    SDL_Rect * rectangle_tile,
    itemMenu * itemsDemandeQuitter,
    int tailleDemande,
    SDL_Texture ** texture_image_perso_gagnant,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Rect * rectangle_demande,
    SDL_Texture ** texture_image_nuage_1,
    SDL_Texture ** texture_image_nuage_2,
    SDL_Color couleurNoire,

```

```

SDL_Texture ** texture_image_fin_premiers_niveaux,
int * avancer,
int * reculer,
int * sauter,
int * position_avant_saut,
int * saut,
int * tombe,
int * position_x_initiale,
int * position_y_initiale,
int * position_x,
int * position_y,
int * largeur,
int * hauteur,
int * largeur_tile,
int * hauteur_tile,
page_t * page_active,
itemMenu * itemsDemandeSauvegarde,
SDL_Keycode * touche_descendre,
SDL_Keycode * touche_interagir,
barreDeSon * barre_de_son,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 300 du fichier [fonctions\\_niveau\\_1.c](#).

## 4.12 fonctions\_niveau\_1.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_niveau_1.c
00003  * \brief Fichier contenant les fonctions servant à la gestion du niveau 1
00004  */
00005 #include <../fichiers_h/fonctions_generales.h>
00006 #include <../fichiers_h/fonctions_niveau_1.h>
00007
00008 /**
00009  * \fn void initialisation_objets_niveau_1(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_sol_surface_niveau_1, SDL_Texture **texture_image_sol_profondeur_niveau_1, SDL_Texture
**texture_image_fond_niveau_1, SDL_Texture **texture_image_nuage_1, SDL_Texture
**texture_image_nuage_2)
00010  * \brief Fonction qui permet d'initialiser les différents objets du niveau 4
00011  * \param renderer Pointeur vers le renderer SDL.
00012  * \param surface Surface SDL.
00013  * \param texture_image_sol_surface_niveau_1 Texture de l'image du sol de surface du niveau 1.
00014  * \param texture_image_sol_profondeur_niveau_1 Texture de l'image du sol de profondeur du niveau 1.
00015  * \param texture_image_fond_niveau_1 Texture de l'image de fond du niveau 1.
00016  * \param texture_image_nuage_1 Texture de l'image du nuage 1.
00017  * \param texture_image_nuage_2 Texture de l'image du nuage 2.
00018  * \see chargement_image
00019  */
00020 void initialisation_objets_niveau_1(SDL_Renderer **renderer, SDL_Surface **surface,
00021                                     SDL_Texture **texture_image_sol_surface_niveau_1, SDL_Texture
**texture_image_sol_profondeur_niveau_1,
00022                                     SDL_Texture **texture_image_fond_niveau_1, SDL_Texture
**texture_image_nuage_1, SDL_Texture **texture_image_nuage_2) {
00023
00024     /* Chargement des images pour le niveau 1 */
00025
00026     chargement_image(renderer, surface, texture_image_sol_surface_niveau_1,
".images/niveau_1/sol_niveau_1_surface.jpg");

```



[illegible]

```

0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 10, 10, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 10, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2},
00071 {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 10, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 10, 10, 10, 10, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2},
00072 {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 1, 10, 10, 0,
0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 10, 10, 12, 12, 12, 12, 11, 0, 0, 0, 0, 0, 0,
0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 7, 0, 2},
00073 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 10, 10, 10, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 10, 10, 10, 10, 10, 10, 10, 10, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
00074 {10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 0, 0, 10, 10, 10, 10, 10, 10, 10, 10, 0, 0, 10, 10, 10, 10, 0, 0, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10},
00075
00076 };
00077
00078 /* Copie du niveau 1 */
00079 for (y = 0; y < 18; y++)
00080     for (x = 0; x < 110; x++)
00081         tile_map_niveau_1[y][x] = initialisation_tile_map[y][x];
00082 }
00083
00084 /**
00085  * \fn void mise_a_jour_rendu_niveau_1(SDL_Renderer **renderer, SDL_Texture
**texture_image_sol_surface, SDL_Texture **texture_image_sol_profondeur, SDL_Texture
**texture_image_fond, SDL_Texture **texture_image_monstre_terrestre, SDL_Rect *rectangle_tile, SDL_Rect
*rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_Texture
**texture_image_personnage, SDL_Rect *rectangle_personnage, SDL_Texture
**texture_image_monstre_volant, SDL_Texture
**texture_image_pique, niveaux *avancee_niveaux, SDL_Texture **texture_image_fin_premiers_niveaux, int
position_x, int position_y, int tile_map[18][32], int secret, SDL_Texture **texture_image_nuage_1,
SDL_Texture **texture_image_nuage_2, int largeur, int hauteur, int largeur_tile, int hauteur_tile)
00086  * \brief Fonction qui permet de mettre à jour le rendu du niveau 4
00087  * \param renderer Pointeur vers le renderer SDL.
00088  * \param texture_image_sol_surface Texture de l'image du sol de surface.
00089  * \param texture_image_sol_profondeur Texture de l'image du sol de profondeur.
00090  * \param texture_image_fond Texture de l'image de fond.
00091  * \param texture Texture SDL.
00092  * \param rectangle_tile Rectangle pour chaque tuile.
00093  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00094  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00095  * \param texture_image_personnage Texture de l'image du personnage.
00096  * \param rectangle_personnage Rectangle du personnage.
00097  * \param texture_image_monstre_terrestre Texture de l'image du monstre terrestre.
00098  * \param texture_image_monstre_volant Texture de l'image du monstre volant.
00099  * \param texture_image_pique Texture de l'image du piège.
00100  * \param avancee_niveaux Structure de progression des niveaux.
00101  * \param texture_image_fin_premiers_niveaux Texture de l'image de fin des premiers niveaux.
00102  * \param position_x Position en x du joueur.
00103  * \param position_y Position en y du joueur.
00104  * \param tile_map Carte du niveau.
00105  * \param secret Indicateur de secret découvert.
00106  * \param texture_image_nuage_1 Texture de l'image du nuage 1.
00107  * \param texture_image_nuage_2 Texture de l'image du nuage 2.
00108  * \param largeur Largeur de l'écran.
00109  * \param hauteur Hauteur de l'écran.
00110  * \param largeur_tile Largeur d'une tuile.
00111  * \param hauteur_tile Hauteur d'une tuile.
00112  * \see erreur
00113  */
00114 void mise_a_jour_rendu_niveau_1(SDL_Renderer **renderer, SDL_Texture **texture_image_sol_surface,
SDL_Texture **texture_image_sol_profondeur, SDL_Texture **texture_image_fond,
SDL_Texture **texture_image_monstre_terrestre, SDL_Rect *rectangle_tile, SDL_Rect
*rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
SDL_Texture **texture_image_personnage, SDL_Rect
*rectangle_personnage, SDL_Texture **texture_image_monstre_volant,
SDL_Texture **texture_image_pique, niveaux *avancee_niveaux,
SDL_Texture **texture_image_fin_premiers_niveaux,
00118     int position_x, int position_y, int tile_map[18][32], int secret,
SDL_Texture **texture_image_nuage_1, SDL_Texture **texture_image_nuage_2,
00119     int largeur, int hauteur, int largeur_tile, int hauteur_tile,
SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix) {
00120
00121     int x, y;
00122
00123     /* Efface le rendu */
00124     if(SDL_RenderClear((*renderer)) != 0)
00125         erreur("Effacement rendu échoué");
00126

```

```

00127     /* Affiche une partie du niveau 1 en fonction des valeurs */
00128     for (y = 0; y < hauteur / hauteur_tile; y++) {
00129
00130         for (x = 0; x < largeur / largeur_tile; x++) {
00131
00132             if((tile_map[y][x] == 1) || ((tile_map[y][x] == 11) && (!secret)))
00133                 (*texture) = (*texture_image_sol_surface);
00134
00135             else if((!(tile_map[y][x])) || (tile_map[y][x] == 2) || (tile_map[y][x] == 3) ||
00136 (tile_map[y][x] == 5) || (tile_map[y][x] == 7) || (tile_map[y][x] == 8) ||
00137 ((tile_map[y][x] == 11) || (tile_map[y][x] == 12)) && (secret)))
00138                 (*texture) = (*texture_image_fond);
00139
00140             else if((tile_map[y][x] == 10) || (tile_map[y][x] == 12))
00141                 (*texture) = (*texture_image_sol_profondeur);
00142
00143             rectangle_tile->x = x * largeur_tile;
00144             rectangle_tile->y = y * hauteur_tile;
00145             rectangle_tile->w = largeur_tile;
00146             rectangle_tile->h = hauteur_tile;
00147
00148             if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile) != 0)
00149                 erreur("Copie de la texture");
00150
00151             if(tile_map[y][x] == 3)
00152                 if(SDL_RenderCopy((*renderer), (*texture_image_pique), NULL, rectangle_tile) != 0)
00153                     erreur("Copie de la texture");
00154
00155             if((tile_map[y][x] == 7) && ((position_x != 29) || (position_y != 15)))
00156                 if(SDL_RenderCopy((*renderer), (*texture_image_fin_premiers_niveaux), NULL,
00157 rectangle_tile) != 0)
00158                     erreur("Copie de la texture");
00159
00160             if(tile_map[y][x] == 8)
00161                 if(SDL_RenderCopy((*renderer), (*texture_image_monstre_terrestre), NULL,
00162 rectangle_tile) != 0)
00163                     erreur("Copie de la texture");
00164
00165             if(tile_map[y][x] == 9)
00166                 if(SDL_RenderCopy((*renderer), (*texture_image_monstre_volant), NULL, rectangle_tile)
00167 != 0)
00168                     erreur("Copie de la texture");
00169
00170             if((tile_map[y][x] == 5) && (y == 8) && (avancee_niveaux[0].numero_collectible[0] == 0))
00171                 if(SDL_RenderCopy((*renderer), avancee_niveaux[0].texture_image_collectible, NULL,
00172 rectangle_tile) != 0)
00173                     erreur("Copie de la texture");
00174
00175             if((tile_map[y][x] == 5) && (y == 15) && (avancee_niveaux[0].numero_collectible[1] == 0))
00176                 if(SDL_RenderCopy((*renderer), avancee_niveaux[0].texture_image_collectible, NULL,
00177 rectangle_tile) != 0)
00178                     erreur("Copie de la texture");
00179
00180             if((tile_map[y][x] == 5) && (y == 3) && (avancee_niveaux[0].numero_collectible[2] == 0))
00181                 if(SDL_RenderCopy((*renderer), avancee_niveaux[0].texture_image_collectible, NULL,
00182 rectangle_tile) != 0)
00183                     erreur("Copie de la texture");
00184
00185             }
00186         }
00187     }
00188
00189     for (y = 0; y < hauteur / hauteur_tile; y++) {
00190
00191         for (x = 0; x < largeur / largeur_tile; x++) {
00192
00193             rectangle_tile->x = (x - 2) * largeur_tile;
00194             rectangle_tile->y = y * hauteur_tile;
00195             rectangle_tile->w = largeur_tile * 4;
00196             rectangle_tile->h = hauteur_tile * 2;
00197
00198             if(tile_map[y][x] == 13)
00199                 if(SDL_RenderCopy((*renderer), (*texture_image_nuage_1), NULL, rectangle_tile) != 0)
00200                     erreur("Copie de la texture");
00201
00202             if(tile_map[y][x] == 14)
00203                 if(SDL_RenderCopy((*renderer), (*texture_image_nuage_2), NULL, rectangle_tile) != 0)
00204                     erreur("Copie de la texture");
00205
00206         }
00207     }
00208
00209     /* Copie la texture de l'image du personnage */
00210
00211     rectangle_personnage->x = position_x * largeur_tile;
00212     rectangle_personnage->y = position_y * hauteur_tile;
00213     rectangle_personnage->w = largeur_tile;
00214     rectangle_personnage->h = hauteur_tile;
00215
00216     if(SDL_RenderCopy((*renderer), (*texture_image_personnage), NULL, rectangle_personnage) != 0)

```

```

00207         erreur("Copie de la texture");
00208
00209     /* Copie la texture de l'image de plein écran */
00210
00211     rectangle_plein_ecran->x = largeur_tile * 31;
00212     rectangle_plein_ecran->y = 0;
00213     rectangle_plein_ecran->w = largeur_tile;
00214     rectangle_plein_ecran->h = hauteur_tile;
00215
00216     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00217         erreur("Copie de la texture");
00218
00219     /* Copie la texture de l'image de la croix */
00220
00221     rectangle_croix->x = 0;
00222     rectangle_croix->y = 0;
00223     rectangle_croix->w = largeur_tile;
00224     rectangle_croix->h = hauteur_tile;
00225
00226     if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00227         erreur("Copie de la texture");
00228
00229     /* Affiche le rendu */
00230     SDL_RenderPresent((*renderer));
00231 }
00232
00233 /**
00234  * \fn void niveau_1(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_Texture
00235  **texture, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
00236  *plein_ecran, SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, SDL_Texture
00237  **texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant, SDL_Texture
00238  **texture_image_sol_surface, SDL_Texture **texture_image_sol_profondeur, SDL_Texture
00239  **texture_image_fond, SDL_Texture **texture_image_pique, niveaux *avancee_niveaux, int
00240  *mouvement_monstre, SDL_Surface **surface, int collectibles_intermediaires[3], time_t *timestamp,
00241  SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode
00242  *touche_sauter_monter, int *decalage, int *secret_1, int *secret_2, int tile_map[18][32], int
00243  tile_map_niveau_1[18][110], SDL_Rect *rectangle_tile, itemMenu *itemsDemandeQuitter, int
00244  tailleDemandeQuitter, SDL_Texture **texture_image_perso_gagnant, SDL_Texture **texture_texte, TTF_Font
00245  **police, SDL_Rect *rectangle_demande_quitter, SDL_Texture **texture_image_nuage_1, SDL_Texture
00246  **texture_image_nuage_2, SDL_Color couleurNoire, SDL_Texture **texture_image_fin_premiers_niveaux, int
00247  *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut, int *tombe, int
00248  *position_x_initiale, int *position_y_initiale, int *position_x, int *position_y, int *largeur, int
00249  *hauteur, int *largeur_tile, int *hauteur_tile, page_t *page_active)
00250  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le niveau 4
00251  * \param event Événement SDL.
00252  * \param window Pointeur vers la fenêtre SDL.
00253  * \param renderer Pointeur vers le renderer SDL.
00254  * \param texture Texture SDL.
00255  * \param rectangle_plein_ecran Rectangle plein écran SDL.
00256  * \param texture_image_plein_ecran Texture de l'image en plein écran.
00257  * \param plein_ecran Indicateur de mode plein écran.
00258  * \param texture_image_personnage Texture de l'image du personnage.
00259  * \param rectangle_personnage Rectangle du personnage.
00260  * \param texture_image_monstre_terrestre Texture de l'image du monstre terrestre.
00261  * \param texture_image_monstre_volant Texture de l'image du monstre volant.
00262  * \param texture_image_sol_surface Texture de l'image du sol de surface.
00263  * \param texture_image_sol_profondeur Texture de l'image du sol de profondeur.
00264  * \param texture_image_fond Texture de l'image de fond.
00265  * \param texture_image_pique Texture de l'image du piège.
00266  * \param avancee_niveaux Structure de progression des niveaux.
00267  * \param mouvement_monstre Indicateur de mouvement des monstres.
00268  * \param surface Surface SDL.
00269  * \param collectibles_intermediaires Tableau de collectibles intermédiaires.
00270  * \param timestamp Horodatage.
00271  * \param touche_aller_a_droite Touche pour aller à droite.
00272  * \param touche_aller_a_gauche Touche pour aller à gauche.
00273  * \param touche_sauter_monter Touche pour sauter/monter.
00274  * \param decalage Décalage.
00275  * \param secret_1 Indicateur du premier secret découvert.
00276  * \param secret_2 Indicateur du deuxième secret découvert.
00277  * \param tile_map Carte du niveau.
00278  * \param tile_map_niveau_1 Carte spécifique du niveau 1.
00279  * \param rectangle_tile Rectangle pour chaque tuile.
00280  * \param itemsDemandeQuitter Tableau d'items pour la demande de quitter.
00281  * \param tailleDemandeQuitter Taille du tableau d'items pour la demande de quitter.
00282  * \param texture_image_perso_gagnant Texture de l'image du personnage gagnant.
00283  * \param texture_texte Texture du texte SDL.
00284  * \param police Police de caractères TTF.
00285  * \param rectangle_demande_quitter Rectangle de la demande de quitter.
00286  * \param texture_image_nuage_1 Texture de l'image du nuage 1.
00287  * \param texture_image_nuage_2 Texture de l'image du nuage 2.
00288  * \param couleurNoire Couleur noire SDL.
00289  * \param texture_image_fin_premiers_niveaux Texture de l'image de fin des premiers niveaux.
00290  * \param avancer Indicateur d'avancer.
00291  * \param reculer Indicateur de reculer.
00292  * \param sauter Indicateur de sauter.
00293  * \param position_avant_saut Position avant le saut.

```

```

00279 * \param saut Indicateur de saut.
00280 * \param tombe Indicateur de chute.
00281 * \param position_x_initiale Position initiale en x.
00282 * \param position_y_initiale Position initiale en y.
00283 * \param position_x Position en x du joueur.
00284 * \param position_y Position en y du joueur.
00285 * \param largeur Largeur de l'écran.
00286 * \param hauteur Hauteur de l'écran.
00287 * \param largeur_tile Largeur d'une tuile.
00288 * \param hauteur_tile Hauteur d'une tuile.
00289 * \param page_active Page active du jeu.
00290 * \see redimensionnement_fenetre
00291 * \see clic_plein_ecran
00292 * \see demande_quitter_niveau
00293 * \see clic_case
00294 * \see deplacement_personnage
00295 * \see erreur
00296 * \see chargement_niveau_1
00297 * \see mise_a_jour_rendu_niveau_1
00298 *
00299 */
00300 void niveau_1(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance, SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix,
00301 SDL_Texture **texture, SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran, SDL_bool *plein_ecran,
00302 SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, SDL_Texture
**texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant,
00303 SDL_Texture **texture_image_sol_surface, SDL_Texture **texture_image_sol_profondeur,
SDL_Texture **texture_image_fond,
00304 SDL_Texture **texture_image_pique, niveaux *avancee_niveaux, int *mouvement_monstre,
00305 SDL_Surface **surface, int collectibles_intermediaires[3], time_t *timestamp,
00306 SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
00307 SDL_Keycode *touche_sauter_monter, int *decalage, int *secret_1, int *secret_2,
00308 int tile_map[18][32], int tile_map_niveau_1[18][110], SDL_Rect *rectangle_tile,
00309 itemMenu *itemsDemandeQuitter, int tailleDemande, SDL_Texture
**texture_image_perso_gagnant,
00310 SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande, SDL_Texture
**texture_image_nuage_1, SDL_Texture **texture_image_nuage_2,
00311 SDL_Color couleurNoire, SDL_Texture **texture_image_fin_premiers_niveaux,
00312 int *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut, int
*tombe,
00313 int *position_x_initiale, int *position_y_initiale, int *position_x, int *position_y,
00314 int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page_t *page_active,
00315 itemMenu *itemsDemandeSauvegarde, SDL_Keycode *touche_descendre, SDL_Keycode
*touche_interagir, barreDeSon *barre_de_son, itemMenu *pseudo,
00316 modes_t *modeActif, personnage_t *personnageActif, position_t *positionActive, int
tailleNiveaux,
00317 time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
avancee_succes_intermediaires[10]) {
00318     SDL_Event event_temporaire;
00319     SDL_bool clic_effectue = SDL_FALSE;
00320
00321     Mix_Chunk *effet_sonore = NULL;
00322
00323     int i, x, y;
00324
00325     /* Mise à jour du rendu */
00326     mise_a_jour_rendu_niveau_1(renderer, texture_image_sol_surface, texture_image_sol_profondeur,
texture_image_fond,
00327 texture, rectangle_tile, rectangle_plein_ecran,
00328 texture_image_plein_ecran,
00329 texture_image_personnage, rectangle_personnage,
00330 texture_image_monstre_terrestre, texture_image_monstre_volant,
00331 texture_image_pique, avancee_niveaux,
00332 (*position_x), (*position_y), tile_map, (*secret_2),
00333 texture_image_nuage_1, texture_image_nuage_2,
00334 (*largeur), (*hauteur), (*largeur_tile), (*hauteur_tile),
00335 texture_image_croix, rectangle_croix);
00336
00337 while(SDL_PollEvent(event)) {
00338     switch(event->type) {
00339         /* Gestion de l'événement de redimensionnement de la fenêtre */
00340         case SDL_WINDOWEVENT:
00341             redimensionnement_fenetre((*event), largeur, hauteur);
00342
00343             (*largeur_tile) = (*largeur) / 32;
00344             (*hauteur_tile) = (*hauteur) / 18;
00345
00346             break;
00347         case SDL_KEYDOWN:
00348             if(event->key.keysym.sym == (*touche_sauter_monter))

```

```

00350         (*sauter) = 1;
00351
00352         if(event->key.keysym.sym == (*touche_aller_a_droite))
00353             (*avancer) = 1;
00354
00355         if(event->key.keysym.sym == (*touche_aller_a_gauche))
00356             (*reculer) = 1;
00357
00358         break;
00359
00360     case SDL_KEYUP:
00361
00362         if(event->key.keysym.sym == (*touche_sauter_monter))
00363             (*sauter) = 0;
00364
00365         if(event->key.keysym.sym == (*touche_aller_a_droite))
00366             (*avancer) = 0;
00367
00368         if(event->key.keysym.sym == (*touche_aller_a_gauche))
00369             (*reculer) = 0;
00370
00371         break;
00372
00373     /* Option plein écran */
00374     case SDL_MOUSEBUTTONDOWN:
00375
00376         if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window)) {
00377
00378             redimensionnement_fenetre((*event), largeur, hauteur);
00379
00380             (*largeur_tile) = (*largeur) / 32;
00381             (*hauteur_tile) = (*hauteur) / 18;
00382         }
00383
00384         /* Demande au joueur s'il veut quitter le niveau */
00385         if(clic_case((*event), (*rectangle_croix))) {
00386
00387             SDL_SetWindowResizable((*window), SDL_FALSE);
00388
00389             demande_quitter_niveau(renderer, rectangle_demande,
00390                                     surface, texture_texte, police, couleurNoire,
00391                                     itemsDemandeQuitter, tailleDemande, (*largeur), (*hauteur));
00392
00393             while (!clic_effectue) {
00394
00395                 while (SDL_PollEvent(&event_temporaire)) {
00396
00397                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00398
00399                         if(clic_case(event_temporaire, itemsDemandeQuitter[1].rectangle)) {
00400
00401                             (*page_active) = CARTE;
00402
00403                             for(i = 0; i < 3; i++)
00404                                 avancee_niveaux[0].numero_collectible[i] =
collectibles_intermediaires[i];
00405
00406                             for(i = 0; i < 10; i++)
00407                                 avancee_succes[i] = avancee_succes_intermediaires[i];
00408
00409                             clic_effectue = SDL_TRUE;
00410                         }
00411
00412                         else if(clic_case(event_temporaire, itemsDemandeQuitter[2].rectangle))
00413                             clic_effectue = SDL_TRUE;
00414                     }
00415                 }
00416             }
00417
00418             SDL_SetWindowResizable((*window), SDL_TRUE);
00419         }
00420
00421         break;
00422
00423     /* Quitter le programme en demandant s'il faut sauvegarder la partie */
00424     case SDL_QUIT:
00425
00426         SDL_SetWindowResizable((*window), SDL_FALSE);
00427
00428         demande_sauvegarde(renderer, rectangle_demande,
00429                             surface, texture_texte, police, couleurNoire,
00430                             itemsDemandeSauvegarde, tailleDemande, (*largeur), (*hauteur));
00431
00432         while (!clic_effectue) {
00433
00434             while (SDL_PollEvent(&event_temporaire)) {
00435

```

```

00436             if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00437
00438                 if(clic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {
00439
00440                     for(i = 0; i < 3; i++)
00441                         avancee_niveaux[0].numero_collectible[i] =
collectibles_intermediaires[i];
00442
00443                     sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
touche_sauter_monter,
00444                                         touche_descendre, touche_interagir, barre_de_son,
pseudo,
00445                                         (*modeActif), (*personnageActif),
(*positionActive),
00446                                         avancee_niveaux, tailleNiveaux,
temps_debut_partie, (*compteur_mort), avancee_succes);
00447
00448                     (*programme_lance) = SDL_FALSE;
00449                     clic_effectue = SDL_TRUE;
00450                 }
00451
00452                 else if(clic_case(event_temporaire, itemsDemandeSauvegarde[2].rectangle))
{
00453                     (*programme_lance) = SDL_FALSE;
00454                     clic_effectue = SDL_TRUE;
00455                 }
00456
00457                 else if(!clic_case(event_temporaire, (*rectangle_demande)))
00458                     clic_effectue = SDL_TRUE;
00459             }
00460         }
00461     }
00462
00463     SDL_SetWindowResizable((*window), SDL_TRUE);
00464
00465     break;
00466
00467     default:
00468         break;
00469 }
00470 }
00471
00472 /* Déplacement du personnage */
00473 deplacement_personnage(saut, tombe, position_x, position_y, position_avant_saut,
(*sauter), (*avancer), (*reculer), tile_map, (*personnageActif));
00475
00476 /* Déplacement du niveau en fonction du joueur */
00477 if((((*position_x) >= 16) && ((*decalage) < 65) && (!(*secret_1))) ||
00478     (((*position_x) <= 16) && ((*decalage) > 0) && (!(*secret_1)))) {
00479     (*decalage) += (*position_x) - 16;
00480
00481     (*position_x) = 16;
00482 }
00483
00484 /* Cas où le joueur découvre le premier secret */
00485 else if(((*position_x) < 0) && (!(*secret_1))) {
00486     (*secret_1) = 1;
00487
00488     (*position_x) = 12;
00489     (*position_y) = 15;
00490 }
00491
00492 else if((*secret_1) && ((*position_x) > 12)) {
00493     (*secret_1) = 0;
00494
00495     (*position_x) = 0;
00496 }
00497
00498 /* Cas où le joueur découvre le deuxième secret */
00499 if((tile_map[(*position_y)][(*position_x)] == 11) && (!(*secret_2))) {
00500
00501     /* Effet sonore quand le joueur découvre le second secret */
00502     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/guilian.wav")) == NULL)
00503         erreur("Chargement de l'effet sonore");
00504
00505     Mix_PlayChannel(1, effet_sonore, 0);
00506 }
00507
00508 if((tile_map[(*position_y)][(*position_x)] == 11) || (tile_map[(*position_y)][(*position_x)] ==
12) || ((tile_map[(*position_y)][(*position_x)] == 5) && ((*position_y) == 15))) {
00509     (*secret_2) = 1;

```

```

00516
00517     tile_map_niveau_1[15][52] = 5;
00518 }
00519
00520 /* Cas où le joueur sort du deuxième secret */
00521 else {
00522     (*secret_2) = 0;
00523
00524     tile_map_niveau_1[15][52] = 12;
00525 }
00526
00527 /* Cas où le joueur récupère un collectible dans le niveau 1 */
00528
00529 if((tile_map[(*position_y)][(*position_x)] == 5) && ((*position_y) == 8) &&
00530 (!avancee_niveaux[0].numero_collectible[0])) {
00531
00532     /* Effet sonore quand on ramasse un collectible */
00533     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00534         erreur("Chargement de l'effet sonore");
00535
00536     Mix_PlayChannel(1, effet_sonore, 0);
00537
00538     avancee_niveaux[0].numero_collectible[0] = 1;
00539 }
00540
00541 if((tile_map[(*position_y)][(*position_x)] == 5) && ((*position_y) == 15) &&
00542 (!avancee_niveaux[0].numero_collectible[1])) {
00543
00544     /* Effet sonore quand on ramasse un collectible */
00545     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00546         erreur("Chargement de l'effet sonore");
00547
00548     Mix_PlayChannel(1, effet_sonore, 0);
00549
00550     avancee_niveaux[0].numero_collectible[1] = 1;
00551 }
00552
00553 if((tile_map[(*position_y)][(*position_x)] == 5) && ((*position_y) == 3) &&
00554 (!avancee_niveaux[0].numero_collectible[2])) {
00555
00556     /* Effet sonore quand on ramasse un collectible */
00557     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00558         erreur("Chargement de l'effet sonore");
00559
00560     Mix_PlayChannel(1, effet_sonore, 0);
00561
00562     avancee_niveaux[0].numero_collectible[2] = 1;
00563 }
00564
00565 if(!(*secret_1))
00566     for(y = 0; y < 18; y++)
00567         for(x = 0; x < 32; x++)
00568             tile_map[y][x] = tile_map_niveau_1[y][13 + (*decalage) + x];
00569
00570 else
00571     for(y = 0; y < 18; y++)
00572         for(x = 0; x < 32; x++)
00573             tile_map[y][x] = tile_map_niveau_1[y][x];
00574
00575 /* Cas où le personnage tue un monstre */
00576 if((tile_map[(*position_y) + 1][(*position_x)] == 8) || (tile_map[(*position_y) +
00577 1][(*position_x)] == 9)) {
00578
00579     /* Effet sonore quand un monstre est tué */
00580     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_monstre.wav")) == NULL)
00581         erreur("Chargement de l'effet sonore");
00582
00583     Mix_PlayChannel(1, effet_sonore, 0);
00584
00585     tile_map_niveau_1[(*position_y) + 1][13 + (*position_x) + (*decalage)] = 0;
00586
00587     (*tombe) = 0;
00588     (*saut) = 1;
00589 }
00590
00591 /* Cas où le personnage meurt dans le vide, par des piques ou par des monstres */
00592 if((((*position_y) == 18) || (tile_map[(*position_y)][(*position_x)] == 3) ||
00593 (tile_map[(*position_y)][(*position_x)] == 8) || (tile_map[(*position_y)][(*position_x)] == 9)) {
00594
00595     (*compteur_mort)++;
00596
00597     if((*personnageActif) == PERSONNAGE_1) {
00598
00599         /* Effet sonore quand le premier personnage meurt */
00600         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_masculin.wav")) == NULL)

```



```

00598         erreur("Chargement de l'effet sonore");
00599     }
00600
00601     else if((*personnageActif) == PERSONNAGE_2) {
00602
00603         /* Effet sonore quand le premier personnage meurt */
00604         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_feminin.wav")) == NULL)
00605             erreur("Chargement de l'effet sonore");
00606     }
00607
00608     Mix_PlayChannel(1, effet_sonore, 0);
00609
00610     (*saut) = 0;
00611     (*tombe) = 0;
00612
00613     (*decalage) = 0;
00614     (*secret_1) = 0;
00615     (*mouvement_monstre) = 0;
00616
00617     chargement_niveau_1(position_x, position_y, position_x_initiale, position_y_initiale,
00618         tile_map_niveau_1);
00619 }
00620
00621 /* Déplacement des monstres */
00622 else if(!((time(NULL) - 1) % 4)) && (!((time(NULL) - 1) % 4)))
00623     (*mouvement_monstre) = 1;
00624
00625 if(((*timestamp) < time(NULL)) && (*mouvement_monstre)) {
00626     (*timestamp) = time(NULL);
00627
00628     for (y = 0; y < 18; y++)
00629         for (x = 0; x < 110; x++)
00630             if(tile_map_niveau_1[y][x] == 8) {
00631
00632                 if(!((*timestamp) % 4) || (!((*timestamp) - 1) % 4)) {
00633
00634                     tile_map_niveau_1[y][x] = 0;
00635                     tile_map_niveau_1[y][x + 1] = 8;
00636                     x++;
00637                 }
00638
00639                 else if(!((*timestamp) % 2) || (!((*timestamp) + 1) % 4)) {
00640
00641                     tile_map_niveau_1[y][x] = 0;
00642                     tile_map_niveau_1[y][x - 1] = 8;
00643                 }
00644             }
00645
00646     else if(tile_map_niveau_1[y][x] == 9) {
00647
00648         if(!((*timestamp) % 4) || (!((*timestamp) - 1) % 4)) {
00649
00650             tile_map_niveau_1[y][x] = 0;
00651             tile_map_niveau_1[y][x + 1] = 9;
00652             x++;
00653         }
00654
00655         else if(!((*timestamp) % 2) || (!((*timestamp) + 1) % 4)) {
00656
00657             tile_map_niveau_1[y][x] = 0;
00658             tile_map_niveau_1[y][x - 1] = 9;
00659         }
00660     }
00661 }
00662
00663 /* Cas où vous avez fini le niveau */
00664 if (tile_map[(*position_y)][(*position_x)] == 7) {
00665
00666     for (y = 0; y < 18; y++)
00667         for (x = 0; x < 110; x++)
00668             if((tile_map[y][x] == 8) || (tile_map[y][x] == 9)) {
00669
00670                 avancee_succes[4] = 0;
00671                 x = 110;
00672                 y = 18;
00673             }
00674
00675     else {
00676
00677         avancee_succes[4] = 1;
00678     }
00679 }
00680
00681 /* Effet sonore quand on fini un niveau */
00682 if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/fin_niveaux.wav")) == NULL)
00683     erreur("Chargement de l'effet sonore");

```

```

00684
00685     Mix_PlayChannel(1, effet_sonore, 0);
00686     /* Mise à jour du rendu */
00687     mise_a_jour_rendu_niveau_1(renderer, texture_image_sol_surface, texture_image_sol_profondeur,
texture_image_fond,
00688                                     texture, rectangle_tile, rectangle_plein_ecran,
texture_image_plein_ecran,
00689                                     texture_image_perso_gagnant, rectangle_personnage,
texture_image_monstre_terrestre, texture_image_monstre_volant,
00690                                     texture_image_pique, avancee_niveaux,
texture_image_fin_premiers_niveaux,
00691                                     (*position_x), (*position_y), tile_map, (*secret_2),
texture_image_nuage_1, texture_image_nuage_2,
00692                                     (*largeur), (*hauteur), (*largeur_tile), (*hauteur_tile),
texture_image_croix, rectangle_croix);
00693
00694     SDL_Delay(2000);
00695
00696     avancee_niveaux[0].niveau_fini = 1;
00697
00698     (*page_active) = CARTE;
00699 }
00700 }

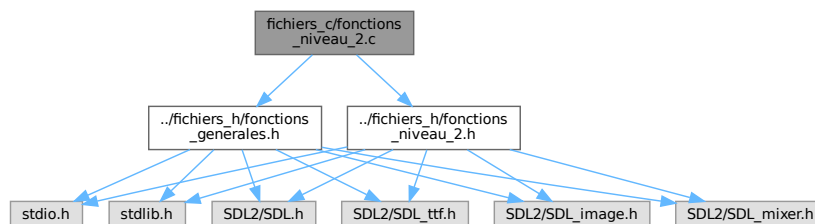
```

## 4.13 Référence du fichier fichiers\_c/fonctions\_niveau\_2.c

```
#include <../fichiers_h/fonctions_generales.h>
```

```
#include <../fichiers_h/fonctions_niveau_2.h>
```

Graphe des dépendances par inclusion de fonctions\_niveau\_2.c:



### Fonctions

- void [initialisation\\_objets\\_niveau\\_2](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_fond\_niveau\_2, SDL\_Texture \*\*texture\_image\_dossier\_niveau\_2, SDL\_Texture \*\*texture\_image\_sol\_niveau\_2, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_courant, SDL\_Texture \*\*texture\_image\_mur\_termine)

*Fonction qui permet d'initialiser les différents objets du niveau 2.*

- void [mini\\_jeu\\_1\\_niveau\\_2](#) (int \*position\_x, int \*position\_y, int tile\_map[19][27])

*Fonction qui permet d'initialiser le premier mini-jeu du niveau 2.*

- int [verification\\_chemin](#) (int x, int y, int x\_precedent, int y\_precedent, int tilemap[19][27], int x\_arrivee, int y\_arrivee)

*Fonction de vérification du chemin.*

- int [mise\\_a\\_jour\\_bordures\\_niveau\\_2](#) (SDL\_Renderer \*renderer, SDL\_Texture \*texture\_image\_mur\_termine, int tilemap[19][27], int x\_tile, int y\_tile, int x, int y, SDL\_Rect \*rectangle\_tile, int largeur\_tile, int hauteur\_tile)

*Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la 9 du labyrinthe.*

- void [mise\\_a\\_jour\\_mini\\_jeu\\_1\\_niveau\\_2](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, int position\_x, int position\_y, int tile\_map\_mini\_jeu\_niveau\_2[19][27], int largeur, int hauteur, int largeur\_tile, int hauteur\_tile, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_courant, SDL\_Texture \*\*texture\_image\_mur\_termine)
- void [mini\\_jeu\\_2\\_niveau\\_2](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], int mode\_difficile)  
*Fonction qui permet d'initialiser le second mini-jeu du niveau 2.*
- void [mise\\_a\\_jour\\_mini\\_jeu\\_2\\_niveau\\_2](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int mini\_jeu\_termine, int position\_x, int position\_y, int tile\_map[18][32], SDL\_Texture \*\*texture\_image\_porte, [niveaux](#) \*avancee\_niveaux, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile)
- void [mini\\_jeux\\_niveau\\_2](#) (SDL\_Event \*event, SDL\_Renderer \*\*renderer, SDL\_Window \*\*window, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_porte, [niveaux](#) \*avancee\_niveaux, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, int \*mini\_jeu, int \*mini\_jeu\_1\_termine, int \*mini\_jeu\_2\_termine, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int mini\_jeu\_termine, int \*position\_x, int \*position\_y, int tile\_map[18][32], int tile\_map\_mini\_jeu\_niveau\_2[19][27], SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_interagir, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, int \*valide, SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_courant, SDL\_Rect \*rectangle\_demande, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemande, int collectibles\_intermediaires[3], SDL\_Texture \*\*texture\_image\_mur\_termine, [page\\_t](#) \*page\_active, Mix\_Music \*\*musique, int \*avancer, int \*reculer, int \*sauter, int \*saut, int \*tombe, [itemMenu](#) \*itemsDemandeSauvegarde, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [modes\\_t](#) \*modeActif, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

## 4.13.1 Documentation des fonctions

### 4.13.1.1 initialisation\_objets\_niveau\_2()

```
void initialisation_objets_niveau_2 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_fond_niveau_2,
    SDL_Texture ** texture_image_dossier_niveau_2,
    SDL_Texture ** texture_image_sol_niveau_2,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Texture ** texture_image_pipe_vertical,
    SDL_Texture ** texture_image_pipe_horizontal,
    SDL_Texture ** texture_image_pipe_haut_droit,
    SDL_Texture ** texture_image_pipe_bas_droit,
    SDL_Texture ** texture_image_pipe_bas_gauche,
```

```

SDL_Texture ** texture_image_pipe_haut_gauche,
SDL_Texture ** texture_image_pipe_courant,
SDL_Texture ** texture_image_mur_termine )

```

Fonction qui permet d'initialiser les différents objets du niveau 2.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Surface SDL.
<i>texture_image_fond_niveau_2</i>	Texture de l'image de fond du niveau 2.
<i>texture_image_dossier_niveau_2</i>	Texture de l'image du dossier du niveau 2.
<i>texture_image_sol_niveau_2</i>	Texture de l'image du sol du niveau 2.
<i>texture_image_mur_mini_jeu</i>	Texture de l'image du mur du mini-jeu.
<i>texture_image_pipe_vertical</i>	Texture de l'image du tuyau vertical.
<i>texture_image_pipe_horizontal</i>	Texture de l'image du tuyau horizontal.
<i>texture_image_pipe_haut_droit</i>	Texture de l'image du tuyau haut droit.
<i>texture_image_pipe_bas_droit</i>	Texture de l'image du tuyau bas droit.
<i>texture_image_pipe_bas_gauche</i>	Texture de l'image du tuyau bas gauche.
<i>texture_image_pipe_haut_gauche</i>	Texture de l'image du tuyau haut gauche.
<i>texture_image_pipe_courant</i>	Texture de l'image du tuyau courant.
<i>texture_image_mur_termine</i>	Texture de l'image du mur terminé.

#### Voir également

[chargement\\_image](#)

Définition à la ligne 27 du fichier [fonctions\\_niveau\\_2.c](#).

#### 4.13.1.2 mini\_jeu\_1\_niveau\_2()

```

void mini_jeu_1_niveau_2 (
    int * position_x,
    int * position_y,
    int tile_map[19][27] )

```

Fonction qui permet d'initialiser le premier mini-jeu du niveau 2.

#### Paramètres

<i>position_x</i>	position horizontale du personnage sur le tile map
<i>position_y</i>	position verticale du personnage sur le tile map
<i>tile_map</i>	map ou se trouve le personnage

Définition à la ligne 59 du fichier [fonctions\\_niveau\\_2.c](#).

## 4.13.1.3 mini\_jeu\_2\_niveau\_2()

```
void mini_jeu_2_niveau_2 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    int mode_difficile )
```

Fonction qui permet d'initialiser le second mini-jeu du niveau 2.

## Paramètres

<i>position_x</i>	position verticale du joueur à l'apparition dans le niveau
<i>position_y</i>	position horizontale du joueur à l'apparition dans le niveau
<i>position_x_initiale</i>	position du joueur verticale si il venait à revenir dans le niveau ou si il venait à mourir
<i>position_y_initiale</i>	position du joueur horizontale si il venait à revenir dans le niveau ou si il venait à mourir
<i>tile_map</i>	map ou se trouve le personnage
<i>mode_difficile</i>	booléen indiquant la présence du mode difficile

Définition à la ligne 513 du fichier [fonctions\\_niveau\\_2.c](#).

## 4.13.1.4 mini\_jeux\_niveau\_2()

```
void mini_jeux_niveau_2 (
    SDL_Event * event,
    SDL_Renderer ** renderer,
    SDL_Window ** window,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_porte,
    niveaux * avancee_niveaux,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    int * mini_jeu,
    int * mini_jeu_1_termine,
    int * mini_jeu_2_termine,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int mini_jeu_termine,
    int * position_x,
    int * position_y,
    int tile_map[18][32],
    int tile_map_mini_jeu_niveau_2[19][27],
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    int * largeur,
    int * hauteur,
```

```

int * largeur_tile,
int * hauteur_tile,
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
SDL_Texture ** texture_image_mur_mini_jeu,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_interagir,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
int * valide,
SDL_Texture ** texture_image_pipe_vertical,
SDL_Texture ** texture_image_pipe_horizontal,
SDL_Texture ** texture_image_pipe_haut_droit,
SDL_Texture ** texture_image_pipe_bas_droit,
SDL_Texture ** texture_image_pipe_bas_gauche,
SDL_Texture ** texture_image_pipe_haut_gauche,
SDL_Texture ** texture_image_pipe_courant,
SDL_Rect * rectangle_demande,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Color couleurNoire,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
int collectibles_intermediaires[3],
SDL_Texture ** texture_image_mur_termine,
page_t * page_active,
Mix_Music ** musique,
int * avancer,
int * reculer,
int * sauter,
int * saut,
int * tombe,
itemMenu * itemsDemandeSauvegarde,
barreDeSon * barre_de_son,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 746 du fichier `fonctions_niveau_2.c`.

#### 4.13.1.5 mise\_a\_jour\_bordures\_niveau\_2()

```

int mise_a_jour_bordures_niveau_2 (
    SDL_Renderer * renderer,
    SDL_Texture * texture_image_mur_termine,
    int tilemap[19][27],
    int x_tile,
    int y_tile,
    int x,

```

```

int y,
SDL_Rect * rectangle_tile,
int largeur_tile,
int hauteur_tile )

```

Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la 9 du labyrinthe.

#### Paramètres

<i>renderer</i>	Renderer SDL.
<i>texture_image_mur_termine</i>	Texture de l'image du mur terminé.
<i>tilemap</i>	Carte de tuiles du niveau 2.
<i>x_tile</i>	Position x de la tuile à mettre à jour.
<i>y_tile</i>	Position y de la tuile à mettre à jour.
<i>x</i>	Position x de la tuile dans l'écran.
<i>y</i>	Position y de la tuile dans l'écran.
<i>rectangle_tile</i>	Rectangle pour chaque tuile.
<i>largeur_tile</i>	Largeur d'une tuile.
<i>hauteur_tile</i>	Hauteur d'une tuile.

#### Renvoie

appel récursif pour mettre les différents rectangle à jour jusqu'à la fin (appel se finissant par un 0)

Définition à la ligne 249 du fichier `fonctions_niveau_2.c`.

#### 4.13.1.6 mise\_a\_jour\_mini\_jeu\_1\_niveau\_2()

```

void mise_a_jour_mini_jeu_1_niveau_2 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    int position_x,
    int position_y,
    int tile_map_mini_jeu_niveau_2[19][27],
    int largeur,
    int hauteur,
    int largeur_tile,
    int hauteur_tile,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Texture ** texture_image_pipe_vertical,
    SDL_Texture ** texture_image_pipe_horizontal,
    SDL_Texture ** texture_image_pipe_haut_droit,
    SDL_Texture ** texture_image_pipe_bas_droit,
    SDL_Texture ** texture_image_pipe_bas_gauche,
    SDL_Texture ** texture_image_pipe_haut_gauche,
    SDL_Texture ** texture_image_pipe_courant,
    SDL_Texture ** texture_image_mur_termine )

```

Définition à la ligne 322 du fichier `fonctions_niveau_2.c`.

#### 4.13.1.7 mise\_a\_jour\_mini\_jeu\_2\_niveau\_2()

```
void mise_a_jour_mini_jeu_2_niveau_2 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int mini_jeu_termine,
    int position_x,
    int position_y,
    int tile_map[18][32],
    SDL_Texture ** texture_image_porte,
    niveaux * avancee_niveaux,
    int largeur,
    int hauteur,
    int largeur_tile,
    int hauteur_tile )
```

Définition à la ligne 583 du fichier `fonctions_niveau_2.c`.

#### 4.13.1.8 verification\_chemin()

```
int verification_chemin (
    int x,
    int y,
    int x_precedent,
    int y_precedent,
    int tilemap[19][27],
    int x_arrivee,
    int y_arrivee )
```

Fonction de vérification du chemin.

##### Paramètres

<code>x</code>	position actuelle verticale dans la vérification du chemin
<code>y</code>	position actuelle horizontale dans la vérification du chemin
<code>x_precedent</code>	position précédente à la position actuelle de x
<code>y_precedent</code>	position précédente à la position actuelle de y
<code>tilemap</code>	map à vérifier
<code>x_arrivee</code>	coordonnée x où se trouve la sortie
<code>y_arrivee</code>	coordonnée y où se trouve la sortie



## Renvoi

booléen (1 si le chemin est bon sinon 0)

Définition à la ligne 115 du fichier [fonctions\\_niveau\\_2.c](#).

## 4.14 fonctions\_niveau\_2.c

[Aller à la documentation de ce fichier.](#)

```
00001 /**
00002  * \file fonctions_niveau_4.c
00003  * \brief Fichier contenant les fonctions servant à la gestion du niveau 2
00004  */
00005 #include <../fichiers_h/fonctions_generales.h>
00006 #include <../fichiers_h/fonctions_niveau_2.h>
00007
00008 /**
00009  * \fn void initialisation_objets_niveau_2(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_fond_niveau_2, SDL_Texture **texture_image_dossier_niveau_2, SDL_Texture
**texture_image_sol_niveau_2, SDL_Texture **texture_image_mur_mini_jeu, SDL_Texture
**texture_image_pipe_vertical, SDL_Texture **texture_image_pipe_horizontal, SDL_Texture
**texture_image_pipe_haut_droit, SDL_Texture **texture_image_pipe_bas_droit, SDL_Texture
**texture_image_pipe_bas_gauche, SDL_Texture **texture_image_pipe_haut_gauche, SDL_Texture
**texture_image_pipe_courant, SDL_Texture **texture_image_mur_termine)
00010  * \brief Fonction qui permet d'initialiser les différents objets du niveau 2
00011  * \param renderer Pointeur vers le renderer SDL.
00012  * \param surface Surface SDL.
00013  * \param texture_image_fond_niveau_2 Texture de l'image de fond du niveau 2.
00014  * \param texture_image_dossier_niveau_2 Texture de l'image du dossier du niveau 2.
00015  * \param texture_image_sol_niveau_2 Texture de l'image du sol du niveau 2.
00016  * \param texture_image_mur_mini_jeu Texture de l'image du mur du mini-jeu.
00017  * \param texture_image_pipe_vertical Texture de l'image du tuyau vertical.
00018  * \param texture_image_pipe_horizontal Texture de l'image du tuyau horizontal.
00019  * \param texture_image_pipe_haut_droit Texture de l'image du tuyau haut droit.
00020  * \param texture_image_pipe_bas_droit Texture de l'image du tuyau bas droit.
00021  * \param texture_image_pipe_bas_gauche Texture de l'image du tuyau bas gauche.
00022  * \param texture_image_pipe_haut_gauche Texture de l'image du tuyau haut gauche.
00023  * \param texture_image_pipe_courant Texture de l'image du tuyau courant.
00024  * \param texture_image_mur_termine Texture de l'image du mur terminé.
00025  * \see chargement_image
00026  */
00027 void initialisation_objets_niveau_2(SDL_Renderer **renderer, SDL_Surface **surface,
00028                                     SDL_Texture **texture_image_fond_niveau_2, SDL_Texture
**texture_image_dossier_niveau_2,
00029                                     SDL_Texture **texture_image_sol_niveau_2, SDL_Texture
**texture_image_mur_mini_jeu,
00030                                     SDL_Texture **texture_image_pipe_vertical, SDL_Texture
**texture_image_pipe_horizontal,
00031                                     SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
**texture_image_pipe_bas_droit,
00032                                     SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
**texture_image_pipe_haut_gauche,
00033                                     SDL_Texture **texture_image_pipe_courant,
00034                                     SDL_Texture **texture_image_mur_termine) {
00035
00036     /* Chargement des images pour le niveau 2 */
00037
00038     chargement_image(renderer, surface, texture_image_fond_niveau_2,
00039                      "./images/niveau_2/fond_niveau_2.png");
00040     chargement_image(renderer, surface, texture_image_dossier_niveau_2,
00041                      "./images/niveau_2/dossier_linux.png");
00042     chargement_image(renderer, surface, texture_image_sol_niveau_2,
00043                      "./images/niveau_2/sol_niveau_2.png");
00044     chargement_image(renderer, surface, texture_image_mur_mini_jeu,
00045                      "./images/labyrinthe/mur_mini_jeux.png");
00046     chargement_image(renderer, surface, texture_image_pipe_vertical,
00047                      "./images/pipe/pipe_vertical.png");
00048     chargement_image(renderer, surface, texture_image_pipe_horizontal,
00049                      "./images/pipe/pipe_horizontal.png");
00050     chargement_image(renderer, surface, texture_image_pipe_haut_droit, "./images/pipe/pipe_HD.png");
00051     chargement_image(renderer, surface, texture_image_pipe_bas_droit, "./images/pipe/pipe_BD.png");
00052     chargement_image(renderer, surface, texture_image_pipe_bas_gauche, "./images/pipe/pipe_BG.png");
00053     chargement_image(renderer, surface, texture_image_pipe_haut_gauche, "./images/pipe/pipe_HG.png");
00054     chargement_image(renderer, surface, texture_image_pipe_courant, "./images/pipe/pipe_courant.png");
00055     chargement_image(renderer, surface, texture_image_mur_termine,
00056                      "./images/labyrinthe/mur_fin_mini_jeux.png");
00057 }
00058 /**
00059  * \fn void mini_jeu_1_niveau_2(int *position_x, int *position_y, int tile_map[19][27])
```

```

00054 * \brief Fonction qui permet d'initialiser le premier mini-jeu du niveau 2
00055 * \param position_x position horizontale du personnage sur le tile map
00056 * \param position_y position verticale du personnage sur le tile map
00057 * \param tile_map map ou se trouve le personnage
00058 */
00059 void mini_jeu_1_niveau_2(int *position_x, int *position_y, int tile_map[19][27]) {
00060     int x, y;
00061
00062     /* Positionnement du curseur en haut à gauche */
00063     (*position_x) = 0;
00064     (*position_y) = 0;
00065
00066     /* Création du premier mini-jeu */
00067     int initialisation_tile_map[19][27] = {
00068
00069         {0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
00070          0},
00071         {0, 4, 4, 6, 5, 2, 4, 3, 5, 4, 1, 2, 5, 0, 4, 6, 2, 6, 3, 4, 6, 4, 6, 2, 4, 0,
00072          0},
00073         {0, 2, 3, 4, 5, 4, 1, 4, 4, 2, 0, 5, 4, 0, 1, 5, 4, 1, 6, 4, 2, 6, 2, 1, 0, 0,
00074          0},
00075         {0, 1, 4, 2, 6, 1, 5, 4, 2, 3, 2, 3, 6, 0, 4, 5, 0, 6, 2, 5, 1, 3, 4, 6, 5, 0,
00076          0},
00077         {0, 2, 6, 3, 4, 5, 0, 0, 3, 1, 3, 6, 5, 0, 5, 6, 0, 6, 2, 3, 5, 2, 3, 0, 1, 4,
00078          0},
00079         {0, 3, 5, 1, 4, 4, 5, 0, 6, 2, 0, 0, 1, 0, 1, 4, 2, 3, 6, 0, 5, 3, 4, 0, 4, 6,
00080          0},
00081         {0, 4, 2, 3, 1, 6, 1, 3, 4, 6, 6, 1, 6, 0, 6, 2, 1, 4, 4, 1, 3, 4, 1, 0, 0, 1,
00082          0},
00083         {0, 0, 6, 3, 5, 4, 4, 3, 2, 6, 6, 2, 2, 0, 5, 3, 6, 2, 3, 4, 6, 5, 2, 2, 5, 4,
00084          0},
00085         {0, 0, 0, 5, 2, 1, 3, 1, 1, 4, 0, 6, 6, 0, 4, 3, 0, 0, 4, 2, 3, 6, 4, 5, 1, 3,
00086          0},
00087         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
00088          0},
00089         {0, 4, 2, 1, 3, 4, 5, 4, 3, 3, 1, 2, 3, 0, 4, 6, 3, 2, 6, 4, 5, 3, 0, 4, 2, 4,
00090          0},
00091         {0, 6, 1, 0, 4, 1, 5, 3, 2, 4, 6, 2, 1, 0, 2, 5, 0, 3, 2, 6, 5, 4, 0, 5, 1, 3,
00092          0},
00093         {0, 4, 4, 0, 6, 2, 3, 5, 6, 2, 3, 6, 6, 0, 5, 1, 0, 6, 2, 3, 2, 4, 3, 6, 5, 4,
00094          0},
00095         {0, 6, 5, 3, 1, 2, 4, 3, 4, 5, 6, 0, 0, 0, 3, 4, 6, 0, 2, 3, 4, 3, 2, 4, 2, 6,
00096          0},
00097         {0, 4, 2, 2, 0, 0, 1, 6, 4, 3, 1, 5, 4, 0, 4, 6, 1, 6, 0, 0, 0, 1, 2, 6, 0, 2,
00098          0},
00099         {0, 5, 6, 0, 5, 4, 3, 1, 3, 3, 3, 0, 4, 0, 5, 4, 6, 2, 1, 0, 3, 4, 2, 1, 0, 1,
00100          0},
00101         {0, 5, 4, 1, 2, 6, 5, 6, 2, 4, 5, 2, 1, 2, 4, 3, 1, 4, 3, 6, 2, 1, 5, 4, 6, 4,
00102          0},
00103         {0, 5, 3, 5, 2, 1, 6, 1, 3, 5, 6, 3, 0, 0, 6, 5, 3, 5, 1, 3, 4, 3, 5, 4, 1, 3,
00104          0},
00105         {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00106          0},
00107     };
00108
00109     /* Copie du tilemap */
00110     for (y = 0; y < 19; y++)
00111         for (x = 0; x < 27; x++)
00112             tile_map[y][x] = initialisation_tile_map[y][x];
00113 }
00114
00115 /**
00116 * \fn int verification_chemin(int x, int y, int x_precedent, int y_precedent, int tilemap[19][27],
00117 * int x_arrivee, int y_arrivee)
00118 * \brief Fonction de vérification du chemin
00119 * \param x position actuelle verticale dans la vérification du chemin
00120 * \param y position actuelle horizontale dans la vérification du chemin
00121 * \param x_precedent position précédente à la position actuelle de x
00122 * \param y_precedent position précédente à la position actuelle de y
00123 * \param tilemap map à vérifier
00124 * \param x_arrivee coordonnée x ou se trouve la sortie
00125 * \param y_arrivee coordonnée y ou se trouve la sortie
00126 * \return booléen (1 si le chemin est bon sinon 0)
00127 */
00128 int verification_chemin(int x, int y, int x_precedent, int y_precedent, int tilemap[19][27], int
x_arrivee, int y_arrivee) {
00129
00130     /* Vérifier si les coordonnées actuelles correspondent aux coordonnées de l'arrivée */
00131     if ((x == x_arrivee) && (y == y_arrivee)) {
00132         return 1;
00133     }

```

```

00120     }
00121     /* Vérification du premier tuyau car il n'est pas dans la boucle de vérification */
00122     if(tilemap[1][2] != 3){
00123         return 0;
00124     }
00125
00126     /* Vérifier si le tuyau actuel est correctement aligné avec le précédent */
00127
00128     switch (tilemap[y][x]) {
00129         case 1: /* Tuyau vertical */
00130             /* On vient du dessus */
00131             if (y_precedent == (y - 1)) {
00132                 /* On regarde que le tuyau suivant peut bien venir d'en haut*/
00133                 if((tilemap[y+1][x] != 2) && (tilemap[y+1][x] != 4) && (tilemap[y+1][x] != 5))
00134                     return verification_chemin(x, y + 1, x, y, tilemap, x_arrivee, y_arrivee); /* On
00135 va en bas */
00136             }
00137
00138             /* On vient d'en dessous */
00139             else
00140                 /* On regarde que le tuyau suivant peut bien venir d'en bas*/
00141                 if((tilemap[y-1][x] != 2) && (tilemap[y-1][x] != 3) && (tilemap[y-1][x] != 6))
00142                     return verification_chemin(x, y - 1, x, y, tilemap, x_arrivee, y_arrivee); /* On
va en haut */
00143
00144                 break;
00145
00146         case 2: /* Tuyau horizontal */
00147             /* On vient de la gauche */
00148             if (x_precedent == (x - 1)) {
00149                 /* On regarde que le tuyau suivant peut bien venir de la gauche */
00150                 if((tilemap[y][x+1] != 1) && (tilemap[y][x+1] != 3) && (tilemap[y][x+1] != 4))
00151                     return verification_chemin(x + 1, y, x, y, tilemap, x_arrivee, y_arrivee); /* On
va à droite */
00152             }
00153
00154             /* On vient de la droite */
00155             else
00156                 /* On regarde que le tuyau suivant peut bien venir de la droite */
00157                 if((tilemap[y][x-1] != 1) && (tilemap[y][x-1] != 5) && (tilemap[y][x-1] != 6))
00158                     return verification_chemin(x - 1, y, x, y, tilemap, x_arrivee, y_arrivee); /* On
va à gauche */
00159
00160                 break;
00161
00162         case 3: /* Tuyau HD */
00163             /* On vient du dessus */
00164             if (y_precedent == (y - 1)) {
00165                 /* On regarde que le tuyau suivant peut bien venir de la gauche */
00166                 if((tilemap[y][x+1] != 1) && (tilemap[y][x+1] != 3) && (tilemap[y][x+1] != 4))
00167                     return verification_chemin(x + 1, y, x, y, tilemap, x_arrivee, y_arrivee); /* On
va à droite */
00168             }
00169
00170             /* On vient de la droite */
00171             else
00172                 /* On regarde que le tuyau suivant peut bien venir d'en bas*/
00173                 if((tilemap[y-1][x] != 2) && (tilemap[y-1][x] != 3) && (tilemap[y-1][x] != 6))
00174                     return verification_chemin(x, y - 1, x, y, tilemap, x_arrivee, y_arrivee); /* On
va en haut */
00175
00176                 break;
00177
00178         case 4: /* Tuyau BD */
00179             /* On vient de la droite */
00180             if (x_precedent == (x + 1)) {
00181                 /* On regarde que le tuyau suivant peut bien venir d'en haut*/
00182                 if((tilemap[y+1][x] != 2) && (tilemap[y+1][x] != 4) && (tilemap[y+1][x] != 5))
00183                     return verification_chemin(x, y + 1, x, y, tilemap, x_arrivee, y_arrivee); /* On
va en bas */
00184             }
00185
00186             /* On vient du bas */
00187             else
00188                 /* On regarde que le tuyau suivant peut bien venir de la gauche */
00189                 if((tilemap[y][x+1] != 1) && (tilemap[y][x+1] != 3) && (tilemap[y][x+1] != 4))
00190                     return verification_chemin(x + 1, y, x, y, tilemap, x_arrivee, y_arrivee); /* On
va à droite */
00191
00192                 break;
00193
00194         case 5: /* Tuyau BG */
00195             /* On vient de la gauche */
00196             if (x_precedent == (x - 1)) {
00197                 /* On regarde que le tuyau suivant peut bien venir d'en haut*/
00198                 if((tilemap[y+1][x] != 2) && (tilemap[y+1][x] != 4) && (tilemap[y+1][x] != 5))

```

```

00199         return verification_chemin(x, y + 1, x, y, tilemap, x_arrivee, y_arrivee); /* On
va en bas */
00200     }
00201
00202     /* On vient du bas */
00203     else
00204         /* On regarde que le tuyau suivant peut bien venir de la droite */
00205         if((tilemap[y][x-1] != 1) && (tilemap[y][x-1] != 5) && (tilemap[y][x-1] != 6))
00206             return verification_chemin(x - 1, y, x, y, tilemap, x_arrivee, y_arrivee); /* On
va à gauche */
00207
00208         break;
00209
00210         case 6: /* Tuyau HG */
00211             /* On vient du haut */
00212             if (y_precedent == (y - 1)) {
00213                 /* On regarde que le tuyau suivant peut bien venir de la droite */
00214                 if((tilemap[y][x-1] != 1) && (tilemap[y][x-1] != 5) && (tilemap[y][x-1] != 6))
00215                     return verification_chemin(x - 1, y, x, y, tilemap, x_arrivee, y_arrivee); /* On
va à gauche */
00216             }
00217
00218             /* On vient de la gauche */
00219             else
00220                 /* On regarde que le tuyau suivant peut bien venir d'en bas */
00221                 if((tilemap[y-1][x] != 2) && (tilemap[y-1][x] != 3) && (tilemap[y-1][x] != 6))
00222                     return verification_chemin(x, y - 1, x, y, tilemap, x_arrivee, y_arrivee); /* On
va au haut */
00223
00224             break;
00225
00226             default:
00227                 return 0;
00228
00229             break;
00230         }
00231         return 0;
00232     }
00233
00234 /**
00235  * \fn int mise_a_jour_bordures_niveau_2(SDL_Renderer* renderer, SDL_Texture*
texture_image_mur_termine, int tilemap[19][27], int x_tile, int y_tile, int x, int y, SDL_Rect
*rectangle_tile, int largeur_tile, int hauteur_tile)
00236  * \brief Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la 9 du labyrinthe
00237  * \param renderer Renderer SDL.
00238  * \param texture_image_mur_termine Texture de l'image du mur terminé.
00239  * \param tilemap Carte de tuiles du niveau 2.
00240  * \param x_tile Position x de la tuile à mettre à jour.
00241  * \param y_tile Position y de la tuile à mettre à jour.
00242  * \param x Position x de la tuile dans l'écran.
00243  * \param y Position y de la tuile dans l'écran.
00244  * \param rectangle_tile Rectangle pour chaque tuile.
00245  * \param largeur_tile Largeur d'une tuile.
00246  * \param hauteur_tile Hauteur d'une tuile.
00247  * \return appel récursif pour mettre les différents rectangle à jour jusqu'à la fin (appel se
finissant par un 0)
00248  */
00249 int mise_a_jour_bordures_niveau_2(SDL_Renderer* renderer, SDL_Texture* texture_image_mur_termine, int
tilemap[19][27], int x_tile, int y_tile, int x, int y,
SDL_Rect *rectangle_tile, int largeur_tile, int hauteur_tile) {
00250
00251     /* Mise à jour du rendu de la tuile courante */
00252     tilemap[y_tile][x_tile] = 9;
00253
00254     rectangle_tile->x = x_tile * largeur_tile;
00255     rectangle_tile->y = y_tile * hauteur_tile;
00256     rectangle_tile->w = largeur_tile;
00257     rectangle_tile->h = hauteur_tile;
00258
00259     SDL_RenderCopy(renderer, texture_image_mur_termine, NULL, rectangle_tile);
00260
00261     SDL_RenderPresent(renderer);
00262
00263     SDL_Delay(20);
00264
00265     /* Vérification des tuiles adjacentes pour permettre un appel récursif et ainsi changé toute les
tuiles de la bordure */
00266
00267     /* Tuile à gauche de la courante */
00268     if((!tilemap[y_tile][x_tile - 1]) && (x_tile > x) && (!y_tile))
00269         return mise_a_jour_bordures_niveau_2(renderer, texture_image_mur_termine, tilemap, x_tile - 1,
y_tile, x, y,
rectangle_tile, largeur_tile, hauteur_tile);
00270
00271     else if((tilemap[y_tile][x_tile - 1] == 8) || (tilemap[y_tile][x_tile - 1] == 1)) && (x_tile > x)
&& (!y_tile))

```

```

00274         return mise_a_jour_bordures_niveau_2(renderer, texture_image_mur_termine, tilemap, x_tile - 2,
00275         y_tile, x, y,
00276         rectangle_tile, largeur_tile, hauteur_tile);
00277     /* Tuile à droite de la courante */
00278     else if(!tilemap[y_tile][x_tile + 1]) && (x_tile < x) && (y_tile == (19 - 1)))
00279         return mise_a_jour_bordures_niveau_2(renderer, texture_image_mur_termine, tilemap, x_tile + 1,
00280         y_tile, x, y,
00281         rectangle_tile, largeur_tile, hauteur_tile);
00282     /* Tuile en bas de la courante */
00283     else if(!tilemap[y_tile + 1][x_tile]) && (y_tile < y) && (!x_tile)
00284         return mise_a_jour_bordures_niveau_2(renderer, texture_image_mur_termine, tilemap, x_tile,
00285         y_tile + 1, x, y,
00286         rectangle_tile, largeur_tile, hauteur_tile);
00287     /* Tuile en haut de la courante */
00288     else if(!tilemap[y_tile - 1][x_tile]) && (y_tile > y) && (x_tile == (27 - 1)))
00289         return mise_a_jour_bordures_niveau_2(renderer, texture_image_mur_termine, tilemap, x_tile,
00290         y_tile - 1, x, y,
00291         rectangle_tile, largeur_tile, hauteur_tile);
00292     return 0;
00293 }
00294 /**
00295 * \fn void mise_a_jour_mini_jeu_1_niveau_2(SDL_Renderer **renderer, SDL_Rect *rectangle_plein_ecran,
00296 * SDL_Texture **texture_image_plein_ecran, SDL_Texture **texture, SDL_Rect *rectangle_tile, int
00297 * position_x, int position_y, int tile_map_mini_jeu_niveau_2[19][27], int largeur, int hauteur, int
00298 * largeur_tile, int hauteur_tile, SDL_Texture **texture_image_mur_mini_jeu, SDL_Texture
00299 * **texture_image_pipe_vertical, SDL_Texture **texture_image_pipe_horizontal, SDL_Texture
00300 * **texture_image_pipe_haut_droit, SDL_Texture **texture_image_pipe_bas_droit, SDL_Texture
00301 * **texture_image_pipe_bas_gauche, SDL_Texture **texture_image_pipe_haut_gauche, SDL_Texture
00302 * **texture_image_pipe_courant, SDL_Texture **texture_image_mur_termine)
00303 * \brief Fonction qui permet de mettre à jour le premier mini-jeu du niveau 2
00304 * \param renderer Pointeur vers le renderer SDL.
00305 * \param rectangle_plein_ecran Rectangle pour le plein écran.
00306 * \param texture_image_plein_ecran Texture de l'image du plein écran.
00307 * \param texture Texture SDL.
00308 * \param rectangle_tile Rectangle pour chaque tuile.
00309 * \param position_x Position x du joueur.
00310 * \param position_y Position y du joueur.
00311 * \param tile_map_mini_jeu_niveau_2 Carte de tuiles du mini-jeu 1 du niveau 2.
00312 * \param largeur Largeur de l'écran.
00313 * \param hauteur Hauteur de l'écran.
00314 * \param largeur_tile Largeur d'une tuile.
00315 * \param hauteur_tile Hauteur d'une tuile.
00316 * \param texture_image_mur_mini_jeu Texture de l'image du mur du mini-jeu.
00317 * \param texture_image_pipe_vertical Texture de l'image du tuyau vertical.
00318 * \param texture_image_pipe_horizontal Texture de l'image du tuyau horizontal.
00319 * \param texture_image_pipe_haut_droit Texture de l'image du tuyau haut droit.
00320 * \param texture_image_pipe_bas_droit Texture de l'image du tuyau bas droit.
00321 * \param texture_image_pipe_bas_gauche Texture de l'image du tuyau bas gauche.
00322 * \param texture_image_pipe_haut_gauche Texture de l'image du tuyau haut gauche.
00323 * \param texture_image_pipe_courant Texture de l'image du tuyau courant.
00324 * \param texture_image_mur_termine Texture de l'image du mur terminé.
00325 * \see erreur
00326 */
00327 void mise_a_jour_mini_jeu_1_niveau_2(SDL_Renderer **renderer, SDL_Texture **texture_image_croix,
00328 SDL_Rect *rectangle_croix,
00329 SDL_Rect *rectangle_plein_ecran, SDL_Texture
00330 **texture_image_plein_ecran,
00331 SDL_Texture **texture, SDL_Rect *rectangle_tile,
00332 int position_x, int position_y, int
00333 tile_map_mini_jeu_niveau_2[19][27],
00334 int largeur, int hauteur, int largeur_tile, int hauteur_tile,
00335 SDL_Texture **texture_image_mur_mini_jeu,
00336 SDL_Texture **texture_image_pipe_vertical, SDL_Texture
00337 **texture_image_pipe_horizontal,
00338 SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
00339 **texture_image_pipe_bas_droit,
00340 SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
00341 **texture_image_pipe_haut_gauche,
00342 SDL_Texture **texture_image_pipe_courant,
00343 SDL_Texture **texture_image_mur_termine) {
00344     int x, y;
00345     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00346     /* Nettoyer le renderer */
00347     if(SDL_RenderClear((*renderer)) != 0)
00348         erreur("Effacement rendu échoué");
00349     /* Rendu tilemap */
00350     for (y = 0; y < hauteur / hauteur_tile; y++)

```

```

00344     for (x = 0; x < largeur / largeur_tile; x++)
00345         if ((x >= 0) && (x < largeur / largeur_tile) && (y >= 0) && (y < hauteur / hauteur_tile))
00346         {
00347             /* Rendu de chaque tuile en fonction de son type */
00348             (*texture) = NULL;
00349             rectangle_tile->x = x * largeur_tile;
00350             rectangle_tile->y = y * hauteur_tile;
00351             rectangle_tile->w = largeur_tile;
00352             rectangle_tile->h = hauteur_tile;
00353             if(SDL_RenderCopy((*render), (*texture_image_mur_mini_jeu), NULL, rectangle_tile) !=
00354 0)
00355                 erreur("Copie de la texture");
00356             switch(tile_map_mini_jeu_niveau_2[y][x]){
00357                 case 1:
00358                     (*texture) = (*texture_image_pipe_vertical);
00359                     break;
00360                 case 2:
00361                     (*texture) = (*texture_image_pipe_horizontal);
00362                     break;
00363                 case 3:
00364                     (*texture) = (*texture_image_pipe_haut_droit);
00365                     break;
00366                 case 4:
00367                     (*texture) = (*texture_image_pipe_bas_droit);
00368                     break;
00369                 case 5:
00370                     (*texture) = (*texture_image_pipe_bas_gauche);
00371                     break;
00372                 case 6:
00373                     (*texture) = (*texture_image_pipe_haut_gauche);
00374                     break;
00375                 case 8:
00376                     (*texture) = (*texture_image_pipe_courant);
00377                     break;
00378                 case 9:
00379                     (*texture) = (*texture_image_mur_termine);
00380                     break;
00381                 default:
00382                     break;
00383             }
00384             if((*texture))
00385                 if(SDL_RenderCopy((*render), (*texture), NULL, rectangle_tile) != 0)
00386                     erreur("Copie de la texture");
00387         }
00388     /* Rendu du curseur en rouge */
00389     SDL_SetRenderDrawColor((*render), 255, 0, 0, 255);
00390     rectangle_tile->x = position_x * largeur_tile - largeur / 320;
00391     rectangle_tile->y = position_y * hauteur_tile - hauteur / 180;
00392     rectangle_tile->w = largeur_tile + largeur / 320 * 2;
00393     rectangle_tile->h = hauteur_tile + hauteur / 180 * 2;
00394     SDL_RenderFillRect((*render), rectangle_tile);
00395     (*texture) = NULL;
00396     rectangle_tile->x = position_x * largeur_tile;
00397     rectangle_tile->y = position_y * hauteur_tile;
00398     rectangle_tile->w = largeur_tile;
00399     rectangle_tile->h = hauteur_tile;
00400     if(SDL_RenderCopy((*render), (*texture_image_mur_mini_jeu), NULL, rectangle_tile) != 0)
00401         erreur("Copie de la texture");
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428

```

```

00429     switch(tile_map_mini_jeu_niveau_2[position_y][position_x]){
00430
00431         case 1:
00432             (*texture) = (*texture_image_pipe_vertical);
00433
00434             break;
00435
00436         case 2:
00437             (*texture) = (*texture_image_pipe_horizontal);
00438
00439             break;
00440
00441         case 3:
00442             (*texture) = (*texture_image_pipe_haut_droit);
00443
00444             break;
00445
00446         case 4:
00447             (*texture) = (*texture_image_pipe_bas_droit);
00448
00449             break;
00450
00451         case 5:
00452             (*texture) = (*texture_image_pipe_bas_gauche);
00453
00454             break;
00455
00456         case 6:
00457             (*texture) = (*texture_image_pipe_haut_gauche);
00458
00459             break;
00460
00461         case 8:
00462             (*texture) = (*texture_image_pipe_courant);
00463
00464             break;
00465
00466         case 9:
00467             (*texture) = (*texture_image_mur_termine);
00468
00469             break;
00470
00471         default:
00472             break;
00473     }
00474
00475     if((*texture))
00476         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile) != 0)
00477             erreur("Copie de la texture");
00478
00479     /* Copie la texture de l'image de plein écran */
00480
00481     rectangle_plein_ecran->x = largeur_tile * 26;
00482     rectangle_plein_ecran->y = 0;
00483     rectangle_plein_ecran->w = largeur_tile;
00484     rectangle_plein_ecran->h = hauteur_tile;
00485
00486     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00487         erreur("Copie de la texture");
00488
00489     /* Copie la texture de l'image de la croix */
00490
00491     rectangle_croix->x = 0;
00492     rectangle_croix->y = 0;
00493     rectangle_croix->w = largeur_tile;
00494     rectangle_croix->h = hauteur_tile;
00495
00496     if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00497         erreur("Copie de la texture");
00498
00499     /* Actualiser le renderer */
00500     SDL_RenderPresent((*renderer));
00501 }
00502
00503 /**
00504  * \fn void mini_jeu_2_niveau_2(int *position_x, int *position_y, int *position_x_initiale, int
00505  * position_y_initiale, int tile_map[18][32], int mode_difficile)
00506  * \brief Fonction qui permet d'initialiser le second mini-jeu du niveau 2
00507  * \param position_x position verticale du joueur à l'apparition dans le niveau
00508  * \param position_y position horizontale du joueur à l'apparition dans le niveau
00509  * \param position_x_initiale position du joueur verticale si il venait à revenir dans le niveau ou si
00510  * il venait à mourir
00511  * \param position_y_initiale position du joueur horizontale si il venait à revenir dans le niveau ou
00512  * si il venait à mourir
00513  * \param tile_map map ou se trouve le personnage
00514  * \param mode_difficile booléen indiquant la présence du mode difficile
00515  */

```





```

00575 * \param avancee_niveaux Pointeur vers l'état d'avancement des niveaux.
00576 * \param largeur Largeur de l'écran.
00577 * \param hauteur Hauteur de l'écran.
00578 * \param largeur_tile Largeur d'une tuile.
00579 * \param hauteur_tile Hauteur d'une tuile.
00580 *
00581 *
00582 */
00583 void mise_a_jour_mini_jeu_2_niveau_2(SDL_Renderer **renderer, SDL_Texture **texture_image_fond,
SDL_Texture **texture_image_sol, SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix,
00584 SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00585 SDL_Texture **texture, SDL_Rect *rectangle_tile, SDL_Texture
**texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant,
00586 SDL_Texture **texture_image_personnage, SDL_Rect
*rectangle_personnage, int mini_jeu_termine,
00587 int position_x, int position_y, int tile_map[18][32], SDL_Texture
**texture_image_porte, niveaux *avancee_niveaux,
00588 int largeur, int hauteur, int largeur_tile, int hauteur_tile) {
00589
00590     int x, y;
00591
00592     /* Efface le rendu */
00593     if(SDL_RenderClear((*renderer)) != 0)
00594         erreur("Effacement rendu échoué");
00595
00596     /* Copie la texture de l'image de fond du salon */
00597     if(SDL_RenderCopy((*renderer), (*texture_image_fond), NULL, NULL) != 0)
00598         erreur("Copie de la texture");
00599
00600     /* Affiche tout le salon en fonction des valeurs */
00601     for (y = 0; y < hauteur / hauteur_tile; y++) {
00602
00603         for (x = 0; x < largeur / largeur_tile; x++) {
00604
00605             if(tile_map[y][x] == 1)
00606                 (*texture) = (*texture_image_sol);
00607
00608             else
00609                 (*texture) = NULL;
00610
00611             rectangle_tile->x = x * largeur_tile;
00612             rectangle_tile->y = y * hauteur_tile;
00613             rectangle_tile->w = largeur_tile;
00614             rectangle_tile->h = hauteur_tile;
00615
00616             if((*texture))
00617                 if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00618                     erreur("Copie de la texture");
00619
00620             if(mini_jeu_termine) {
00621
00622                 if(tile_map[y][x] == 4)
00623                     if(SDL_RenderCopy((*renderer), (*texture_image_porte), NULL, rectangle_tile))
00624                         erreur("Copie de la texture");
00625
00626                 if((tile_map[y][x] == 5) && (avancee_niveaux[1].numero_collectible[1] == 0))
00627                     if(SDL_RenderCopy((*renderer), avancee_niveaux[1].texture_image_collectible, NULL,
rectangle_tile))
00628                         erreur("Copie de la texture");
00629             }
00630
00631             if(tile_map[y][x] == 8)
00632                 if(SDL_RenderCopy((*renderer), (*texture_image_monstre_terrestre), NULL,
rectangle_tile))
00633                     erreur("Copie de la texture");
00634
00635             if(tile_map[y][x] == 9)
00636                 if(SDL_RenderCopy((*renderer), (*texture_image_monstre_volant), NULL, rectangle_tile))
00637                     erreur("Copie de la texture");
00638         }
00639     }
00640
00641     /* Copie la texture de l'image du personnage */
00642
00643     rectangle_personnage->x = position_x * largeur_tile;
00644     rectangle_personnage->y = position_y * hauteur_tile;
00645     rectangle_personnage->w = largeur_tile;
00646     rectangle_personnage->h = hauteur_tile;
00647
00648     if(SDL_RenderCopy((*renderer), (*texture_image_personnage), NULL, rectangle_personnage) != 0)
00649         erreur("Copie de la texture");
00650
00651     /* Copie la texture de l'image de plein écran */
00652
00653     rectangle_plein_ecran->x = largeur_tile * 31;
00654     rectangle_plein_ecran->y = 0;

```

```

00655     rectangle_plein_ecran->w = largeur_tile;
00656     rectangle_plein_ecran->h = hauteur_tile;
00657
00658     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00659         erreur("Copie de la texture");
00660
00661     /* Copie la texture de l'image de la croix */
00662
00663     rectangle_croix->x = 0;
00664     rectangle_croix->y = 0;
00665     rectangle_croix->w = largeur_tile;
00666     rectangle_croix->h = hauteur_tile;
00667
00668     if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00669         erreur("Copie de la texture");
00670
00671     /* Affiche le rendu */
00672     SDL_RenderPresent((*renderer));
00673 }
00674
00675
00676 /**
00677  * \fn
00678  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans les mini-jeux
00679  du niveau 2
00680  * \param event Événement SDL.
00681  * \param renderer Pointeur vers le renderer SDL.
00682  * \param window Pointeur vers la fenêtre SDL.
00683  * \param texture_image_fond Texture de l'image de fond.
00684  * \param texture_image_sol Texture de l'image du sol.
00685  * \param rectangle_plein_ecran Rectangle pour le plein écran.
00686  * \param texture_image_plein_ecran Texture de l'image du plein écran.
00687  * \param plein_ecran Indicateur de mode plein écran.
00688  * \param texture_image_porte Texture de l'image de la porte.
00689  * \param avancee_niveaux Pointeur vers l'état d'avancement des niveaux.
00690  * \param texture Texture SDL.
00691  * \param rectangle_tile Rectangle pour chaque tuile.
00692  * \param mini_jeu Indicateur du mini-jeu en cours.
00693  * \param mini_jeu_1_termine Indicateur de la fin du mini-jeu 1.
00694  * \param mini_jeu_2_termine Indicateur de la fin du mini-jeu 2.
00695  * \param texture_image_personnage Texture de l'image du personnage.
00696  * \param rectangle_personnage Rectangle pour le personnage.
00697  * \param mini_jeu_termine Indicateur de la fin du mini-jeu.
00698  * \param position_x Position x du joueur.
00699  * \param position_y Position y du joueur.
00700  * \param tile_map Carte de tuiles du niveau.
00701  * \param tile_map_mini_jeu_niveau_2 Carte de tuiles du mini-jeu 1 du niveau 2.
00702  * \param texture_image_monstre_terrestre Texture de l'image du monstre terrestre.
00703  * \param texture_image_monstre_volant Texture de l'image du monstre volant.
00704  * \param largeur Largeur de l'écran.
00705  * \param hauteur Hauteur de l'écran.
00706  * \param largeur_tile Largeur d'une tuile.
00707  * \param hauteur_tile Hauteur d'une tuile.
00708  * \param texture_image_mur_mini_jeu Texture de l'image du mur du mini-jeu.
00709  * \param touche_aller_a_droite Touche pour aller à droite.
00710  * \param touche_aller_a_gauche Touche pour aller à gauche.
00711  * \param touche_interagir Touche pour interagir.
00712  * \param touche_sauter_monter Touche pour sauter/monter.
00713  * \param touche_descendre Touche pour descendre.
00714  * \param valide Indicateur de la validité de l'action.
00715  * \param texture_image_pipe_vertical Texture de l'image du tuyau vertical.
00716  * \param texture_image_pipe_horizontal Texture de l'image du tuyau horizontal.
00717  * \param texture_image_pipe_haut_droit Texture de l'image du tuyau haut droit.
00718  * \param texture_image_pipe_bas_droit Texture de l'image du tuyau bas droit.
00719  * \param texture_image_pipe_bas_gauche Texture de l'image du tuyau bas gauche.
00720  * \param texture_image_pipe_haut_gauche Texture de l'image du tuyau haut gauche.
00721  * \param texture_image_pipe_courant Texture de l'image du tuyau courant.
00722  * \param rectangle_demande_quitter Rectangle pour la demande de quitter.
00723  * \param surface Surface SDL.
00724  * \param texture_texte Texture du texte.
00725  * \param police Police de caractères.
00726  * \param couleurNoire Couleur noire.
00727  * \param itemsDemandeQuitter Liste des items pour la demande de quitter.
00728  * \param tailleDemandeQuitter Taille de la liste des items pour la demande de quitter.
00729  * \param collectibles_intermediaires Tableau des collectibles intermédiaires.
00730  * \param texture_image_mur_termine Texture de l'image du mur terminé.
00731  * \param page_active Page active.
00732  * \param musique Musique SDL.
00733  * \param avancer Indicateur de déplacement vers l'avant.
00734  * \param reculer Indicateur de déplacement vers l'arrière.
00735  * \param sauter Indicateur de saut.
00736  * \param saut Indicateur de saut en cours.
00737  * \param tombe Indicateur de chute.
00738  * \see redimensionnement_fenetre
00739  * \see clic_plein_ecran
00740  * \see demande_quitter_niveau
00741  * \see clic_case

```

```

00741 * \see verification_chemin
00742 * \see mise_a_jour_bordures_niveau_2
00743 * \see salon_arrivee_niveaux_2_3
00744 * \see mise_a_jour_mini_jeu_1_niveau_2
00745 */
00746 void mini_jeux_niveau_2(SDL_Event *event, SDL_Renderer **renderer, SDL_Window **window, SDL_bool
*programme_lance, SDL_Texture **texture_image_fond, SDL_Texture **texture_image_sol,
00747 SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
SDL_bool *plein_ecran, SDL_Texture **texture_image_porte, niveaux *avancee_niveaux,
00748 SDL_Texture **texture, SDL_Rect *rectangle_tile, int *mini_jeu, int
*mini_jeu_1_termine, int *mini_jeu_2_termine,
00749 SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int
mini_jeu_termine,
00750 int *position_x, int *position_y, int tile_map[18][32], int
tile_map_mini_jeu_niveau_2[19][27], SDL_Texture **texture_image_monstre_terrestre, SDL_Texture
**texture_image_monstre_volant,
00751 int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, SDL_Texture
**texture_image_croix, SDL_Rect *rectangle_croix,
00752 SDL_Texture **texture_image_mur_mini_jeu, SDL_Keycode *touche_aller_a_droite,
SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_interagir,
00753 SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, int *valide,
00754 SDL_Texture **texture_image_pipe_vertical, SDL_Texture
**texture_image_pipe_horizontal,
00755 SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
**texture_image_pipe_bas_droit,
00756 SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
**texture_image_pipe_haut_gauche,
00757 SDL_Texture **texture_image_pipe_courant, SDL_Rect *rectangle_demande,
00758 SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
SDL_Color couleurNoire,
00759 itemMenu *itemsDemandeQuitter, int tailleDemande, int
collectibles_intermediaires[3],
00760 SDL_Texture **texture_image_mur_termine, page_t *page_active, Mix_Music
**musique,
00761 int *avancer, int *reculer, int *sauter, int *saut, int *tombe,
00762 itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
00763 modes_t *modeActif, personnage_t *personnageActif, position_t *positionActive,
int tailleNiveaux,
00764 time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
avancee_succes_intermediaires[10]) {
00765
00766 /* Cas où on est dans le premier mini-jeu */
00767 if((*mini_jeu) == 1) {
00768
00769 SDL_Event event_temporaire;
00770 SDL_bool clic_effectue = SDL_FALSE;
00771
00772 int courant_active = 0;
00773
00774 int i;
00775
00776 while(SDL_PollEvent(event)) {
00777
00778 switch(event->type){
00779
00780 /* Gestion de l'événement de redimensionnement de la fenêtre */
00781 case SDL_WINDOWEVENT:
00782 redimensionnement_fenetre((event), largeur, hauteur);
00783
00784 (*largeur_tile) = (*largeur) / 27;
00785 (*hauteur_tile) = (*hauteur) / 19;
00786
00787 break;
00788
00789 case SDL_KEYDOWN:
00790
00791 if(event->key.keysym.sym == (*touche_sauter_monter))
00792 if((*position_y) > 0)
00793 (*position_y)--;
00794
00795 if(event->key.keysym.sym == (*touche_descendre))
00796 if((*position_y) < (19 - 1))
00797 (*position_y)++;
00798
00799 if(event->key.keysym.sym == (*touche_aller_a_gauche))
00800 if((*position_x) > 0)
00801 (*position_x)--;
00802
00803 if(event->key.keysym.sym == (*touche_aller_a_droite))
00804 if((*position_x) < (27 - 1))
00805 (*position_x)++;
00806
00807 if(event->key.keysym.sym == (*touche_interagir)) {
00808
00809 if((tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] != 0) &&
(tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] != 7) &&
(tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] != 8)) {

```

```

00810
00811         if ((*valide) == 1) && ((*position_x) <= 12) && ((*position_y) <= 9))
00812             break;
00813
00814         if ((*valide) == 2) && ((*position_x) <= 12) && ((*position_y) <= 18))
00815             break;
00816
00817         if ((*valide) == 3 && ((*position_x) <= 12 && (*position_y) <= 18) ||
00818             ((*position_x) >= 12 && ((*position_y) >= 9)))
00819             break;
00820
00821         if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 1 &&
00822             (*position_y) != 0 && (*position_y) != 0)
00823             tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] = 2;
00824
00825         else if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 2)
00826             tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] = 1;
00827
00828         else if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 3)
00829             tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] = 4;
00830
00831         else if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 4)
00832             tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] = 5;
00833
00834         else if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 5)
00835             tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] = 6;
00836
00837         else if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 6)
00838             tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] = 3;
00839     }
00840     if (tile_map_mini_jeu_niveau_2[(*position_y)][(*position_x)] == 8 &&
00841         (*position_y) == 0 && (*position_x) == 2)
00842         courant_active = 1;
00843     }
00844     break;
00845
00846     /* Option plein écran */
00847     case SDL_MOUSEBUTTONDOWN:
00848
00849         if (clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window)) {
00850             redimensionnement_fenetre((*event), largeur, hauteur);
00851
00852             (*largeur_tile) = (*largeur) / 27;
00853             (*hauteur_tile) = (*hauteur) / 19;
00854         }
00855
00856     /* Demande au joueur s'il veut quitter le niveau */
00857     if (clic_case((*event), (*rectangle_croix))) {
00858         SDL_SetWindowResizable((*window), SDL_FALSE);
00859
00860         demande_quitter_niveau(renderer, rectangle_demande,
00861             surface, texture_texte, police, couleurNoire,
00862             itemsDemandeQuitter, tailleDemande, (*largeur),
00863             (*hauteur));
00864
00865         while (!clic_effectue) {
00866             while (SDL_PollEvent(&event_temporaire)) {
00867                 if (event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00868                     if (clic_case(event_temporaire, itemsDemandeQuitter[1].rectangle))
00869                     {
00870                         (*page_active) = CARTE;
00871
00872                         for (i = 0; i < 3; i++)
00873                             avancee_niveaux[1].numero_collectible[i] =
00874                                 collectibles_intermediaires[i];
00875
00876                         for (i = 0; i < 10; i++)
00877                             avancee_succes[i] = avancee_succes_intermediaires[i];
00878
00879                         clic_effectue = SDL_TRUE;
00880                     }
00881
00882                     else if (clic_case(event_temporaire,
00883                         itemsDemandeQuitter[2].rectangle))
00884                         clic_effectue = SDL_TRUE;
00885                 }
00886             }
00887         }
00888     }
00889 }

```

```

00890
00891         SDL_SetWindowResizable((*window), SDL_TRUE);
00892     }
00893
00894     break;
00895
00896     /* Quitter le programme en demandant s'il faut sauvergar la partie */
00897     case SDL_QUIT:
00898
00899         SDL_SetWindowResizable((*window), SDL_FALSE);
00900
00901         demande_sauvegarde(renderer, rectangle_demande,
00902                             surface, texture_texte, police, couleurNoire,
00903                             itemsDemandeSauvegarde, tailleDemande, (*largeur),
00904                             (*hauteur));
00905
00906         while (!cllic_effectue) {
00907             while (SDL_PollEvent(&event_temporaire)) {
00908                 if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00909                     if(cllic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {
00910                         for(i = 0; i < 3; i++)
00911                             avancee_niveaux[1].numero_collectible[i] =
00912                                 collectibles_intermediaires[i];
00913
00914                         sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
00915                                             touche_sauter_monter,
00916                                             touche_descendre, touche_interagir,
00917                                             barre_de_son, pseudo,
00918                                             (*modeActif), (*personnageActif),
00919                                             (*positionActive),
00920                                             temps_debut_partie, (*compteur_mort), avancee_succes);
00921
00922                         (*programme_lance) = SDL_FALSE;
00923                         clic_effectue = SDL_TRUE;
00924                     }
00925                     else if(cllic_case(event_temporaire,
00926                                     itemsDemandeSauvegarde[2].rectangle)) {
00927                         (*programme_lance) = SDL_FALSE;
00928                         clic_effectue = SDL_TRUE;
00929                     }
00930                     else if(!cllic_case(event_temporaire, (*rectangle_demande)))
00931                         clic_effectue = SDL_TRUE;
00932                 }
00933             }
00934         }
00935
00936         SDL_SetWindowResizable((*window), SDL_TRUE);
00937
00938         break;
00939
00940     default:
00941         break;
00942     }
00943 }
00944
00945 /* Vérifier le chemin */
00946 if(courant_active){
00947
00948     /* Vérifier si le joueur a gagné */
00949     if (verification_chemin(2, 0 + 1, 2, 0, tile_map_mini_jeu_niveau_2, 11, 9)){
00950         mise_a_jour_bordures_niveau_2((*renderer), (*texture_image_mur_termine),
00951                                     tile_map_mini_jeu_niveau_2, 12, 0, 0, 9,
00952                                     rectangle_tile, (*largeur_tile), (*hauteur_tile));
00953         (*valide) = 1;
00954     }
00955
00956     if ((verification_chemin(2, 0 + 1, 2, 0, tile_map_mini_jeu_niveau_2, 11, 9)) &&
00957         (verification_chemin(11, 9 + 1, 11, 9, tile_map_mini_jeu_niveau_2, 13, 16))){
00958         mise_a_jour_bordures_niveau_2((*renderer), (*texture_image_mur_termine),
00959                                     tile_map_mini_jeu_niveau_2, 0, 10, 12, (19 - 1),
00960                                     rectangle_tile, (*largeur_tile), (*hauteur_tile));
00961         (*valide) = 2;
00962     }
00963
00964     if ((verification_chemin(2, 0 + 1, 2, 0, tile_map_mini_jeu_niveau_2, 11, 9)) &&
00965         (verification_chemin(11, 9 + 1, 11, 9, tile_map_mini_jeu_niveau_2, 13, 16)) &&
00966         verification_chemin(13, 16, 13 - 1, 16, tile_map_mini_jeu_niveau_2, 23, 9)){
00967         mise_a_jour_bordures_niveau_2((*renderer), (*texture_image_mur_termine),
00968                                     tile_map_mini_jeu_niveau_2, 13, (19 - 1), (27 - 1), 9,

```

```

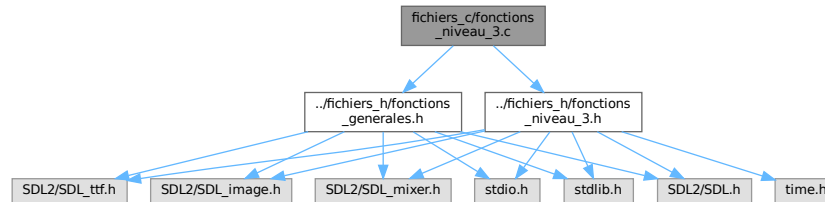
00967                                     rectangle_tile, (*largeur_tile), (*hauteur_tile));
00968             (*valide) = 3;
00969         }
00970
00971         if((verification_chemin(2, 0 + 1, 2, 0, tile_map_mini_jeu_niveau_2, 11, 9)) &&
00972            (verification_chemin(11, 9 + 1, 11, 9, tile_map_mini_jeu_niveau_2, 13, 16)) &&
00973            verification_chemin(13, 16, 13 - 1, 16, tile_map_mini_jeu_niveau_2, 23, 9)) &&
00974            verification_chemin(23, 9 - 1, 23, 9, tile_map_mini_jeu_niveau_2, 24, 0)) {
00975
00976             mise_a_jour_bordures_niveau_2((*renderer), (*texture_image_mur_termine),
tile_map_mini_jeu_niveau_2, (27 - 1), 9, 13, 0,
00977                                     rectangle_tile, (*largeur_tile), (*hauteur_tile));
00978
00979             /* Musique du salon */
00980             if(((*musique) = Mix_LoadMUS("./sons/musiques/salon.mp3")) == NULL)
00981                 erreur("Chargement de la musique");
00982
00983             Mix_PlayMusic((*musique), -1);
00984
00985             (*mini_jeu) = 0;
00986             (*mini_jeu_1_termine) = 1;
00987
00988             (*largeur_tile) = (*largeur) / 32;
00989             (*hauteur_tile) = (*hauteur) / 18;
00990
00991             (*sauter) = 0;
00992             (*avancer) = 0;
00993             (*reculer) = 0;
00994             (*tombe) = 0;
00995             (*saut) = 0;
00996
00997             salon_arrivee_niveaux_2_3(position_x, position_y, tile_map, (*page_active));
00998
00999             tile_map[4][2] = 0;
01000             tile_map[6][3] = 5;
01001
01002             if((*mini_jeu_2_termine))
01003                 tile_map[2][27] = 0;
01004         }
01005
01006         courant_active = 0;
01007     }
01008
01009     /* Mise à jour du rendu */
01010     mise_a_jour_mini_jeu_1_niveau_2(renderer, texture_image_croix, rectangle_croix,
01011                                     rectangle_plein_ecran, texture_image_plein_ecran,
01012                                     texture, rectangle_tile,
01013                                     (*position_x), (*position_y), tile_map_mini_jeu_niveau_2,
01014                                     (*largeur), (*hauteur), (*largeur_tile), (*hauteur_tile),
01015                                     texture_image_mur_mini_jeu,
01016                                     texture_image_pipe_vertical, texture_image_pipe_horizontal,
01017                                     texture_image_pipe_haut_droit, texture_image_pipe_bas_droit,
01018                                     texture_image_pipe_bas_gauche, texture_image_pipe_haut_gauche,
01019                                     texture_image_pipe_courant,
01020                                     texture_image_mur_termine);
01021 }
01022
01023 /* Cas où on est dans le deuxième mini-jeu */
01024 else if((*mini_jeu) == 2) {
01025
01026     if(mini_jeu_termine) {
01027         tile_map[2][29] = 5;
01028         tile_map[16][20] = 4;
01029     }
01030
01031     /* Mise à jour du rendu */
01032     mise_a_jour_mini_jeu_2_niveau_2(renderer, texture_image_fond, texture_image_sol,
01033                                     texture_image_croix, rectangle_croix,
01034                                     rectangle_plein_ecran, texture_image_plein_ecran,
01035                                     texture, rectangle_tile, texture_image_monstre_terrestre,
01036                                     texture_image_monstre_volant,
01037                                     texture_image_personnage, rectangle_personnage,
01038                                     mini_jeu_termine,
01039                                     (*position_x), (*position_y), tile_map, texture_image_porte,
01040                                     avancee_niveaux,
01041                                     (*largeur), (*hauteur), (*largeur_tile), (*hauteur_tile));
01042 }
01043 }
01044 }

```

## 4.15 Référence du fichier fichiers\_c/fonctions\_niveau\_3.c

```
#include <../fichiers_h/fonctions_generales.h>
#include <../fichiers_h/fonctions_niveau_3.h>
```

Graphe des dépendances par inclusion de fonctions\_niveau\_3.c:



### Fonctions

- void [initialisation\\_objets\\_niveau\\_3](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_fond\_niveau\_3, SDL\_Texture \*\*texture\_image\_dossier\_niveau\_3, SDL\_Texture \*\*texture\_image\_sol\_niveau\_3, SDL\_Texture \*\*barre\_windows\_1, SDL\_Texture \*\*barre\_windows\_2, SDL\_Texture \*\*barre\_windows\_3, SDL\_Texture \*\*barre\_windows\_4, SDL\_Texture \*\*texture\_image\_puzzle, SDL\_Texture \*\*texture\_image\_sol\_labyrinthe, SDL\_Texture \*\*texture\_image\_bordure\_labyrinthe, SDL\_Texture \*\*texture\_image\_fin\_labyrinthe)  
*Fonction qui permet d'initialiser les différents objets du niveau 2.*
- SDL\_Rect [rectangle\\_piece\\_aleatoire](#) (int largeur, int hauteur)  
*Fonction pour obtenir un rectangle représentant une pièce de puzzle aléatoire.*
- void [mise\\_a\\_jour\\_mini\\_jeu\\_1\\_niveau\\_3](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_puzzle, SDL\_Rect rectangle\_piece[45], SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix)
- int [piece\\_proche\\_position\\_correcte](#) (SDL\_Rect rectangle\_piece, SDL\_Rect rectangle\_correct)  
*Fonction pour vérifier si une pièce est proche de sa position correcte.*
- int [verification\\_puzzle\\_fini](#) (const int piece\_bloquee[])  
*Fonction pour vérifier si toutes les pièces du puzzle sont bloquées (à leur position correcte)*
- void [mini\\_jeu\\_2\\_niveau\\_3](#) (int \*position\_x, int \*position\_y, int \*bloc\_x, int \*bloc\_y, int tile\_map[24][32])  
*Fonction qui permet d'initialiser le second mini-jeu du niveau 3.*
- void [traitement\\_touches](#) (int \*position\_x, int \*position\_y, int \*bloc\_x, int \*bloc\_y, int tilemap[24][32], int direction)  
*Fonction pour traiter les commandes utilisateur.*
- int [mise\\_a\\_jour\\_bordures\\_niveau\\_3](#) (SDL\_Renderer \*renderer, SDL\_Texture \*texture\_image\_mur\_termine, int tilemap[24][32], int x\_tile, int y\_tile, SDL\_Rect \*rectangle\_tile, int largeur\_tile, int hauteur\_tile)  
*Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la fin du labyrinthe.*
- void [mise\\_a\\_jour\\_mini\\_jeu\\_2\\_niveau\\_3](#) (SDL\_Renderer \*\*renderer, [modes\\_t](#) \*modeActif, SDL\_Texture \*\*texture\_image\_sol\_labyrinthe, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Texture \*\*texture\_image\_bordure\_labyrinthe, SDL\_Texture \*\*texture\_image\_mur\_termine, SDL\_Texture \*\*texture\_image\_fin\_labyrinthe, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, int bloc\_x, int bloc\_y, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int position\_x, int position\_y, int tile\_map\_mini\_jeu\_niveau\_3[24][32], [niveaux](#) \*avancee\_niveaux, int largeur\_tile, int hauteur\_tile)
- void [mini\\_jeux\\_niveau\\_3](#) (SDL\_Event \*event, SDL\_Renderer \*\*renderer, SDL\_Window \*\*window, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, [niveaux](#) \*avancee\_niveaux, int tile\_map[18][32], SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int \*mini\_jeu, int \*mini\_jeu\_1\_termine, int \*mini\_jeu\_2\_termine, int \*position\_x, int \*position\_y, SDL\_Texture \*\*texture, int \*largeur, int \*hauteur, SDL\_Rect \*rectangle\_demande, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, [itemMenu](#)

```

*itemsDemandeQuitter, int tailleDemande, int collectibles_intermediaires[3], page\_t *page_active, SDL_Rect
*rectangle_tile, int *largeur_tile, int *hauteur_tile, int *avancer, int *reculer, int *sauter, int *saut, int *tombe,
SDL_Rect rectangle_piece[45], int piece_bloquee[45], SDL_Rect rectangle_emplacement_piece[45], int
*piece_selectionnee, int *decalage_x, int *decalage_y, SDL_Texture **texture_image_puzzle, int tile_map↵
_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int *bloc_x, int *bloc_y, SDL_Texture **texture_↵
image_sol_labyrinthe, SDL_Texture **texture_image_bordure_labyrinthe, SDL_Texture **texture_image_↵
fin_labyrinthe, Mix_Music **musique, SDL_Texture **texture_image_personnage, SDL_Rect *rectangle↵
_personnage, SDL_Texture **texture_image_mur_termine, SDL_Texture **texture_image_mur_mini_jeu,
SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_↵
interagir, SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, modes\_t *modeActif,
itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo, personnage\_t
*personnageActif, position\_t *positionActive, int tailleNiveaux, time_t temps_debut_partie, int *compteur↵
_mort, int *avancee_succes, int avancee_succes_intermediaires[10])

```

## 4.15.1 Documentation des fonctions

### 4.15.1.1 initialisation\_objets\_niveau\_3()

```

void initialisation_objets_niveau_3 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_fond_niveau_3,
    SDL_Texture ** texture_image_dossier_niveau_3,
    SDL_Texture ** texture_image_sol_niveau_3,
    SDL_Texture ** barre_windows_1,
    SDL_Texture ** barre_windows_2,
    SDL_Texture ** barre_windows_3,
    SDL_Texture ** barre_windows_4,
    SDL_Texture ** texture_image_puzzle,
    SDL_Texture ** texture_image_sol_labyrinthe,
    SDL_Texture ** texture_image_bordure_labyrinthe,
    SDL_Texture ** texture_image_fin_labyrinthe )

```

Fonction qui permet d'initialiser les différents objets du niveau 2.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_fond_niveau_3</i>	Texture de l'image de fond du niveau 3.
<i>texture_image_dossier_niveau_3</i>	Texture de l'image du dossier pour le niveau 3.
<i>texture_image_sol_niveau_3</i>	Texture de l'image du sol du niveau 3.
<i>barre_windows_1</i>	Texture de la barre Windows 1.
<i>barre_windows_2</i>	Texture de la barre Windows 2.
<i>barre_windows_3</i>	Texture de la barre Windows 3.
<i>barre_windows_4</i>	Texture de la barre Windows 4.
<i>texture_image_puzzle</i>	Texture de l'image du puzzle.
<i>texture_image_sol_labyrinthe</i>	Texture de l'image du sol du labyrinthe.
<i>texture_image_bordure_labyrinthe</i>	Texture de l'image de la bordure du labyrinthe.
<i>texture_image_fin_labyrinthe</i>	Texture de l'image de fin du labyrinthe.



Voir également

[chargement\\_image](#)

Définition à la ligne 27 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.2 mini\_jeu\_2\_niveau\_3()

```
void mini_jeu_2_niveau_3 (
    int * position_x,
    int * position_y,
    int * bloc_x,
    int * bloc_y,
    int tile_map[24][32] )
```

Fonction qui permet d'initialiser le second mini-jeu du niveau 3.

##### Paramètres

<i>position_x</i>	position x du personnage
<i>position_y</i>	position y du personnage
<i>bloc_x</i>	position x du bloc à déplacer
<i>bloc_y</i>	position y du bloc à déplacer
<i>tile_map</i>	map ou se trouve le personnage

Définition à la ligne 147 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.3 mini\_jeux\_niveau\_3()

```
void mini_jeux_niveau_3 (
    SDL_Event * event,
    SDL_Renderer ** renderer,
    SDL_Window ** window,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    niveaux * avancee_niveaux,
    int tile_map[18][32],
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    int * mini_jeu,
    int * mini_jeu_1_termine,
    int * mini_jeu_2_termine,
    int * position_x,
    int * position_y,
    SDL_Texture ** texture,
    int * largeur,
    int * hauteur,
    SDL_Rect * rectangle_demande,
```

```

SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Color couleurNoire,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
int collectibles_intermediaires[3],
page_t * page_active,
SDL_Rect * rectangle_tile,
int * largeur_tile,
int * hauteur_tile,
int * avancer,
int * reculer,
int * sauter,
int * saut,
int * tombe,
SDL_Rect rectangle_piece[45],
int piece_bloquee[45],
SDL_Rect rectangle_emplacement_piece[45],
int * piece_selectionnee,
int * decalage_x,
int * decalage_y,
SDL_Texture ** texture_image_puzzle,
int tile_map_mini_jeu_niveau_3[24][32],
int * descendre,
int * interagir,
int * bloc_x,
int * bloc_y,
SDL_Texture ** texture_image_sol_labyrinthe,
SDL_Texture ** texture_image_bordure_labyrinthe,
SDL_Texture ** texture_image_fin_labyrinthe,
Mix_Music ** musique,
SDL_Texture ** texture_image_personnage,
SDL_Rect * rectangle_personnage,
SDL_Texture ** texture_image_mur_termine,
SDL_Texture ** texture_image_mur_mini_jeu,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_interagir,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
modes_t * modeActif,
itemMenu * itemsDemandeSauvegarde,
barreDeSon * barre_de_son,
itemMenu * pseudo,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 682 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.4 mise\_a\_jour\_bordures\_niveau\_3()

```
int mise_a_jour_bordures_niveau_3 (
```

```

SDL_Renderer * renderer,
SDL_Texture * texture_image_mur_termine,
int tilemap[24][32],
int x_tile,
int y_tile,
SDL_Rect * rectangle_tile,
int largeur_tile,
int hauteur_tile )

```

Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la fin du labyrinthe.

#### Paramètres

<i>renderer</i>	rendu de la fenêtre
<i>texture_image_mur_termine</i>	Texture du mur quand une partie ou tout le niveau est terminé
<i>tilemap</i>	map ou se trouve le personnage
<i>x_tile</i>	position x de la tuile
<i>y_tile</i>	position y de la tuile
<i>rectangle_tile</i>	rectangle à déplacer
<i>largeur_tile</i>	largeur du rectangle
<i>hauteur_tile</i>	hauteur du rectangle

#### Renvoie

appel récursif pour mettre à jour les différents bloc (termine l'appel récursif à 0)

Définition à la ligne 332 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.5 mise\_a\_jour\_mini\_jeu\_1\_niveau\_3()

```

void mise_a_jour_mini_jeu_1_niveau_3 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_puzzle,
    SDL_Rect rectangle_piece[45],
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix )

```

Définition à la ligne 76 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.6 mise\_a\_jour\_mini\_jeu\_2\_niveau\_3()

```

void mise_a_jour_mini_jeu_2_niveau_3 (
    SDL_Renderer ** renderer,
    modes_t * modeActif,
    SDL_Texture ** texture_image_sol_labyrinthe,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Texture ** texture_image_bordure_labyrinthe,
    SDL_Texture ** texture_image_mur_termine,
    SDL_Texture ** texture_image_fin_labyrinthe,
    SDL_Texture ** texture_image_croix,

```

```

SDL_Rect * rectangle_croix,
SDL_Texture ** texture,
SDL_Rect * rectangle_tile,
int bloc_x,
int bloc_y,
SDL_Texture ** texture_image_personnage,
SDL_Rect * rectangle_personnage,
int position_x,
int position_y,
int tile_map_mini_jeu_niveau_3[24][32],
niveaux * avancee_niveaux,
int largeur_tile,
int hauteur_tile )

```

Définition à la ligne 400 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.7 piece\_proche\_position\_correcte()

```

int piece_proche_position_correcte (
    SDL_Rect rectangle_piece,
    SDL_Rect rectangle_correct )

```

Fonction pour vérifier si une pièce est proche de sa position correcte.

##### Paramètres

<i>rectangle_piece</i>	rectangle représentant la pièce de puzzle à vérifier
<i>rectangle_correct</i>	rectangle représentant la bonne position de la pièce

##### Renvoie

booléen de si c'est proche de la bonne position (1 si succès sinon 0)

Définition à la ligne 113 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.15.1.8 rectangle\_piece\_aleatoire()

```

SDL_Rect rectangle_piece_aleatoire (
    int largeur,
    int hauteur )

```

Fonction pour obtenir un rectangle représentant une pièce de puzzle aléatoire.

##### Paramètres

<i>largeur</i>	représente la largeur du rectangle de la fenêtre ou se trouve le mini-jeu
<i>hauteur</i>	représente la hauteur du rectangle de la fenêtre ou se trouve le mini-jeu

Définition à la ligne 56 du fichier [fonctions\\_niveau\\_3.c](#).

## 4.15.1.9 traitement\_touches()

```

void traitement_touches (
    int * position_x,
    int * position_y,
    int * bloc_x,
    int * bloc_y,
    int tilemap[24][32],
    int direction )

```

Fonction pour traiter les commandes utilisateur.

## Paramètres

<i>position_x</i>	pointeur sur la position x du joueur
<i>position_y</i>	pointeur sur la position y du joueur
<i>bloc_x</i>	pointeur sur la position x du bloc à déplacer
<i>bloc_y</i>	pointeur sur la position y du bloc à déplacer
<i>tilemap</i>	map ou se trouve le personnage et le bloc
<i>direction</i>	booléen représentant l'action pousser / tirer

Définition à la ligne 206 du fichier [fonctions\\_niveau\\_3.c](#).

## 4.15.1.10 verification\_puzzle\_fini()

```

int verification_puzzle_fini (
    const int piece_bloquee[] )

```

Fonction pour vérifier si toutes les pièces du puzzle sont bloquées (à leur position correcte)

## Paramètres

<i>piece_bloquee</i>	tableau des pièces bloquées
----------------------	-----------------------------

## Renvoie

renvoie un booléen (1 si le puzzle est valide sinon 0)

Définition à la ligne 124 du fichier [fonctions\\_niveau\\_3.c](#).

## 4.16 fonctions\_niveau\_3.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_niveau_4.c
00003  * \brief Fichier contenant les fonctions servant à la gestion du niveau 3
00004  */

```

```

00005
00006 #include <../fichiers_h/fonctions_generales.h>
00007 #include <../fichiers_h/fonctions_niveau_3.h>
00008
00009 /**
00010  * \fn void initialisation_objets_niveau_3(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00011  **texture_image_fond_niveau_3, SDL_Texture **texture_image_dossier_niveau_3, SDL_Texture
00012  **texture_image_sol_niveau_3, SDL_Texture **barre_windows_1, SDL_Texture **barre_windows_2,
00013  SDL_Texture **barre_windows_3, SDL_Texture **barre_windows_4, SDL_Texture **texture_image_puzzle,
00014  SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture **texture_image_bordure_labyrinthe,
00015  SDL_Texture **texture_image_fin_labyrinthe)
00016  * \brief Fonction qui permet d'initialiser les différents objets du niveau 2
00017  * \param renderer Pointeur vers le renderer SDL.
00018  * \param surface Pointeur vers la surface SDL.
00019  * \param texture_image_fond_niveau_3 Texture de l'image de fond du niveau 3.
00020  * \param texture_image_dossier_niveau_3 Texture de l'image du dossier pour le niveau 3.
00021  * \param texture_image_sol_niveau_3 Texture de l'image du sol du niveau 3.
00022  * \param barre_windows_1 Texture de la barre Windows 1.
00023  * \param barre_windows_2 Texture de la barre Windows 2.
00024  * \param barre_windows_3 Texture de la barre Windows 3.
00025  * \param barre_windows_4 Texture de la barre Windows 4.
00026  * \param texture_image_puzzle Texture de l'image du puzzle.
00027  * \param texture_image_sol_labyrinthe Texture de l'image du sol du labyrinthe.
00028  * \param texture_image_bordure_labyrinthe Texture de l'image de la bordure du labyrinthe.
00029  * \param texture_image_fin_labyrinthe Texture de l'image de fin du labyrinthe.
00030  * \see chargement_image
00031  */
00032 void initialisation_objets_niveau_3(SDL_Renderer **renderer, SDL_Surface **surface,
00033 SDL_Texture **texture_image_fond_niveau_3, SDL_Texture
00034 **texture_image_dossier_niveau_3,
00035 SDL_Texture **texture_image_sol_niveau_3, SDL_Texture
00036 **barre_windows_1, SDL_Texture **barre_windows_2,
00037 SDL_Texture **barre_windows_3, SDL_Texture **barre_windows_4,
00038 SDL_Texture **texture_image_puzzle, SDL_Texture
00039 **texture_image_sol_labyrinthe,
00040 SDL_Texture **texture_image_bordure_labyrinthe, SDL_Texture
00041 **texture_image_fin_labyrinthe) {
00042
00043     /* Chargement des images pour le niveau 3 */
00044
00045     chargement_image(renderer, surface, texture_image_fond_niveau_3,
00046     "./images/niveau_3/fond_niveau_3.jpg");
00047     chargement_image(renderer, surface, texture_image_dossier_niveau_3,
00048     "./images/niveau_3/dossier_windows_xp.png");
00049     chargement_image(renderer, surface, texture_image_sol_niveau_3,
00050     "./images/niveau_3/sol_niveau_3.jpg");
00051     chargement_image(renderer, surface, barre_windows_1, "./images/niveau_3/windows_1.png");
00052     chargement_image(renderer, surface, barre_windows_2, "./images/niveau_3/windows_2.png");
00053     chargement_image(renderer, surface, barre_windows_3, "./images/niveau_3/windows_3.png");
00054     chargement_image(renderer, surface, barre_windows_4, "./images/niveau_3/windows_4.png");
00055     chargement_image(renderer, surface, texture_image_puzzle, "./images/niveau_3/ventilo.png");
00056     chargement_image(renderer, surface, texture_image_sol_labyrinthe,
00057     "./images/labyrinthe/sol_labyrinthe.png");
00058     chargement_image(renderer, surface, texture_image_bordure_labyrinthe,
00059     "./images/labyrinthe/bordure_labyrinthe.png");
00060     chargement_image(renderer, surface, texture_image_fin_labyrinthe,
00061     "./images/labyrinthe/fin_labyrinthe.png");
00062 }
00063
00064 /**
00065  * \fn SDL_Rect rectangle_piece_aleatoire(int largeur, int hauteur)
00066  * \brief Fonction pour obtenir un rectangle représentant une pièce de puzzle aléatoire
00067  * \param largeur représente la largeur du rectangle de la fenêtre ou se trouve le mini-jeu
00068  * \param hauteur représente la hauteur du rectangle de la fenêtre ou se trouve le mini-jeu
00069  */
00070 SDL_Rect rectangle_piece_aleatoire(int largeur, int hauteur) {
00071
00072     SDL_Rect rectangle_aleatoire;
00073
00074     rectangle_aleatoire.x = rand() % (largeur - largeur / 9);
00075     rectangle_aleatoire.y = rand() % (hauteur - hauteur / 5);
00076     rectangle_aleatoire.w = largeur/9;
00077     rectangle_aleatoire.h = hauteur/5;
00078
00079     return rectangle_aleatoire;
00080 }
00081
00082 /**
00083  * \fn void mise_a_jour_mini_jeu_1_niveau_3(SDL_Renderer** renderer, SDL_Texture**
00084  texture_image_puzzle, SDL_Rect rectangle_piece[45])
00085  * \brief Fonction pour afficher le rendu du puzzle
00086  * \param renderer rendu de l'écran
00087  * \param texture_image_puzzle image final du puzzle
00088  * \param rectangle_piece les différentes pièces du puzzle
00089  * \see erreur
00090  */

```

```

00076 void mise_a_jour_mini_jeu_1_niveau_3(SDL_Renderer** renderer, SDL_Texture** texture_image_puzzle,
    SDL_Rect rectangle_piece[45],
    SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix) {
00077
00078     int i;
00079
00080     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00081
00082     /* Nettoyer le rendu */
00083     if(SDL_RenderClear((*renderer)) != 0)
00084         erreur("Effacement rendu échoué");
00085
00086     for (i = 0; i < 45; i++) {
00087
00088         SDL_Rect rectangle_piece_bis = {i % 9 * 800 / 9, i / 9 * 260 / 5, 800 / 9, 260 / 5};
00089         SDL_RenderCopy((*renderer), (*texture_image_puzzle), &rectangle_piece_bis,
00090             &rectangle_piece[i]);
00091     }
00092
00093     /* Copie la texture de l'image de la croix */
00094
00095     rectangle_croix->x = 0;
00096     rectangle_croix->y = 0;
00097     rectangle_croix->w = 30;
00098     rectangle_croix->h = 30;
00099
00100     if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00101         erreur("Copie de la texture");
00102
00103     SDL_RenderPresent((*renderer));
00104 }
00105
00106 /**
00107  * \fn int piece_proche_position_correcte(SDL_Rect rectangle_piece, SDL_Rect rectangle_correct)
00108  * \brief Fonction pour vérifier si une pièce est proche de sa position correcte
00109  * \param rectangle_piece rectangle représentant la pièce de puzzle à vérifier
00110  * \param rectangle_correct rectangle représentant la bonne position de la pièce
00111  * \return booléen de si c'est proche de la bonne position (1 si succès sinon 0)
00112  */
00113 int piece_proche_position_correcte(SDL_Rect rectangle_piece, SDL_Rect rectangle_correct) {
00114
00115     return ((abs(rectangle_piece.x - rectangle_correct.x) <= 20) && (abs(rectangle_piece.y -
    rectangle_correct.y) <= 20));
00116 }
00117
00118 /**
00119  * \fn int verification_puzzle_fini(const int piece_bloquee[])
00120  * \brief Fonction pour vérifier si toutes les pièces du puzzle sont bloquées (à leur position
    correcte)
00121  * \param piece_bloquee tableau des pièces bloquées
00122  * \return renvoie un booléen (1 si le puzzle est valide sinon 0)
00123  */
00124 int verification_puzzle_fini(const int piece_bloquee[]) {
00125
00126     int i;
00127
00128     for (i = 0; i < 45; i++)
00129
00130         if (!piece_bloquee[i])
00131             return 0; /* Au moins une pièce n'est pas verrouillée */
00132
00133     return 1; /* Toutes les pièces sont verrouillées */
00134 }
00135
00136 /**
00137  * \fn void mini_jeu_2_niveau_3(int *position_x, int *position_y, int *bloc_x, int *bloc_y, int
    tile_map[24][32])
00138  * \brief Fonction qui permet d'initialiser le second mini-jeu du niveau 3
00139  * \param position_x position x du personnage
00140  * \param position_y position y du personnage
00141  * \param bloc_x position x du bloc à déplacer
00142  * \param bloc_y position y du bloc à déplacer
00143  * \param tile_map map ou se trouve le personnage
00144  */
00145
00146 void mini_jeu_2_niveau_3(int *position_x, int *position_y, int *bloc_x, int *bloc_y, int
    tile_map[24][32]) {
00147
00148     int x, y;
00149
00150     /* Positionnement du personnage en haut à gauche */
00151
00152     (*position_x) = 1;
00153     (*position_y) = 1;
00154
00155     /* Position initiale du bloc à déplacer en coordonnées de tuile */

```

```

00157
00158     (*bloc_x) = 2;
00159     (*bloc_y) = 1;
00160
00161     /* Création du second mini-jeu */
00162     int initialisation_tile_map[24][32] = {
00163         {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00164         2},
00165         {2, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
00166         2},
00167         {2, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
00168         2},
00169         {2, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
00170         2},
00171         {2, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
00172         2},
00173         {2, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0,
00174         2},
00175         {2, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
00176         2},
00177         {2, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
00178         2},
00179         {2, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
00180         2},
00181         {2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
00182         2},
00183         {2, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00184         2},
00185         {2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 4, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
00186         2},
00187         {2, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 4, 4, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
00188         2},
00189         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1,
00190         2},
00191         {2, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
00192         2},
00193         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00194         2},
00195         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00196         2},
00197         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00198         2},
00199         {2, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
00200         2},
00201         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00202         2},
00203         {2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00204         2},
00205         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00206         2},
00207         {2, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00208         2},
00209         {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00210         2}
00211     };
00212
00213     /* Copie du tilemap */
00214     for (y = 0; y < 24; y++)
00215         for (x = 0; x < 32; x++)
00216             tile_map[y][x] = initialisation_tile_map[y][x];
00217 }
00218 /**
00219  * \fn void traitement_touches(int *position_x, int *position_y, int *bloc_x, int *bloc_y, int
00220  * tilemap[24][32],int direction)
00221  * \brief Fonction pour traiter les commandes utilisateur
00222  * \param position_x pointeur sur la position x du joueur
00223  * \param position_y pointeur sur la position y du joueur
00224  * \param position_x pointeur sur la position x du bloc à déplacer
00225  * \param position_y pointeur sur la position y du bloc à déplacer
00226  * \param tilemap map ou se trouve le personnage et le bloc
00227  * \param direction booléen représentant l'action pousser / tirer
00228  */
00229 void traitement_touches(int *position_x, int *position_y, int *bloc_x, int *bloc_y, int
00230 tilemap[24][32], int direction) {
00231
00232     /* Vérification si le joueur est à côté du bloc */
00233     if ((((*position_x) == (*bloc_x) - 1) && ((*position_y) == (*bloc_y))) || /* Le joueur est à
00234     gauche du bloc */
00235         (((*position_x) == (*bloc_x) + 1) && ((*position_y) == (*bloc_y))) || /* Le joueur est à
00236     droite du bloc */
00237         (((*position_x) == (*bloc_x)) && ((*position_y) == (*bloc_y) - 1)) || /* Le joueur est au
00238     dessus du bloc */
00239         (((*position_x) == (*bloc_x)) && ((*position_y) == (*bloc_y) + 1))) { /* Le joueur est en
00240     dessous du bloc */
00241

```



```

00214      /* Le personnage est au dessus du bloc */
00215      if (((*position_x) == (*bloc_x)) && ((*position_y) == (*bloc_y) - 1)) {
00216
00217          /* On veut pousser le bloc */
00218          if(direction == 1) {
00219              /* On regarde si on peut pousser le bloc sur la case suivante */
00220              if ((tilemap[*bloc_y + 1][*bloc_x] == 0) || (tilemap[*bloc_y + 1][*bloc_x] == 3)) {
00221
00222                  (*bloc_y)++;
00223                  (*position_y)++;
00224                  SDL_Delay(150);
00225              }
00226          }
00227
00228          /* On veut tirer le bloc */
00229          if(direction == 0){
00230              /* On regarde si le personnage peut être sur la case de derrière */
00231              if ((tilemap[*position_y - 1][*position_x] == 0) || (tilemap[*position_y -
00232 1][*position_x] == 3)) {
00233
00234                  (*position_y)--;
00235                  (*bloc_y)--;
00236                  SDL_Delay(150);
00237              }
00238          }
00239
00240      /* Le personnage est en dessous du bloc */
00241      else if (((*position_x) == (*bloc_x)) && ((*position_y) == (*bloc_y + 1))) {
00242
00243          /* On veut pousser le bloc */
00244          if(direction == 0) {
00245              /* On regarde si on peut pousser le bloc sur la case suivante */
00246              if ((tilemap[*bloc_y - 1][*bloc_x] == 0) || (tilemap[*bloc_y - 1][*bloc_x] == 3)) {
00247
00248                  (*bloc_y)--;
00249                  (*position_y)--;
00250                  SDL_Delay(150);
00251              }
00252          }
00253
00254          /* On veut tirer le bloc */
00255          if(direction == 1) {
00256              /* On regarde si le personnage peut être sur la case de derrière */
00257              if ((tilemap[*position_y + 1][*position_x] == 0) || (tilemap[*position_y +
00258 1][*position_x] == 3)) {
00259
00260                  (*position_y)++;
00261                  (*bloc_y)++;
00262                  SDL_Delay(150);
00263              }
00264          }
00265
00266      /* Le personnage est à gauche du bloc */
00267      else if (((*position_x) == (*bloc_x - 1)) && ((*position_y) == (*bloc_y))) {
00268
00269          /* On veut pousser le bloc */
00270          if(direction == 3) {
00271              /* On regarde si on peut pousser le bloc sur la case suivante */
00272              if ((tilemap[*bloc_y][*bloc_x + 1] == 0) || (tilemap[*bloc_y][*bloc_x + 1] == 3)) {
00273
00274                  (*bloc_x)++;
00275                  (*position_x)++;
00276                  SDL_Delay(150);
00277              }
00278          }
00279
00280          /* On veut tirer le bloc */
00281          if(direction == 2) {
00282              /* On regarde si le personnage peut être sur la case de derrière */
00283              if ((tilemap[*position_y][*position_x - 1] == 0) || (tilemap[*position_y][*position_x
00284 - 1] == 3)) {
00285
00286                  (*position_x)--;
00287                  (*bloc_x)--;
00288                  SDL_Delay(150);
00289              }
00290          }
00291
00292      /* Le personnage est à droite du bloc */
00293      else if (((*position_x) == (*bloc_x + 1)) && ((*position_y) == (*bloc_y))) {
00294
00295          /* On veut pousser le bloc */
00296          if(direction == 2) {
00297              /* On regarde si on peut pousser le bloc sur la case suivante */

```

```

00298             if ((tilemap[*bloc_y][*bloc_x - 1] == 0) || (tilemap[*bloc_y][*bloc_x - 1] == 3)) {
00299
00300                 (*bloc_x)--;
00301                 (*position_x)--;
00302                 SDL_Delay(150);
00303             }
00304         }
00305
00306         /* On veut tirer le bloc */
00307         if(direction == 3) {
00308             /* On regarde si le personnage peut être sur la case de derrière */
00309             if ((tilemap[*position_y][*position_x + 1] == 0) || (tilemap[*position_y][*position_x
+ 1] == 3)) {
00310                 (*position_x)++;
00311                 (*bloc_x)++;
00312                 SDL_Delay(150);
00313             }
00314         }
00315     }
00316 }
00317 }
00318
00319 /**
00320  * \fn int mise_a_jour_bordures_niveau_3(SDL_Renderer* renderer, SDL_Texture*
texture_image_mur_termine, int tilemap[24][32], int x_tile, int y_tile, SDL_Rect *rectangle_tile, int
largeur_tile, int hauteur_tile)
00321  * \brief Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la fin du
labyrinthe
00322  * \param renderer rendu de la fenêtre
00323  * \param texture_image_mur_termine Texture du mur quand une partie ou tout le niveau est terminé
00324  * \param tilemap map ou se trouve le personnage
00325  * \param x_tile position x de la tuile
00326  * \param y_tile position y de la tuile
00327  * \param rectangle_tile rectangle à déplacer
00328  * \param largeur_tile largeur du rectangle
00329  * \param hauteur_tile hauteur du rectangle
00330  * \return appel récursif pour mettre à jour les différents bloc (termine l'appel récursif à 0)
00331  */
00332 int mise_a_jour_bordures_niveau_3(SDL_Renderer* renderer, SDL_Texture* texture_image_mur_termine, int
tilemap[24][32], int x_tile, int y_tile,
00333     SDL_Rect *rectangle_tile, int largeur_tile, int hauteur_tile) {
00334
00335     /* Mise à jour du rendu de la tuile courante */
00336     tilemap[y_tile][x_tile] = 3;
00337
00338     rectangle_tile->x = x_tile * largeur_tile;
00339     rectangle_tile->y = y_tile * hauteur_tile;
00340     rectangle_tile->w = largeur_tile;
00341     rectangle_tile->h = hauteur_tile;
00342
00343     SDL_RenderCopy(renderer, texture_image_mur_termine, NULL, rectangle_tile);
00344
00345     SDL_RenderPresent(renderer);
00346
00347     SDL_Delay(20);
00348
00349     /* Vérification des tuiles adjacentes pour permettre un appel récursif et ainsi changé toute les
tuiles de la bordure */
00350
00351     /* Tuile à droite de la courante */
00352     if((tilemap[y_tile][x_tile + 1] == 2) && (x_tile < 31))
00353         return mise_a_jour_bordures_niveau_3(renderer, texture_image_mur_termine, tilemap, x_tile + 1,
y_tile,
00354             rectangle_tile, largeur_tile, hauteur_tile);
00355
00356     /* Tuile à gauche de la courante */
00357     else if((tilemap[y_tile][x_tile - 1] == 2) && (x_tile > 0))
00358         return mise_a_jour_bordures_niveau_3(renderer, texture_image_mur_termine, tilemap, x_tile - 1,
y_tile,
00359             rectangle_tile, largeur_tile, hauteur_tile);
00360
00361     /* Tuile en dessous de la courante */
00362     else if((tilemap[y_tile + 1][x_tile] == 2) && (y_tile < 31))
00363         return mise_a_jour_bordures_niveau_3(renderer, texture_image_mur_termine, tilemap, x_tile,
y_tile + 1,
00364             rectangle_tile, largeur_tile, hauteur_tile);
00365
00366     /* Tuile au dessus de la courante */
00367     else if((tilemap[y_tile - 1][x_tile] == 2) && (y_tile > 0))
00368         return mise_a_jour_bordures_niveau_3(renderer, texture_image_mur_termine, tilemap, x_tile,
y_tile - 1,
00369             rectangle_tile, largeur_tile, hauteur_tile);
00370
00371     return 0;
00372 }
00373
00374 /**

```

```

00375 * \fn void mise_a_jour_mini_jeu_2_niveau_3(SDL_Renderer **renderer, modes_t *modeActif, SDL_Texture
**texture_image_sol_labyrinthe, SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran, SDL_Texture **texture_image_mur_mini_jeu, SDL_Texture
**texture_image_bordure_labyrinthe, SDL_Texture **texture_image_mur_termine, SDL_Texture
**texture_image_fin_labyrinthe, SDL_Texture **texture, SDL_Rect *rectangle_tile, int bloc_x, int
bloc_y, SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int position_x, int
position_y, int tile_map_mini_jeu_niveau_3[24][32], niveaux *avancee_niveaux, int largeur_tile, int
hauteur_tile)
00376 * \brief Fonction qui permet de mettre à jour le second mini-jeu du niveau 2
00377 * \param renderer Pointeur vers le renderer SDL.
00378 * \param modeActif Mode actif du jeu.
00379 * \param texture_image_sol_labyrinthe Texture de l'image du sol du labyrinthe.
00380 * \param rectangle_plein_ecran Rectangle représentant l'écran complet.
00381 * \param texture_image_plein_ecran Texture de l'image de l'écran complet.
00382 * \param texture_image_mur_mini_jeu Texture de l'image du mur du mini-jeu.
00383 * \param texture_image_bordure_labyrinthe Texture de l'image de la bordure du labyrinthe.
00384 * \param texture_image_mur_termine Texture de l'image du mur terminé.
00385 * \param texture_image_fin_labyrinthe Texture de l'image de fin du labyrinthe.
00386 * \param texture Texture utilisée pour le rendu.
00387 * \param rectangle_tile Rectangle représentant une tuile.
00388 * \param bloc_x Position X du bloc.
00389 * \param bloc_y Position Y du bloc.
00390 * \param texture_image_personnage Texture de l'image du personnage.
00391 * \param rectangle_personnage Rectangle représentant le personnage.
00392 * \param position_x Position X actuelle.
00393 * \param position_y Position Y actuelle.
00394 * \param tile_map_mini_jeu_niveau_3 Tableau de tuiles pour le mini-jeu 2 du niveau 3.
00395 * \param avancee_niveaux Structure de données représentant l'avancée dans les niveaux.
00396 * \param largeur_tile Largeur d'une tuile.
00397 * \param hauteur_tile Hauteur d'une tuile.
00398 *
00399 */
00400 void mise_a_jour_mini_jeu_2_niveau_3(SDL_Renderer **renderer, modes_t *modeActif, SDL_Texture
**texture_image_sol_labyrinthe,
00401                                     SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00402                                     SDL_Texture **texture_image_mur_mini_jeu, SDL_Texture
**texture_image_bordure_labyrinthe, SDL_Texture **texture_image_mur_termine,
00403                                     SDL_Texture **texture_image_fin_labyrinthe, SDL_Texture
**texture_image_croix, SDL_Rect *rectangle_croix,
00404                                     SDL_Texture **texture, SDL_Rect *rectangle_tile, int bloc_x, int
bloc_y,
00405                                     SDL_Texture **texture_image_personnage, SDL_Rect
*rectangle_personnage,
00406                                     int position_x, int position_y, int
tile_map_mini_jeu_niveau_3[24][32], niveaux *avancee_niveaux,
00407                                     int largeur_tile, int hauteur_tile) {
00408
00409     int x, y;
00410
00411     Mix_Chunk *effet_sonore = NULL;
00412
00413     /* Nettoyer le renderer */
00414     SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00415
00416     /* Efface le rendu */
00417     if(SDL_RenderClear((*renderer)) != 0)
00418         erreur("Effacement rendu échoué");
00419
00420     /* Rendu tilemap */
00421     /* Mode Normal */
00422     if((*modeActif) == MODE_NORMAL)
00423         for(y = 0; y < 24; y++)
00424             for(x = 0; x < 32; x++) {
00425
00426                 /* Rendu de chaque tuile en fonction de son type */
00427
00428                 if((!tile_map_mini_jeu_niveau_3[y][x]) || (tile_map_mini_jeu_niveau_3[y][x] == 5))
00429                     (*texture) = (*texture_image_sol_labyrinthe);
00430
00431                 else if(tile_map_mini_jeu_niveau_3[y][x] == 1)
00432                     (*texture) = (*texture_image_mur_mini_jeu);
00433
00434                 else if(tile_map_mini_jeu_niveau_3[y][x] == 2)
00435                     (*texture) = (*texture_image_bordure_labyrinthe);
00436
00437                 else if(tile_map_mini_jeu_niveau_3[y][x] == 3)
00438                     (*texture) = (*texture_image_mur_termine);
00439
00440                 else if(tile_map_mini_jeu_niveau_3[y][x] == 4)
00441                     (*texture) = (*texture_image_mur_mini_jeu);
00442
00443                 rectangle_tile->x = x * largeur_tile;
00444                 rectangle_tile->y = y * hauteur_tile;
00445                 rectangle_tile->w = largeur_tile;
00446                 rectangle_tile->h = hauteur_tile;
00447

```

```

00448         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00449             erreur("Copie de la texture");
00450
00451         if((tile_map_mini_jeu_niveau_3[y][x] == 5) &&
00452            (avancee_niveaux[2].numero_collectible[1] == 0))
00453             if(SDL_RenderCopy((*renderer), avancee_niveaux[2].texture_image_collectible, NULL,
rectangle_tile) != 0)
00454                 erreur("Copie de la texture");
00455     }
00456     /* Mode Difficile */
00457     else if((*modeActif) == MODE_HARD)
00458         for(y = position_y - 2; y <= position_y + 2; y++)
00459             for(x = position_x - 2; x <= position_x + 2; x++) {
00460
00461                 /* Rendu de chaque tuile en fonction de son type */
00462
00463                 if(!tile_map_mini_jeu_niveau_3[y][x] || (tile_map_mini_jeu_niveau_3[y][x] == 5))
00464                     (*texture) = (*texture_image_sol_labyrinthe);
00465
00466                 else if(tile_map_mini_jeu_niveau_3[y][x] == 1)
00467                     (*texture) = (*texture_image_mur_mini_jeu);
00468
00469                 else if(tile_map_mini_jeu_niveau_3[y][x] == 2)
00470                     (*texture) = (*texture_image_bordure_labyrinthe);
00471
00472                 else if(tile_map_mini_jeu_niveau_3[y][x] == 3)
00473                     (*texture) = (*texture_image_mur_termine);
00474
00475                 else if(tile_map_mini_jeu_niveau_3[y][x] == 4)
00476                     (*texture) = (*texture_image_mur_mini_jeu);
00477
00478                 rectangle_tile->x = x * largeur_tile;
00479                 rectangle_tile->y = y * hauteur_tile;
00480                 rectangle_tile->w = largeur_tile;
00481                 rectangle_tile->h = hauteur_tile;
00482
00483                 if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00484                     erreur("Copie de la texture");
00485
00486                 if((tile_map_mini_jeu_niveau_3[y][x] == 5) &&
00487                    (avancee_niveaux[2].numero_collectible[1] == 0))
00488                     if(SDL_RenderCopy((*renderer), avancee_niveaux[2].texture_image_collectible, NULL,
rectangle_tile) != 0)
00489                         erreur("Copie de la texture");
00490             }
00491     /* Passage secret vers un collectible */
00492     if((position_x == 12) && (position_y == 11) && (tile_map_mini_jeu_niveau_3[position_y][position_x]
== 4)){
00493
00494         /* Effet sonore quand on découvre le passage secret */
00495         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/nathan.wav")) == NULL)
00496             erreur("Chargement de l'effet sonore");
00497
00498         Mix_PlayChannel(1, effet_sonore, 0);
00499
00500         tile_map_mini_jeu_niveau_3[11][12] = 0;
00501         tile_map_mini_jeu_niveau_3[12][12] = 0;
00502         tile_map_mini_jeu_niveau_3[12][13] = 5;
00503
00504         (*texture) = (*texture_image_sol_labyrinthe);
00505
00506         rectangle_tile->x = 12 * largeur_tile;
00507         rectangle_tile->y = 11 * hauteur_tile;
00508         rectangle_tile->w = largeur_tile;
00509         rectangle_tile->h = hauteur_tile;
00510
00511         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00512             erreur("Copie de la texture");
00513
00514         rectangle_tile->y = 12 * hauteur_tile;
00515
00516         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00517             erreur("Copie de la texture");
00518
00519         rectangle_tile->x = 13 * largeur_tile;
00520
00521         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00522             erreur("Copie de la texture");
00523     }
00524
00525     /* Remise à l'état initial */
00526     if((position_x == 12) && (position_y == 10) && (!tile_map_mini_jeu_niveau_3[11][12])){
00527
00528         tile_map_mini_jeu_niveau_3[11][12] = 4;
00529         tile_map_mini_jeu_niveau_3[12][12] = 4;

```

```

00530         tile_map_mini_jeu_niveau_3[12][13] = 4;
00531
00532         (*texture) = (*texture_image_mur_mini_jeu);
00533
00534         rectangle_tile->x = 12 * largeur_tile;
00535         rectangle_tile->y = 11 * hauteur_tile;
00536         rectangle_tile->w = largeur_tile;
00537         rectangle_tile->h = hauteur_tile;
00538
00539         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00540             erreur("Copie de la texture");
00541
00542         rectangle_tile->y = 12 * hauteur_tile;
00543
00544         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00545             erreur("Copie de la texture");
00546
00547         rectangle_tile->x = 13 * largeur_tile;
00548
00549         if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile))
00550             erreur("Copie de la texture");
00551     }
00552
00553     /* Rendu du bloc à déplacer */
00554
00555     rectangle_tile->x = bloc_x * largeur_tile;
00556     rectangle_tile->y = bloc_y * hauteur_tile;
00557     rectangle_tile->w = largeur_tile;
00558     rectangle_tile->h = hauteur_tile;
00559
00560     if(SDL_RenderCopy((*renderer), (*texture_image_fin_labyrinthe), NULL, rectangle_tile))
00561         erreur("Copie de la texture");
00562
00563     /* Rendu du bloc d'arriver */
00564
00565     rectangle_tile->x = 30 * largeur_tile;
00566     rectangle_tile->y = 22 * hauteur_tile;
00567
00568     if(SDL_RenderCopy((*renderer), (*texture_image_mur_termine), NULL, rectangle_tile))
00569         erreur("Copie de la texture");
00570
00571     /* Rendu du joueur */
00572
00573     rectangle_personnage->x = position_x * largeur_tile;
00574     rectangle_personnage->y = position_y * hauteur_tile;
00575     rectangle_personnage->w = largeur_tile;
00576     rectangle_personnage->h = hauteur_tile;
00577
00578     if(SDL_RenderCopy((*renderer), (*texture_image_personnage), NULL, rectangle_personnage))
00579         erreur("Copie de la texture");
00580
00581     /* Copie la texture de l'image de plein écran */
00582
00583     rectangle_plein_ecran->x = largeur_tile * 31;
00584     rectangle_plein_ecran->y = 0;
00585     rectangle_plein_ecran->w = largeur_tile;
00586     rectangle_plein_ecran->h = hauteur_tile;
00587
00588     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00589         erreur("Copie de la texture");
00590
00591     /* Copie la texture de l'image de la croix */
00592
00593     rectangle_croix->x = 0;
00594     rectangle_croix->y = 0;
00595     rectangle_croix->w = largeur_tile;
00596     rectangle_croix->h = hauteur_tile;
00597
00598     if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00599         erreur("Copie de la texture");
00600
00601     /* Affiche le rendu */
00602     SDL_RenderPresent((*renderer));
00603 }
00604
00605 /**
00606 * \fn void mini_jeux_niveau_3(SDL_Event *event, SDL_Renderer **renderer, SDL_Window **window, SDL_Rect
00607 *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool *plein_ecran, niveaux
00608 *avancee_niveaux, int tile_map[18][32], int *mini_jeu, int *mini_jeu_l_termine, int
00609 *mini_jeu_2_termine, int *position_x, int *position_y, SDL_Texture **texture, int *largeur, int
00610 *hauteur, SDL_Rect *rectangle_demande_quitter, SDL_Surface **surface, SDL_Texture **texture_texte,
00611 TTF_Font **police, SDL_Color couleurNoire, itemMenu *itemsDemandeQuitter, int tailleDemandeQuitter, int
00612 collectibles_intermediaires[3], page_t *page_active, SDL_Rect *rectangle_tile, int *largeur_tile, int
00613 *hauteur_tile, int *avancer, int *reculer, int *sauter, int *saut, int *tombe, SDL_Rect
00614 rectangle_piece[45], int piece_bloquee[45], SDL_Rect rectangle_emplacement_piece[45], int

```

```

    *piece_selectionnee,int *decalage_x, int *decalage_y, SDL_Texture **texture_image_puzzle,int
    tile_map_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int *bloc_x, int
    *bloc_y,SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture
    **texture_image_bordure_labyrinthe,SDL_Texture **texture_image_fin_labyrinthe, Mix_Music
    **musique,SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage,SDL_Texture
    **texture_image_mur_termine, SDL_Texture **texture_image_mur_mini_jeu, SDL_Keycode
    *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_interagir,SDL_Keycode
    *touche_sauter_monter, SDL_Keycode *touche_descendre, modes_t *modeActif)
00609 * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans les mini-jeux
    du niveau 3
00610 * \param event Pointeur vers l'événement SDL.
00611 * \param renderer Pointeur vers le renderer SDL.
00612 * \param window Pointeur vers la fenêtre SDL.
00613 * \param rectangle_plein_ecran Rectangle représentant l'écran complet.
00614 * \param texture_image_plein_ecran Texture de l'image de l'écran complet.
00615 * \param plein_ecran Booléen indiquant si le mode plein écran est activé.
00616 * \param avancee_niveaux Structure de données représentant l'avancée dans les niveaux.
00617 * \param tile_map Tableau de tuiles du niveau 3.
00618 * \param mini_jeu Pointeur vers le numéro du mini-jeu en cours.
00619 * \param mini_jeu_1_termine Booléen indiquant si le premier mini-jeu est terminé.
00620 * \param mini_jeu_2_termine Booléen indiquant si le deuxième mini-jeu est terminé.
00621 * \param position_x Position X du joueur.
00622 * \param position_y Position Y du joueur.
00623 * \param texture Texture utilisée pour le rendu.
00624 * \param largeur Largeur de l'écran.
00625 * \param hauteur Hauteur de l'écran.
00626 * \param rectangle_demande_quitter Rectangle pour la demande de quitter.
00627 * \param surface Surface SDL pour le texte.
00628 * \param texture_texte Texture pour le texte.
00629 * \param police Police utilisée pour le texte.
00630 * \param couleurNoire Couleur noire.
00631 * \param itemsDemandeQuitter Tableau d'items pour la demande de quitter.
00632 * \param tailleDemandeQuitter Taille du tableau d'items de demande de quitter.
00633 * \param collectibles_intermediaires Tableau des collectibles intermédiaires.
00634 * \param page_active Page active dans le jeu.
00635 * \param rectangle_tile Rectangle représentant une tuile.
00636 * \param largeur_tile Largeur d'une tuile.
00637 * \param hauteur_tile Hauteur d'une tuile.
00638 * \param avancer Booléen indiquant si le personnage avance.
00639 * \param reculer Booléen indiquant si le personnage recule.
00640 * \param sauter Booléen indiquant si le personnage saute.
00641 * \param saut Booléen indiquant si le personnage est en saut.
00642 * \param tombe Booléen indiquant si le personnage tombe.
00643 * \param rectangle_piece Rectangle représentant une pièce.
00644 * \param piece_bloquee Tableau indiquant si une pièce est bloquée.
00645 * \param rectangle_emplacement_piece Rectangle représentant un emplacement de pièce.
00646 * \param piece_selectionnee Indice de la pièce sélectionnée.
00647 * \param decalage_x Décalage en X.
00648 * \param decalage_y Décalage en Y.
00649 * \param texture_image_puzzle Texture de l'image du puzzle.
00650 * \param tile_map_mini_jeu_niveau_3 Tableau de tuiles pour le mini-jeu 3 du niveau 3.
00651 * \param descendre Booléen indiquant si le personnage descend.
00652 * \param interagir Booléen indiquant si le personnage interagit.
00653 * \param bloc_x Position X du bloc.
00654 * \param bloc_y Position Y du bloc.
00655 * \param texture_image_sol_labyrinthe Texture de l'image du sol du labyrinthe.
00656 * \param texture_image_bordure_labyrinthe Texture de l'image de la bordure du labyrinthe.
00657 * \param texture_image_fin_labyrinthe Texture de l'image de fin du labyrinthe.
00658 * \param musique Pointeur vers la musique.
00659 * \param texture_image_personnage Texture de l'image du personnage.
00660 * \param rectangle_personnage Rectangle représentant le personnage.
00661 * \param texture_image_mur_termine Texture de l'image du mur terminé.
00662 * \param texture_image_mur_mini_jeu Texture de l'image du mur du mini-jeu.
00663 * \param touche_aller_a_droite Touche pour aller à droite.
00664 * \param touche_aller_a_gauche Touche pour aller à gauche.
00665 * \param touche_interagir Touche pour interagir.
00666 * \param touche_sauter_monter Touche pour sauter/monter.
00667 * \param touche_descendre Touche pour descendre.
00668 * \param modeActif Mode actif du jeu.
00669 * \see clic_case
00670 * \see clic_plein_ecran
00671 * \see demande_quitter_niveau
00672 * \see redimensionnement_fenetre
00673 * \see erreur
00674 * \see traitement_touches
00675 * \see piece_proche_position_correcte
00676 * \see verification_puzzle_fini
00677 * \see salon_arrivee_niveaux_2_3
00678 * \see mise_a_jour_bordures_niveau_3
00679 * \see mise_a_jour_mini_jeu_1_niveau_3
00680 * \see mise_a_jour_mini_jeu_2_niveau_3
00681 */
00682 void mini_jeux_niveau_3(SDL_Event *event, SDL_Renderer **renderer, SDL_Window **window, SDL_bool
    *programme_lance,
00683                        SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
    SDL_bool *plein_ecran,
00684                        niveaux *avancee_niveaux, int tile_map[18][32], SDL_Texture

```

```

    **texture_image_croix, SDL_Rect *rectangle_croix,
00685         int *mini_jeu, int *mini_jeu_1_termine, int *mini_jeu_2_termine,
00686         int *position_x, int *position_y, SDL_Texture **texture,
00687         int *largeur, int *hauteur, SDL_Rect *rectangle_demande,
00688         SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
    SDL_Color couleurNoire,
00689         itemMenu *itemsDemandeQuitter, int tailleDemande, int
    collectibles_intermediaires[3],
00690         page_t *page_active, SDL_Rect *rectangle_tile, int *largeur_tile, int
    *hauteur_tile,
00691         int *avancer, int *reculer, int *sauter, int *saut, int *tombe,
00692         SDL_Rect rectangle_piece[45], int piece_bloquee[45], SDL_Rect
    rectangle_emplacement_piece[45], int *piece_selectionnee,
00693         int *decalage_x, int *decalage_y, SDL_Texture **texture_image_puzzle,
00694         int tile_map_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int
    *bloc_x, int *bloc_y,
00695         SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture
    **texture_image_bordure_labyrinthe,
00696         SDL_Texture **texture_image_fin_labyrinthe, Mix_Music **musique,
00697         SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage,
00698         SDL_Texture **texture_image_mur_termine, SDL_Texture
    **texture_image_mur_mini_jeu,
00699         SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
    SDL_Keycode *touche_interagir,
00700         SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, modes_t
    *modeActif,
00701         itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
00702         personnage_t *personnageActif, position_t *positionActive, int tailleNiveaux,
00703         time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
    avancee_succes_intermediaires[10]) {
00704
00705     int i;
00706
00707     SDL_Event event_temporaire;
00708     SDL_bool clic_effectue = SDL_FALSE;
00709
00710     Mix_Chunk *effet_sonore = NULL;
00711
00712     /* Cas où on se trouve dans le premier mini-jeu */
00713     if ((*mini_jeu) == 1) {
00714
00715         while (SDL_PollEvent(&event) != 0) {
00716
00717             switch(event->type) {
00718
00719                 case SDL_MOUSEMOTION:
00720
00721                     if ((*piece_selectionnee) != -1) {
00722
00723                         (*position_x) = event->motion.x;
00724                         (*position_y) = event->motion.y;
00725
00726                         /* Assurer que la pièce ne sort pas de l'écran */
00727                         rectangle_piece[(*piece_selectionnee)].x = (*position_x) - (*decalage_x);
00728                         rectangle_piece[(*piece_selectionnee)].y = (*position_y) - (*decalage_y);
00729
00730                         /* Correction pour empêcher la pièce de sortir de l'écran */
00731                         if(rectangle_piece[(*piece_selectionnee)].x < 0)
00732                             rectangle_piece[(*piece_selectionnee)].x = 0;
00733
00734                         else if(rectangle_piece[(*piece_selectionnee)].x > 1600 - 1600 / 9)
00735                             rectangle_piece[(*piece_selectionnee)].x = 1600 - 1600 / 9;
00736
00737                         if(rectangle_piece[(*piece_selectionnee)].y < 0)
00738                             rectangle_piece[(*piece_selectionnee)].y = 0;
00739
00740                         else if(rectangle_piece[(*piece_selectionnee)].y > 520 - 520 / 5)
00741                             rectangle_piece[(*piece_selectionnee)].y = 520 - 520 / 5;
00742
00743                     }
00744
00745                     break;
00746
00747                 case SDL_MOUSEBUTTONDOWN:
00748
00749                     if ((*piece_selectionnee) != -1) {
00750
00751                         /* Vérifier si la pièce est à proximité de sa position correcte */
00752                         if (piece_proche_position_correcte(rectangle_piece[(*piece_selectionnee)],
    rectangle_emplacement_piece[(*piece_selectionnee)])) {
00753
00754                             /* Déplacer la pièce à sa position correcte et bloquer son mouvement */
00755                             rectangle_piece[(*piece_selectionnee)] =
    rectangle_emplacement_piece[(*piece_selectionnee)];
00756                             piece_bloquee[(*piece_selectionnee)] = 1; /* Verrouiller la pièce */
00757                         }
00758

```

```

00759             (*piece_selectionnee) = -1; /* Réinitialiser la pièce sélectionnée */
00760         }
00761
00762         break;
00763
00764         case SDL_MOUSEBUTTONDOWN :
00765             SDL_GetMouseState(position_x, position_y);
00766
00767             for (i = 44; i >= 0; i--)
00768
00769                 /* Vérifier si l'utilisateur a cliqué sur une pièce non bloquée */
00770                 if ((!piece_bloquee[i]) && ((*position_x) >= rectangle_piece[i].x) &&
00771                     ((*position_x) < rectangle_piece[i].x + 1600 / 9) &&
00772                     ((*position_y) >= rectangle_piece[i].y) && ((*position_y) <
rectangle_piece[i].y + 520 / 5)) {
00773
00774                     /* Sélectionner la pièce et calculer le décalage de position */
00775                     (*piece_selectionnee) = i;
00776
00777                     (*decalage_x) = (*position_x) - rectangle_piece[i].x;
00778                     (*decalage_y) = (*position_y) - rectangle_piece[i].y;
00779
00780                     break;
00781             }
00782
00783             /* Demande au joueur s'il veut quitter le niveau */
00784             if(clic_case((*event), (*rectangle_croix))) {
00785
00786                 demande_quitter_niveau(renderer, rectangle_demande,
00787                                         surface, texture_texte, police, couleurNoire,
00788                                         itemsDemandeQuitter, tailleDemande, 1600, 520);
00789
00790                 while (!clic_effectue) {
00791
00792                     while (SDL_PollEvent(&event_temporaire)) {
00793
00794                         if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00795
00796                             if(clic_case(event_temporaire, itemsDemandeQuitter[1].rectangle))
00797
00798                                 {
00799                                     for(i = 0; i < 3; i++)
00800                                         avancee_niveaux[2].numero_collectible[i] =
collectibles_intermediaires[i];
00801
00802                                     for(i = 0; i < 10; i++)
00803                                         avancee_succes[i] = avancee_succes_intermediaires[i];
00804
00805                                     (*largeur) = 960;
00806                                     (*hauteur) = 540;
00807
00808                                     SDL_SetWindowSize((*window), (*largeur), (*hauteur));
00809                                     SDL_SetWindowPosition((*window), SDL_WINDOWPOS_CENTERED,
SDL_WINDOWPOS_CENTERED);
00810
00811                                     (*page_active) = CARTE;
00812                                     clic_effectue = SDL_TRUE;
00813                                 }
00814
00815                                 else if(clic_case(event_temporaire,
itemsDemandeQuitter[2].rectangle))
00816                                     clic_effectue = SDL_TRUE;
00817                                 }
00818                             }
00819                         }
00820
00821                     SDL_SetWindowResizable((*window), SDL_TRUE);
00822                 }
00823
00824                 break;
00825
00826             /* Quitter le programme en demandant s'il faut sauvegarder la partie */
00827             case SDL_QUIT:
00828
00829                 demande_sauvegarde(renderer, rectangle_demande,
00830                                     surface, texture_texte, police, couleurNoire,
00831                                     itemsDemandeSauvegarde, tailleDemande, 1600, 520);
00832
00833                 while (!clic_effectue) {
00834
00835                     while (SDL_PollEvent(&event_temporaire)) {
00836
00837                         if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00838
00839                             if(clic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {

```



```

00840
00841                                     for(i = 0; i < 3; i++)
00842                                     avancee_niveaux[2].numero_collectible[i] =
00843                                     collectibles_intermediaires[i];
00844                                     sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
00845                                     touche_sauter_monter,
00846                                     barre_de_son, pseudo,
00847                                     (*modeActif), (*personnageActif),
00848                                     temps_debut_partie, (*compteur_mort), avancee_succes);
00849                                     (*programme_lance) = SDL_FALSE;
00850                                     clic_effectue = SDL_TRUE;
00851                                     }
00852
00853                                     else if(clic_case(event_temporaire,
00854                                     itemsDemandeSauvegarde[2].rectangle)) {
00855                                     (*programme_lance) = SDL_FALSE;
00856                                     clic_effectue = SDL_TRUE;
00857                                     }
00858                                     else if(!clic_case(event_temporaire, (*rectangle_demande)))
00859                                     clic_effectue = SDL_TRUE;
00860                                     }
00861                                     }
00862                                     }
00863
00864                                     break;
00865
00866                                     default:
00867                                     break;
00868                                     }
00869                                     }
00870
00871                                     /* Vérifier si toutes les pièces sont bloquées à leur position correcte */
00872                                     if (verification_puzzle_fini(piece_bloquee)) {
00873
00874                                     if ((*musique) = Mix_LoadMUS("./sons/musiques/salon.mp3")) == NULL)
00875                                     erreur("Chargement de la musique");
00876
00877                                     Mix_PlayMusic((*musique), -1);
00878
00879                                     (*mini_jeu) = 0;
00880                                     (*mini_jeu_1_termine) = 1;
00881
00882                                     (*sauter) = 0;
00883                                     (*avancer) = 0;
00884                                     (*reculer) = 0;
00885                                     (*tombe) = 0;
00886                                     (*saut) = 0;
00887
00888                                     salon_arrivee_niveaux_2_3(position_x, position_y, tile_map, (*page_active));
00889
00890                                     tile_map[3][5] = 0;
00891                                     tile_map[5][6] = 5;
00892
00893                                     if ((*mini_jeu_2_termine))
00894                                     tile_map[3][24] = 0;
00895
00896                                     (*largeur) = 960;
00897                                     (*hauteur) = 540;
00898
00899                                     SDL_SetWindowSize((*window), (*largeur), (*hauteur));
00900                                     SDL_SetWindowPosition((*window), SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED);
00901
00902                                     SDL_SetWindowResizable((*window), SDL_TRUE);
00903                                     }
00904
00905                                     /* Mise à jour du rendu */
00906                                     mise_a_jour_mini_jeu_1_niveau_3(renderer, texture_image_puzzle, rectangle_piece,
00907                                     texture_image_croix, rectangle_croix);
00908                                     }
00909
00910                                     /* Cas où on se trouve dans le second mini-jeu */
00911                                     else if ((*mini_jeu) == 2) {
00912
00913                                     while (SDL_PollEvent(event)) {
00914
00915                                     switch(event->type) {
00916
00917                                     /* Gestion de l'événement de redimensionnement de la fenêtre */
00918                                     case SDL_WINDOWEVENT:
00919                                     redimensionnement_fenetre((*event), largeur, hauteur);
00920

```

```

00921         (*largeur_tile) = (*largeur) / 32;
00922         (*hauteur_tile) = (*hauteur) / 24;
00923
00924         break;
00925
00926     /*
00927     On peut pas traiter l'appui de 2 touches en même temps donc
00928     On stocke une valeur pour permettre l'appui de 2 touches en même
00929     Et ainsi pouvoir traiter cette information
00930     */
00931     case SDL_KEYDOWN:
00932
00933         if(event->key.keysym.sym == (*touche_sauter_monter))
00934             (*sauter) = 1;
00935
00936         if(event->key.keysym.sym == (*touche_descendre))
00937             (*descendre) = 1;
00938
00939         if(event->key.keysym.sym == (*touche_aller_a_gauche))
00940             (*reculer) = 1;
00941
00942         if(event->key.keysym.sym == (*touche_aller_a_droite))
00943             (*avancer) = 1;
00944
00945         if(event->key.keysym.sym == (*touche_interagir))
00946             (*interagir) = 1;
00947
00948         break;
00949
00950     case SDL_KEYUP:
00951
00952         if(event->key.keysym.sym == (*touche_sauter_monter))
00953             (*sauter) = 0;
00954
00955         if(event->key.keysym.sym == (*touche_descendre))
00956             (*descendre) = 0;
00957
00958         if(event->key.keysym.sym == (*touche_aller_a_gauche))
00959             (*reculer) = 0;
00960
00961         if(event->key.keysym.sym == (*touche_aller_a_droite))
00962             (*avancer) = 0;
00963
00964         if(event->key.keysym.sym == (*touche_interagir))
00965             (*interagir) = 0;
00966
00967         break;
00968
00969     /* Option plein écran */
00970     case SDL_MOUSEBUTTONDOWN:
00971
00972         if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window)) {
00973
00974             redimensionnement_fenetre((*event), largeur, hauteur);
00975
00976             (*largeur_tile) = (*largeur) / 32;
00977             (*hauteur_tile) = (*hauteur) / 24;
00978         }
00979
00980         /* Demande au joueur s'il veut quitter le niveau */
00981         if(clic_case((*event), (*rectangle_croix))) {
00982
00983             SDL_SetWindowResizable((*window), SDL_FALSE);
00984
00985             demande_quitter_niveau(renderer, rectangle_demande,
00986                                     surface, texture_texte, police, couleurNoire,
00987                                     itemsDemandeQuitter, tailleDemande, (*largeur),
00988                                     (*hauteur));
00989
00990             while (!clic_effectue) {
00991
00992                 while (SDL_PollEvent(&event_temporaire)) {
00993
00994                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00995
00996                         if(clic_case(event_temporaire, itemsDemandeQuitter[1].rectangle))
00997
00998                             {
00999
01000                                 for(i = 0; i < 3; i++)
01001                                     avancee_niveaux[2].numero_collectible[i] =
01002                                     collectibles_intermediaires[i];
01003
01004                                 for(i = 0; i < 10; i++)
01005                                     avancee_succes[i] = avancee_succes_intermediaires[i];
01006
01007                                 (*page_active) = CARTE;
01008                             }
01009

```

```

01005             clic_effectue = SDL_TRUE;
01006         }
01007     }
01008     else if(clic_case(event_temporaire,
itemsDemandeQuitter[2].rectangle))
01009         clic_effectue = SDL_TRUE;
01010     }
01011 }
01012 }
01013
01014     SDL_SetWindowResizable((*window), SDL_TRUE);
01015 }
01016
01017     break;
01018
01019     /* Quitter le programme en demandant s'il faut sauvergar la partie */
01020     case SDL_QUIT:
01021
01022         SDL_SetWindowResizable((*window), SDL_FALSE);
01023
01024         demande_sauvegarde(renderer, rectangle_demande,
01025             surface, texture_texte, police, couleurNoire,
01026             itemsDemandeSauvegarde, tailleDemande, (*largeur),
01027             (*hauteur));
01028
01029         while (!clic_effectue) {
01030             while (SDL_PollEvent(&event_temporaire)) {
01031                 if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
01032                     if(clic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {
01033                         for(i = 0; i < 3; i++)
01034                             avancee_niveaux[2].numero_collectible[i] =
01035                                 collectibles_intermediaires[i];
01036
01037                         sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
01038                             touche_sauter_monter,
01039                             touche_descendre, touche_interagir,
01040                             barre_de_son, pseudo,
01041                             (*modeActif), (*personnageActif),
01042                             (*positionActive),
01043                             avancee_niveaux, tailleNiveaux,
01044                             temps_debut_partie, (*compteur_mort), avancee_succes);
01045
01046                         (*programme_lance) = SDL_FALSE;
01047                         clic_effectue = SDL_TRUE;
01048                     }
01049                     else if(clic_case(event_temporaire,
itemsDemandeSauvegarde[2].rectangle)) {
01050                         (*programme_lance) = SDL_FALSE;
01051                         clic_effectue = SDL_TRUE;
01052                     }
01053                     else if(!clic_case(event_temporaire, (*rectangle_demande)))
01054                         clic_effectue = SDL_TRUE;
01055                 }
01056             }
01057         }
01058
01059         SDL_SetWindowResizable((*window), SDL_TRUE);
01060
01061         break;
01062
01063     default:
01064         break;
01065 }
01066 }
01067
01068     /* Traitement de l'appuie des touches */
01069     if((*sauter)) {
01070         if((*interagir))
01071             traitement_touches(position_x, position_y, bloc_x, bloc_y, tile_map_mini_jeu_niveau_3,
01072                 0);
01073
01074         /* On vérifie que c'est bien un chemin et qu'il n'y a pas le bloc */
01075         else if (((tile_map_mini_jeu_niveau_3[(*position_y) - 1][(*position_x)] == 0) ||
01076             (tile_map_mini_jeu_niveau_3[(*position_y) - 1][(*position_x)] == 3) ||
01077             (tile_map_mini_jeu_niveau_3[(*position_y) - 1][(*position_x)] == 4) ||
01078             (tile_map_mini_jeu_niveau_3[(*position_y) - 1][(*position_x)] == 5)) &&
01079             (!((((*position_x) == (*bloc_x)) && ((*position_y) == (*bloc_y) + 1)))) {
01080             (*position_y)--;
01081         }
01082     }

```

```

01083         SDL_Delay(150);
01084     }
01085 }
01086
01087 if((*descendre)) {
01088
01089     if((*interagir))
01090         traitement_touches(position_x, position_y, bloc_x, bloc_y, tile_map_mini_jeu_niveau_3,
01091 1);
01092
01093     /* On vérifie que c'est bien un chemin et qu'il n'y a pas le bloc */
01094     else if (((tile_map_mini_jeu_niveau_3[(*position_y) + 1][(*position_x)] == 0) ||
01095             (tile_map_mini_jeu_niveau_3[(*position_y) + 1][(*position_x)] == 3) ||
01096             (tile_map_mini_jeu_niveau_3[(*position_y) + 1][(*position_x)] == 4) ||
01097             (tile_map_mini_jeu_niveau_3[(*position_y) + 1][(*position_x)] == 5)) &&
01098             (!(((*position_x) == (*bloc_x)) && ((*position_y) == (*bloc_y) - 1)))) {
01099
01100         (*position_y)++;
01101         SDL_Delay(150);
01102     }
01103
01104     if((*reculer)) {
01105
01106         if((*interagir))
01107             traitement_touches(position_x, position_y, bloc_x, bloc_y, tile_map_mini_jeu_niveau_3,
01108 2);
01109
01110         /* On vérifie que c'est bien un chemin et qu'il n'y a pas le bloc */
01111         else if (((tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) - 1] == 0) ||
01112             (tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) - 1] == 3) ||
01113             (tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) - 1] == 4) ||
01114             (tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) - 1] == 5)) &&
01115             (!(((*position_x) == (*bloc_x) + 1) && ((*position_y) == (*bloc_y))))) {
01116
01117             (*position_x)--;
01118             SDL_Delay(150);
01119         }
01120
01121         if((*avancer)) {
01122
01123             if((*interagir))
01124                 traitement_touches(position_x, position_y, bloc_x, bloc_y, tile_map_mini_jeu_niveau_3,
01125 3);
01126
01127             /* On vérifie que c'est bien un chemin et qu'il n'y a pas le bloc */
01128             else if (((tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) + 1] == 0) ||
01129                 (tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) + 1] == 3) ||
01130                 (tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) + 1] == 4) ||
01131                 (tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x) + 1] == 5)) &&
01132                 (!(((*position_x) == (*bloc_x) - 1) && ((*position_y) == (*bloc_y))))) {
01133
01134                 (*position_x)++;
01135                 SDL_Delay(150);
01136             }
01137
01138             /* Cas où le joueur récupère le second collectible */
01139             if((tile_map_mini_jeu_niveau_3[(*position_y)][(*position_x)] == 5) &&
01140                 (!avancee_niveaux[2].numero_collectible[1])) {
01141
01142                 /* Effet sonore quand on ramasse un collectible */
01143                 if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
01144                     erreur("Chargement de l'effet sonore");
01145
01146                 Mix_PlayChannel(1, effet_sonore, 0);
01147
01148                 avancee_niveaux[2].numero_collectible[1] = 1;
01149             }
01150
01151             /* Vérifier si le bloc atteint la fin du labyrinthe et mettre à jour les tuiles de bordure si
01152             nécessaire */
01153             if((((*bloc_x) == 30) && ((*bloc_y) == 22)) {
01154                 mise_a_jour_bordures_niveau_3((*render), (*texture_image_mur_termine),
01155                 tile_map_mini_jeu_niveau_3, 31, 23,
01156                 rectangle_tile, (*largeur_tile), (*hauteur_tile));
01157
01158                 /* Musique du salon */
01159                 if(((*musique) = Mix_LoadMUS("./sons/musiques/salon.mp3")) == NULL)
01160                     erreur("Chargement de la musique");
01161
01162                 Mix_PlayMusic((*musique), -1);
01163
01164                 (*mini_jeu) = 0;
01165                 (*mini_jeu_2_termine) = 1;
01166

```

```

01164         (*largeur_tile) = (*largeur) / 32;
01165         (*hauteur_tile) = (*hauteur) / 18;
01166
01167         (*sauter) = 0;
01168         (*avancer) = 0;
01169         (*reculer) = 0;
01170         (*tombe) = 0;
01171         (*saut) = 0;
01172
01173         salon_arrivee_niveaux_2_3(position_x, position_y, tile_map, (*page_active));
01174
01175         tile_map[3][24] = 0;
01176
01177         if((*mini_jeu_1_termine)) {
01178             tile_map[3][5] = 0;
01179             tile_map[5][6] = 5;
01180         }
01181     }
01182     /* Mise à jour du rendu */
01183     mise_a_jour_mini_jeu_2_niveau_3(renderer, modeActif, texture_image_sol_labyrinthe,
01184                                     rectangle_plein_ecran, texture_image_plein_ecran,
01185                                     texture_image_mur_mini_jeu, texture_image_bordure_labyrinthe,
01186                                     texture_image_mur_termine,
01187                                     texture_image_fin_labyrinthe, texture_image_croix,
01188                                     rectangle_croix,
01189                                     texture, rectangle_tile, (*bloc_x), (*bloc_y),
01190                                     texture_image_personnage, rectangle_personnage,
01191                                     (*position_x), (*position_y), tile_map_mini_jeu_niveau_3,
01192                                     avancee_niveaux,
01193                                     (*largeur_tile), (*hauteur_tile));
01194 }
01195 }

```

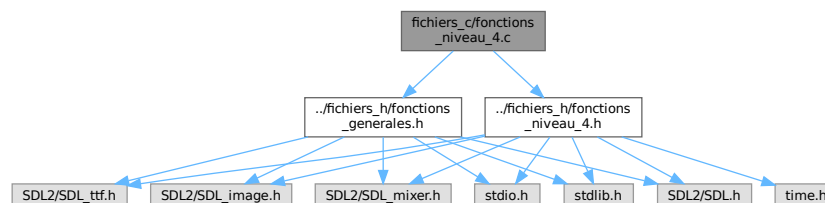
## 4.17 Référence du fichier fichiers\_c/fonctions\_niveau\_4.c

Fichier contenant les fonctions servant à la gestion du niveau 2

```
#include <../fichiers_h/fonctions_generales.h>
```

```
#include <../fichiers_h/fonctions_niveau_4.h>
```

Graphe des dépendances par inclusion de fonctions\_niveau\_4.c:



### Fonctions

- void [initialisation\\_objets\\_niveau\\_4](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_bordure, SDL\_Texture \*\*texture\_image\_porte, SDL\_Texture \*\*texture\_image\_pique, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau)

*Fonction qui permet d'initialiser les différents objets du niveau 4.*

- void [etage\\_1](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)

*Fonction qui permet de créer l'étage 1.*

- void [etage\\_2](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)

*Fonction qui permet de créer l'étage 2.*

- void [etage\\_3](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)

*Fonction qui permet de créer l'étage 3.*

- void [etage\\_4](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)

*Fonction qui permet de créer l'étage 4.*

- void [etage\\_5](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)

*Fonction qui permet de créer l'étage 5.*

- void [mise\\_a\\_jour\\_rendu\\_niveau\\_4](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_mur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_bordure, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_tile, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_porte, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, int numero\_etage, int position\_x, int position\_y, int tile\_map[18][32], SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile)
- void [niveau\\_4](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_mur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_bordure, SDL\_Texture \*\*texture\_image\_porte, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, SDL\_Surface \*\*surface, [modes\\_t](#) \*modeActif, int collectibles\_intermediaires[3], SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_interagir, int tile\_map[18][32], SDL\_Rect \*rectangle\_tile, SDL\_Texture \*\*texture\_image\_perso\_gagnant, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemande, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Rect \*rectangle\_demande, SDL\_Color couleurNoire, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, int \*avancer, int \*reculer, int \*sauter, int \*position\_avant\_saut, int \*saut, int \*tombe, int \*numero\_etage, int \*position\_x\_initiale, int \*position\_y\_initiale, int \*position\_x, int \*position\_y, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, [page\\_t](#) \*page\_active, [itemMenu](#) \*itemsDemandeSauvegarde, SDL\_Keycode \*touche\_descendre, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

### 4.17.1 Description détaillée

Fichier contenant les fonctions servant à la gestion du niveau 2

Fichier contenant les fonctions servant à la gestion du niveau 4

Fichier contenant les fonctions servant à la gestion du niveau 3

Définition dans le fichier [fonctions\\_niveau\\_4.c](#).

### 4.17.2 Documentation des fonctions

#### 4.17.2.1 etage\_1()

```
void etage_1 (
    int * position_x,
```

```

int * position_y,
int * position_x_initiale,
int * position_y_initiale,
int tile_map[18][32],
SDL_Renderer ** renderer,
SDL_Surface ** surface,
SDL_Texture ** texture_image_mur )

```

Fonction qui permet de créer l'étage 1.

#### Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

Voir également

[chargement\\_image](#)

Définition à la ligne 47 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.17.2.2 etage\_2()

```

void etage_2 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )

```

Fonction qui permet de créer l'étage 2.

#### Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

Voir également

[chargement\\_image](#)

Définition à la ligne 105 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.17.2.3 etage\_3()

```
void etage_3 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 3.

##### Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

Voir également

[chargement\\_image](#)

Définition à la ligne 163 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.17.2.4 etage\_4()

```
void etage_4 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 4.



## Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>render</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

## Voir également

[chargement\\_image](#)

Définition à la ligne 221 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.17.2.5 etage\_5()

```
void etage_5 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** render,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 5.

## Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>render</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

## Voir également

[chargement\\_image](#)

Définition à la ligne 279 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.17.2.6 initialisation\_objets\_niveau\_4()

```
void initialisation_objets_niveau_4 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_bordure,
    SDL_Texture ** texture_image_porte,
    SDL_Texture ** texture_image_pique,
    SDL_Texture ** texture_image_fin_dernier_niveau )
```

Fonction qui permet d'initialiser les différents objets du niveau 4.

##### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_fond</i>	Texture de l'image du fond.
<i>texture_image_bordure</i>	Texture de l'image de la bordure.
<i>texture_image_porte</i>	Texture de l'image de la porte.
<i>texture_image_pique</i>	Texture de l'image du pique.
<i>texture_image_fin_dernier_niveau</i>	Texture de l'image de fin du dernier niveau.

##### Voir également

[chargement\\_image](#)

Définition à la ligne 20 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.17.2.7 mise\_a\_jour\_rendu\_niveau\_4()

```
void mise_a_jour_rendu_niveau_4 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_mur,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_bordure,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_porte,
    SDL_Texture ** texture_image_fin_dernier_niveau,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    SDL_Texture ** texture_image_pique,
    niveaux * avancee_niveaux,
    int numero_etage,
    int position_x,
    int position_y,
    int tile_map[18][32],
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    int largeur,
```

```
int hauteur,  
int largeur_tile,  
int hauteur_tile )
```

Définition à la ligne 351 du fichier `fonctions_niveau_4.c`.

#### 4.17.2.8 niveau\_4()

```
void niveau_4 (  
    SDL_Event * event,  
    SDL_Window ** window,  
    SDL_Renderer ** renderer,  
    SDL_bool * programme_lance,  
    SDL_Texture ** texture,  
    SDL_Rect * rectangle_plein_ecran,  
    SDL_Texture ** texture_image_plein_ecran,  
    SDL_bool * plein_ecran,  
    SDL_Texture ** texture_image_personnage,  
    SDL_Rect * rectangle_personnage,  
    SDL_Texture ** texture_image_mur,  
    SDL_Texture ** texture_image_fond,  
    SDL_Texture ** texture_image_bordure,  
    SDL_Texture ** texture_image_porte,  
    SDL_Texture ** texture_image_pique,  
    niveaux * avancee_niveaux,  
    SDL_Surface ** surface,  
    modes_t * modeActif,  
    int collectibles_intermediaires[3],  
    SDL_Keycode * touche_aller_a_droite,  
    SDL_Keycode * touche_aller_a_gauche,  
    SDL_Keycode * touche_sauter_monter,  
    SDL_Keycode * touche_interagir,  
    int tile_map[18][32],  
    SDL_Rect * rectangle_tile,  
    SDL_Texture ** texture_image_perso_gagnant,  
    itemMenu * itemsDemandeQuitter,  
    int tailleDemande,  
    SDL_Texture ** texture_image_croix,  
    SDL_Rect * rectangle_croix,  
    SDL_Texture ** texture_texte,  
    TTF_Font ** police,  
    SDL_Rect * rectangle_demande,  
    SDL_Color couleurNoire,  
    SDL_Texture ** texture_image_fin_dernier_niveau,  
    int * avancer,  
    int * reculer,  
    int * sauter,  
    int * position_avant_saut,  
    int * saut,  
    int * tombe,  
    int * numero_etage,  
    int * position_x_initiale,  
    int * position_y_initiale,  
    int * position_x,  
    int * position_y,  
    int * largeur,  
    int * hauteur,
```

```

    int * largeur_tile,
    int * hauteur_tile,
    page_t * page_active,
    itemMenu * itemsDemandeSauvegarde,
    SDL_Keycode * touche_descendre,
    barreDeSon * barre_de_son,
    itemMenu * pseudo,
    personnage_t * personnageActif,
    position_t * positionActive,
    int tailleNiveaux,
    time_t temps_debut_partie,
    int * compteur_mort,
    int * avancee_succes,
    int avancee_succes_intermediaires[10] )

```

Définition à la ligne 510 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.18 fonctions\_niveau\_4.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_niveau_4.c
00003  * \brief Fichier contenant les fonctions servant à la gestion du niveau 4
00004  */
00005 #include <../fichiers_h/fonctions_generales.h>
00006 #include <../fichiers_h/fonctions_niveau_4.h>
00007
00008 /**
00009  * \fn void initialisation_objets_niveau_4(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00010  **texture_image_fond, SDL_Texture **texture_image_bordure, SDL_Texture **texture_image_porte,
00011  SDL_Texture **texture_image_pique, SDL_Texture **texture_image_fin_dernier_niveau)
00012  * \brief Fonction qui permet d'initialiser les différents objets du niveau 4
00013  * \param renderer Pointeur vers le render SDL.
00014  * \param surface Pointeur vers la surface SDL.
00015  * \param texture_image_fond Texture de l'image du fond.
00016  * \param texture_image_bordure Texture de l'image de la bordure.
00017  * \param texture_image_porte Texture de l'image de la porte.
00018  * \param texture_image_pique Texture de l'image du pique.
00019  * \param texture_image_fin_dernier_niveau Texture de l'image de fin du dernier niveau.
00020  * \see chargement_image
00021  */
00022 void initialisation_objets_niveau_4(SDL_Renderer **renderer, SDL_Surface **surface,
00023                                     SDL_Texture **texture_image_fond, SDL_Texture
00024                                     **texture_image_bordure,
00025                                     SDL_Texture **texture_image_porte, SDL_Texture
00026                                     **texture_image_pique,
00027                                     SDL_Texture **texture_image_fin_dernier_niveau) {
00028
00029     /* Chargement des images pour le niveau 4 */
00030     chargement_image(renderer, surface, texture_image_fond, "./images/niveau_4/fond_niveau_4.jpg");
00031     chargement_image(renderer, surface, texture_image_bordure,
00032                     "./images/niveau_4/bordure_niveau_4.png");
00033     chargement_image(renderer, surface, texture_image_porte, "./images/niveau_4/porte.png");
00034     chargement_image(renderer, surface, texture_image_pique, "./images/pique.png");
00035     chargement_image(renderer, surface, texture_image_fin_dernier_niveau,
00036                     "./images/niveau_4/fin_dernier_niveau.png");
00037 }
00038
00039 /**
00040  * \fn void etage_1(int *position_x, int *position_y, int *position_x_initiale, int
00041  *position_y_initiale, int tile_map[18][32],SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00042  **texture_image_mur)
00043  * \brief Fonction qui permet de créer l'étage 1
00044  * \param position_x Pointeur vers la position en abscisse du joueur.
00045  * \param position_y Pointeur vers la position en ordonnée du joueur.
00046  * \param position_x_initiale Pointeur vers la position initiale en abscisse du joueur.
00047  * \param position_y_initiale Pointeur vers la position initiale en ordonnée du joueur.
00048  * \param tile_map Tableau représentant la carte du niveau.
00049  * \param renderer Pointeur vers le render SDL.
00050  * \param surface Pointeur vers la surface SDL.
00051  * \param texture_image_mur Texture de l'image des murs.
00052  * \see chargement_image
00053  */

```

```

00047 void etage_1(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
tile_map[18][32],
00048     SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur) {
00049
00050     int x, y;
00051
00052     /* Positionnement du personnage au début de l'étage */
00053
00054     (*position_x) = 2;
00055     (*position_y) = 15;
00056
00057     (*position_x_initiale) = (*position_x);
00058     (*position_y_initiale) = (*position_y);
00059
00060     /* Création de l'étage 1 */
00061     int initialisation_tile_map[18][32] = {
00062         {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00063         2},
00064         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00065         2},
00066         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0,
00067         2},
00068         {2, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
00069         2},
00070         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
00071         2},
00072         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00073         2},
00074         {2, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00075         2},
00076         {2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
00077         2},
00078         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00079         2},
00080         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00081         2},
00082         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00083         2},
00084         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00085         2},
00086         {2, 0, 4, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00087         2},
00088         {2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
00089         2},
00090         {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00091         2}
00092     };
00093
00094     /* Copie du nouvel étage */
00095     for (y = 0; y < 18; y++)
00096         for (x = 0; x < 32; x++)
00097             tile_map[y][x] = initialisation_tile_map[y][x];
00098
00099     /* Changement de mur pour l'étage 1 */
00100     chargement_image(renderer, surface, texture_image_mur, "./images/niveau_4/mur_etage_1.png");
00101 }
00102
00103 /**
00104  * \fn void etage_2(int *position_x, int *position_y, int *position_x_initiale, int
00105  * position_y_initiale, int tile_map[18][32],SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00106  * texture_image_mur)
00107  * \brief Fonction qui permet de créer l'étage 2
00108  * \param position_x Pointeur vers la position en abscisse du joueur.
00109  * \param position_y Pointeur vers la position en ordonnée du joueur.
00110  * \param position_x_initiale Pointeur vers la position initiale en abscisse du joueur.
00111  * \param position_y_initiale Pointeur vers la position initiale en ordonnée du joueur.
00112  * \param tile_map Tableau représentant la carte du niveau.
00113  * \param renderer Pointeur vers le renderer SDL.
00114  * \param surface Pointeur vers la surface SDL.
00115  * \param texture_image_mur Texture de l'image des murs.
00116  * \see chargement_image
00117  */
00118 void etage_2(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
tile_map[18][32],
00119     SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur) {
00120
00121     int x, y;
00122
00123     /* Positionnement du personnage au début de l'étage */

```



```

00178     {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00179     2},
00179     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
00180     2},
00180     {2, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
00181     2},
00181     {2, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
00182     2},
00182     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00183     2},
00183     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
00184     2},
00184     {2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
00185     2},
00185     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00186     2},
00186     {2, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00187     2},
00187     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
00188     2},
00188     {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00189     2},
00189     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
00190     2},
00190     {2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0,
00191     2},
00191     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
00192     2},
00192     {2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
00193     2},
00193     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 1, 0, 0, 0, 0, 4, 0,
00194     2},
00194     {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
00195     2},
00195     {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00196     2}
00196
00197     };
00198
00199     /* Copie du nouvel étage */
00200     for (y = 0; y < 18; y++)
00201         for (x = 0; x < 32; x++)
00202             tile_map[y][x] = initialisation_tile_map[y][x];
00203
00204     /* Changement de mur pour l'étage 3 */
00205     chargement_image(renderer, surface, texture_image_mur, "./images/niveau_4/mur_etage_3.png");
00206 }
00207
00208 /**
00209  * \fn void etage_4(int *position_x, int *position_y, int *position_x_initiale, int
00210  * position_y_initiale, int tile_map[18][32],SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00211  **texture_image_mur)
00212  * \brief Fonction qui permet de créer l'étage 4
00213  * \param position_x Pointeur vers la position en abscisse du joueur.
00214  * \param position_y Pointeur vers la position en ordonnée du joueur.
00215  * \param position_x_initiale Pointeur vers la position initiale en abscisse du joueur.
00216  * \param position_y_initiale Pointeur vers la position initiale en ordonnée du joueur.
00217  * \param tile_map Tableau représentant la carte du niveau.
00218  * \param renderer Pointeur vers le renderer SDL.
00219  * \param surface Pointeur vers la surface SDL.
00220  * \param texture_image_mur Texture de l'image des murs.
00221  * \see chargement_image
00222  */
00221 void etage_4(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
00222 tile_map[18][32],
00223 SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur) {
00224     int x, y;
00225
00226     /* Positionnement du personnage au début de l'étage */
00227
00228     (*position_x) = 2;
00229     (*position_y) = 15;
00230
00231     (*position_x_initiale) = (*position_x);
00232     (*position_y_initiale) = (*position_y);
00233
00234     /* Création de l'étage 4 */
00235     int initialisation_tile_map[18][32] = {
00236     {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00237     2},
00237     {2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
00238     2},
00238     {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
00239     2},
00239     {2, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
00240     2},

```

```

00240         {2, 0, 0, 1, 1, 3, 1, 1, 1, 3, 3, 1, 1, 3, 1, 1, 1, 3, 3, 1, 1, 3, 1, 1, 1, 3, 3, 0, 0, 4, 0,
00241         2},
00241         {2, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
00242         2},
00242         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
00243         2},
00243         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
00244         2},
00244         {2, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
00245         2},
00245         {2, 3, 3, 1, 3, 3, 1, 1, 1, 3, 1, 1, 3, 3, 1, 1, 1, 3, 1, 1, 3, 3, 1, 1, 1, 3, 1, 1, 0, 0, 0,
00246         2},
00246         {2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
00247         2},
00247         {2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00248         2},
00248         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00249         2},
00249         {2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
00250         2},
00250         {2, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00251         2},
00251         {2, 0, 4, 0, 0, 0, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
00252         2},
00252         {2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
00253         2},
00253         {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00254         2}
00254     };
00255
00256     /* Copie du nouvel étage */
00257     for (y = 0; y < 18; y++)
00258         for (x = 0; x < 32; x++)
00259             tile_map[y][x] = initialisation_tile_map[y][x];
00260
00261     /* Changement de mur pour l'étage 4 */
00262     chargement_image(renderer, surface, texture_image_mur, "./images/niveau_4/mur_etage_4.png");
00263 }
00264
00265 /**
00266  * \fn void etage_5(int *position_x, int *position_y, int *position_x_initiale, int
00267  * position_y_initiale, int tile_map[18][32],SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00268  * texture_image_mur)
00269  * \brief Fonction qui permet de créer l'étage 5
00270  * \param position_x Pointeur vers la position en abscisse du joueur.
00271  * \param position_y Pointeur vers la position en ordonnée du joueur.
00272  * \param position_x_initiale Pointeur vers la position initiale en abscisse du joueur.
00273  * \param position_y_initiale Pointeur vers la position initiale en ordonnée du joueur.
00274  * \param tile_map Tableau représentant la carte du niveau.
00275  * \param renderer Pointeur vers le renderer SDL.
00276  * \param surface Pointeur vers la surface SDL.
00277  * \param texture_image_mur Texture de l'image des murs.
00278  * \see chargement_image
00279  */
00279 void etage_5(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
00280 tile_map[18][32],
00281 SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur) {
00282     int x, y;
00283
00284     /* Positionnement du personnage au début de l'étage */
00285
00286     (*position_x) = 29;
00287     (*position_y) = 15;
00288
00289     (*position_x_initiale) = (*position_x);
00290     (*position_y_initiale) = (*position_y);
00291
00292     /* Création de l'étage 5 */
00293     int initialisation_tile_map[18][32] = {
00294         {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
00295         2},
00295         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
00296         2},
00296         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
00297         2},
00297         {2, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
00298         2},
00298         {2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00299         2},
00299         {2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
00300         2},
00300         {2, 3, 3, 1, 3, 3, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 3, 3, 1, 1, 0, 0, 0,
00301         2},
00301         {2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
00302         2},

```



[illegible]

```

00367
00368         for (x = 0; x < largeur / largeur_tile; x++) {
00369
00370             if(tile_map[y][x] == 1)
00371                 (*texture) = (*texture_image_mur);
00372
00373             else if((!(tile_map[y][x])) || (tile_map[y][x] == 3) || (tile_map[y][x] == 4) ||
00374 (tile_map[y][x] == 5) || (tile_map[y][x] == 7))
00375                 (*texture) = (*texture_image_fond);
00376
00377             else if(tile_map[y][x] == 2)
00378                 (*texture) = (*texture_image_bordure);
00379
00380             rectangle_tile->x = x * largeur_tile;
00381             rectangle_tile->y = y * hauteur_tile;
00382             rectangle_tile->w = largeur_tile;
00383             rectangle_tile->h = hauteur_tile;
00384
00385             if(SDL_RenderCopy((*renderer), (*texture), NULL, rectangle_tile) != 0)
00386                 erreur("Copie de la texture");
00387
00388             if(tile_map[y][x] == 3)
00389                 if(SDL_RenderCopy((*renderer), (*texture_image_pique), NULL, rectangle_tile) != 0)
00390                     erreur("Copie de la texture");
00391
00392             if(tile_map[y][x] == 4)
00393                 if(SDL_RenderCopy((*renderer), (*texture_image_porte), NULL, rectangle_tile) != 0)
00394                     erreur("Copie de la texture");
00395
00396             if(tile_map[y][x] == 7)
00397                 if(SDL_RenderCopy((*renderer), (*texture_image_fin_dernier_niveau), NULL,
00398 rectangle_tile) != 0)
00399                     erreur("Copie de la texture");
00400
00401             if((tile_map[y][x] == 5) && (numero_etage == 2) &&
00402 (avancee_niveaux[3].numero_collectible[0] == 0))
00403                 if(SDL_RenderCopy((*renderer), avancee_niveaux[3].texture_image_collectible, NULL,
00404 rectangle_tile) != 0)
00405                     erreur("Copie de la texture");
00406
00407             if((tile_map[y][x] == 5) && (numero_etage == 3) &&
00408 (avancee_niveaux[3].numero_collectible[1] == 0))
00409                 if(SDL_RenderCopy((*renderer), avancee_niveaux[3].texture_image_collectible, NULL,
00410 rectangle_tile) != 0)
00411                     erreur("Copie de la texture");
00412
00413             /* Copie la texture de l'image du personnage */
00414
00415             rectangle_personnage->x = position_x * largeur_tile;
00416             rectangle_personnage->y = position_y * hauteur_tile;
00417             rectangle_personnage->w = largeur_tile;
00418             rectangle_personnage->h = hauteur_tile;
00419
00420             if(SDL_RenderCopy((*renderer), (*texture_image_personnage), NULL, rectangle_personnage) != 0)
00421                 erreur("Copie de la texture");
00422
00423             /* Copie la texture de l'image de plein écran */
00424
00425             rectangle_plein_ecran->x = largeur_tile * 31;
00426             rectangle_plein_ecran->y = 0;
00427             rectangle_plein_ecran->w = largeur_tile;
00428             rectangle_plein_ecran->h = hauteur_tile;
00429
00430             if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00431                 erreur("Copie de la texture");
00432
00433             /* Copie la texture de l'image de la croix */
00434
00435             rectangle_croix->x = 0;
00436             rectangle_croix->y = 0;
00437             rectangle_croix->w = largeur_tile;
00438             rectangle_croix->h = hauteur_tile;
00439
00440             if(SDL_RenderCopy((*renderer), (*texture_image_croix), NULL, rectangle_croix) != 0)
00441                 erreur("Copie de la texture");
00442
00443             /* Affiche le rendu */
00444             SDL_RenderPresent((*renderer));
00445 }

```

```

00446
00447 /**
00448 * \fn void niveau_4(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_Texture
**texture, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
*plein_ecran, SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, SDL_Texture
**texture_image_mur, SDL_Texture **texture_image_fond, SDL_Texture **texture_image_bordure,
SDL_Texture **texture_image_porte, SDL_Texture **texture_image_pique, niveaux *avancee_niveaux,
SDL_Surface **surface, modes_t *modeActif, int collectibles_intermediaires[3], SDL_Keycode
*touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter,
SDL_Keycode *touche_interagir, int tile_map[18][32], SDL_Rect *rectangle_tile, SDL_Texture
**texture_image_perso_gagnant, itemMenu *itemsDemandeQuitter, int tailleDemandeQuitter, SDL_Texture
**texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande_quitter, SDL_Color couleurNoire,
SDL_Texture **texture_image_fin_dernier_niveau, int *avancer, int *reculer, int *sauter, int
*position_avant_saut, int *saut, int *tombe, int *numero_etage, int *position_x_initiale, int
*position_y_initiale, int *position_x, int *position_y, int *largeur, int *hauteur, int *largeur_tile,
int *hauteur_tile, page_t *page_active)
00449 * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le niveau 4
00450 * \param event Pointeur vers la structure d'événements SDL.
00451 * \param window Pointeur vers la fenêtre SDL.
00452 * \param renderer Pointeur vers le renderer SDL.
00453 * \param texture Texture générale.
00454 * \param rectangle_plein_ecran Rectangle pour l'affichage en plein écran.
00455 * \param texture_image_plein_ecran Texture de l'image en plein écran.
00456 * \param plein_ecran Pointeur vers un booléen indiquant si le jeu est en mode plein écran.
00457 * \param texture_image_personnage Texture de l'image du personnage.
00458 * \param rectangle_personnage Rectangle du personnage.
00459 * \param texture_image_mur Texture de l'image des murs.
00460 * \param texture_image_fond Texture de l'image du fond.
00461 * \param texture_image_bordure Texture de l'image de la bordure.
00462 * \param texture_image_porte Texture de l'image de la porte.
00463 * \param texture_image_pique Texture de l'image des pics.
00464 * \param avancee_niveaux Structure contenant l'avancement dans les niveaux.
00465 * \param surface Surface SDL pour le texte.
00466 * \param modeActif Pointeur vers le mode de jeu actif.
00467 * \param collectibles_intermediaires Tableau des collectibles intermédiaires.
00468 * \param touche_aller_a_droite Touche pour aller à droite.
00469 * \param touche_aller_a_gauche Touche pour aller à gauche.
00470 * \param touche_sauter_monter Touche pour sauter ou monter.
00471 * \param touche_interagir Touche pour interagir.
00472 * \param tile_map Tableau représentant la carte du niveau.
00473 * \param rectangle_tile Rectangle de la tuile.
00474 * \param texture_image_perso_gagnant Texture de l'image du personnage gagnant.
00475 * \param itemsDemandeQuitter Tableau des items pour demander à quitter.
00476 * \param tailleDemandeQuitter Taille du tableau des items pour demander à quitter.
00477 * \param texture_texte Texture pour le texte.
00478 * \param police Police de caractères pour le texte.
00479 * \param rectangle_demande_quitter Rectangle pour la demande de quitter.
00480 * \param couleurNoire Couleur noire pour le texte.
00481 * \param texture_image_fin_dernier_niveau Texture de l'image de fin du dernier niveau.
00482 * \param avancer Pointeur vers la commande d'avancer.
00483 * \param reculer Pointeur vers la commande de reculer.
00484 * \param sauter Pointeur vers la commande de sauter.
00485 * \param position_avant_saut Position avant le saut.
00486 * \param tombe Pointeur vers la commande de tomber.
00487 * \param numero_etage Numéro de l'étage.
00488 * \param position_x_initiale Position initiale en abscisse.
00489 * \param position_y_initiale Position initiale en ordonnée.
00490 * \param position_x Position en abscisse du joueur.
00491 * \param position_y Position en ordonnée du joueur.
00492 * \param largeur Largeur de l'écran.
00493 * \param hauteur Hauteur de l'écran.
00494 * \param largeur_tile Largeur d'une tuile.
00495 * \param hauteur_tile Hauteur d'une tuile.
00496 * \param page_active Pointeur vers la page active.
00497 * \see redimensionnement_fenetre
00498 * \see erreur
00499 * \see etage_1
00500 * \see etage_2
00501 * \see etage_3
00502 * \see etage_4
00503 * \see etage_5
00504 * \see clic_plein_ecran
00505 * \see clic_case
00506 * \see demande_quitter_niveau
00507 * \see deplacement_personnage
00508 * \see mise_a_jour_rendu_niveau_4
00509 */
00510 void niveau_4(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance,
00511     SDL_Texture **texture, SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran, SDL_bool *plein_ecran,
00512     SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage,
00513     SDL_Texture **texture_image_mur, SDL_Texture **texture_image_fond,
00514     SDL_Texture **texture_image_bordure, SDL_Texture **texture_image_porte,
00515     SDL_Texture **texture_image_pique, niveaux *avancee_niveaux,
00516     SDL_Surface **surface, modes_t *modeActif, int collectibles_intermediaires[3],
00517     SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,

```

```

00518         SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_interagir,
00519         int tile_map[18][32], SDL_Rect *rectangle_tile, SDL_Texture
**texture_image_perso_gagnant,
00520         itemMenu *itemsDemandeQuitter, int tailleDemande, SDL_Texture **texture_image_croix,
SDL_Rect *rectangle_croix,
00521         SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande,
00522         SDL_Color couleurNoire, SDL_Texture **texture_image_fin_dernier_niveau,
00523         int *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut, int
*tombe, int *numero_etage,
00524         int *position_x_initiale, int *position_y_initiale, int *position_x, int *position_y,
00525         int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page_t *page_active,
00526         itemMenu *itemsDemandeSauvegarde, SDL_Keycode *touche_descendre, barreDeSon
*barre_de_son, itemMenu *pseudo,
00527         personnage_t *personnageActif, position_t *positionActive, int tailleNiveaux,
00528         time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
avancee_succes_intermediaires[10]) {
00529
00530     SDL_Event event_temporaire;
00531     SDL_bool clic_effectue = SDL_FALSE;
00532
00533     Mix_Chunk *effet_sonore = NULL;
00534
00535     int i;
00536
00537     while(SDL_PollEvent(&event)) {
00538
00539         switch(event->type) {
00540
00541             /* Gestion de l'événement de redimensionnement de la fenêtre */
00542             case SDL_WINDOWEVENT:
00543                 redimensionnement_fenetre((event), largeur, hauteur);
00544
00545                 (*largeur_tile) = (*largeur) / 32;
00546                 (*hauteur_tile) = (*hauteur) / 18;
00547
00548                 break;
00549
00550             case SDL_KEYDOWN:
00551
00552                 if(event->key.keysym.sym == (*touche_sauter_monter))
00553                     (*sauter) = 1;
00554
00555                 if(event->key.keysym.sym == (*touche_aller_a_droite))
00556                     (*avancer) = 1;
00557
00558                 if(event->key.keysym.sym == (*touche_aller_a_gauche))
00559                     (*reculer) = 1;
00560
00561                 if(event->key.keysym.sym == (*touche_interagir)) {
00562
00563                     /* Cas où vous retournez sur la carte */
00564                     if (((*numero_etage) == 1) && ((*position_x) == 2) && ((*position_y) == 15)) {
00565
00566                         /* Effet sonore quand on passe dans une porte */
00567                         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00568                             erreur("Chargement de l'effet sonore");
00569
00570                         Mix_PlayChannel(1, effet_sonore, 0);
00571
00572                         (*page_active) = CARTE;
00573                     }
00574
00575                     /* Cas où vous montez à l'étage 2 */
00576                     else if (((*numero_etage) == 1) && ((*position_x) == 29) && ((*position_y) == 2))
00577
00578
00579                         /* Effet sonore quand on passe dans une porte */
00580                         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00581                             erreur("Chargement de l'effet sonore");
00582
00583                         Mix_PlayChannel(1, effet_sonore, 0);
00584
00585                         etage_2(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00586                             renderer, surface, texture_image_mur);
00587
00588                         (*numero_etage) = 2;
00589                     }
00590
00591                     /* Cas où vous descendez à l'étage 1 */
00592                     else if (((*numero_etage) == 2) && ((*position_x) == 29) && ((*position_y) == 15))
00593
00594
00595                         /* Effet sonore quand on passe dans une porte */
00596                         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
                             erreur("Chargement de l'effet sonore");

```

```

00597
00598         Mix_PlayChannel(1, effet_sonore, 0);
00599
00600         etage_1(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00601             renderer, surface, texture_image_mur);
00602
00603         (*position_x) = 29;
00604         (*position_y) = 2;
00605         (*numero_etage) = 1;
00606     }
00607
00608     /* Cas où vous montez à l'étage 3 */
00609     else if (((*numero_etage) == 2) && ((*position_x) == 29) && ((*position_y) == 2))
    {
00610
00611         /* Effet sonore quand on passe dans une porte */
00612         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00613             erreur("Chargement de l'effet sonore");
00614
00615         Mix_PlayChannel(1, effet_sonore, 0);
00616
00617         etage_3(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00618             renderer, surface, texture_image_mur);
00619
00620         (*numero_etage) = 3;
00621     }
00622
00623     /* Cas où vous descendez à l'étage 2 */
00624     else if (((*numero_etage) == 3) && ((*position_x) == 29) && ((*position_y) == 15))
    {
00625
00626         /* Effet sonore quand on passe dans une porte */
00627         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00628             erreur("Chargement de l'effet sonore");
00629
00630         Mix_PlayChannel(1, effet_sonore, 0);
00631
00632         etage_2(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00633             renderer, surface, texture_image_mur);
00634
00635         (*position_x) = 29;
00636         (*position_y) = 2;
00637         (*numero_etage) = 2;
00638     }
00639
00640     /* Cas où vous montez à l'étage 4 */
00641     else if (((*numero_etage) == 3) && ((*position_x) == 2) && ((*position_y) == 2)) {
00642
00643         /* Effet sonore quand on passe dans une porte */
00644         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00645             erreur("Chargement de l'effet sonore");
00646
00647         Mix_PlayChannel(1, effet_sonore, 0);
00648
00649         etage_4(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00650             renderer, surface, texture_image_mur);
00651
00652         (*numero_etage) = 4;
00653     }
00654
00655     /* Cas où vous descendez à l'étage 3 */
00656     else if (((*numero_etage) == 4) && ((*position_x) == 2) && ((*position_y) == 15))
    {
00657
00658         /* Effet sonore quand on passe dans une porte */
00659         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00660             erreur("Chargement de l'effet sonore");
00661
00662         Mix_PlayChannel(1, effet_sonore, 0);
00663
00664         etage_3(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00665             renderer, surface, texture_image_mur);
00666
00667         (*position_x) = 2;
00668         (*position_y) = 2;
00669         (*numero_etage) = 3;
00670     }
00671
00672     /* Cas où vous montez à l'étage 5 */
00673     else if (((*numero_etage) == 4) && ((*position_x) == 29) && ((*position_y) == 2))
    {
00674

```

```

00675             /* Effet sonore quand on passe dans une porte */
00676             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00677                 erreur("Chargement de l'effet sonore");
00678
00679             Mix_PlayChannel(1, effet_sonore, 0);
00680
00681             etage_5(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00682                     renderer, surface, texture_image_mur);
00683
00684             (*numero_etage) = 5;
00685         }
00686
00687         /* Cas où vous descendez à l'étage 4 */
00688         else if (((*numero_etage) == 5) && ((*position_x) == 29) && ((*position_y) == 15))
{
00689
00690             /* Effet sonore quand on passe dans une porte */
00691             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/porte.wav")) == NULL)
00692                 erreur("Chargement de l'effet sonore");
00693
00694             Mix_PlayChannel(1, effet_sonore, 0);
00695
00696             etage_4(position_x, position_y, position_x_initiale, position_y_initiale,
tile_map,
00697                     renderer, surface, texture_image_mur);
00698
00699             (*position_x) = 29;
00700             (*position_y) = 2;
00701             (*numero_etage) = 4;
00702         }
00703     }
00704
00705     break;
00706
00707     case SDL_KEYUP:
00708
00709         if(event->key.keysym.sym == (*touche_sauter_monter))
00710             (*sauter) = 0;
00711
00712         if(event->key.keysym.sym == (*touche_aller_a_droite))
00713             (*avancer) = 0;
00714
00715         if(event->key.keysym.sym == (*touche_aller_a_gauche))
00716             (*reculer) = 0;
00717
00718         break;
00719
00720     /* Option plein écran */
00721     case SDL_MOUSEBUTTONDOWN:
00722
00723         if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window)) {
00724
00725             redimensionnement_fenetre((*event), largeur, hauteur);
00726
00727             (*largeur_tile) = (*largeur) / 32;
00728             (*hauteur_tile) = (*hauteur) / 18;
00729         }
00730
00731         /* Demande au joueur s'il veut quitter le niveau */
00732         if(clic_case((*event), (*rectangle_croix))) {
00733
00734             SDL_SetWindowResizable((*window), SDL_FALSE);
00735
00736             demande_quitter_niveau(renderer, rectangle_demande,
00737                                     surface, texture_texte, police, couleurNoire,
00738                                     itemsDemandeQuitter, tailleDemande, (*largeur), (*hauteur));
00739
00740             while (!clic_effectue) {
00741
00742                 while (SDL_PollEvent(&event_temporaire)) {
00743
00744                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00745
00746                         if(clic_case(event_temporaire, itemsDemandeQuitter[1].rectangle)) {
00747
00748                             (*page_active) = CARTE;
00749
00750                             for(i = 0; i < 3; i++)
00751                                 avancee_niveaux[3].numero_collectible[i] =
collectibles_intermediaires[i];
00752
00753                             for(i = 0; i < 10; i++)
00754                                 avancee_succes[i] = avancee_succes_intermediaires[i];
00755
00756                             clic_effectue = SDL_TRUE;
00757                         }

```

```

00758
00759         else if(clic_case(event_temporaire, itemsDemandeQuitter[2].rectangle))
00760             clic_effectue = SDL_TRUE;
00761     }
00762 }
00763 }
00764
00765     SDL_SetWindowResizable((*window), SDL_TRUE);
00766 }
00767
00768     break;
00769
00770     /* Quitter le programme en demandant s'il faut sauvegarder la partie */
00771     case SDL_QUIT:
00772
00773         SDL_SetWindowResizable((*window), SDL_FALSE);
00774
00775         demande_sauvegarde(renderer, rectangle_demande,
00776             surface, texture_texte, police, couleurNoire,
00777             itemsDemandeSauvegarde, tailleDemande, (*largeur), (*hauteur));
00778
00779         while (!clic_effectue) {
00780
00781             while (SDL_PollEvent(&event_temporaire)) {
00782
00783                 if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00784
00785                     if(clic_case(event_temporaire, itemsDemandeSauvegarde[1].rectangle)) {
00786
00787                         for(i = 0; i < 3; i++)
00788                             avancee_niveaux[3].numero_collectible[i] =
00789                                 collectibles_intermediaires[i];
00790
00791                         sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
00792                             touche_sauter_monter,
00793                             touche_descendre, touche_interagir, barre_de_son,
00794                             pseudo,
00795                             (*modeActif), (*personnageActif),
00796                             (*positionActive),
00797                             avancee_niveaux, tailleNiveaux,
00798                             temps_debut_partie, (*compteur_mort), avancee_succes);
00799
00800                             (*programme_lance) = SDL_FALSE;
00801                             clic_effectue = SDL_TRUE;
00802                     }
00803
00804                     else if(clic_case(event_temporaire, itemsDemandeSauvegarde[2].rectangle))
00805                     {
00806                         (*programme_lance) = SDL_FALSE;
00807                         clic_effectue = SDL_TRUE;
00808                     }
00809
00810                     else if(!clic_case(event_temporaire, (*rectangle_demande)))
00811                         clic_effectue = SDL_TRUE;
00812                 }
00813             }
00814         }
00815
00816         SDL_SetWindowResizable((*window), SDL_TRUE);
00817
00818     default:
00819         break;
00820 }
00821
00822     /* Déplacement du personnage */
00823     deplacement_personnage(saut, tombe, position_x, position_y, position_avant_saut,
00824         (*sauter), (*avancer), (*reculer), tile_map, (*personnageActif));
00825
00826     /* Cas où le personnage meurt dans le vide ou par des piques */
00827     if(((tile_map[(position_y)][(position_x)] == 2) || (tile_map[(position_y)][(position_x)] ==
00828         3)) && ((*modeActif) == MODE_NORMAL)) {
00829
00830         (*compteur_mort)++;
00831
00832         if((*personnageActif) == PERSONNAGE_1) {
00833
00834             /* Effet sonore quand le premier personnage meurt */
00835             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_masculin.wav")) == NULL)
00836                 erreur("Chargement de l'effet sonore");
00837         }
00838
00839         else if((*personnageActif) == PERSONNAGE_2) {
00840
00841             /* Effet sonore quand le premier personnage meurt */
00842             if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_feminin.wav")) == NULL)

```

```

00838         erreur("Chargement de l'effet sonore");
00839     }
00840
00841     Mix_PlayChannel(1, effet_sonore, 0);
00842
00843     (*position_x) = (*position_x_initiale);
00844     (*position_y) = (*position_y_initiale);
00845 }
00846
00847 else if((tile_map[(*position_y)][(*position_x)] == 2) || (tile_map[(*position_y)][(*position_x)]
== 3)) && ((*modeActif) == MODE_HARD)) {
00848
00849     if((*personnageActif) == PERSONNAGE_1) {
00850
00851         /* Effet sonore quand le premier personnage meurt */
00852         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_masculin.wav")) == NULL)
00853             erreur("Chargement de l'effet sonore");
00854     }
00855
00856     else if((*personnageActif) == PERSONNAGE_2) {
00857
00858         /* Effet sonore quand le premier personnage meurt */
00859         if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/mort_feminin.wav")) == NULL)
00860             erreur("Chargement de l'effet sonore");
00861     }
00862
00863     Mix_PlayChannel(1, effet_sonore, 0);
00864
00865     (*numero_etage) = 1;
00866
00867     etage_1(position_x, position_y, position_x_initiale, position_y_initiale, tile_map,
00868             renderer, surface, texture_image_mur);
00869 }
00870
00871 /* Cas où le joueur récupère un collectible */
00872
00873 if(((*numero_etage) == 2) && (tile_map[(*position_y)][(*position_x)] == 5) &&
(!avancee_niveaux[3].numero_collectible[0])) {
00874
00875     /* Effet sonore quand on ramasse un collectible */
00876     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00877         erreur("Chargement de l'effet sonore");
00878
00879     Mix_PlayChannel(1, effet_sonore, 0);
00880
00881     avancee_niveaux[3].numero_collectible[0] = 1;
00882 }
00883
00884 if(((*numero_etage) == 3) && (tile_map[(*position_y)][(*position_x)] == 5) &&
(!avancee_niveaux[3].numero_collectible[1])) {
00885
00886     /* Effet sonore quand on ramasse un collectible */
00887     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00888         erreur("Chargement de l'effet sonore");
00889
00890     Mix_PlayChannel(1, effet_sonore, 0);
00891
00892     avancee_niveaux[3].numero_collectible[1] = 1;
00893 }
00894
00895 if(((*numero_etage) == 5) && (tile_map[(*position_y)][(*position_x)] == 5) &&
(!avancee_niveaux[3].numero_collectible[2])) {
00896
00897     /* Effet sonore quand on ramasse un collectible */
00898     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/collectibles.wav")) == NULL)
00899         erreur("Chargement de l'effet sonore");
00900
00901     Mix_PlayChannel(1, effet_sonore, 0);
00902
00903     avancee_niveaux[3].numero_collectible[2] = 1;
00904 }
00905
00906 /* Cas où vous avez fini le niveau */
00907 if (tile_map[(*position_y)][(*position_x)] == 7) {
00908
00909     if((effet_sonore = Mix_LoadWAV("./sons/effets_sonores/fin_niveaux.wav")) == NULL)
00910         erreur("Chargement de l'effet sonore");
00911
00912     Mix_PlayChannel(1, effet_sonore, 0);
00913
00914     /* Mise à jour du rendu */
00915     mise_a_jour_rendu_niveau_4(renderer, texture_image_mur, texture_image_fond,
00916 texture_image_bordure,
00917 texture, rectangle_tile, rectangle_plein_ecran,
00918 texture_image_plein_ecran,
00919 texture_image_porte, texture_image_fin_dernier_niveau,
00920 texture_image_perso_gagnant, rectangle_personnage,

```



```

00919                                     texture_image_pique, avancee_niveaux, (*numero_etage),
00920                                     (*position_x), (*position_y), tile_map, texture_image_croix,
rectangle_croix,
00921                                     (*largeur), (*hauteur), (*largeur_tile), (*hauteur_tile));
00922
00923     SDL_Delay(1000);
00924
00925     avancee_niveaux[3].niveau_fini = 1;
00926
00927     avancee_succes[0] = 1;
00928
00929     if((time(NULL) - temps_debut_partie) < 600)
00930         avancee_succes[2] = 1;
00931
00932     if((*modeActif) == MODE_HARD)
00933         avancee_succes[3] = 1;
00934
00935     if(!(*compteur_mort))
00936         avancee_succes[5] = 1;
00937
00938     if((*modeActif) == MODE_HARD) && ((time(NULL) - temps_debut_partie) < 600))
00939         avancee_succes[6] = 1;
00940
00941     if((*modeActif) == MODE_HARD) && ((time(NULL) - temps_debut_partie) < 600) &&
00942     (!(*compteur_mort)))
00943         avancee_succes[8] = 1;
00944
00945     (*numero_etage) = 1;
00946
00947     (*page_active) = CARTE;
00948 }
00949 /* Mise à jour du rendu */
00950 mise_a_jour_rendu_niveau_4(renderer, texture_image_mur, texture_image_fond, texture_image_bordure,
texture_image_plein_ecran,
texture_image_plein_ecran,
texture_image_porte, texture_image_fin_dernier_niveau,
texture_image_personnage, rectangle_personnage,
texture_image_pique, avancee_niveaux, (*numero_etage),
(*position_x), (*position_y), tile_map, texture_image_croix,
rectangle_croix,
(*largeur), (*hauteur), (*largeur_tile), (*hauteur_tile));
00955 }
00956

```

## 4.19 Référence du fichier fichiers\_c/fonctions\_nouvelle\_partie.c

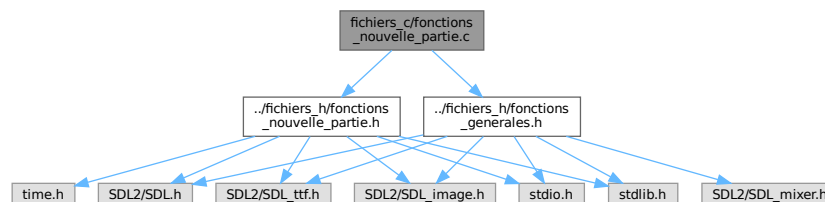
Fichier contenant les fonctions servant à la fenêtre d'une nouvelle partie.

```

#include <../fichiers_h/fonctions_generales.h>
#include <../fichiers_h/fonctions_nouvelle_partie.h>

```

Graphe des dépendances par inclusion de fonctions\_nouvelle\_partie.c:



### Fonctions

- void [initialisation\\_objets\\_nouvelle\\_partie](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, [itemMenu](#) \*titres, [itemMenu](#) \*items, [Menu](#), [itemMenu](#) \*valider)

*Fonction qui permet d'initialiser les différents objets de la nouvelle partie.*

- void [mise\\_a\\_jour\\_rendu\\_nouvelle\\_partie](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_plein\_écran, SDL\_Texture \*\*texture\_image\_plein\_écran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, [modes\\_t](#) modeActif, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Rect \*rectangle\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, SDL\_Rect \*rectangle\_perso\_2, [personnage\\_t](#) personnageActif, [itemMenu](#) \*pseudo, SDL\_Rect \*rectangle\_pseudo, [itemMenu](#) \*titres, int tailleTitres, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, int modeSaisie, [itemMenu](#) \*itemsMenu, [itemMenu](#) \*valider, int largeur, int hauteur)
- void [nouvelle\\_partie](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_plein\_écran, SDL\_Texture \*\*texture\_image\_plein\_écran, SDL\_bool \*plein\_écran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, int \*modeSaisie, [modes\\_t](#) \*modeActif, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Rect \*rectangle\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, SDL\_Rect \*rectangle\_perso\_2, [personnage\\_t](#) \*personnageActif, [itemMenu](#) \*pseudo, SDL\_Rect \*rectangle\_pseudo, [barreDeSon](#) \*barre\_de\_son, int \*pseudo\_valide, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_interagir, [itemMenu](#) \*titres, int tailleTitres, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, [position\\_t](#) \*positionActive, [niveaux](#) \*avancee\_niveaux, int tailleNiveaux, [itemMenu](#) \*itemsMenu, [itemMenu](#) \*valider, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active, time\_t \*temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes)

## 4.19.1 Description détaillée

Fichier contenant les fonctions servant à la fenêtre d'une nouvelle partie.

Définition dans le fichier [fonctions\\_nouvelle\\_partie.c](#).

## 4.19.2 Documentation des fonctions

### 4.19.2.1 initialisation\_objets\_nouvelle\_partie()

```
void initialisation_objets_nouvelle_partie (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_perso_1,
    SDL_Texture ** texture_image_perso_2,
    itemMenu * titres,
    itemMenu * itemsMenu,
    itemMenu * valider )
```

Fonction qui permet d'initialiser les différents objets de la nouvelle partie.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_perso_1</i>	Texture pour le premier personnage.
<i>texture_image_perso_2</i>	Texture pour le deuxième personnage.
<i>titres</i>	Tableau des titres des sections.
<i>itemsMenu</i>	Tableau des éléments de menu.
<i>valider</i>	Élément de menu pour valider.

Voir également

[chargement\\_image](#)

Définition à la ligne 20 du fichier [fonctions\\_nouvelle\\_partie.c](#).

#### 4.19.2.2 mise\_a\_jour\_rendu\_nouvelle\_partie()

```
void mise_a_jour_rendu_nouvelle_partie (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    modes_t modeActif,
    SDL_Texture ** texture_image_perso_1,
    SDL_Rect * rectangle_perso_1,
    SDL_Texture ** texture_image_perso_2,
    SDL_Rect * rectangle_perso_2,
    personnage_t personnageActif,
    itemMenu * pseudo,
    SDL_Rect * rectangle_pseudo,
    itemMenu * titres,
    int tailleTitres,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurNoire,
    int modeSaisie,
    itemMenu * itemsMenu,
    itemMenu * valider,
    int largeur,
    int hauteur )
```

Définition à la ligne 74 du fichier [fonctions\\_nouvelle\\_partie.c](#).

#### 4.19.2.3 nouvelle\_partie()

```
void nouvelle_partie (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    int * modeSaisie,
    modes_t * modeActif,
    SDL_Texture ** texture_image_perso_1,
```

```

SDL_Rect * rectangle_perso_1,
SDL_Texture ** texture_image_perso_2,
SDL_Rect * rectangle_perso_2,
personnage_t * personnageActif,
itemMenu * pseudo,
SDL_Rect * rectangle_pseudo,
barreDeSon * barre_de_son,
int * pseudo_valide,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
SDL_Keycode * touche_interagir,
itemMenu * titres,
int tailleTitres,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Color couleurNoire,
position_t * positionActive,
niveaux * avancee_niveaux,
int tailleNiveaux,
itemMenu * itemsMenu,
itemMenu * valider,
int * largeur,
int * hauteur,
page_t * page_active,
time_t * temps_debut_partie,
int * compteur_mort,
int * avancee_succes )

```

Définition à la ligne 337 du fichier [fonctions\\_nouvelle\\_partie.c](#).

## 4.20 fonctions\_nouvelle\_partie.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_nouvelle_partie.c
00003  * \brief Fichier contenant les fonctions servant à la fenêtre d'une nouvelle partie
00004  */
00005 #include <../fichiers_h/fonctions_generales.h>
00006 #include <../fichiers_h/fonctions_nouvelle_partie.h>
00007
00008 /**
00009  * \fn void initialisation_objets_nouvelle_partie(SDL_Renderer **renderer, SDL_Surface **surface,
00010  * \param SDL_Texture **texture_image_perso_1, SDL_Texture **texture_image_perso_2, itemMenu *titres, itemMenu
00011  * \param itemsMenu, itemMenu *valider)
00012  * \brief Fonction qui permet d'initialiser les différents objets de la nouvelle partie
00013  * \param renderer Pointeur vers le renderer SDL.
00014  * \param surface Pointeur vers la surface SDL.
00015  * \param texture_image_perso_1 Texture pour le premier personnage.
00016  * \param texture_image_perso_2 Texture pour le deuxième personnage.
00017  * \param titres Tableau des titres des sections.
00018  * \param itemsMenu Tableau des éléments de menu.
00019  * \param valider Élément de menu pour valider.
00020  * \see chargement_image
00021  */
00022 void initialisation_objets_nouvelle_partie(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
00023 **texture_image_perso_1,
00024                                     SDL_Texture **texture_image_perso_2,
00025                                     itemMenu *titres, itemMenu *itemsMenu, itemMenu *valider) {
00026
00027     /* Initialisation de l'image du premier personnage */
00028     chargement_image(renderer, surface, texture_image_perso_1,
00029                     ".images/personnages/personnage_masculin.png");
00030 }

```

```

00027      /* Initialisation de l'image du deuxième personnage */
00028      chargement_image(renderer, surface, texture_image_perso_2,
00029      ".images/personnages/personnage_feminin.png");
00029
00030      /* Initialisation des titres du menu nouvelle partie */
00031      sprintf(titres[0].texte, "    Pseudo :    ");
00032      sprintf(titres[1].texte, " Personnage : ");
00033      sprintf(titres[2].texte, " Mode de jeu : ");
00034
00035      /* Initialisation du texte dans les items de la difficulté */
00036      sprintf(itemsMenu[0].texte, " Normal ");
00037      sprintf(itemsMenu[1].texte, " Difficile ");
00038
00039      /* Initialisation du texte dans l'item pour commencer la partie */
00040      sprintf(valider->texte, " Commencer la partie ");
00041  }
00042
00043  /**
00044   * \fn void mise_a_jour_rendu_nouvelle_partie(SDL_Renderer **renderer, SDL_Rect
00045   *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_Rect
00046   *rectangle_retour_en_arriere, SDL_Texture **texture_image_retour_en_arriere, SDL_Rect
00047   *rectangle_options, SDL_Texture **texture_image_options, modes_t modeActif, SDL_Texture
00048   **texture_image_perso_1, SDL_Rect *rectangle_perso_1, SDL_Texture **texture_image_perso_2, SDL_Rect
00049   *rectangle_perso_2, personnage_t personnageActif, itemMenu *pseudo, SDL_Rect *rectangle_pseudo,
00050   itemMenu *titres, int tailleTitres, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font
00051   **police, SDL_Color couleurNoire, itemMenu *itemsMenu, itemMenu *valider, int largeur, int hauteur)
00052   * \brief Fonction qui met à jour le rendu du menu nouvelle partie
00053   * \param renderer Pointeur vers le renderer SDL.
00054   * \param rectangle_plein_ecran Rectangle représentant l'écran entier.
00055   * \param texture_image_plein_ecran Texture pour l'écran entier.
00056   * \param rectangle_retour_en_arriere Rectangle pour le bouton retour.
00057   * \param texture_image_retour_en_arriere Texture pour le bouton retour.
00058   * \param rectangle_options Rectangle pour les options.
00059   * \param texture_image_options Texture pour les options.
00060   * \param modeActif Mode de jeu actif.
00061   * \param texture_image_perso_1 Texture pour le premier personnage.
00062   * \param rectangle_perso_1 Rectangle pour le premier personnage.
00063   * \param texture_image_perso_2 Texture pour le deuxième personnage.
00064   * \param rectangle_perso_2 Rectangle pour le deuxième personnage.
00065   * \param personnageActif Personnage actif.
00066   * \param pseudo Pseudo du joueur.
00067   * \param rectangle_pseudo Rectangle pour le pseudo.
00068   * \param titres Tableau des titres des sections.
00069   * \param tailleTitres Taille du tableau des titres.
00070   * \param surface Pointeur vers la surface SDL.
00071   * \param texture_texte Texture pour le texte.
00072   * \param police Police de caractères.
00073   * \param couleurNoire Couleur noire.
00074   * \param itemsMenu Tableau des éléments de menu.
00075   * \param valider Élément de menu pour valider.
00076   * \param largeur Largeur de l'écran.
00077   * \param hauteur Hauteur de l'écran.
00078   * \see erreur
00079   * \see affichage_texte
00080   */
00081  void mise_a_jour_rendu_nouvelle_partie(SDL_Renderer **renderer, SDL_Rect *rectangle_plein_ecran,
00082  SDL_Texture **texture_image_plein_ecran,
00083  SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
00084  **texture_image_retour_en_arriere,
00085  SDL_Rect *rectangle_options, SDL_Texture
00086  **texture_image_options,
00087  modes_t modeActif, SDL_Texture **texture_image_perso_1,
00088  SDL_Rect *rectangle_perso_1,
00089  SDL_Texture **texture_image_perso_2, SDL_Rect
00090  *rectangle_perso_2, personnage_t personnageActif,
00091  itemMenu *pseudo, SDL_Rect *rectangle_pseudo,
00092  itemMenu *titres, int tailleTitres, SDL_Surface **surface,
00093  SDL_Texture **texture_texte,
00094  TTF_Font **police, SDL_Color couleurNoire, int modeSaisie,
00095  itemMenu *itemsMenu, itemMenu *valider, int largeur, int
00096  hauteur) {
00097      int i;
00098
00099      /* Efface le rendu */
00100      if(SDL_RenderClear((*renderer)) != 0)
00101          erreur("Effacement rendu échoué");
00102
00103      /* Utilisation de la fusion pour un rendu avec transparence */
00104      SDL_SetRenderDrawBlendMode((*renderer), SDL_BLENDMODE_BLEND);
00105
00106      /* Copie la texture de l'image de plein écran */
00107
00108      rectangle_plein_ecran->x = largeur - largeur / 21 - largeur / 53;
00109      rectangle_plein_ecran->y = hauteur / 30;
00110      rectangle_plein_ecran->w = largeur / 21;
00111      rectangle_plein_ecran->h = hauteur / 12;

```

```

00099
00100     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00101         erreur("Copie de la texture");
00102
00103     /* Copie la texture de l'image du retour en arriere */
00104
00105     rectangle_retour_en_arriere->x = largeur / 53;
00106     rectangle_retour_en_arriere->y = hauteur / 30;
00107     rectangle_retour_en_arriere->w = largeur / 21;
00108     rectangle_retour_en_arriere->h = hauteur / 12;
00109
00110     if(SDL_RenderCopy((*renderer), (*texture_image_retour_en_arriere), NULL,
rectangle_retour_en_arriere) != 0)
00111         erreur("Copie de la texture");
00112
00113     /* Copie la texture de l'image des options */
00114
00115     rectangle_options->x = largeur - largeur / 21 - largeur / 53;
00116     rectangle_options->y = hauteur - hauteur / 12 - hauteur / 30;
00117     rectangle_options->w = largeur / 21;
00118     rectangle_options->h = hauteur / 12;
00119
00120     if(SDL_RenderCopy((*renderer), (*texture_image_options), NULL, rectangle_options) != 0)
00121         erreur("Copie de la texture");
00122
00123     /* Dessine les items du menu nouvelle partie */
00124
00125     SDL_SetRenderDrawColor((*renderer), 240, 240, 240, 0);
00126
00127     for(i = 0; i < tailleTitres; i++) {
00128
00129         titres[i].rectangle.x = largeur / 3;
00130         titres[i].rectangle.y = hauteur / 20 + i * hauteur / 4;
00131         titres[i].rectangle.w = largeur / 3;
00132         titres[i].rectangle.h = hauteur / 14;
00133
00134         affichage_texte(renderer, surface, texture_texte, &(titres[i]),
00135             police, couleurNoire);
00136     }
00137
00138     /* Dessine le rectangle pour le pseudo */
00139
00140     rectangle_pseudo->x = largeur / 3;
00141     rectangle_pseudo->y = hauteur / 8;
00142     rectangle_pseudo->w = largeur / 3;
00143     rectangle_pseudo->h = hauteur / 8;
00144
00145     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00146     SDL_RenderFillRect((*renderer), rectangle_pseudo);
00147
00148     SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00149     SDL_RenderDrawRect((*renderer), rectangle_pseudo);
00150
00151     /* Rendu du texte actuel sur la surface texte */
00152
00153     (*surface) = TTF_RenderUTF8_Blended((*police), pseudo->texte, couleurNoire);
00154     (*texture_texte) = SDL_CreateTextureFromSurface((*renderer), (*surface));
00155
00156     /* Récupération des dimensions du texte */
00157
00158     int largeur_texte = 0;
00159     int hauteur_texte = 0;
00160     SDL_QueryTexture((*texture_texte), NULL, NULL, &largeur_texte, &hauteur_texte);
00161
00162     /* Positionnement du texte au centre */
00163
00164     pseudo->rectangle.x = largeur / 3 + largeur / 100;
00165     pseudo->rectangle.y = hauteur / 8;
00166     pseudo->rectangle.w = largeur_texte;
00167     pseudo->rectangle.h = hauteur / 8;
00168
00169     /* Affichage de la texture texture_texte */
00170
00171     SDL_RenderCopy((*renderer), (*texture_texte), NULL, &(pseudo->rectangle));
00172
00173     SDL_FreeSurface((*surface));
00174     SDL_DestroyTexture((*texture_texte));
00175
00176     pseudo->rectangle.y = hauteur / 7;
00177     pseudo->rectangle.w = largeur / 200;
00178     pseudo->rectangle.h = hauteur / 11;
00179
00180     /* Clignotement d'une barre quand le joueur peut écrire son pseudo */
00181     if (modeSaisie) {
00182
00183         static int conteur = 0;
00184

```

```

00185         if (conteur < 30) {
00186             SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00187             pseudo->rectangle.x = largeur / 3 + largeur / 100 + largeur_texte;
00188             SDL_RenderFillRect((*renderer), &(pseudo->rectangle));
00189         }
00190         conteur = (conteur + 1) % 60;
00191     }
00192     SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00193     if(personnageActif == PERSONNAGE_1)
00194         SDL_RenderFillRect((*renderer), rectangle_perso_1);
00195     else
00196         SDL_RenderFillRect((*renderer), rectangle_perso_2);
00197     /* Copie la texture de l'image du premier personnage */
00198     rectangle_perso_1->x = largeur / 4 + largeur / 100;
00199     rectangle_perso_1->y = hauteur / 3 + hauteur / 20;
00200     rectangle_perso_1->w = largeur / 16;
00201     rectangle_perso_1->h = hauteur / 7;
00202     if(SDL_RenderCopy((*renderer), (*texture_image_perso_1), NULL, rectangle_perso_1) != 0)
00203         erreur("Copie de la texture");
00204     /* Copie la texture de l'image du deuxième personnage */
00205     rectangle_perso_2->x = largeur - largeur / 4 - largeur / 16 - largeur / 100;
00206     rectangle_perso_2->y = hauteur / 3 + hauteur / 20;
00207     rectangle_perso_2->w = largeur / 16;
00208     rectangle_perso_2->h = hauteur / 7;
00209     if(SDL_RenderCopy((*renderer), (*texture_image_perso_2), NULL, rectangle_perso_2) != 0)
00210         erreur("Copie de la texture");
00211     /* Dessine les éléments du menu pour la difficulté */
00212     SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00213     if(modeActif == MODE_NORMAL) {
00214         SDL_RenderFillRect((*renderer), &(itemsMenu[0].rectangle));
00215         SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00216         SDL_RenderFillRect((*renderer), &(itemsMenu[1].rectangle));
00217     }
00218     else {
00219         SDL_RenderFillRect((*renderer), &(itemsMenu[1].rectangle));
00220         SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00221         SDL_RenderFillRect((*renderer), &(itemsMenu[0].rectangle));
00222     }
00223     /* Décide de la couleur en fonction de l'onglet actif */
00224     if(modeActif == MODE_NORMAL)
00225         SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00226     else
00227         SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00228     itemsMenu[0].rectangle.x = largeur / 6;
00229     itemsMenu[1].rectangle.x = largeur - largeur / 6 - largeur / 4;
00230     for(i = 0; i < 2; i++) {
00231         if((i) && (modeActif == MODE_HARD))
00232             SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00233         else if(i)
00234             SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);
00235         itemsMenu[i].rectangle.y = hauteur / 2 + hauteur / 6;
00236         itemsMenu[i].rectangle.w = largeur / 4;
00237         itemsMenu[i].rectangle.h = hauteur / 12;
00238         affichage_texte(renderer, surface, texture_texte, &(itemsMenu[i]),
00239             police, couleurNoire);
00240     }
00241     /* Dessine les éléments du menu pour le bouton "Commencer la partie" */
00242     SDL_SetRenderDrawColor((*renderer), 255, 255, 255, 255);

```

```

00272
00273     valider->rectangle.x = largeur / 3;
00274     valider->rectangle.y = hauteur / 2 + hauteur / 3;
00275     valider->rectangle.w = largeur / 3;
00276     valider->rectangle.h = hauteur / 8;
00277
00278     affichage_texte(renderer, surface, texture_texte, valider,
00279                    police, couleurNoire);
00280
00281     SDL_SetRenderDrawColor((*renderer), 240, 240, 240, 0);
00282
00283     /* Affiche le rendu */
00284     SDL_RenderPresent((*renderer));
00285 }
00286
00287 /**
00288  * \fn void nouvelle_partie(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
  *programme_lance, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
  *plein_ecran, SDL_Rect *rectangle_retour_en_arriere, SDL_Texture **texture_image_retour_en_arriere,
  SDL_Rect *rectangle_options, SDL_Texture **texture_image_options, int *modeSaisie, modes_t *modeActif,
  SDL_Texture **texture_image_perso_1, SDL_Rect *rectangle_perso_1, SDL_Texture **texture_image_perso_2,
  SDL_Rect *rectangle_perso_2, personnage_t *personnageActif, itemMenu *pseudo, SDL_Rect
  *rectangle_pseudo, barreDeSon *barre_de_son, int *pseudo_valide, SDL_Keycode *touche_aller_a_droite,
  SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre,
  SDL_Keycode *touche_interagir, itemMenu *titres, int tailleTitres, SDL_Surface **surface, SDL_Texture
  **texture_texte, TTF_Font **police, SDL_Color couleurNoire, position_t *positionActive, niveaux
  *avancee_niveaux, int tailleNiveaux, itemMenu *itemsMenu, itemMenu *valider, int *largeur, int
  *hauteur, page_t *page_active)
00289  * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le menu
  nouvelle_partie
00290  * \param event Événement SDL.
00291  * \param window Pointeur vers la fenêtre SDL.
00292  * \param renderer Pointeur vers le renderer SDL.
00293  * \param programme_lance Booléen pour vérifier si le programme est en cours d'exécution.
00294  * \param rectangle_plein_ecran Rectangle représentant l'écran entier.
00295  * \param texture_image_plein_ecran Texture pour l'écran entier.
00296  * \param plein_ecran Booléen pour vérifier si le jeu est en mode plein écran.
00297  * \param rectangle_retour_en_arriere Rectangle pour le bouton retour.
00298  * \param texture_image_retour_en_arriere Texture pour le bouton retour.
00299  * \param rectangle_options Rectangle pour les options.
00300  * \param texture_image_options Texture pour les options.
00301  * \param modeSaisie Mode de saisie pour le pseudo.
00302  * \param modeActif Mode de jeu actif.
00303  * \param texture_image_perso_1 Texture pour le premier personnage.
00304  * \param rectangle_perso_1 Rectangle pour le premier personnage.
00305  * \param texture_image_perso_2 Texture pour le deuxième personnage.
00306  * \param rectangle_perso_2 Rectangle pour le deuxième personnage.
00307  * \param personnageActif Personnage actif.
00308  * \param pseudo Pseudo du joueur.
00309  * \param rectangle_pseudo Rectangle pour le pseudo.
00310  * \param barre_de_son Barre de son.
00311  * \param pseudo_valide Booléen pour vérifier si le pseudo est valide.
00312  * \param touche_aller_a_droite Touche pour aller à droite.
00313  * \param touche_aller_a_gauche Touche pour aller à gauche.
00314  * \param touche_sauter_monter Touche pour sauter ou monter.
00315  * \param touche_descendre Touche pour descendre.
00316  * \param touche_interagir Touche pour interagir.
00317  * \param titres Tableau des titres des sections.
00318  * \param tailleTitres Taille du tableau des titres.
00319  * \param surface Pointeur vers la surface SDL.
00320  * \param texture_texte Texture pour le texte.
00321  * \param police Police de caractères.
00322  * \param couleurNoire Couleur noire.
00323  * \param positionActive Position active.
00324  * \param avancee_niveaux Tableau de l'avancée des niveaux.
00325  * \param tailleNiveaux Taille du tableau de l'avancée des niveaux.
00326  * \param itemsMenu Tableau des éléments de menu.
00327  * \param valider Élément de menu pour valider.
00328  * \param largeur Largeur de l'écran.
00329  * \param hauteur Hauteur de l'écran.
00330  * \param page_active Page active.
00331  * \see redimensionnement_fenetre
00332  * \see clic_case
00333  * \see clic_plein_ecran
00334  * \see sauvegarder_partie
00335  * \see mise_a_jour_rendu_nouvelle_partie
00336  */
00337 void nouvelle_partie(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
  *programme_lance,
00338                      SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
  SDL_bool *plein_ecran,
00339                      SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
  **texture_image_retour_en_arriere,
00340                      SDL_Rect *rectangle_options, SDL_Texture **texture_image_options, int
  *modeSaisie,
00341                      modes_t *modeActif, SDL_Texture **texture_image_perso_1, SDL_Rect
  *rectangle_perso_1,

```



```

00342         SDL_Texture **texture_image_perso_2, SDL_Rect *rectangle_perso_2, personnage_t
00343         *personnageActif,
00344         itemMenu *pseudo, SDL_Rect *rectangle_pseudo, barreDeSon *barre_de_son, int
00345         *pseudo_valide,
00346         SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
00347         SDL_Keycode *touche_sauter_monter,
00348         SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, itemMenu *titres,
00349         int tailleTitres, SDL_Surface **surface, SDL_Texture **texture_texte,
00350         TTF_Font **police, SDL_Color couleurNoire, position_t *positionActive, niveaux
00351         *avancee_niveaux, int tailleNiveaux,
00352         itemMenu *itemsMenu, itemMenu *valider, int *largeur, int *hauteur, page_t
00353         *page_active,
00354         time_t *temps_debut_partie, int *compteur_mort, int *avancee_succes) {
00355
00356     int i;
00357
00358     while(SDL_PollEvent(event)) {
00359
00360         switch(event->type) {
00361
00362             /* Gestion de l'événement de redimensionnement de la fenêtre */
00363             case SDL_WINDOWEVENT:
00364                 redimensionnement_fenetre((*event), largeur, hauteur);
00365
00366             /* Actualisation de la taille de la police */
00367             (*police) = TTF_OpenFont("./polices/04B_11__.TTF", (*largeur) / 35);
00368
00369             break;
00370
00371             /* Si l'utilisateur tape quelque chose */
00372             case SDL_TEXTINPUT:
00373                 if((*modeSaisie))
00374                     /* Concatène le texte saisi au pseudo */
00375                     if(strlen(pseudo->texte) + strlen(event->text.text) <= 10)
00376                         strcat(pseudo->texte, event->text.text);
00377
00378                 break;
00379
00380             /* Si l'utilisateur clic quelque part */
00381             case SDL_MOUSEBUTTONDOWN:
00382                 if(clic_case((*event), (*rectangle_pseudo)))
00383                     (*modeSaisie) = 1;
00384                 else
00385                     (*modeSaisie) = 0;
00386
00387                 if(clic_case((*event), itemsMenu[0].rectangle))
00388                     (*modeActif) = MODE_NORMAL;
00389                 else if(clic_case((*event), itemsMenu[1].rectangle))
00390                     (*modeActif) = MODE_HARD;
00391
00392                 if(clic_case((*event), (*rectangle_perso_1)))
00393                     (*personnageActif) = PERSONNAGE_1;
00394                 else if(clic_case((*event), (*rectangle_perso_2)))
00395                     (*personnageActif) = PERSONNAGE_2;
00396
00397             /* Options plein écran, options et retour en arrière */
00398
00399             if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window))
00400                 redimensionnement_fenetre((*event), largeur, hauteur);
00401
00402             if(clic_case((*event), (*rectangle_retour_en_arriere)))
00403                 (*page_active) = MENU_PRINCIPAL;
00404
00405             if(clic_case((*event), (*rectangle_options)))
00406                 (*page_active) = OPTIONS;
00407
00408             /* Options valider */
00409             if((clic_case((*event), valider->rectangle)) && (strcmp(pseudo->texte, "\\0"))) {
00410
00411                 for(i = 0; pseudo->texte[i] != '\\0'; i++)
00412                     if(pseudo->texte[i] != ' ')
00413                         (*pseudo_valide) = 1;
00414
00415                 if((*pseudo_valide)) {
00416
00417                     (*positionActive) = NIVEAU1;
00418
00419                     for(i = 0; i < tailleNiveaux; i++) {
00420
00421                         avancee_niveaux[i].niveau_fini = 0;
00422                         avancee_niveaux[i].numero_collectible[0] = 0;
00423                         avancee_niveaux[i].numero_collectible[1] = 0;
00424                         avancee_niveaux[i].numero_collectible[2] = 0;
00425                     }
00426
00427                     (*temps_debut_partie) = time(NULL);
00428                     (*compteur_mort) = 0;
00429

```

```

00423
00424         for(i = 0; i < 10; i++)
00425             avancee_succes[i] = 0;
00426
00427         sauvegarder_partie(touche_aller_a_droite, touche_aller_a_gauche,
00428             touche_sauter_monter,
00429             touche_descendre, touche_interagir, barre_de_son, pseudo,
00430             (*modeActif), (*personnageActif), (*positionActive),
00431             avancee_niveaux, tailleNiveaux, (*temps_debut_partie),
00432             (*compteur_mort), avancee_succes);
00433
00434         (*page_active) = INTRODUCTION;
00435     }
00436 }
00437
00438     break;
00439
00440     /* Mode saisie du pseudo */
00441     case SDL_KEYDOWN:
00442         if((*modeSaisie))
00443             if(event->key.keysym.sym == SDLK_BACKSPACE && strlen(pseudo->texte) > 0)
00444                 pseudo->texte[strlen(pseudo->texte) - 1] = '\0'; /* Supprime le dernier
00445                 caractère du pseudo */
00446
00447         break;
00448
00449     /* Quitter le programme */
00450     case SDL_QUIT:
00451         (*programme_lance) = SDL_FALSE;
00452         break;
00453
00454     default:
00455         break;
00456 }
00457
00458 /* Mise à jour du rendu */
00459 mise_a_jour_rendu_nouvelle_partie(renderer, rectangle_plein_ecran, texture_image_plein_ecran,
00460     rectangle_retour_en_arriere, texture_image_retour_en_arriere,
00461     rectangle_options, texture_image_options,
00462     (*modeActif), texture_image_perso_1, rectangle_perso_1,
00463     texture_image_perso_2, rectangle_perso_2, (*personnageActif),
00464     pseudo, rectangle_pseudo,
00465     titres, tailleTitres, surface, texture_texte,
00466     police, couleurNoire, (*modeSaisie),
00467     itemsMenu, valider, (*largeur), (*hauteur));
00468 }

```

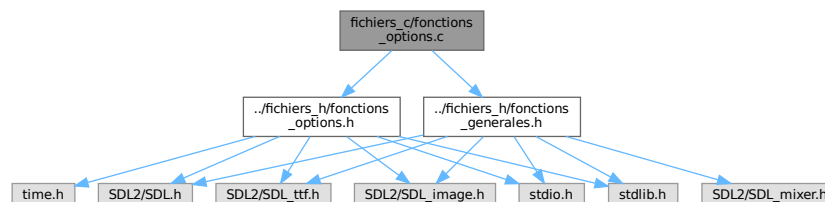
## 4.21 Référence du fichier fichiers\_c/fonctions\_options.c

Fichier qui réunit les fonctions s'occupant de la fenêtre des options du jeu.

```
#include <../fichiers_h/fonctions_generales.h>
```

```
#include <../fichiers_h/fonctions_options.h>
```

Graphe des dépendances par inclusion de fonctions\_options.c:



## Fonctions

- void [initialisation\\_objets\\_options](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_hautParleurActif, SDL\_Texture \*\*texture\_image\_hautParleurDesactive, [itemMenu](#) \*titre, [itemMenu](#) \*itemsMenu, [itemMenu](#) \*itemsTouches, [itemMenu](#) \*itemsBarres)  
*Fonction qui permet d'initialiser les différents objets des options.*
- void [mise\\_a\\_jour\\_rendu\\_options](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_hautParleurActif, SDL\_Texture \*\*texture\_image\_hautParleurDesactive, SDL\_bool \*sonsActifs, SDL\_Rect \*rectangles\_boutons\_sons, [option\\_t](#) ongletActif, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, int selection\_touche, [itemMenu](#) \*itemsMenu, int tailleMenu, [itemMenu](#) \*itemsTouches, int tailleTouches, [barreDeSon](#) \*barre\_de\_son, int tailleBarres, [itemMenu](#) \*itemsBarres, int largeur, int hauteur)  
*Fonction qui met à jour le rendu des options.*
- void [mise\\_a\\_jour\\_barre\\_de\\_son](#) (SDL\_Event \*event, [barreDeSon](#) \*barre\_de\_son, SDL\_bool \*sonsActifs)  
*Fonction qui permet de mettre à jour les barres de sons.*
- void [mise\\_a\\_jour\\_touches](#) (SDL\_Event \*event, SDL\_Keycode \*touche, int \*selection\_touche, [itemMenu](#) \*itemsTouches)  
*Fonction qui permet de mettre à jour les touches.*
- void [options](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_hautParleurActif, SDL\_Rect \*rectangle\_demande\_sauvegarde, [itemMenu](#) \*itemsDemandeSauvegarde, int tailleDemandeSauvegarde, SDL\_Texture \*\*texture\_image\_hautParleurDesactive, SDL\_bool \*sonsActifs, SDL\_Rect \*rectangles\_boutons\_sons, [option\\_t](#) \*ongletActif, [itemMenu](#) \*pseudo, [modes\\_t](#) \*modeActif, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, [niveaux](#) \*avancee\_niveaux, int tailleNiveaux, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, int \*selection\_touche, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_interagir, SDL\_Color couleurNoire, [itemMenu](#) \*itemsMenu, int tailleMenu, [itemMenu](#) \*itemsTouches, int tailleTouches, [barreDeSon](#) \*barre\_de\_son, int tailleBarres, [itemMenu](#) \*itemsBarres, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active, [page\\_t](#) \*page\_precedente, int \*maintient\_clic, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes)

### 4.21.1 Description détaillée

Fichier qui réunit les fonctions s'occupant de la fenêtre des options du jeu.

Définition dans le fichier [fonctions\\_options.c](#).

### 4.21.2 Documentation des fonctions

#### 4.21.2.1 initialisation\_objets\_options()

```
void initialisation_objets_options (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_hautParleurActif,
    SDL_Texture ** texture_image_hautParleurDesactive,
    itemMenu * titre,
    itemMenu * itemsMenu,
    itemMenu * itemsTouches,
    itemMenu * itemsBarres )
```

Fonction qui permet d'initialiser les différents objets des options.

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_hautParleurActif</i>	Texture pour l'icône de haut-parleur actif.
<i>texture_image_hautParleurDesactive</i>	Texture pour l'icône de haut-parleur désactivé.
<i>titre</i>	Pointeur vers l'item du titre du menu.
<i>itemsMenu</i>	Tableau des items du menu.
<i>itemsTouches</i>	Tableau des items pour les touches du clavier.
<i>itemsBarres</i>	Tableau des items pour les barres de volume.

## Voir également

[chargement\\_image](#)

Définition à la ligne 19 du fichier [fonctions\\_options.c](#).

## 4.21.2.2 mise\_a\_jour\_barre\_de\_son()

```
void mise_a_jour_barre_de_son (
    SDL_Event * event,
    barreDeSon * barre_de_son,
    SDL_bool * sonsActifs )
```

Fonction qui permet de mettre à jour les barres de sons.

## Paramètres

<i>event</i>	Pointeur vers l'événement SDL.
<i>barre_de_son</i>	Pointeur vers la barre de son.
<i>sonsActifs</i>	Pointeur booléen pour l'état des sons.

Définition à la ligne 297 du fichier [fonctions\\_options.c](#).

## 4.21.2.3 mise\_a\_jour\_rendu\_options()

```
void mise_a_jour_rendu_options (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Texture ** texture_image_hautParleurActif,
    SDL_Texture ** texture_image_hautParleurDesactive,
    SDL_bool * sonsActifs,
    SDL_Rect * rectangles_boutons_sons,
    option_t ongletActif,
    itemMenu * titre,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
```

```

TTF_Font ** police,
SDL_Color couleurNoire,
int selection_touche,
itemMenu * itemsMenu,
int tailleMenu,
itemMenu * itemsTouches,
int tailleTouches,
barreDeSon * barre_de_son,
int tailleBarres,
itemMenu * itemsBarres,
int largeur,
int hauteur )

```

Fonction qui met à jour le rendu des options.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>rectangle_plein_ecran</i>	Rectangle pour le plein écran.
<i>texture_image_plein_ecran</i>	Texture pour l'image du plein écran.
<i>rectangle_retour_en_arriere</i>	Rectangle pour le bouton de retour en arrière.
<i>texture_image_retour_en_arriere</i>	Texture pour l'image du bouton de retour en arrière.
<i>texture_image_hautParleurActif</i>	Texture pour l'icône de haut-parleur actif.
<i>texture_image_hautParleurDesactive</i>	Texture pour l'icône de haut-parleur désactivé.
<i>sonsActifs</i>	Pointeur booléen pour l'état des sons.
<i>rectangles_boutons_sons</i>	Tableau des rectangles pour les boutons de sons.
<i>ongletActif</i>	Onglet actif dans le menu des options.
<i>titre</i>	Pointeur vers l'item du titre du menu.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_texte</i>	Texture pour le texte.
<i>police</i>	Pointeur vers la police TTF.
<i>couleurNoire</i>	Couleur noire.
<i>selection_touche</i>	Sélection de la touche du clavier.
<i>itemsMenu</i>	Tableau des items du menu.
<i>tailleMenu</i>	Taille du tableau des items du menu.
<i>itemsTouches</i>	Tableau des items pour les touches du clavier.
<i>tailleTouches</i>	Taille du tableau des items pour les touches du clavier.
<i>barre_de_son</i>	Tableau des barres de son.
<i>tailleBarres</i>	Taille du tableau des barres de son.
<i>itemsBarres</i>	Tableau des items pour les barres de volume.
<i>largeur</i>	Largeur de la fenêtre.
<i>hauteur</i>	Hauteur de la fenêtre.

#### Voir également

[erreur](#)  
[affichage\\_texte](#)

Définition à la ligne 83 du fichier `fonctions_options.c`.

#### 4.21.2.4 mise\_a\_jour\_touches()

```
void mise_a_jour_touches (
    SDL_Event * event,
    SDL_Keycode * touche,
    int * selection_touche,
    itemMenu * itemsTouches )
```

Fonction qui permet de mettre à jour les touches.

##### Paramètres

<i>event</i>	Pointeur vers l'événement SDL.
<i>touche</i>	Pointeur vers la touche sélectionnée.
<i>selection_touche</i>	Pointeur vers l'indice de la touche sélectionnée.
<i>itemsTouches</i>	Tableau des éléments de menu des touches.

Définition à la ligne 315 du fichier [fonctions\\_options.c](#).

#### 4.21.2.5 options()

```
void options (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Texture ** texture_image_hautParleurActif,
    SDL_Rect * rectangle_demande_sauvegarde,
    itemMenu * itemsDemandeSauvegarde,
    int tailleDemandeSauvegarde,
    SDL_Texture ** texture_image_hautParleurDesactive,
    SDL_bool * sonsActifs,
    SDL_Rect * rectangles_boutons_sons,
    option_t * ongletActif,
    itemMenu * pseudo,
    modes_t * modeActif,
    personnage_t * personnageActif,
    position_t * positionActive,
    niveaux * avancee_niveaux,
    int tailleNiveaux,
    itemMenu * titre,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    int * selection_touche,
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_sauter_monter,
    SDL_Keycode * touche_descendre,
```

```

    SDL_Keycode * touche_interagir,
    SDL_Color couleurNoire,
    itemMenu * itemsMenu,
    int tailleMenu,
    itemMenu * itemsTouches,
    int tailleTouches,
    barreDeSon * barre_de_son,
    int tailleBarres,
    itemMenu * itemsBarres,
    int * largeur,
    int * hauteur,
    page_t * page_active,
    page_t * page_precedente,
    int * maintient_clic,
    time_t temps_debut_partie,
    int * compteur_mort,
    int * avancee_succes )

```

Définition à la ligne 381 du fichier [fonctions\\_options.c](#).

## 4.22 fonctions\_options.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file fonctions_options.c
00003  * \brief Fichier qui réunit les fonctions s'occupant de la fenêtre des options du jeu
00004  */
00005 #include <../fichiers_h/fonctions_generales.h>
00006 #include <../fichiers_h/fonctions_options.h>
00007
00008 /** \brief Fonction qui permet d'initialiser les différents objets des options
00009  * \param renderer Pointeur vers le renderer SDL.
00010  * \param surface Pointeur vers la surface SDL.
00011  * \param texture_image_hautParleurActif Texture pour l'icône de haut-parleur actif.
00012  * \param texture_image_hautParleurDesactive Texture pour l'icône de haut-parleur désactivé.
00013  * \param titre Pointeur vers l'item du titre du menu.
00014  * \param itemsMenu Tableau des items du menu.
00015  * \param itemsTouches Tableau des items pour les touches du clavier.
00016  * \param itemsBarres Tableau des items pour les barres de volume.
00017  * \see chargement_image
00018  */
00019 void initialisation_objets_options(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_hautParleurActif,
00020                                     SDL_Texture **texture_image_hautParleurDesactive,
00021                                     itemMenu *titre, itemMenu *itemsMenu, itemMenu *itemsTouches,
00022                                     itemMenu *itemsBarres) {
00023     /* Initialisation de l'image du haut parleur actif */
00024     chargement_image(renderer, surface, texture_image_hautParleurActif,
00025                     "./images/options/haut_parleur_actif.png");
00026     /* Initialisation de l'image du haut parleur desactif */
00027     chargement_image(renderer, surface, texture_image_hautParleurDesactive,
00028                     "./images/options/haut_parleur_desactive.png");
00029     /* Initialisation du titre des options */
00030     sprintf(titre->texte, " Options ");
00031
00032     /* Initialisation du texte des onglets */
00033     sprintf(itemsMenu[0].texte, "      Son      ");
00034     sprintf(itemsMenu[1].texte, "      Touches      ");
00035
00036     /* Initialisation du texte dans les items */
00037     sprintf(itemsTouches[0].texte, " Aller vers la droite : ");
00038     sprintf(itemsTouches[1].texte, "      %s      ", SDL_GetKeyName(SDLK_RIGHT));
00039     sprintf(itemsTouches[2].texte, " Aller vers la gauche : ");
00040     sprintf(itemsTouches[3].texte, "      %s      ", SDL_GetKeyName(SDLK_LEFT));
00041     sprintf(itemsTouches[4].texte, " Sauter / monter : ");
00042     sprintf(itemsTouches[5].texte, "      %s      ", SDL_GetKeyName(SDLK_UP));
00043     sprintf(itemsTouches[6].texte, " Descendre : ");
00044     sprintf(itemsTouches[7].texte, "      %s      ", SDL_GetKeyName(SDLK_DOWN));
00045     sprintf(itemsTouches[8].texte, " Interagir : ");

```

```

00046     sprintf(itemsTouches[9].texte, "          %s          ", SDL_GetKeyName(SDLK_SPACE));
00047     sprintf(itemsBarres[0].texte, "          Musique :          ");
00048     sprintf(itemsBarres[1].texte, "          Effets sonores :          ");
00049 }
00050
00051
00052 /**
00053  * \fn void mise_a_jour_rendu_options(SDL_Renderer **renderer, SDL_Rect *rectangle_plein_ecran,
    SDL_Texture **texture_image_plein_ecran, SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
    **texture_image_retour_en_arriere, SDL_Texture **texture_image_hautParleurActif, SDL_Texture
    **texture_image_hautParleurDesactive, SDL_bool *sonsActifs, SDL_Rect *rectangles_boutons_sons,
    option_t ongletActif, itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font
    **police, SDL_Color couleurNoire, int selection_touche, itemMenu *itemsMenu, int tailleMenu, itemMenu
    *itemsTouches, int tailleTouches, barreDeSon *barre_de_son, int tailleBarres, itemMenu *itemsBarres,
    int largeur, int hauteur) {
00054  * \brief Fonction qui met à jour le rendu des options
00055  * \param renderer Pointeur vers le renderer SDL.
00056  * \param rectangle_plein_ecran Rectangle pour le plein écran.
00057  * \param texture_image_plein_ecran Texture pour l'image du plein écran.
00058  * \param rectangle_retour_en_arriere Rectangle pour le bouton de retour en arrière.
00059  * \param texture_image_retour_en_arriere Texture pour l'image du bouton de retour en arrière.
00060  * \param texture_image_hautParleurActif Texture pour l'icône de haut-parleur actif.
00061  * \param texture_image_hautParleurDesactive Texture pour l'icône de haut-parleur désactivé.
00062  * \param sonsActifs Pointeur booléen pour l'état des sons.
00063  * \param rectangles_boutons_sons Tableau des rectangles pour les boutons de sons.
00064  * \param ongletActif Onglet actif dans le menu des options.
00065  * \param titre Pointeur vers l'item du titre du menu.
00066  * \param surface Pointeur vers la surface SDL.
00067  * \param texture_texte Texture pour le texte.
00068  * \param police Pointeur vers la police TTF.
00069  * \param couleurNoire Couleur noire.
00070  * \param selection_touche Sélection de la touche du clavier.
00071  * \param itemsMenu Tableau des items du menu.
00072  * \param tailleMenu Taille du tableau des items du menu.
00073  * \param itemsTouches Tableau des items pour les touches du clavier.
00074  * \param tailleTouches Taille du tableau des items pour les touches du clavier.
00075  * \param barre_de_son Tableau des barres de son.
00076  * \param tailleBarres Taille du tableau des barres de son.
00077  * \param itemsBarres Tableau des items pour les barres de volume.
00078  * \param largeur Largeur de la fenêtre.
00079  * \param hauteur Hauteur de la fenêtre.
00080  * \see erreur
00081  * \see affichage_texte
00082  */
00083 void mise_a_jour_rendu_options(SDL_Renderer **renderer, SDL_Rect *rectangle_plein_ecran, SDL_Texture
    **texture_image_plein_ecran,
00084     SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
    **texture_image_retour_en_arriere,
00085     SDL_Texture **texture_image_hautParleurActif,
00086     SDL_Texture **texture_image_hautParleurDesactive, SDL_bool *sonsActifs,
00087     SDL_Rect *rectangles_boutons_sons, option_t ongletActif,
00088     itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte,
    TTF_Font **police,
00089     SDL_Color couleurNoire, int selection_touche,
00090     itemMenu *itemsMenu, int tailleMenu, itemMenu *itemsTouches, int
    tailleTouches,
00091     barreDeSon *barre_de_son, int tailleBarres, itemMenu *itemsBarres,
    int largeur, int hauteur) {
00092
00093     int i;
00094
00095     /* Efface le rendu */
00096     if(SDL_RenderClear((*renderer)) != 0)
00097         erreur("Effacement rendu échoué");
00098
00099     /* Utilisation de la fusion pour un rendu avec transparence */
00100     SDL_SetRenderDrawBlendMode((*renderer), SDL_BLENDMODE_BLEND);
00101
00102     /* Copie la texture de l'image de plein écran */
00103
00104     rectangle_plein_ecran->x = largeur - largeur / 21 - largeur / 53;
00105     rectangle_plein_ecran->y = hauteur / 30;
00106     rectangle_plein_ecran->w = largeur / 21;
00107     rectangle_plein_ecran->h = hauteur / 12;
00108
00109     if(SDL_RenderCopy((*renderer), (*texture_image_plein_ecran), NULL, rectangle_plein_ecran) != 0)
00110         erreur("Copie de la texture");
00111
00112     /* Copie la texture de l'image du retour en arrière */
00113
00114     rectangle_retour_en_arriere->x = largeur / 53;
00115     rectangle_retour_en_arriere->y = hauteur / 30;
00116     rectangle_retour_en_arriere->w = largeur / 21;
00117     rectangle_retour_en_arriere->h = hauteur / 12;
00118
00119     if(SDL_RenderCopy((*renderer), (*texture_image_retour_en_arriere), NULL,
    rectangle_retour_en_arriere) != 0)

```



```

00121         erreur("Copie de la texture");
00122
00123     /* Dessine le titre des options */
00124     SDL_SetRenderDrawColor((*renderer), 240, 240, 240, 255);
00125
00126     titre->rectangle.x = largeur / 3;
00127     titre->rectangle.y = hauteur / 15;
00128     titre->rectangle.w = largeur / 3;
00129     titre->rectangle.h = hauteur / 10;
00130
00131     affichage_texte(renderer, surface, texture_texte, titre,
00132                    police, couleurNoire);
00133
00134     /* Décide de la couleur en fonction de l'onglet actif */
00135     if(ongletActif == ONGLET_SON)
00136         SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00137     else
00138         SDL_SetRenderDrawColor((*renderer), 180, 180, 180, 255);
00139
00140     /* Dessine les éléments des onglets */
00141     for (i = 0; i < tailleMenu; i++) {
00142
00143         if((i) && (ongletActif == ONGLET_TOUCHES))
00144             SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00145         else if(i)
00146             SDL_SetRenderDrawColor((*renderer), 180, 180, 180, 255);
00147
00148         itemsMenu[i].rectangle.x = i * largeur / 2;
00149         itemsMenu[i].rectangle.y = hauteur / 15 * 3;
00150         itemsMenu[i].rectangle.w = largeur / 2;
00151         itemsMenu[i].rectangle.h = hauteur / 10;
00152
00153         affichage_texte(renderer, surface, texture_texte, &(itemsMenu[i]),
00154                        police, couleurNoire);
00155     }
00156
00157     if(ongletActif == ONGLET_SON) {
00158
00159         /* Dessine les items pour les deux barres de sons */
00160         SDL_SetRenderDrawColor((*renderer), 180, 180, 180, 255);
00161
00162         for (i = 0; i < tailleBarres; i++) {
00163
00164             itemsBarres[i].rectangle.x = largeur / 8;
00165             itemsBarres[i].rectangle.y = hauteur / 2 + hauteur / 50 + i * hauteur / 5 - hauteur / 100;
00166             itemsBarres[i].rectangle.w = largeur / 2 - largeur / 7;
00167             itemsBarres[i].rectangle.h = hauteur / 15;
00168
00169             affichage_texte(renderer, surface, texture_texte, &(itemsBarres[i]),
00170                            police, couleurNoire);
00171         }
00172
00173         /* Dessine les boutons sons actifs ou desactifs des deux barres de sons */
00174         for (i = 0; i < tailleBarres; i++) {
00175
00176             rectangles_boutons_sons[i].x = largeur - largeur / 9;
00177             rectangles_boutons_sons[i].y = hauteur / 2 + hauteur / 50 + i * hauteur / 5;
00178             rectangles_boutons_sons[i].w = largeur / 35;
00179             rectangles_boutons_sons[i].h = hauteur / 20;
00180
00181             if(sonsActifs[i]) {
00182                 if(SDL_RenderCopy((*renderer), (*texture_image_hautParleurActif), NULL,
00183                                &(rectangles_boutons_sons[i])) != 0)
00184                     erreur("Copie de la texture");
00185             }
00186             else
00187                 if(SDL_RenderCopy((*renderer), (*texture_image_hautParleurDesactive), NULL,
00188                                &(rectangles_boutons_sons[i])) != 0)
00189                     erreur("Copie de la texture");
00190         }
00191
00192         /* Dessine les rectangles et les curseurs des deux barres de sons */
00193         for (i = 0; i < tailleBarres; i++) {
00194
00195             barre_de_son[i].barre.x = largeur - largeur / 8 - (largeur / 2 - largeur / 7);
00196             barre_de_son[i].barre.y = hauteur / 2 + hauteur / 50 + i * hauteur / 5;
00197             barre_de_son[i].barre.w = largeur / 2 - largeur / 7;
00198             barre_de_son[i].barre.h = hauteur / 20;
00199
00200             barre_de_son[i].curseur.x = barre_de_son[i].barre.x + (largeur / 2 - largeur / 7) *
00201             barre_de_son[i].volume - largeur / 45 / 2;
00202             barre_de_son[i].curseur.y = hauteur / 2 + i * hauteur / 5;
00203             barre_de_son[i].curseur.w = largeur / 45;
00204             barre_de_son[i].curseur.h = hauteur / 12;
00205
00206             SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00207             SDL_RenderFillRect((*renderer), &(barre_de_son[i].barre));
00208         }
00209     }

```

```

00205
00206         SDL_SetRenderDrawColor((*renderer), 0, 0, 0, 255);
00207         SDL_RenderFillRect((*renderer), &(barre_de_son[i].curseur));
00208     }
00209
00210     for (i = 0; i < tailleTouches; i++) {
00211         itemsTouches[i].rectangle.x = 0;
00212         itemsTouches[i].rectangle.y = 0;
00213         itemsTouches[i].rectangle.w = 0;
00214         itemsTouches[i].rectangle.h = 0;
00215     }
00216 }
00217
00218 else if(ongletActif == ONGLET_TOUCHES) {
00219     for (i = 0; i < tailleBarres; i++) {
00220         rectangles_boutons_sons[i].x = 0;
00221         rectangles_boutons_sons[i].y = 0;
00222         rectangles_boutons_sons[i].w = 0;
00223         rectangles_boutons_sons[i].h = 0;
00224
00225         barre_de_son[i].barre.x = 0;
00226         barre_de_son[i].barre.y = 0;
00227         barre_de_son[i].barre.w = 0;
00228         barre_de_son[i].barre.h = 0;
00229
00230         barre_de_son[i].curseur.x = 0;
00231         barre_de_son[i].curseur.y = 0;
00232         barre_de_son[i].curseur.w = 0;
00233         barre_de_son[i].curseur.h = 0;
00234
00235         itemsBarres[i].rectangle.x = 0;
00236         itemsBarres[i].rectangle.y = 0;
00237         itemsBarres[i].rectangle.w = 0;
00238         itemsBarres[i].rectangle.h = 0;
00239     }
00240
00241     /* Dessine les items pour les cinq touches différentes */
00242     SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00243
00244     for (i = 0; i < tailleTouches; i+=2) {
00245         itemsTouches[i].rectangle.x = largeur / 7;
00246         itemsTouches[i].rectangle.y = hauteur / 2 - hauteur / 7 + i * hauteur / 15;
00247         itemsTouches[i].rectangle.w = largeur / 2 - largeur / 7;
00248         itemsTouches[i].rectangle.h = hauteur / 15;
00249
00250         affichage_texte(renderer, surface, texture_texte, &(itemsTouches[i]),
00251             police, couleurNoire);
00252     }
00253
00254     /* Dessine les rectangles pour les cinq touches différentes */
00255     for (i = 1; i < tailleTouches; i+=2) {
00256         itemsTouches[i].rectangle.x = largeur / 2;
00257         itemsTouches[i].rectangle.y = hauteur / 2 - hauteur / 7 + (i-1) * hauteur / 15;
00258         itemsTouches[i].rectangle.w = largeur / 2 - largeur / 7;
00259         itemsTouches[i].rectangle.h = hauteur / 15;
00260
00261         if(selection_touche == i) {
00262             SDL_SetRenderDrawColor((*renderer), 175, 95, 185, 255);
00263             SDL_RenderFillRect((*renderer), &(itemsTouches[i].rectangle));
00264
00265             itemsTouches[i].rectangle.y = hauteur / 2 - hauteur / 7 + (i-1) * hauteur / 15;
00266             itemsTouches[i].rectangle.w = largeur / 2 - largeur / 8;
00267             itemsTouches[i].rectangle.h = hauteur / 16;
00268         }
00269
00270         SDL_SetRenderDrawColor((*renderer), 180, 180, 180, 255);
00271
00272         affichage_texte(renderer, surface, texture_texte, &(itemsTouches[i]),
00273             police, couleurNoire);
00274     }
00275 }
00276
00277 SDL_SetRenderDrawColor((*renderer), 240, 240, 240, 0);
00278
00279 /* Affiche le rendu */
00280 SDL_RenderPresent((*renderer));
00281 }
00282
00283 /**
00284 * \fn void mise_a_jour_barre_de_son(SDL_Event *event, barreDeSon *barre_de_son, SDL_bool *sonsActifs)

```

```

00292 * \brief Fonction qui permet de mettre à jour les barres de sons
00293 * @param event Pointeur vers l'événement SDL.
00294 * @param barre_de_son Pointeur vers la barre de son.
00295 * @param sonsActifs Pointeur booléen pour l'état des sons.
00296 */
00297 void mise_a_jour_barre_de_son(SDL_Event *event, barreDeSon *barre_de_son, SDL_bool *sonsActifs) {
00298     barre_de_son->volume_precedent = barre_de_son->volume;
00299     barre_de_son->volume = (event->motion.x - barre_de_son->barre.x) * 1.0 / barre_de_son->barre.w;
00300     if((*sonsActifs) == SDL_FALSE) && (barre_de_son->volume != 0))
00301         (*sonsActifs) = SDL_TRUE;
00302     if(barre_de_son->volume == 0)
00303         (*sonsActifs) = SDL_FALSE;
00304 }
00305
00306 /**
00307 * \fn void mise_a_jour_touches(SDL_Event *event, SDL_Keycode *touche, int *selection_touche, itemMenu
00308 *itemsTouches)
00309 * \brief Fonction qui permet de mettre à jour les touches
00310 * @param event Pointeur vers l'événement SDL.
00311 * @param touche Pointeur vers la touche sélectionnée.
00312 * @param selection_touche Pointeur vers l'indice de la touche sélectionnée.
00313 * @param itemsTouches Tableau des éléments de menu des touches.
00314 */
00315 void mise_a_jour_touches(SDL_Event *event, SDL_Keycode *touche, int *selection_touche, itemMenu
00316 *itemsTouches) {
00317     (*touche) = event->key.keysym.sym;
00318     sprintf(itemsTouches[*selection_touche].texte, " %s ",
00319         SDL_GetKeyName((*touche)));
00319     (*selection_touche) = 0;
00320 }
00321
00322 /**
00323 * \fn void options(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
00324 *programme_lance, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
00325 *plein_ecran, SDL_Rect *rectangle_retour_en_arriere, SDL_Texture **texture_image_retour_en_arriere,
00326 *SDL_Texture **texture_image_hautParleurActif, SDL_Rect *rectangle_demande_sauvegarde, itemMenu
00327 *itemsDemandeSauvegarde, int tailleDemandeSauvegarde, SDL_Texture
00328 **texture_image_hautParleurDesactive, SDL_bool *sonsActifs, SDL_Rect *rectangles_boutons_sons,
00329 option_t *ongletActif, itemMenu *pseudo, modes_t *modeActif, personnage_t *personnageActif, position_t
00330 *positionActive, niveaux *avancee_niveaux, int tailleNiveaux, itemMenu *titre, SDL_Surface **surface,
00331 SDL_Texture **texture_texte, TTF_Font **police, int *selection_touche, SDL_Keycode
00332 *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter,
00333 SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, SDL_Color couleurNoire, itemMenu
00334 *itemsMenu, int tailleMenu, itemMenu *itemsTouches, int tailleTouches, barreDeSon *barre_de_son, int
00335 tailleBarres, itemMenu *itemsBarres, int *largeur, int *hauteur, page_t *page_active, page_t
00336 *page_precedente, int *maintient_clic)
00337 * \brief Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans les options
00338 * @param event Pointeur vers l'événement SDL.
00339 * @param window Pointeur vers la fenêtre SDL.
00340 * @param renderer Pointeur vers le renderer SDL.
00341 * @param programme_lance Pointeur vers le booléen indiquant si le programme est en cours d'exécution.
00342 * @param rectangle_plein_ecran Rectangle pour le mode plein écran.
00343 * @param texture_image_plein_ecran Texture pour le bouton du mode plein écran.
00344 * @param plein_ecran Pointeur vers le booléen indiquant si le jeu est en mode plein écran.
00345 * @param rectangle_retour_en_arriere Rectangle pour le bouton de retour en arrière.
00346 * @param texture_image_retour_en_arriere Texture pour le bouton de retour en arrière.
00347 * @param texture_image_hautParleurActif Texture pour l'icône du haut-parleur actif.
00348 * @param rectangle_demande_sauvegarde Rectangle pour le bouton de demande de sauvegarde.
00349 * @param itemsDemandeSauvegarde Tableau des éléments de menu pour la demande de sauvegarde.
00350 * @param tailleDemandeSauvegarde Taille du tableau des éléments de menu de demande de sauvegarde.
00351 * @param texture_image_hautParleurDesactive Texture pour l'icône du haut-parleur désactivé.
00352 * @param sonsActifs Pointeur vers le booléen indiquant si les sons sont actifs.
00353 * @param rectangles_boutons_sons Tableau des rectangles pour les boutons des sons.
00354 * @param ongletActif Pointeur vers l'onglet actif dans les options.
00355 * @param pseudo Pseudo du joueur.
00356 * @param modeActif Mode de jeu actif.
00357 * @param personnageActif Personnage actif.
00358 * @param positionActive Position active du joueur.
00359 * @param avancee_niveaux Tableau de progression des niveaux.
00360 * @param tailleNiveaux Taille du tableau de progression des niveaux.
00361 * @param titre Élément de menu pour le titre.
00362 * @param surface Surface SDL pour le rendu de texte.
00363 * @param texture_texte Texture SDL pour le rendu de texte.
00364 * @param police TTF pour le rendu de texte.
00365 * @param selection_touche Indice de la touche sélectionnée.
00366 * @param touche_aller_a_droite Touche pour aller à droite.
00367 * @param touche_aller_a_gauche Touche pour aller à gauche.
00368 * @param touche_sauter_monter Touche pour sauter/monter.
00369 * @param touche_descendre Touche pour descendre.
00370 * @param touche_interagir Touche pour interagir.
00371 * @param couleurNoire Couleur noire pour le texte.
00372 * @param itemsMenu Tableau des éléments de menu.
00373 * @param tailleMenu Taille du tableau des éléments de menu.
00374 * @param itemsTouches Tableau des éléments de menu pour les touches.

```

```

00363 * @param tailleTouches Taille du tableau des éléments de menu pour les touches.
00364 * @param barre_de_son Tableau des barres de son.
00365 * @param tailleBarres Taille du tableau des barres de son.
00366 * @param itemsBarres Tableau des éléments de menu pour les barres de son.
00367 * @param largeur Largeur de la fenêtre.
00368 * @param hauteur Hauteur de la fenêtre.
00369 * @param page_active Page active.
00370 * @param page_precedente Page précédente.
00371 * @param maintient_clic Booléen indiquant si le clic est maintenu.
00372 * \see redimensionnement_fenetre
00373 * \see clic_case
00374 * \see clic_plein_ecran
00375 * \see mise_a_jour_barre_de_son
00376 * \see mise_a_jour_touches
00377 * \see mise_a_jour_rendu_options
00378 * \see demande_sauvegarde
00379 * \see sauvegarder_partie
00380 */
00381 void options(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance,
00382             SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
*plein_ecran,
00383             SDL_Rect *rectangle_retour_en_arriere, SDL_Texture **texture_image_retour_en_arriere,
00384             SDL_Texture **texture_image_hautParleurActif, SDL_Rect *rectangle_demande_sauvegarde,
itemMenu *itemsDemandeSauvegarde, int tailleDemandeSauvegarde,
00385             SDL_Texture **texture_image_hautParleurDesactive, SDL_bool *sonsActifs,
00386             SDL_Rect *rectangles_boutons_sons, option_t *ongletActif, itemMenu *pseudo,
00387             modes_t *modeActif, personnage_t *personnageActif, position_t *positionActive,
00388             niveaux *avancee_niveaux, int tailleNiveaux,
00389             itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00390             int *selection_touche, SDL_Keycode *touche_aller_a_droite, SDL_Keycode
*touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter,
00391             SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, SDL_Color couleurNoire,
00392             itemMenu *itemsMenu, int tailleMenu, itemMenu *itemsTouches, int tailleTouches,
00393             barreDeSon *barre_de_son, int tailleBarres, itemMenu *itemsBarres,
00394             int *largeur, int *hauteur, page_t *page_active, page_t *page_precedente, int
*maintient_clic,
00395             time_t temps_debut_partie, int *compteur_mort, int *avancee_succes) {
00396
00397     SDL_Event event_temporaire;
00398     SDL_bool clic_effectue = SDL_FALSE;
00399
00400     while(SDL_PollEvent(event)) {
00401
00402         switch(event->type) {
00403
00404             /* Gestion de l'événement de redimensionnement de la fenêtre */
00405             case SDL_WINDOWEVENT:
00406                 redimensionnement_fenetre((*event), largeur, hauteur);
00407
00408                 break;
00409
00410             /* Changement du volume des barres de sons */
00411             case SDL_MOUSEBUTTONDOWN:
00412
00413                 (*maintient_clic) = 1;
00414
00415                 /* Clic sur les rectangles de changement d'onglet */
00416                 if(clic_case((*event), itemsMenu[0].rectangle))
00417                     (*ongletActif) = ONGLET_SON;
00418                 if(clic_case((*event), itemsMenu[1].rectangle))
00419                     (*ongletActif) = ONGLET_TOUCHES;
00420
00421                 /* Onglet Son */
00422                 /* Désactivation du musiquie */
00423                 if((clic_case((*event), rectangles_boutons_sons[0])) && (sonsActifs[0] ==
SDL_TRUE)) {
00424
00425                     sonsActifs[0] = SDL_FALSE;
00426                     barre_de_son[0].volume_precedent = barre_de_son[0].volume;
00427                     barre_de_son[0].volume = 0;
00428
00429                     /* Réactivation du musiquie */
00430                     else if((clic_case((*event), rectangles_boutons_sons[0])) &&
(barre_de_son[0].volume_precedent != 0)) {
00431                         sonsActifs[0] = SDL_TRUE;
00432                         barre_de_son[0].volume = barre_de_son[0].volume_precedent;
00433
00434                     /* Désactivation du effet sonore */
00435                     if((clic_case((*event), rectangles_boutons_sons[1])) && (sonsActifs[1] ==
SDL_TRUE)) {
00436
00437                         sonsActifs[1] = SDL_FALSE;
00438                         barre_de_son[1].volume_precedent = barre_de_son[1].volume;
00439                         barre_de_son[1].volume = 0;
00440
00441                         /* Réactivation du effet sonore */
00442                         else if(clic_case((*event), rectangles_boutons_sons[1])) &&
(barre_de_son[1].volume_precedent != 0)) {

```

```

00441         sonsActifs[1] = SDL_TRUE;
00442         barre_de_son[1].volume = barre_de_son[1].volume_precedent;
00443     }
00444
00445     /* Onglet Touches */
00446     if(clic_case((*event), itemsTouchés[1].rectangle))
00447         (*selection_touche) = 1;
00448     else if(clic_case((*event), itemsTouchés[3].rectangle))
00449         (*selection_touche) = 3;
00450     else if(clic_case((*event), itemsTouchés[5].rectangle))
00451         (*selection_touche) = 5;
00452     else if(clic_case((*event), itemsTouchés[7].rectangle))
00453         (*selection_touche) = 7;
00454     else if(clic_case((*event), itemsTouchés[9].rectangle))
00455         (*selection_touche) = 9;
00456     else
00457         (*selection_touche) = 0;
00458
00459     /* Options plein écran et retour en arrière */
00460
00461     if(clic_plein_ecran((*event), rectangle_plein_ecran, plein_ecran, window))
00462         redimensionnement_fenetre((*event), largeur, hauteur);
00463
00464     if(clic_case((*event), (*rectangle_retour_en_arriere)))
00465         (*page_active) = (*page_precedente);
00466
00467     break;
00468
00469     /* Changement du volume */
00470     case SDL_MOUSEBUTTONDOWN :
00471
00472         (*maintient_clic) = 0;
00473
00474         if(clic_case((*event), barre_de_son[0].barre)){
00475             mise_a_jour_barre_de_son(event, &(barre_de_son[0]), &(sonsActifs[0]));
00476         }
00477
00478         else if(clic_case((*event), barre_de_son[1].barre)){
00479             mise_a_jour_barre_de_son(event, &(barre_de_son[1]), &(sonsActifs[1]));
00480         }
00481
00482         break;
00483
00484     /* Changement des touches */
00485     case SDL_KEYDOWN :
00486         if((event->key.keysym.sym == SDLK_ESCAPE) && (*selection_touche)) {
00487             (*touche_aller_a_droite) = SDLK_RIGHT;
00488             (*touche_aller_a_gauche) = SDLK_LEFT;
00489             (*touche_sauter_monter) = SDLK_UP;
00490             (*touche_descendre) = SDLK_DOWN;
00491             (*touche_interagir) = SDLK_SPACE;
00492             sprintf(itemsTouchés[1].texte, "          %s          ",
00493                 SDL_GetKeyName((*touche_aller_a_droite)));
00494             sprintf(itemsTouchés[3].texte, "          %s          ",
00495                 SDL_GetKeyName((*touche_aller_a_gauche)));
00496             sprintf(itemsTouchés[5].texte, "          %s          ",
00497                 SDL_GetKeyName((*touche_sauter_monter)));
00498             sprintf(itemsTouchés[7].texte, "          %s          ",
00499                 SDL_GetKeyName((*touche_descendre)));
00500             sprintf(itemsTouchés[9].texte, "          %s          ",
00501                 SDL_GetKeyName((*touche_interagir)));
00502             (*selection_touche) = 0;
00503         }
00504
00505         else if((*selection_touche) == 1) &&
00506             (event->key.keysym.sym != (*touche_aller_a_gauche)) &&
00507             (event->key.keysym.sym != (*touche_sauter_monter)) &&
00508             (event->key.keysym.sym != (*touche_descendre)) &&
00509             (event->key.keysym.sym != (*touche_interagir)))
00510             mise_a_jour_touches(event, touche_aller_a_droite, selection_touche,
00511                 itemsTouchés);
00512
00513         else if((*selection_touche) == 3) &&
00514             (event->key.keysym.sym != (*touche_aller_a_droite)) &&
00515             (event->key.keysym.sym != (*touche_sauter_monter)) &&
00516             (event->key.keysym.sym != (*touche_descendre)) &&
00517             (event->key.keysym.sym != (*touche_interagir)))
00518             mise_a_jour_touches(event, touche_aller_a_gauche, selection_touche,
00519                 itemsTouchés);
00520
00521         else if((*selection_touche) == 5) &&
00522             (event->key.keysym.sym != (*touche_aller_a_droite)) &&
00523             (event->key.keysym.sym != (*touche_aller_a_gauche)) &&
00524             (event->key.keysym.sym != (*touche_descendre)) &&
00525             (event->key.keysym.sym != (*touche_interagir)))
00526             mise_a_jour_touches(event, touche_sauter_monter, selection_touche,
00527                 itemsTouchés);

```

```

00520
00521         else if ((*selection_touche) == 7) &&
00522             (event->key.keysym.sym != (*touche_aller_a_droite)) &&
00523             (event->key.keysym.sym != (*touche_aller_a_gauche)) &&
00524             (event->key.keysym.sym != (*touche_sauter_monter)) &&
00525             (event->key.keysym.sym != (*touche_interagir))
00526             mise_a_jour_touches(event, touche_descendre, selection_touche, itemsTouches);
00527
00528         else if ((*selection_touche) == 9) &&
00529             (event->key.keysym.sym != (*touche_aller_a_droite)) &&
00530             (event->key.keysym.sym != (*touche_aller_a_gauche)) &&
00531             (event->key.keysym.sym != (*touche_sauter_monter)) &&
00532             (event->key.keysym.sym != (*touche_descendre))
00533             mise_a_jour_touches(event, touche_interagir, selection_touche, itemsTouches);
00534
00535         break;
00536
00537     case SDL_MOUSEMOTION :
00538
00539         if ((*maintient_clic)) {
00540
00541             /* Si le mouvement de la souris est dans la zone de la barre de son, ajuste le
00542             volume en fonction de la position du curseur */
00542             if ((event->motion.y >= barre_de_son[0].barre.y - barre_de_son[0].barre.h / 2)
00543             &&
00544                 (event->motion.y <= barre_de_son[0].barre.y + barre_de_son[0].barre.h) &&
00545                 (event->motion.x >= barre_de_son[0].barre.x - barre_de_son[0].barre.w / 2)
00546                 &&
00547                 (event->motion.x <= barre_de_son[0].barre.x + barre_de_son[0].barre.w)) {
00548                 barre_de_son[0].volume = (event->motion.x - ((*largeur) / 2 -
00549                 barre_de_son[0].barre.w / 2)) * 100 / barre_de_son[0].barre.w;
00550                 mise_a_jour_barre_de_son(event, &(barre_de_son[0]), &(sonsActifs[0]));
00551             }
00552             if ((event->motion.y >= barre_de_son[1].barre.y - barre_de_son[1].barre.h / 2)
00553             &&
00554                 (event->motion.y <= barre_de_son[1].barre.y + barre_de_son[1].barre.h) &&
00555                 (event->motion.x >= barre_de_son[1].barre.x - barre_de_son[1].barre.w / 2)
00556                 &&
00557                 (event->motion.x <= barre_de_son[1].barre.x + barre_de_son[1].barre.w)) {
00558                 barre_de_son[1].volume = (event->motion.x - ((*largeur) / 2 -
00559                 barre_de_son[1].barre.w / 2)) * 100 / barre_de_son[1].barre.w;
00560                 mise_a_jour_barre_de_son(event, &(barre_de_son[1]), &(sonsActifs[1]));
00561             }
00562         }
00563         break;
00564
00565     /* Quitter le programme */
00566     case SDL_QUIT :
00567
00568         if ((*page_precedente) == CARTE) {
00569             SDL_SetWindowResizable((*window), SDL_FALSE);
00570             demande_sauvegarde(renderer, rectangle_demande_sauvegarde,
00571             surface, texture_texte, police, couleurNoire,
00572             itemsDemandeSauvegarde, tailleDemandeSauvegarde,
00573             (*largeur), (*hauteur));
00574
00575             while (!clic_effectue) {
00576                 while (SDL_PollEvent(&event_temporaire)) {
00577                     if(event_temporaire.type == SDL_MOUSEBUTTONDOWN) {
00578                         /* Cas où la personne veut sauvegarder */
00579                         if(clic_case(event_temporaire,
00580                         itemsDemandeSauvegarde[1].rectangle)) {
00581                             sauvegarder_partie(touche_aller_a_droite,
00582                             touche_aller_a_gauche, touche_sauter_monter,
00583                             touche_descendre, touche_interagir,
00584                             barre_de_son, pseudo,
00585                             (*modeActif), (*personnageActif),
00586                             (*positionActive),
00587                             avancee_niveaux, tailleNiveaux,
00588                             temps_debut_partie, (*compteur_mort), avancee_succes);
00589                             (*programme_lance) = SDL_FALSE;
00590                             clic_effectue = SDL_TRUE;
00591                         }
00592                         /* Cas où la personne ne veut pas sauvegarder */
00593                         else if(clic_case(event_temporaire,

```

```

        itemsDemandeSauvegarde[2].rectangle)) {
00594                                     (*programme_lance) = SDL_FALSE;
00595                                     clic_effectue = SDL_TRUE;
00596                                     }
00597
00598                                     else if(!clic_case(event_temporaire,
        (*rectangle_demande_sauvegarde)))
00599                                     clic_effectue = SDL_TRUE;
00600                                     }
00601                                     }
00602                                     }
00603
00604                                     SDL_SetWindowResizable((*window), SDL_TRUE);
00605                                     }
00606
00607                                     else
00608                                     (*programme_lance) = SDL_FALSE;
00609
00610                                     break;
00611
00612                                     default:
00613                                     break;
00614                                     }
00615     }
00616
00617     /* Mise à jour du rendu */
00618     mise_a_jour_rendu_options(renderer, rectangle_plein_ecran, texture_image_plein_ecran,
00619                               rectangle_retour_en_arriere, texture_image_retour_en_arriere,
00620                               texture_image_hautParleurActif,
00621                               texture_image_hautParleurDesactive, sonsActifs,
00622                               rectangles_boutons_sons, (*ongletActif),
00623                               titre, surface, texture_texte, police,
00624                               couleurNoire, (*selection_touche),
00625                               itemsMenu, tailleMenu, itemsTouches, tailleTouches,
00626                               barre_de_son, tailleBarres, itemsBarres,
00627                               (*largeur), (*hauteur));
00628 }

```

## 4.23 Référence du fichier fichiers\_c/metatravers.c

Fichier qui réunit les différents modules pour le bon fonctionnement du programme.

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
#include "../fichiers_h/fonctions_generales.h"
#include "../fichiers_h/fonctions_menu_principal.h"
#include "../fichiers_h/fonctions_options.h"
#include "../fichiers_h/fonctions_nouvelle_partie.h"
#include "../fichiers_h/fonctions_introduction.h"
#include "../fichiers_h/fonctions_carte.h"
#include "../fichiers_h/fonctions_niveau_1.h"
#include "../fichiers_h/fonctions_niveau_2.h"
#include "../fichiers_h/fonctions_niveau_3.h"
#include "../fichiers_h/fonctions_arrivee_niveaux_2_3.h"
#include "../fichiers_h/fonctions_niveau_4.h"

```

Graphe des dépendances par inclusion de metatravers.c:



## Fonctions

- `int main ()`  
*Fonction principale du programme.*

### 4.23.1 Description détaillée

Fichier qui réunit les différents modules pour le bon fonctionnement du programme.

Définition dans le fichier [metatravers.c](#).

### 4.23.2 Documentation des fonctions

#### 4.23.2.1 main()

```
int main (  
    void )
```

Fonction principale du programme.

#### Renvoie

déroulement du programme (0 si bien passé sinon code erreur)

#### Voir également

[creer\\_fenetre\\_rendu](#)  
[initialisation\\_objets](#)  
[verification\\_sauvegarde](#)  
[erreur](#)  
[initialisation\\_objets\\_menu\\_principal](#)  
[initialisation\\_objets\\_options](#)  
[initialisation\\_objets\\_nouvelle\\_partie](#)  
[initialisation\\_objets\\_carte](#)  
[initialisation\\_objets\\_niveau\\_1](#)  
[initialisation\\_objets\\_niveau\\_2](#)  
[initialisation\\_objets\\_niveau\\_3](#)  
[initialisation\\_objets\\_niveau\\_4](#)  
[chargement\\_image](#)  
[menu\\_principal](#)  
[options](#)  
[nouvelle\\_partie](#)  
[introduction](#)  
[carte](#)  
[chargement\\_niveau\\_1](#)  
[salon\\_arrivee\\_niveaux\\_2\\_3](#)  
[etage\\_1](#)  
[arrivee\\_niveaux\\_2\\_3](#)  
[niveau\\_1](#)  
[niveau\\_4](#)  
[detruire\\_objets](#)  
[detruire\\_fenetre\\_rendu](#)

Définition à la ligne 54 du fichier [metatravers.c](#).



## 4.24 metatravers.c

[Aller à la documentation de ce fichier.](#)

```

00001 /**
00002  * \file metatravers.c
00003  * \brief Fichier qui réunit les différents modules pour le bon fonctionnement du programme
00004  */
00005 #include <stdio.h>
00006 #include <stdlib.h>
00007 #include <time.h>
00008 #include <SDL2/SDL.h>
00009 #include <SDL2/SDL_ttf.h>
00010 #include <SDL2/SDL_image.h>
00011 #include <SDL2/SDL_mixer.h>
00012 #include "../fichiers_h/fonctions_generales.h"
00013 #include "../fichiers_h/fonctions_menu_principal.h"
00014 #include "../fichiers_h/fonctions_options.h"
00015 #include "../fichiers_h/fonctions_nouvelle_partie.h"
00016 #include "../fichiers_h/fonctions_introduction.h"
00017 #include "../fichiers_h/fonctions_carte.h"
00018 #include "../fichiers_h/fonctions_niveau_1.h"
00019 #include "../fichiers_h/fonctions_niveau_2.h"
00020 #include "../fichiers_h/fonctions_niveau_3.h"
00021 #include "../fichiers_h/fonctions_arrivee_niveaux_2_3.h"
00022 #include "../fichiers_h/fonctions_niveau_4.h"
00023 /**
00024  * \fn int main(void)
00025  * \brief Fonction principale du programme
00026  * \return déroulement du programme (0 si bien passé sinon code erreur)
00027  * \see creer_fenetre_rendu
00028  * \see initialisation_objets
00029  * \see verification_sauvegarde
00030  * \see erreur
00031  * \see initialisation_objets_menu_principal
00032  * \see initialisation_objets_options
00033  * \see initialisation_objets_nouvelle_partie
00034  * \see initialisation_objets_carte
00035  * \see initialisation_objets_niveau_1
00036  * \see initialisation_objets_niveau_2
00037  * \see initialisation_objets_niveau_3
00038  * \see initialisation_objets_niveau_4
00039  * \see chargement_image
00040  * \see menu_principal
00041  * \see options
00042  * \see nouvelle_partie
00043  * \see introduction
00044  * \see carte
00045  * \see chargement_niveau_1
00046  * \see salon_arrivee_niveaux_2_3
00047  * \see etage_1
00048  * \see arrivee_niveaux_2_3
00049  * \see niveau_1
00050  * \see niveau_4
00051  * \see detruire_objets
00052  * \see detruire_fenetre_rendu
00053  */
00054 int main() {
00055
00056     /* Initialisation de la largeur de la fenêtre */
00057     int largeur = 960;
00058     /* Initialisation de la hauteur de la fenêtre */
00059     int hauteur = 540;
00060
00061     int i, x, y;
00062
00063     /* Création des pointeurs sur la fenêtre et sur le rendu */
00064     SDL_Window *window = NULL;
00065     SDL_Renderer *renderer = NULL;
00066
00067     /* Initialisation de la surface */
00068     SDL_Surface *surface = NULL;
00069
00070     /* Initialisation de la texture */
00071     SDL_Texture *texture = NULL;
00072
00073     /* Création du pointeur sur la texture du texte */
00074     SDL_Texture *texture_texte = NULL;
00075
00076     /* Création du pointeur sur la texture de l'image du plein écran et du rectangle où se trouvera
l'image */
00077     SDL_Texture *texture_image_plein_ecran = NULL;
00078     SDL_Rect rectangle_plein_ecran;
00079
00080     /* Création du pointeur sur la texture de l'image du retour en arrière et du rectangle où se
trouvera l'image */

```

```

00081     SDL_Texture *texture_image_retour_en_arriere = NULL;
00082     SDL_Rect rectangle_retour_en_arriere;
00083
00084     /* Création du pointeur sur la texture de l'image des options et du rectangle où se trouvera
l'image */
00085     SDL_Texture *texture_image_options = NULL;
00086     SDL_Rect rectangle_options;
00087
00088     /* Création du pointeur sur la texture de l'image du retour au menu principal et du où se trouvera
l'image */
00089     SDL_Texture *texture_image_retour_menu = NULL;
00090     SDL_Rect rectangle_retour_menu;
00091
00092     /* Création du pointeur sur la texture de l'image du plein écran et du rectangle où se trouvera
l'image */
00093     SDL_Texture *texture_image_passer = NULL;
00094     SDL_Rect rectangle_passer;
00095
00096     /* Création du pointeur sur la texture de l'image de la croix et du rectangle où se trouvera
l'image */
00097     SDL_Texture *texture_image_croix = NULL;
00098     SDL_Rect rectangle_croix;
00099
00100     /* Pointeur sur la police */
00101     TTF_Font *police = NULL;
00102
00103     /* Création du pointeur sur la texture de l'image du menu */
00104     SDL_Texture *texture_image_menu = NULL;
00105
00106     /* Création du pointeur sur la texture de l'image de la carte */
00107     SDL_Texture *texture_image_carte = NULL;
00108
00109     /* Création du pointeur sur la texture de l'image haut parleur actif */
00110     SDL_Texture *texture_image_hautParleurActif = NULL;
00111
00112     /* Création du pointeur sur la texture de l'image haut parleur désactivé */
00113     SDL_Texture *texture_image_hautParleurDesactive = NULL;
00114
00115     /* Touches pour les déplacements du personnage */
00116     SDL_Keycode touche_aller_a_droite = SDLK_RIGHT;
00117     SDL_Keycode touche_aller_a_gauche = SDLK_LEFT;
00118     SDL_Keycode touche_sauter_monter = SDLK_UP;
00119     SDL_Keycode touche_descendre = SDLK_DOWN;
00120     SDL_Keycode touche_interagir = SDLK_SPACE;
00121
00122     /* Création d'un rectangle pour le pseudo */
00123     SDL_Rect rectangle_pseudo;
00124
00125     /* Création du pointeur sur la texture de l'image du premier personnage et de son rectangle */
00126     SDL_Texture *texture_image_perso_1 = NULL;
00127     SDL_Rect rectangle_perso_1;
00128
00129     /* Création du pointeur sur la texture de l'image du deuxième personnage et de son rectangle */
00130     SDL_Texture *texture_image_perso_2 = NULL;
00131     SDL_Rect rectangle_perso_2;
00132
00133     /* Création des pointeurs sur la texture des différentes images pour le premier personnage */
00134     SDL_Texture *texture_image_perso_1_bas_1 = NULL;
00135     SDL_Texture *texture_image_perso_1_bas_2 = NULL;
00136     SDL_Texture *texture_image_perso_1_haut_1 = NULL;
00137     SDL_Texture *texture_image_perso_1_haut_2 = NULL;
00138     SDL_Texture *texture_image_perso_1_bas_gauche_1 = NULL;
00139     SDL_Texture *texture_image_perso_1_bas_gauche_2 = NULL;
00140     SDL_Texture *texture_image_perso_1_haut = NULL;
00141     SDL_Texture *texture_image_perso_1_droite = NULL;
00142     SDL_Texture *texture_image_perso_1_gauche = NULL;
00143     SDL_Texture *texture_image_perso_1_pose = NULL;
00144     SDL_Texture *texture_image_perso_1_gagnant = NULL;
00145
00146     /* Création des pointeurs sur la texture des différentes images pour le deuxième personnage */
00147     SDL_Texture *texture_image_perso_2_bas_1 = NULL;
00148     SDL_Texture *texture_image_perso_2_bas_2 = NULL;
00149     SDL_Texture *texture_image_perso_2_haut_1 = NULL;
00150     SDL_Texture *texture_image_perso_2_haut_2 = NULL;
00151     SDL_Texture *texture_image_perso_2_bas_gauche_1 = NULL;
00152     SDL_Texture *texture_image_perso_2_bas_gauche_2 = NULL;
00153     SDL_Texture *texture_image_perso_2_haut = NULL;
00154     SDL_Texture *texture_image_perso_2_droite = NULL;
00155     SDL_Texture *texture_image_perso_2_gauche = NULL;
00156     SDL_Texture *texture_image_perso_2_pose = NULL;
00157     SDL_Texture *texture_image_perso_2_gagnant = NULL;
00158
00159     /* Création des pointeur sur la texture des différentes images pour les monstres */
00160     SDL_Texture *texture_image_monstre_terrestre = NULL;
00161     SDL_Texture *texture_image_monstre_volant = NULL;
00162
00163     /* Création des pointeurs sur la texture des différentes images du niveau 1 */

```

```

00164     SDL_Texture *texture_image_fond_niveau_1 = NULL;
00165     SDL_Texture *texture_image_sol_surface_niveau_1 = NULL;
00166     SDL_Texture *texture_image_sol_profondeur_niveau_1 = NULL;
00167     SDL_Texture *texture_image_nuage_1 = NULL;
00168     SDL_Texture *texture_image_nuage_2 = NULL;
00169
00170     /* Création des pointeurs sur la texture des différentes images du salon en arrivant dans le
niveau 2 */
00171     SDL_Texture *texture_image_fond_niveau_2 = NULL;
00172     SDL_Texture *texture_image_dossier_niveau_2 = NULL;
00173     SDL_Texture *texture_image_sol_niveau_2 = NULL;
00174
00175     /* Création des pointeurs sur la texture des différentes images des pipes pour le niveau 2 */
00176     SDL_Texture *texture_image_mur_mini_jeu = NULL;
00177     SDL_Texture *texture_image_pipe_vertical = NULL;
00178     SDL_Texture *texture_image_pipe_horizontal = NULL;
00179     SDL_Texture *texture_image_pipe_haut_droit = NULL;
00180     SDL_Texture *texture_image_pipe_bas_droit = NULL;
00181     SDL_Texture *texture_image_pipe_bas_gauche = NULL;
00182     SDL_Texture *texture_image_pipe_haut_gauche = NULL;
00183     SDL_Texture *texture_image_pipe_courant = NULL;
00184     SDL_Texture *texture_image_mur_termine = NULL;
00185
00186     /* Création des pointeurs sur la texture des différentes images du salon en arrivant dans le
niveau 3 */
00187     SDL_Texture *texture_image_fond_niveau_3 = NULL;
00188     SDL_Texture *texture_image_dossier_niveau_3 = NULL;
00189     SDL_Texture *texture_image_sol_niveau_3 = NULL;
00190     SDL_Texture *barre_windows_1 = NULL;
00191     SDL_Texture *barre_windows_2 = NULL;
00192     SDL_Texture *barre_windows_3 = NULL;
00193     SDL_Texture *barre_windows_4 = NULL;
00194
00195     /* Création des pointeurs sur la texture de l'image du puzzle */
00196     SDL_Texture* texture_image_puzzle = NULL;
00197
00198     /* Création des pointeurs sur la texture des différentes images du labyrinthe */
00199     SDL_Texture *texture_image_sol_labyrinthe = NULL;
00200     SDL_Texture *texture_image_bordure_labyrinthe = NULL;
00201     SDL_Texture *texture_image_fin_labyrinthe = NULL;
00202
00203     /* Création des pointeurs sur la texture des différentes images pour les étages */
00204     SDL_Texture *texture_image_mur = NULL;
00205     SDL_Texture *texture_image_fond_niveau_4 = NULL;
00206     SDL_Texture *texture_image_bordure_niveau_4 = NULL;
00207     SDL_Texture *texture_image_porte = NULL;
00208     SDL_Texture *texture_image_pique = NULL;
00209
00210     /* Création des pointeurs sur la texture des différentes images pour la fin des niveaux */
00211     SDL_Texture *texture_image_fin_premiers_niveaux = NULL;
00212     SDL_Texture *texture_image_fin_dernier_niveau = NULL;
00213
00214     /* Création des pointeurs sur la texture des différentes images pour les succès et pour leur
rectangle */
00215     SDL_Texture *textures_images_succes[11] = {NULL};
00216     SDL_Rect rectangle_succes;
00217
00218     /* Création du rectangle pour le texte de l'introduction */
00219     SDL_Rect rectangle_texte_introduction;
00220
00221     /* Création du rectangle pour la demande de sauvegarde */
00222     SDL_Rect rectangle_demande;
00223
00224     /* Variable de couleur noire */
00225     SDL_Color couleurNoire = {0, 0, 0, 255};
00226
00227     /* Variable de la couleur du titre principal */
00228     SDL_Color couleurTitre = {200, 200, 200, 255};
00229
00230     /* Variable de couleur blanche */
00231     SDL_Color couleurBlanche = {255, 255, 255, 255};
00232
00233     /* Lancement de VIDEO SDL */
00234     if(SDL_Init(SDL_INIT_VIDEO) != 0)
00235         erreur("Initialisation VIDEO SDL");
00236
00237     /* Lancement de TTF */
00238     if(TTF_Init() == -1)
00239         erreur("Initialisation TTF");
00240
00241     /* Lancement de AUDIO SDL */
00242     if(SDL_Init(SDL_INIT_AUDIO) == -1)
00243         erreur("Initialisation AUDIO SDL");
00244
00245     /* Lancement du mixer */
00246     if(Mix_OpenAudio(96000, MIX_DEFAULT_FORMAT, MIX_DEFAULT_CHANNELS, 1024) < 0)
00247         erreur("Initialisation AUDIO SDL");

```

```

00248
00249     /* Initialisation de la musique */
00250     Mix_Music *musique = NULL;
00251
00252     creer_fenetre_rendu(&window, &renderer, largeur, hauteur);
00253
00254     /*----- Initialisation des objets -----*/
00255
00256     /* Objets globaux */
00257
00258     int tailleDemande = 3;
00259
00260     itemMenu itemsDemandeSauvegarde[tailleDemande];
00261
00262     itemMenu itemsDemandeQuitter[tailleDemande];
00263
00264     int tailleNiveaux = 4;
00265
00266     niveaux avancee_niveaux[tailleNiveaux];
00267
00268     time_t temps_debut_partie;
00269
00270     int compteur_mort;
00271
00272     int avancee_succes[10];
00273
00274     int avancee_succes_intermediaires[10];
00275
00276     int collectibles_intermediaires[3];
00277
00278     SDL_bool programme_lance = SDL_TRUE;
00279
00280     SDL_bool plein_ecran = SDL_FALSE;
00281
00282     initialisation_objets(&renderer, &surface, &texture_image_plein_ecran,
00283                          &texture_image_retour_en_arriere, &texture_image_options,
00284                          &texture_image_passer, itemsDemandeSauvegarde, itemsDemandeQuitter,
00285                          &texture_image_fin_premiers_niveaux, &texture_image_monstre_terrestre,
00286                          &texture_image_monstre_volant, &texture_image_perso_1_gagnant,
00287                          &texture_image_perso_2_gagnant,
00288                          avancee_niveaux, &police, &texture_image_croix);
00289
00290     /* Objets du menu principal */
00291
00292     int selection_menu = 0;
00293
00294     /* Initialisation des objets intermédiaires pour la sauvegarde */
00295     int niveau_fini[4] = {0};
00296     int collectibles[12] = {0};
00297     position_t position_intermediaire;
00298
00299     int code_de_triche[3] = {0};
00300
00301     itemMenu titre_menu_principal;
00302
00303     int tailleMenuPrincipal;
00304
00305     if(!verification_sauvegarde()) {
00306         tailleMenuPrincipal = 2;
00307     }
00308     else
00309         tailleMenuPrincipal = 3;
00310
00311     /* Allocation dynamique de mémoire pour le tableau itemsMenuPrincipal en fonction de
    tailleMenuPrincipal */
00312     itemMenu *itemsMenuPrincipal = malloc(tailleMenuPrincipal * sizeof(itemMenu));
00313     if (itemsMenuPrincipal == NULL) {
00314         erreur("Allocation de la mémoire");
00315     }
00316
00317     initialisation_objets_menu_principal(&renderer, &surface, &texture_image_menu,
00318                                         &titre_menu_principal, itemsMenuPrincipal,
00319                                         tailleMenuPrincipal);
00320
00321     /* Objets des options */
00322
00323     itemMenu titre_options;
00324
00325     int tailleBarres = 2;
00326
00327     barreDeSon barre_de_son[tailleBarres];
00328
00329     int tailleMenuOptions = 2;
00330
00331     itemMenu itemsMenuOptions[tailleMenuOptions];
00332
00333

```

```

00332     int tailleTouches = 10;
00333
00334     itemMenu itemsTouches[tailleTouches];
00335
00336     itemMenu itemsBarres[tailleBarres];
00337
00338     int maintient_clic = 0;
00339
00340     initialisation_objets_options(&renderer, &surface, &texture_image_hautParleurActif,
00341                                   &texture_image_hautParleurDesactive,
00342                                   &titre_options, itemsMenuOptions, itemsTouches, itemsBarres);
00343
00344     /* Variable pour suivre l'onglet actif */
00345     option_t ongletActif = ONGLET_SON;
00346
00347     /* Variable pour suivre l'état du son (activé/désactivé) */
00348     SDL_bool sonsActifs[tailleBarres];
00349
00350     /* Rectangles des boutons de son */
00351     SDL_Rect rectangles_boutons_sons[tailleBarres];
00352
00353     /* Variable pour suivre l'état de la sélection de touche (activé/désactivé) */
00354     int selection_touche = 0;
00355
00356     /* Objets du menu nouvelle partie */
00357
00358     int pseudo_valide;
00359
00360     itemMenu pseudo;
00361     pseudo.texte[0] = '\0';
00362
00363     int tailleTitres = 3;
00364
00365     itemMenu titres[tailleTitres];
00366
00367     int tailleMenuNouvellePartie = 2;
00368
00369     itemMenu itemsMenuNouvellePartie[tailleMenuNouvellePartie];
00370
00371     itemMenu valider;
00372
00373     char pseudo_temporaire[11];
00374     pseudo_temporaire[0] = '\0';
00375
00376     personnage_t personnage_temporaire;
00377
00378     modes_t mode_temporaire;
00379
00380     initialisation_objets_nouvelle_partie(&renderer, &surface, &texture_image_perso_1,
00381                                           &texture_image_perso_2,
00382                                           titres, itemsMenuNouvellePartie, &valider);
00383
00384     /* Variable pour suivre la saisie */
00385     int modeSaisie = 0;
00386
00387     /* Variable pour suivre le personnage actif */
00388     personnage_t personnageActif = PERSONNAGE_1;
00389
00390     /* Variable pour suivre le mode actif */
00391     modes_t modeActif = MODE_NORMAL;
00392
00393     /* Objets de la carte */
00394
00395     int touche_presse = 0;
00396
00397     itemMenu itemsNiveaux[tailleNiveaux];
00398
00399     position_t positionActive = NIVEAU1;
00400
00401     direction_t direction = BAS;
00402
00403     itemMenu itemsSucces[12];
00404
00405     initialisation_objets_carte(&renderer, &surface, &texture_image_carte,
00406                                &texture_image_perso_1_bas_1, &texture_image_perso_1_bas_2,
00407                                &texture_image_perso_1_haut_1, &texture_image_perso_1_haut_2,
00408                                &texture_image_perso_1_bas_gauche_1,
00409                                &texture_image_perso_1_bas_gauche_2,
00410                                &texture_image_perso_1_haut, &texture_image_perso_1_droite,
00411                                &texture_image_perso_1_gauche, &texture_image_perso_1_pose,
00412                                &texture_image_perso_2_bas_1, &texture_image_perso_2_bas_2,
00413                                &texture_image_perso_2_haut_1, &texture_image_perso_2_haut_2,
00414                                &texture_image_perso_2_bas_gauche_1,
00415                                &texture_image_perso_2_bas_gauche_2,
00416                                &texture_image_perso_2_haut, &texture_image_perso_2_droite,
00417                                &texture_image_perso_2_gauche, &texture_image_perso_2_pose,
00418                                itemsNiveaux, &texture_image_retour_menu,

```

```

00417             itemsSucces, textures_images_succes);
00418
00419     /* Objets des niveaux */
00420
00421     /* Initialisation de la largeur de chaque case */
00422     int largeur_tile;
00423     /* Initialisation de la hauteur de chaque case */
00424     int hauteur_tile;
00425
00426     /* Initialisation de l'étage avec la méthode du tile mapping */
00427
00428     int tile_map[18][32];
00429
00430     SDL_Rect rectangle_tile;
00431
00432     /* Positions et état du joueur */
00433
00434     int position_x;
00435     int position_y;
00436
00437     int position_x_initiale;
00438     int position_y_initiale;
00439
00440     int avancer;
00441     int reculer;
00442     int sauter;
00443     int position_avant_saut;
00444     int saut;
00445     int tombe;
00446
00447     int mini_jeu;
00448
00449     time_t timestamp = time(NULL);
00450     int mouvement_monstre;
00451
00452     int mode_difficile;
00453
00454     int mini_jeu_1_termine;
00455     int mini_jeu_2_termine;
00456
00457     itemMenu itemsExplications[2];
00458
00459     /* Objets du niveau 1 */
00460
00461     initialisation_objets_niveau_1(&renderer, &surface,
00462                                     &texture_image_sol_surface_niveau_1,
00463                                     &texture_image_fond_niveau_1, &texture_image_nuage_1,
00464                                     &texture_image_nuage_2);
00465
00466     int tile_map_niveau_1[18][110];
00467
00468     int decalage;
00469     int secret_1;
00470     int secret_2;
00471
00472     /* Objets du niveau 2 */
00473
00474     initialisation_objets_niveau_2(&renderer, &surface,
00475                                     &texture_image_fond_niveau_2, &texture_image_dossier_niveau_2,
00476                                     &texture_image_sol_niveau_2, &texture_image_mur_mini_jeu,
00477                                     &texture_image_pipe_vertical, &texture_image_pipe_horizontal,
00478                                     &texture_image_pipe_haut_droit, &texture_image_pipe_bas_droit,
00479                                     &texture_image_pipe_bas_gauche, &texture_image_pipe_haut_gauche,
00480                                     &texture_image_pipe_courant,
00481                                     &texture_image_mur_termine);
00482
00483     int tile_map_mini_jeu_niveau_2[19][27];
00484
00485     int mini_jeu_termine;
00486
00487     /* Objets du niveau 3 */
00488
00489     SDL_Rect rectangle_piece[45];
00490
00491     /* Initialise toutes les pièces comme non verrouillées */
00492     int piece_bloquee[45];
00493
00494     SDL_Rect rectangle_emplacement_piece[45];
00495
00496     int piece_selectionnee;
00497
00498     /* Décalage en X/Y entre le coin supérieur gauche de la pièce et la position du curseur de la
00499     souris */
00500     int decalage_x;
00501     int decalage_y;

```

```

00501     int valide;
00502
00503     initialisation_objets_niveau_3(&renderer, &surface,
00504                                     &texture_image_fond_niveau_3, &texture_image_dossier_niveau_3,
00505                                     &texture_image_sol_niveau_3, &barre_windows_1, &barre_windows_2,
&barre_windows_3,
00506                                     &barre_windows_4,
00507                                     &texture_image_puzzle, &texture_image_sol_labyrinthe,
00508                                     &texture_image_bordure_labyrinthe, &texture_image_fin_labyrinthe);
00509
00510     int tile_map_mini_jeu_niveau_3[24][32];
00511
00512     int descendre;
00513     int interagir;
00514
00515     int bloc_x;
00516     int bloc_y;
00517
00518     /* Objets du niveau 4 */
00519
00520     initialisation_objets_niveau_4(&renderer, &surface,
00521                                     &texture_image_fond_niveau_4, &texture_image_bordure_niveau_4,
00522                                     &texture_image_porte, &texture_image_pique,
00523                                     &texture_image_fin_dernier_niveau);
00524
00525     int numero_etage = 1;
00526
00527     /* Chargement de la sauvegarde s'il y en a une */
00528     if(verification_sauvegarde()) {
00529
00530         FILE *fichier_sauvegarde;
00531
00532         /* Ouverture du fichier en mode lecture */
00533         fichier_sauvegarde = fopen("./sauvegardes/sauvegarde.txt", "r");
00534
00535         fscanf(fichier_sauvegarde, "%f\n", &(barre_de_son[0].volume));
00536         fscanf(fichier_sauvegarde, "%f\n", &(barre_de_son[1].volume));
00537
00538         for(i = 1; i < tailleTouches; i+=2)
00539             fscanf(fichier_sauvegarde, "%s\n", itemsTouches[i].texte);
00540
00541         /* Assignation des touches sauvegardées au touches de contrôle */
00542         touche_aller_a_droite = SDL_GetKeyFromName(itemsTouches[1].texte);
00543         touche_aller_a_gauche = SDL_GetKeyFromName(itemsTouches[3].texte);
00544         touche_sauter_monter = SDL_GetKeyFromName(itemsTouches[5].texte);
00545         touche_descendre = SDL_GetKeyFromName(itemsTouches[7].texte);
00546         touche_interagir = SDL_GetKeyFromName(itemsTouches[9].texte);
00547
00548         sprintf(itemsTouches[1].texte, "                %s                ",
SDL_GetKeyName(touche_aller_a_droite));
00549         sprintf(itemsTouches[3].texte, "                %s                ",
SDL_GetKeyName(touche_aller_a_gauche));
00550         sprintf(itemsTouches[5].texte, "                %s                ",
SDL_GetKeyName(touche_sauter_monter));
00551         sprintf(itemsTouches[7].texte, "                %s                ",
SDL_GetKeyName(touche_descendre));
00552         sprintf(itemsTouches[9].texte, "                %s                ",
SDL_GetKeyName(touche_interagir));
00553
00554         /* Récupération des informations */
00555         fscanf(fichier_sauvegarde, "%[^\\n]\\n", pseudo.texte);
00556
00557         fscanf(fichier_sauvegarde, "%d\\n", (int*)&personnageActif);
00558
00559         fscanf(fichier_sauvegarde, "%d\\n", (int*)&modeActif);
00560
00561         fscanf(fichier_sauvegarde, "%d\\n", (int*)&positionActive);
00562
00563         for(i = 0; i < tailleNiveaux; i++)
00564             fscanf(fichier_sauvegarde, "%d %d %d %d\\n", &(avancee_niveaux[i].niveau_fini),
&(avancee_niveaux[i].numero_collectible[0]),
00565                                     &(avancee_niveaux[i].numero_collectible[1]),
00566                                     &(avancee_niveaux[i].numero_collectible[2]));
00567
00568         fscanf(fichier_sauvegarde, "%ld\\n", &temps_debut_partie);
00569
00570
00571         fscanf(fichier_sauvegarde, "%d\\n", &compteur_mort);
00572
00573         for(i = 0; i < 10; i++)
00574             fscanf(fichier_sauvegarde, "%d ", &(avancee_succes[i]));
00575
00576         /* Fermeture du fichier */
00577         if (fclose(fichier_sauvegarde) != 0)
00578             erreur("Fermeture du fichier");
00579     }
00580
00581     /* Initialisation par défaut du son et de l'avancer dans le jeu */

```

```

00582     else {
00583         barre_de_son[0].volume = 0.5;
00584         barre_de_son[1].volume = 0.5;
00585
00586         for(i = 0; i < tailleNiveaux; i++) {
00587
00588             avancee_niveaux[i].niveau_fini = 0;
00589             avancee_niveaux[i].numero_collectible[0] = 0;
00590             avancee_niveaux[i].numero_collectible[1] = 0;
00591             avancee_niveaux[i].numero_collectible[2] = 0;
00592         }
00593
00594         for(i = 0; i < 10; i++)
00595             avancee_succes[i] = 0;
00596
00597         compteur_mort = 0;
00598     }
00599
00600     if(barre_de_son[0].volume)
00601         sonsActifs[0] = SDL_TRUE;
00602
00603     else
00604         sonsActifs[0] = SDL_FALSE;
00605
00606     if(barre_de_son[1].volume)
00607         sonsActifs[1] = SDL_TRUE;
00608
00609     else
00610         sonsActifs[1] = SDL_FALSE;
00611
00612     /*----- Chargement du jeu -----*/
00613
00614     SDL_SetWindowResizable(window, SDL_FALSE);
00615
00616     /* Chargement des sprites */
00617     SDL_Texture *sprites[3];
00618
00619     chargement_image(&renderer, &surface, &sprites[0],
00620 ".images/personnages/personnage_masculin_droite_NB.png");
00621     chargement_image(&renderer, &surface, &sprites[1],
00622 ".images/personnages/personnage_masculin_bas_droit_1_NB.png");
00623     chargement_image(&renderer, &surface, &sprites[2],
00624 ".images/personnages/personnage_masculin_bas_droit_2_NB.png");
00625
00626     int chargement = 1;
00627     int indice_sprite = 0;
00628     int pourcentage = 0;
00629
00630     while (chargement) {
00631         srand(time(NULL));
00632
00633         /* Simulation du chargement du jeu */
00634         pourcentage += (rand() % 6) + 1;
00635
00636         if (pourcentage >= 100)
00637             chargement = 0;
00638
00639         if(pourcentage > 100)
00640             pourcentage = 100;
00641
00642         /* Rendu du chargement du jeu */
00643
00644         /* Nettoyage du rendu */
00645         SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);
00646         SDL_RenderClear(renderer);
00647
00648         /* Afficher le sprite actuel */
00649
00650         rectangle_perso_1.x = ((largeur - largeur / 2) / 2 - 50) + (largeur / 2 * pourcentage / 100);
00651         rectangle_perso_1.y = (hauteur - 100) / 3;
00652         rectangle_perso_1.w = 100;
00653         rectangle_perso_1.h = 100;
00654
00655         SDL_RenderCopy(renderer, sprites[indice_sprite], NULL, &rectangle_perso_1);
00656
00657         /* Barre de chargement */
00658
00659         SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255);
00660
00661         rectangle_pseudo.x = (largeur - largeur / 2) / 2;
00662         rectangle_pseudo.y = hauteur / 2 - hauteur / 40;
00663         rectangle_pseudo.w = largeur / 2;
00664         rectangle_pseudo.h = hauteur / 20;
00665
00666         SDL_RenderFillRect(renderer, &rectangle_pseudo);

```



```

00665
00666     rectangle_pseudo.w = largeur / 2 * pourcentage / 100;
00667
00668     SDL_SetRenderDrawColor(renderer, 0, 255, 0, 255);
00669
00670     SDL_RenderFillRect(renderer, &rectangle_pseudo);
00671
00672     /* Texte du pourcentage */
00673
00674     char texte_chargement[20];
00675
00676     sprintf(texte_chargement, "Chargement... %d%%", pourcentage);
00677
00678     surface = TTF_RenderText_Solid(police, texte_chargement, couleurBlanche);
00679
00680     texture_texte = SDL_CreateTextureFromSurface(renderer, surface);
00681
00682     rectangle_texte_introduction.x = (largeur - surface->w) / 2;
00683     rectangle_texte_introduction.y = hauteur / 2 + surface->h + 20;
00684     rectangle_texte_introduction.w = surface->w;
00685     rectangle_texte_introduction.h = surface->h;
00686
00687     SDL_RenderCopy(renderer, texture_texte, NULL, &rectangle_texte_introduction);
00688
00689     SDL_FreeSurface(surface);
00690     SDL_DestroyTexture(texture_texte);
00691
00692     /* Mise à jour de l'écran */
00693     SDL_RenderPresent(renderer);
00694
00695     /* Ajout d'un délai pour simuler le chargement */
00696     SDL_Delay(200);
00697
00698     /* Passer au sprite suivant */
00699     indice_sprite++;
00700
00701     /* Revenir au premier sprite si tous les sprites ont été affichés */
00702     if (indice_sprite >= 3)
00703         indice_sprite = 0;
00704 }
00705
00706 SDL_SetWindowResizable(window, SDL_TRUE);
00707
00708 /*----- Début du jeu -----*/
00709
00710 page_t page_active = MENU_PRINCIPAL;
00711
00712 page_t page_precedente = MENU_PRINCIPAL;
00713
00714 SDL_Event event;
00715
00716 Mix_VolumeMusic(barre_de_son[0].volume * 100);
00717
00718 Mix_Volume(1, barre_de_son[1].volume * 100);
00719
00720 /* Musique du menu principal */
00721 if ((musique = Mix_LoadMUS("./sons/musiques/menu_principal.mp3")) == NULL)
00722     erreur("Chargement de la musique");
00723
00724 Mix_PlayMusic(musique, -1);
00725
00726 while(programme_lance) {
00727
00728     /* Page du menu principal */
00729     if (page_active == MENU_PRINCIPAL) {
00730
00731         page_precedente = MENU_PRINCIPAL;
00732
00733         menu_principal(&event, &window, &renderer, &programme_lance, &texture_image_menu,
00734             &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
00735             &titre_menu_principal, &surface, &texture_texte, &police,
00736             couleurTitre, couleurNoire, code_de_triche, &selection_menu,
00737             itemsMenuPrincipal, tailleMenuPrincipal, &largeur, &hauteur, &page_active);
00738
00739         if ((code_de_triche[0]) && (code_de_triche[1]) && (code_de_triche[2])) {
00740
00741             for (i = 0; i < 4; i++) {
00742
00743                 avancee_niveaux[i].niveau_fini = 1;
00744
00745                 for (x = 0; x < 3; x++)
00746                     avancee_niveaux[i].numero_collectible[x] = 0;
00747             }
00748
00749             for (i = 0; i < 10; i++) {
00750

```

```

00751         if((!i) || (i == 2) || (i == 5) || (i == 7))
00752             avancee_succes[i] = 1;
00753         else
00754             avancee_succes[i] = 0;
00755     }
00756
00757     temps_debut_partie = time(NULL);
00758
00759     compteur_mort = 0;
00760
00761     strcpy(pseudo.texte, "mode Dev");
00762
00763     personnageActif = PERSONNAGE_1;
00764
00765     modeActif = MODE_NORMAL;
00766
00767     positionActive = NIVEAU1;
00768
00769     /* Libération de la mémoire allouée dynamiquement */
00770     free(itemsMenuPrincipal);
00771
00772     tailleMenuPrincipal = 3;
00773
00774     /* Allocation dynamique de mémoire pour le tableau itemsMenuPrincipal en fonction de
00775     tailleMenuPrincipal */
00776     itemsMenuPrincipal = malloc(tailleMenuPrincipal * sizeof(itemMenu));
00777
00778     initialisation_objets_menu_principal(&renderer, &surface, &texture_image_menu,
00779     &titre_menu_principal, itemsMenuPrincipal,
00780     tailleMenuPrincipal);
00781
00782     code_de_triche[0] = 0;
00783     code_de_triche[1] = 0;
00784     code_de_triche[2] = 0;
00785 }
00786
00787 /* Cas où on clique sur nouvelle partie */
00788 if(page_active == NOUVELLE_PARTIE) {
00789     selection_menu = 0;
00790
00791     pseudo_valide = 0;
00792
00793     /* Sauvegarde des variables de la nouvelle partie dans des variables temporaires */
00794     strcpy(pseudo_temporaire, pseudo.texte);
00795     personnage_temporaire = personnageActif;
00796     mode_temporaire = modeActif;
00797
00798     /* Initialisation des variables de la nouvelle partie à NULL */
00799     pseudo.texte[0] = '\0';
00800     personnageActif = PERSONNAGE_1;
00801     modeActif = MODE_NORMAL;
00802
00803     /* Initialisation de la police */
00804     if((police= TTF_OpenFont("./polices/04B_11__.TTF", largeur / 35)) == NULL)
00805         erreur("Chargement de la police");
00806 }
00807
00808 /* Cas où on clique sur options ou nouvelle partie */
00809 if((page_active == OPTIONS) || (page_active == NOUVELLE_PARTIE)) {
00810     selection_menu = 0;
00811
00812     for(i = 0; i < 4; i++) {
00813         niveau_fini[i] = 0;
00814
00815         for(x = 0; x < 3; x++)
00816             collectibles[i + x] = 0;
00817
00818         position_intermediaire = NIVEAU1;
00819     }
00820
00821     chargement_image(&renderer, &surface, &texture_image_plein_ecran,
00822     "./images/plein_ecran.png");
00823     chargement_image(&renderer, &surface, &texture_image_options, "./images/options.png");
00824 }
00825
00826 /* Cas où on clique sur continuer */
00827 if(page_active == CARTE) {
00828     selection_menu = 0;
00829
00830     for(i = 0; i < 4; i++) {
00831         niveau_fini[i] = avancee_niveaux[i].niveau_fini;
00832     }
00833 }
00834

```

```

00835         for(x = 0; x < 3; x++)
00836             collectibles[i + x] = avancee_niveaux[i].numero_collectible[x];
00837
00838         position_intermediaire = positionActive;
00839     }
00840
00841     for(i = 0; i < 10; i++)
00842         avancee_succes_intermediaires[i] = avancee_succes[i];
00843
00844     if(!avancee_niveaux[0].niveau_fini)
00845         chargement_image(&renderer, &surface, &texture_image_carte,
00846             "./images/carte_niveau_2_bloque.jpg");
00847
00848     else if(!avancee_niveaux[1].niveau_fini)
00849         chargement_image(&renderer, &surface, &texture_image_carte,
00850             "./images/carte_niveau_3_bloque.jpg");
00851
00852     else if(!avancee_niveaux[2].niveau_fini)
00853         chargement_image(&renderer, &surface, &texture_image_carte,
00854             "./images/carte_niveau_4_bloque.jpg");
00855
00856     else
00857         chargement_image(&renderer, &surface, &texture_image_carte, "./images/carte.jpg");
00858
00859     direction = BAS;
00860
00861     /* Musique de la carte */
00862     if((musique = Mix_LoadMUS("./sons/musiques/carte.mp3")) == NULL)
00863         erreur("Chargement de la musique");
00864
00865     Mix_PlayMusic(musique, -1);
00866 }
00867
00868 /* Page des options */
00869 else if(page_active == OPTIONS) {
00870
00871     /* Mise à jour du volume d ela musique et des effets sonores*/
00872     Mix_VolumeMusic(barre_de_son[0].volume * 100);
00873     Mix_Volume(1, barre_de_son[1].volume * 100);
00874
00875     options(&event, &window, &renderer, &programme_lance,
00876         &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
00877         &rectangle_retour_en_arriere, &texture_image_retour_en_arriere,
00878         &texture_image_hautParleurActif, &rectangle_demande, itemsDemandeSauvegarde,
00879         tailleDemande,
00880         &texture_image_hautParleurDesactive, sonsActifs,
00881         rectangles_boutons_sons, &ongletActif, &pseudo,
00882         &modeActif, &personnageActif, &positionActive,
00883         avancee_niveaux, tailleNiveaux,
00884         &titre_options, &surface, &texture_texte, &police,
00885         &selection_touche, &touche_aller_a_droite, &touche_aller_a_gauche,
00886         &touche_sauter_monter,
00887         &touche_descendre, &touche_interagir, couleurNoire,
00888         itemsMenuOptions, tailleMenuOptions, itemsTouches, tailleTouches,
00889         barre_de_son, tailleBarres, itemsBarres,
00890         &largeur, &hauteur, &page_active, &page_precedente, &maintient_clic,
00891         temps_debut_partie, &compteur_mort, avancee_succes);
00892
00893     if((page_active != OPTIONS) && (page_active != NOUVELLE_PARTIE)) {
00894         chargement_image(&renderer, &surface, &texture_image_plein_ecran,
00895             "./images/plein_ecran_blanc.png");
00896         chargement_image(&renderer, &surface, &texture_image_options,
00897             "./images/options_blanc.png");
00898     }
00899
00900     /* Page de la nouvelle partie */
00901     else if(page_active == NOUVELLE_PARTIE) {
00902
00903         page_precedente = NOUVELLE_PARTIE;
00904
00905         nouvelle_partie(&event, &window, &renderer, &programme_lance,
00906             &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
00907             &rectangle_retour_en_arriere, &texture_image_retour_en_arriere,
00908             &rectangle_options, &texture_image_options, &modeSaisie,
00909             &modeActif, &texture_image_perso_1, &rectangle_perso_1,
00910             &texture_image_perso_2, &rectangle_perso_2, &personnageActif,
00911             &pseudo, &rectangle_pseudo, barre_de_son, &pseudo_valide,
00912             &touche_aller_a_droite, &touche_aller_a_gauche, &touche_sauter_monter,
00913             &touche_descendre, &touche_interagir, titres, tailleTitres, &surface,
00914             &texture_texte,
00915             &police, couleurNoire, &positionActive, avancee_niveaux, tailleNiveaux,
00916             itemsMenuNouvellePartie, &valider, &largeur, &hauteur, &page_active,
00917             &temps_debut_partie, &compteur_mort, avancee_succes);
00918
00919     /* Cas où on retourne en arrière */

```

```

00914         if(page_active == MENU_PRINCIPAL) {
00915             chargement_image(&renderer, &surface, &texture_image_plein_ecran,
00916                 "./images/plein_ecran_blanc.png");
00917             chargement_image(&renderer, &surface, &texture_image_options,
00918                 "./images/options_blanc.png");
00918
00919             /* Réinitialisation des variables de la nouvelle partie avec les anciennes valeurs
sauvegardées */
00920             strcpy(pseudo.texte, pseudo_temporaire);
00921             personnageActif = personnage_temporaire;
00922             modeActif = mode_temporaire;
00923         }
00924
00925         /* Cas où on commence une nouvelle partie */
00926         else if(page_active == INTRODUCTION) {
00927             /* Musique de l'introduction */
00928             if((musique = Mix_LoadMUS("./sons/musiques/introduction.mp3")) == NULL)
00929                 erreur("Chargement de la musique");
00930
00931             Mix_PlayMusic(musique, -1);
00932
00933             chargement_image(&renderer, &surface, &texture_image_carte,
00934                 "./images/carte_niveau_2_bloque.jpg");
00935
00936             /* Actualisation de la taille de la police pour l'introduction */
00937             if((police = TTF_OpenFont("./polices/02587_ARIALMT.ttf", largeur / 50)) == NULL)
00938                 erreur("Chargement de la police");
00939         }
00940     }
00941
00942     /* Page de l'introduction */
00943
00944     else if(page_active == INTRODUCTION) {
00945         introduction(&event, &window, &renderer, &programme_lance,
00946             &rectangle_passer, &texture_image_passer,
00947             &rectangle_texte_introduction, &surface, &texture_texte, &police,
00948             &personnageActif, couleurBlanche,
00949             &largeur, &hauteur, &page_active);
00950
00951         /* On entre directement dans le niveau 1 après l'introduction */
00952         page_active = NIVEAU_1;
00953
00954         /* Initialisation de la police d'origine */
00955         if((police = TTF_OpenFont("./polices/04B_11_.TTF", 20)) == NULL)
00956             erreur("Chargement de la police");
00957
00958         /* Musique du niveau 1 */
00959         if((musique = Mix_LoadMUS("./sons/musiques/niveau_1.mp3")) == NULL)
00960             erreur("Chargement de la musique");
00961
00962         Mix_PlayMusic(musique, -1);
00963
00964         /* Initialisation par défaut du niveau */
00965         largeur_tile = largeur / 32;
00966         hauteur_tile = hauteur / 18;
00967
00968         decalage = 0;
00969         secret_1 = 0;
00970         secret_2 = 0;
00971
00972         sauter = 0;
00973         avancer = 0;
00974         reculer = 0;
00975         tombe = 0;
00976         saut = 0;
00977
00978         mouvement_monstre = 0;
00979
00980         /* Initialisation collectible récolté */
00981         for(i = 0; i < 3; i++)
00982             collectibles_intermediaires[i] = 0;
00983
00984         /* Initialisation des succès déjà obtenus */
00985         for(i = 0; i < 10; i++)
00986             avancee_succes_intermediaires[i] = 0;
00987
00988         chargement_image(&renderer, &surface, &texture_image_plein_ecran,
00989             "./images/plein_ecran_blanc.png");
00990         chargement_image(&renderer, &surface, &texture_image_options,
00991             "./images/options_blanc.png");
00992         chargement_niveau_1(&position_x, &position_y, &position_x_initiale, &position_y_initiale,
00993             tile_map_niveau_1);
00994     }

```

```

00994         for (y = 0; y < 18; y++)
00995             for (x = 0; x < 32; x++)
00996                 tile_map[y][x] = tile_map_niveau_1[y][13 + x];
00997     }
00998
00999     /* Page de la carte */
01000     else if (page_active == CARTE) {
01001
01002         page_precedente = CARTE;
01003
01004         /* Cas où le personnage choisit est masculin */
01005         if (personnageActif == PERSONNAGE_1)
01006             carte(&event, &window, &renderer, &programme_lance, &texture_image_carte,
01007                 &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
01008                 &rectangle_options, &texture_image_options, &rectangle_retour_menu,
01009                 &texture_image_retour_menu,
01010                 &texture_image_perso_1_bas_1, &texture_image_perso_1_bas_2,
01011                 &texture_image_perso_1_haut_1, &texture_image_perso_1_haut_2,
01012                 &texture_image_perso_1_bas_gauche_1, &texture_image_perso_1_bas_gauche_2,
01013                 &texture_image_perso_1_haut, &texture_image_perso_1_droite,
01014                 &texture_image_perso_1_gauche, &texture_image_perso_1_pose,
01015                 &texture_image_perso_1, &rectangle_perso_1, avancee_niveaux,
01016                 &texture_image_fin_dernier_niveau, &rectangle_succes,
01017                 &surface, &texture_texte, &police, &direction, &touche_pressee,
01018                 &rectangle_demande, itemsDemandeSauvegarde, tailleDemande,
01019                 &positionActive, barre_de_son, &pseudo, &modeActif, &personnageActif,
01020                 couleurNoire, &touche_aller_a_droite, &touche_aller_a_gauche,
01021                 &touche_sauter_monter, &touche_descendre, &touche_interagir,
01022                 itemsNiveaux, tailleNiveaux, &largeur, &hauteur, &page_active,
01023                 itemsSucces, textures_images_succes,
01024                 temps_debut_partie, &compteur_mort, avancee_succes,
01025                 avancee_succes_intermediaires);
01026
01027         /* Cas où le personnage choisit est féminin */
01028         else
01029             carte(&event, &window, &renderer, &programme_lance, &texture_image_carte,
01030                 &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
01031                 &rectangle_options, &texture_image_options, &rectangle_retour_menu,
01032                 &texture_image_retour_menu,
01033                 &texture_image_perso_2_bas_1, &texture_image_perso_2_bas_2,
01034                 &texture_image_perso_2_haut_1, &texture_image_perso_2_haut_2,
01035                 &texture_image_perso_2_bas_gauche_1, &texture_image_perso_2_bas_gauche_2,
01036                 &texture_image_perso_2_haut, &texture_image_perso_2_droite,
01037                 &texture_image_perso_2_gauche, &texture_image_perso_2_pose,
01038                 &texture_image_perso_2, &rectangle_perso_1, avancee_niveaux,
01039                 &texture_image_fin_dernier_niveau, &rectangle_succes,
01040                 &surface, &texture_texte, &police, &direction, &touche_pressee,
01041                 &rectangle_demande, itemsDemandeSauvegarde, tailleDemande,
01042                 &positionActive, barre_de_son, &pseudo, &modeActif, &personnageActif,
01043                 couleurNoire, &touche_aller_a_droite, &touche_aller_a_gauche,
01044                 &touche_sauter_monter, &touche_descendre, &touche_interagir,
01045                 itemsNiveaux, tailleNiveaux, &largeur, &hauteur, &page_active,
01046                 itemsSucces, textures_images_succes,
01047                 temps_debut_partie, &compteur_mort, avancee_succes,
01048                 avancee_succes_intermediaires);
01049
01050         /* Cas où on va dans les options */
01051         if (page_active == OPTIONS) {
01052             chargement_image(&renderer, &surface, &texture_image_plein_ecran,
01053                 &texture_image_plein_ecran.png);
01054             chargement_image(&renderer, &surface, &texture_image_options, &texture_image_options.png);
01055         }
01056
01057         /* Ca soù on retourne dans le menu principal */
01058         if (page_active == MENU_PRINCIPAL) {
01059
01060             /* Musique du menu principal */
01061             if ((musique = Mix_LoadMUS("./sons/musiques/menu_principal.mp3")) == NULL)
01062                 erreur("Chargement de la musique");
01063
01064             Mix_PlayMusic(musique, -1);
01065         }
01066
01067         /* Cas où on rentre dans le niveau 1 */
01068         if (page_active == NIVEAU_1) {
01069
01070             /* Musique du niveau 1 */
01071             if ((musique = Mix_LoadMUS("./sons/musiques/niveau_1.mp3")) == NULL)
01072                 erreur("Chargement de la musique");
01073
01074             Mix_PlayMusic(musique, -1);
01075
01076             /* Initialisation par défaut du niveau */
01077             largeur_tile = largeur / 32;
01078             hauteur_tile = hauteur / 18;

```

```

01074
01075         decalage = 0;
01076         secret_1 = 0;
01077         secret_2 = 0;
01078
01079         sauter = 0;
01080         avancer = 0;
01081         reculer = 0;
01082         tombe = 0;
01083         saut = 0;
01084
01085         mouvement_monstre = 0;
01086
01087         /* Initialisation collectible récolté */
01088         for(i = 0; i < 3; i++)
01089             collectibles_intermediaires[i] = avancee_niveaux[0].numero_collectible[i];
01090
01091         /* Initialisation des succès déjà obtenus */
01092         for(i = 0; i < 10; i++)
01093             avancee_succes_intermediaires[i] = avancee_succes[i];
01094
01095         chargement_niveau_1(&position_x, &position_y, &position_x_initiale,
01096                             &position_y_initiale, tile_map_niveau_1);
01097
01098         for (y = 0; y < 18; y++)
01099             for (x = 0; x < 32; x++)
01100                 tile_map[y][x] = tile_map_niveau_1[y][13 + x];
01101     }
01102
01103     /* Cas où on rentre dans le niveau 2 ou 3 */
01104     else if((page_active == NIVEAU_2) || (page_active == NIVEAU_3)) {
01105
01106         /* Musique du salon */
01107         if((musique = Mix_LoadMUS("./sons/musiques/salon.mp3")) == NULL)
01108             erreur("Chargement de la musique");
01109
01110         Mix_PlayMusic(musique, -1);
01111
01112         /* Initialisation par défaut du niveau */
01113         largeur_tile = largeur / 32;
01114         hauteur_tile = hauteur / 18;
01115
01116         sauter = 0;
01117         avancer = 0;
01118         reculer = 0;
01119         tombe = 0;
01120         saut = 0;
01121
01122         mouvement_monstre = 0;
01123
01124         mini_jeu = 0;
01125
01126         mode_difficile = 0;
01127
01128         mini_jeu_1_termine = 0;
01129         mini_jeu_2_termine = 0;
01130
01131         mini_jeu_termine = 0;
01132
01133         /* Initialisation collectible récolté */
01134         if(page_active == NIVEAU_2)
01135             for(i = 0; i < 3; i++)
01136                 collectibles_intermediaires[i] = avancee_niveaux[1].numero_collectible[i];
01137
01138         /* Initialisation collectible récolté */
01139         else
01140             for(i = 0; i < 3; i++)
01141                 collectibles_intermediaires[i] = avancee_niveaux[2].numero_collectible[i];
01142
01143         /* Initialisation des succès déjà obtenus */
01144         for(i = 0; i < 10; i++)
01145             avancee_succes_intermediaires[i] = avancee_succes[i];
01146
01147         salon_arrivee_niveaux_2_3(&position_x, &position_y, tile_map, page_active);
01148     }
01149
01150     /* Cas où on rentre dans le niveau 4 */
01151     else if(page_active == NIVEAU_4) {
01152
01153         /* Musique du niveau 4 */
01154         if((musique = Mix_LoadMUS("./sons/musiques/niveau_4.mp3")) == NULL)
01155             erreur("Chargement de la musique");
01156
01157         Mix_PlayMusic(musique, -1);
01158
01159         /* Initialisation par défaut du niveau */

```

```

01160         largeur_tile = largeur / 32;
01161         hauteur_tile = hauteur / 18;
01162
01163         sauter = 0;
01164         avancer = 0;
01165         reculer = 0;
01166         tombe = 0;
01167         saut = 0;
01168
01169         /* Initialisation collectible récolté */
01170         for(i = 0; i < 3; i++)
01171             collectibles_intermediaires[i] = avancee_niveaux[3].numero_collectible[i];
01172
01173         /* Initialisation des succès déjà obtenus */
01174         for(i = 0; i < 10; i++)
01175             avancee_succes_intermediaires[i] = avancee_succes[i];
01176
01177         /* On entre dans l'étage 1 de la tour (total 5 étages) */
01178         etage_1(&position_x, &position_y, &position_x_initiale, &position_y_initiale,
01179 tile_map,
01180             &renderer, &surface, &texture_image_mur);
01181     }
01182     else if(page_active == MENU_PRINCIPAL) {
01183
01184         /* Libération de la mémoire allouée dynamiquement */
01185         free(itemsMenuPrincipal);
01186
01187         tailleMenuPrincipal = 3;
01188
01189         /* Allocation dynamique de mémoire pour le tableau itemsMenuPrincipal en fonction de
01190 tailleMenuPrincipal */
01191         itemsMenuPrincipal = malloc(tailleMenuPrincipal * sizeof(itemMenu));
01192
01193         initialisation_objets_menu_principal(&renderer, &surface, &texture_image_menu,
01194             &titre_menu_principal, itemsMenuPrincipal,
01195         tailleMenuPrincipal);
01196     }
01197     touche_pressee = 0;
01198 }
01199
01200 /* Page du niveau 1 */
01201 else if(page_active == NIVEAU_1) {
01202     /* Cas où le personnage choisit est masculin */
01203     if(personnageActif == PERSONNAGE_1)
01204         niveau_1(&event, &window, &renderer, &programme_lance, &texture_image_croix,
01205 &rectangle_croix,
01206             &texture, &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
01207             &texture_image_perso_1, &rectangle_perso_1, &texture_image_monstre_terrestre,
01208             &texture_image_monstre_volant,
01209             &texture_image_sol_surface_niveau_1, &texture_image_sol_profondeur_niveau_1,
01210             &texture_image_fond_niveau_1,
01211             &texture_image_pique, avancee_niveaux, &mouvement_monstre,
01212             &surface, collectibles_intermediaires, &timestamp,
01213             &touche_aller_a_droite, &touche_aller_a_gauche,
01214             &touche_sauter_monter, &decalage, &secret_1, &secret_2,
01215             tile_map, tile_map_niveau_1, &rectangle_tile,
01216             itemsDemandeQuitter, tailleDemande, &texture_image_perso_1_gagnant,
01217             &texture_texte, &police, &rectangle_demande, &texture_image_nuage_1,
01218             &texture_image_nuage_2,
01219             couleurNoire, &texture_image_fin_premiers_niveaux,
01220             &avancer, &reculer, &sauter, &position_avant_saut, &saut, &tombe,
01221             &position_x_initiale, &position_y_initiale, &position_x, &position_y,
01222             &largeur, &hauteur, &largeur_tile, &hauteur_tile, &page_active,
01223             itemsDemandeSauvegarde, &touche_descendre, &touche_interagir, barre_de_son,
01224             &pseudo,
01225             &modeActif, &personnageActif, &positionActive, tailleNiveaux,
01226             temps_debut_partie, &compteur_mort, avancee_succes,
01227             avancee_succes_intermediaires);
01228
01229     /* Cas où le personnage choisit est féminin */
01230     else
01231         niveau_1(&event, &window, &renderer, &programme_lance, &texture_image_croix,
01232 &rectangle_croix,
01233             &texture, &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
01234             &texture_image_perso_2, &rectangle_perso_2, &texture_image_monstre_terrestre,
01235             &texture_image_monstre_volant,
01236             &texture_image_sol_surface_niveau_1, &texture_image_sol_profondeur_niveau_1,
01237             &texture_image_fond_niveau_1,
01238             &texture_image_pique, avancee_niveaux, &mouvement_monstre,
01239             &surface, collectibles_intermediaires, &timestamp,
01240             &touche_aller_a_droite, &touche_aller_a_gauche,
01241             &touche_sauter_monter, &decalage, &secret_1, &secret_2,
01242             tile_map, tile_map_niveau_1, &rectangle_tile,
01243             itemsDemandeQuitter, tailleDemande, &texture_image_perso_2_gagnant,

```





```

01300         /* Cas où le personnage choisit est féminin */
01301         else
01302             arrivee_niveaux_2_3(&event, &window, &renderer, &programme_lance, &mini_jeu,
&texture_image_fin_premiers_niveaux,
01303                                     &texture, &surface, &rectangle_plein_ecran,
&texture_image_plein_ecran, &plein_ecran,
01304                                     &texture_image_perso_2, &rectangle_perso_2, &mini_jeu_termine,
&mini_jeu_1_termine, &mini_jeu_2_termine,
01305                                     &texture_image_fond_niveau_2, &texture_image_sol_niveau_2,
&texture_image_monstre_terrestre, &texture_image_monstre_volant,
01306                                     &touche_aller_a_droite, &touche_aller_a_gauche, &touche_interagir,
&texture_image_porte, avancee_niveaux,
01307                                     &touche_sauter_monter, &touche_descendre,
&texture_image_dossier_niveau_2,
01308                                     NULL, NULL, NULL, NULL,
01309                                     tile_map, &rectangle_tile, &mouvement_monstre, &modeActif,
&mode_difficile,
01310                                     itemsDemandeQuitter, tailleDemande, couleurNoire,
tile_map_mini_jeu_niveau_2,
01311                                     &texture_texte, &police, &rectangle_demande, &timestamp,
&texture_image_perso_2_gagnant,
01312                                     &avancer, &reculer, &sauter, &position_avant_saut, &saut, &tombe,
&position_x_initiale, &position_y_initiale, &position_x,
01313                                     &position_y,
&largeur, &hauteur, &largeur_tile, &hauteur_tile, &page_active,
01314                                     &texture_image_mur_mini_jeu, collectibles_intermediaires,
01315                                     itemsExplications,
&texture_image_pipe_vertical, &texture_image_pipe_horizontal,
01316                                     &texture_image_pipe_haut_droit, &texture_image_pipe_bas_droit,
&texture_image_pipe_bas_gauche, &texture_image_pipe_haut_gauche,
01317                                     &texture_image_pipe_courant, &texture_image_mur_termine, &valide,
01318                                     rectangle_piece, piece_bloquee, rectangle_emplacement_piece,
01319                                     &piece_selectionnee,
&decalage_x, &decalage_y, &texture_image_puzzle, &musique,
01320                                     &texture_image_croix, &rectangle_croix,
01321                                     tile_map_mini_jeu_niveau_3, &descendre, &interagir, &bloc_x,
&bloc_y,
01322                                     &texture_image_sol_labyrinthe, &texture_image_bordure_labyrinthe,
&texture_image_fin_labyrinthe, couleurTitre,
01323                                     itemsDemandeSauvegarde, barre_de_son, &pseudo,
01324                                     &personnageActif, &positionActive, tailleNiveaux,
01325                                     temps_debut_partie, &compteur_mort, avancee_succes,
01326                                     avancee_succes_intermediaires);
01327
01328         /* Retour sur la carte */
01329         if(page_active == CARTE) {
01330
01331             if(!avancee_niveaux[0].niveau_fini)
01332                 chargement_image(&renderer, &surface, &texture_image_carte,
"./images/carte_niveau_2_bloque.jpg");
01333
01334             else if(!avancee_niveaux[1].niveau_fini)
01335                 chargement_image(&renderer, &surface, &texture_image_carte,
"./images/carte_niveau_3_bloque.jpg");
01336
01337             else if(!avancee_niveaux[2].niveau_fini)
01338                 chargement_image(&renderer, &surface, &texture_image_carte,
"./images/carte_niveau_4_bloque.jpg");
01339
01340             else
01341                 chargement_image(&renderer, &surface, &texture_image_carte, "./images/carte.jpg");
01342
01343             direction = BAS;
01344
01345             /* Musique de la carte */
01346             if((musique = Mix_LoadMUS("./sons/musiques/carte.mp3")) == NULL)
01347                 erreur("Chargement de la musique");
01348
01349             Mix_PlayMusic(musique, -1);
01350         }
01351     }
01352
01353     /* Page du niveau 3 */
01354     else if(page_active == NIVEAU_3) {
01355
01356         /* Cas où le personnage choisit est masculin */
01357         if(personnageActif == PERSONNAGE_1)
01358             arrivee_niveaux_2_3(&event, &window, &renderer, &programme_lance, &mini_jeu,
&texture_image_fin_premiers_niveaux,
01359                                     &texture, &surface, &rectangle_plein_ecran,
&texture_image_plein_ecran, &plein_ecran,
01360                                     &texture_image_perso_1, &rectangle_perso_1, &mini_jeu_termine,
&mini_jeu_1_termine, &mini_jeu_2_termine,
01361                                     &texture_image_fond_niveau_3, &texture_image_sol_niveau_3, NULL,
NULL,
01362                                     &touche_aller_a_droite, &touche_aller_a_gauche, &touche_interagir,
&texture_image_porte, avancee_niveaux,

```

```

01363         &texture_image_dossier_niveau_3,          &touche_sauter_monter, &touche_descendre,
01364         &barre_windows_1, &barre_windows_2, &barre_windows_3,
01365         &barre_windows_4, tile_map, &rectangle_tile, &mouvement_monstre,
01366         &modeActif, &mode_difficile,
01367         itemsDemandeQuitter, tailleDemande, couleurNoire,
01368         &texture_texte, &police, &rectangle_demande, &timestamp,
01369         &avancer, &reculer, &sauter, &position_avant_saut, &saut, &tombe,
01370         &position_x_initiale, &position_y_initiale, &position_x,
01371         &position_y,
01372         &largeur, &hauteur, &largeur_tile, &hauteur_tile, &page_active,
01373         &texture_image_mur_mini_jeu, collectibles_intermediaires,
01374         &texture_image_pipe_vertical, &texture_image_pipe_horizontal,
01375         &texture_image_pipe_haut_droit, &texture_image_pipe_bas_droit,
01376         &texture_image_pipe_bas_gauche, &texture_image_pipe_haut_gauche,
01377         &texture_image_pipe_courant, &texture_image_mur_termine, &valide,
01378         rectangle_piece, piece_bloquee, rectangle_emplacement_piece,
01379         &piece_selectionnee,
01380         &decalage_x, &decalage_y, &texture_image_puzzle, &musique,
01381         &texture_image_croix, &rectangle_croix,
01382         tile_map_mini_jeu_niveau_3, &descendre, &interagir, &bloc_x,
01383         &bloc_y,
01384         &texture_image_sol_labyrinthe, &texture_image_bordure_labyrinthe,
01385         &texture_image_fin_labyrinthe, couleurTitre,
01386         itemsDemandeSauvegarde, barre_de_son, &pseudo,
01387         &personnageActif, &positionActive, tailleNiveaux,
01388         temps_debut_partie, &compteur_mort, avancee_succes,
01389         avancee_succes_intermediaires);
01390
01391         /* Cas où le personnage choisit est féminin */
01392         else
01393             arrivee_niveaux_2_3(&event, &window, &render, &programme_lance, &mini_jeu,
01394             &texture_image_fin_premiers_niveaux,
01395             &texture, &surface, &rectangle_plein_ecran,
01396             &texture_image_plein_ecran, &plein_ecran,
01397             &texture_image_perso_2, &rectangle_perso_2, &mini_jeu_termine,
01398             &mini_jeu_1_termine, &mini_jeu_2_termine,
01399             &texture_image_fond_niveau_3, &texture_image_sol_niveau_3, NULL,
01400             NULL,
01401             &touche_aller_a_droite, &touche_aller_a_gauche, &touche_interagir,
01402             &texture_image_porte, avancee_niveaux,
01403             &touche_sauter_monter, &touche_descendre,
01404             &texture_image_dossier_niveau_3,
01405             &barre_windows_1, &barre_windows_2, &barre_windows_3,
01406             &barre_windows_4, tile_map, &rectangle_tile, &mouvement_monstre,
01407             &modeActif, &mode_difficile,
01408             itemsDemandeQuitter, tailleDemande, couleurNoire,
01409             &texture_texte, &police, &rectangle_demande, &timestamp,
01410             &avancer, &reculer, &sauter, &position_avant_saut, &saut, &tombe,
01411             &position_x_initiale, &position_y_initiale, &position_x,
01412             &position_y,
01413             &largeur, &hauteur, &largeur_tile, &hauteur_tile, &page_active,
01414             &texture_image_mur_mini_jeu, collectibles_intermediaires,
01415             itemsExplications,
01416             &texture_image_pipe_vertical, &texture_image_pipe_horizontal,
01417             &texture_image_pipe_haut_droit, &texture_image_pipe_bas_droit,
01418             &texture_image_pipe_bas_gauche, &texture_image_pipe_haut_gauche,
01419             &texture_image_pipe_courant, &texture_image_mur_termine, &valide,
01420             rectangle_piece, piece_bloquee, rectangle_emplacement_piece,
01421             &piece_selectionnee, &decalage_x, &decalage_y,
01422             &texture_image_puzzle, &musique, &texture_image_croix, &rectangle_croix,
01423             tile_map_mini_jeu_niveau_3, &descendre, &interagir, &bloc_x,
01424             &bloc_y,
01425             &texture_image_sol_labyrinthe, &texture_image_bordure_labyrinthe,
01426             &texture_image_fin_labyrinthe, couleurTitre,
01427             itemsDemandeSauvegarde, barre_de_son, &pseudo,
01428             &personnageActif, &positionActive, tailleNiveaux,
01429             temps_debut_partie, &compteur_mort, avancee_succes,
01430             avancee_succes_intermediaires);
01431
01432         /* Retour sur la carte */
01433         if(page_active == CARTE) {
01434             if(!avancee_niveaux[0].niveau_fini)
01435                 chargement_image(&render, &surface, &texture_image_carte,
01436                 ".images/carte_niveau_2_bloque.jpg");
01437             else if(!avancee_niveaux[1].niveau_fini)
01438                 chargement_image(&render, &surface, &texture_image_carte,
01439                 ".images/carte_niveau_3_bloque.jpg");
01440             else if(!avancee_niveaux[2].niveau_fini)

```

```

01422         chargement_image(&renderer, &surface, &texture_image_carte,
01423         "./images/carte_niveau_4_bloque.jpg");
01424
01425         else
01426             chargement_image(&renderer, &surface, &texture_image_carte, "./images/carte.jpg");
01427
01428         direction = BAS;
01429
01429         /* Musique de la carte */
01430         if((musique = Mix_LoadMUS("./sons/musiques/carte.mp3")) == NULL)
01431             erreur("Chargement de la musique");
01432
01433         Mix_PlayMusic(musique, -1);
01434     }
01435 }
01436
01437 /* Page du niveau 4 */
01438 else if(page_active == NIVEAU_4) {
01439
01440     /* Cas où le personnage choisit est masculin */
01441     if(personnageActif == PERSONNAGE_1)
01442         niveau_4(&event, &window, &renderer, &programme_lance,
01443         &texture, &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
01444         &texture_image_perso_1, &rectangle_perso_1,
01445         &texture_image_mur, &texture_image_fond_niveau_4,
01446         &texture_image_bordure_niveau_4, &texture_image_porte,
01447         &texture_image_pique, avancee_niveaux,
01448         &surface, &modeActif, collectibles_intermediaires,
01449         &touche_aller_a_droite, &touche_aller_a_gauche,
01450         &touche_sauter_monter, &touche_interagir,
01451         tile_map, &rectangle_tile, &texture_image_perso_1_gagnant,
01452         itemsDemandeQuitter, tailleDemande, &texture_image_croix, &rectangle_croix,
01453         &texture_texte, &police, &rectangle_demande,
01454         couleurNoire, &texture_image_fin_dernier_niveau,
01455         &avancer, &reculer, &sauter, &position_avant_saut, &saut, &tombe,
01456         &numero_etage,
01457         &position_x_initiale, &position_y_initiale, &position_x, &position_y,
01458         &largeur, &hauteur, &largeur_tile, &hauteur_tile, &page_active,
01459         itemsDemandeSauvegarde, &touche_descendre, barre_de_son, &pseudo,
01460         &personnageActif, &positionActive, tailleNiveaux,
01461         temps_debut_partie, &compteur_mort, avancee_succes,
01462         avancee_succes_intermediaires);
01463
01464     /* Cas où le personnage choisit est féminin */
01465     else
01466         niveau_4(&event, &window, &renderer, &programme_lance,
01467         &texture, &rectangle_plein_ecran, &texture_image_plein_ecran, &plein_ecran,
01468         &texture_image_perso_2, &rectangle_perso_2,
01469         &texture_image_mur, &texture_image_fond_niveau_4,
01470         &texture_image_bordure_niveau_4, &texture_image_porte,
01471         &texture_image_pique, avancee_niveaux,
01472         &surface, &modeActif, collectibles_intermediaires,
01473         &touche_aller_a_droite, &touche_aller_a_gauche,
01474         &touche_sauter_monter, &touche_interagir,
01475         tile_map, &rectangle_tile, &texture_image_perso_2_gagnant,
01476         itemsDemandeQuitter, tailleDemande, &texture_image_croix, &rectangle_croix,
01477         &texture_texte, &police, &rectangle_demande,
01478         couleurNoire, &texture_image_fin_dernier_niveau,
01479         &avancer, &reculer, &sauter, &position_avant_saut, &saut, &tombe,
01480         &numero_etage,
01481         &position_x_initiale, &position_y_initiale, &position_x, &position_y,
01482         &largeur, &hauteur, &largeur_tile, &hauteur_tile, &page_active,
01483         itemsDemandeSauvegarde, &touche_descendre, barre_de_son, &pseudo,
01484         &personnageActif, &positionActive, tailleNiveaux,
01485         temps_debut_partie, &compteur_mort, avancee_succes,
01486         avancee_succes_intermediaires);
01487
01488     /* Retour sur la carte */
01489     if(page_active == CARTE) {
01490
01491         if(!avancee_niveaux[0].niveau_fini)
01492             chargement_image(&renderer, &surface, &texture_image_carte,
01493             "./images/carte_niveau_2_bloque.jpg");
01494
01495         else if(!avancee_niveaux[1].niveau_fini)
01496             chargement_image(&renderer, &surface, &texture_image_carte,
01497             "./images/carte_niveau_3_bloque.jpg");
01498
01499         else if(!avancee_niveaux[2].niveau_fini)
01500             chargement_image(&renderer, &surface, &texture_image_carte,
01501             "./images/carte_niveau_4_bloque.jpg");
01502
01503         else
01504             chargement_image(&renderer, &surface, &texture_image_carte, "./images/carte.jpg");
01505
01506         direction = BAS;
01507     }
01508 }

```

```

01501             /* Musique de la carte */
01502             if((musique = Mix_LoadMUS("./sons/musiques/carte.mp3")) == NULL)
01503                 erreur("Chargement de la musique");
01504
01505             Mix_PlayMusic(musique, -1);
01506         }
01507     }
01508 }
01509
01510 /*----- Fin du jeu -----*/
01511
01512 /* Libération de la mémoire allouée dynamiquement */
01513 free(itemsMenuPrincipal);
01514
01515 /* Destruction des différents objets */
01516 detruire_objets(&police, &musique, &texture_image_plein_ecran, &texture_image_retour_en_arriere,
01517               &texture_image_options, &texture_image_passer, &texture_image_menu,
01518               &texture_image_carte,
01519               &texture_image_hautParleurActif, &texture_image_hautParleurDesactive,
01520               &texture_image_perso_1, &texture_image_perso_2, &texture_image_perso_1_bas_1,
01521               &texture_image_perso_1_bas_2, &texture_image_perso_1_haut_1,
01522               &texture_image_perso_1_haut_2,
01523               &texture_image_perso_1_bas_gauche_1, &texture_image_perso_1_bas_gauche_2,
01524               &texture_image_perso_2_bas_1, &texture_image_perso_2_bas_2,
01525               &texture_image_perso_2_haut_1, &texture_image_perso_2_haut_2,
01526               &texture_image_perso_2_bas_gauche_1, &texture_image_perso_2_bas_gauche_2,
01527               &texture_image_perso_1_haut, &texture_image_perso_1_droite,
01528               &texture_image_perso_1_gauche, &texture_image_perso_1_pose,
01529               &texture_image_perso_2_haut, &texture_image_perso_2_droite,
01530               &texture_image_perso_2_gauche, &texture_image_perso_2_pose,
01531               &texture_texte, &texture_image_fond_niveau_2,
01532               &texture_image_dossier_niveau_2, &texture_image_sol_niveau_2,
01533               &texture_image_fond_niveau_3, &texture_image_dossier_niveau_3,
01534               &texture_image_sol_niveau_3, &texture_image_mur, &texture_image_fond_niveau_4,
01535               &texture_image_bordure_niveau_4, &texture_image_porte, &texture_image_pique,
01536               &texture_image_fin_premiers_niveaux, &texture_image_fin_dernier_niveau,
01537               &texture_image_fond_niveau_1, &texture_image_sol_surface_niveau_1,
01538               &texture_image_sol_profondeur_niveau_1, &texture_image_monstre_terrestre,
01539               &texture_image_monstre_volant, &barre_windows_1, &barre_windows_2,
01540               &barre_windows_3, &barre_windows_4, &texture_image_mur_mini_jeu,
01541               &texture_image_pipe_vertical, &texture_image_pipe_horizontal,
01542               &texture_image_pipe_haut_droit, &texture_image_pipe_bas_droit,
01543               &texture_image_pipe_bas_gauche, &texture_image_pipe_haut_gauche,
01544               &texture_image_pipe_courant, &texture_image_mur_termine,
01545               &texture_image_perso_1_gagnant, &texture_image_perso_2_gagnant,
01546               &texture_image_puzzle, &texture_image_retour_menu,
01547               &texture_image_sol_labyrinthe, &texture_image_bordure_labyrinthe,
01548               &texture_image_fin_labyrinthe, &texture_image_nuage_1,
01549               &texture_image_nuage_2, &texture_image_croix,
01550               textures_images_succes);
01551
01552 /* Destruction du rendu et de la fenêtre*/
01553 detruire_fenetre_rendu(&render, &window);
01554
01555 /* Quitter TTF, VIDEO SDL et AUDIO SDL */
01556 TTF_Quit();
01557 SDL_Quit();
01558 Mix_CloseAudio();
01559
01560 return EXIT_SUCCESS; /* return 0; */
01561 }

```

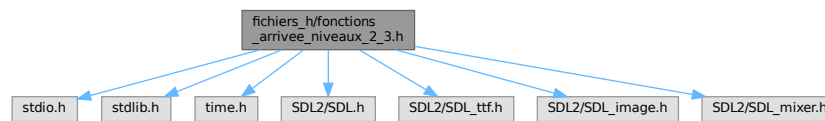
## 4.25 Référence du fichier fichiers\_h/fonctions\_arrivee\_niveaux\_2\_3.h

```

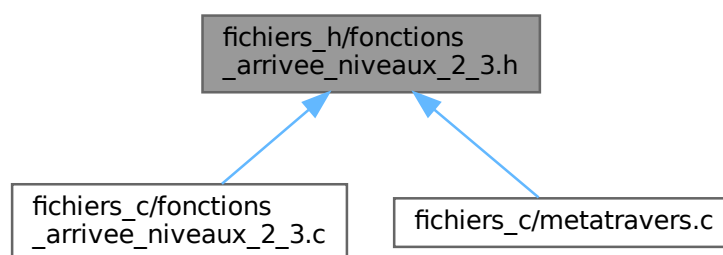
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

```

Graphe des dépendances par inclusion de fonctions\_arrivee\_niveaux\_2\_3.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [salon\\_arrivee\\_niveaux\\_2\\_3](#) (int \*position\_x, int \*position\_y, int tile\_map[18][32], [page\\_t](#) page\_active)  
*Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.*
- void [mise\\_a\\_jour\\_rendu\\_arrivee\\_niveaux\\_2\\_3](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, SDL\_Texture \*\*texture\_image\_fin\_plein\_ecran, SDL\_Rect \*rectangle\_tile, SDL\_Texture \*\*texture\_image\_dossier, SDL\_Texture \*\*barre\_windows\_1, SDL\_Texture \*\*barre\_windows\_2, SDL\_Texture \*\*barre\_windows\_3, SDL\_Texture \*\*barre\_windows\_4, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int position\_x, int position\_y, int tile\_map[18][32], [niveaux](#) \*avancee\_niveaux, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile, [page\\_t](#) page\_active)
- void [explications](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_explications, SDL\_Keycode touche\_interagir, SDL\_Keycode touche\_sauter\_monter, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleur, [itemMenu](#) \*itemsExplications, int largeur, int hauteur, int numero\_mini\_jeu)
- void [arrivee\\_niveaux\\_2\\_3](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, int \*mini\_jeu, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Surface \*\*surface, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int \*mini\_jeu\_termine, int \*mini\_jeu\_1\_termine, int \*mini\_jeu\_2\_termine, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_interagir, SDL\_Texture \*\*texture\_image\_porte, [niveaux](#) \*avancee\_niveaux, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, SDL\_Texture \*\*texture\_image\_dossier, SDL\_Texture \*\*barre\_windows\_1, SDL\_Texture \*\*barre\_windows\_2, SDL\_Texture \*\*barre\_windows\_3, SDL\_Texture \*\*barre\_windows\_4, int tile\_map[18][32], SDL\_Rect

```

*rectangle_tile, int *mouvement_monstre, modes\_t *modeActif, int *mode_difficile, itemMenu *items↵
DemandeQuitter, int tailleDemande, SDL_Color couleurNoire, int tile_map_mini_jeu_niveau_2[19][27],
SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande, time_t *timestamp,
SDL_Texture **texture_image_perso_gagnant, int *avancer, int *reculer, int *sauter, int *position↵
_avant_saut, int *saut, int *tombe, int *position_x_initiale, int *position_y_initiale, int *position_x, int
*position_y, int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page\_t *page_active, SDL_↵
Texture **texture_image_mur_mini_jeu, int collectibles_intermediaires[3], itemMenu *itemsExplications,
SDL_Texture **texture_image_pipe_vertical, SDL_Texture **texture_image_pipe_horizontal, SDL_Texture
**texture_image_pipe_haut_droit, SDL_Texture **texture_image_pipe_bas_droit, SDL_Texture **texture↵
_image_pipe_bas_gauche, SDL_Texture **texture_image_pipe_haut_gauche, SDL_Texture **texture↵
_image_pipe_courant, SDL_Texture **texture_image_mur_termine, int *valide, SDL_Rect rectangle_↵
piece[45], int piece_bloquee[45], SDL_Rect rectangle_emplacement_piece[45], int *piece_selectionnee,
int *decalage_x, int *decalage_y, SDL_Texture **texture_image_puzzle, Mix_Music **musique, SDL_↵
_Texture **texture_image_croix, SDL_Rect *rectangle_croix, int tile_map_mini_jeu_niveau_3[24][32], int
*descendre, int *interagir, int *bloc_x, int *bloc_y, SDL_Texture **texture_image_sol_labyrinthe, SDL_↵
_Texture **texture_image_bordure_labyrinthe, SDL_Texture **texture_image_fin_labyrinthe, SDL_Color
couleurTitre, itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
personnage\_t *personnageActif, position\_t *positionActive, int tailleNiveaux, time_t temps_debut_partie, int
*compteur_mort, int *avancee_succes, int avancee_succes_intermediaires[10])

```

## 4.25.1 Documentation des fonctions

### 4.25.1.1 arrivee\_niveaux\_2\_3()

```

void arrivee_niveaux_2_3 (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    int * mini_jeu,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    SDL_Texture ** texture,
    SDL_Surface ** surface,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int * mini_jeu_termine,
    int * mini_jeu_1_termine,
    int * mini_jeu_2_termine,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_interagir,
    SDL_Texture ** texture_image_porte,
    niveaux * avancee_niveaux,
    SDL_Keycode * touche_sauter_monter,
    SDL_Keycode * touche_descendre,
    SDL_Texture ** texture_image_dossier,
    SDL_Texture ** barre_windows_1,
    SDL_Texture ** barre_windows_2,
    SDL_Texture ** barre_windows_3,

```

```
SDL_Texture ** barre_windows_4,
int tile_map[18][32],
SDL_Rect * rectangle_tile,
int * mouvement_monstre,
modes_t * modeActif,
int * mode_difficile,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
SDL_Color couleurNoire,
int tile_map_mini_jeu_niveau_2[19][27],
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Rect * rectangle_demande,
time_t * timestamp,
SDL_Texture ** texture_image_perso_gagnant,
int * avancer,
int * reculer,
int * sauter,
int * position_avant_saut,
int * saut,
int * tombe,
int * position_x_initiale,
int * position_y_initiale,
int * position_x,
int * position_y,
int * largeur,
int * hauteur,
int * largeur_tile,
int * hauteur_tile,
page_t * page_active,
SDL_Texture ** texture_image_mur_mini_jeu,
int collectibles_intermediaires[3],
itemMenu * itemsExplications,
SDL_Texture ** texture_image_pipe_vertical,
SDL_Texture ** texture_image_pipe_horizontal,
SDL_Texture ** texture_image_pipe_haut_droit,
SDL_Texture ** texture_image_pipe_bas_droit,
SDL_Texture ** texture_image_pipe_bas_gauche,
SDL_Texture ** texture_image_pipe_haut_gauche,
SDL_Texture ** texture_image_pipe_courant,
SDL_Texture ** texture_image_mur_termine,
int * valide,
SDL_Rect rectangle_piece[45],
int piece_bloquee[45],
SDL_Rect rectangle_emplacement_piece[45],
int * piece_selectionnee,
int * decalage_x,
int * decalage_y,
SDL_Texture ** texture_image_puzzle,
Mix_Music ** musique,
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
int tile_map_mini_jeu_niveau_3[24][32],
int * descendre,
int * interagir,
int * bloc_x,
int * bloc_y,
SDL_Texture ** texture_image_sol_labyrinthe,
```

```

SDL_Texture ** texture_image_bordure_labyrinthe,
SDL_Texture ** texture_image_fin_labyrinthe,
SDL_Color couleurTitre,
itemMenu * itemsDemandeSauvegarde,
barreDeSon * barre_de_son,
itemMenu * pseudo,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 482 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

#### 4.25.1.2 explications()

```

void explications (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_explications,
    SDL_Keycode touche_interagir,
    SDL_Keycode touche_sauter_monter,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleur,
    itemMenu * itemsExplications,
    int largeur,
    int hauteur,
    int numero_mini_jeu )

```

Définition à la ligne 269 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

#### 4.25.1.3 mise\_a\_jour\_rendu\_arrivee\_niveaux\_2\_3()

```

void mise_a_jour_rendu_arrivee_niveaux_2_3 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Texture ** texture_image_dossier,
    SDL_Texture ** barre_windows_1,
    SDL_Texture ** barre_windows_2,
    SDL_Texture ** barre_windows_3,
    SDL_Texture ** barre_windows_4,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int position_x,
    int position_y,
    int tile_map[18][32],

```



```

niveaux * avancee_niveaux,
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
int largeur,
int hauteur,
int largeur_tile,
int hauteur_tile,
page_t page_active )

```

Définition à la ligne 125 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

#### 4.25.1.4 salon\_arrivee\_niveaux\_2\_3()

```

void salon_arrivee_niveaux_2_3 (
    int * position_x,
    int * position_y,
    int tile_map[18][32],
    page_t page_active )

```

Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.

##### Paramètres

<i>position_x</i>	pointeur sur la position du personnage sur l'horizontal du tilemap
<i>position_y</i>	pointeur sur la position du perosnnage sur la verticale du tilemap
<i>tile_map</i>	Matrice représentant la map ou se trouve le personnage
<i>page_active</i>	Enumération représentant sur quel page on se trouve

Définition à la ligne 21 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

## 4.26 fonctions\_arrivee\_niveaux\_2\_3.h

[Aller à la documentation de ce fichier.](#)

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <time.h>
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_ttf.h>
00006 #include <SDL2/SDL_image.h>
00007 #include <SDL2/SDL_mixer.h>
00008
00009 /* Squelette de la fonction salon_arrivee_niveaux_2_3 */
00010 void salon_arrivee_niveaux_2_3(int *position_x, int *position_y, int tile_map[18][32], page_t
    page_active);
00011
00012 /* Squelette de la fonction mise_a_jour_rendu_arrivee_niveaux_2_3 */
00013 void mise_a_jour_rendu_arrivee_niveaux_2_3(SDL_Renderer **renderer, SDL_Texture **texture_image_fond,
    SDL_Texture **texture_image_sol,
00014     SDL_Rect *rectangle_plein_ecran, SDL_Texture
    **texture_image_plein_ecran, SDL_Texture **texture_image_fin_premiers_niveaux,
00015     SDL_Texture **texture, SDL_Rect *rectangle_tile,
    SDL_Texture **texture_image_dossier,
00016     SDL_Texture **barre_windows_1, SDL_Texture
    **barre_windows_2, SDL_Texture **barre_windows_3,
00017     SDL_Texture **barre_windows_4, SDL_Texture
    **texture_image_personnage, SDL_Rect *rectangle_personnage,
00018     int position_x, int position_y, int tile_map[18][32],
    niveaux *avancee_niveaux, SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix,
00019     int largeur, int hauteur, int largeur_tile, int
    hauteur_tile, page_t page_active);
00020

```

```

00021 /* Squelette de la fonction explications */
00022 void explications(SDL_Renderer **renderer, SDL_Rect *rectangle_explications, SDL_Keycode
00023     touche_interagir, SDL_Keycode touche_sauter_monter,
00024     SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, SDL_Color
00025     couleur,
00026     itemMenu *itemsExplications, int largeur, int hauteur, int numero_mini_jeu);
00027 /* Squelette de la fonction arrivee_niveaux_2_3 */
00028 void arrivee_niveaux_2_3(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
00029     *programme_lance, int *mini_jeu, SDL_Texture **texture_image_fin_premiers_niveaux,
00030     SDL_Texture **texture, SDL_Surface **surface, SDL_Rect
00031     *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool *plein_ecran,
00032     SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int
00033     *mini_jeu_termine, int *mini_jeu_1_termine, int *mini_jeu_2_termine,
00034     SDL_Texture **texture_image_fond, SDL_Texture **texture_image_sol,
00035     SDL_Texture **texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant,
00036     SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
00037     SDL_Keycode *touche_interagir, SDL_Texture **texture_image_porte, niveaux *avancee_niveaux,
00038     SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Texture
00039     **texture_image_dossier,
00040     SDL_Texture **barre_windows_1, SDL_Texture **barre_windows_2, SDL_Texture
00041     **barre_windows_3,
00042     SDL_Texture **barre_windows_4, int tile_map[18][32], SDL_Rect
00043     *rectangle_tile, int *mouvement_monstre, modes_t *modeActif, int *mode_difficile,
00044     itemMenu *itemsDemandeQuitter, int tailleDemande, SDL_Color couleurNoire, int
00045     tile_map_mini_jeu_niveau_2[19][27],
00046     SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande,
00047     time_t *timestamp, SDL_Texture **texture_image_perso_gagnant,
00048     int *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut,
00049     int *tombe,
00050     int *position_x_initiale, int *position_y_initiale, int *position_x, int
00051     *position_y,
00052     int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page_t
00053     *page_active,
00054     SDL_Texture **texture_image_mur_mini_jeu, int collectibles_intermediaires[3],
00055     itemMenu *itemsExplications,
00056     SDL_Texture **texture_image_pipe_vertical, SDL_Texture
00057     **texture_image_pipe_horizontal,
00058     SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
00059     **texture_image_pipe_bas_droit,
00060     SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
00061     **texture_image_pipe_haut_gauche,
00062     SDL_Texture **texture_image_pipe_courant, SDL_Texture
00063     **texture_image_mur_termine, int *valide,
00064     SDL_Rect rectangle_piece[45], int piece_bloquee[45], SDL_Rect
00065     rectangle_emplacement_piece[45],
00066     int *piece_selectionnee, int *decalage_x, int *decalage_y, SDL_Texture
00067     **texture_image_puzzle, Mix_Music **musique, SDL_Texture **texture_image_croix, SDL_Rect
00068     *rectangle_croix,
00069     int tile_map_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int
00070     *bloc_x, int *bloc_y,
00071     SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture
00072     **texture_image_bordure_labyrinthe,
00073     SDL_Texture **texture_image_fin_labyrinthe, SDL_Color couleurTitre,
00074     itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
00075     personnage_t *personnageActif, position_t *positionActive, int tailleNiveaux,
00076     time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
00077     avancee_succes_intermediaires[10]);

```

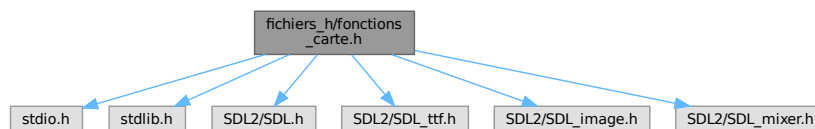
## 4.27 Référence du fichier fichiers\_h/fonctions\_carte.h

```

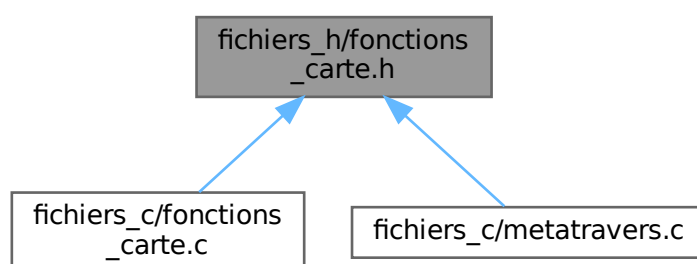
#include <stdio.h>
#include <stdlib.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

```

Graphe des dépendances par inclusion de fonctions\_carte.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [initialisation\\_objets\\_carte](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_carte, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_1, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_2, SDL\_Texture \*\*texture\_image\_perso\_1\_haut\_1, SDL\_Texture \*\*texture\_image\_perso\_1\_haut\_2, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_gauche\_1, SDL\_Texture \*\*texture\_image\_perso\_1\_bas\_gauche\_2, SDL\_Texture \*\*texture\_image\_perso\_1\_haut, SDL\_Texture \*\*texture\_image\_perso\_1\_droite, SDL\_Texture \*\*texture\_image\_perso\_1\_gauche, SDL\_Texture \*\*texture\_image\_perso\_1\_pose, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_1, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_2, SDL\_Texture \*\*texture\_image\_perso\_2\_haut\_1, SDL\_Texture \*\*texture\_image\_perso\_2\_haut\_2, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_gauche\_1, SDL\_Texture \*\*texture\_image\_perso\_2\_bas\_gauche\_2, SDL\_Texture \*\*texture\_image\_perso\_2\_haut, SDL\_Texture \*\*texture\_image\_perso\_2\_droite, SDL\_Texture \*\*texture\_image\_perso\_2\_gauche, SDL\_Texture \*\*texture\_image\_perso\_2\_pose, [itemMenu](#) \*itemsNiveaux, SDL\_Texture \*\*texture\_image\_retour\_menu, [itemMenu](#) \*itemsSucces, SDL\_Texture \*\*textures\_images\_succes)
- void [mise\\_a\\_jour\\_rendu\\_carte](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_carte, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, SDL\_Rect \*rectangle\_perso, SDL\_Texture \*\*texture\_image\_perso, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Rect \*rectangle\_succes, [position\\_t](#) positionActive, SDL\_Color couleurNoire, SDL\_Rect \*rectangle\_retour\_menu, SDL\_Texture \*\*texture\_image\_retour\_menu, [itemMenu](#) \*itemsNiveaux, int tailleNiveaux, int largeur, int hauteur, [niveau](#) \*avancee\_niveaux)
- void [deplacement\\_personnage\\_carte](#) (SDL\_Renderer \*\*renderer, SDL\_Window \*\*window, SDL\_Texture \*\*texture\_image\_carte, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, SDL\_Rect \*rectangle\_perso, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Rect

```

*rectangle_succes, position\_t *positionActive, SDL_Color couleurNoire, SDL_Rect *rectangle_retour_↵
menu, SDL_Texture **texture_image_retour_menu, itemMenu *itemsNiveaux, int tailleNiveaux, int largeur,
int hauteur, int valeur_maximale, direction\_t direction, niveaux *avancee_niveaux)
— void carte(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool *programme_↵
lance, SDL_Texture **texture_image_carte, SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_↵
image_plein_ecran, SDL_bool *plein_ecran, SDL_Rect *rectangle_options, SDL_Texture **texture_image_↵
_options, SDL_Rect *rectangle_retour_menu, SDL_Texture **texture_image_retour_menu, SDL_Texture
**texture_image_perso_bas_1, SDL_Texture **texture_image_perso_bas_2, SDL_Texture **texture_↵
_image_perso_haut_1, SDL_Texture **texture_image_perso_haut_2, SDL_Texture **texture_image_↵
perso_bas_gauche_1, SDL_Texture **texture_image_perso_bas_gauche_2, SDL_Texture **texture_↵
image_perso_haut, SDL_Texture **texture_image_perso_droite, SDL_Texture **texture_image_perso_↵
_gauche, SDL_Texture **texture_image_perso_pose, SDL_Texture **texture_image_perso, SDL_Rect
*rectangle_perso, niveaux *avancee_niveaux, int niveau_fini[4], int collectibles[12], position\_t *position_↵
intermediaire, SDL_Texture **texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes, SDL_↵
Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, direction\_t *direction, int *touche_↵
pressee, SDL_Rect *rectangle_demande_sauvegarde, itemMenu *itemsDemandeSauvegarde, int taille_↵
DemandeSauvegarde, position\_t *positionActive, barreDeSon *barre_de_son, itemMenu *pseudo, modes\_t
*modeActif, personnage\_t *personnageActif, SDL_Color couleurNoire, SDL_Keycode *touche_aller_↵
a_droite, SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter, SDL_Keycode
*touche_descendre, SDL_Keycode *touche_interagir, itemMenu *itemsNiveaux, int tailleNiveaux, int
*largeur, int *hauteur, page\_t *page_active, itemMenu *itemsSucces, SDL_Texture **textures_images_↵
succes, time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int avancee_succes_↵
intermediaires[10])

```

## 4.27.1 Documentation des fonctions

### 4.27.1.1 `carte()`

```

void carte (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_carte,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    SDL_Rect * rectangle_retour_menu,
    SDL_Texture ** texture_image_retour_menu,
    SDL_Texture ** texture_image_perso_bas_1,
    SDL_Texture ** texture_image_perso_bas_2,
    SDL_Texture ** texture_image_perso_haut_1,
    SDL_Texture ** texture_image_perso_haut_2,
    SDL_Texture ** texture_image_perso_bas_gauche_1,
    SDL_Texture ** texture_image_perso_bas_gauche_2,
    SDL_Texture ** texture_image_perso_haut,
    SDL_Texture ** texture_image_perso_droite,
    SDL_Texture ** texture_image_perso_gauche,
    SDL_Texture ** texture_image_perso_pose,
    SDL_Texture ** texture_image_perso,
    SDL_Rect * rectangle_perso,
    niveaux * avancee_niveaux,
    int niveau_fini[4],
    int collectibles[12],

```

```

position_t * position_intermediaire,
SDL_Texture ** texture_image_fin_dernier_niveau,
SDL_Rect * rectangle_succes,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
direction_t * direction,
int * touche_pressee,
SDL_Rect * rectangle_demande_sauvegarde,
itemMenu * itemsDemandeSauvegarde,
int tailleDemandeSauvegarde,
position_t * positionActive,
barreDeSon * barre_de_son,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
SDL_Color couleurNoire,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
SDL_Keycode * touche_interagir,
itemMenu * itemsNiveaux,
int tailleNiveaux,
int * largeur,
int * hauteur,
page_t * page_active,
itemMenu * itemsSucces,
SDL_Texture ** textures_images_succes,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 539 du fichier fonctions\_carte.c.

#### 4.27.1.2 `deplacement_personnage_carte()`

```

void deplacement_personnage_carte (
    SDL_Renderer ** renderer,
    SDL_Window ** window,
    SDL_Texture ** texture_image_carte,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    SDL_Rect * rectangle_perso,
    SDL_Texture ** texture_image_perso_1,
    SDL_Texture ** texture_image_perso_2,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Texture ** texture_image_fin_dernier_niveau,
    SDL_Rect * rectangle_succes,
    position_t * positionActive,
    SDL_Color couleurNoire,
    SDL_Rect * rectangle_retour_menu,

```

```

SDL_Texture ** texture_image_retour_menu,
itemMenu * itemsNiveaux,
int tailleNiveaux,
int largeur,
int hauteur,
int valeur_maximale,
direction_t direction,
niveaux * avancee_niveaux )

```

Définition à la ligne 376 du fichier [fonctions\\_carte.c](#).

#### 4.27.1.3 initialisation\_objets\_carte()

```

void initialisation_objets_carte (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_carte,
    SDL_Texture ** texture_image_perso_1_bas_1,
    SDL_Texture ** texture_image_perso_1_bas_2,
    SDL_Texture ** texture_image_perso_1_haut_1,
    SDL_Texture ** texture_image_perso_1_haut_2,
    SDL_Texture ** texture_image_perso_1_bas_gauche_1,
    SDL_Texture ** texture_image_perso_1_bas_gauche_2,
    SDL_Texture ** texture_image_perso_1_haut,
    SDL_Texture ** texture_image_perso_1_droite,
    SDL_Texture ** texture_image_perso_1_gauche,
    SDL_Texture ** texture_image_perso_1_pose,
    SDL_Texture ** texture_image_perso_2_bas_1,
    SDL_Texture ** texture_image_perso_2_bas_2,
    SDL_Texture ** texture_image_perso_2_haut_1,
    SDL_Texture ** texture_image_perso_2_haut_2,
    SDL_Texture ** texture_image_perso_2_bas_gauche_1,
    SDL_Texture ** texture_image_perso_2_bas_gauche_2,
    SDL_Texture ** texture_image_perso_2_haut,
    SDL_Texture ** texture_image_perso_2_droite,
    SDL_Texture ** texture_image_perso_2_gauche,
    SDL_Texture ** texture_image_perso_2_pose,
    itemMenu * itemsNiveaux,
    SDL_Texture ** texture_image_retour_menu,
    itemMenu * itemsSucces,
    SDL_Texture ** textures_images_succes )

```

Définition à la ligne 39 du fichier [fonctions\\_carte.c](#).

#### 4.27.1.4 mise\_a\_jour\_rendu\_carte()

```

void mise_a_jour_rendu_carte (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_carte,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    SDL_Rect * rectangle_perso,
    SDL_Texture ** texture_image_perso,

```

```

    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Texture ** texture_image_fin_dernier_niveau,
    SDL_Rect * rectangle_succes,
    position_t positionActive,
    SDL_Color couleurNoire,
    SDL_Rect * rectangle_retour_menu,
    SDL_Texture ** texture_image_retour_menu,
    itemMenu * itemsNiveaux,
    int tailleNiveaux,
    int largeur,
    int hauteur,
    niveaux * avancee_niveaux )

```

Définition à la ligne 141 du fichier [fonctions\\_carte.c](#).

## 4.28 fonctions\_carte.h

Aller à la documentation de ce fichier.

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <SDL2/SDL.h>
00004 #include <SDL2/SDL_ttf.h>
00005 #include <SDL2/SDL_image.h>
00006 #include <SDL2/SDL_mixer.h>
00007
00008 /* Squelette de la fonction initialisation_objets_carte */
00009 void initialisation_objets_carte(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_carte,
00010                                SDL_Texture **texture_image_perso_1_bas_1, SDL_Texture
**texture_image_perso_1_bas_2,
00011                                SDL_Texture **texture_image_perso_1_haut_1, SDL_Texture
**texture_image_perso_1_haut_2,
00012                                SDL_Texture **texture_image_perso_1_bas_gauche_1, SDL_Texture
**texture_image_perso_1_bas_gauche_2,
00013                                SDL_Texture **texture_image_perso_1_haut, SDL_Texture
**texture_image_perso_1_droite,
00014                                SDL_Texture **texture_image_perso_1_gauche, SDL_Texture
**texture_image_perso_1_pose,
00015                                SDL_Texture **texture_image_perso_2_bas_1, SDL_Texture
**texture_image_perso_2_bas_2,
00016                                SDL_Texture **texture_image_perso_2_haut_1, SDL_Texture
**texture_image_perso_2_haut_2,
00017                                SDL_Texture **texture_image_perso_2_bas_gauche_1, SDL_Texture
**texture_image_perso_2_bas_gauche_2,
00018                                SDL_Texture **texture_image_perso_2_haut, SDL_Texture
**texture_image_perso_2_droite,
00019                                SDL_Texture **texture_image_perso_2_gauche, SDL_Texture
**texture_image_perso_2_pose,
00020                                itemMenu *itemsNiveaux, SDL_Texture **texture_image_retour_menu,
00021                                itemMenu *itemsSucces, SDL_Texture **textures_images_succes);
00022
00023 /* Squelette de la fonction mise_a_jour_rendu_carte */
00024 void mise_a_jour_rendu_carte(SDL_Renderer **renderer, SDL_Texture **texture_image_carte,
00025                               SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
00026                               SDL_Rect *rectangle_options, SDL_Texture **texture_image_options,
00027                               SDL_Rect *rectangle_perso, SDL_Texture **texture_image_perso,
00028                               SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00029                               SDL_Texture **texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes,
00030                               position_t positionActive, SDL_Color couleurNoire, SDL_Rect
*rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
00031                               itemMenu *itemsNiveaux, int tailleNiveaux, int largeur, int hauteur,
00032                               niveaux *avancee_niveaux);
00033
00034 /* Squelette de la fonction deplacement_personnage_carte */
00035 void deplacement_personnage_carte(SDL_Renderer **renderer, SDL_Window **window, SDL_Texture
**texture_image_carte,
00036                                SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00037                                SDL_Rect *rectangle_options, SDL_Texture **texture_image_options,
00038                                SDL_Rect *rectangle_perso, SDL_Texture **texture_image_perso_1,
00039                                SDL_Texture **texture_image_perso_2,

```

```

00037             SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font
**police, SDL_Texture **texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes,
00038             position_t *positionActive, SDL_Color couleurNoire, SDL_Rect
*rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
00039             itemMenu *itemsNiveaux, int tailleNiveaux, int largeur, int hauteur,
00040             int valeur_maximale, direction_t direction, niveaux
*avancee_niveaux);
00041
00042 /* Squelette de la fonction carte */
00043 void carte(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool *programme_lance,
SDL_Texture **texture_image_carte,
00044             SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
*plein_ecran,
00045             SDL_Rect *rectangle_options, SDL_Texture **texture_image_options, SDL_Rect
*rectangle_retour_menu, SDL_Texture **texture_image_retour_menu,
00046             SDL_Texture **texture_image_perso_bas_1, SDL_Texture **texture_image_perso_bas_2,
00047             SDL_Texture **texture_image_perso_haut_1, SDL_Texture **texture_image_perso_haut_2,
00048             SDL_Texture **texture_image_perso_bas_gauche_1, SDL_Texture
**texture_image_perso_bas_gauche_2,
00049             SDL_Texture **texture_image_perso_haut, SDL_Texture **texture_image_perso_droite,
00050             SDL_Texture **texture_image_perso_gauche, SDL_Texture **texture_image_perso_pose,
00051             SDL_Texture **texture_image_perso, SDL_Rect *rectangle_perso, niveaux *avancee_niveaux,
00052             int niveau_fini[4], int collectibles[12], position_t *position_intermediaire, SDL_Texture
**texture_image_fin_dernier_niveau, SDL_Rect *rectangle_succes,
00053             SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police, direction_t
*direction, int *touche_pressee,
00054             SDL_Rect *rectangle_demande_sauvegarde, itemMenu *itemsDemandeSauvegarde, int
tailleDemandeSauvegarde,
00055             position_t *positionActive, barreDeSon *barre_de_son, itemMenu *pseudo, modes_t *modeActif,
personnage_t *personnageActif,
00056             SDL_Color couleurNoire, SDL_Keycode *touche_aller_a_droite, SDL_Keycode
*touche_aller_a_gauche,
00057             SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, SDL_Keycode
*touche_interagir,
00058             itemMenu *itemsNiveaux, int tailleNiveaux, int *largeur, int *hauteur, page_t *page_active,
00059             itemMenu *itemsSucces, SDL_Texture **textures_images_succes,
00060             time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
avancee_succes_intermediaires[10]);

```

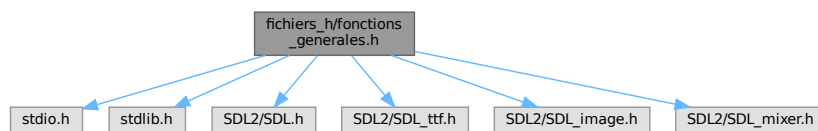
## 4.29 Référence du fichier fichiers\_h/fonctions\_generales.h

```

#include <stdio.h>
#include <stdlib.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

```

Graphe des dépendances par inclusion de fonctions\_generales.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :





## Classes

- struct [itemMenu](#)
- struct [barreDeSon](#)
- struct [niveaux](#)

## Définitions de type

- typedef enum [option\\_s](#) [option\\_t](#)
- typedef enum [modes\\_s](#) [modes\\_t](#)
- typedef enum [personnage\\_s](#) [personnage\\_t](#)
- typedef enum [page\\_s](#) [page\\_t](#)
- typedef enum [position\\_s](#) [position\\_t](#)
- typedef enum [direction\\_s](#) [direction\\_t](#)

## Énumérations

- enum [option\\_s](#) { [ONGLET\\_SON](#) , [ONGLET\\_TOUCHES](#) }
- enum [modes\\_s](#) { [MODE\\_NORMAL](#) , [MODE\\_HARD](#) }
- enum [personnage\\_s](#) { [PERSONNAGE\\_1](#) , [PERSONNAGE\\_2](#) }
- enum [page\\_s](#) {  
[MENU\\_PRINCIPAL](#) , [OPTIONS](#) , [NOUVELLE\\_PARTIE](#) , [INTRODUCTION](#) ,  
[CARTE](#) , [NIVEAU\\_1](#) , [NIVEAU\\_2](#) , [NIVEAU\\_3](#) ,  
[NIVEAU\\_4](#) }
- enum [position\\_s](#) {  
[NIVEAU0](#) , [NIVEAU1](#) , [NIVEAU2](#) , [NIVEAU3](#) ,  
[NIVEAU4](#) }
- enum [direction\\_s](#) {  
[HAUT](#) , [BAS](#) , [GAUCHE](#) , [DROITE](#) ,  
[HAUT\\_DROITE](#) , [BAS\\_GAUCHE](#) }

## Fonctions

- void [erreur](#) (const char \*message)  
*Affiche l'erreur en cas de problème et ferme la SDL.*
- void [chargement\\_image](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture, char \*chemin)  
*Fonction qui permet de charger une image.*
- void [affichage\\_texte](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture, [itemMenu](#) \*item, TTF\_Font \*\*police, SDL\_Color couleur)
- void [creer\\_fenetre\\_rendu](#) (SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, int largeur, int hauteur)  
*Fonction qui permet de créer une fenêtre et le rendu.*
- void [initialisation\\_objets](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture ↵  
image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image ↵  
options, SDL\_Texture \*\*texture\_image\_passer, [itemMenu](#) \*itemsDemandeSauvegarde, [itemMenu](#) \*items ↵  
DemandeQuitter, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, SDL\_Texture \*\*texture\_image ↵  
\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image ↵  
perso\_1\_gagnant, SDL\_Texture \*\*texture\_image\_perso\_2\_gagnant, [niveaux](#) \*avancee\_niveaux, TTF\_Font ↵  
\*\*police, SDL\_Texture \*\*texture\_image\_croix)
- void [demande\\_sauvegarde](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_demande\_sauvegarde, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleur, [itemMenu](#) \*itemsDemandeSauvegarde, int tailleDemandeSauvegarde, int largeur, int hauteur)  
*fenêtre se chargeant de demander à l'utilisateur si il souhaite sauvegarder*
- void [demande\\_quitter\\_niveau](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_demande\_quitter, SDL ↵  
\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleur, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemandeQuitter, int largeur, int hauteur)

- Fonction qui permet de demander à l'utilisateur de quitter le niveau.*
- void `redimensionnement_fenetre` (SDL\_Event event, int \*largeur, int \*hauteur)  
*Fonction qui permet de récupérer les nouvelles dimensions de la fenêtre pour redimensionner cette dernière et les différents objets.*
  - int `verification_sauvegarde` ()  
*Vérifie si une sauvegarde existe.*
  - void `sauvegarder_partie` (SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_↵ interagir, `barreDeSon` \*barre\_de\_son, `itemMenu` \*pseudo, `modes_t` modeActif, `personnage_t` personnage\_↵ Actif, `position_t` positionActive, `niveaux` \*avancee\_niveaux, int tailleNiveaux, time\_t temps\_debut\_partie, int compteur\_mort, int avancee\_succes[10])
  - int `clic_case` (SDL\_Event event, SDL\_Rect rectangle)  
*Fonction qui permet de renvoyer vrai quand on clique sur un rectangle, faux sinon.*
  - void `deplacement_personnage` (int \*saut, int \*tombe, int \*position\_x, int \*position\_y, int \*position\_avant\_↵ saut, int sauter, int avancer, int reculer, int tile\_map[18][32], `personnage_t` personnageActif)
  - int `clic_plein_ecran` (SDL\_Event event, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_↵ Window \*\*window)  
*Fonction qui permet de mettre la fenêtre en plein écran quand on clique sur le bouton plein écran.*
  - void `destruire_objets` (TTF\_Font \*\*police, Mix\_Music \*\*musique, SDL\_Texture \*\*texture1, SDL\_Texture \*\*texture2, SDL\_Texture \*\*texture3, SDL\_Texture \*\*texture4, SDL\_Texture \*\*texture5, SDL\_Texture \*\*texture6, SDL\_Texture \*\*texture7, SDL\_Texture \*\*texture8, SDL\_Texture \*\*texture9, SDL\_Texture \*\*texture10, SDL\_Texture \*\*texture11, SDL\_Texture \*\*texture12, SDL\_Texture \*\*texture13, SDL\_Texture \*\*texture14, SDL\_Texture \*\*texture15, SDL\_Texture \*\*texture16, SDL\_Texture \*\*texture17, SDL\_Texture \*\*texture18, SDL\_Texture \*\*texture19, SDL\_Texture \*\*texture20, SDL\_Texture \*\*texture21, SDL\_Texture \*\*texture22, SDL\_Texture \*\*texture23, SDL\_Texture \*\*texture24, SDL\_Texture \*\*texture25, SDL\_Texture \*\*texture26, SDL\_Texture \*\*texture27, SDL\_Texture \*\*texture28, SDL\_Texture \*\*texture29, SDL\_Texture \*\*texture30, SDL\_Texture \*\*texture31, SDL\_Texture \*\*texture32, SDL\_Texture \*\*texture33, SDL\_Texture \*\*texture34, SDL\_Texture \*\*texture35, SDL\_Texture \*\*texture36, SDL\_Texture \*\*texture37, SDL\_Texture \*\*texture38, SDL\_Texture \*\*texture39, SDL\_Texture \*\*texture40, SDL\_Texture \*\*texture41, SDL\_Texture \*\*texture42, SDL\_Texture \*\*texture43, SDL\_Texture \*\*texture44, SDL\_Texture \*\*texture45, SDL\_Texture \*\*texture46, SDL\_Texture \*\*texture47, SDL\_Texture \*\*texture48, SDL\_Texture \*\*texture49, SDL\_Texture \*\*texture50, SDL\_Texture \*\*texture51, SDL\_Texture \*\*texture52, SDL\_Texture \*\*texture53, SDL\_Texture \*\*texture54, SDL\_Texture \*\*texture55, SDL\_Texture \*\*texture56, SDL\_Texture \*\*texture57, SDL\_Texture \*\*texture58, SDL\_Texture \*\*texture59, SDL\_Texture \*\*texture60, SDL\_Texture \*\*texture61, SDL\_Texture \*\*texture62, SDL\_Texture \*\*texture63, SDL\_Texture \*\*texture64, SDL\_Texture \*\*texture65, SDL\_Texture \*\*texture66, SDL\_Texture \*\*texture67, SDL\_Texture \*\*texture68, SDL\_Texture \*\*texture69, SDL\_Texture \*\*texture70, SDL\_Texture \*\*texture71, SDL\_Texture \*\*texture72, SDL\_Texture \*\*textures\_images\_succes)
  - void `destruire_fenetre_rendu` (SDL\_Renderer \*\*renderer, SDL\_Window \*\*window)  
*Fonction qui permet de détruire le rendu et la fenêtre.*

## 4.29.1 Documentation des définitions de type

### 4.29.1.1 direction\_t

```
typedef enum direction_s direction_t
```

### 4.29.1.2 modes\_t

```
typedef enum modes_s modes_t
```

### 4.29.1.3 option\_t

```
typedef enum option_s option_t
```

#### 4.29.1.4 page\_t

```
typedef enum page_s page_t
```

#### 4.29.1.5 personnage\_t

```
typedef enum personnage_s personnage_t
```

#### 4.29.1.6 position\_t

```
typedef enum position_s position_t
```

### 4.29.2 Documentation du type de l'énumération

#### 4.29.2.1 direction\_s

```
enum direction_s
```

Valeurs énumérées

HAUT	
BAS	
GAUCHE	
DROITE	
HAUT_DROITE	
BAS_GAUCHE	

Définition à la ligne 24 du fichier [fonctions\\_generales.h](#).

#### 4.29.2.2 modes\_s

```
enum modes_s
```

Valeurs énumérées

MODE_NORMAL	
MODE_HARD	

Définition à la ligne 12 du fichier [fonctions\\_generales.h](#).

#### 4.29.2.3 option\_s

```
enum option_s
```

## Valeurs énumérées

ONGLET_SON	
ONGLET_TOUCHES	

Définition à la ligne 9 du fichier [fonctions\\_generales.h](#).

**4.29.2.4 page\_s**

enum [page\\_s](#)

## Valeurs énumérées

MENU_PRINCIPAL	
OPTIONS	
NOUVELLE_PARTIE	
INTRODUCTION	
CARTE	
NIVEAU_1	
NIVEAU_2	
NIVEAU_3	
NIVEAU_4	

Définition à la ligne 18 du fichier [fonctions\\_generales.h](#).

**4.29.2.5 personnage\_s**

enum [personnage\\_s](#)

## Valeurs énumérées

PERSONNAGE↔ _1	
PERSONNAGE↔ _2	

Définition à la ligne 15 du fichier [fonctions\\_generales.h](#).

**4.29.2.6 position\_s**

enum [position\\_s](#)

## Valeurs énumérées

NIVEAU0	
NIVEAU1	
NIVEAU2	
NIVEAU3	
NIVEAU4	

Définition à la ligne 21 du fichier [fonctions\\_generales.h](#).

### 4.29.3 Documentation des fonctions

#### 4.29.3.1 `affichage_texte()`

```
void affichage_texte (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture,
    itemMenu * item,
    TTF_Font ** police,
    SDL_Color couleur )
```

Définition à la ligne 55 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.2 `chargement_image()`

```
void chargement_image (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture,
    char * chemin )
```

Fonction qui permet de charger une image.

##### Paramètres

<i>renderer</i>	rendu sur lequel posé l'image
<i>surface</i>	Surface à utiliser pour récupérer l'image
<i>texture</i>	Texture à crée
<i>chemin</i>	Pointeur sur caractère représentant le chemin d'accès du fichier

##### Voir également

[erreur](#)

Définition à la ligne 30 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.3 `clic_case()`

```
int clic_case (
    SDL_Event event,
    SDL_Rect rectangle )
```

Fonction qui permet de renvoyer vrai quand on clique sur un rectangle, faux sinon.

##### Paramètres

<i>event</i>	Evenement SDL
<i>rectangle</i>	Rectangle qui a été cliqué ou non

**Renvoie**

booléen représentant si le clic s'est fait dans le rectangle (1 si c'est le cas sinon 0)

Définition à la ligne 414 du fichier [fonctions\\_generales.c](#).

**4.29.3.4 clic\_plein\_ecran()**

```
int clic_plein_ecran (
    SDL_Event event,
    SDL_Rect * rectangle_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Window ** window )
```

Fonction qui permet de mettre la fenêtre en plein écran quand on clique sur le bouton plein écran.

**Paramètres**

<i>event</i>	Evenement SDL
<i>rectangle_plein_ecran</i>	Rectangle ou se situe le bouton pour afficher le plein écran ou le retirer
<i>plein_ecran</i>	booléen qui dit si il est en mode plein écran
<i>window</i>	fenêtre à changer en pleine écran ou non

**Renvoie**

le changement d'état sous la forme d'un booléen

Définition à la ligne 445 du fichier [fonctions\\_generales.c](#).

**4.29.3.5 creer\_fenetre\_rendu()**

```
void creer_fenetre_rendu (
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    int largeur,
    int hauteur )
```

Fonction qui permet de créer une fenêtre et le rendu.

**Paramètres**

<i>window</i>	fenêtre à créer
<i>renderer</i>	Rendu de la fenêtre à créer
<i>largeur</i>	largeur de la fenêtre souhaité
<i>hauteur</i>	hauteur de la fenêtre souhaité

**Voir également**

[erreur](#)

Définition à la ligne 78 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.6 demande\_quitter\_niveau()

```
void demande_quitter_niveau (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_demande_quitter,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleur,
    itemMenu * itemsDemandeQuitter,
    int tailleDemandeQuitter,
    int largeur,
    int hauteur )
```

Fonction qui permet de demander à l'utilisateur de quitter le niveau.

##### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>rectangle_demande_quitter</i>	Rectangle de la demande de quitter le niveau SDL.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleur</i>	Couleur du texte.
<i>itemsDemandeQuitter</i>	Tableau d'items pour la demande de quitter le niveau.
<i>tailleDemandeQuitter</i>	Taille du tableau d'items pour la demande de quitter le niveau.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

##### Voir également

[affichage\\_texte](#)

Définition à la ligne 252 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.7 demande\_sauvegarde()

```
void demande_sauvegarde (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_demande_sauvegarde,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleur,
    itemMenu * itemsDemandeSauvegarde,
    int tailleDemandeSauvegarde,
    int largeur,
    int hauteur )
```

fenêtre se chargeant de demander à l'utilisateur si il souhaite sauvegarder

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>rectangle_demande_sauvegarde</i>	Rectangle de la demande de sauvegarde SDL.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleur</i>	Couleur du texte.
<i>itemsDemandeSauvegarde</i>	Tableau d'items pour la demande de sauvegarde.
<i>tailleDemandeSauvegarde</i>	Taille du tableau d'items pour la demande de sauvegarde.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

## Voir également

[affichage\\_texte](#)

Définition à la ligne [187](#) du fichier [fonctions\\_generales.c](#).

**4.29.3.8 déplacement\_personnage()**

```
void deplacement_personnage (
    int * saut,
    int * tombe,
    int * position_x,
    int * position_y,
    int * position_avant_saut,
    int sauter,
    int avancer,
    int reculer,
    int tile_map[18][32],
    personnage_t personnageActif )
```

Définition à la ligne [491](#) du fichier [fonctions\\_generales.c](#).

**4.29.3.9 detruire\_fenetre\_rendu()**

```
void detruire_fenetre_rendu (
    SDL_Renderer ** renderer,
    SDL_Window ** window )
```

Fonction qui permet de détruire le rendu et la fenêtre.

## Paramètres

<i>renderer</i>	Rendu à détruire
<i>widnow</i>	fenêtre à détruire

Définition à la ligne [690](#) du fichier [fonctions\\_generales.c](#).



#### 4.29.3.10 detruire\_objets()

```
void detruire_objets (
    TTF_Font ** police,
    Mix_Music ** musique,
    SDL_Texture ** texture1,
    SDL_Texture ** texture2,
    SDL_Texture ** texture3,
    SDL_Texture ** texture4,
    SDL_Texture ** texture5,
    SDL_Texture ** texture6,
    SDL_Texture ** texture7,
    SDL_Texture ** texture8,
    SDL_Texture ** texture9,
    SDL_Texture ** texture10,
    SDL_Texture ** texture11,
    SDL_Texture ** texture12,
    SDL_Texture ** texture13,
    SDL_Texture ** texture14,
    SDL_Texture ** texture15,
    SDL_Texture ** texture16,
    SDL_Texture ** texture17,
    SDL_Texture ** texture18,
    SDL_Texture ** texture19,
    SDL_Texture ** texture20,
    SDL_Texture ** texture21,
    SDL_Texture ** texture22,
    SDL_Texture ** texture23,
    SDL_Texture ** texture24,
    SDL_Texture ** texture25,
    SDL_Texture ** texture26,
    SDL_Texture ** texture27,
    SDL_Texture ** texture28,
    SDL_Texture ** texture29,
    SDL_Texture ** texture30,
    SDL_Texture ** texture31,
    SDL_Texture ** texture32,
    SDL_Texture ** texture33,
    SDL_Texture ** texture34,
    SDL_Texture ** texture35,
    SDL_Texture ** texture36,
    SDL_Texture ** texture37,
    SDL_Texture ** texture38,
    SDL_Texture ** texture39,
    SDL_Texture ** texture40,
    SDL_Texture ** texture41,
    SDL_Texture ** texture42,
    SDL_Texture ** texture43,
    SDL_Texture ** texture44,
    SDL_Texture ** texture45,
    SDL_Texture ** texture46,
    SDL_Texture ** texture47,
    SDL_Texture ** texture48,
    SDL_Texture ** texture49,
    SDL_Texture ** texture50,
    SDL_Texture ** texture51,
    SDL_Texture ** texture52,
    SDL_Texture ** texture53,
```

```

SDL_Texture ** texture54,
SDL_Texture ** texture55,
SDL_Texture ** texture56,
SDL_Texture ** texture57,
SDL_Texture ** texture58,
SDL_Texture ** texture59,
SDL_Texture ** texture60,
SDL_Texture ** texture61,
SDL_Texture ** texture62,
SDL_Texture ** texture63,
SDL_Texture ** texture64,
SDL_Texture ** texture65,
SDL_Texture ** texture66,
SDL_Texture ** texture67,
SDL_Texture ** texture68,
SDL_Texture ** texture69,
SDL_Texture ** texture70,
SDL_Texture ** texture71,
SDL_Texture ** texture72,
SDL_Texture ** textures_images_succes )

```

Définition à la ligne [599](#) du fichier [fonctions\\_generales.c](#).

#### 4.29.3.11 erreur()

```

void erreur (
    const char * message )

```

Affiche l'erreur en cas de problème et ferme la SDL.

##### Paramètres

<i>message</i>	Un pointeur sur caractère en lecture représentant le message d'erreur
----------------	---

##### Renvoie

Arrêt du programme en Echec

Définition à la ligne [14](#) du fichier [fonctions\\_generales.c](#).

#### 4.29.3.12 initialisation\_objets()

```

void initialisation_objets (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Texture ** texture_image_options,
    SDL_Texture ** texture_image_passer,
    itemMenu * itemsDemandeSauvegarde,
    itemMenu * itemsDemandeQuitter,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    SDL_Texture ** texture_image_monstre_terrestre,

```

```

SDL_Texture ** texture_image_monstre_volant,
SDL_Texture ** texture_image_perso_1_gagnant,
SDL_Texture ** texture_image_perso_2_gagnant,
niveaux * avancee_niveaux,
TTF_Font ** police,
SDL_Texture ** texture_image_croix )

```

Définition à la ligne 117 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.13 redimensionnement\_fenetre()

```

void redimensionnement_fenetre (
    SDL_Event event,
    int * largeur,
    int * hauteur )

```

Fonction qui permet de récupérer les nouvelles dimensions de la fenêtre pour redimensionner cette dernière et les différents objets.

##### Paramètres

<i>event</i>	Evenement SDL
<i>largeur</i>	largeur de la fenêtre a redimensionné
<i>hauteur</i>	hauteur de la fenêtre a redimensionné

Définition à la ligne 309 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.14 sauvegarder\_partie()

```

void sauvegarder_partie (
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_sauter_monter,
    SDL_Keycode * touche_descendre,
    SDL_Keycode * touche_interagir,
    barreDeSon * barre_de_son,
    itemMenu * pseudo,
    modes_t modeActif,
    personnage_t personnageActif,
    position_t positionActive,
    niveaux * avancee_niveaux,
    int tailleNiveaux,
    time_t temps_debut_partie,
    int compteur_mort,
    int avancee_succes[10] )

```

Définition à la ligne 359 du fichier [fonctions\\_generales.c](#).

#### 4.29.3.15 verification\_sauvegarde()

```

int verification_sauvegarde ( )

```

Vérifie si une sauvegarde existe.

**Renvoie**

booléen représentant si il y a une sauvegarde ou non (1 si existant, sinon 0)

**Voir également**

[erreur](#)

Définition à la ligne 324 du fichier [fonctions\\_generales.c](#).

## 4.30 fonctions\_generales.h

[Aller à la documentation de ce fichier.](#)

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <SDL2/SDL.h>
00004 #include <SDL2/SDL_ttf.h>
00005 #include <SDL2/SDL_image.h>
00006 #include <SDL2/SDL_mixer.h>
00007
00008 /* Énumération de constantes pour l'onglet actif des options */
00009 typedef enum option_s {ONGLET_SON, ONGLET_TOUCHES} option_t;
00010
00011 /* Énumération de constantes pour le mode sélectionné */
00012 typedef enum modes_s {MODE_NORMAL, MODE_HARD} modes_t;
00013
00014 /* Énumération de constantes pour le personnage sélectionné */
00015 typedef enum personnage_s {PERSONNAGE_1, PERSONNAGE_2} personnage_t;
00016
00017 /* Énumération de constantes pour la page */
00018 typedef enum page_s {MENU_PRINCIPAL, OPTIONS, NOUVELLE_PARTIE, INTRODUCTION, CARTE, NIVEAU_1,
00019                     NIVEAU_2, NIVEAU_3, NIVEAU_4} page_t;
00020
00021 /* Énumération de constantes pour la position sur la carte */
00022 typedef enum position_s {NIVEAU0, NIVEAU1, NIVEAU2, NIVEAU3, NIVEAU4} position_t;
00023
00024 /* Énumération de constantes pour la direction du personnage sur la carte */
00025 typedef enum direction_s {HAUT, BAS, GAUCHE, DROITE, HAUT_DROITE, BAS_GAUCHE} direction_t;
00026
00027 /* Structure pour représenter une case avec un rectangle et du texte */
00028 typedef struct {
00029     SDL_Rect rectangle;
00030     char texte[100];
00031 } itemMenu;
00032
00033 /* Structure pour représenter une barre de son */
00034 typedef struct {
00035     SDL_Rect barre;
00036     SDL_Rect curseur;
00037     float volume;
00038     float volume_precedent;
00039 } barreDeSon;
00040
00041 /* Structure pour représenter les collectibles de chaque niveaux */
00042 typedef struct {
00043     int niveau_fini;
00044     SDL_Texture *texture_image_collectible;
00045     int numero_collectible[3];
00046 } niveaux;
00047
00048 /* Squelette de la fonction erreur */
00049 void erreur(const char *message);
00050
00051 /* Squelette de la fonction chargement_image */
00052 void chargement_image(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, char
    *chemin);
00053
00054 /* Squelette de la fonction affichage_texte */
00055 void affichage_texte(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, itemMenu
    *item,
00056                     TTF_Font **police, SDL_Color couleur);
00057
00058 /* Squelette de la fonction creer_fenetre_rendu */
00059 void creer_fenetre_rendu(SDL_Window **window, SDL_Renderer **renderer, int largeur, int hauteur);
00060
00061 /* Squelette de la fonction initialisation_objets */
```

```

00061 void initialisation_objets(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
    **texture_image_plein_ecran,
00062                               SDL_Texture **texture_image_retour_en_arriere, SDL_Texture
    **texture_image_options,
00063                               SDL_Texture **texture_image_passer, itemMenu *itemsDemandeSauvegarde,
    itemMenu *itemsDemandeQuitter,
00064                               SDL_Texture **texture_image_fin_premiers_niveaux, SDL_Texture
    **texture_image_monstre_terrestre,
00065                               SDL_Texture **texture_image_monstre_volant, SDL_Texture
    **texture_image_perso_1_gagnant,
00066                               SDL_Texture **texture_image_perso_2_gagnant,
00067                               niveaux *avancee_niveaux, TTF_Font **police, SDL_Texture
    **texture_image_croix);
00068
00069 /* Squelette de la fonction demande_sauvegarde */
00070 void demande_sauvegarde(SDL_Renderer **renderer, SDL_Rect *rectangle_demande_sauvegarde,
    SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00071                          SDL_Color couleur,
    itemMenu *itemsDemandeSauvegarde, int tailleDemandeSauvegarde, int largeur,
00072                          int hauteur);
00073
00074 /* Squelette de la fonction demande_quitter_niveau */
00075 void demande_quitter_niveau(SDL_Renderer **renderer, SDL_Rect *rectangle_demande_quitter,
    SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00076                             SDL_Color couleur,
    itemMenu *itemsDemandeQuitter, int tailleDemandeQuitter, int largeur, int
00077                             hauteur);
00078
00079 /* Squelette de la fonction redimensionnement_fenetre */
00080 void redimensionnement_fenetre(SDL_Event event, int *largeur, int *hauteur);
00081
00082 /* Squelette de la fonction verification_sauvegarde */
00083 int verification_sauvegarde();
00084
00085 /* Squelette de la fonction sauvegarder_partie */
00086 void sauvegarder_partie(SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
    SDL_Keycode *touche_sauter_monter,
00087                        SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, barreDeSon
    *barre_de_son, itemMenu *pseudo,
00088                        modes_t modeActif, personnage_t personnageActif, position_t positionActive,
    niveaux *avancee_niveaux, int tailleNiveaux, time_t temps_debut_partie, int
00089                        compteur_mort, int avancee_succes[10]);
00090
00091 /* Squelette de la fonction clic_case */
00092 int clic_case(SDL_Event event, SDL_Rect rectangle);
00093
00094 /* Squelette de la fonction deplacement_personnage */
00095 void deplacement_personnage(int *saut, int *tombe, int *position_x, int *position_y, int
    *position_avant_saut,
00096                             int sauter, int avancer, int reculer, int tile_map[18][32], personnage_t
    personnageActif);
00097
00098 /* Squelette de la fonction clic_plein_ecran */
00099 int clic_plein_ecran(SDL_Event event, SDL_Rect *rectangle_plein_ecran, SDL_bool *plein_ecran,
    SDL_Window **window);
00100
00101 /* Squelette de la fonction detruire_objets */
00102 void detruire_objets(TTF_Font **police, Mix_Music **musique, SDL_Texture **texture1, SDL_Texture
    **texture2,
00103                     SDL_Texture **texture3, SDL_Texture **texture4, SDL_Texture **texture5,
    SDL_Texture **texture6,
00104                     SDL_Texture **texture7, SDL_Texture **texture8,
    SDL_Texture **texture9, SDL_Texture **texture10, SDL_Texture **texture11,
00105                     SDL_Texture **texture12, SDL_Texture **texture13, SDL_Texture **texture14,
    SDL_Texture **texture15, SDL_Texture **texture16,
00106                     SDL_Texture **texture17, SDL_Texture **texture18,
    SDL_Texture **texture19, SDL_Texture **texture20,
00107                     SDL_Texture **texture21, SDL_Texture **texture22,
    SDL_Texture **texture23, SDL_Texture **texture24,
00108                     SDL_Texture **texture25, SDL_Texture **texture26,
    SDL_Texture **texture27, SDL_Texture **texture28,
00109                     SDL_Texture **texture29, SDL_Texture **texture30,
    SDL_Texture **texture31, SDL_Texture **texture32,
00110                     SDL_Texture **texture33, SDL_Texture **texture34,
    SDL_Texture **texture35, SDL_Texture **texture36, SDL_Texture **texture37,
00111                     SDL_Texture **texture38, SDL_Texture **texture39, SDL_Texture **texture40,
    SDL_Texture **texture41, SDL_Texture **texture42,
00112                     SDL_Texture **texture43, SDL_Texture **texture44,
    SDL_Texture **texture45, SDL_Texture **texture46,
00113                     SDL_Texture **texture47, SDL_Texture **texture48,
    SDL_Texture **texture49, SDL_Texture **texture50,
00114                     SDL_Texture **texture51, SDL_Texture **texture52,
    SDL_Texture **texture53, SDL_Texture **texture54,
00115                     SDL_Texture **texture55, SDL_Texture **texture56,
    SDL_Texture **texture57, SDL_Texture **texture58,
00116                     SDL_Texture **texture59, SDL_Texture **texture60,
    SDL_Texture **texture61, SDL_Texture **texture62,

```

```

00130         SDL_Texture **texture63, SDL_Texture **texture64,
00131         SDL_Texture **texture65, SDL_Texture **texture66,
00132         SDL_Texture **texture67, SDL_Texture **texture68,
00133         SDL_Texture **texture69, SDL_Texture **texture70,
00134         SDL_Texture **texture71, SDL_Texture **texture72,
00135         SDL_Texture **textures_images_succes);
00136
00137
00138 /* Squelette de la fonction detruire_fenetre_rendu */
00139 void detruire_fenetre_rendu(SDL_Renderer **renderer, SDL_Window **window);

```

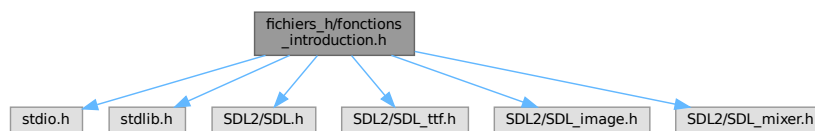
### 4.31 Référence du fichier fichiers\_h/fonctions\_introduction.h

```

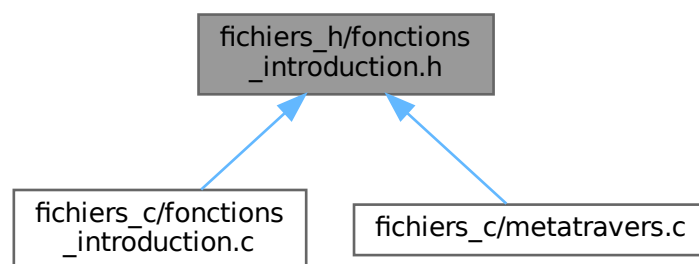
#include <stdio.h>
#include <stdlib.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

```

Graphe des dépendances par inclusion de fonctions\_introduction.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



#### Fonctions

- void [mise\\_a\\_jour\\_rendu\\_introduction](#) (SDL\_Renderer \*\*renderer, int indice, char \*ligne, SDL\_Rect \*rectangle\_passer, SDL\_Texture \*\*texture\_image\_passer, SDL\_Rect \*rectangle\_texte\_introduction, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurBlanche, int largeur, int hauteur)

*Fonction qui met à jour le rendu de l'introduction.*

— void [introduction](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_passer, SDL\_Texture \*\*texture\_image\_passer, SDL\_Rect \*rectangle\_texte\_introduction, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, [personnage\\_t](#) \*personnageActif, SDL\_Color couleurBlanche, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active)

*Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans l'introduction.*

## 4.31.1 Documentation des fonctions

### 4.31.1.1 introduction()

```
void introduction (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_passer,
    SDL_Texture ** texture_image_passer,
    SDL_Rect * rectangle_texte_introduction,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    personnage\_t * personnageActif,
    SDL_Color couleurBlanche,
    int * largeur,
    int * hauteur,
    page\_t * page_active )
```

Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans l'introduction.

#### Paramètres

<i>event</i>	Événement SDL.
<i>window</i>	Pointeur vers la fenêtre SDL.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>programme_lance</i>	Indicateur de lancement du programme.
<i>rectangle_passer</i>	Rectangle pour le bouton "Passer".
<i>texture_image_passer</i>	Texture de l'image du bouton "Passer".
<i>rectangle_texte_introduction</i>	Rectangle pour le texte d'introduction.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>personnageActif</i>	Personnage actif du joueur.
<i>couleurBlanche</i>	Couleur blanche SDL.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.
<i>page_active</i>	Page active du jeu.

#### Voir également

[erreur](#)  
[redimensionnement\\_fenetre](#)  
[clic\\_case](#)

### [mise\\_a\\_jour\\_rendu\\_introduction](#)

Définition à la ligne 103 du fichier [fonctions\\_introduction.c](#).

#### 4.31.1.2 `mise_a_jour_rendu_introduction()`

```
void mise_a_jour_rendu_introduction (
    SDL_Renderer ** renderer,
    int indice,
    char * ligne,
    SDL_Rect * rectangle_passer,
    SDL_Texture ** texture_image_passer,
    SDL_Rect * rectangle_texte_introduction,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurBlanche,
    int largeur,
    int hauteur )
```

Fonction qui met à jour le rendu de l'introduction.

##### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>indice</i>	Indice de la ligne de texte.
<i>ligne</i>	Ligne de texte à afficher.
<i>rectangle_passer</i>	Rectangle pour le bouton "Passer".
<i>texture_image_passer</i>	Texture de l'image du bouton "Passer".
<i>rectangle_texte_introduction</i>	Rectangle pour le texte d'introduction.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleurBlanche</i>	Couleur blanche SDL.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

Voir également

[erreur](#)

Définition à la ligne 25 du fichier [fonctions\\_introduction.c](#).

## 4.32 `fonctions_introduction.h`

[Aller à la documentation de ce fichier.](#)

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <SDL2/SDL.h>
00004 #include <SDL2/SDL_ttf.h>
00005 #include <SDL2/SDL_image.h>
00006 #include <SDL2/SDL_mixer.h>
```



```

00007
00008 /* Squelette de la fonction mise_a_jour_rendu_introduction */
00009 void mise_a_jour_rendu_introduction(SDL_Renderer **renderer, int indice, char *ligne,
00010                                     SDL_Rect *rectangle_passer, SDL_Texture **texture_image_passer,
00011                                     SDL_Rect *rectangle_texte_introduction, SDL_Surface **surface,
00012                                     SDL_Texture **texture_texte,
00013                                     TTF_Font **police, SDL_Color couleurBlanche,
00014                                     int largeur, int hauteur);
00015 /* Squelette de la fonction introduction */
00016 void introduction(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
00017                  *programme_lance,
00018                  SDL_Rect *rectangle_passer, SDL_Texture **texture_image_passer,
00019                  SDL_Rect *rectangle_texte_introduction, SDL_Surface **surface, SDL_Texture
00020                  **texture_texte, TTF_Font **police,
00021                  personnage_t *personnageActif, SDL_Color couleurBlanche,
00022                  int *largeur, int *hauteur, page_t *page_active);

```

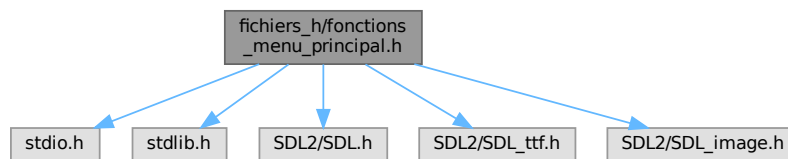
### 4.33 Référence du fichier fichiers\_h/fonctions\_menu\_principal.h

```

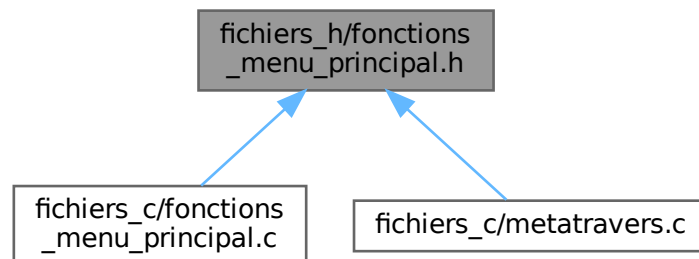
#include <stdio.h>
#include <stdlib.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>

```

Graphe des dépendances par inclusion de fonctions\_menu\_principal.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [initialisation\\_objets\\_menu\\_principal](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_menu, [itemMenu](#) \*titre, [itemMenu](#) \*itemsMenu, int tailleMenu)

*Fonction qui permet d'initialiser les différents objets du menu principal.*

- void [mise\\_a\\_jour\\_rendu\\_menu\\_principal](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_menu, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurTitre, SDL\_Color couleurNoire, int selection\_menu, [itemMenu](#) \*itemsMenu, int tailleMenu, int largeur, int hauteur)

*Fonction qui met à jour le rendu du menu principal.*

- void [menu\\_principal](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_menu, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurTitre, SDL\_Color couleurNoire, int code\_de\_triche[3], int \*selection\_menu, [itemMenu](#) \*itemsMenu, int tailleMenu, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active)

*Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le menu principal.*

## 4.33.1 Documentation des fonctions

### 4.33.1.1 initialisation\_objets\_menu\_principal()

```
void initialisation_objets_menu_principal (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_menu,
    itemMenu * titre,
    itemMenu * itemsMenu,
    int tailleMenu )
```

Fonction qui permet d'initialiser les différents objets du menu principal.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Surface SDL.
<i>texture_image_menu</i>	Texture de l'image du menu principal.
<i>titre</i>	Titre du menu.
<i>itemsMenu</i>	Tableau d'items pour le menu principal.
<i>tailleMenu</i>	Taille du tableau d'items pour le menu principal.

#### Voir également

[chargement\\_image](#)

Définition à la ligne 21 du fichier [fonctions\\_menu\\_principal.c](#).

### 4.33.1.2 menu\_principal()

```
void menu_principal (
    SDL_Event * event,
```

```

SDL_Window ** window,
SDL_Renderer ** renderer,
SDL_bool * programme_lance,
SDL_Texture ** texture_image_menu,
SDL_Rect * rectangle_plein_ecran,
SDL_Texture ** texture_image_plein_ecran,
SDL_bool * plein_ecran,
itemMenu * titre,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Color couleurTitre,
SDL_Color couleurNoire,
int code_de_triche[3],
int * selection_menu,
itemMenu * itemsMenu,
int tailleMenu,
int * largeur,
int * hauteur,
page_t * page_active )

```

Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le menu principal.

#### Paramètres

<i>event</i>	Événement SDL.
<i>window</i>	Pointeur vers la fenêtre SDL.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>programme_lance</i>	Indicateur de lancement du programme.
<i>texture_image_menu</i>	Texture de l'image du menu principal.
<i>rectangle_plein_ecran</i>	Rectangle plein écran SDL.
<i>texture_image_plein_ecran</i>	Texture de l'image en plein écran.
<i>plein_ecran</i>	Indicateur de mode plein écran.
<i>titre</i>	Titre du menu principal.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleurTitre</i>	Couleur du titre.
<i>couleurNoire</i>	Couleur noire SDL.
<i>code_de_triche</i>	Tableau de codes de triche.
<i>selection_menu</i>	Sélection actuelle du menu.
<i>itemsMenu</i>	Tableau d'items pour le menu principal.
<i>tailleMenu</i>	Taille du tableau d'items pour le menu principal.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.
<i>page_active</i>	Page active du menu.

#### Voir également

[redimensionnement\\_fenetre](#)  
[clik\\_case](#)  
[clik\\_plein\\_ecran](#)  
[mise\\_a\\_jour\\_rendu\\_menu\\_principal](#)

Définition à la ligne 214 du fichier [fonctions\\_menu\\_principal.c](#).

#### 4.33.1.3 mise\_a\_jour\_rendu\_menu\_principal()

```
void mise_a_jour_rendu_menu_principal (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_menu,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    itemMenu * titre,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurTitre,
    SDL_Color couleurNoire,
    int selection_menu,
    itemMenu * itemsMenu,
    int tailleMenu,
    int largeur,
    int hauteur )
```

Fonction qui met à jour le rendu du menu principal.

##### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>texture_image_menu</i>	Texture de l'image du menu principal.
<i>rectangle_plein_ecran</i>	Rectangle plein écran SDL.
<i>texture_image_plein_ecran</i>	Texture de l'image en plein écran.
<i>titre</i>	Titre du menu principal.
<i>surface</i>	Surface SDL.
<i>texture_texte</i>	Texture du texte SDL.
<i>police</i>	Police de caractères TTF.
<i>couleurTitre</i>	Couleur du titre.
<i>couleurNoire</i>	Couleur noire SDL.
<i>selection_menu</i>	Sélection actuelle du menu.
<i>itemsMenu</i>	Tableau d'items pour le menu principal.
<i>tailleMenu</i>	Taille du tableau d'items pour le menu principal.
<i>largeur</i>	Largeur de l'écran.
<i>hauteur</i>	Hauteur de l'écran.

Voir également

[erreur](#)  
[affichage\\_texte](#)

Définition à la ligne 84 du fichier [fonctions\\_menu\\_principal.c](#).

## 4.34 fonctions\_menu\_principal.h

[Aller à la documentation de ce fichier.](#)

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <SDL2/SDL.h>
```

```

00004 #include <SDL2/SDL_ttf.h>
00005 #include <SDL2/SDL_image.h>
00006
00007 /* Squelette de la fonction initialisation_objets_menu_principal */
00008 void initialisation_objets_menu_principal(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
    **texture_image_menu,
00009                                         itemMenu *titre, itemMenu *itemsMenu, int tailleMenu);
00010
00011 /* Squelette de la fonction mise_a_jour_rendu_menu_principal */
00012 void mise_a_jour_rendu_menu_principal(SDL_Renderer **renderer, SDL_Texture **texture_image_menu,
    SDL_Rect *rectangle_plein_ecran, SDL_Texture
00013
00014 **texture_image_plein_ecran,
00015                                         itemMenu *titre, SDL_Surface **surface, SDL_Texture
00016 **texture_texte, TTF_Font **police,
00017                                         SDL_Color couleurTitre, SDL_Color couleurNoire, int
00018 selection_menu,
00019                                         itemMenu *itemsMenu, int tailleMenu, int largeur, int hauteur);
00020
00021 /* Squelette de la fonction menu_principal */
00022 void menu_principal(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
    *programme_lance, SDL_Texture **texture_image_menu,
00023                     SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
00024 *plein_ecran,
00025                     itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font
00026 **police,
00027                     SDL_Color couleurTitre, SDL_Color couleurNoire, int code_de_triche[3], int
00028 *selection_menu,
00029                     itemMenu *itemsMenu, int tailleMenu, int *largeur, int *hauteur, page_t
00030 *page_active);

```

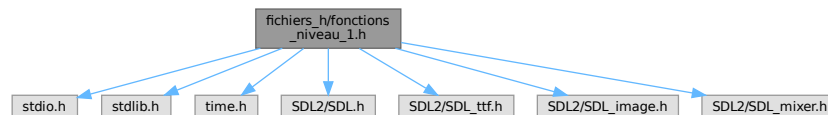
## 4.35 Référence du fichier fichiers\_h/fonctions\_niveau\_1.h

```

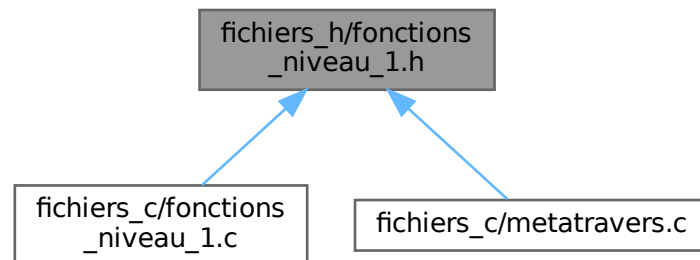
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

```

Graphe des dépendances par inclusion de fonctions\_niveau\_1.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [initialisation\\_objets\\_niveau\\_1](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_sol\_surface\_niveau\_1, SDL\_Texture \*\*texture\_image\_sol\_profondeur\_niveau\_1, SDL\_Texture \*\*texture\_image\_fond\_niveau\_1, SDL\_Texture \*\*texture\_image\_nuage\_1, SDL\_Texture \*\*texture\_image\_nuage\_2)  
*Fonction qui permet d'initialiser les différents objets du niveau 4.*
- void [chargement\\_niveau\\_1](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map\_niveau\_1[18][110])  
*Fonction qui permet de créer l'étage 1.*
- void [mise\\_a\\_jour\\_rendu\\_niveau\\_1](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_sol\_surface, SDL\_Texture \*\*texture\_image\_sol\_profondeur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_tile, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, int position\_x, int position\_y, int tile\_map[18][32], int secret, SDL\_Texture \*\*texture\_image\_nuage\_1, SDL\_Texture \*\*texture\_image\_nuage\_2, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix)
- void [niveau\\_1](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image\_sol\_surface, SDL\_Texture \*\*texture\_image\_sol\_profondeur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, int \*mouvement\_monstre, SDL\_Surface \*\*surface, int collectibles\_intermediaires[3], time\_t \*timestamp, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, int \*decalage, int \*secret\_1, int \*secret\_2, int tile\_map[18][32], int tile\_map\_niveau\_1[18][110], SDL\_Rect \*rectangle\_tile, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemande, SDL\_Texture \*\*texture\_image\_perso\_gagnant, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Rect \*rectangle\_demande, SDL\_Texture \*\*texture\_image\_nuage\_1, SDL\_Texture \*\*texture\_image\_nuage\_2, SDL\_Color couleurNoire, SDL\_Texture \*\*texture\_image\_fin\_premiers\_niveaux, int \*avancer, int \*reculer, int \*sauter, int \*position\_avant\_saut, int \*saut, int \*tombe, int \*position\_x\_initiale, int \*position\_y\_initiale, int \*position\_x, int \*position\_y, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, [page\\_t](#) \*page\_active, [itemMenu](#) \*itemsDemandeSauvegarde, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_interagir, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [modes\\_t](#) \*modeActif, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

## 4.35.1 Documentation des fonctions

### 4.35.1.1 chargement\_niveau\_1()

```
void chargement_niveau_1 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map_niveau_1[18][110] )
```

Fonction qui permet de créer l'étage 1.

#### Paramètres

<i>position_x</i>	Position du personnage à l'apparition sur la verticale
<i>position_y</i>	Position du personnage à l'apparition sur l'horizontale
<i>position_x_initiale</i>	Position initiale verticale du niveau en cas de mort du personnage
<i>position_y_initiale</i>	Position initiale horizontale du niveau en cas de mort du personnage
<i>tile_map_niveau_1</i>	Map du niveau 1

Définition à la ligne 43 du fichier [fonctions\\_niveau\\_1.c](#).

### 4.35.1.2 initialisation\_objets\_niveau\_1()

```
void initialisation_objets_niveau_1 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_sol_surface_niveau_1,
    SDL_Texture ** texture_image_sol_profondeur_niveau_1,
    SDL_Texture ** texture_image_fond_niveau_1,
    SDL_Texture ** texture_image_nuage_1,
    SDL_Texture ** texture_image_nuage_2 )
```

Fonction qui permet d'initialiser les différents objets du niveau 4.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Surface SDL.
<i>texture_image_sol_surface_niveau_1</i>	Texture de l'image du sol de surface du niveau 1.
<i>texture_image_sol_profondeur_niveau_1</i>	Texture de l'image du sol de profondeur du niveau 1.
<i>texture_image_fond_niveau_1</i>	Texture de l'image de fond du niveau 1.
<i>texture_image_nuage_1</i>	Texture de l'image du nuage 1.
<i>texture_image_nuage_2</i>	Texture de l'image du nuage 2.

Voir également

[chargement\\_image](#)

Définition à la ligne 20 du fichier [fonctions\\_niveau\\_1.c](#).

#### 4.35.1.3 mise\_a\_jour\_rendu\_niveau\_1()

```
void mise_a_jour_rendu_niveau_1 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_sol_surface,
    SDL_Texture ** texture_image_sol_profondeur,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    SDL_Texture ** texture_image_pique,
    niveaux * avancee_niveaux,
    SDL_Texture ** texture_image_fin_premiers_niveaux,
    int position_x,
    int position_y,
    int tile_map[18][32],
    int secret,
    SDL_Texture ** texture_image_nuage_1,
    SDL_Texture ** texture_image_nuage_2,
    int largeur,
    int hauteur,
    int largeur_tile,
    int hauteur_tile,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix )
```

Définition à la ligne 114 du fichier [fonctions\\_niveau\\_1.c](#).

#### 4.35.1.4 niveau\_1()

```
void niveau_1 (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    SDL_Texture ** texture_image_monstre_terrestre,
```



```

SDL_Texture ** texture_image_monstre_volant,
SDL_Texture ** texture_image_sol_surface,
SDL_Texture ** texture_image_sol_profondeur,
SDL_Texture ** texture_image_fond,
SDL_Texture ** texture_image_pique,
niveau * avancee_niveaux,
int * mouvement_monstre,
SDL_Surface ** surface,
int collectibles_intermediaires[3],
time_t * timestamp,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_sauter_monter,
int * decalage,
int * secret_1,
int * secret_2,
int tile_map[18][32],
int tile_map_niveau_1[18][110],
SDL_Rect * rectangle_tile,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
SDL_Texture ** texture_image_perso_gagnant,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Rect * rectangle_demande,
SDL_Texture ** texture_image_nuage_1,
SDL_Texture ** texture_image_nuage_2,
SDL_Color couleurNoire,
SDL_Texture ** texture_image_fin_premiers_niveaux,
int * avancer,
int * reculer,
int * sauter,
int * position_avant_saut,
int * saut,
int * tombe,
int * position_x_initiale,
int * position_y_initiale,
int * position_x,
int * position_y,
int * largeur,
int * hauteur,
int * largeur_tile,
int * hauteur_tile,
page_t * page_active,
itemMenu * itemsDemandeSauvegarde,
SDL_Keycode * touche_descendre,
SDL_Keycode * touche_interagir,
barreDeSon * barre_de_son,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 300 du fichier fonctions\_niveau\_1.c.

## 4.36 fonctions\_niveau\_1.h

[Aller à la documentation de ce fichier.](#)

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <time.h>
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_ttf.h>
00006 #include <SDL2/SDL_image.h>
00007 #include <SDL2/SDL_mixer.h>
00008
00009 /* Fonction qui permet d'initialiser les différents objets du niveau 4 */
00010 void initialisation_objets_niveau_1(SDL_Renderer **renderer, SDL_Surface **surface,
00011                                     SDL_Texture **texture_image_sol_surface_niveau_1, SDL_Texture
00012                                     **texture_image_sol_profondeur_niveau_1,
00013                                     SDL_Texture **texture_image_fond_niveau_1, SDL_Texture
00014                                     **texture_image_nuage_1, SDL_Texture **texture_image_nuage_2);
00015
00016 /* Fonction qui permet de créer l'étage 1 */
00017 void chargement_niveau_1(int *position_x, int *position_y, int *position_x_initiale, int
00018                          *position_y_initiale, int tile_map_niveau_1[18][110]);
00019
00020 /* Fonction qui permet de mettre à jour le rendu du niveau 4 */
00021 void mise_a_jour_rendu_niveau_1(SDL_Renderer **renderer, SDL_Texture **texture_image_sol_surface,
00022                                  SDL_Texture **texture_image_sol_profondeur, SDL_Texture **texture_image_fond,
00023                                  SDL_Texture **texture, SDL_Rect *rectangle_tile, SDL_Rect
00024                                  *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
00025                                  SDL_Texture **texture_image_personnage, SDL_Rect
00026                                  *rectangle_personnage, SDL_Texture **texture_image_monstre_terrestre, SDL_Texture
00027                                  **texture_image_monstre_volant,
00028                                  SDL_Texture **texture_image_pique, niveaux *avancee_niveaux,
00029                                  SDL_Texture **texture_image_fin_premiers_niveaux,
00030                                  int position_x, int position_y, int tile_map[18][32], int secret,
00031                                  SDL_Texture **texture_image_nuage_1, SDL_Texture **texture_image_nuage_2,
00032                                  int largeur, int hauteur, int largeur_tile, int hauteur_tile,
00033                                  SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix);
00034
00035 /* Fonction qui permet de gérer toutes les possibilités qui sont possiblent dans le niveau 4 */
00036 void niveau_1(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
00037               *programme_lance, SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix,
00038               SDL_Texture **texture, SDL_Rect *rectangle_plein_ecran, SDL_Texture
00039               **texture_image_plein_ecran, SDL_bool *plein_ecran,
00040               SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, SDL_Texture
00041               **texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant,
00042               SDL_Texture **texture_image_sol_surface, SDL_Texture **texture_image_sol_profondeur,
00043               SDL_Texture **texture_image_fond,
00044               SDL_Texture **texture_image_pique, niveaux *avancee_niveaux, int *mouvement_monstre,
00045               SDL_Surface **surface, int collectibles_intermediaires[3], time_t *timestamp,
00046               SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
00047               SDL_Keycode *touche_sauter_monter, int *decalage, int *secret_1, int *secret_2,
00048               int tile_map[18][32], int tile_map_niveau_1[18][110], SDL_Rect *rectangle_tile,
00049               itemMenu *itemsDemandeQuitter, int tailleDemande, SDL_Texture
00050               **texture_image_perso_gagnant,
00051               SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande, SDL_Texture
00052               **texture_image_nuage_1, SDL_Texture **texture_image_nuage_2,
00053               SDL_Color couleurNoire, SDL_Texture **texture_image_fin_premiers_niveaux,
00054               int *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut, int
00055               *tombe,
00056               int *position_x_initiale, int *position_y_initiale, int *position_x, int *position_y,
00057               int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page_t *page_active,
00058               itemMenu *itemsDemandeSauvegarde, SDL_Keycode *touche_descendre, SDL_Keycode
00059               *touche_interagir, barreDeSon *barre_de_son, itemMenu *pseudo,
00060               modes_t *modeActif, personnage_t *personnageActif, position_t *positionActive, int
00061               tailleNiveaux,
00062               time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
00063               avancee_succes_intermediaires[10]);

```

## 4.37 Référence du fichier fichiers\_h/fonctions\_niveau\_2.h

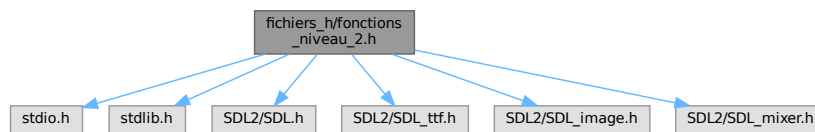
```

#include <stdio.h>
#include <stdlib.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>

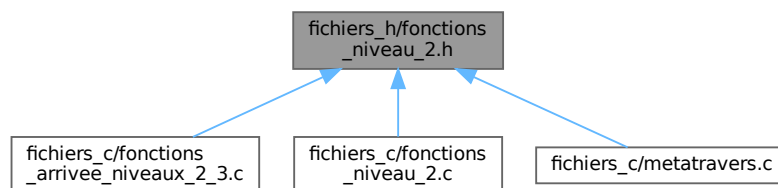
```

```
#include <SDL2/SDL_mixer.h>
```

Graphe des dépendances par inclusion de fonctions\_niveau\_2.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [salon\\_arrivee\\_niveaux\\_2\\_3](#) (int \*position\_x, int \*position\_y, int tile\_map[18][32], [page\\_t](#) page\_active)  
*Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.*
- void [initialisation\\_objets\\_niveau\\_2](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_fond\_niveau\_2, SDL\_Texture \*\*texture\_image\_dossier\_niveau\_2, SDL\_Texture \*\*texture\_image\_sol\_niveau\_2, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_courant, SDL\_Texture \*\*texture\_image\_mur\_termine)  
*Fonction qui permet d'initialiser les différents objets du niveau 2.*
- void [mini\\_jeu\\_1\\_niveau\\_2](#) (int \*position\_x, int \*position\_y, int tile\_map[19][27])  
*Fonction qui permet d'initialiser le premier mini-jeu du niveau 2.*
- int [verification\\_chemin](#) (int x, int y, int x\_precedent, int y\_precedent, int tilemap[19][27], int x\_arrivee, int y\_arrivee)  
*Fonction de vérification du chemin.*
- int [mise\\_a\\_jour\\_bordures\\_niveau\\_2](#) (SDL\_Renderer \*renderer, SDL\_Texture \*texture\_image\_mur\_termine, int tilemap[19][27], int x\_tile, int y\_tile, int x, int y, SDL\_Rect \*rectangle\_tile, int largeur\_tile, int hauteur\_tile)  
*Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la 9 du labyrinthe.*
- void [mise\\_a\\_jour\\_mini\\_jeu\\_1\\_niveau\\_2](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, int position\_x, int position\_y, int tile\_map\_mini\_jeu\_niveau\_2[19][27], int largeur, int hauteur, int largeur\_tile, int hauteur\_tile, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_courant, SDL\_Texture \*\*texture\_image\_mur\_termine)
- void [mini\\_jeu\\_2\\_niveau\\_2](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], int mode\_difficile)

Fonction qui permet d'initialiser le second mini-jeu du niveau 2.

- void [mise\\_a\\_jour\\_mini\\_jeu\\_2\\_niveau\\_2](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int mini\_jeu\_termine, int position\_x, int position\_y, int tile\_map[18][32], SDL\_Texture \*\*texture\_image\_porte, [niveaux](#) \*avancee\_niveaux, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile)
- void [mini\\_jeux\\_niveau\\_2](#) (SDL\_Event \*event, SDL\_Renderer \*\*renderer, SDL\_Window \*\*window, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_sol, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_porte, [niveaux](#) \*avancee\_niveaux, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, int \*mini\_jeu, int \*mini\_jeu\_1\_termine, int \*mini\_jeu\_2\_termine, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int mini\_jeu\_termine, int \*position\_x, int \*position\_y, int tile\_map[18][32], int tile\_map\_mini\_jeu\_niveau\_2[19][27], SDL\_Texture \*\*texture\_image\_monstre\_terrestre, SDL\_Texture \*\*texture\_image\_monstre\_volant, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_interagir, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, int \*valide, SDL\_Texture \*\*texture\_image\_pipe\_vertical, SDL\_Texture \*\*texture\_image\_pipe\_horizontal, SDL\_Texture \*\*texture\_image\_pipe\_haut\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_droit, SDL\_Texture \*\*texture\_image\_pipe\_bas\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_haut\_gauche, SDL\_Texture \*\*texture\_image\_pipe\_courant, SDL\_Rect \*rectangle\_demande, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemande, int collectibles\_intermediaires[3], SDL\_Texture \*\*texture\_image\_mur\_termine, [page\\_t](#) \*page\_active, Mix\_Music \*\*musique, int \*avancer, int \*reculer, int \*sauter, int \*saut, int \*tombe, [itemMenu](#) \*itemsDemandeSauvegarde, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [modes\\_t](#) \*modeActif, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

## 4.37.1 Documentation des fonctions

### 4.37.1.1 initialisation\_objets\_niveau\_2()

```
void initialisation_objets_niveau_2 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_fond_niveau_2,
    SDL_Texture ** texture_image_dossier_niveau_2,
    SDL_Texture ** texture_image_sol_niveau_2,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Texture ** texture_image_pipe_vertical,
    SDL_Texture ** texture_image_pipe_horizontal,
    SDL_Texture ** texture_image_pipe_haut_droit,
    SDL_Texture ** texture_image_pipe_bas_droit,
    SDL_Texture ** texture_image_pipe_bas_gauche,
    SDL_Texture ** texture_image_pipe_haut_gauche,
    SDL_Texture ** texture_image_pipe_courant,
    SDL_Texture ** texture_image_mur_termine )
```

Fonction qui permet d'initialiser les différents objets du niveau 2.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Surface SDL.

## Paramètres

<i>texture_image_fond_niveau_2</i>	Texture de l'image de fond du niveau 2.
<i>texture_image_dossier_niveau_2</i>	Texture de l'image du dossier du niveau 2.
<i>texture_image_sol_niveau_2</i>	Texture de l'image du sol du niveau 2.
<i>texture_image_mur_mini_jeu</i>	Texture de l'image du mur du mini-jeu.
<i>texture_image_pipe_vertical</i>	Texture de l'image du tuyau vertical.
<i>texture_image_pipe_horizontal</i>	Texture de l'image du tuyau horizontal.
<i>texture_image_pipe_haut_droit</i>	Texture de l'image du tuyau haut droit.
<i>texture_image_pipe_bas_droit</i>	Texture de l'image du tuyau bas droit.
<i>texture_image_pipe_bas_gauche</i>	Texture de l'image du tuyau bas gauche.
<i>texture_image_pipe_haut_gauche</i>	Texture de l'image du tuyau haut gauche.
<i>texture_image_pipe_courant</i>	Texture de l'image du tuyau courant.
<i>texture_image_mur_termine</i>	Texture de l'image du mur terminé.

## Voir également

[chargement\\_image](#)

Définition à la ligne 27 du fichier [fonctions\\_niveau\\_2.c](#).

## 4.37.1.2 mini\_jeu\_1\_niveau\_2()

```
void mini_jeu_1_niveau_2 (
    int * position_x,
    int * position_y,
    int tile_map[19][27] )
```

Fonction qui permet d'initialiser le premier mini-jeu du niveau 2.

## Paramètres

<i>position_x</i>	position horizontale du personnage sur le tile map
<i>position_y</i>	position verticale du personnage sur le tile map
<i>tile_map</i>	map ou se trouve le personnage

Définition à la ligne 59 du fichier [fonctions\\_niveau\\_2.c](#).

## 4.37.1.3 mini\_jeu\_2\_niveau\_2()

```
void mini_jeu_2_niveau_2 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
```

```
int tile_map[18][32],
int mode_difficile )
```

Fonction qui permet d'initialiser le second mini-jeu du niveau 2.

#### Paramètres

<i>position_x</i>	position verticale du joueur à l'apparition dans le niveau
<i>position_y</i>	position horizontale du joueur à l'apparition dans le niveau
<i>position_x_initiale</i>	position du joueur verticale si il venait à revenir dans le niveau ou si il venait à mourir
<i>position_y_initiale</i>	position du joueur horizontale si il venait à revenir dans le niveau ou si il venait à mourir
<i>tile_map</i>	map ou se trouve le personnage
<i>mode_difficile</i>	booléen indiquant la présence du mode difficile

Définition à la ligne 513 du fichier [fonctions\\_niveau\\_2.c](#).

#### 4.37.1.4 mini\_jeux\_niveau\_2()

```
void mini_jeux_niveau_2 (
    SDL_Event * event,
    SDL_Renderer ** renderer,
    SDL_Window ** window,
    SDL_bool * programme_lance,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_porte,
    niveaux * avancee_niveaux,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    int * mini_jeu,
    int * mini_jeu_1_termine,
    int * mini_jeu_2_termine,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int mini_jeu_termine,
    int * position_x,
    int * position_y,
    int tile_map[18][32],
    int tile_map_mini_jeu_niveau_2[19][27],
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    int * largeur,
    int * hauteur,
    int * largeur_tile,
    int * hauteur_tile,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_interagir,
```

```

SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
int * valide,
SDL_Texture ** texture_image_pipe_vertical,
SDL_Texture ** texture_image_pipe_horizontal,
SDL_Texture ** texture_image_pipe_haut_droit,
SDL_Texture ** texture_image_pipe_bas_droit,
SDL_Texture ** texture_image_pipe_bas_gauche,
SDL_Texture ** texture_image_pipe_haut_gauche,
SDL_Texture ** texture_image_pipe_courant,
SDL_Rect * rectangle_demande,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Color couleurNoire,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
int collectibles_intermediaires[3],
SDL_Texture ** texture_image_mur_termine,
page_t * page_active,
Mix_Music ** musique,
int * avancer,
int * reculer,
int * sauter,
int * saut,
int * tombe,
itemMenu * itemsDemandeSauvegarde,
barreDeSon * barre_de_son,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 746 du fichier [fonctions\\_niveau\\_2.c](#).

#### 4.37.1.5 mise\_a\_jour\_bordures\_niveau\_2()

```

int mise_a_jour_bordures_niveau_2 (
    SDL_Renderer * renderer,
    SDL_Texture * texture_image_mur_termine,
    int tilemap[19][27],
    int x_tile,
    int y_tile,
    int x,
    int y,
    SDL_Rect * rectangle_tile,
    int largeur_tile,
    int hauteur_tile )

```

Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la 9 du labyrinthe.

## Paramètres

<i>render</i>	Renderer SDL.
<i>texture_image_mur_termine</i>	Texture de l'image du mur terminé.
<i>tilemap</i>	Carte de tuiles du niveau 2.
<i>x_tile</i>	Position x de la tuile à mettre à jour.
<i>y_tile</i>	Position y de la tuile à mettre à jour.
<i>x</i>	Position x de la tuile dans l'écran.
<i>y</i>	Position y de la tuile dans l'écran.
<i>rectangle_tile</i>	Rectangle pour chaque tuile.
<i>largeur_tile</i>	Largeur d'une tuile.
<i>hauteur_tile</i>	Hauteur d'une tuile.

## Renvoie

appel récursif pour mettre les différents rectangle à jour jusqu'à la fin (appel se finissant par un 0)

Définition à la ligne [249](#) du fichier [fonctions\\_niveau\\_2.c](#).

## 4.37.1.6 mise\_a\_jour\_mini\_jeu\_1\_niveau\_2()

```
void mise_a_jour_mini_jeu_1_niveau_2 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    int position_x,
    int position_y,
    int tile_map_mini_jeu_niveau_2[19][27],
    int largeur,
    int hauteur,
    int largeur_tile,
    int hauteur_tile,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Texture ** texture_image_pipe_vertical,
    SDL_Texture ** texture_image_pipe_horizontal,
    SDL_Texture ** texture_image_pipe_haut_droit,
    SDL_Texture ** texture_image_pipe_bas_droit,
    SDL_Texture ** texture_image_pipe_bas_gauche,
    SDL_Texture ** texture_image_pipe_haut_gauche,
    SDL_Texture ** texture_image_pipe_courant,
    SDL_Texture ** texture_image_mur_termine )
```

Définition à la ligne [322](#) du fichier [fonctions\\_niveau\\_2.c](#).



#### 4.37.1.7 mise\_a\_jour\_mini\_jeu\_2\_niveau\_2()

```
void mise_a_jour_mini_jeu_2_niveau_2 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_sol,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    SDL_Texture ** texture_image_monstre_terrestre,
    SDL_Texture ** texture_image_monstre_volant,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int mini_jeu_termine,
    int position_x,
    int position_y,
    int tile_map[18][32],
    SDL_Texture ** texture_image_porte,
    niveaux * avancee_niveaux,
    int largeur,
    int hauteur,
    int largeur_tile,
    int hauteur_tile )
```

Définition à la ligne 583 du fichier [fonctions\\_niveau\\_2.c](#).

#### 4.37.1.8 salon\_arrivee\_niveaux\_2\_3()

```
void salon_arrivee_niveaux_2_3 (
    int * position_x,
    int * position_y,
    int tile_map[18][32],
    page_t page_active )
```

Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.

##### Paramètres

<i>position_x</i>	pointeur sur la position du personnage sur l'horizontal du tilemap
<i>position_y</i>	pointeur sur la position du perosnnage sur la verticale du tilemap
<i>tile_map</i>	Matrice représentant la map ou se trouve le personnage
<i>page_active</i>	Enumération représentant sur quel page on se trouve

Définition à la ligne 21 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

#### 4.37.1.9 verification\_chemin()

```
int verification_chemin (
    int x,
```

```

    int y,
    int x_precedent,
    int y_precedent,
    int tilemap[19][27],
    int x_arrivee,
    int y_arrivee )

```

Fonction de vérification du chemin.

#### Paramètres

<i>x</i>	position actuelle verticale dans la vérification du chemin
<i>y</i>	position actuelle horizontale dans la vérification du chemin
<i>x_precedent</i>	position précédente à la position actuelle de x
<i>y_precedent</i>	position précédente à la position actuelle de y
<i>tilemap</i>	map à vérifier
<i>x_arrivee</i>	coordonnée x ou se trouve la sortie
<i>y_arrivee</i>	coordonnée y ou se trouve la sortie

#### Renvoie

booléen (1 si le chemin est bon sinon 0)

Définition à la ligne 115 du fichier [fonctions\\_niveau\\_2.c](#).

## 4.38 fonctions\_niveau\_2.h

[Aller à la documentation de ce fichier.](#)

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <SDL2/SDL.h>
00004 #include <SDL2/SDL_ttf.h>
00005 #include <SDL2/SDL_image.h>
00006 #include <SDL2/SDL_mixer.h>
00007
00008 /* Squelette de la fonction salon_arrivee_niveaux_2_3 */
00009 void salon_arrivee_niveaux_2_3(int *position_x, int *position_y, int tile_map[18][32], page_t
page_active);
00010
00011 /* Squelette de la fonction initialisation_objets_niveau_2 */
00012 void initialisation_objets_niveau_2(SDL_Renderer **renderer, SDL_Surface **surface,
00013                                     SDL_Texture **texture_image_fond_niveau_2, SDL_Texture
**texture_image_dossier_niveau_2,
00014                                     SDL_Texture **texture_image_sol_niveau_2, SDL_Texture
**texture_image_mur_mini_jeu,
00015                                     SDL_Texture **texture_image_pipe_vertical, SDL_Texture
**texture_image_pipe_horizontal,
00016                                     SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
**texture_image_pipe_bas_droit,
00017                                     SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
**texture_image_pipe_haut_gauche,
00018                                     SDL_Texture **texture_image_pipe_courant,
00019                                     SDL_Texture **texture_image_mur_termine);
00020
00021 /* Squelette de la fonction mini_jeu_1_niveau_2 */
00022 void mini_jeu_1_niveau_2(int *position_x, int *position_y, int tile_map[19][27]);
00023
00024 /* Squelette de la fonction verification_chemin */
00025 int verification_chemin(int x, int y, int x_precedent, int y_precedent, int tilemap[19][27], int
x_arrivee, int y_arrivee);
00026
00027 /* Squelette de la fonction mise_a_jour_bordures */
00028 int mise_a_jour_bordures_niveau_2(SDL_Renderer* renderer, SDL_Texture* texture_image_mur_termine, int
tilemap[19][27], int x_tile, int y_tile, int x, int y,
00029                                     SDL_Rect *rectangle_tile, int largeur_tile, int hauteur_tile);
00030

```

```

00031 /* Squelette de la fonction mise_a_jour_mini_jeu_1_niveau_2 */
00032 void mise_a_jour_mini_jeu_1_niveau_2(SDL_Renderer **renderer, SDL_Texture **texture_image_croix,
SDL_Rect *rectangle_croix,
00033                                     SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00034                                     SDL_Texture **texture, SDL_Rect *rectangle_tile,
00035                                     int position_x, int position_y, int
tile_map_mini_jeu_niveau_2[19][27],
00036                                     int largeur, int hauteur, int largeur_tile, int hauteur_tile,
00037                                     SDL_Texture **texture_image_mur_mini_jeu,
00038                                     SDL_Texture **texture_image_pipe_vertical, SDL_Texture
**texture_image_pipe_horizontal,
00039                                     SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
**texture_image_pipe_bas_droit,
00040                                     SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
**texture_image_pipe_haut_gauche,
00041                                     SDL_Texture **texture_image_pipe_courant,
00042                                     SDL_Texture **texture_image_mur_termine);
00043
00044 /* Squelette de la fonction mini_jeu_2_niveau_2 */
00045 void mini_jeu_2_niveau_2(int *position_x, int *position_y, int *position_x_initiale, int
*position_y_initiale, int tile_map[18][32], int mode_difficile);
00046
00047 /* Squelette de la fonction mise_a_jour_mini_jeu_2_niveau_2 */
00048 void mise_a_jour_mini_jeu_2_niveau_2(SDL_Renderer **renderer, SDL_Texture **texture_image_fond,
SDL_Texture **texture_image_sol, SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix,
00049                                     SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00050                                     SDL_Texture **texture, SDL_Rect *rectangle_tile, SDL_Texture
**texture_image_monstre_terrestre, SDL_Texture **texture_image_monstre_volant,
00051                                     SDL_Texture **texture_image_personnage, SDL_Rect
*rectangle_personnage, int mini_jeu_termine,
00052                                     int position_x, int position_y, int tile_map[18][32], SDL_Texture
**texture_image_porte, niveaux *avancee_niveaux,
00053                                     int largeur, int hauteur, int largeur_tile, int hauteur_tile);
00054
00055 /* Squelette de la fonction mini_jeux_niveau_2 */
00056 void mini_jeux_niveau_2(SDL_Event *event, SDL_Renderer **renderer, SDL_Window **window, SDL_bool
*programme_lance, SDL_Texture **texture_image_fond, SDL_Texture **texture_image_sol,
00057                                     SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
SDL_bool *plein_ecran, SDL_Texture **texture_image_porte, niveaux *avancee_niveaux,
00058                                     SDL_Texture **texture, SDL_Rect *rectangle_tile, int *mini_jeu, int
*mini_jeu_1_termine, int *mini_jeu_2_termine,
00059                                     SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage, int
mini_jeu_termine,
00060                                     int *position_x, int *position_y, int tile_map[18][32], int
tile_map_mini_jeu_niveau_2[19][27], SDL_Texture **texture_image_monstre_terrestre, SDL_Texture
**texture_image_monstre_volant,
00061                                     int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, SDL_Texture
**texture_image_croix, SDL_Rect *rectangle_croix,
00062                                     SDL_Texture **texture_image_mur_mini_jeu, SDL_Keycode *touche_aller_a_droite,
SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_interagir,
00063                                     SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, int *valide,
SDL_Texture **texture_image_pipe_vertical, SDL_Texture
00064                                     **texture_image_pipe_horizontal,
SDL_Texture **texture_image_pipe_haut_droit, SDL_Texture
00065                                     **texture_image_pipe_bas_droit,
SDL_Texture **texture_image_pipe_bas_gauche, SDL_Texture
00066                                     **texture_image_pipe_haut_gauche,
SDL_Texture **texture_image_pipe_courant, SDL_Rect *rectangle_demande,
00067                                     SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
SDL_Color couleurNoire,
00069                                     itemMenu *itemsDemandeQuitter, int tailleDemande, int
collectibles_intermediaires[3],
00070                                     SDL_Texture **texture_image_mur_termine, page_t *page_active, Mix_Music
**musique,
00071                                     int *avancer, int *reculer, int *sauter, int *saut, int *tombe,
itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
00072                                     modes_t *modeActif, personnage_t *personnageActif, position_t *positionActive,
int tailleNiveaux,
00074                                     time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
avancee_succes_intermediaires[10]);

```

## 4.39 Référence du fichier fichiers\_h/fonctions\_niveau\_3.h

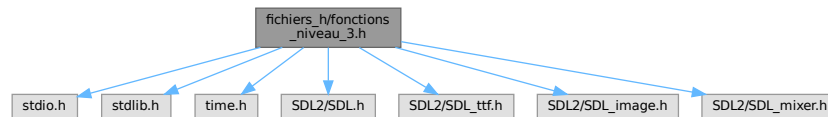
```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>

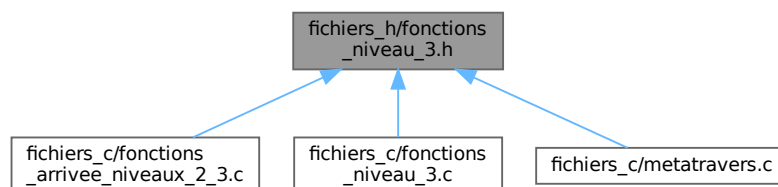
```

```
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
```

Graphe des dépendances par inclusion de fonctions\_niveau\_3.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [salon\\_arrivee\\_niveaux\\_2\\_3](#) (int \*position\_x, int \*position\_y, int tile\_map[18][32], [page\\_t](#) page\_active)  
*Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.*
- void [initialisation\\_objets\\_niveau\\_3](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_fond\_niveau\_3, SDL\_Texture \*\*texture\_image\_dossier\_niveau\_3, SDL\_Texture \*\*texture\_image\_sol\_niveau\_3, SDL\_Texture \*\*barre\_windows\_1, SDL\_Texture \*\*barre\_windows\_2, SDL\_Texture \*\*barre\_windows\_3, SDL\_Texture \*\*barre\_windows\_4, SDL\_Texture \*\*texture\_image\_puzzle, SDL\_Texture \*\*texture\_image\_sol\_labyrinthe, SDL\_Texture \*\*texture\_image\_bordure\_labyrinthe, SDL\_Texture \*\*texture\_image\_fin\_labyrinthe)  
*Fonction qui permet d'initialiser les différents objets du niveau 2.*
- SDL\_Rect [rectangle\\_piece\\_aleatoire](#) (int largeur, int hauteur)  
*Fonction pour obtenir un rectangle représentant une pièce de puzzle aléatoire.*
- void [mise\\_a\\_jour\\_mini\\_jeu\\_1\\_niveau\\_3](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_puzzle, SDL\_Rect rectangle\_piece[45], SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix)
- int [piece\\_proche\\_position\\_correcte](#) (SDL\_Rect rectangle\_piece, SDL\_Rect rectangle\_correct)  
*Fonction pour vérifier si une pièce est proche de sa position correcte.*
- int [verification\\_puzzle\\_fini](#) (const int piece\_bloquee[])  
*Fonction pour vérifier si toutes les pièces du puzzle sont bloquées (à leur position correcte)*
- void [mini\\_jeu\\_2\\_niveau\\_3](#) (int \*position\_x, int \*position\_y, int \*bloc\_x, int \*bloc\_y, int tile\_map[24][32])  
*Fonction qui permet d'initialiser le second mini-jeu du niveau 3.*
- void [traitement\\_touches](#) (int \*position\_x, int \*position\_y, int \*bloc\_x, int \*bloc\_y, int tilemap[24][32], int direction)  
*Fonction pour traiter les commandes utilisateur.*
- int [mise\\_a\\_jour\\_bordures\\_niveau\\_3](#) (SDL\_Renderer \*renderer, SDL\_Texture \*texture\_image\_mur\_termine, int tilemap[24][32], int x\_tile, int y\_tile, SDL\_Rect \*rectangle\_tile, int largeur\_tile, int hauteur\_tile)  
*Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la fin du labyrinthe.*

- void `mise_a_jour_mini_jeu_2_niveau_3` (SDL\_Renderer \*\*renderer, `modes_t` \*modeActif, SDL\_Texture \*\*texture\_image\_sol\_labyrinthe, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_écran, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Texture \*\*texture\_image\_bordure\_labyrinthe, SDL\_Texture \*\*texture\_image\_mur\_termine, SDL\_Texture \*\*texture\_image\_fin\_labyrinthe, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_tile, int bloc\_x, int bloc\_y, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, int position\_x, int position\_y, int tile\_map\_mini\_jeu\_niveau\_3[24][32], `niveaux` \*avancee\_niveaux, int largeur\_tile, int hauteur\_tile)
- void `mini_jeux_niveau_3` (SDL\_Event \*event, SDL\_Renderer \*\*renderer, SDL\_Window \*\*window, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, `niveaux` \*avancee\_niveaux, int tile\_map[18][32], SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int \*mini\_jeu, int \*mini\_jeu\_1\_termine, int \*mini\_jeu\_2\_termine, int \*position\_x, int \*position\_y, SDL\_Texture \*\*texture, int \*largeur, int \*hauteur, SDL\_Rect \*rectangle\_demande, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, `itemMenu` \*itemsDemandeQuitter, int tailleDemande, int collectibles\_intermediaires[3], `page_t` \*page\_active, SDL\_Rect \*rectangle\_tile, int \*largeur\_tile, int \*hauteur\_tile, int \*avancer, int \*reculer, int \*sauter, int \*saut, int \*tombe, SDL\_Rect rectangle\_piece[45], int piece\_bloquee[45], SDL\_Rect rectangle\_emplacement\_piece[45], int \*piece\_selectionnee, int \*decalage\_x, int \*decalage\_y, SDL\_Texture \*\*texture\_image\_puzzle, int tile\_map\_mini\_jeu\_niveau\_3[24][32], int \*descendre, int \*interagir, int \*bloc\_x, int \*bloc\_y, SDL\_Texture \*\*texture\_image\_sol\_labyrinthe, SDL\_Texture \*\*texture\_image\_bordure\_labyrinthe, SDL\_Texture \*\*texture\_image\_fin\_labyrinthe, Mix\_Music \*\*musique, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_mur\_termine, SDL\_Texture \*\*texture\_image\_mur\_mini\_jeu, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_interagir, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, `modes_t` \*modeActif, `itemMenu` \*itemsDemandeSauvegarde, `barreDeSon` \*barre\_de\_son, `itemMenu` \*pseudo, `personnage_t` \*personnageActif, `position_t` \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

## 4.39.1 Documentation des fonctions

### 4.39.1.1 initialisation\_objets\_niveau\_3()

```
void initialisation_objets_niveau_3 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_fond_niveau_3,
    SDL_Texture ** texture_image_dossier_niveau_3,
    SDL_Texture ** texture_image_sol_niveau_3,
    SDL_Texture ** barre_windows_1,
    SDL_Texture ** barre_windows_2,
    SDL_Texture ** barre_windows_3,
    SDL_Texture ** barre_windows_4,
    SDL_Texture ** texture_image_puzzle,
    SDL_Texture ** texture_image_sol_labyrinthe,
    SDL_Texture ** texture_image_bordure_labyrinthe,
    SDL_Texture ** texture_image_fin_labyrinthe )
```

Fonction qui permet d'initialiser les différents objets du niveau 2.

#### Paramètres

<code>renderer</code>	Pointeur vers le renderer SDL.
<code>surface</code>	Pointeur vers la surface SDL.
<code>texture_image_fond_niveau_3</code>	Texture de l'image de fond du niveau 3.
<code>texture_image_dossier_niveau_3</code>	Texture de l'image du dossier pour le niveau 3.

## Paramètres

<i>texture_image_sol_niveau_3</i>	Texture de l'image du sol du niveau 3.
<i>barre_windows_1</i>	Texture de la barre Windows 1.
<i>barre_windows_2</i>	Texture de la barre Windows 2.
<i>barre_windows_3</i>	Texture de la barre Windows 3.
<i>barre_windows_4</i>	Texture de la barre Windows 4.
<i>texture_image_puzzle</i>	Texture de l'image du puzzle.
<i>texture_image_sol_labyrinthe</i>	Texture de l'image du sol du labyrinthe.
<i>texture_image_bordure_labyrinthe</i>	Texture de l'image de la bordure du labyrinthe.
<i>texture_image_fin_labyrinthe</i>	Texture de l'image de fin du labyrinthe.

## Voir également

[chargement\\_image](#)

Définition à la ligne 27 du fichier [fonctions\\_niveau\\_3.c](#).

## 4.39.1.2 mini\_jeu\_2\_niveau\_3()

```
void mini_jeu_2_niveau_3 (
    int * position_x,
    int * position_y,
    int * bloc_x,
    int * bloc_y,
    int tile_map[24][32] )
```

Fonction qui permet d'initialiser le second mini-jeu du niveau 3.

## Paramètres

<i>position↔ _x</i>	position x du personnage
<i>position↔ _y</i>	position y du personnage
<i>bloc_x</i>	position x du bloc à déplacer
<i>bloc_y</i>	position y du bloc à déplacer
<i>tile_map</i>	map ou se trouve le personnage

Définition à la ligne 147 du fichier [fonctions\\_niveau\\_3.c](#).

## 4.39.1.3 mini\_jeux\_niveau\_3()

```
void mini_jeux_niveau_3 (
    SDL_Event * event,
    SDL_Renderer ** renderer,
    SDL_Window ** window,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_plein_ecran,
```

```
SDL_Texture ** texture_image_plein_ecran,
SDL_bool * plein_ecran,
niveaux * avancee_niveaux,
int tile_map[18][32],
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
int * mini_jeu,
int * mini_jeu_1_termine,
int * mini_jeu_2_termine,
int * position_x,
int * position_y,
SDL_Texture ** texture,
int * largeur,
int * hauteur,
SDL_Rect * rectangle_demande,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Color couleurNoire,
itemMenu * itemsDemandeQuitter,
int tailleDemande,
int collectibles_intermediaires[3],
page_t * page_active,
SDL_Rect * rectangle_tile,
int * largeur_tile,
int * hauteur_tile,
int * avancer,
int * reculer,
int * sauter,
int * saut,
int * tombe,
SDL_Rect rectangle_piece[45],
int piece_bloquee[45],
SDL_Rect rectangle_emplacement_piece[45],
int * piece_selectionnee,
int * decalage_x,
int * decalage_y,
SDL_Texture ** texture_image_puzzle,
int tile_map_mini_jeu_niveau_3[24][32],
int * descendre,
int * interagir,
int * bloc_x,
int * bloc_y,
SDL_Texture ** texture_image_sol_labyrinthe,
SDL_Texture ** texture_image_bordure_labyrinthe,
SDL_Texture ** texture_image_fin_labyrinthe,
Mix_Music ** musique,
SDL_Texture ** texture_image_personnage,
SDL_Rect * rectangle_personnage,
SDL_Texture ** texture_image_mur_termine,
SDL_Texture ** texture_image_mur_mini_jeu,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_interagir,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
modes_t * modeActif,
itemMenu * itemsDemandeSauvegarde,
```

```

barreDeSon * barre_de_son,
itemMenu * pseudo,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 682 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.39.1.4 mise\_a\_jour\_bordures\_niveau\_3()

```

int mise_a_jour_bordures_niveau_3 (
    SDL_Renderer * renderer,
    SDL_Texture * texture_image_mur_termine,
    int tilemap[24][32],
    int x_tile,
    int y_tile,
    SDL_Rect * rectangle_tile,
    int largeur_tile,
    int hauteur_tile )

```

Fonction pour mettre à jour les tuiles de bordure lorsque le bloc atteint la fin du labyrinthe.

##### Paramètres

<i>renderer</i>	rendu de la fenêtre
<i>texture_image_mur_termine</i>	Texture du mur quand une partie ou tout le niveau est terminé
<i>tilemap</i>	map ou se trouve le personnage
<i>x_tile</i>	position x de la tuile
<i>y_tile</i>	position y de la tuile
<i>rectangle_tile</i>	rectangle à déplacer
<i>largeur_tile</i>	largeur du rectangle
<i>hauteur_tile</i>	hauteur du rectangle

##### Renvoie

appel récursif pour mettre à jour les différents bloc (termine l'appel récursif à 0)

Définition à la ligne 332 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.39.1.5 mise\_a\_jour\_mini\_jeu\_1\_niveau\_3()

```

void mise_a_jour_mini_jeu_1_niveau_3 (
    SDL_Renderer ** renderer,
    SDL_Texture ** texture_image_puzzle,
    SDL_Rect rectangle_piece[45],
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix )

```

Définition à la ligne 76 du fichier [fonctions\\_niveau\\_3.c](#).



#### 4.39.1.6 mise\_a\_jour\_mini\_jeu\_2\_niveau\_3()

```
void mise_a_jour_mini_jeu_2_niveau_3 (
    SDL_Renderer ** renderer,
    modes_t * modeActif,
    SDL_Texture ** texture_image_sol_labyrinthe,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Texture ** texture_image_mur_mini_jeu,
    SDL_Texture ** texture_image_bordure_labyrinthe,
    SDL_Texture ** texture_image_mur_termine,
    SDL_Texture ** texture_image_fin_labyrinthe,
    SDL_Texture ** texture_image_croix,
    SDL_Rect * rectangle_croix,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_tile,
    int bloc_x,
    int bloc_y,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    int position_x,
    int position_y,
    int tile_map_mini_jeu_niveau_3[24][32],
    niveaux * avancee_niveaux,
    int largeur_tile,
    int hauteur_tile )
```

Définition à la ligne 400 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.39.1.7 piece\_proche\_position\_correcte()

```
int piece_proche_position_correcte (
    SDL_Rect rectangle_piece,
    SDL_Rect rectangle_correct )
```

Fonction pour vérifier si une pièce est proche de sa position correcte.

##### Paramètres

<i>rectangle_piece</i>	rectangle représentant la pièce de puzzle à vérifier
<i>rectangle_correct</i>	rectangle représentant la bonne position de la pièce

##### Renvoie

booléen de si c'est proche de la bonne position (1 si succès sinon 0)

Définition à la ligne 113 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.39.1.8 rectangle\_piece\_aleatoire()

```
SDL_Rect rectangle_piece_aleatoire (
    int largeur,
    int hauteur )
```

Fonction pour obtenir un rectangle représentant une pièce de puzzle aléatoire.

## Paramètres

<i>largeur</i>	représente la largeur du rectangle de la fenêtre ou se trouve le mini-jeu
<i>hauteur</i>	représente la hauteur du rectangle de la fenêtre ou se trouve le mini-jeu

Définition à la ligne 56 du fichier [fonctions\\_niveau\\_3.c](#).

4.39.1.9 **salon\_arrivee\_niveaux\_2\_3()**

```
void salon_arrivee_niveaux_2_3 (
    int * position_x,
    int * position_y,
    int tile_map[18][32],
    page_t page_active )
```

Fonction qui permet de créer le salon en arrivant dans le niveau 2 ou 3.

## Paramètres

<i>position_x</i>	pointeur sur la position du personnage sur l'horizontal du tilemap
<i>position_y</i>	pointeur sur la position du perosnnage sur la verticale du tilemap
<i>tile_map</i>	Matrice représentant la map ou se trouve le personnage
<i>page_active</i>	Enumération représentant sur quel page on se trouve

Définition à la ligne 21 du fichier [fonctions\\_arrivee\\_niveaux\\_2\\_3.c](#).

4.39.1.10 **traitement\_touches()**

```
void traitement_touches (
    int * position_x,
    int * position_y,
    int * bloc_x,
    int * bloc_y,
    int tilemap[24][32],
    int direction )
```

Fonction pour traiter les commandes utilisateur.

## Paramètres

<i>position↔ _x</i>	pointeur sur la position x du joueur
<i>position↔ _y</i>	pointeur sur la position y du joueur
<i>position↔ _x</i>	pointeur sur la position x du bloc à déplacer
<i>position↔ _y</i>	pointeur sur la position y du bloc à déplacer
<i>tilemap</i>	map ou se trouve le personnage te le bloc
<i>direction</i>	booléen représentant l'action pousser / tirer

Définition à la ligne 206 du fichier [fonctions\\_niveau\\_3.c](#).

#### 4.39.1.11 verification\_puzzle\_fini()

```
int verification_puzzle_fini (
    const int piece_bloquee[] )
```

Fonction pour vérifier si toutes les pièces du puzzle sont bloquées (à leur position correcte)

##### Paramètres

<i>piece_bloquee</i>	tableau des pièces bloquées
----------------------	-----------------------------

##### Renvoie

renvoie un booléen (1 si le puzzle est valide sinon 0)

Définition à la ligne 124 du fichier [fonctions\\_niveau\\_3.c](#).

## 4.40 fonctions\_niveau\_3.h

[Aller à la documentation de ce fichier.](#)

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <time.h>
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_ttf.h>
00006 #include <SDL2/SDL_image.h>
00007 #include <SDL2/SDL_mixer.h>
00008
00009 /* Squelette de la fonction salon_arrivee_niveaux_2_3 */
00010 void salon_arrivee_niveaux_2_3(int *position_x, int *position_y, int tile_map[18][32], page_t
page_active);
00011
00012 /* Squelette de la fonction initialisation_objets_niveau_3 */
00013 void initialisation_objets_niveau_3(SDL_Renderer **renderer, SDL_Surface **surface,
00014     SDL_Texture **texture_image_fond_niveau_3, SDL_Texture
**texture_image_dossier_niveau_3,
00015     SDL_Texture **texture_image_sol_niveau_3, SDL_Texture
**barre_windows_1, SDL_Texture **barre_windows_2,
00016     SDL_Texture **barre_windows_3, SDL_Texture **barre_windows_4,
00017     SDL_Texture **texture_image_puzzle, SDL_Texture
**texture_image_sol_labyrinthe,
00018     SDL_Texture **texture_image_bordure_labyrinthe, SDL_Texture
**texture_image_fin_labyrinthe);
00019
00020 /* Squelette de la fonction rectangle_piece_aleatoire */
00021 SDL_Rect rectangle_piece_aleatoire(int largeur, int hauteur);
00022
00023 /* Squelette de la fonction mise_a_jour_mini_jeu_1_niveau_3 */
00024 void mise_a_jour_mini_jeu_1_niveau_3(SDL_Renderer** renderer, SDL_Texture** texture_image_puzzle,
SDL_Rect rectangle_piece[45],
00025     SDL_Texture **texture_image_croix, SDL_Rect *rectangle_croix);
00026
00027 /* Squelette de la fonction piece_proche_position_correcte */
00028 int piece_proche_position_correcte(SDL_Rect rectangle_piece, SDL_Rect rectangle_correct);
00029
00030 /* Squelette de la fonction verification_puzzle_fini */
00031 int verification_puzzle_fini(const int piece_bloquee[]);
00032
00033 /* Squelette de la fonction mini_jeu_2_niveau_3 */
00034 void mini_jeu_2_niveau_3(int *position_x, int *position_y, int *bloc_x, int *bloc_y, int
tile_map[24][32]);
00035
00036 /* Squelette de la fonction traitement_touches */
00037 void traitement_touches(int *position_x, int *position_y, int *bloc_x, int *bloc_y, int
tilemap[24][32], int direction);
00038
```

```

00039 /* Squelette de la fonction mise_a_jour_bordures_niveau_3 */
00040 int mise_a_jour_bordures_niveau_3(SDL_Renderer* renderer, SDL_Texture* texture_image_mur_termine, int
00041 tilemap[24][32], int x_tile, int y_tile,
00042 SDL_Rect *rectangle_tile, int largeur_tile, int hauteur_tile);
00043 /* Squelette de la fonction mise_a_jour_mini_jeu_2_niveau_3 */
00044 void mise_a_jour_mini_jeu_2_niveau_3(SDL_Renderer **renderer, modes_t *modeActif, SDL_Texture
00045 **texture_image_sol_labyrinthe,
00046 SDL_Rect *rectangle_plein_ecran, SDL_Texture
00047 **texture_image_plein_ecran,
00048 SDL_Texture **texture_image_mur_mini_jeu, SDL_Texture
00049 **texture_image_bordure_labyrinthe, SDL_Texture **texture_image_mur_termine,
00050 SDL_Texture **texture_image_fin_labyrinthe, SDL_Texture
00051 **texture_image_croix, SDL_Rect *rectangle_croix,
00052 SDL_Texture **texture, SDL_Rect *rectangle_tile, int bloc_x, int
00053 bloc_y,
00054 SDL_Texture **texture_image_personnage, SDL_Rect
00055 *rectangle_personnage,
00056 int position_x, int position_y, int
00057 tile_map_mini_jeu_niveau_3[24][32], niveaux *avancee_niveaux,
00058 int largeur_tile, int hauteur_tile);
00059 /* Squelette de la fonction mini_jeux_niveau_3 */
00060 void mini_jeux_niveau_3(SDL_Event *event, SDL_Renderer **renderer, SDL_Window **window, SDL_bool
00061 *programme_lance,
00062 SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
00063 SDL_bool *plein_ecran,
00064 niveaux *avancee_niveaux, int tile_map[18][32], SDL_Texture
00065 **texture_image_croix, SDL_Rect *rectangle_croix,
00066 int *mini_jeu, int *mini_jeu_1_termine, int *mini_jeu_2_termine,
00067 int *position_x, int *position_y, SDL_Texture **texture,
00068 int *largeur, int *hauteur, SDL_Rect *rectangle_demande,
00069 SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00070 SDL_Color couleurNoire,
00071 itemMenu *itemsDemandeQuitter, int tailleDemande, int
00072 collectibles_intermediaires[3],
00073 page_t *page_active, SDL_Rect *rectangle_tile, int *largeur_tile, int
00074 *hauteur_tile,
00075 int *avancer, int *reculer, int *sauter, int *saut, int *tombe,
00076 SDL_Rect rectangle_piece[45], int piece_bloquee[45], SDL_Rect
00077 rectangle_emplacement_piece[45], int *piece_selectionnee,
00078 int *decalage_x, int *decalage_y, SDL_Texture **texture_image_puzzle,
00079 int tile_map_mini_jeu_niveau_3[24][32], int *descendre, int *interagir, int
00080 *bloc_x, int *bloc_y,
00081 SDL_Texture **texture_image_sol_labyrinthe, SDL_Texture
00082 **texture_image_bordure_labyrinthe,
00083 SDL_Texture **texture_image_fin_labyrinthe, Mix_Music **musique,
00084 SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage,
00085 SDL_Texture **texture_image_mur_termine, SDL_Texture
00086 **texture_image_mur_mini_jeu,
00087 SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
00088 SDL_Keycode *touche_interagir,
00089 SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_descendre, modes_t
00090 *modeActif,
00091 itemMenu *itemsDemandeSauvegarde, barreDeSon *barre_de_son, itemMenu *pseudo,
00092 personnage_t *personnageActif, position_t *positionActive, int tailleNiveaux,
00093 time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
00094 avancee_succes_intermediaires[10]);

```

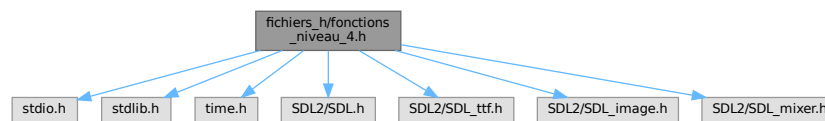
## 4.41 Référence du fichier fichiers\_h/fonctions\_niveau\_4.h

```

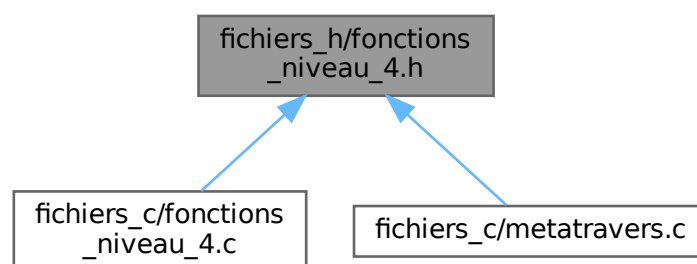
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>

```

Graphe des dépendances par inclusion de fonctions\_niveau\_4.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [initialisation\\_objets\\_niveau\\_4](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_bordure, SDL\_Texture \*\*texture\_image\_porte, SDL\_Texture \*\*texture\_image\_pique, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau)  
*Fonction qui permet d'initialiser les différents objets du niveau 4.*
- void [etage\\_1](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)  
*Fonction qui permet de créer l'étage 1.*
- void [etage\\_2](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)  
*Fonction qui permet de créer l'étage 2.*
- void [etage\\_3](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)  
*Fonction qui permet de créer l'étage 3.*
- void [etage\\_4](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)  
*Fonction qui permet de créer l'étage 4.*
- void [etage\\_5](#) (int \*position\_x, int \*position\_y, int \*position\_x\_initiale, int \*position\_y\_initiale, int tile\_map[18][32], SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_mur)  
*Fonction qui permet de créer l'étage 5.*
- void [mise\\_a\\_jour\\_rendu\\_niveau\\_4](#) (SDL\_Renderer \*\*renderer, SDL\_Texture \*\*texture\_image\_mur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_bordure, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_tile, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_porte, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, int numero\_etage, int position\_x, int position\_y, int tile\_map[18][32], SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, int largeur, int hauteur, int largeur\_tile, int hauteur\_tile)

— void [niveau\\_4](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Texture \*\*texture, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Texture \*\*texture\_image\_personnage, SDL\_Rect \*rectangle\_personnage, SDL\_Texture \*\*texture\_image\_mur, SDL\_Texture \*\*texture\_image\_fond, SDL\_Texture \*\*texture\_image\_bordure, SDL\_Texture \*\*texture\_image\_porte, SDL\_Texture \*\*texture\_image\_pique, [niveaux](#) \*avancee\_niveaux, SDL\_Surface \*\*surface, [modes\\_t](#) \*modeActif, int collectibles\_intermediaires[3], SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_interagir, int tile\_map[18][32], SDL\_Rect \*rectangle\_tile, SDL\_Texture \*\*texture\_image\_perso\_gagnant, [itemMenu](#) \*itemsDemandeQuitter, int tailleDemande, SDL\_Texture \*\*texture\_image\_croix, SDL\_Rect \*rectangle\_croix, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Rect \*rectangle\_demande, SDL\_Color couleurNoire, SDL\_Texture \*\*texture\_image\_fin\_dernier\_niveau, int \*avancer, int \*reculer, int \*sauter, int \*position\_avant\_saut, int \*saut, int \*tombe, int \*numero\_etage, int \*position\_x\_initiale, int \*position\_y\_initiale, int \*position\_x, int \*position\_y, int \*largeur, int \*hauteur, int \*largeur\_tile, int \*hauteur\_tile, [page\\_t](#) \*page\_active, [itemMenu](#) \*itemsDemandeSauvegarde, SDL\_Keycode \*touche\_descendre, [barreDeSon](#) \*barre\_de\_son, [itemMenu](#) \*pseudo, [personnage\\_t](#) \*personnageActif, [position\\_t](#) \*positionActive, int tailleNiveaux, time\_t temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes, int avancee\_succes\_intermediaires[10])

## 4.41.1 Documentation des fonctions

### 4.41.1.1 etage\_1()

```
void etage_1 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 1.

#### Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

#### Voir également

[chargement\\_image](#)

Définition à la ligne 47 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.41.1.2 etage\_2()

```
void etage_2 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 2.

## Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

## Voir également

[chargement\\_image](#)

Définition à la ligne 105 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.41.1.3 etage\_3()

```
void etage_3 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 3.

## Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

Voir également

[chargement\\_image](#)

Définition à la ligne 163 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.41.1.4 etage\_4()

```
void etage_4 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 4.

##### Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

Voir également

[chargement\\_image](#)

Définition à la ligne 221 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.41.1.5 etage\_5()

```
void etage_5 (
    int * position_x,
    int * position_y,
    int * position_x_initiale,
    int * position_y_initiale,
    int tile_map[18][32],
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_mur )
```

Fonction qui permet de créer l'étage 5.



## Paramètres

<i>position_x</i>	Pointeur vers la position en abscisse du joueur.
<i>position_y</i>	Pointeur vers la position en ordonnée du joueur.
<i>position_x_initiale</i>	Pointeur vers la position initiale en abscisse du joueur.
<i>position_y_initiale</i>	Pointeur vers la position initiale en ordonnée du joueur.
<i>tile_map</i>	Tableau représentant la carte du niveau.
<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_mur</i>	Texture de l'image des murs.

## Voir également

[chargement\\_image](#)

Définition à la ligne 279 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.41.1.6 initialisation\_objets\_niveau\_4()

```
void initialisation_objets_niveau_4 (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_bordure,
    SDL_Texture ** texture_image_porte,
    SDL_Texture ** texture_image_pique,
    SDL_Texture ** texture_image_fin_dernier_niveau )
```

Fonction qui permet d'initialiser les différents objets du niveau 4.

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_fond</i>	Texture de l'image du fond.
<i>texture_image_bordure</i>	Texture de l'image de la bordure.
<i>texture_image_porte</i>	Texture de l'image de la porte.
<i>texture_image_pique</i>	Texture de l'image du pique.
<i>texture_image_fin_dernier_niveau</i>	Texture de l'image de fin du dernier niveau.

## Voir également

[chargement\\_image](#)

Définition à la ligne 20 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.41.1.7 mise\_a\_jour\_rendu\_niveau\_4()

```
void mise_a_jour_rendu_niveau_4 (
    SDL_Renderer ** renderer,
```

```

SDL_Texture ** texture_image_mur,
SDL_Texture ** texture_image_fond,
SDL_Texture ** texture_image_bordure,
SDL_Texture ** texture,
SDL_Rect * rectangle_tile,
SDL_Rect * rectangle_plein_ecran,
SDL_Texture ** texture_image_plein_ecran,
SDL_Texture ** texture_image_porte,
SDL_Texture ** texture_image_fin_dernier_niveau,
SDL_Texture ** texture_image_personnage,
SDL_Rect * rectangle_personnage,
SDL_Texture ** texture_image_pique,
niveaux * avancee_niveaux,
int numero_etage,
int position_x,
int position_y,
int tile_map[18][32],
SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
int largeur,
int hauteur,
int largeur_tile,
int hauteur_tile )

```

Définition à la ligne 351 du fichier [fonctions\\_niveau\\_4.c](#).

#### 4.41.1.8 niveau\_4()

```

void niveau_4 (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
    SDL_Texture ** texture,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Texture ** texture_image_personnage,
    SDL_Rect * rectangle_personnage,
    SDL_Texture ** texture_image_mur,
    SDL_Texture ** texture_image_fond,
    SDL_Texture ** texture_image_bordure,
    SDL_Texture ** texture_image_porte,
    SDL_Texture ** texture_image_pique,
    niveaux * avancee_niveaux,
    SDL_Surface ** surface,
    modes_t * modeActif,
    int collectibles_intermediaires[3],
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_sauter_monter,
    SDL_Keycode * touche_interagir,
    int tile_map[18][32],
    SDL_Rect * rectangle_tile,
    SDL_Texture ** texture_image_perso_gagnant,
    itemMenu * itemsDemandeQuitter,
    int tailleDemande,

```

```

SDL_Texture ** texture_image_croix,
SDL_Rect * rectangle_croix,
SDL_Texture ** texture_texte,
TTF_Font ** police,
SDL_Rect * rectangle_demande,
SDL_Color couleurNoire,
SDL_Texture ** texture_image_fin_dernier_niveau,
int * avancer,
int * reculer,
int * sauter,
int * position_avant_saut,
int * saut,
int * tombe,
int * numero_etage,
int * position_x_initiale,
int * position_y_initiale,
int * position_x,
int * position_y,
int * largeur,
int * hauteur,
int * largeur_tile,
int * hauteur_tile,
page_t * page_active,
itemMenu * itemsDemandeSauvegarde,
SDL_Keycode * touche_descendre,
barreDeSon * barre_de_son,
itemMenu * pseudo,
personnage_t * personnageActif,
position_t * positionActive,
int tailleNiveaux,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes,
int avancee_succes_intermediaires[10] )

```

Définition à la ligne 510 du fichier [fonctions\\_niveau\\_4.c](#).

## 4.42 fonctions\_niveau\_4.h

[Aller à la documentation de ce fichier.](#)

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <time.h>
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_ttf.h>
00006 #include <SDL2/SDL_image.h>
00007 #include <SDL2/SDL_mixer.h>
00008
00009 /* Squelette de la fonction initialisation_objets_niveau_4 */
00010 void initialisation_objets_niveau_4(SDL_Renderer **renderer, SDL_Surface **surface,
00011                                     SDL_Texture **texture_image_fond, SDL_Texture
00012                                     **texture_image_bordure,
00013                                     SDL_Texture **texture_image_porte, SDL_Texture
00014                                     **texture_image_pique,
00015                                     SDL_Texture **texture_image_fin_dernier_niveau);
00016
00017 /* Squelette de la fonction etage_1 */
00018 void etage_1(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
00019             tile_map[18][32],
00020             SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur);
00021
00022 /* Squelette de la fonction etage_2 */
00023 void etage_2(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
00024             tile_map[18][32],

```

```

00021         SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur);
00022
00023 /* Squelette de la fonction etage_3 */
00024 void etage_3(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
    tile_map[18][32],
00025         SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur);
00026
00027 /* Squelette de la fonction etage_4 */
00028 void etage_4(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
    tile_map[18][32],
00029         SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur);
00030
00031 /* Squelette de la fonction etage_5 */
00032 void etage_5(int *position_x, int *position_y, int *position_x_initiale, int *position_y_initiale, int
    tile_map[18][32],
00033         SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture_image_mur);
00034
00035 /* Squelette de la fonction mise_a_jour_rendu_niveau_4 */
00036 void mise_a_jour_rendu_niveau_4(SDL_Renderer **renderer, SDL_Texture **texture_image_mur, SDL_Texture
    **texture_image_fond, SDL_Texture **texture_image_bordure,
00037         SDL_Texture **texture, SDL_Rect *rectangle_tile, SDL_Rect
    *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
00038         SDL_Texture **texture_image_porte, SDL_Texture
    **texture_image_fin_dernier_niveau,
00039         SDL_Texture **texture_image_personnage, SDL_Rect
    *rectangle_personnage,
00040         SDL_Texture **texture_image_pique, niveaux *avancee_niveaux, int
    numero_etage,
00041         int position_x, int position_y, int tile_map[18][32], SDL_Texture
    **texture_image_croix, SDL_Rect *rectangle_croix,
00042         int largeur, int hauteur, int largeur_tile, int hauteur_tile);
00043
00044 /* Squelette de la fonction niveau_4 */
00045 void niveau_4(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
    *programme_lance,
00046         SDL_Texture **texture, SDL_Rect *rectangle_plein_ecran, SDL_Texture
    **texture_image_plein_ecran, SDL_bool *plein_ecran,
00047         SDL_Texture **texture_image_personnage, SDL_Rect *rectangle_personnage,
    SDL_Texture **texture_image_mur, SDL_Texture **texture_image_fond,
00048         SDL_Texture **texture_image_bordure, SDL_Texture **texture_image_porte,
    SDL_Texture **texture_image_pique, niveaux *avancee_niveaux,
00049         SDL_Surface **surface, modes_t *modeActif, int collectibles_intermediaires[3],
    SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
00050         SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_interagir,
    int tile_map[18][32], SDL_Rect *rectangle_tile, SDL_Texture
    **texture_image_perso_gagnant,
00051         itemMenu *itemsDemandeQuitter, int tailleDemande, SDL_Texture **texture_image_croix,
    SDL_Rect *rectangle_croix,
00052         SDL_Texture **texture_texte, TTF_Font **police, SDL_Rect *rectangle_demande,
    SDL_Color couleurNoire, SDL_Texture **texture_image_fin_dernier_niveau,
00053         int *avancer, int *reculer, int *sauter, int *position_avant_saut, int *saut, int
    *tombe, int *numero_etage,
00054         int *position_x_initiale, int *position_y_initiale, int *position_x, int *position_y,
    int *largeur, int *hauteur, int *largeur_tile, int *hauteur_tile, page_t *page_active,
00055         itemMenu *itemsDemandeSauvegarde, SDL_Keycode *touche_descendre, barreDeSon
    *barre_de_son, itemMenu *pseudo,
00056         personnage_t *personnageActif, position_t *positionActive, int tailleNiveaux,
    time_t temps_debut_partie, int *compteur_mort, int *avancee_succes, int
    avancee_succes_intermediaires[10]);

```

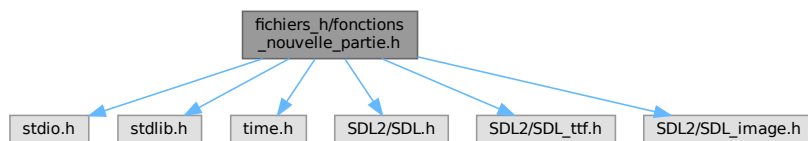
## 4.43 Référence du fichier fichiers\_h/fonctions\_nouvelle\_partie.h

```

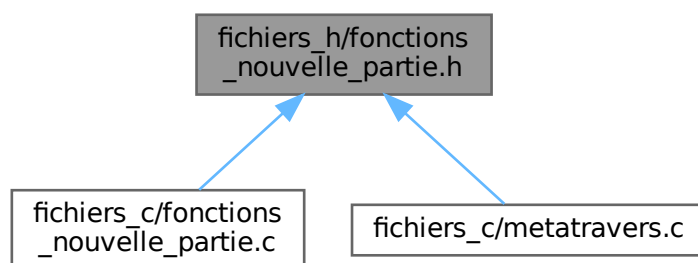
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>

```

Graph des dépendances par inclusion de fonctions\_nouvelle\_partie.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [initialisation\\_objets\\_nouvelle\\_partie](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, [itemMenu](#) \*titres, [itemMenu](#) \*items↔ Menu, [itemMenu](#) \*valider)
- Fonction qui permet d'initialiser les différents objets de la nouvelle partie.*
- void [mise\\_a\\_jour\\_rendu\\_nouvelle\\_partie](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image↔ options, [modes\\_t](#) modeActif, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Rect \*rectangle\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, SDL\_Rect \*rectangle\_perso\_2, [personnage\\_t](#) personnageActif, [itemMenu](#) \*pseudo, SDL\_Rect \*rectangle\_pseudo, [itemMenu](#) \*titres, int tailleTitres, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, int modeSaisie, [itemMenu](#) \*itemsMenu, [itemMenu](#) \*valider, int largeur, int hauteur)
- void [nouvelle\\_partie](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL↔ bool \*programme\_lance, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour↔ en\_arriere, SDL\_Rect \*rectangle\_options, SDL\_Texture \*\*texture\_image\_options, int \*modeSaisie, [modes\\_t](#) \*modeActif, SDL\_Texture \*\*texture\_image\_perso\_1, SDL\_Rect \*rectangle\_perso\_1, SDL\_Texture \*\*texture\_image\_perso\_2, SDL\_Rect \*rectangle\_perso\_2, [personnage\\_t](#) \*personnageActif, [itemMenu](#) \*pseudo, SDL\_Rect \*rectangle\_pseudo, [barreDeSon](#) \*barre\_de\_son, int \*pseudo\_valide, SDL\_Keycode \*touche\_aller\_a\_droite, SDL\_Keycode \*touche\_aller\_a\_gauche, SDL\_Keycode \*touche\_sauter\_monter, SDL\_Keycode \*touche\_descendre, SDL\_Keycode \*touche\_interagir, [itemMenu](#) \*titres, int tailleTitres, SDL↔ \_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, [position\\_t](#) \*positionActive, [niveaux](#) \*avancee\_niveaux, int tailleNiveaux, [itemMenu](#) \*itemsMenu, [itemMenu](#) \*valider, int \*largeur, int \*hauteur, [page\\_t](#) \*page\_active, time\_t \*temps\_debut\_partie, int \*compteur\_mort, int \*avancee\_succes)

## 4.43.1 Documentation des fonctions

### 4.43.1.1 initialisation\_objets\_nouvelle\_partie()

```
void initialisation_objets_nouvelle_partie (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_perso_1,
    SDL_Texture ** texture_image_perso_2,
    itemMenu * titres,
    itemMenu * itemsMenu,
    itemMenu * valider )
```

Fonction qui permet d'initialiser les différents objets de la nouvelle partie.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_perso_1</i>	Texture pour le premier personnage.
<i>texture_image_perso_2</i>	Texture pour le deuxième personnage.
<i>titres</i>	Tableau des titres des sections.
<i>itemsMenu</i>	Tableau des éléments de menu.
<i>valider</i>	Élément de menu pour valider.

#### Voir également

[chargement\\_image](#)

Définition à la ligne 20 du fichier [fonctions\\_nouvelle\\_partie.c](#).

### 4.43.1.2 mise\_a\_jour\_rendu\_nouvelle\_partie()

```
void mise_a_jour_rendu_nouvelle_partie (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    modes_t modeActif,
    SDL_Texture ** texture_image_perso_1,
    SDL_Rect * rectangle_perso_1,
    SDL_Texture ** texture_image_perso_2,
    SDL_Rect * rectangle_perso_2,
    personnage_t personnageActif,
    itemMenu * pseudo,
    SDL_Rect * rectangle_pseudo,
    itemMenu * titres,
    int tailleTitres,
```

```

    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurNoire,
    int modeSaisie,
    itemMenu * itemsMenu,
    itemMenu * valider,
    int largeur,
    int hauteur )

```

Définition à la ligne 74 du fichier [fonctions\\_nouvelle\\_partie.c](#).

#### 4.43.1.3 nouvelle\_partie()

```

void nouvelle_partie (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** render,
    SDL_bool * programme_lance,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_bool * plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Rect * rectangle_options,
    SDL_Texture ** texture_image_options,
    int * modeSaisie,
    modes_t * modeActif,
    SDL_Texture ** texture_image_perso_1,
    SDL_Rect * rectangle_perso_1,
    SDL_Texture ** texture_image_perso_2,
    SDL_Rect * rectangle_perso_2,
    personnage_t * personnageActif,
    itemMenu * pseudo,
    SDL_Rect * rectangle_pseudo,
    barreDeSon * barre_de_son,
    int * pseudo_valide,
    SDL_Keycode * touche_aller_a_droite,
    SDL_Keycode * touche_aller_a_gauche,
    SDL_Keycode * touche_sauter_monter,
    SDL_Keycode * touche_descendre,
    SDL_Keycode * touche_interagir,
    itemMenu * titres,
    int tailleTitres,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurNoire,
    position_t * positionActive,
    niveaux * avancee_niveaux,
    int tailleNiveaux,
    itemMenu * itemsMenu,
    itemMenu * valider,
    int * largeur,
    int * hauteur,
    page_t * page_active,
    time_t * temps_debut_partie,

```

```

    int * compteur_mort,
    int * avancee_succes )

```

Définition à la ligne 337 du fichier [fonctions\\_nouvelle\\_partie.c](#).

## 4.44 fonctions\_nouvelle\_partie.h

[Aller à la documentation de ce fichier.](#)

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <time.h>
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_ttf.h>
00006 #include <SDL2/SDL_image.h>
00007
00008 /* Squelette de la fonction initialisation_objets_nouvelle_partie */
00009 void initialisation_objets_nouvelle_partie(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_perso_1,
00010                                           SDL_Texture **texture_image_perso_2,
00011                                           itemMenu *titres, itemMenu *itemsMenu, itemMenu *valider);
00012
00013 /* Squelette de la fonction mise_a_jour_rendu_nouvelle_partie */
00014 void mise_a_jour_rendu_nouvelle_partie(SDL_Renderer **renderer, SDL_Rect *rectangle_plein_ecran,
SDL_Texture **texture_image_plein_ecran,
00015                                         SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
**texture_image_retour_en_arriere,
00016                                         SDL_Rect *rectangle_options, SDL_Texture
**texture_image_options,
00017                                         modes_t modeActif, SDL_Texture **texture_image_perso_1,
SDL_Rect *rectangle_perso_1,
00018                                         SDL_Texture **texture_image_perso_2, SDL_Rect
*rectangle_perso_2, personnage_t personnageActif,
00019                                         itemMenu *pseudo, SDL_Rect *rectangle_pseudo,
00020                                         itemMenu *titres, int tailleTitres, SDL_Surface **surface,
SDL_Texture **texture_texte,
00021                                         TTF_Font **police, SDL_Color couleurNoire, int modeSaisie,
00022                                         itemMenu *itemsMenu, itemMenu *valider, int largeur, int
hauteur);
00023
00024 /* Squelette de la fonction nouvelle_partie */
00025 void nouvelle_partie(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance,
00026                     SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran,
SDL_bool *plein_ecran,
00027                     SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
**texture_image_retour_en_arriere,
00028                     SDL_Rect *rectangle_options, SDL_Texture **texture_image_options, int
*modeSaisie,
00029                     modes_t *modeActif, SDL_Texture **texture_image_perso_1, SDL_Rect
*rectangle_perso_1,
00030                     SDL_Texture **texture_image_perso_2, SDL_Rect *rectangle_perso_2, personnage_t
*personnageActif,
00031                     itemMenu *pseudo, SDL_Rect *rectangle_pseudo, barreDeSon *barre_de_son, int
*pseudo_valide,
00032                     SDL_Keycode *touche_aller_a_droite, SDL_Keycode *touche_aller_a_gauche,
SDL_Keycode *touche_sauter_monter,
00033                     SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, itemMenu *titres,
int tailleTitres, SDL_Surface **surface, SDL_Texture **texture_texte,
00034                     TTF_Font **police, SDL_Color couleurNoire, position_t *positionActive, niveaux
*avancee_niveaux, int tailleNiveaux,
00035                     itemMenu *itemsMenu, itemMenu *valider, int *largeur, int *hauteur, page_t
*page_active,
00036                     time_t *temps_debut_partie, int *compteur_mort, int *avancee_succes);

```

## 4.45 Référence du fichier fichiers\_h/fonctions\_options.h

```

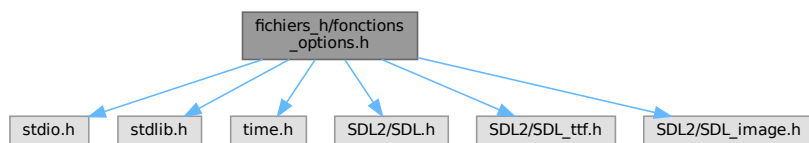
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>

```

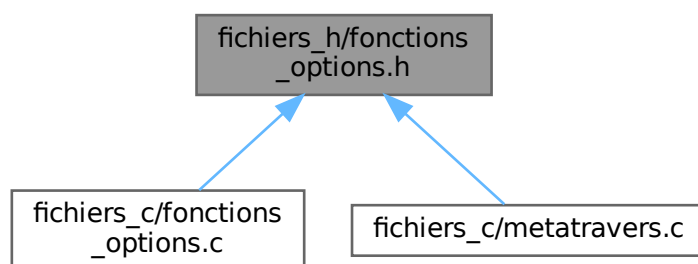


```
#include <SDL2/SDL_image.h>
```

Graphe des dépendances par inclusion de fonctions\_options.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- void [initialisation\\_objets\\_options](#) (SDL\_Renderer \*\*renderer, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_image\_hautParleurActif, SDL\_Texture \*\*texture\_image\_hautParleurDesactive, [itemMenu](#) \*titre, [itemMenu](#) \*itemsMenu, [itemMenu](#) \*itemsTouches, [itemMenu](#) \*itemsBarres)

*Fonction qui permet d'initialiser les différents objets des options.*

- void [mise\\_a\\_jour\\_rendu\\_options](#) (SDL\_Renderer \*\*renderer, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_hautParleurActif, SDL\_Texture \*\*texture\_image\_hautParleurDesactive, SDL\_bool \*sonsActifs, SDL\_Rect \*rectangles\_boutons\_sons, [option\\_t](#) ongletActif, [itemMenu](#) \*titre, SDL\_Surface \*\*surface, SDL\_Texture \*\*texture\_texte, TTF\_Font \*\*police, SDL\_Color couleurNoire, int selection\_touche, [itemMenu](#) \*itemsMenu, int tailleMenu, [itemMenu](#) \*itemsTouches, int tailleTouches, [barreDeSon](#) \*barre\_de\_son, int tailleBarres, [itemMenu](#) \*itemsBarres, int largeur, int hauteur)

*Fonction qui met à jour le rendu des options.*

- void [mise\\_a\\_jour\\_barre\\_de\\_son](#) (SDL\_Event \*event, [barreDeSon](#) \*barre\_de\_son, SDL\_bool \*sonsActifs)

*Fonction qui permet de mettre à jour les barres de sons.*

- void [mise\\_a\\_jour\\_touches](#) (SDL\_Event \*event, SDL\_Keycode \*touche, int \*selection\_touche, [itemMenu](#) \*itemsTouches)

*Fonction qui permet de mettre à jour les touches.*

- void [options](#) (SDL\_Event \*event, SDL\_Window \*\*window, SDL\_Renderer \*\*renderer, SDL\_bool \*programme\_lance, SDL\_Rect \*rectangle\_plein\_ecran, SDL\_Texture \*\*texture\_image\_plein\_ecran, SDL\_bool \*plein\_ecran, SDL\_Rect \*rectangle\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_retour\_en\_arriere, SDL\_Texture \*\*texture\_image\_hautParleurActif, SDL\_Rect \*rectangle\_demande\_sauvegarde,

```

itemMenu *itemsDemandeSauvegarde, int tailleDemandeSauvegarde, SDL_Texture **texture_image_↵
hautParleurDesactive, SDL_bool *sonsActifs, SDL_Rect *rectangles_boutons_sons, option_t *onglet_↵
Actif, itemMenu *pseudo, modes_t *modeActif, personnage_t *personnageActif, position_t *position_↵
Active, niveaux *avancee_niveaux, int tailleNiveaux, itemMenu *titre, SDL_Surface **surface, SDL_↵
Texture **texture_texte, TTF_Font **police, int *selection_touche, SDL_Keycode *touche_aller_a_droite,
SDL_Keycode *touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter, SDL_Keycode *touche_↵
descendre, SDL_Keycode *touche_interagir, SDL_Color couleurNoire, itemMenu *itemsMenu, int tailleMenu,
itemMenu *itemsTouches, int tailleTouches, barreDeSon *barre_de_son, int tailleBarres, itemMenu *items_↵
Barres, int *largeur, int *hauteur, page_t *page_active, page_t *page_precedente, int *maintient Clic,
time_t temps_debut_partie, int *compteur_mort, int *avancee_succes)

```

## 4.45.1 Documentation des fonctions

### 4.45.1.1 initialisation\_objets\_options()

```

void initialisation_objets_options (
    SDL_Renderer ** renderer,
    SDL_Surface ** surface,
    SDL_Texture ** texture_image_hautParleurActif,
    SDL_Texture ** texture_image_hautParleurDesactive,
    itemMenu * titre,
    itemMenu * itemsMenu,
    itemMenu * itemsTouches,
    itemMenu * itemsBarres )

```

Fonction qui permet d'initialiser les différents objets des options.

#### Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>surface</i>	Pointeur vers la surface SDL.
<i>texture_image_hautParleurActif</i>	Texture pour l'icône de haut-parleur actif.
<i>texture_image_hautParleurDesactive</i>	Texture pour l'icône de haut-parleur désactivé.
<i>titre</i>	Pointeur vers l'item du titre du menu.
<i>itemsMenu</i>	Tableau des items du menu.
<i>itemsTouches</i>	Tableau des items pour les touches du clavier.
<i>itemsBarres</i>	Tableau des items pour les barres de volume.

Voir également

[chargement\\_image](#)

Définition à la ligne 19 du fichier [fonctions\\_options.c](#).

### 4.45.1.2 mise\_a\_jour\_barre\_de\_son()

```

void mise_a_jour_barre_de_son (
    SDL_Event * event,
    barreDeSon * barre_de_son,
    SDL_bool * sonsActifs )

```

Fonction qui permet de mettre à jour les barres de sons.

## Paramètres

<i>event</i>	Pointeur vers l'événement SDL.
<i>barre_de_son</i>	Pointeur vers la barre de son.
<i>sonsActifs</i>	Pointeur booléen pour l'état des sons.

Définition à la ligne 297 du fichier [fonctions\\_options.c](#).

## 4.45.1.3 mise\_a\_jour\_rendu\_options()

```
void mise_a_jour_rendu_options (
    SDL_Renderer ** renderer,
    SDL_Rect * rectangle_plein_ecran,
    SDL_Texture ** texture_image_plein_ecran,
    SDL_Rect * rectangle_retour_en_arriere,
    SDL_Texture ** texture_image_retour_en_arriere,
    SDL_Texture ** texture_image_hautParleurActif,
    SDL_Texture ** texture_image_hautParleurDesactive,
    SDL_bool * sonsActifs,
    SDL_Rect * rectangles_boutons_sons,
    option_t ongletActif,
    itemMenu * titre,
    SDL_Surface ** surface,
    SDL_Texture ** texture_texte,
    TTF_Font ** police,
    SDL_Color couleurNoire,
    int selection_touche,
    itemMenu * itemsMenu,
    int tailleMenu,
    itemMenu * itemsTouches,
    int tailleTouches,
    barreDeSon * barre_de_son,
    int tailleBarres,
    itemMenu * itemsBarres,
    int largeur,
    int hauteur )
```

Fonction qui met à jour le rendu des options.

## Paramètres

<i>renderer</i>	Pointeur vers le renderer SDL.
<i>rectangle_plein_ecran</i>	Rectangle pour le plein écran.
<i>texture_image_plein_ecran</i>	Texture pour l'image du plein écran.
<i>rectangle_retour_en_arriere</i>	Rectangle pour le bouton de retour en arrière.
<i>texture_image_retour_en_arriere</i>	Texture pour l'image du bouton de retour en arrière.
<i>texture_image_hautParleurActif</i>	Texture pour l'icône de haut-parleur actif.
<i>texture_image_hautParleurDesactive</i>	Texture pour l'icône de haut-parleur désactivé.
<i>sonsActifs</i>	Pointeur booléen pour l'état des sons.
<i>rectangles_boutons_sons</i>	Tableau des rectangles pour les boutons de sons.
<i>ongletActif</i>	Onglet actif dans le menu des options.
<i>titre</i>	Pointeur vers l'item du titre du menu.
<i>surface</i>	Pointeur vers la surface SDL.

## Paramètres

<i>texture_texte</i>	Texture pour le texte.
<i>police</i>	Pointeur vers la police TTF.
<i>couleurNoire</i>	Couleur noire.
<i>selection_touche</i>	Sélection de la touche du clavier.
<i>itemsMenu</i>	Tableau des items du menu.
<i>tailleMenu</i>	Taille du tableau des items du menu.
<i>itemsTouches</i>	Tableau des items pour les touches du clavier.
<i>tailleTouches</i>	Taille du tableau des items pour les touches du clavier.
<i>barre_de_son</i>	Tableau des barres de son.
<i>tailleBarres</i>	Taille du tableau des barres de son.
<i>itemsBarres</i>	Tableau des items pour les barres de volume.
<i>largeur</i>	Largeur de la fenêtre.
<i>hauteur</i>	Hauteur de la fenêtre.

## Voir également

[erreur](#)  
[affichage\\_texte](#)

Définition à la ligne 83 du fichier [fonctions\\_options.c](#).

## 4.45.1.4 mise\_a\_jour\_touches()

```
void mise_a_jour_touches (
    SDL_Event * event,
    SDL_Keycode * touche,
    int * selection_touche,
    itemMenu * itemsTouches )
```

Fonction qui permet de mettre à jour les touches.

## Paramètres

<i>event</i>	Pointeur vers l'événement SDL.
<i>touche</i>	Pointeur vers la touche sélectionnée.
<i>selection_touche</i>	Pointeur vers l'indice de la touche sélectionnée.
<i>itemsTouches</i>	Tableau des éléments de menu des touches.

Définition à la ligne 315 du fichier [fonctions\\_options.c](#).

## 4.45.1.5 options()

```
void options (
    SDL_Event * event,
    SDL_Window ** window,
    SDL_Renderer ** renderer,
    SDL_bool * programme_lance,
```

```

SDL_Rect * rectangle_plein_ecran,
SDL_Texture ** texture_image_plein_ecran,
SDL_bool * plein_ecran,
SDL_Rect * rectangle_retour_en_arriere,
SDL_Texture ** texture_image_retour_en_arriere,
SDL_Texture ** texture_image_hautParleurActif,
SDL_Rect * rectangle_demande_sauvegarde,
itemMenu * itemsDemandeSauvegarde,
int tailleDemandeSauvegarde,
SDL_Texture ** texture_image_hautParleurDesactive,
SDL_bool * sonsActifs,
SDL_Rect * rectangles_boutons_sons,
option_t * ongletActif,
itemMenu * pseudo,
modes_t * modeActif,
personnage_t * personnageActif,
position_t * positionActive,
niveaux * avancee_niveaux,
int tailleNiveaux,
itemMenu * titre,
SDL_Surface ** surface,
SDL_Texture ** texture_texte,
TTF_Font ** police,
int * selection_touche,
SDL_Keycode * touche_aller_a_droite,
SDL_Keycode * touche_aller_a_gauche,
SDL_Keycode * touche_sauter_monter,
SDL_Keycode * touche_descendre,
SDL_Keycode * touche_interagir,
SDL_Color couleurNoire,
itemMenu * itemsMenu,
int tailleMenu,
itemMenu * itemsTouches,
int tailleTouches,
barreDeSon * barre_de_son,
int tailleBarres,
itemMenu * itemsBarres,
int * largeur,
int * hauteur,
page_t * page_active,
page_t * page_precedente,
int * maintient_clic,
time_t temps_debut_partie,
int * compteur_mort,
int * avancee_succes )

```

Définition à la ligne 381 du fichier [fonctions\\_options.c](#).

## 4.46 fonctions\_options.h

[Aller à la documentation de ce fichier.](#)

```

00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <time.h>
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_ttf.h>
00006 #include <SDL2/SDL_image.h>
00007

```

```

00008 /* Squelette de la fonction initialisation_objets_options */
00009 void initialisation_objets_options(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture
**texture_image_hautParleurActif,
00010                                     SDL_Texture **texture_image_hautParleurDesactive,
00011                                     itemMenu *titre, itemMenu *itemsMenu, itemMenu *itemsTouches,
itemMenu *itemsBarres);
00012
00013 /* Squelette de la fonction mise_a_jour_rendu_options */
00014 void mise_a_jour_rendu_options(SDL_Renderer **renderer, SDL_Rect *rectangle_plein_ecran, SDL_Texture
**texture_image_plein_ecran,
00015                               SDL_Rect *rectangle_retour_en_arriere, SDL_Texture
**texture_image_retour_en_arriere,
00016                               SDL_Texture **texture_image_hautParleurActif,
00017                               SDL_Texture **texture_image_hautParleurDesactive, SDL_bool *sonsActifs,
00018                               SDL_Rect *rectangles_boutons_sons, option_t ongletActif,
00019                               itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte,
TTF_Font **police,
00020                               SDL_Color couleurNoire, int selection_touche,
00021                               itemMenu *itemsMenu, int tailleMenu, itemMenu *itemsTouches, int
tailleTouches,
00022                               barreDeSon *barre_de_son, int tailleBarres, itemMenu *itemsBarres,
00023                               int largeur, int hauteur);
00024
00025 /* Squelette de la fonction mise_a_jour_barre_de_son */
00026 void mise_a_jour_barre_de_son(SDL_Event *event, barreDeSon *barre_de_son, SDL_bool *sonsActifs);
00027
00028 /* Squelette de la fonction mise_a_jour_touches */
00029 void mise_a_jour_touches(SDL_Event *event, SDL_Keycode *touche, int *selection_touche, itemMenu
*itemsTouches);
00030
00031 /* Squelette de la fonction options */
00032 void options(SDL_Event *event, SDL_Window **window, SDL_Renderer **renderer, SDL_bool
*programme_lance,
00033              SDL_Rect *rectangle_plein_ecran, SDL_Texture **texture_image_plein_ecran, SDL_bool
*plein_ecran,
00034              SDL_Rect *rectangle_retour_en_arriere, SDL_Texture **texture_image_retour_en_arriere,
00035              SDL_Texture **texture_image_hautParleurActif, SDL_Rect *rectangle_demande_sauvegarde,
itemMenu *itemsDemandeSauvegarde, int tailleDemandeSauvegarde,
00036              SDL_Texture **texture_image_hautParleurDesactive, SDL_bool *sonsActifs,
00037              SDL_Rect *rectangles_boutons_sons, option_t ongletActif, itemMenu *pseudo,
00038              modes_t *modeActif, personnage_t *personnageActif, position_t *positionActive,
00039              niveaux *avancee_niveaux, int tailleNiveaux,
00040              itemMenu *titre, SDL_Surface **surface, SDL_Texture **texture_texte, TTF_Font **police,
00041              int *selection_touche, SDL_Keycode *touche_aller_a_droite, SDL_Keycode
*touche_aller_a_gauche, SDL_Keycode *touche_sauter_monter,
00042              SDL_Keycode *touche_descendre, SDL_Keycode *touche_interagir, SDL_Color couleurNoire,
00043              itemMenu *itemsMenu, int tailleMenu, itemMenu *itemsTouches, int tailleTouches,
00044              barreDeSon *barre_de_son, int tailleBarres, itemMenu *itemsBarres,
00045              int *largeur, int *hauteur, page_t *page_active, page_t *page_precedente, int
*maintient_clic,
00046              time_t temps_debut_partie, int *compteur_mort, int *avancee_succes);

```

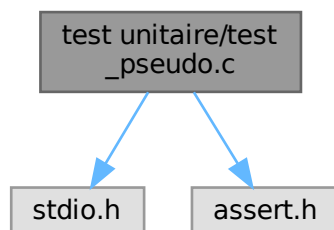
## 4.47 Référence du fichier test unitaire/test\_pseudo.c

```

#include <stdio.h>
#include <assert.h>

```

Graphe des dépendances par inclusion de test\_pseudo.c:



## Macros

— `#define BUFFER_SIZE 5000`

## Fonctions

— `int main ()`

### 4.47.1 Documentation des macros

#### 4.47.1.1 BUFFER\_SIZE

```
#define BUFFER_SIZE 5000
```

Définition à la ligne 3 du fichier `test_pseudo.c`.

### 4.47.2 Documentation des fonctions

#### 4.47.2.1 main()

```
int main ( )
```

Définition à la ligne 6 du fichier `test_pseudo.c`.

## 4.48 test\_pseudo.c

[Aller à la documentation de ce fichier.](#)

```
00001 #include <stdio.h>
00002 #include <assert.h>
00003 #define BUFFER_SIZE 5000
00004
00005
00006 int main() {
00007     char nom[BUFFER_SIZE];
00008
00009
00010
00011
00012     return 0;
00013 }
```





# Index

- affichage\_texte
  - fonctions\_generales.c, [51](#)
  - fonctions\_generales.h, [219](#)
- arrivee\_niveaux\_2\_3
  - fonctions\_arrivee\_niveaux\_2\_3.c, [10](#)
  - fonctions\_arrivee\_niveaux\_2\_3.h, [204](#)
- barre
  - barreDeSon, [6](#)
- barreDeSon, [5](#)
  - barre, [6](#)
  - curseur, [6](#)
  - volume, [6](#)
  - volume\_precedent, [6](#)
- BAS
  - fonctions\_generales.h, [217](#)
- BAS\_GAUCHE
  - fonctions\_generales.h, [217](#)
- BUFFER\_SIZE
  - test\_pseudo.c, [277](#)
- CARTE
  - fonctions\_generales.h, [218](#)
- carte
  - fonctions\_carte.c, [31](#)
  - fonctions\_carte.h, [210](#)
- chargement\_image
  - fonctions\_generales.c, [51](#)
  - fonctions\_generales.h, [219](#)
- chargement\_niveau\_1
  - fonctions\_niveau\_1.c, [83](#)
  - fonctions\_niveau\_1.h, [237](#)
- clic\_case
  - fonctions\_generales.c, [52](#)
  - fonctions\_generales.h, [219](#)
- clic\_plein\_ecran
  - fonctions\_generales.c, [52](#)
  - fonctions\_generales.h, [220](#)
- creer\_fenetre\_rendu
  - fonctions\_generales.c, [53](#)
  - fonctions\_generales.h, [220](#)
- curseur
  - barreDeSon, [6](#)
- demande\_quitter\_niveau
  - fonctions\_generales.c, [53](#)
  - fonctions\_generales.h, [220](#)
- demande\_sauvegarde
  - fonctions\_generales.c, [54](#)
  - fonctions\_generales.h, [221](#)
- deplacement\_personnage
  - fonctions\_generales.c, [55](#)
  - fonctions\_generales.h, [222](#)
- deplacement\_personnage\_carte
  - fonctions\_carte.c, [32](#)
  - fonctions\_carte.h, [211](#)
- detruire\_fenetre\_rendu
  - fonctions\_generales.c, [55](#)
  - fonctions\_generales.h, [222](#)
- detruire\_objets
  - fonctions\_generales.c, [55](#)
  - fonctions\_generales.h, [222](#)
- direction\_s
  - fonctions\_generales.h, [217](#)
- direction\_t
  - fonctions\_generales.h, [216](#)
- DROITE
  - fonctions\_generales.h, [217](#)
- erreur
  - fonctions\_generales.c, [57](#)
  - fonctions\_generales.h, [224](#)
- etage\_1
  - fonctions\_niveau\_4.c, [140](#)
  - fonctions\_niveau\_4.h, [260](#)
- etage\_2
  - fonctions\_niveau\_4.c, [141](#)
  - fonctions\_niveau\_4.h, [260](#)
- etage\_3
  - fonctions\_niveau\_4.c, [142](#)
  - fonctions\_niveau\_4.h, [261](#)
- etage\_4
  - fonctions\_niveau\_4.c, [142](#)
  - fonctions\_niveau\_4.h, [262](#)
- etage\_5
  - fonctions\_niveau\_4.c, [143](#)
  - fonctions\_niveau\_4.h, [262](#)
- explications
  - fonctions\_arrivee\_niveaux\_2\_3.c, [12](#)
  - fonctions\_arrivee\_niveaux\_2\_3.h, [206](#)
- fichiers\_c/fonctions\_arrivee\_niveaux\_2\_3.c, [9](#), [14](#)
- fichiers\_c/fonctions\_carte.c, [29](#), [34](#)
- fichiers\_c/fonctions\_generales.c, [50](#), [60](#)
- fichiers\_c/fonctions\_introduction.c, [68](#), [71](#)
- fichiers\_c/fonctions\_menu\_principal.c, [74](#), [78](#)
- fichiers\_c/fonctions\_niveau\_1.c, [82](#), [86](#)
- fichiers\_c/fonctions\_niveau\_2.c, [96](#), [103](#)
- fichiers\_c/fonctions\_niveau\_3.c, [117](#), [123](#)
- fichiers\_c/fonctions\_niveau\_4.c, [139](#), [146](#)

fichiers\_c/fonctions\_nouvelle\_partie.c, 159, 162  
 fichiers\_c/fonctions\_options.c, 168, 173  
 fichiers\_c/metatravers.c, 181, 183  
 fichiers\_h/fonctions\_arrivee\_niveaux\_2\_3.h, 202, 207  
 fichiers\_h/fonctions\_carte.h, 208, 213  
 fichiers\_h/fonctions\_generales.h, 214, 226  
 fichiers\_h/fonctions\_introduction.h, 228, 230  
 fichiers\_h/fonctions\_menu\_principal.h, 231, 234  
 fichiers\_h/fonctions\_niveau\_1.h, 235, 240  
 fichiers\_h/fonctions\_niveau\_2.h, 240, 248  
 fichiers\_h/fonctions\_niveau\_3.h, 249, 257  
 fichiers\_h/fonctions\_niveau\_4.h, 258, 265  
 fichiers\_h/fonctions\_nouvelle\_partie.h, 266, 270  
 fichiers\_h/fonctions\_options.h, 270, 275  
 fonctions\_arrivee\_niveaux\_2\_3.c  
     arrivee\_niveaux\_2\_3, 10  
     explications, 12  
     mise\_a\_jour\_rendu\_arrivee\_niveaux\_2\_3, 12  
     salon\_arrivee\_niveaux\_2\_3, 13  
 fonctions\_arrivee\_niveaux\_2\_3.h  
     arrivee\_niveaux\_2\_3, 204  
     explications, 206  
     mise\_a\_jour\_rendu\_arrivee\_niveaux\_2\_3, 206  
     salon\_arrivee\_niveaux\_2\_3, 207  
 fonctions\_carte.c  
     carte, 31  
     deplacement\_personnage\_carte, 32  
     initialisation\_objets\_carte, 32  
     mise\_a\_jour\_rendu\_carte, 33  
 fonctions\_carte.h  
     carte, 210  
     deplacement\_personnage\_carte, 211  
     initialisation\_objets\_carte, 212  
     mise\_a\_jour\_rendu\_carte, 212  
 fonctions\_generales.c  
     affichage\_texte, 51  
     chargement\_image, 51  
     clic\_case, 52  
     clic\_plein\_ecran, 52  
     creer\_fenetre\_rendu, 53  
     demande\_quitter\_niveau, 53  
     demande\_sauvegarde, 54  
     deplacement\_personnage, 55  
     detruire\_fenetre\_rendu, 55  
     detruire\_objets, 55  
     erreur, 57  
     initialisation\_objets, 57  
     redimensionnement\_fenetre, 57  
     sauvegarder\_partie, 59  
     verification\_sauvegarde, 59  
 fonctions\_generales.h  
     affichage\_texte, 219  
     BAS, 217  
     BAS\_GAUCHE, 217  
     CARTE, 218  
     chargement\_image, 219  
     clic\_case, 219  
     clic\_plein\_ecran, 220  
     creer\_fenetre\_rendu, 220  
     demande\_quitter\_niveau, 220  
     demande\_sauvegarde, 221  
     deplacement\_personnage, 222  
     detruire\_fenetre\_rendu, 222  
     detruire\_objets, 222  
     direction\_s, 217  
     direction\_t, 216  
     DROITE, 217  
     erreur, 224  
     GAUCHE, 217  
     HAUT, 217  
     HAUT\_DROITE, 217  
     initialisation\_objets, 224  
     INTRODUCTION, 218  
     MENU\_PRINCIPAL, 218  
     MODE\_HARD, 217  
     MODE\_NORMAL, 217  
     modes\_s, 217  
     modes\_t, 216  
     NIVEAU0, 218  
     NIVEAU1, 218  
     NIVEAU2, 218  
     NIVEAU3, 218  
     NIVEAU4, 218  
     NIVEAU\_1, 218  
     NIVEAU\_2, 218  
     NIVEAU\_3, 218  
     NIVEAU\_4, 218  
     NOUVELLE\_PARTIE, 218  
     ONGLET\_SON, 218  
     ONGLET\_TOUCHES, 218  
     option\_s, 217  
     option\_t, 216  
     OPTIONS, 218  
     page\_s, 218  
     page\_t, 216  
     PERSONNAGE\_1, 218  
     PERSONNAGE\_2, 218  
     personnage\_s, 218  
     personnage\_t, 217  
     position\_s, 218  
     position\_t, 217  
     redimensionnement\_fenetre, 225  
     sauvegarder\_partie, 225  
     verification\_sauvegarde, 225  
 fonctions\_introduction.c  
     introduction, 69  
     mise\_a\_jour\_rendu\_introduction, 70  
 fonctions\_introduction.h  
     introduction, 229  
     mise\_a\_jour\_rendu\_introduction, 230  
 fonctions\_menu\_principal.c  
     initialisation\_objets\_menu\_principal, 75  
     menu\_principal, 76  
     mise\_a\_jour\_rendu\_menu\_principal, 77  
 fonctions\_menu\_principal.h  
     initialisation\_objets\_menu\_principal, 232

- menu\_principal, 232
- mise\_a\_jour\_rendu\_menu\_principal, 233
- fonctions\_niveau\_1.c
  - chargement\_niveau\_1, 83
  - initialisation\_objets\_niveau\_1, 84
  - mise\_a\_jour\_rendu\_niveau\_1, 84
  - niveau\_1, 85
- fonctions\_niveau\_1.h
  - chargement\_niveau\_1, 237
  - initialisation\_objets\_niveau\_1, 237
  - mise\_a\_jour\_rendu\_niveau\_1, 238
  - niveau\_1, 238
- fonctions\_niveau\_2.c
  - initialisation\_objets\_niveau\_2, 97
  - mini\_jeu\_1\_niveau\_2, 98
  - mini\_jeu\_2\_niveau\_2, 98
  - mini\_jeux\_niveau\_2, 99
  - mise\_a\_jour\_bordures\_niveau\_2, 100
  - mise\_a\_jour\_mini\_jeu\_1\_niveau\_2, 101
  - mise\_a\_jour\_mini\_jeu\_2\_niveau\_2, 101
  - verification\_chemin, 102
- fonctions\_niveau\_2.h
  - initialisation\_objets\_niveau\_2, 242
  - mini\_jeu\_1\_niveau\_2, 243
  - mini\_jeu\_2\_niveau\_2, 243
  - mini\_jeux\_niveau\_2, 244
  - mise\_a\_jour\_bordures\_niveau\_2, 245
  - mise\_a\_jour\_mini\_jeu\_1\_niveau\_2, 246
  - mise\_a\_jour\_mini\_jeu\_2\_niveau\_2, 246
  - salon\_arrivee\_niveaux\_2\_3, 247
  - verification\_chemin, 247
- fonctions\_niveau\_3.c
  - initialisation\_objets\_niveau\_3, 118
  - mini\_jeu\_2\_niveau\_3, 119
  - mini\_jeux\_niveau\_3, 119
  - mise\_a\_jour\_bordures\_niveau\_3, 120
  - mise\_a\_jour\_mini\_jeu\_1\_niveau\_3, 121
  - mise\_a\_jour\_mini\_jeu\_2\_niveau\_3, 121
  - piece\_proche\_position\_correcte, 122
  - rectangle\_piece\_aleatoire, 122
  - traitement\_touches, 122
  - verification\_puzzle\_fini, 123
- fonctions\_niveau\_3.h
  - initialisation\_objets\_niveau\_3, 251
  - mini\_jeu\_2\_niveau\_3, 252
  - mini\_jeux\_niveau\_3, 252
  - mise\_a\_jour\_bordures\_niveau\_3, 254
  - mise\_a\_jour\_mini\_jeu\_1\_niveau\_3, 254
  - mise\_a\_jour\_mini\_jeu\_2\_niveau\_3, 254
  - piece\_proche\_position\_correcte, 255
  - rectangle\_piece\_aleatoire, 255
  - salon\_arrivee\_niveaux\_2\_3, 256
  - traitement\_touches, 256
  - verification\_puzzle\_fini, 257
- fonctions\_niveau\_4.c
  - etage\_1, 140
  - etage\_2, 141
  - etage\_3, 142
  - etage\_4, 142
  - etage\_5, 143
  - initialisation\_objets\_niveau\_4, 143
  - mise\_a\_jour\_rendu\_niveau\_4, 144
  - niveau\_4, 145
- fonctions\_niveau\_4.h
  - etage\_1, 260
  - etage\_2, 260
  - etage\_3, 261
  - etage\_4, 262
  - etage\_5, 262
  - initialisation\_objets\_niveau\_4, 263
  - mise\_a\_jour\_rendu\_niveau\_4, 263
  - niveau\_4, 264
- fonctions\_nouvelle\_partie.c
  - initialisation\_objets\_nouvelle\_partie, 160
  - mise\_a\_jour\_rendu\_nouvelle\_partie, 161
  - nouvelle\_partie, 161
- fonctions\_nouvelle\_partie.h
  - initialisation\_objets\_nouvelle\_partie, 268
  - mise\_a\_jour\_rendu\_nouvelle\_partie, 268
  - nouvelle\_partie, 269
- fonctions\_options.c
  - initialisation\_objets\_options, 169
  - mise\_a\_jour\_barre\_de\_son, 170
  - mise\_a\_jour\_rendu\_options, 170
  - mise\_a\_jour\_touches, 171
  - options, 172
- fonctions\_options.h
  - initialisation\_objets\_options, 272
  - mise\_a\_jour\_barre\_de\_son, 272
  - mise\_a\_jour\_rendu\_options, 273
  - mise\_a\_jour\_touches, 274
  - options, 274
- GAUCHE
  - fonctions\_generales.h, 217
- HAUT
  - fonctions\_generales.h, 217
- HAUT\_DROITE
  - fonctions\_generales.h, 217
- initialisation\_objets
  - fonctions\_generales.c, 57
  - fonctions\_generales.h, 224
- initialisation\_objets\_carte
  - fonctions\_carte.c, 32
  - fonctions\_carte.h, 212
- initialisation\_objets\_menu\_principal
  - fonctions\_menu\_principal.c, 75
  - fonctions\_menu\_principal.h, 232
- initialisation\_objets\_niveau\_1
  - fonctions\_niveau\_1.c, 84
  - fonctions\_niveau\_1.h, 237
- initialisation\_objets\_niveau\_2
  - fonctions\_niveau\_2.c, 97
  - fonctions\_niveau\_2.h, 242
- initialisation\_objets\_niveau\_3

- fonctions\_niveau\_3.c, [118](#)
- fonctions\_niveau\_3.h, [251](#)
- initialisation\_objets\_niveau\_4
  - fonctions\_niveau\_4.c, [143](#)
  - fonctions\_niveau\_4.h, [263](#)
- initialisation\_objets\_nouvelle\_partie
  - fonctions\_nouvelle\_partie.c, [160](#)
  - fonctions\_nouvelle\_partie.h, [268](#)
- initialisation\_objets\_options
  - fonctions\_options.c, [169](#)
  - fonctions\_options.h, [272](#)
- INTRODUCTION
  - fonctions\_generales.h, [218](#)
- introduction
  - fonctions\_introduction.c, [69](#)
  - fonctions\_introduction.h, [229](#)
- itemMenu, [6](#)
  - rectangle, [7](#)
  - texte, [7](#)
- main
  - metatravers.c, [182](#)
  - test\_pseudo.c, [277](#)
- MENU\_PRINCIPAL
  - fonctions\_generales.h, [218](#)
- menu\_principal
  - fonctions\_menu\_principal.c, [76](#)
  - fonctions\_menu\_principal.h, [232](#)
- metatravers.c
  - main, [182](#)
- mini\_jeu\_1\_niveau\_2
  - fonctions\_niveau\_2.c, [98](#)
  - fonctions\_niveau\_2.h, [243](#)
- mini\_jeu\_2\_niveau\_2
  - fonctions\_niveau\_2.c, [98](#)
  - fonctions\_niveau\_2.h, [243](#)
- mini\_jeu\_2\_niveau\_3
  - fonctions\_niveau\_3.c, [119](#)
  - fonctions\_niveau\_3.h, [252](#)
- mini\_jeux\_niveau\_2
  - fonctions\_niveau\_2.c, [99](#)
  - fonctions\_niveau\_2.h, [244](#)
- mini\_jeux\_niveau\_3
  - fonctions\_niveau\_3.c, [119](#)
  - fonctions\_niveau\_3.h, [252](#)
- mise\_a\_jour\_barre\_de\_son
  - fonctions\_options.c, [170](#)
  - fonctions\_options.h, [272](#)
- mise\_a\_jour\_bordures\_niveau\_2
  - fonctions\_niveau\_2.c, [100](#)
  - fonctions\_niveau\_2.h, [245](#)
- mise\_a\_jour\_bordures\_niveau\_3
  - fonctions\_niveau\_3.c, [120](#)
  - fonctions\_niveau\_3.h, [254](#)
- mise\_a\_jour\_mini\_jeu\_1\_niveau\_2
  - fonctions\_niveau\_2.c, [101](#)
  - fonctions\_niveau\_2.h, [246](#)
- mise\_a\_jour\_mini\_jeu\_1\_niveau\_3
  - fonctions\_niveau\_3.c, [121](#)
  - fonctions\_niveau\_3.h, [254](#)
- mise\_a\_jour\_mini\_jeu\_2\_niveau\_2
  - fonctions\_niveau\_2.c, [101](#)
  - fonctions\_niveau\_2.h, [246](#)
- mise\_a\_jour\_mini\_jeu\_2\_niveau\_3
  - fonctions\_niveau\_3.c, [121](#)
  - fonctions\_niveau\_3.h, [254](#)
- mise\_a\_jour\_rendu\_arrivee\_niveaux\_2\_3
  - fonctions\_arrivee\_niveaux\_2\_3.c, [12](#)
  - fonctions\_arrivee\_niveaux\_2\_3.h, [206](#)
- mise\_a\_jour\_rendu\_carte
  - fonctions\_carte.c, [33](#)
  - fonctions\_carte.h, [212](#)
- mise\_a\_jour\_rendu\_introduction
  - fonctions\_introduction.c, [70](#)
  - fonctions\_introduction.h, [230](#)
- mise\_a\_jour\_rendu\_menu\_principal
  - fonctions\_menu\_principal.c, [77](#)
  - fonctions\_menu\_principal.h, [233](#)
- mise\_a\_jour\_rendu\_niveau\_1
  - fonctions\_niveau\_1.c, [84](#)
  - fonctions\_niveau\_1.h, [238](#)
- mise\_a\_jour\_rendu\_niveau\_4
  - fonctions\_niveau\_4.c, [144](#)
  - fonctions\_niveau\_4.h, [263](#)
- mise\_a\_jour\_rendu\_nouvelle\_partie
  - fonctions\_nouvelle\_partie.c, [161](#)
  - fonctions\_nouvelle\_partie.h, [268](#)
- mise\_a\_jour\_rendu\_options
  - fonctions\_options.c, [170](#)
  - fonctions\_options.h, [273](#)
- mise\_a\_jour\_touches
  - fonctions\_options.c, [171](#)
  - fonctions\_options.h, [274](#)
- MODE\_HARD
  - fonctions\_generales.h, [217](#)
- MODE\_NORMAL
  - fonctions\_generales.h, [217](#)
- modes\_s
  - fonctions\_generales.h, [217](#)
- modes\_t
  - fonctions\_generales.h, [216](#)
- NIVEAU0
  - fonctions\_generales.h, [218](#)
- NIVEAU1
  - fonctions\_generales.h, [218](#)
- NIVEAU2
  - fonctions\_generales.h, [218](#)
- NIVEAU3
  - fonctions\_generales.h, [218](#)
- NIVEAU4
  - fonctions\_generales.h, [218](#)
- NIVEAU\_1
  - fonctions\_generales.h, [218](#)
- niveau\_1
  - fonctions\_niveau\_1.c, [85](#)
  - fonctions\_niveau\_1.h, [238](#)
- NIVEAU\_2

- fonctions\_generales.h, [218](#)
- NIVEAU\_3
  - fonctions\_generales.h, [218](#)
- NIVEAU\_4
  - fonctions\_generales.h, [218](#)
- niveau\_4
  - fonctions\_niveau\_4.c, [145](#)
  - fonctions\_niveau\_4.h, [264](#)
- niveau\_fini
  - niveaux, [8](#)
- niveaux, [7](#)
  - niveau\_fini, [8](#)
  - numero\_collectible, [8](#)
  - texture\_image\_collectible, [8](#)
- NOUVELLE\_PARTIE
  - fonctions\_generales.h, [218](#)
- nouvelle\_partie
  - fonctions\_nouvelle\_partie.c, [161](#)
  - fonctions\_nouvelle\_partie.h, [269](#)
- numero\_collectible
  - niveaux, [8](#)
- ONGLET\_SON
  - fonctions\_generales.h, [218](#)
- ONGLET\_TOUCHES
  - fonctions\_generales.h, [218](#)
- option\_s
  - fonctions\_generales.h, [217](#)
- option\_t
  - fonctions\_generales.h, [216](#)
- OPTIONS
  - fonctions\_generales.h, [218](#)
- options
  - fonctions\_options.c, [172](#)
  - fonctions\_options.h, [274](#)
- page\_s
  - fonctions\_generales.h, [218](#)
- page\_t
  - fonctions\_generales.h, [216](#)
- PERSONNAGE\_1
  - fonctions\_generales.h, [218](#)
- PERSONNAGE\_2
  - fonctions\_generales.h, [218](#)
- personnage\_s
  - fonctions\_generales.h, [218](#)
- personnage\_t
  - fonctions\_generales.h, [217](#)
- piece\_proche\_position\_correcte
  - fonctions\_niveau\_3.c, [122](#)
  - fonctions\_niveau\_3.h, [255](#)
- position\_s
  - fonctions\_generales.h, [218](#)
- position\_t
  - fonctions\_generales.h, [217](#)
- rectangle
  - itemMenu, [7](#)
- rectangle\_piece\_aleatoire
- fonctions\_niveau\_3.c, [122](#)
- fonctions\_niveau\_3.h, [255](#)
- redimensionnement\_fenetre
  - fonctions\_generales.c, [57](#)
  - fonctions\_generales.h, [225](#)
- salon\_arrivee\_niveaux\_2\_3
  - fonctions\_arrivee\_niveaux\_2\_3.c, [13](#)
  - fonctions\_arrivee\_niveaux\_2\_3.h, [207](#)
  - fonctions\_niveau\_2.h, [247](#)
  - fonctions\_niveau\_3.h, [256](#)
- sauvegarder\_partie
  - fonctions\_generales.c, [59](#)
  - fonctions\_generales.h, [225](#)
- test\_unitaire/test\_pseudo.c, [276](#), [277](#)
- test\_pseudo.c
  - BUFFER\_SIZE, [277](#)
  - main, [277](#)
- texte
  - itemMenu, [7](#)
- texture\_image\_collectible
  - niveaux, [8](#)
- traitement\_touches
  - fonctions\_niveau\_3.c, [122](#)
  - fonctions\_niveau\_3.h, [256](#)
- verification\_chemin
  - fonctions\_niveau\_2.c, [102](#)
  - fonctions\_niveau\_2.h, [247](#)
- verification\_puzzle\_fini
  - fonctions\_niveau\_3.c, [123](#)
  - fonctions\_niveau\_3.h, [257](#)
- verification\_sauvegarde
  - fonctions\_generales.c, [59](#)
  - fonctions\_generales.h, [225](#)
- volume
  - barreDeSon, [6](#)
- volume\_precedent
  - barreDeSon, [6](#)