

Bloody Sanada

Généré par Doxygen 1.8.17

1 Liste des choses à faire	1
2 Liste des éléments obsolètes	3
3 Index des classes	5
3.1 Liste des classes	5
4 Index des fichiers	7
4.1 Liste des fichiers	7
5 Documentation des classes	9
5.1 Référence de la structure base_coffre_s	9
5.1.1 Description détaillée	9
5.1.2 Documentation des données membres	9
5.1.2.1 fichier_image	9
5.1.2.2 hitbox	9
5.1.2.3 nom_coffre	10
5.2 Référence de la structure base_monstre_s	10
5.2.1 Documentation des données membres	10
5.2.1.1 attaque	10
5.2.1.2 fichier_image	10
5.2.1.3 gainXp	10
5.2.1.4 hitbox	10
5.2.1.5 nom_monstre	11
5.2.1.6 pdv	11
5.2.1.7 vitesse	11
5.3 Référence de la structure base_monstre_t	11
5.3.1 Description détaillée	11
5.4 Référence de la structure base_sort_s	11
5.4.1 Documentation des données membres	11
5.4.1.1 collision	12
5.4.1.2 degat	12
5.4.1.3 type	12
5.5 Référence de la structure boss_s	12
5.5.1 Documentation des données membres	13
5.5.1.1 action	13
5.5.1.2 attaque	13
5.5.1.3 cible	13
5.5.1.4 collision	13
5.5.1.5 duree	13
5.5.1.6 pdv	13
5.5.1.7 texture	13
5.5.1.8 texture_temp	14
5.5.1.9 type	14

5.5.1.10 xp	14
5.6 Référence de la structure coffre_s	14
5.6.1 Documentation des données membres	14
5.6.1.1 collision	14
5.6.1.2 etat	14
5.6.1.3 id_cle	15
5.6.1.4 id_loot	15
5.6.1.5 orientation	15
5.6.1.6 texture	15
5.6.1.7 type	15
5.7 Référence de la structure coffre_t	15
5.7.1 Description détaillée	15
5.8 Référence de la structure element	16
5.8.1 Description détaillée	16
5.8.2 Documentation des données membres	16
5.8.2.1 pred	16
5.8.2.2 succ	16
5.8.2.3 valeur	17
5.9 Référence de la structure inventaire_t	17
5.9.1 Documentation des données membres	17
5.9.1.1 equipe	17
5.9.1.2 sac	17
5.10 Référence de la structure inventaire_t	18
5.10.1 Description détaillée	18
5.11 Référence de la structure joueur_t	18
5.11.1 Description détaillée	19
5.11.2 Documentation des données membres	19
5.11.2.1 attaque	19
5.11.2.2 attaque_actif	19
5.11.2.3 defense	19
5.11.2.4 defense_actif	19
5.11.2.5 inventaire	19
5.11.2.6 maxPdv	20
5.11.2.7 niveau	20
5.11.2.8 nom_pers	20
5.11.2.9 pdv	20
5.11.2.10 statut	20
5.11.2.11 textures_joueur	20
5.11.2.12 trigger	20
5.11.2.13 vitesse	20
5.11.2.14 vitesse_actif	21
5.11.2.15 xp	21

5.12 Référence de la structure list	21
5.12.1 Description détaillée	21
5.12.2 Documentation des données membres	21
5.12.2.1 aff	21
5.12.2.2 ajout	22
5.12.2.3 del	22
5.12.2.4 ec	22
5.12.2.5 flag	22
5.12.2.6 nb_elem	22
5.13 Référence de la structure liste_base_coffres_s	22
5.13.1 Description détaillée	22
5.13.2 Documentation des données membres	23
5.13.2.1 nb_coffre	23
5.13.2.2 tab	23
5.14 Référence de la structure liste_base_monstres_t	23
5.14.1 Documentation des données membres	23
5.14.1.1 nb_monstre	24
5.14.1.2 tab	24
5.15 Référence de la structure liste_base_monstres_t	24
5.15.1 Description détaillée	24
5.16 Référence de la structure lobjet_t	24
5.16.1 Description détaillée	25
5.16.2 Documentation des données membres	25
5.16.2.1 liste	25
5.16.2.2 nb	25
5.17 Référence de la structure lobjet_t	25
5.17.1 Description détaillée	25
5.18 Référence de la structure monstre_s	26
5.18.1 Documentation des données membres	26
5.18.1.1 action	26
5.18.1.2 attaque	26
5.18.1.3 collision	26
5.18.1.4 duree	26
5.18.1.5 gainXp	26
5.18.1.6 orientation	27
5.18.1.7 pdv	27
5.18.1.8 texture	27
5.18.1.9 type	27
5.18.1.10 vitesse	27
5.19 Référence de la structure monstre_t	27
5.19.1 Description détaillée	27
5.20 Référence de la structure objet_t	28

5.20.1 Description détaillée	28
5.20.2 Documentation des données membres	28
5.20.2.1 attaque	28
5.20.2.2 defense	28
5.20.2.3 id	28
5.20.2.4 niveau	28
5.20.2.5 nom	29
5.20.2.6 texture	29
5.20.2.7 texture_src	29
5.20.2.8 type	29
5.20.2.9 vitesse	29
5.21 Référence de la structure objet_t	29
5.21.1 Description détaillée	29
5.22 Référence de la structure point	30
5.22.1 Documentation des données membres	30
5.22.1.1 x	30
5.22.1.2 y	30
5.23 Référence de la structure sort_s	30
5.23.1 Documentation des données membres	30
5.23.1.1 collision	30
5.23.1.2 degat	31
5.23.1.3 texture	31
5.23.1.4 type	31
5.24 Référence de la structure sort_t	31
5.24.1 Description détaillée	31
5.25 Référence de la structure statut_s	31
5.25.1 Description détaillée	32
5.25.2 Documentation des données membres	32
5.25.2.1 action	32
5.25.2.2 animation	32
5.25.2.3 bouclier_equipe	32
5.25.2.4 duree	32
5.25.2.5 duree_anim	32
5.25.2.6 en_mouvement	32
5.25.2.7 orient_att	33
5.25.2.8 orient_dep	33
5.25.2.9 texture_prec	33
5.25.2.10 vrai_zone_collision	33
5.25.2.11 zone_colision	33
5.26 Référence de la structure struct	33
5.26.1 Description détaillée	33
5.27 Référence de la structure t_aff	34

5.27.1 Description détaillée	34
5.27.2 Le but de la structure	34
5.27.3 Éléments de la structure	34
5.27.4 Documentation des données membres	34
5.27.4.1 aff_fenetre	35
5.27.4.2 compteur_frame_anim	35
5.27.4.3 duree_frame_anim	35
5.27.4.4 frame_anim	35
5.27.4.5 height	35
5.27.4.6 multipli_taille	35
5.27.4.7 texture	35
5.27.4.8 width	36
5.28 Référence de la structure t_l_aff	36
5.28.1 Description détaillée	36
5.28.2 Documentation des données membres	36
5.28.2.1 liste	36
5.28.2.2 nb_valeurs	36
5.29 Référence de la structure t_map	37
5.29.1 Description détaillée	37
5.29.2 Documentation des données membres	38
5.29.2.1 cases_x	38
5.29.2.2 cases_y	38
5.29.2.3 height	38
5.29.2.4 id_map	38
5.29.2.5 liste_coffres	38
5.29.2.6 liste_collisions	38
5.29.2.7 liste_monstres	38
5.29.2.8 liste_sorts	39
5.29.2.9 liste_zone_tp	39
5.29.2.10 taille_case	39
5.29.2.11 text_map	39
5.29.2.12 text_sol	39
5.29.2.13 texture_superposition	39
5.29.2.14 width	39
5.30 Référence de la structure zone_tp	40
5.30.1 Description détaillée	40
5.30.2 Documentation des données membres	40
5.30.2.1 dest	40
5.30.2.2 id_map	40
5.30.2.3 zone	40

6.1 Référence du fichier affichage.c	41
6.1.1 Description détaillée	43
6.1.2 Documentation des fonctions	44
6.1.2.1 afficher_animations()	44
6.1.2.2 afficher_buffer()	45
6.1.2.3 afficher_coffres()	46
6.1.2.4 afficher_monstres()	47
6.1.2.5 afficher_sorts()	48
6.1.2.6 afficher_texture()	49
6.1.2.7 ajout_text_liste()	50
6.1.2.8 appliquer_coord_rect()	51
6.1.2.9 color()	52
6.1.2.10 creer_texture()	52
6.1.2.11 Déroulement	53
6.1.2.12 Lors d'une erreur	53
6.1.2.13 A noter	53
6.1.2.14 current_frame_x()	54
6.1.2.15 current_frame_y()	55
6.1.2.16 def_texture_taille()	55
6.1.2.17 deplacement_x_entite()	56
6.1.2.18 deplacement_x_joueur_secondaire()	57
6.1.2.19 deplacement_x_pers()	58
6.1.2.20 deplacement_y_entite()	60
6.1.2.21 deplacement_y_joueur_secondaire()	61
6.1.2.22 deplacement_y_pers()	62
6.1.2.23 replacer_rect_haut_droit()	63
6.1.2.24 replacer_rect_origine()	64
6.1.2.25 replacer_texture_bas_droit()	65
6.1.2.26 replacer_texture_bas_gauche()	65
6.1.2.27 replacer_texture_centre()	66
6.1.2.28 replacer_texture_haut_droit()	67
6.1.2.29 replacer_texture_origine()	67
6.1.2.30 detruire_collision_dans_liste()	68
6.1.2.31 detruire_liste_textures()	69
6.1.2.32 detruire_texture()	70
6.1.2.33 get_rect_center()	70
6.1.2.34 get_rect_center_coord()	71
6.1.2.35 get_screen_center()	72
6.1.2.36 info_texture()	72
6.1.2.37 init_animations()	73
6.1.2.38 init_texture_joueur()	73
6.1.2.39 init_textures_joueur()	74

6.1.2.40 lister_animations()	75
6.1.2.41 modif_affichage_rect()	76
6.1.2.42 next_frame_animation()	76
6.1.2.43 next_frame_indice()	77
6.1.2.44 next_frame_joueur()	78
6.1.2.45 next_frame_x()	79
6.1.2.46 next_frame_x_indice()	80
6.1.2.47 next_frame_y()	81
6.1.2.48 next_frame_y_indice()	82
6.1.2.49 place_rect_center_from_point()	83
6.1.2.50 placer_texture()	84
6.1.2.51 point_in_rect()	85
6.1.2.52 rect_centre()	85
6.1.2.53 rect_centre_rect()	86
6.1.2.54 rect_centre_rect_x()	87
6.1.2.55 rect_centre_rect_y()	88
6.1.2.56 rect_centre_x()	89
6.1.2.57 rect_centre_y()	89
6.1.2.58 rect_correct_texture()	90
6.1.2.59 rect_ecran_to_rect_map()	91
6.1.2.60 rects_egal_x()	91
6.1.2.61 rects_egal_y()	92
6.1.2.62 text_copier_position()	93
6.1.2.63 update_frame_texture()	94
6.1.3 Documentation des variables	94
6.1.3.1 bloquer	94
6.1.3.2 compteur	94
6.1.3.3 fenetre_finale	94
6.1.3.4 FENETRE_LARGEUR	94
6.1.3.5 FENETRE_LONGUEUR	95
6.1.3.6 heal	95
6.1.3.7 liste_animations	95
6.1.3.8 listeDeTextures	95
6.1.3.9 multiplicateur_x	95
6.1.3.10 multiplicateur_y	95
6.1.3.11 tx	95
6.1.3.12 ty	95
6.2 Référence du fichier affichage.h	96
6.2.1 Description détaillée	99
6.2.2 Documentation des macros	99
6.2.2.1 LARGEUR_ENTITE	99
6.2.2.2 LONGUEUR_ENTITE	99

6.2.2.3 N_T_ATTAQUE	100
6.2.2.4 N_T_ATTAQUE2	100
6.2.2.5 N_T_ATTAQUE_CHARGEED	100
6.2.2.6 N_T_ATTAQUE_CHARGEED2	100
6.2.2.7 N_T_CHARGER	100
6.2.2.8 N_T_CHARGER2	100
6.2.2.9 N_T_MARCHER	100
6.2.2.10 N_T_MARCHER2	100
6.2.2.11 N_T_MARCHER_BOUCLIER	101
6.2.2.12 N_T_MARCHER_BOUCLIER2	101
6.2.2.13 NB_FPS	101
6.2.2.14 NB_SPRITE_JOUEUR	101
6.2.3 Documentation du type de l'énumération	101
6.2.3.1 t_texture_perso	101
6.2.4 Documentation des fonctions	102
6.2.4.1 afficher_animations()	102
6.2.4.2 afficher_buffer()	103
6.2.4.3 afficher_coffres()	103
6.2.4.4 afficher_monstres()	104
6.2.4.5 afficher_sorts()	105
6.2.4.6 afficher_texture()	106
6.2.4.7 ajout_text_liste()	107
6.2.4.8 color()	108
6.2.4.9 creer_texture()	109
6.2.4.10 Déroulement	109
6.2.4.11 Lors d'une erreur	109
6.2.4.12 A noter	110
6.2.4.13 current_frame_x()	112
6.2.4.14 current_frame_y()	113
6.2.4.15 def_texture_taille()	113
6.2.4.16 deplacement_x_entite()	114
6.2.4.17 deplacement_x_joueur_secondaire()	115
6.2.4.18 deplacement_x_pers()	116
6.2.4.19 deplacement_y_entite()	118
6.2.4.20 deplacement_y_joueur_secondaire()	119
6.2.4.21 deplacement_y_pers()	120
6.2.4.22 déplacer_rect_haut_droit()	121
6.2.4.23 déplacer_rect_origine()	122
6.2.4.24 déplacer_texture_bas_droit()	123
6.2.4.25 déplacer_texture_bas_gauche()	123
6.2.4.26 déplacer_texture_centre()	124
6.2.4.27 déplacer_texture_haut_droit()	125

6.2.4.28 <code>deplacer_texture_origine()</code>	125
6.2.4.29 <code>detruire_collision_dans_liste()</code>	126
6.2.4.30 <code>detruire_liste_textures()</code>	127
6.2.4.31 <code>detruire_texture()</code>	128
6.2.4.32 <code>get_rect_center()</code>	128
6.2.4.33 <code>get_rect_center_coord()</code>	129
6.2.4.34 <code>info_texture()</code>	130
6.2.4.35 <code>init_animations()</code>	130
6.2.4.36 <code>init_texture_joueur()</code>	131
6.2.4.37 <code>init_textures_joueur()</code>	131
6.2.4.38 <code>lister_animations()</code>	132
6.2.4.39 <code>modif_affichage_rect()</code>	133
6.2.4.40 <code>next_frame_animation()</code>	133
6.2.4.41 <code>next_frame_indice()</code>	134
6.2.4.42 <code>next_frame_joueur()</code>	135
6.2.4.43 <code>next_frame_x()</code>	136
6.2.4.44 <code>next_frame_x_indice()</code>	137
6.2.4.45 <code>next_frame_y()</code>	138
6.2.4.46 <code>next_frame_y_indice()</code>	139
6.2.4.47 <code>place_rect_center_from_point()</code>	140
6.2.4.48 <code>placer_texture()</code>	141
6.2.4.49 <code>rect_centre()</code>	142
6.2.4.50 <code>rect_centre_rect()</code>	143
6.2.4.51 <code>rect_centre_rect_x()</code>	144
6.2.4.52 <code>rect_centre_rect_y()</code>	144
6.2.4.53 <code>rect_centre_x()</code>	145
6.2.4.54 <code>rect_centre_y()</code>	146
6.2.4.55 <code>rect_correct_texture()</code>	147
6.2.4.56 <code>rects_egal_x()</code>	148
6.2.4.57 <code>rects_egal_y()</code>	149
6.2.4.58 <code>text_copier_position()</code>	150
6.2.5 Documentation des variables	150
6.2.5.1 <code>bloquer</code>	150
6.2.5.2 <code>compteur</code>	150
6.2.5.3 <code>fenetre_finale</code>	151
6.2.5.4 <code>heal</code>	151
6.2.5.5 <code>liste_animations</code>	151
6.2.5.6 <code>listeDeTextures</code>	151
6.2.5.7 <code>multiplicateur_x</code>	151
6.2.5.8 <code>multiplicateur_y</code>	151
6.2.5.9 <code>tx</code>	151
6.2.5.10 <code>ty</code>	151

6.3 Référence du fichier boss.c	152
6.3.1 Documentation des fonctions	152
6.3.1.1 affichage_chargement_attaque_boss()	152
6.3.1.2 affichage_clonage_boss()	153
6.3.1.3 afficher_boss()	154
6.3.1.4 afficher_bosses()	154
6.3.1.5 creer_boss_clone()	155
6.3.1.6 deplacement_boss_aleatoire()	155
6.3.1.7 intro_boss()	155
6.4 Référence du fichier boss.h	156
6.4.1 Description détaillée	157
6.4.2 Documentation des macros	157
6.4.2.1 CHEMIN_BOSS	157
6.4.2.2 DUREE_CLONAGE	157
6.4.3 Documentation du type de l'énumération	157
6.4.3.1 action_boss_1_t	157
6.4.3.2 type_boss_1_t	158
6.5 Référence du fichier code_erreur.h	158
6.5.1 Description détaillée	159
6.5.2 Documentation des macros	159
6.5.2.1 erreur	159
6.5.2.2 warning	160
6.5.3 Documentation des définitions de type	160
6.5.3.1 err_t	160
6.5.4 Documentation du type de l'énumération	160
6.5.4.1 types_erreur	160
6.6 Référence du fichier coffres.c	161
6.6.1 Description détaillée	162
6.6.2 Documentation des fonctions	163
6.6.2.1 charger_base_coffre()	163
6.6.2.2 creer_coffre()	163
6.6.2.3 info_coffre()	165
6.6.2.4 interaction_coffre()	165
6.6.2.5 inverser_direction()	166
6.6.2.6 nom_coffre_to_type_coffre()	167
6.6.3 Documentation des variables	168
6.6.3.1 liste_base_coffres	168
6.7 Référence du fichier coffres.h	168
6.7.1 Description détaillée	170
6.7.2 Documentation des macros	170
6.7.2.1 COFFRE_FACE_OUVERT	170
6.7.2.2 COFFRE_PROFIL_OUVERT	170

6.7.3 Documentation du type de l'énumération	170
6.7.3.1 etat_coffre_t	170
6.7.3.2 type_coffre_t	171
6.7.4 Documentation des fonctions	171
6.7.4.1 charger_base_coffre()	171
6.7.4.2 creer_coffre()	172
6.7.4.3 info_coffre()	173
6.7.4.4 interaction_coffre()	173
6.7.4.5 inverser_direction()	174
6.7.4.6 nom_coffre_to_type_coffre()	175
6.7.5 Documentation des variables	176
6.7.5.1 liste_base_coffres	176
6.8 Référence du fichier commun.h	176
6.8.1 Description détaillée	177
6.8.2 Documentation des fonctions	177
6.8.2.1 init()	177
6.9 Référence du fichier definition_commun.h	178
6.9.1 Description détaillée	180
6.9.2 Documentation des macros	180
6.9.2.1 bool	180
6.9.2.2 faux	180
6.9.2.3 SAVE_PATH	181
6.9.2.4 vrai	181
6.9.3 Documentation des définitions de type	181
6.9.3.1 byte	181
6.9.4 Documentation du type de l'énumération	181
6.9.4.1 t_direction_1	181
6.9.4.2 t_direction_2	181
6.9.5 Documentation des fonctions	182
6.9.5.1 fermer_programme()	182
6.9.6 Documentation des variables	183
6.9.6.1 FENETRE_LARGEUR	183
6.9.6.2 FENETRE_LONGUEUR	183
6.9.6.3 fenetre_Principale	184
6.9.6.4 rendu_principal	184
6.9.6.5 running	184
6.10 Référence du fichier event.c	184
6.10.1 Description détaillée	185
6.10.2 Documentation des fonctions	185
6.10.2.1 jeu_event()	185
6.10.2.2 jeu_event_manette()	186
6.10.2.3 joystick_button_down()	187

6.10.2.4 joystick_button_up()	188
6.10.2.5 joystick_stick()	189
6.10.2.6 keyDown()	190
6.10.2.7 keyUp()	191
6.10.2.8 logo_passer()	191
6.10.2.9 mouseButtonDown()	192
6.10.2.10 mouseButtonUp()	193
6.10.3 Documentation des variables	194
6.10.3.1 manette	194
6.11 Référence du fichier event.h	194
6.11.1 Description détaillée	195
6.11.2 Documentation des macros	196
6.11.2.1 TOUCHE_BAS	196
6.11.2.2 TOUCHE_CONSUMMABLE	196
6.11.2.3 TOUCHE_DROITE	196
6.11.2.4 TOUCHE_ECHAP	196
6.11.2.5 TOUCHE_GAUCHE	196
6.11.2.6 TOUCHE_HAUT	196
6.11.2.7 TOUCHE_TAB	196
6.11.3 Documentation des fonctions	196
6.11.3.1 jeu_event()	196
6.11.3.2 jeu_event_manette()	197
6.11.3.3 logo_passer()	198
6.11.4 Documentation des variables	199
6.11.4.1 manette	199
6.12 Référence du fichier init_close.c	199
6.12.1 Description détaillée	200
6.12.2 Documentation des fonctions	201
6.12.2.1 aff_cleanup()	201
6.12.2.2 detruire_renderer()	201
6.12.2.3 fermer_programme()	202
6.12.2.4 fermer SDL()	203
6.12.2.5 init()	204
6.12.2.6 init_affichage()	205
6.12.2.7 init_rc_commun()	205
6.12.2.8 init SDL()	206
6.12.3 Documentation des variables	206
6.12.3.1 f_close	206
6.12.3.2 fenetre_Principale	206
6.12.3.3 fenetre_sous_rendu	206
6.12.3.4 hors_hitbox	206
6.12.3.5 rendu_principal	207

6.12.3.6 running	207
6.12.3.7 sous_rendu	207
6.13 Référence du fichier interface.c	207
6.13.1 Description détaillée	208
6.13.2 Documentation des fonctions	208
6.13.2.1 RenderHPBar()	208
6.14 Référence du fichier interface.h	209
6.14.1 Description détaillée	210
6.14.2 Documentation des fonctions	210
6.14.2.1 RenderHPBar()	210
6.15 Référence du fichier inventaire.c	211
6.15.1 Description détaillée	212
6.15.2 Documentation des fonctions	212
6.15.2.1 changement_statistiques()	212
6.15.2.2 consommer_objet()	213
6.15.2.3 creer_inventaire()	213
6.15.2.4 desequiper()	214
6.15.2.5 desequiper_slot()	215
6.15.2.6 detruire_inventaire()	216
6.15.2.7 equiper_objet()	217
6.15.2.8 equiper_sac_slot()	218
6.15.2.9 ramasser_objet()	219
6.15.2.10 tout_ramasser()	220
6.16 Référence du fichier inventaire.h	220
6.16.1 Description détaillée	222
6.16.2 Documentation des macros	222
6.16.2.1 CAPACITE_SAC	222
6.16.3 Documentation des fonctions	222
6.16.3.1 changement_statistiques()	222
6.16.3.2 consommer_objet()	223
6.16.3.3 creer_inventaire()	223
6.16.3.4 desequiper()	224
6.16.3.5 desequiper_slot()	225
6.16.3.6 detruire_inventaire()	225
6.16.3.7 equiper_objet()	226
6.16.3.8 equiper_sac_slot()	227
6.16.3.9 ramasser_objet()	228
6.16.3.10 tout_ramasser()	229
6.17 Référence du fichier liste_objet.c	230
6.17.1 Description détaillée	230
6.17.2 Documentation des fonctions	231
6.17.2.1 afficher_liste_objet()	231

6.17.2.2 afficher_textures_equipe()	231
6.17.2.3 afficher_textures_sac()	232
6.17.2.4 creer_liste_objet()	233
6.17.2.5 creer_liste_objet_equipe()	234
6.17.2.6 creer_liste_objet_vide()	235
6.17.2.7 creer_textures_objets()	235
6.17.2.8 detruire_liste_objet()	236
6.17.2.9 effacer_liste_objet()	237
6.17.2.10 placer_objet_sac()	237
6.17.3 Documentation des variables	238
6.17.3.1 objets	238
6.18 Référence du fichier liste_objet.h	238
6.18.1 Description détaillée	240
6.18.2 Documentation des fonctions	240
6.18.2.1 afficher_liste_objet()	240
6.18.2.2 afficher_textures_equipe()	241
6.18.2.3 afficher_textures_sac()	241
6.18.2.4 creer_liste_objet()	242
6.18.2.5 creer_liste_objet_equipe()	243
6.18.2.6 creer_liste_objet_vide()	244
6.18.2.7 creer_textures_objets()	244
6.18.2.8 detruire_liste_objet()	245
6.18.2.9 effacer_liste_objet()	246
6.18.2.10 placer_objet_sac()	246
6.18.3 Documentation des variables	247
6.18.3.1 objets	247
6.19 Référence du fichier listes.c	247
6.19.1 Description détaillée	248
6.19.2 Documentation des fonctions	248
6.19.2.1 afficher_liste()	249
6.19.2.2 ajout_droit()	249
6.19.2.3 ajout_gauche()	250
6.19.2.4 detruire_liste()	250
6.19.2.5 en_queue()	251
6.19.2.6 en_tete()	252
6.19.2.7 hors_liste()	254
6.19.2.8 init_liste()	256
6.19.2.9 liste_vide()	257
6.19.2.10 modif_elt()	258
6.19.2.11 oter_elt()	259
6.19.2.12 precedent()	260
6.19.2.13 selectionner_element()	261

6.19.2.14 suivant()	262
6.19.2.15 taille_liste()	263
6.19.2.16 valeur_elt()	264
6.19.2.17 vider_liste()	265
6.20 Référence du fichier listes.h	266
6.20.1 Description détaillée	268
6.20.2 Documentation des fonctions	268
6.20.2.1 afficher_liste()	268
6.20.2.2 ajout_droit()	269
6.20.2.3 ajout_gauche()	270
6.20.2.4 detruire_liste()	270
6.20.2.5 en_queue()	271
6.20.2.6 en_tete()	272
6.20.2.7 hors_liste()	274
6.20.2.8 init_liste()	276
6.20.2.9 liste_vide()	277
6.20.2.10 modif_elt()	278
6.20.2.11 oter_elt()	279
6.20.2.12 precedent()	280
6.20.2.13 selectionner_element()	281
6.20.2.14 suivant()	282
6.20.2.15 taille_liste()	283
6.20.2.16 valeur_elt()	284
6.20.2.17 vider_liste()	285
6.21 Référence du fichier main.c	286
6.21.1 Description détaillée	287
6.21.2 Documentation des macros	287
6.21.2.1 SDL_MAIN_HANDLED	288
6.21.3 Documentation des fonctions	288
6.21.3.1 afficher_intro()	288
6.21.3.2 main()	289
6.22 Référence du fichier map.c	290
6.22.1 Description détaillée	290
6.22.2 Documentation des fonctions	291
6.22.2.1 afficher_zone_tp()	291
6.22.2.2 charger_map()	291
6.22.2.3 detruire_map()	293
6.22.2.4 hors_map_monstre()	294
6.22.2.5 init_sousbuffer()	294
6.22.2.6 taille_ecran_cases()	295
6.22.2.7 texture_map()	295
6.22.2.8 tp_joueurs()	296

6.22.2.9 transition()	297
6.22.3 Documentation des variables	298
6.22.3.1 map	298
6.23 Référence du fichier map.h	298
6.23.1 Description détaillée	299
6.23.2 Documentation des macros	299
6.23.2.1 TAILLE_CASE	300
6.23.3 Documentation des fonctions	300
6.23.3.1 charger_map()	300
6.23.3.2 detruire_map()	301
6.23.3.3 init_sousbuffer()	302
6.23.3.4 texture_map()	303
6.23.3.5 tp_joueurs()	304
6.23.3.6 transition()	304
6.23.4 Documentation des variables	305
6.23.4.1 map	305
6.24 Référence du fichier menus.c	305
6.24.1 Description détaillée	306
6.24.2 Documentation des fonctions	306
6.24.2.1 afficher_inventaire()	307
6.24.2.2 afficher_inventaire_manette()	307
6.24.2.3 afficher_menu_accueil()	308
6.24.2.4 afficher_menu_accueil_manette()	309
6.24.2.5 afficher_menu_pause()	310
6.24.2.6 afficher_menu_pause_manette()	311
6.24.2.7 creer_inventaire_j2()	312
6.24.2.8 draw_rect_epaisseur()	313
6.24.2.9 init_text_menus()	314
6.24.3 Documentation des variables	315
6.24.3.1 text_accueil	315
6.24.3.2 text_inventaire1	315
6.24.3.3 text_inventaire2	315
6.24.3.4 text_pause	315
6.25 Référence du fichier menus.h	315
6.25.1 Description détaillée	316
6.25.2 Documentation des fonctions	317
6.25.2.1 afficher_inventaire()	317
6.25.2.2 afficher_inventaire_manette()	317
6.25.2.3 afficher_menu_accueil()	318
6.25.2.4 afficher_menu_accueil_manette()	319
6.25.2.5 afficher_menu_pause()	320
6.25.2.6 afficher_menu_pause_manette()	321

6.25.2.7 creer_inventaire_j2()	322
6.25.2.8 init_text_menus()	323
6.25.3 Documentation des variables	323
6.25.3.1 text_accueil	323
6.25.3.2 text_inventaire1	324
6.25.3.3 text_inventaire2	324
6.25.3.4 text_pause	324
6.26 Référence du fichier monstres.c	324
6.26.1 Description détaillée	325
6.26.2 Documentation des fonctions	326
6.26.2.1 action_monstre()	326
6.26.2.2 agro_knight()	327
6.26.2.3 agro_monstre()	328
6.26.2.4 agro_witcher()	328
6.26.2.5 ajouter_monstre()	329
6.26.2.6 ajouter_monstre_cb()	330
6.26.2.7 charger_base_monstre()	331
6.26.2.8 creer_monstre()	331
6.26.2.9 detruire_liste_base_monstres()	332
6.26.2.10 detruire_monstre()	332
6.26.2.11 detruire_monstre_cb()	332
6.26.2.12 fuir_joueur()	333
6.26.2.13 marcher_monstre()	334
6.26.2.14 monstre_attaque()	335
6.26.2.15 monstre_en_garde()	336
6.26.2.16 nom_monstre_to_type_monstre()	337
6.26.2.17 orienter_monstre()	337
6.26.2.18 orienter_monstre_vers_joueur()	338
6.26.2.19 ronde_monstre()	339
6.26.2.20 rush_joueur()	340
6.26.3 Documentation des variables	341
6.26.3.1 liste_base_monstres	341
6.27 Référence du fichier monstres.h	341
6.27.1 Description détaillée	343
6.27.2 Documentation des macros	343
6.27.2.1 DISTANCE_AGRO	344
6.27.2.2 DUREE_MONSTRE_ATTAQUE	344
6.27.2.3 DUREE_MONSTRE_BLESSE	344
6.27.2.4 DUREE_MONSTRE_EN_GARDE	344
6.27.2.5 DUREE_MONSTRE_MARCHER	344
6.27.2.6 DUREE_MONSTRE_PAUSE	344
6.27.2.7 DUREE_RUSH_OU_FUITE	344

6.27.3 Documentation du type de l'énumération	344
6.27.3.1 action_monstre_t	344
6.27.3.2 type_monstre_t	345
6.27.4 Documentation des fonctions	345
6.27.4.1 action_monstre()	345
6.27.4.2 ajouter_monstre_cb()	346
6.27.4.3 charger_base_monstre()	347
6.27.4.4 creer_monstre()	348
6.27.4.5 detruire_liste_base_monstres()	349
6.27.4.6 detruire_monstre_cb()	349
6.27.4.7 nom_monstre_to_type_monstre()	349
6.27.5 Documentation des variables	350
6.27.5.1 liste_base_monstres	350
6.28 Référence du fichier objet.c	350
6.28.1 Description détaillée	351
6.28.2 Documentation des fonctions	351
6.28.2.1 afficher_objet()	351
6.28.2.2 creer_objet()	352
6.28.2.3 detruire_objet()	353
6.29 Référence du fichier objet.h	354
6.29.1 Description détaillée	355
6.29.2 Documentation des macros	355
6.29.2.1 NB_TYPE_OBJ	355
6.29.3 Documentation du type de l'énumération	355
6.29.3.1 t_item	355
6.29.4 Documentation des fonctions	356
6.29.4.1 afficher_objet()	356
6.29.4.2 creer_objet()	356
6.29.4.3 detruire_objet()	357
6.30 Référence du fichier personnage.c	358
6.30.1 Description détaillée	359
6.30.2 Documentation des fonctions	359
6.30.2.1 afficher_statistiques()	360
6.30.2.2 charger_sauvegarde_joueur()	360
6.30.2.3 check_repertoire_jeux()	361
6.30.2.4 copy()	362
6.30.2.5 creer_joueur()	362
6.30.2.6 creer_sauvegarde_json()	363
6.30.2.7 detruire_joueur()	364
6.30.2.8 distance_joueur()	365
6.30.2.9 distance_x_joueur()	366
6.30.2.10 distance_y_joueur()	367

6.30.2.11 entite_en_collision()	368
6.30.2.12 entite_subit_attaque()	369
6.30.2.13 environnement_joueurs()	369
6.30.2.14 gain_xp()	370
6.30.2.15 levelup()	371
6.30.2.16 maj_statistiques()	372
6.30.2.17 new_joueur()	372
6.30.2.18 sauve_existe()	373
6.30.2.19 stoper_mouvement_joueurs()	373
6.30.2.20 zone_en_dehors_hitbox()	374
6.30.3 Documentation des variables	374
6.30.3.1 save_path	374
6.31 Référence du fichier personnage.h	375
6.31.1 Description détaillée	376
6.31.2 Documentation des macros	377
6.31.2.1 DUREE_ATTAQUE	377
6.31.2.2 DUREE_ATTAQUE_CHARGEE	377
6.31.2.3 DUREE_ATTAQUE_OU_CHARGEE	377
6.31.2.4 DUREE_BLOQUER	377
6.31.2.5 DUREE_JOUEUR_BLESSE	377
6.31.2.6 DUREE_SOIN	377
6.31.2.7 TAILLE_PERSONNAGE	378
6.31.2.8 TAILLE_TRIGGER	378
6.31.3 Documentation des définitions de type	378
6.31.3.1 byte	378
6.31.4 Documentation du type de l'énumération	378
6.31.4.1 action_t	378
6.31.5 Documentation des fonctions	378
6.31.5.1 afficher_statistiques()	379
6.31.5.2 charger_sauvegarde_joueur()	379
6.31.5.3 check_reperoire_jeux()	380
6.31.5.4 creer_joueur()	381
6.31.5.5 creer_sauvegarde_json()	382
6.31.5.6 detruire_joueur()	382
6.31.5.7 distance_joueur()	383
6.31.5.8 distance_x_joueur()	384
6.31.5.9 distance_y_joueur()	385
6.31.5.10 environnement_joueurs()	386
6.31.5.11 gain_xp()	387
6.31.5.12 levelup()	388
6.31.5.13 maj_statistiques()	389
6.31.5.14 new_joueur()	389

6.31.5.15 stoper_mouvement_joueurs()	390
6.31.5.16 zone_en_dehors_hitbox()	391
6.31.6 Documentation des variables	391
6.31.6.1 save_path	391
6.32 Référence du fichier sorts.c	392
6.32.1 Description détaillée	392
6.32.2 Documentation des fonctions	393
6.32.2.1 action_sort()	393
6.32.2.2 ajouter_sort()	394
6.32.2.3 ajouter_sort_cb()	394
6.32.2.4 creer_sort_monstre()	395
6.32.2.5 detruire_sort()	396
6.32.2.6 detruire_sort_cb()	396
6.32.2.7 init_liste_base_sort()	397
6.32.2.8 orienter_sort_vers_joueur()	398
6.32.3 Documentation des variables	398
6.32.3.1 liste_base_sort	399
6.33 Référence du fichier sorts.h	399
6.33.1 Description détaillée	400
6.33.2 Documentation des macros	400
6.33.2.1 CONVERTIR_RADIANT_DEGREE	400
6.33.2.2 PATH_SPELL_BOSS	401
6.33.2.3 PATH_SPELL_WITCHER	401
6.33.3 Documentation du type de l'énumération	401
6.33.3.1 type_sort_t	401
6.33.4 Documentation des fonctions	401
6.33.4.1 action_sort()	401
6.33.4.2 ajouter_sort_cb()	402
6.33.4.3 creer_sort_monstre()	402
6.33.4.4 detruire_sort_cb()	403
6.33.4.5 init_liste_base_sort()	404
6.33.4.6 orienter_sort_vers_joueur()	405
6.33.5 Documentation des variables	405
6.33.5.1 liste_base_sort	406
6.34 Référence du fichier test_affichage.c	406
6.34.1 Description détaillée	406
6.34.2 Documentation des fonctions	407
6.34.2.1 main()	407
6.34.3 Documentation des variables	407
6.34.3.1 compteur	407
6.34.3.2 FENETRE_LARGEUR	407
6.34.3.3 FENETRE_LONGUEUR	407

6.34.3.4 test_map	407
6.35 Référence du fichier test_inventaire.c	407
6.35.1 Description détaillée	408
6.35.2 Documentation des fonctions	408
6.35.2.1 clean_suite()	408
6.35.2.2 generation_inventaire()	409
6.35.2.3 init_suite()	409
6.35.2.4 main()	410
6.35.2.5 modification_inventaire()	410
6.35.3 Documentation des variables	411
6.35.3.1 compteur	411
6.35.3.2 equipe	411
6.35.3.3 FENETRE_LARGEUR	411
6.35.3.4 FENETRE_LONGUEUR	412
6.35.3.5 objet_test	412
6.35.3.6 perso_principal	412
6.35.3.7 sac	412
6.35.3.8 test_map	412
6.36 Référence du fichier test_liste_objet.c	412
6.36.1 Description détaillée	413
6.36.2 Documentation des fonctions	413
6.36.2.1 main()	413
6.36.3 Documentation des variables	413
6.36.3.1 compteur	413
6.36.3.2 FENETRE_LARGEUR	414
6.36.3.3 FENETRE_LONGUEUR	414
6.36.3.4 test_map	414
6.37 Référence du fichier test_listes.c	414
6.37.1 Description détaillée	415
6.37.2 Documentation des fonctions	415
6.37.2.1 afficher_int()	415
6.37.2.2 main()	415
6.37.3 Documentation des variables	415
6.37.3.1 compteur	415
6.37.3.2 FENETRE_LARGEUR	415
6.37.3.3 FENETRE_LONGUEUR	416
6.37.3.4 test_map	416
6.38 Référence du fichier test_map.c	416
6.38.1 Description détaillée	417
6.38.2 Documentation des fonctions	417
6.38.2.1 main()	417
6.38.3 Documentation des variables	417

6.38.3.1 compteur	417
6.38.3.2 FENETRE_LARGEUR	417
6.38.3.3 FENETRE_LONGUEUR	418
6.38.3.4 test_map	418
6.39 Référence du fichier test_monstres.c	418
6.39.1 Description détaillée	418
6.39.2 Documentation des fonctions	419
6.39.2.1 main()	419
6.40 Référence du fichier test_personnage.c	419
6.40.1 Description détaillée	420
6.40.2 Documentation des fonctions	420
6.40.2.1 clean_suite()	420
6.40.2.2 creation_personnage()	421
6.40.2.3 init_suite()	421
6.40.2.4 main()	422
6.40.2.5 modification_personnage()	422
6.40.3 Documentation des variables	423
6.40.3.1 compteur	423
6.40.3.2 FENETRE_LARGEUR	423
6.40.3.3 FENETRE_LONGUEUR	423
6.40.3.4 perso_principal	423
6.40.3.5 test_map	423
6.41 Référence du fichier test SDL.c	424
6.41.1 Description détaillée	424
6.41.2 Documentation des fonctions	425
6.41.2.1 clean_suite()	425
6.41.2.2 init_suite()	425
6.41.2.3 main()	426
6.41.2.4 test SDL_close()	426
6.41.2.5 test SDL_init()	427
6.41.3 Documentation des variables	427
6.41.3.1 FENETRE_LARGEUR	427
6.41.3.2 FENETRE_LONGUEUR	427
6.41.3.3 fenetre_Principale	427
6.41.3.4 rendu_principal	427
Index	429

Chapitre 1

Liste des choses à faire

Membre **afficher_menu_accueil_manette** (**int *nb_joueur**)

Ajout du multijoueur.

Membre **afficher_menu_pause_manette** (**joueur_t *joueur**)

Permettre la sauvegarde et le chargement de celle-ci avec la manette.

Chapitre 2

Liste des éléments obsolètes

Membre `def_texture_taille (t_aff *a_modifier, const int longueur, const int largeur)`

L'utilisation de cette fonction n'a plus trop de sens étant donné que le moteur gère automatiquement la taille des textures.

Membre `deplacer_texture_centre (t_aff *texture, int x, int y)`

Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez `place_rect_center_from_point` à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

Membre `rect_centre (SDL_Rect *rectangle)`

Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez `place_rect_center_from_point` à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

Membre `rect_centre_x (SDL_Rect *rectangle)`

Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez `place_rect_center_from_point` à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

Membre `rect_centre_y (SDL_Rect *rectangle)`

Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez `place_rect_center_from_point` à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

Chapitre 3

Index des classes

3.1 Liste des classes

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

base_coffre_s	Structure contenant les propriétées du coffre importé	9
base_monstre_s	10
base_monstre_t	Structure contenant les propriétées du modèle d'un monstre	11
base_sort_s	11
boss_s	12
coffre_s	14
coffre_t	Structure contenant les propriétées du coffre	15
element	Définition du type element : un élément d'une liste	16
inventaire_t	17
inventaire_t	Structure inventaire divisor en deux parties: le sac et les objets équipés	18
joueur_t	Structure non manipulable hors des fonctions du personnage contenant les informations sur le joueur	18
list	Définition du type list : une liste générique	21
liste_base_coffres_s	Structure contenant un tableau avec tous les coffres possibles du jeu	22
liste_base_monstres_t	23
liste_base_monstres_t	Structure contenant un tableau avec tous les monstres différent(modèle de monstre) que l'on peut utiliser dans le jeu	24
lobjet_t	Définition de la structure lobjet	24
lobjet_t	Structure de liste d'objets	25
monstre_s	26
monstre_t	Structure contenant les propriétées du monstre en jeu	27
objet_t	Définition de la structure objet	28

objet_t	Structure objet	29
point	30
sort_s	30
sort_t	Structure contenant les propriétées d'un sort en jeu	31
statut_s	Structure contenant les éléments nécessaires au choix de l'affichage des sprites du personnage	31
struct	Structure contenant les propriétés d'origine des sorts (modèles de sort)	33
t_aff	Structure qui permet l'affichage d'une texture à l'écran de manière précise	34
t_l_aff	Structure contenant la liste des textures créées par le programme	36
t_map	Structure représentant une map	37
zone_tp	Structure représentant une zone de tp	40

Chapitre 4

Index des fichiers

4.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

affichage.c	Fonctions liées au module affichage	41
affichage.h	Fichier contenant les définitions et les fonctions liées au module affichage	96
boss.c	152
boss.h	156
code_erreur.h	Fichier contenant les codes d'erreur du programme	158
coffres.c	Fichier contenant toutes les fonctions concernant les coffres	161
coffres.h	Fonctions concernant les coffres	168
commun.h	Fichier contenant les headers nécessaires au programme principal	176
definition_commun.h	Contient toutes les définitions communes à tout les fichiers	178
event.c	Fonctions de gestion des événements du jeu	184
event.h	Définitions relatives aux événements	194
init_close.c	Contient toutes les fonctions pour initialiser le programme	199
interface.c	Fonctions liées à l'interface joueur	207
interface.h	Définition des fonctions relatives à l'interface joueur	209
inventaire.c	Fonctions liées au module inventaire	211
inventaire.h	Définition des fonctions relatives au module inventaire	220
liste_objet.c	Fonctions relatives aux structures contenant les objets	230
liste_objet.h	Définition relatives aux structures contenant les objets	238
listes.c	Fonctions relatives au type de structure liste générique	247

listes.h	Fonction relatives au type de structure liste générique	266
main.c	Programme principal du jeu	286
map.c	Fonctions de gestion de la map	290
map.h	Définitions des fonctions de gestion de la map	298
menus.c	Fonctions des menus du jeu	305
menus.h	Définitions relatives à la gestion des menus du jeu	315
monstres.c	Fichier contenant toutes les fonctions concernant les monstres	324
monstres.h	Fonctions concernant les monstres	341
objet.c	Fichier contenant toutes les fonctions concernant les objets	350
objet.h	Définitions concernant les objets	354
personnage.c	Fichier contenant toutes les fonctions concernant le personnage	358
personnage.h	Fichier contenant toutes les définitions concernant le personnage	375
sorts.c	Fichier contenant toutes les fonctions concernant les sorts	392
sorts.h	Fichier contenant toutes les définitions concernant les sorts	399
test_affichage.c	Programme de test du module affichage	406
test_inventaire.c	Programme de test du module inventaire	407
test_liste_objet.c	Programme de test du module liste_objet	412
test_listes.c	Programme de test du type de structure liste générique	414
test_map.c	Programme de test du module map	416
test_monstres.c	Programme de test du module monstres	418
test_personnage.c	Programme de test du module personnage	419
test SDL.c	Programme de test des fonctions principales de la SDL	424

Chapitre 5

Documentation des classes

5.1 Référence de la structure `base_coffre_s`

Structure contenant les propriétés du coffre importé

Attributs publics

- char `fichier_image` [50]
- char `nom_coffre` [20]
- `SDL_Rect hitbox`

5.1.1 Description détaillée

Auteur

Max Descomps

5.1.2 Documentation des données membres

5.1.2.1 `fichier_image`

```
char base_coffre_s::fichier_image[50]
```

Nom du fichier image

5.1.2.2 `hitbox`

```
SDL_Rect base_coffre_s::hitbox
```

Hitbox du modèle de coffre

5.1.2.3 nom_coffre

```
char base_coffre_s::nom_coffre[20]
```

Nom du modèle de coffre

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

— [coffres.h](#)

5.2 Référence de la structure base_monstre_s

Attributs publics

- char [fichier_image](#) [50]
- char [nom_monstre](#) [25]
- int [pdv](#)
- int [attaque](#)
- float [vitesse](#)
- int [gainXp](#)
- [SDL_Rect](#) [hitbox](#)

5.2.1 Documentation des données membres

5.2.1.1 attaque

```
int base_monstre_s::attaque
```

attaque

5.2.1.2 fichier_image

```
char base_monstre_s::fichier_image[50]
```

chemin d'accès du sprite du monstre

5.2.1.3 gainXp

```
int base_monstre_s::gainXp
```

gain d'xp pour le joueur

5.2.1.4 hitbox

```
SDL_Rect base_monstre_s::hitbox
```

hitbox du monstre

5.2.1.5 `nom_monstre`

```
char base_monstre_s::nom_monstre[25]
```

nom du monstre

5.2.1.6 `pdv`

```
int base_monstre_s::pdv
```

points de vie

5.2.1.7 `vitesse`

```
float base_monstre_s::vitesse
```

vitesse de déplacement

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

— [monstres.h](#)

5.3 Référence de la structure `base_monstre_t`

Structure contenant les propriétés du modèle d'un monstre.

5.3.1 Description détaillée

Auteur

Bruneau Antoine

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

— [monstres.h](#)

5.4 Référence de la structure `base_sort_s`

Attributs publics

- [type_sort_t type](#)
- [int degat](#)
- [SDL_Rect collision](#)

5.4.1 Documentation des données membres

5.4.1.1 collision

`SDL_Rect base_sort_s::collision`

hitbox du sort

5.4.1.2 degat

`int base_sort_s::degat`

degat du sort

5.4.1.3 type

`type_sort_t base_sort_s::type`

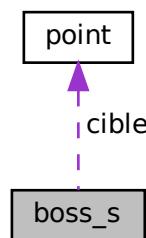
type de sort

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

— `sorts.h`

5.5 Référence de la structure boss_s

Graphe de collaboration de `boss_s`:



Attributs publics

- `type_boss_1_t type`
- `int pdv`
- `int attaque`
- `int xp`
- `point cible`
- `action_boss_1_t action`
- `int duree`
- `SDL_Rect collision`
- `t_aff * texture`
- `t_aff * texture_temp [2]`

5.5.1 Documentation des données membres

5.5.1.1 action

```
action_boss_1_t boss_s::action
```

5.5.1.2 attaque

```
int boss_s::attaque
```

5.5.1.3 cible

```
point boss_s::cible
```

5.5.1.4 collision

```
SDL_Rect boss_s::collision
```

5.5.1.5 duree

```
int boss_s::duree
```

5.5.1.6 pdv

```
int boss_s::pdv
```

5.5.1.7 texture

```
t_aff* boss_s::texture
```

5.5.1.8 texture_temp

```
t_aff* boss_s::texture_temp[2]
```

5.5.1.9 type

```
type_boss_1_t boss_s::type
```

5.5.1.10 xp

```
int boss_s::xp
```

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

— [boss.h](#)

5.6 Référence de la structure coffre_s

Attributs publics

- int [id_cle](#)
- int [id_loot](#)
- [type_coffre_t](#) [type](#)
- [t_direction_1](#) [orientation](#)
- [etat_coffre_t](#) [etat](#)
- [SDL_Rect](#) [collision](#)
- [t_aff](#)* [texture](#)

5.6.1 Documentation des données membres

5.6.1.1 collision

```
SDL_Rect coffre_s::collision
```

Coordonnées

5.6.1.2 etat

```
etat_coffre_t coffre_s::etat
```

5.6.1.3 id_cle

```
int coffre_s::id_cle
```

Identificateur de l'objet de quête nécessaire pour ouvrir le coffre sinon 0

5.6.1.4 id_loot

```
int coffre_s::id_loot
```

Identificateur de l'objet obtenu en ouvrant le coffre sinon 0

5.6.1.5 orientation

```
t_direction_1 coffre_s::orientation
```

5.6.1.6 texture

```
t_aff* coffre_s::texture
```

Texture

5.6.1.7 type

```
type_coffre_t coffre_s::type
```

Type de coffre

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

— [coffres.h](#)

5.7 Référence de la structure coffre_t

Structure contenant les propriétées du coffre.

5.7.1 Description détaillée

Auteur

Max Descomps

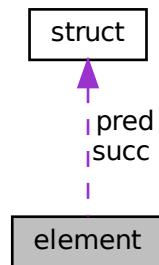
La documentation de cette structure a été générée à partir du fichier suivant :

— [coffres.h](#)

5.8 Référence de la structure element

Définition du type element : un élément d'une liste.

Graphe de collaboration de element:



Attributs publics

- void * valeur
- struct element * pred
- struct element * succ

5.8.1 Description détaillée

Auteur

Ange Despert

5.8.2 Documentation des données membres

5.8.2.1 pred

```
struct element* element::pred
```

L'élément précédent

5.8.2.2 succ

```
struct element* element::succ
```

L'élément suivant

5.8.2.3 valeur

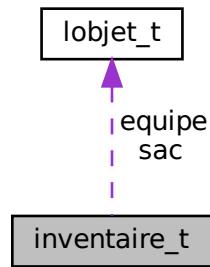
```
void* element::valeur
```

La valeur de l'élément, un pointeur générique

La documentation de cette structure a été générée à partir du fichier suivant :
— [listes.h](#)

5.9 Référence de la structure inventaire_t

Graphe de collaboration de inventaire_t:



Attributs publics

- `lobjet_t * equipe`
- `lobjet_t * sac`

5.9.1 Documentation des données membres

5.9.1.1 equipe

```
lobjet_t* inventaire_t::equipe
```

Objets équipés

5.9.1.2 sac

```
lobjet_t* inventaire_t::sac
```

Objets dans le sac

La documentation de cette structure a été générée à partir du fichier suivant :
— [inventaire.h](#)

5.10 Référence de la structure inventaire_t

Structure inventaire diviser en deux parties: le sac et les objets équipés.

5.10.1 Description détaillée

Auteur

Max Descomps

Rafael Doneau

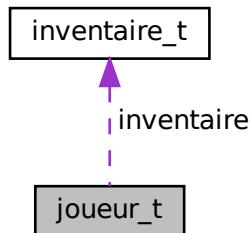
La documentation de cette structure a été générée à partir du fichier suivant :

— [inventaire.h](#)

5.11 Référence de la structure joueur_t

Structure non manipulable hors des fonctions du personnage contenant les informations sur le joueur.

Graphe de collaboration de joueur_t:



Attributs publics

- char * **nom_pers**
- short int **niveau**
- int **xp**
- byte * **trigger**
- int **maxPdv**
- int **pdv**
- int **attaque**
- int **defense**
- int **vitesse**
- int **attaque_actif**
- int **defense_actif**
- int **vitesse_actif**
- statut_t * **statut**
- t_l_aff * **textures_joueur**
- inventaire_t * **inventaire**

5.11.1 Description détaillée

Définition de la structure joueur.

Auteurs

Despert Ange
Descomps Max

Cette définition est la pour éviter une inclusion mutuelle des fichiers [inventaire.h](#) et [personnage.h](#).

5.11.2 Documentation des données membres

5.11.2.1 attaque

```
int joueur_t::attaque
```

attaque de base du joueur

5.11.2.2 attaque_actif

```
int joueur_t::attaque_actif
```

attaque du joueur avec bonus d'équipement

5.11.2.3 defense

```
int joueur_t::defense
```

defense de base du joueur

5.11.2.4 defense_actif

```
int joueur_t::defense_actif
```

defense du joueur avec bonus d'équipement

5.11.2.5 inventaire

```
inventaire_t* joueur_t::inventaire
```

Inventaire du joueur

5.11.2.6 maxPdv

```
int joueur_t::maxPdv
```

Le nombre de Pv max du joueur

5.11.2.7 niveau

```
short int joueur_t::niveau
```

Le niveau du joueur

5.11.2.8 nom_pers

```
char* joueur_t::nom_pers
```

Le nom du personnage

5.11.2.9 pdv

```
int joueur_t::pdv
```

Les points de vie actuels du joueur

5.11.2.10 statut

```
statut_t* joueur_t::statut
```

statut du joueur

5.11.2.11 textures_joueur

```
t_l_aff* joueur_t::textures_joueur
```

Tableau contenant toutes les textures du joueur

5.11.2.12 trigger

```
byte* joueur_t::trigger
```

Une variable contenant des triggers logiques concernant le personnage

5.11.2.13 vitesse

```
int joueur_t::vitesse
```

vitesse de déplacement de base du joueur

5.11.2.14 vitesse_actif

int joueur_t::vitesse_actif

vitesse du joueur avec bonus d'équipement

5.11.2.15 xp

int joueur_t::xp

Le nombre de points d'expérience que possède le joueur

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

— [personnage.h](#)

5.12 Référence de la structure list

Définition du type list : une liste générique.

Attributs publics

— t_element * [flag](#)
— t_element * [ec](#)
— unsigned int [nb_elem](#)
— void *(* [ajout](#))(void *)
— void(* [del](#))(void *)
— void(* [aff](#))(void *)

5.12.1 Description détaillée

Auteurs

Ange Despert

Max Descomps

5.12.2 Documentation des données membres

5.12.2.1 aff

void(* list::aff) (void *)

Une fonction pour afficher un élément de la liste

5.12.2.2 ajout

```
void*(* list::ajout) (void *)
```

Une fonction pour ajouter un élément dans la liste

5.12.2.3 del

```
void(* list::del) (void *)
```

Une fonction pour retirer un élément de la liste

5.12.2.4 ec

```
t_element* list::ec
```

L'élément courant

5.12.2.5 flag

```
t_element* list::flag
```

Le drapeau (1er élément)

5.12.2.6 nb_elem

```
unsigned int list::nb_elem
```

Le nombre d'élément dans la liste

La documentation de cette structure a été générée à partir du fichier suivant :
— [listes.h](#)

5.13 Référence de la structure liste_base_coffres_s

Structure contenant un tableau avec tous les coffres possibles du jeu.

Attributs publics

- int [nb_coffre](#)
- base_coffre_t * [tab](#)

5.13.1 Description détaillée

Auteur

Max Descomps

5.13.2 Documentation des données membres

5.13.2.1 nb_coffre

```
int liste_base_coffres_s::nb_coffre
```

Nombre de modèles de coffres du jeu

5.13.2.2 tab

```
base_coffre_t* liste_base_coffres_s::tab
```

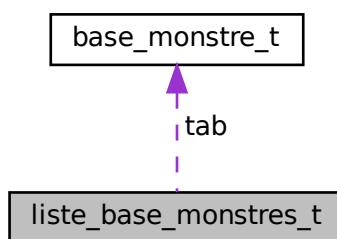
Modèles des coffres du jeu

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

— [coffres.h](#)

5.14 Référence de la structure liste_base_monstres_t

Graphe de collaboration de liste_base_monstres_t:



Attributs publics

- int [nb_monstre](#)
- [base_monstre_t * tab](#)

5.14.1 Documentation des données membres

5.14.1.1 nb_monstre

`int liste_base_monstres_t::nb_monstre`

nombre de modèles de monstre

5.14.1.2 tab

`base_monstre_t* liste_base_monstres_t::tab`

tableau de modèles de monstre

La documentation de cette structure a été générée à partir du fichier suivant :
— [monstres.h](#)

5.15 Référence de la structure liste_base_monstres_t

Structure contenant un tableau avec tous les monstres différent(modèle de monstre) que l'on peut utiliser dans le jeu.

5.15.1 Description détaillée

Auteur

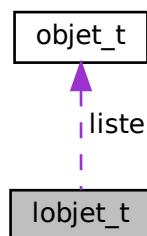
Bruneau Antoine

La documentation de cette structure a été générée à partir du fichier suivant :
— [monstres.h](#)

5.16 Référence de la structure lobjet_t

Définition de la structure lobjet.

Graphe de collaboration de lobjet_t:



Attributs publics

- int [nb](#)
- [objet_t](#) ** [liste](#)

5.16.1 Description détaillée

Cette définition est là pour éviter une inclusion mutuelle des fichiers [inventaire.h](#) et [liste_objet.h](#).

5.16.2 Documentation des données membres

5.16.2.1 liste

[objet_t](#) ** [lobjet_t](#)::[liste](#)

Objets stockés

5.16.2.2 nb

int [lobjet_t](#)::[nb](#)

Nombre d'objets stockés

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

- [liste_objet.h](#)

5.17 Référence de la structure lobjet_t

Structure de liste d'objets.

5.17.1 Description détaillée

Auteur

Descomps Max

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

- [liste_objet.h](#)

5.18 Référence de la structure monstre_s

Attributs publics

```
— type_monstre_t type
— int pdv
— int attaque
— float vitesse
— int gainXp
— t_direction_1 orientation
— action_monstre_t action
— int duree
— SDL_Rect collision
— t_aff * texture
```

5.18.1 Documentation des données membres

5.18.1.1 action

```
action_monstre_t monstre_s::action
```

5.18.1.2 attaque

```
int monstre_s::attaque  
attaque
```

5.18.1.3 collision

```
SDL_Rect monstre_s::collision  
zone de collision du monstre (hitbox) + ses coordonnées
```

5.18.1.4 duree

```
int monstre_s::duree
```

5.18.1.5 gainXp

```
int monstre_s::gainXp  
gain d'xp pour le joueur
```

5.18.1.6 orientation

`t_direction_1` monstre_s::orientation

5.18.1.7 pdv

`int` monstre_s::pdv

points de vie

5.18.1.8 texture

`t_aff*` monstre_s::texture

texture du monstre

5.18.1.9 type

`type_monstre_t` monstre_s::type

type de monstre

5.18.1.10 vitesse

`float` monstre_s::vitesse

vitesse de déplacement

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

— [monstres.h](#)

5.19 Référence de la structure monstre_t

Structure contenant les propriétées du monstre en jeu.

5.19.1 Description détaillée

Auteur

Bruneau Antoine

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

— [monstres.h](#)

5.20 Référence de la structure objet_t

Définition de la structure objet.

Attributs publics

```
— int id
— t_aff * texture
— char * texture_src
— t_item type
— char * nom
— short int niveau
— int attaque
— int defense
— int vitesse
```

5.20.1 Description détaillée

Cette définition est là pour éviter une inclusion mutuelle des fichiers [inventaire.h](#) et [objet.h](#).

5.20.2 Documentation des données membres

5.20.2.1 attaque

```
int objet_t::attaque
```

modificateur d'attaque de l'objet

5.20.2.2 defense

```
int objet_t::defense
```

modificateur de defense de l'objet

5.20.2.3 id

```
int objet_t::id
```

Identificateur de l'objet

5.20.2.4 niveau

```
short int objet_t::niveau
```

Le niveau nécessaire pour équiper l'objet

5.20.2.5 nom

```
char* objet_t::nom
```

Le nom de l'objet

5.20.2.6 texture

```
t_aff* objet_t::texture
```

Image de l'objet

5.20.2.7 texture_src

```
char* objet_t::texture_src
```

Chemin de l'image de l'objet

5.20.2.8 type

```
t_item objet_t::type
```

Le type d'objet permet de contrôler sa bonne utilisation

5.20.2.9 vitesse

```
int objet_t::vitesse
```

modificateur de vitesse de l'objet

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

— [objet.h](#)

5.21 Référence de la structure objet_t

Structure objet.

5.21.1 Description détaillée

Auteur

Max Descomps

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

— [objet.h](#)

5.22 Référence de la structure point

Attributs publics

- int `x`
- int `y`

5.22.1 Documentation des données membres

5.22.1.1 `x`

```
int point::x
```

5.22.1.2 `y`

```
int point::y
```

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

- [definition_commun.h](#)

5.23 Référence de la structure sort_s

Attributs publics

- `type_sort_t type`
- int `degat`
- `SDL_Rect collision`
- `t_aff * texture`

5.23.1 Documentation des données membres

5.23.1.1 `collision`

```
SDL_Rect sort_s::collision
```

hitbox du sort

5.23.1.2 degat

int sort_s::degat

degat du sort

5.23.1.3 texture

t_aff* sort_s::texture

texture du sort

5.23.1.4 type

type_sort_t sort_s::type

type de sort

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

— [sorts.h](#)

5.24 Référence de la structure sort_t

Structure contenant les propriétées d'un sort en jeu.

5.24.1 Description détaillée

Auteur

Bruneau Antoine

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

— [sorts.h](#)

5.25 Référence de la structure statut_s

Structure contenant les éléments nécessaires au choix de l'affichage des sprites du personnage.

Attributs publics

- _Bool en_mouvement
- t_direction_1 orient_dep
- t_direction_2 orient_att
- _Bool bouclier_equipe
- int duree
- int duree_anim
- action_t action
- action_t animation
- SDL_Rect zone_colision
- SDL_Rect vrai_zone_collision
- t_aff * texture_prec

5.25.1 Description détaillée

Auteurs

Bruneau Antoine
Ange Despert

5.25.2 Documentation des données membres

5.25.2.1 action

`action_t statut_s::action`

l'action du personnage

5.25.2.2 animation

`action_t statut_s::animation`

Animation sur le personnage

5.25.2.3 bouclier_equipe

`_Bool statut_s::bouclier_equipe`

<Définition du type booléen personnage à un bouclier d'équipé

5.25.2.4 duree

`int statut_s::duree`

durée de l'action à réaliser

5.25.2.5 duree_anim

`int statut_s::duree_anim`

durée d'une animation éventuelle sur le joueur

5.25.2.6 en_mouvement

`_Bool statut_s::en_mouvement`

<Définition du type booléen personnage en mouvement

5.25.2.7 orient_att

```
t_direction_2 statut_s::orient_att
```

orientaiton attaque du personnage

5.25.2.8 orient_dep

```
t_direction_1 statut_s::orient_dep
```

orientation deplacement du personnage

5.25.2.9 texture_prec

```
t_aff* statut_s::texture_prec
```

la texture precedente du personnage

5.25.2.10 vrai_zone_collision

```
SDL_Rect statut_s::vrai_zone_collision
```

La vrai zone de collision du J1 sur la carte

5.25.2.11 zone_colision

```
SDL_Rect statut_s::zone_colision
```

zone de colision du personnage

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

— [personnage.h](#)

5.26 Référence de la structure struct

Structure contenant les propriétées d'origine des sorts (modèles de sort)

5.26.1 Description détaillée

Auteur

Bruneau Antoine

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

— [sorts.h](#)

5.27 Référence de la structure t_aff

Structure qui permet l'affichage d'une texture à l'écran de manière précise.

Attributs publics

- `SDL_Texture * texture`
- `SDL_Rect * frame_anim`
- `SDL_Rect * aff_fenetre`
- `int width`
- `int height`
- `float multipli_taille`
- `unsigned int duree_frame_anim`
- `unsigned int compteur_frame_anim`

5.27.1 Description détaillée

Définition de la structure de texture.

5.27.2 Le but de la structure

Cette structure à pour but de permettre l'affichage d'une texture à l'écran de manière précise. Ainsi, on peut facilement placer une texture où l'on veut sans savoir utiliser la SDL.

5.27.3 Éléments de la structure

- `texture` : la `SDL_Texture` que l'on veut afficher à l'écran. Elle est créée par la fonction `creer_texture`. à partir d'une surface chargée à partir d'un fichier.
- `frame_anim` : Un rectangle permettant de n'afficher qu'une partie bien précise de la texture. C'est à dire, un rectangle qui contient la frame que l'on veut afficher. Cela permet donc d'animer la texture.
- `aff_fenetre` : Un rectangle qui permet de savoir où placer la texture à l'écran et de connaître sa taille. Sa taille est définie à l'aide d'un multiplicateur où bien elle peut être choisie grâce à l'appel d'une fonction.
- `width` : La largeur de la texture. Cette valeur est obtenue grâce à une requête SDL et renseignée dans la structure par la fonction `creer_texture`.
- `height` : La hauteur de la texture. Cette valeur est obtenue grâce à une requête SDL et renseignée dans la structure par la fonction `creer_texture`.
- `multipli_taille` : Un multiplicateur qui permet de modifier la taille de la texture. Il s'agit de celui que l'on renseigne dans la fonction `creer_texture`.
- `duree_frame_anim` : entier qui permet de savoir combien de temps une frame dure. C'est à dire, combien de temps une frame dure sur l'écran.

Cette définition est la pour éviter une inclusion mutuelle des fichiers `coffres.h` et `affichage.h` .

Cette définition est la pour éviter une inclusion mutuelle des fichiers `objet.h` et `affichage.h` .

5.27.4 Documentation des données membres

5.27.4.1 aff_fenetre

```
SDL_Rect* t_aff::aff_fenetre
```

Désigne l'emplacement et la taille de l'objet à l'écran

5.27.4.2 compteur_frame_anim

```
unsigned int t_aff::compteur_frame_anim
```

Stade de l'animation

5.27.4.3 duree_frame_anim

```
unsigned int t_aff::duree_frame_anim
```

Durée d'une frame

5.27.4.4 frame_anim

```
SDL_Rect* t_aff::frame_anim
```

Désigne la zone de la texture à afficher

5.27.4.5 height

```
int t_aff::height
```

Hauteur de la texture

5.27.4.6 multipli_taille

```
float t_aff::multipli_taille
```

Sauvegarde du multiplicateur de taille de la texture

5.27.4.7 texture

```
SDL_Texture* t_aff::texture
```

Texture utilisée

5.27.4.8 width

```
int t_aff::width
```

Longueur de la texture

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

- [affichage.h](#)

5.28 Référence de la structure t_l_aff

Structure contenant la liste des textures créées par le programme.

Attributs publics

- `t_aff ** liste`
- `unsigned int nb_valeurs`

5.28.1 Description détaillée

5.28.2 Documentation des données membres

5.28.2.1 liste

```
t_aff** t_l_aff::liste
```

5.28.2.2 nb_valeurs

```
unsigned int t_l_aff::nb_valeurs
```

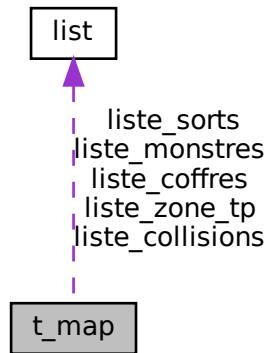
La documentation de cette structure a été générée à partir du fichier suivant :

- [affichage.h](#)

5.29 Référence de la structure t_map

Structure représentant une map.

Graphe de collaboration de t_map:



Attributs publics

```

— unsigned int id_map
— t_aff * text_map
— t_aff * text_sol
— t_aff * texture_superposition
— unsigned int width
— unsigned int height
— unsigned int taille_case
— unsigned int cases_x
— unsigned int cases_y
— list * liste_monstres
— list * liste_sorts
— list * liste_collisions
— list * liste_coffres
— list * liste_zone_tp
  
```

5.29.1 Description détaillée

Cette structure doit permettre de stocker toutes les informations liées à la map.

Pour cela, On va stocké sous forme de listes toutes les [zones de collisions](#), [les sorts](#), [les monstres](#), [les coffres](#) et enfin [les zones de tp](#).

On retient également des informations importantes comme des textures : [la texture de fond](#), [une texture à supposer](#) et enfin [la map avec toutes les entités](#).

Auteurs

Ange Despert

Antoine Bruneau

Max Descomps

5.29.2 Documentation des données membres

5.29.2.1 cases_x

```
unsigned int t_map::cases_x
```

Le nombre de cases affichées en x

5.29.2.2 cases_y

```
unsigned int t_map::cases_y
```

Le nombre de cases affichées en y

5.29.2.3 height

```
unsigned int t_map::height
```

la hauteur de la map

5.29.2.4 id_map

```
unsigned int t_map::id_map
```

L'identificateur de la map

5.29.2.5 liste_coffres

```
list* t_map::liste_coffres
```

La liste de tous les coffres

5.29.2.6 liste_collisions

```
list* t_map::liste_collisions
```

La liste de toutes les les collisions

5.29.2.7 liste_monstres

```
list* t_map::liste_monstres
```

La liste des monstres de la map

5.29.2.8 liste_sorts

```
list* t_map::liste_sorts
```

La liste des sorts de la map

5.29.2.9 liste_zone_tp

```
list* t_map::liste_zone_tp
```

La liste des points de téléportation

5.29.2.10 taille_case

```
unsigned int t_map::taille_case
```

La taille d'une case

5.29.2.11 text_map

```
t_aff* t_map::text_map
```

La texture de la map

5.29.2.12 text_sol

```
t_aff* t_map::text_sol
```

La texture du sol

5.29.2.13 texture_superposition

```
t_aff* t_map::texture_superposition
```

La texture à superposer devant le personnage

5.29.2.14 width

```
unsigned int t_map::width
```

La longueur de la map

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

— [map.h](#)

5.30 Référence de la structure zone_tp

Structure représentant une zone de tp.

Attributs publics

- `SDL_Rect zone`
- `unsigned int id_map`
- `SDL_Point dest`

5.30.1 Description détaillée

Il s'agit ici d'identifier une zone de collision ou le joueur sera téléporté Pour cela, on doit également savoir dans quelle map on va atterrir mais également à quel endroit.

Auteur

Ange Despert

5.30.2 Documentation des données membres

5.30.2.1 dest

`SDL_Point zone_tp::dest`

Les coordonnées du point d'apparition sur la map

5.30.2.2 id_map

`unsigned int zone_tp::id_map`

l'id de la map de destination

5.30.2.3 zone

`SDL_Rect zone_tp::zone`

Rectangle représentant la zone de tp

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

- [map.h](#)

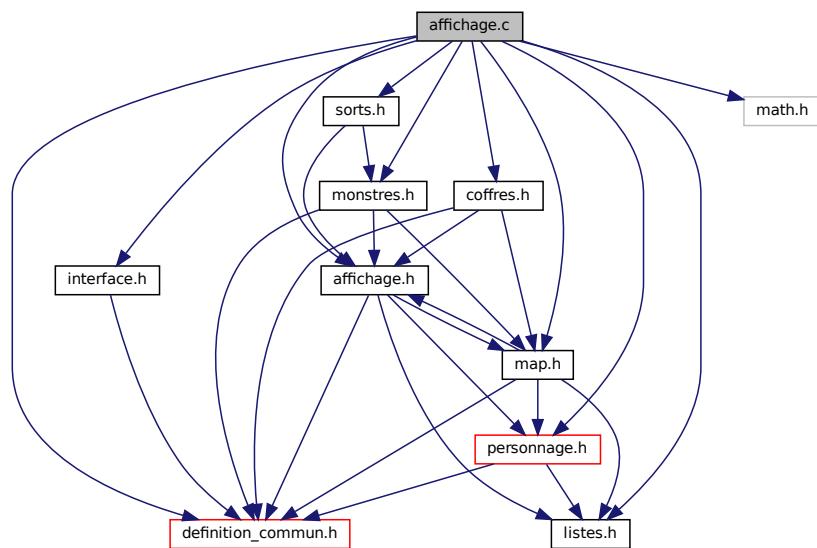
Chapitre 6

Documentation des fichiers

6.1 Référence du fichier affichage.c

Fonctions liées au module affichage.

Graphe des dépendances par inclusion de affichage.c:



Fonctions

- void * ajout_text_liste (void *t)
Fonction qui ajoute une texture dans une liste générique.
- void detruire_texture (t_aff **texture)
Fonction qui détruit une structure d'affichage de texture passée en paramètre.
- void detruire_liste_textures (t_l_aff **l_texture)
Fonction qui détruit une liste de textures.
- bool rect_correct_texture (const SDL_Rect *const to_verify, const int width, const int height)
Fonction qui détermine si la structure SDL_Rect ne dépasse pas la taille de la texture.
- err_t update_frame_texture (t_aff *texture, const int x, const int y)

- void **next_frame_x** (*t_aff *texture*)

Fonction qui positionne la texture au sprite d'après sur l'axe des x.
- void **next_frame_y** (*t_aff *texture*)

Fonction qui positionne la texture au sprite d'après sur l'axe des y.
- **err_t next_frame_indice** (*t_aff *texture, const unsigned int x, const unsigned int y*)

Fonction qui positionne la texture au n-ème sprite sur l'axe des x.
- **err_t next_frame_x_indice** (*t_aff *texture, const unsigned int indice*)

Fonction qui positionne la texture au n-ème sprite sur l'axe des x.
- **err_t next_frame_y_indice** (*t_aff *texture, const unsigned int indice*)

Fonction qui positionne la texture au n-ème sprite sur l'axe des y.
- void **def_texture_taille** (*t_aff *a_modifier, const int longueur, const int largeur*)

Fonction qui permet de définir exactement la taille de la texture.
- void **info_texture** (*t_aff *texture*)

Affiche des informations en console sur une texture.
- **t_aff * creer_texture** (*const char *nom_fichier, const int taille_t_x, const int taille_t_y, const int x, const int y, const float multiplicateur_taille*)

Fonction qui renvoie, charge une texture et la prépare à se faire afficher.
- **err_t afficher_texture** (*t_aff *texture, SDL_Renderer *rendu*)

Affiche la texture donnée en paramètre à l'écran.
- **t_l_aff * init_textures_joueur** (*joueur_t *j, int num_j*)

Fonction qui créer et renvoie une liste de textures pour le personnage (joueur)
- **t_aff * init_texture_joueur** (*t_l_aff *textures_joueur, joueur_t *joueur*)

Fonction qui renvoie la texture de départ du personnage (joueur)
- void **appliquer_coord_rect** (*const SDL_Rect *const a_copier, t_l_aff *textures*)
- **t_aff * next_frame_joueur** (*joueur_t *j*)
- void **afficher_monstres** (*list *liste_monstre, joueur_t *joueur*)

Fonction qui affiche les monstres.
- void **afficher_coffres** (*list *liste_coffre*)

Fonction qui affiche la texture des coffres.
- void **afficher_sorts** (*list *liste_sorts, joueur_t *joueur*)

Fonction qui affiche les sorts.
- **err_t afficher_buffer** (*list *buffer, SDL_Renderer *rendu*)

Fonction qui affiche les textures contenues dans la liste en paramètre.
- **point get_screen_center** ()
- **bool point_in_rect** (*SDL_Rect r, point p*)
- void **rect_centre_x** (*SDL_Rect *rectangle*)

Fonction qui permet de placer un rectangle au centre de l'écran sur l'axe des x.
- void **rect_centre_y** (*SDL_Rect *rectangle*)

Fonction qui permet de placer un rectangle au centre de l'écran sur l'axe des y.
- void **rect_centre** (*SDL_Rect *rectangle*)

Fonction qui permet de placer un rectangle au centre de l'écran.
- void **rect_centre_rect_x** (*SDL_Rect *rectangle, SDL_Rect *rectangle_centre*)

Fonction qui permet de placer un rectangle au centre d'un autre sur l'axe des x.
- void **rect_centre_rect_y** (*SDL_Rect *rectangle, SDL_Rect *rectangle_centre*)

Fonction qui permet de placer un rectangle au centre d'un autre sur l'axe des y.
- void **rect_centre_rect** (*SDL_Rect *rectangle, SDL_Rect *rectangle_centre*)

Fonction qui permet de placer un rectangle au centre d'un autre.
- void **deplacer_texture_centre** (*t_aff *texture, int x, int y*)

Déplace la texture pour que son centre soit au centre de l'écran.
- void **deplacer_rect_origine** (*SDL_Rect *r, int x, int y*)

Déplace un rectangle depuis l'origine de l'écran.
- void **deplacer_texture_origine** (*t_aff *texture, int x, int y*)

Déplace l'origine de la texture aux coordonnées données.
- void **deplacer_rect_haut_droit** (*SDL_Rect *r, int x, int y*)

Place un rectangle en haut à droite de l'écran puis le replace à partir de cette origine.
- void **deplacer_texture_haut_droit** (*t_aff *texture, int x, int y*)

La texture est déplacée vers la droite et vers le haut.
- void **deplacer_texture_bas_gauche** (*t_aff *texture, int x, int y*)

La texture est déplacée vers le coin inférieur gauche de l'écran.
- void **deplacer_texture_bas_droit** (*t_aff *texture, int x, int y*)

La texture est déplacée vers le coin inférieur droit de la fenêtre.
- void **modif_affichage_rect** (*t_aff *texture, SDL_Rect r*)

Modifie le rectangle qui définit la zone de l'écran qui sera utilisée pour le rendu de la texture.
- **bool deplacement_x_pers** (*t_map *m, joueur_t **joueurs, unsigned short int nb_joueurs, int x, lobjet_t *objets*)

Permet de déplacer le personnage de x unités horizontales sur la map.
- **bool deplacement_y_pers** (*t_map *m, joueur_t **joueurs, unsigned short int nb_joueurs, int y, lobjet_t *objets*)

Permet de déplacer le personnage principal de y unités verticales sur la map.
- **bool deplacement_x_joueur_secondaire** (*t_map *m, joueur_t *joueur, int x, SDL_Rect *r, lobjet_t *objets*)

- Permet de déplacer un joueur secondaire de x unités horizontales sur la map.
- **bool deplacement_y_joueur_secondaire** (*t_map *m*, *joueur_t *joueur*, *int y*, *SDL_Rect *r*, *lobjet_t *objets*)
 - Permet de déplacer un joueur secondaire de y unités verticales sur la map.
- **void text_copier_position** (*t_aff *a_modifier*, *const t_aff *const original*)
 - Fonction qui permet de placer 2 textures aux mêmes endroit à l'écran.
- **bool rects_egal_x** (*const SDL_Rect *const r1*, *SDL_Rect const *const r2*)
 - Fonction qui permet de savoir si deux rectangles sont égaux sur l'axe x.
- **bool rects_egal_y** (*const SDL_Rect *const r1*, *SDL_Rect const *const r2*)
 - Fonction qui permet de savoir si deux rectangles sont égaux sur l'axe y.
- **SDL_Color color** (*Uint8 r*, *Uint8 g*, *Uint8 b*, *Uint8 a*)
 - Fonction qui permet de choisir une couleur SDL.
- **void placer_texture** (*t_aff *texture*, *int x*, *int y*)
 - Place une texture sur l'écran.
- **void rect_ecran_to_rect_map** (*SDL_Rect *ecran*, *SDL_Rect *r_map*, *int x*, *int y*)
- **bool deplacement_x_entite** (*t_map *m*, *t_aff *texture*, *int x*, *SDL_Rect *r*)
 - Fonction qui permet le déplacement d'une entité
- **bool deplacement_y_entite** (*t_map *m*, *t_aff *texture*, *int y*, *SDL_Rect *r*)
 - Fonction qui permet le déplacement d'une entité
- **void int_animations** ()
 - Initialise les textures des animations.
- **t_aff * next_frame_animation** (*joueur_t *joueur*)
 - Fait évoluer les animations en jeu.
- **void lister_animations** (*joueur_t **joueurs*, *list *animations*)
 - Liste les animations en jeu.
- **void afficher_animations** (*list *animations*)
 - Fonction qui affiche la texture des animations.
- **int current_frame_x** (*t_aff *texture*)
 - Fonction qui donne l'indice sur l'axe des abscisses actuelle de la texture.
- **int current_frame_y** (*t_aff *texture*)
 - Fonction qui donne l'indice sur l'axe des ordonnées actuelle de la texture.
- **void detruire_collision_dans_liste** (*list *liste_collisions*, *SDL_Rect *collision*)
 - Fonction qui détruit une collision donnée dans une liste de collisions.
- **SDL_Point get_rect_center** (*const SDL_Rect *const r*)
 - Renvoie les coordonnées du centre du rectangle.
- **SDL_Point get_rect_center_coord** (*const SDL_Rect *const r*)
 - Renvoie les coordonnées du centre du rectangle.
- **void place_rect_center_from_point** (*SDL_Rect *r*, *SDL_Point p*)
 - Fonction qui permet de placer le centre du rectangle donné en paramètre à un point précis.

Variables

- **list * listeDeTextures**
- **list * liste_animations** = NULL
- **t_aff * heal** = NULL
- **t_aff * bloquer** = NULL
- **t_aff * fenetre_finale** = NULL
- **long int compteur**
- **unsigned int FENETRE_LONGUEUR**
- **unsigned int FENETRE_LARGEUR**
- **SDL_Rect tx**
- **SDL_Rect ty**
- **float multiplicateur_x**
- **float multiplicateur_y**

6.1.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
 Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.1.2 Documentation des fonctions**6.1.2.1 afficher_animations()**

```
void afficher_animations (
    list * animations )
```

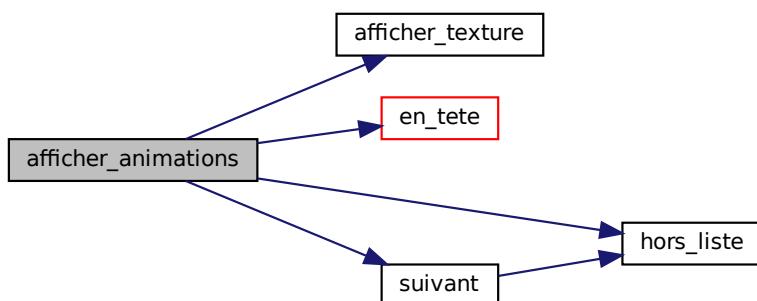
Auteur

Max Descomps

Paramètres

<i>animations</i>	Une liste d'animations
-------------------	------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.2 afficher_buffer()

```
err_t afficher_buffer (
    list * buffer,
    SDI_Renderer * rendu )
```

Auteur

Ange Despert

Les premiers éléments seront en arrière plan et les derniers seront en 1er plan.

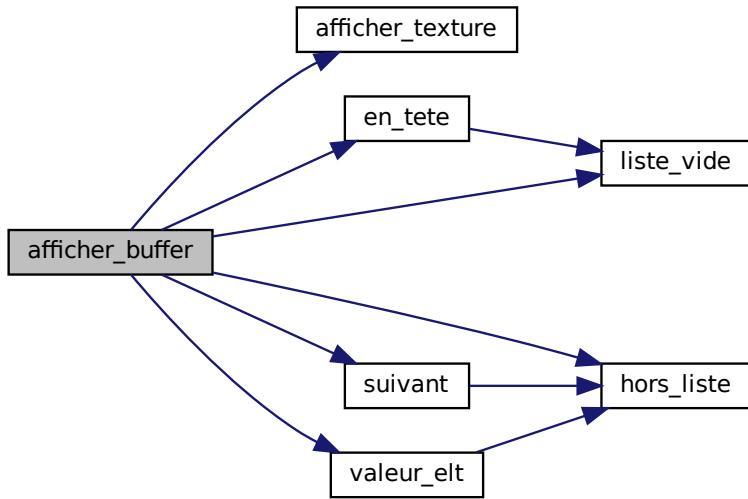
Paramètres

<i>buffer</i>	La liste de textures que l'on veut afficher
<i>rendu</i>	Le rendu sur lequel on veut afficher ces textures

Renvoie

Une valeur différente à 0 lors d'une erreur

Voici le graphe d'appel pour cette fonction :



6.1.2.3 afficher_coffres()

```
void afficher_coffres (
    list * liste_coffre )
```

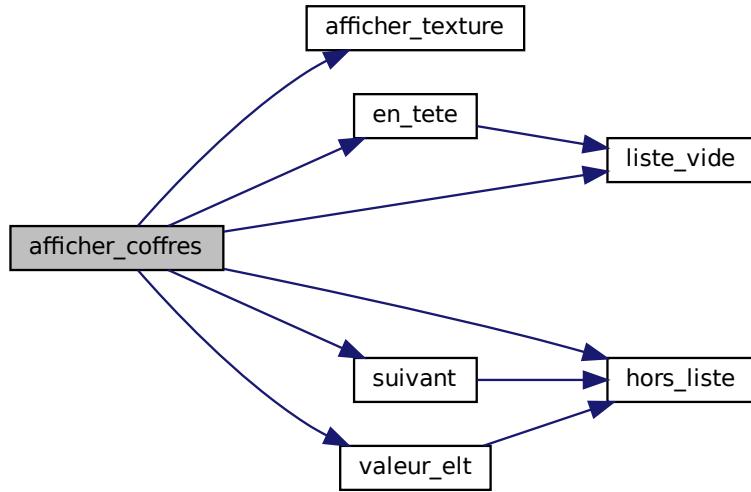
Auteur

Max Descomps

Paramètres

<code>liste_coffre</code>	Une liste de coffres
---------------------------	----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



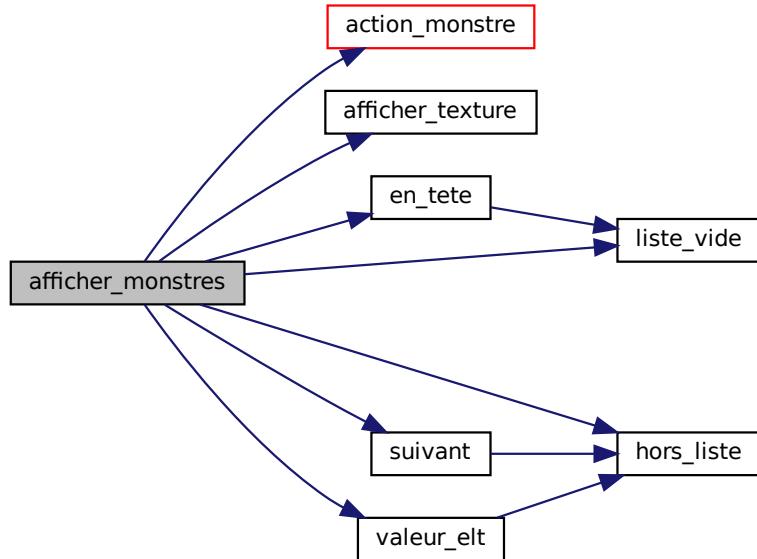
6.1.2.4 afficher_monstres()

```
void afficher_monstres (
    list * liste_monstre,
    joueur_t * joueur )
```

Paramètres

<i>liste_monstre</i>	une liste de monstres
<i>joueur</i>	le joueur qui influe sur les monstres

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



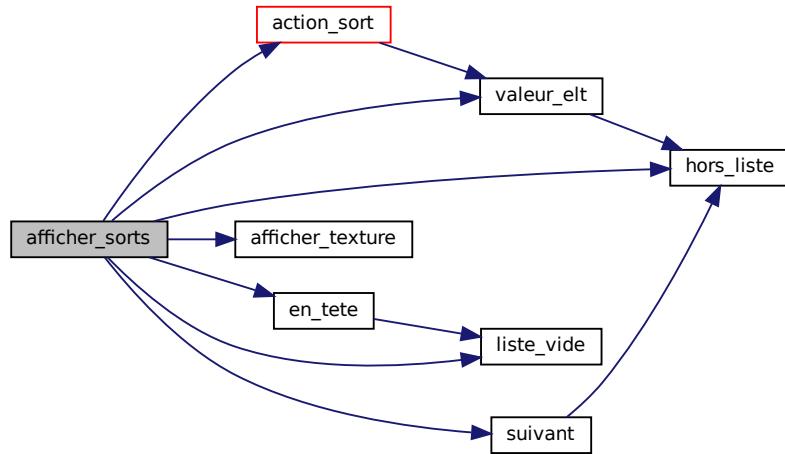
6.1.2.5 afficher_sorts()

```
void afficher_sorts (
    list * liste_sorts,
    joueur_t * joueur )
```

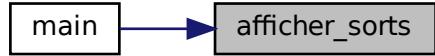
Paramètres

<i>liste_sorts</i>	une liste de sorts
<i>joueur</i>	le joueur qui influe sur les sorts

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.6 afficher_texture()

```
err_t afficher_texture (
    t_aff * texture,
    SDL_Renderer * rendu )
```

Auteur

Ange Despert

Cette fonction permet d'afficher à l'écran une texture utilisant le type personnalisé `t_aff` à l'écran avec le rendu donné en paramètre.

Rien ne sera afficher si la texture donnée en entrée est égale à NULL.

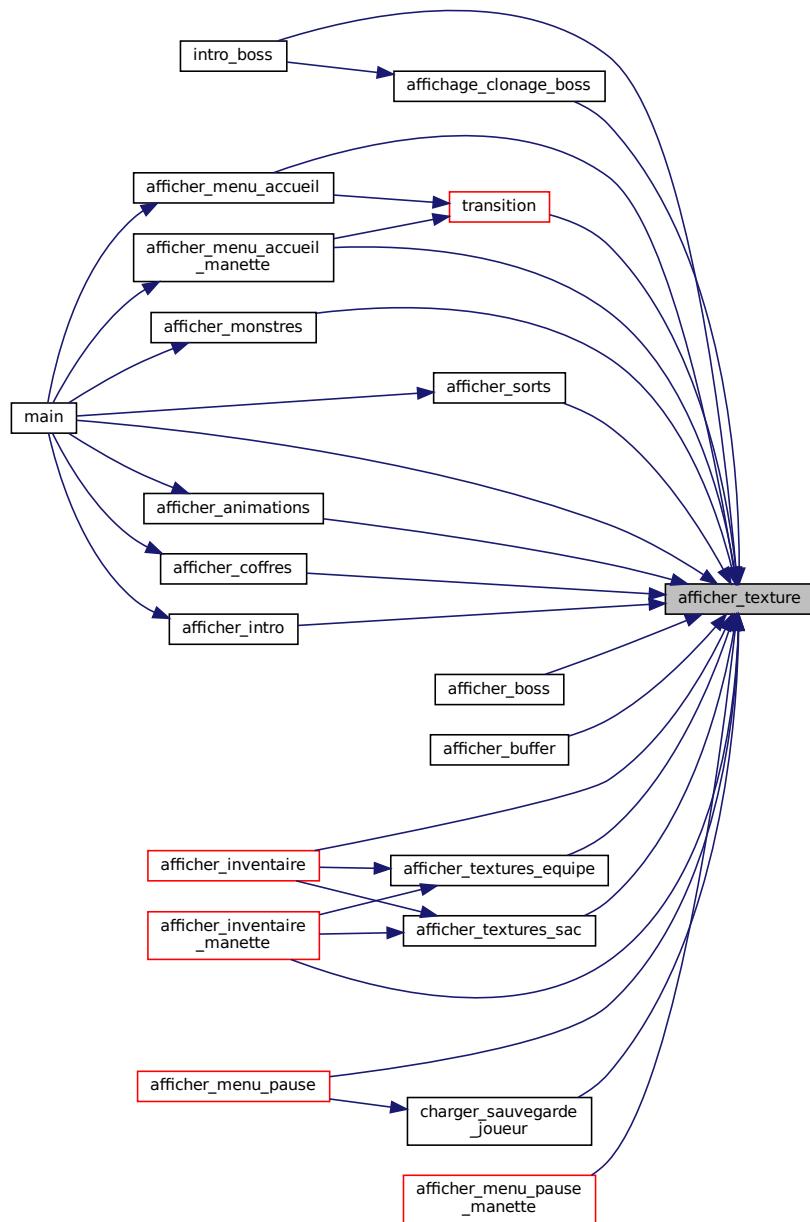
Paramètres

<code>texture</code>	La texture à afficher
<code>rendu</code>	Le rendu sur lequel afficher la texture à l'écran

Renvoie

0 s'il n'y a pas eu d'erreur sinon un entier négatif

Voici le graphe des appels de cette fonction :



6.1.2.7 ajout_text_liste()

```
void * ajout_text_liste (
    void * t )
```

Auteur

Ange Despert

Cette fonction sert juste à rérerencier les textures dans la [liste de textures](#).

Paramètres

<i>t</i>	une texture
----------	-------------

Renvoie

void * l'element à ajouter à la liste

Voici le graphe des appelants de cette fonction :



6.1.2.8 appliquer_coord_rect()

```
void appliquer_coord_rect (
    const SDL_Rect *const a_copier,
    t_l_aff * textures )
```

Voici le graphe des appelants de cette fonction :



6.1.2.9 color()

```
SDL_Color color (
    Uint8 r,
    Uint8 g,
    Uint8 b,
    Uint8 a )
```

Auteur

Max Descomps

Paramètres

<i>r</i>	niveau de rouge
<i>g</i>	niveau de vert
<i>b</i>	niveau de bleu
<i>a</i>	niveau d'opacité

Renvoie

SDL_Color une couleur SDL

Voici le graphe des appelants de cette fonction :



6.1.2.10 creer_texture()

```
t_aff* creer_texture (
    const char * nom_fichier,
    const int taille_t_x,
    const int taille_t_y,
    const int x,
    const int y,
    const float multiplicateur_taille )
```

Auteur

Ange Despert

Fonction qui permet de créer un texture sous forme d'un type personnalisé [t_aff](#) qui contient pleins d'informations sur la texture.

6.1.2.11 Déroulement

Le fichier de la texture est chargé dans une surface.

On va converir cette surface en texture.

On va remplir les informations de la structure `t_aff` avec les information en entrée et des informations obtenues via requêtes SDL.

Bien entendu, on testera pour être sur qu'il n'y a aucune erreur.

6.1.2.12 Lors d'une erreur

Contrairement à la plupart des fonctions qui gèrent les textures, celle-ci affichera surement un warning à l'écran si l'on ne peut pas créer la texture. Il est donc important de ce rendre compte que cette fonction ne ferme pas le programme lors d'une erreur. C'est donc au développeur de décider si une impossibilité de créer une texture est une raison de fermer le programme.

6.1.2.13 A noter

Si l'on donne -1 et -1 pour les paramètres `taille_t_x` et `taille_t_y` le rectangle `frame_anim` de la structure `t_aff` sera NULL.

Il est donc important de prendre cela en compte pour ne pas causer des erreurs de segmentation.

Donner NULL en nom de texture ne la créera pas mais créera tout les autres attributs de la structure `t_aff`.

Donner 0 en multiplicateur de taille agrandira la texture pour qu'elle face la taille de l'écran.

Paramètres

<code>nom_fichier</code>	Le nom du fichier contenant la texture
<code>taille_t_x</code>	La longueur de la texture à montrer
<code>taille_t_y</code>	La largeur de la texture à montrer
<code>x</code>	La coordonnée x où afficher la texture à l'écran
<code>y</code>	La coordonnée y où afficher la texture à l'écran
<code>multiplicateur_taille</code>	Une valeur par laquelle multiplier la taille de la texture

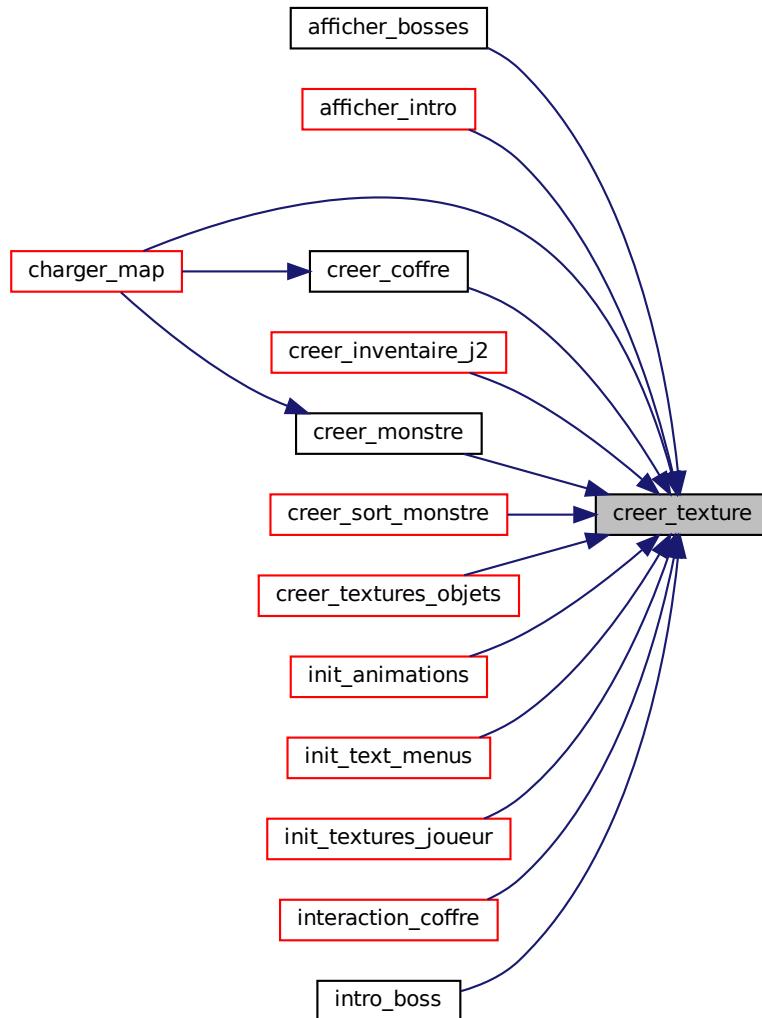
Renvoie

`t_aff*` Une structure qui permet l'affichage de la texture à l'écran ou NULL s'il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.14 current_frame_x()

```
int current_frame_x (
    t_aff * texture )
```

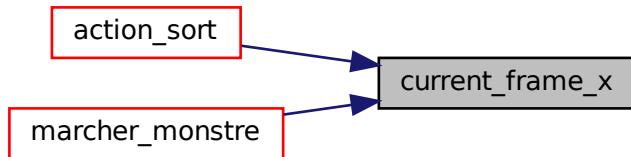
Paramètres

<i>texture</i>	une texture
----------------	-------------

Renvoie

un entier correspondant à l'indice

Voici le graphe des appelants de cette fonction :



6.1.2.15 current_frame_y()

```
int current_frame_y (
    t_aff * texture )
```

Paramètres

<i>texture</i>	une texture
----------------	-------------

Renvoie

un entier correspondant à l'indice

Voici le graphe des appelants de cette fonction :



6.1.2.16 def_texture_taille()

```
void def_texture_taille (
    t_aff * a_modifier,
    const int longueur,
    const int largeur )
```

Auteur

Ange Despert

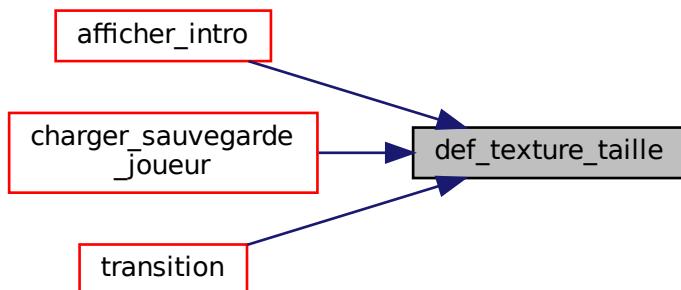
Obsolète L'utilisation de cette fonction n'a plus trop de sens étant donné que le moteur gère automatiquement la taille des textures.

Fonction qui permet de modifier le rectangle de la structure `t_aff : aff_fenetre` avec de nouvelles valeurs.

Paramètres

<code>a_modifie</code>	La texture à modifier
<code>longueur</code>	La nouvelle longueur en pixel à appliquer
<code>largeur</code>	La nouvelle largeur en pixel à appliquer

Voici le graphe des appels de cette fonction :



6.1.2.17 `deplacement_x_entite()`

```
bool deplacement_x_entite (
    t_map * m,
    t_aff * texture,
    int x,
    SDL_Rect * r )
```

Auteur

Ange Despert

Cette fonction permet à une entité de se déplacer sur l'axe x.

Cette fonction gère les collisions et empêchera l'entité de sortir des limites de la map.

Cette dernière prend également les collisions définies dans la liste des collisions de la map : [liste_collisions](#).

Il est à noté que pour éviter les déplacement trop rapides la fonction utilise l'entier `duree_frame_anim` qui permet d'empêcher le déplacement tout les x frames.

On peut récupérer l'élément que l'entité a touché en regardant l'élément courant de la [liste de collisions](#) de la map.

Paramètres

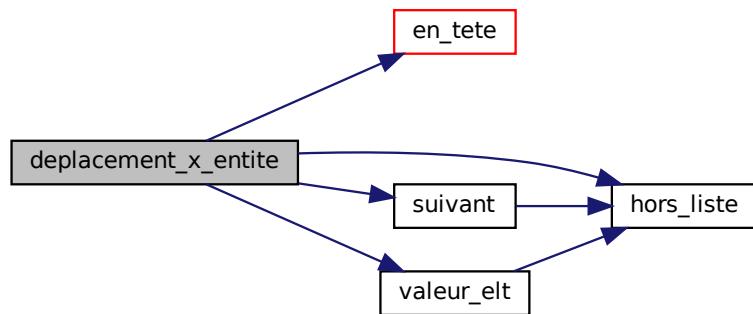
<i>m</i>	La map actuelle
<i>texture</i>	La texture de l'entité que l'on veut bouger
<i>x</i>	La nouvelle coordonnée du rectangle de l'entité
<i>r</i>	Le rectangle représentant la zone de collision de l'entité

Renvoie

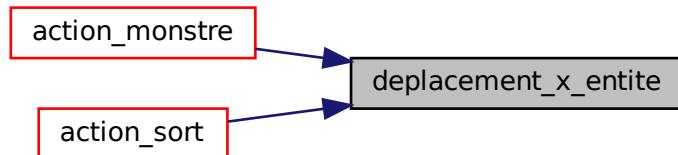
vrai : Si l'entité a réussi à se déplacer

faux : Si l'entité n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

6.1.2.18 `deplacement_x_joueur_secondaire()`

```

bool deplacement_x_joueur_secondaire (
    t_map * map,
    joueur_t * joueur,
    int x,
    SDL_Rect * r,
    lobjet_t * objets )
  
```

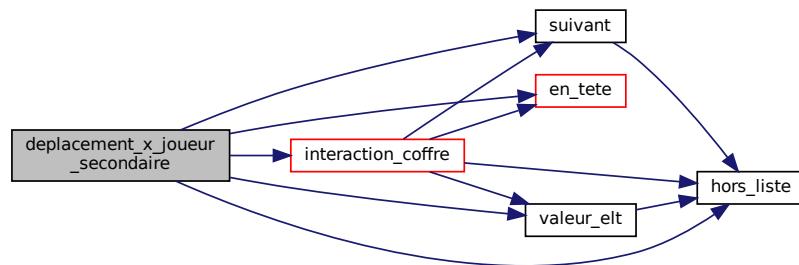
Paramètres

<i>map</i>	La map sur laquelle le personnage se déplace
<i>joueur</i>	Le joueur qui se déplace
<i>x</i>	Le nombre d'unités de déplacements
<i>r</i>	La zone de collision du joueur
<i>objets</i>	Les objets du jeu

Renvoie

vrai : Si le joueur a réussi à se déplacer
faux : Si le joueur n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.19 deplacement_x_pers()

```

bool deplacement_x_pers (
    t_map * map,
    joueur_t ** joueurs,
    unsigned short int nb_joueurs,
    int x,
    lobjet_t * objets )
    
```

Auteur

Ange Despert

Cette fonction utilise un rectangle `tx` pour savoir quand elle doit bouger le personnage ou bien déplacer la camera. Elle prendra en compte les collisions de la [liste de collisions de la map](#).

Elle empêche le personnage de sortir des bordures de la map.

Paramètres

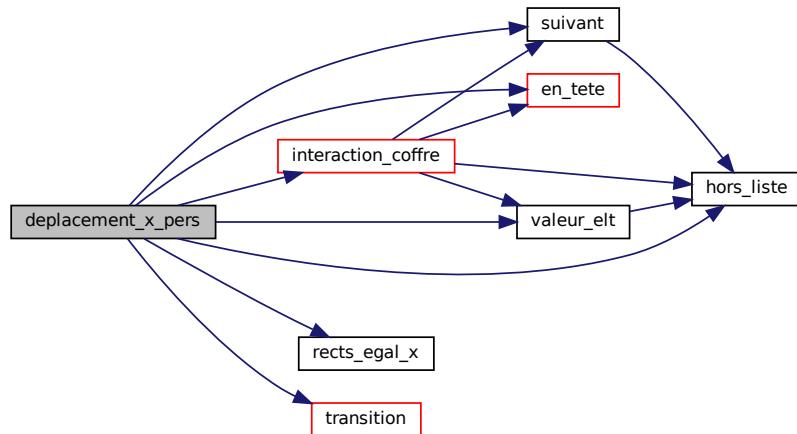
<code>map</code>	La map sur laquelle le personnage se déplace
<code>joueurs</code>	Les joueurs en jeu
<code>nb_joueurs</code>	Le nombre de joueurs en jeu
<code>x</code>	Le nombre d'unités de déplacements
<code>objets</code>	Les objets du jeu

Renvoie

vrai : Si le joueur s'est téléporté

faux : Si le joueur ne s'est pas téléporté <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.20 déplacement_y_entite()

```
bool déplacement_y_entite (
    t_map * m,
    t_aff * texture,
    int y,
    SDL_Rect * r )
```

Auteur

Ange Despert

Cette fonction permet à une entité de se déplacer sur l'axe y.

Cette fonction gère les collisions et empêchera l'entité de sortir des limites de la map.

Cette dernière prend également les collisions définies dans la liste des collisions de la map : [liste_collisions](#).

Il est à noté que pour éviter les déplacement trop rapides la fonction utilise l'entier [duree_frame_anim](#) qui permet d'empêcher le déplacement tout les x frames.

On peut récupérer l'élément que l'entité a touché en regardant l'élément courant de la [liste de collisions](#) de la map.

Paramètres

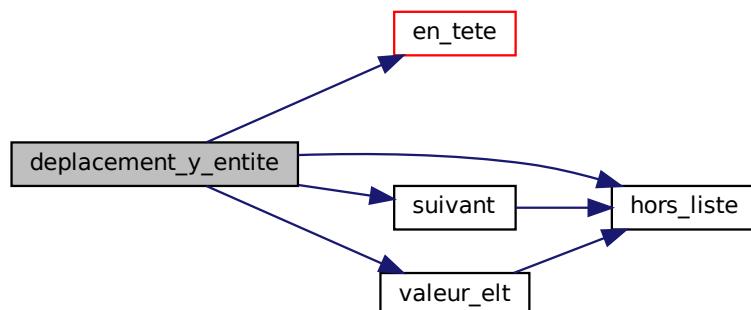
<i>m</i>	La map actuelle
<i>texture</i>	La texture de l'entité que l'on veut bouger
<i>y</i>	La nouvelle coordonnée du rectangle de l'entité
<i>r</i>	Le rectangle représentant la zone de collision de l'entité

Renvoie

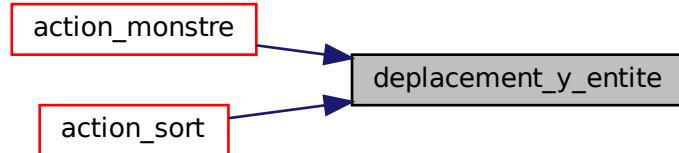
vrai : Si l'entité a réussi à se déplacer

faux : Si l'entité n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.21 deplacement_y_joueur_secondaire()

```

bool deplacement_y_joueur_secondaire (
    t_map * map,
    joueur_t * joueur,
    int y,
    SDL_Rect * r,
    lobjet_t * objets )
  
```

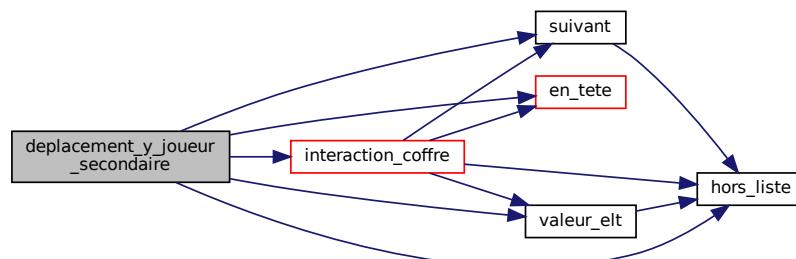
Paramètres

<i>map</i>	La map sur laquelle le personnage se déplace
<i>joueur</i>	Le joueur qui se déplace
<i>y</i>	Le nombre d'unités de déplacements
<i>r</i>	La zone de collision du joueur
<i>objets</i>	Les objets du jeu

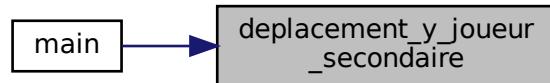
Renvoie

vrai : Si le joueur a réussi à se déplacer
faux : Si le joueur n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.22 deplacement_y_pers()

```

bool deplacement_y_pers (
    t_map * map,
    joueur_t ** joueurs,
    unsigned short int nb_joueurs,
    int y,
    lobjet_t * objets )
  
```

Auteur

Ange Despert

Cette fonction utilise un rectangle [ty](#) pour savoir quand elle doit bouger le personnage ou bien déplacer la camera. Elle prendra en compte les collisions de la [liste de collisions de la map](#).
Elle empêche le personnage de sortir des bordures de la map.

Paramètres

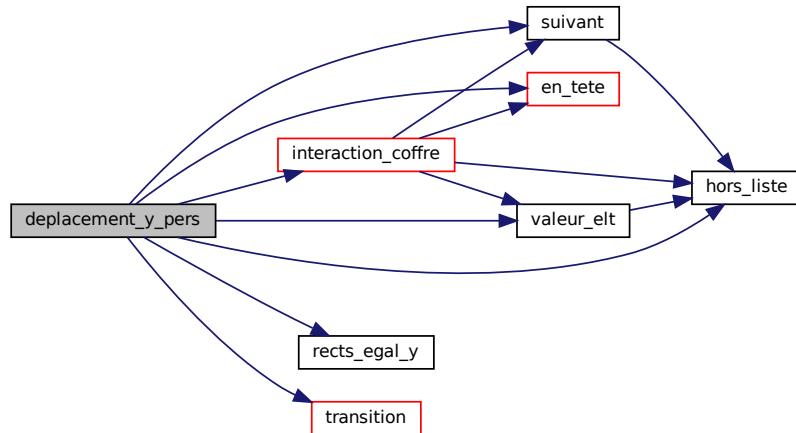
<i>map</i>	La map sur laquelle le personnage se déplace
<i>joueurs</i>	Les joueurs en jeu
<i>nb_joueurs</i>	Le nombre de joueurs en jeu
<i>y</i>	Le nombre d'unités de déplacements
<i>objets</i>	Les objets du jeu

Renvoie

vrai : Si le joueur s'est téléporté

faux : Si le joueur ne s'est pas téléporté <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.23 deplacer_rect_haut_droit()

```

void deplacer_rect_haut_droit (
    SDL_Rect * r,
    int x,
    int y )
  
```

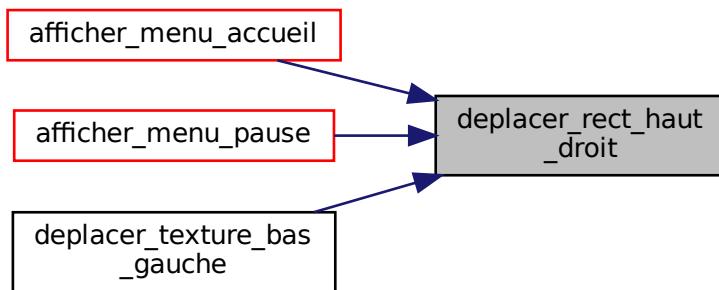
Auteur

Ange Despert

Paramètres

<i>r</i>	Le rectangle à placer
<i>x</i>	La coordonnée x du rectangle depuis la nouvelle origine.
<i>y</i>	La coordonnée y du rectangle depuis la nouvelle origine.

Voici le graphe des appelants de cette fonction :

**6.1.2.24 deplacer_rect_origine()**

```
void deplacer_rect_origine (
    SDL_Rect * r,
    int x,
    int y )
```

Auteur

Ange Despert

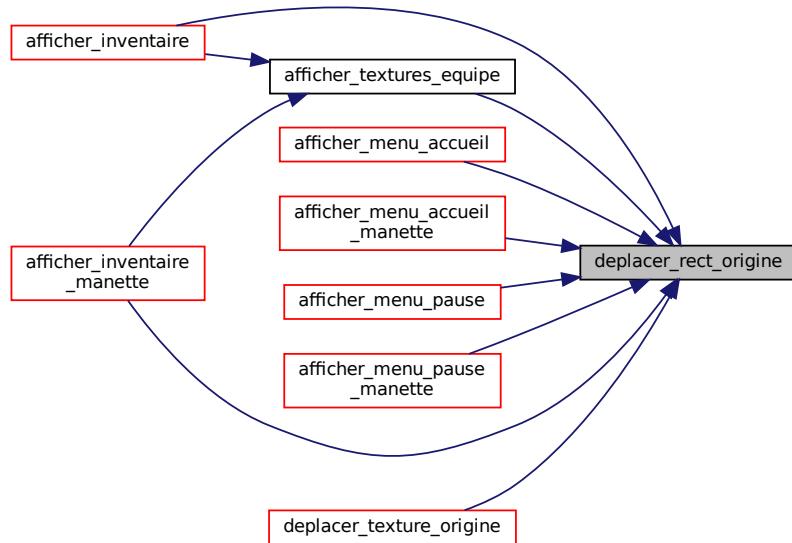
Paramètres

<i>r</i>	Le rectangle à déplacer
<i>x</i>	La position horizontale du rectangle
<i>y</i>	La position verticale du rectangle

Renvoie

Une valeur différente à 0 lors d'une erreur

Voici le graphe des appels de cette fonction :



6.1.2.25 `deplacer_texture_bas_droit()`

```
void deplacer_texture_bas_droit (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	Coordonnée x du coin supérieur gauche de la texture.
<i>y</i>	Coordonnée y du coin supérieur gauche de la texture.

6.1.2.26 `deplacer_texture_bas_gauche()`

```
void deplacer_texture_bas_gauche (
```

```
t_aff * texture,
int x,
int y )
```

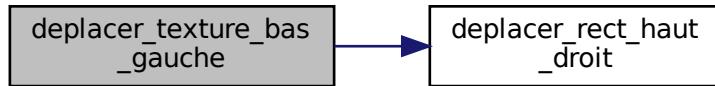
Auteur

Ange Despert

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	La coordonnée x du coin supérieur gauche de la texture.
<i>y</i>	Coordonnée y du coin supérieur gauche de la texture.

Voici le graphe d'appel pour cette fonction :

**6.1.2.27 deplacer_texture_centre()**

```
void deplacer_texture_centre (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez [place_rect_center_from_point](#) à l'aide du rectangle [frame_anim](#) de la variable [fenetre_finale](#).

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	La coordonnée x du centre de la texture.
<i>y</i>	La coordonnée y du centre de la texture.

Voici le graphe d'appel pour cette fonction :



6.1.2.28 `deplacer_texture_haut_droit()`

```
void deplacer_texture_haut_droit (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	La coordonnée x du coin supérieur gauche de la texture.
<i>y</i>	Coordonnée y du coin supérieur gauche du rectangle.

6.1.2.29 `deplacer_texture_origine()`

```
void deplacer_texture_origine (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

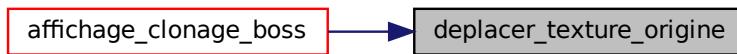
Paramètres

<i>texture</i>	La texture à déplacer
<i>x</i>	La coordonnée x de l'origine de la texture.
<i>y</i>	La coordonnée y de l'origine de la texture.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



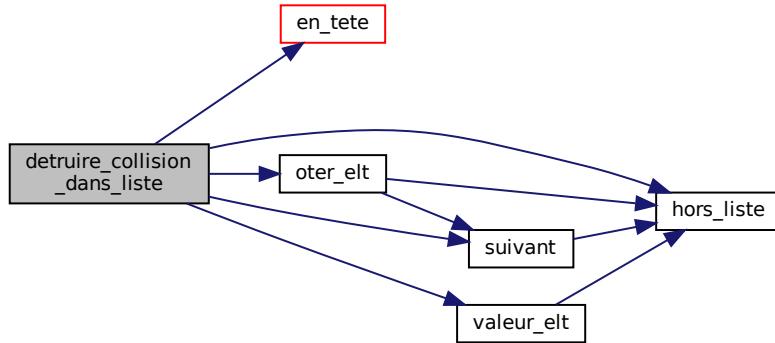
6.1.2.30 detruire_collision_dans_liste()

```
void detruire_collision_dans_liste (
    list * liste_collisions,
    SDL_Rect * collision )
```

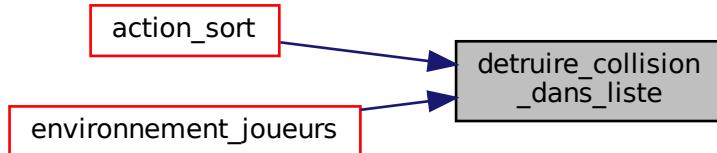
Paramètres

<i>liste_collisions</i>	Une liste de collisions
<i>collision</i>	La collision à détruire dans la liste

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.31 `detruire_liste_textures()`

```
void detruire_liste_textures (
    t_l_aff ** l_texture )
```

Auteur

Ange Despert

Paramètres

<code>l_texture</code>	L'adresse de la liste de texture à détruire
------------------------	---

6.1.2.32 detruire_texture()

```
void detruire_texture (
    t_aff ** texture )
```

Auteur

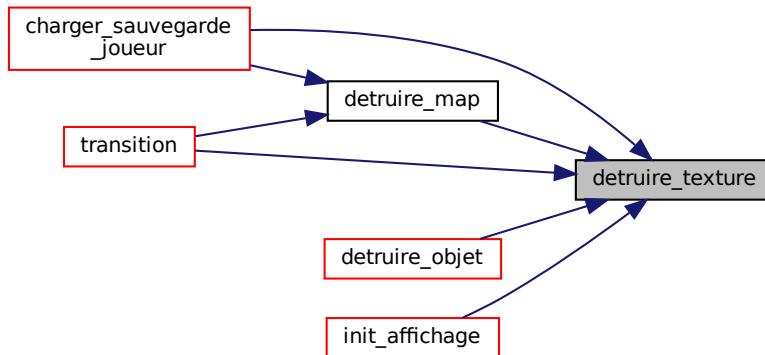
Despert Ange

Libère la mémoire allouée à la texture et la mets à NULL.

Paramètres

<i>texture</i>	L'adresse du pointeur sur la structure à détruire
----------------	---

Voici le graphe des appels de cette fonction :



6.1.2.33 get_rect_center()

```
SDL_Point get_rect_center (
    const SDL_Rect *const r )
```

Cette fonction contrairement à la fonction [get_rect_center_coord](#) donne les coordonnées strictes du centre.

Auteur

Ange Despert

Paramètres

<i>r</i>	Le rectangle dont on veut les coordonnées du milieu
----------	---

Renvoie

Les coordonnées du milieu du rectangle

Voici le graphe des appelants de cette fonction :



6.1.2.34 get_rect_center_coord()

```
SDL_Point get_rect_center_coord (
    const SDL_Rect *const r )
```

Cette fonction contrairement à la fonction [get_rect_center](#) donne les coordonnées du centre en prenant en compte les coordonnées actuelles du rectangle.

Cette fonction est donc faite pour être utilisée conjointement à la fonction [place_rect_center_from_point\(SDL_Rect *r, SDL_Point p\)](#)

Auteur

Ange Despert

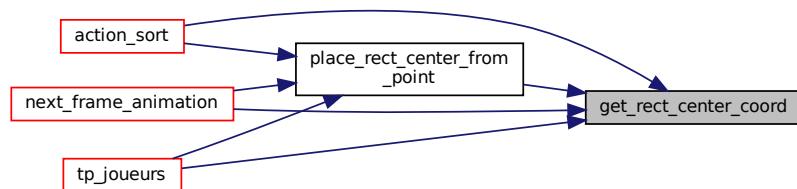
Paramètres

<i>r</i>	Le rectangle dont on veut les coordonnées du milieu.
----------	--

Renvoie

Les coordonnées du milieu du rectangle.

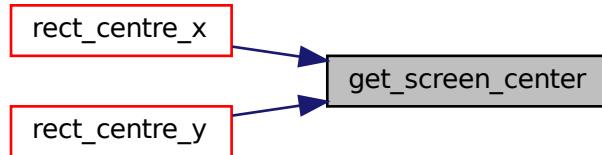
Voici le graphe des appelants de cette fonction :



6.1.2.35 get_screen_center()

```
point get_screen_center ( )
```

Voici le graphe des appelants de cette fonction :



6.1.2.36 info_texture()

```
void info_texture ( t_aff * texture )
```

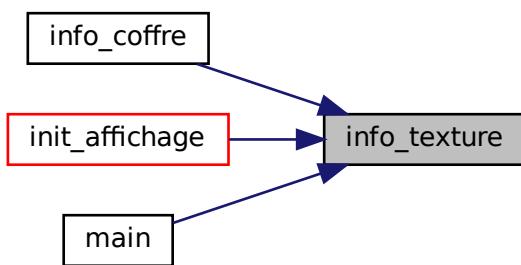
Auteur

Max Descomps

Paramètres

<code>texture</code>	la texture sur laquelle on se renseigne
----------------------	---

Voici le graphe des appelants de cette fonction :



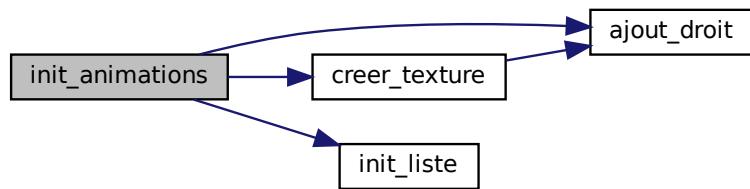
6.1.2.37 init_animations()

```
void init_animations ( )
```

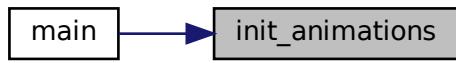
Auteur

Max Descomps

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.38 init_texture_joueur()

```
t_aff* init_texture_joueur (
    t_l_aff * textures_joueur,
    joueur_t * joueur )
```

Auteur

Antoine Bruneau

Paramètres

<i>textures_joueur</i>	Liste de textures personnage
<i>joueur</i>	Joueur dont on veut la texture de départ

Renvoie

*t_aff** Une textures personnage

6.1.2.39 init_textures_joueur()

```
t_l_aff * init_textures_joueur (
    joueur_t * j,
    int num_j )
```

Auteur

Antoine Bruneau

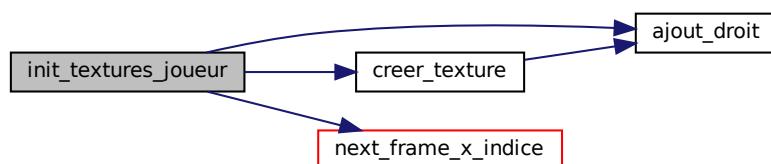
Paramètres

<i>j</i>	Le joueur dont on initialise les textures
<i>num_j</i>	Indice du joueur dans le tableau des joueurs

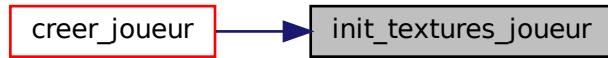
Renvoie

*t_l_aff** Une liste de textures

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.40 lister_animations()

```
void lister_animations (
    joueur_t ** joueurs,
    list * animations )
```

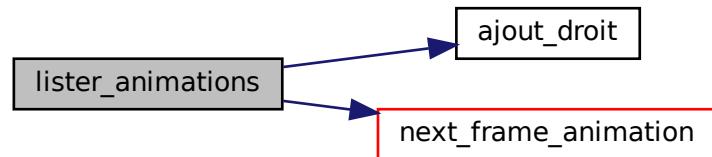
Auteur

Max Descomps

Paramètres

<i>joueurs</i>	Les joueurs sur lesquels placer les animation
<i>animations</i>	Liste regroupant les animations trouvées

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.41 modif_affichage_rect()

```
void modif_affichage_rect (
    t_aff * texture,
    SDI_Rect r )
```

Auteur

Ange Despert

Paramètres

<i>texture</i>	La texture à modifier.
<i>r</i>	Le rectangle à appliquer.

6.1.2.42 next_frame_animation()

```
t_aff * next_frame_animation (
    joueur_t * joueur )
```

Auteur

Max Descomps

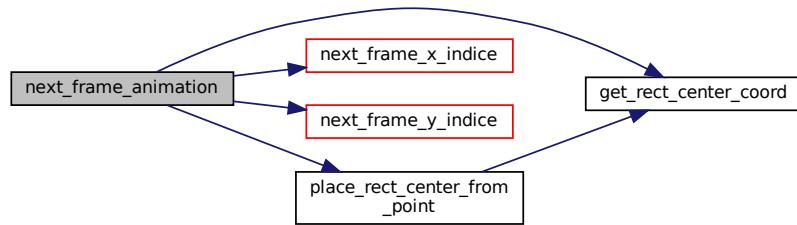
Paramètres

<i>joueur</i>	Le joueur sur lequel placer l'animation
---------------	---

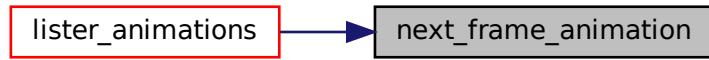
Renvoie

La texture de l'animation

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.43 next_frame_indice()

```
void next_frame_indice (
    t_aff * texture,
    const unsigned int x,
    const unsigned int y )
```

Auteur

Despert Ange

Fonction qui permet de choisir le sprite à un certain indice de l'image.

Cette fonction est conçue pour être utiliser avec une texture qui contient plusieurs sprites de même taille.

La fonction utilisera la taille du rectangle charger d'afficher qu'une partie d'une image.

Cette fonction est la pour éviter deux appels de fonctions, en l'occurrence [next_frame_x_indice](#) et [next_frame_y_indice](#)

Paramètres

<i>texture</i>	une texture joueur
<i>x</i>	qui correspond au n-ème sprite sur l'axe des x ou l'on souhaite positionner la texture
<i>y</i>	qui correspond au n-ème sprite sur l'axe des y ou l'on souhaite positionner la texture

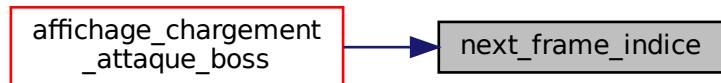
Renvoie

`err_t` un entier pour savoir si il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



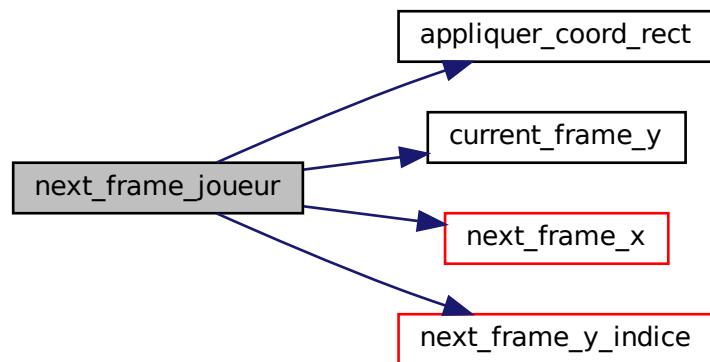
Voici le graphe des appels de cette fonction :



6.1.2.44 `next_frame_joueur()`

```
t_aff* next_frame_joueur (
    joueur_t * j )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.45 next_frame_x()

```
void next_frame_x (
    t_aff * texture )
```

Auteur

Ange Despert

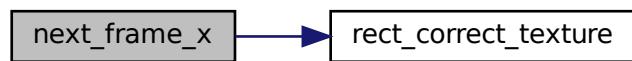
Fonction qui passe au sprite suivant dans l'axe x. Cette fonction fera une boucle complète de l'image, il n'y a donc pas à s'inquiéter d'arriver à la fin de l'image.

Comme les fonction à indice, cette fonction fait appel au rectangle `frame_anim` de la structure [d'affichage](#).

Paramètres

<code>texture*</code>	une texture joueur
-----------------------	--------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.46 next_frame_x_indice()

```
void next_frame_x_indice (
    t_aff * texture,
    const unsigned int indice )
```

Auteur

Ange Despert

Fonction qui permet de choisir le sprite a un certain indice de l'image.

Cette fonction est concue pour être utiliser avec une texture qui contient plusieurs sprites de même taille.

La fonction utilisera la taille du [rectangle](#) charger d'afficher qu'une partie d'une image.

Paramètres

<i>texture*</i>	une texture joueur
<i>indice</i>	unsigned int qui correspond au n-ème sprite sur l'axe des x ou l'on souhaite positionner la texture

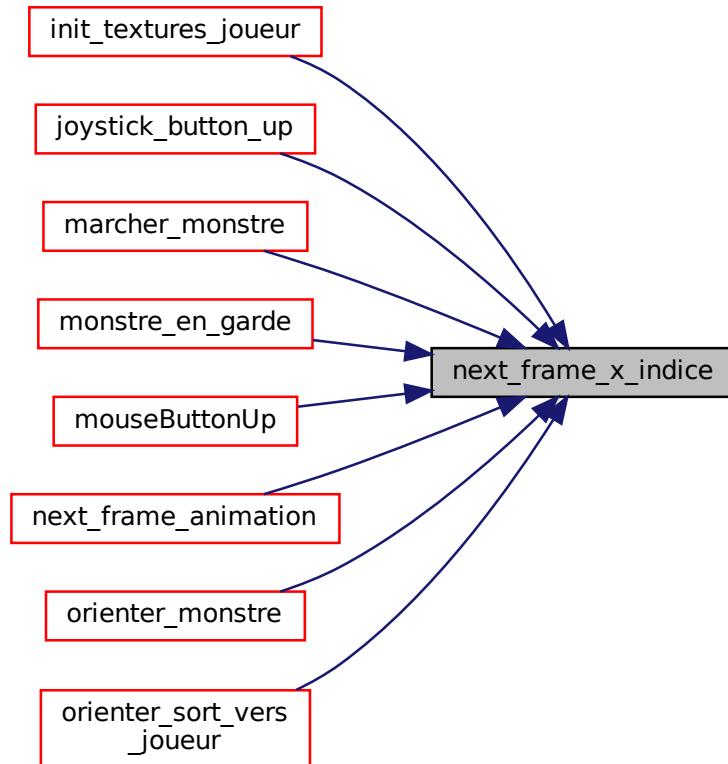
Renvoie

err_t un entier pour savoir si il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.47 next_frame_y()

```
void next_frame_y (
    t_aff * texture )
```

Auteur

Ange Despert

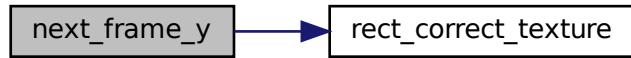
Fonction qui passe au sprite suivant dans l'axe y. Cette fonction fera une boucle complète de l'image, il n'y a donc pas à s'inquiéter d'arriver à la fin de l'image.

Comme les fonction à indice, cette fonction fait appel au rectangle [frame_anim](#) de la structure [d'affichage](#).

Paramètres

<i>texture*</i>	une texture joueur
-----------------	--------------------

Voici le graphe d'appel pour cette fonction :



6.1.2.48 `next_frame_y_indice()`

```
void next_frame_y_indice (
    t_aff * texture,
    const unsigned int indice )
```

Paramètres

<i>texture</i>	Une texture joueur
<i>indice</i>	Correspond au n-ème sprite sur l'axe des y ou l'on souhaite positionner la texture

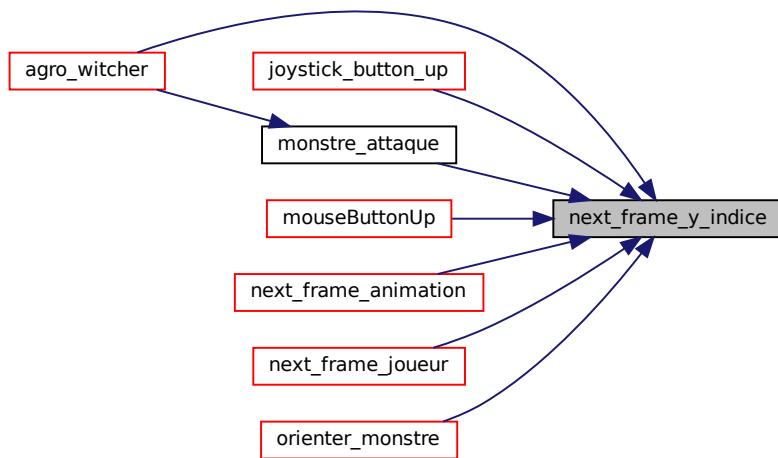
Renvoie

`err_t` un entier pour savoir si il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.49 place_rect_center_from_point()

```
void place_rect_center_from_point (
    SDL_Rect * r,
    SDL_Point p )
```

Le but de cette fonction est de pouvoir placer plusieurs rectangles au même endroit peut importe leur taille. C'est fonction est donc très puissante si elle est utilisée conjointement à la fonction [get_rect_center_coord\(const SDL_Rect *const r\)](#)

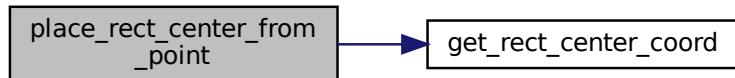
Auteur

Ange Despert

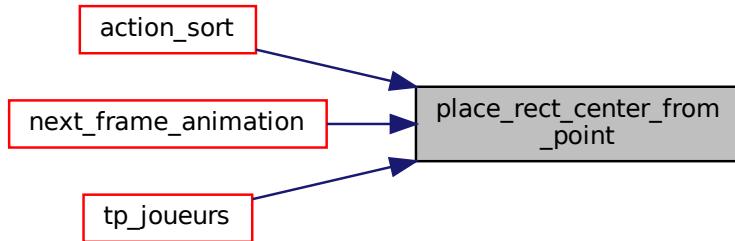
Paramètres

<i>r</i>	Le rectangle que l'on veut déplacer
<i>p</i>	Le point où on veut placer le milieu du rectangle

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.50 placer_texture()

```
void placer_texture (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Max Descomps

Paramètres

<i>texture</i>	Texture à placer
<i>x</i>	Position horizontale
<i>y</i>	Position verticale

Voici le graphe des appels de cette fonction :



6.1.2.51 point_in_rect()

```
bool point_in_rect (
    SDL_Rect r,
    point p )
```

6.1.2.52 rect_centre()

```
void rect_centre (
    SDL_Rect * rectangle )
```

Auteur

Ange Despert

Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.
Utilisez [place_rect_center_from_point](#) à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

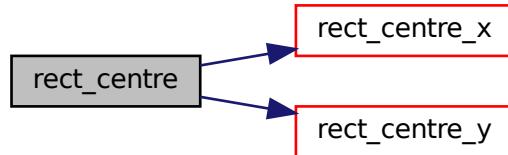
Paramètres

<code>rectangle</code>	Le rectangle à placer
------------------------	-----------------------

Renvoie

Un booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.53 rect_centre_rect()

```
void rect_centre_rect (
    SDL_Rect * rectangle,
    SDL_Rect * rectangle_centre )
```

Auteur

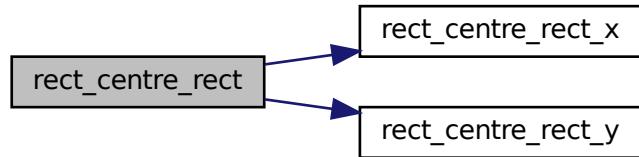
Ange Despert

Fonctionnement similaire à l'utilisation conjointe des fonctions [place_rect_center_from_point](#) et [get_rect_center_coord](#)

Paramètres

<i>rectangle</i>	Le rectangle à centrer
<i>rectangle_centre</i>	Le rectangle dans lequel en centre un autre

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.1.2.54 `rect_centre_rect_x()`

```
void rect_centre_rect_x (
    SDL_Rect * rectangle,
    SDL_Rect * rectangle_centre )
```

Auteur

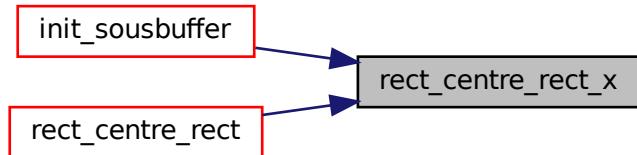
Ange Despert

Fonction qui fonctionne de la même manière que la fonction [rect_centre_rect](#) mais permet de seulement centrer l'axe x.

Paramètres

<code>rectangle</code>	Le rectangle à centrer
<code>rectangle_centre</code>	Le rectangle dans lequel en centre un autre

Voici le graphe des appelants de cette fonction :



6.1.2.55 rect_centre_rect_y()

```
void rect_centre_rect_y (
    SDL_Rect * rectangle,
    SDL_Rect * rectangle_centre )
```

Auteur

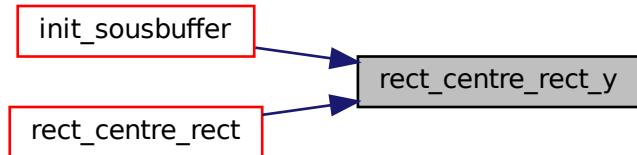
Ange Despert

Fonction qui fonctionne de la même manière que la fonction [rect_centre_rect](#) mais permet de seulement centrer l'axe y.

Paramètres

<i>rectangle</i>	Le rectangle à centrer
<i>rectangle_centre</i>	Le rectangle dans lequel en centre un autre

Voici le graphe des appelants de cette fonction :



6.1.2.56 rect_centre_x()

```
void rect_centre_x (
    SDL_Rect * rectangle )
```

Auteur

Ange Despert

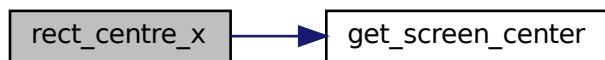
Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez [place_rect_center_from_point](#) à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

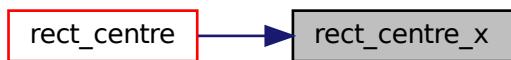
Paramètres

<i>rectangle</i>	Le rectangle à placer
------------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.57 rect_centre_y()

```
void rect_centre_y (
    SDL_Rect * rectangle )
```

Auteur

Ange Despert

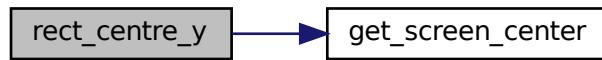
Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez [place_rect_center_from_point](#) à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

Paramètres

<i>rectangle</i>	Le rectangle à placer
------------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.1.2.58 rect_correct_texture()

```
bool rect_correct_texture (
    const SDL_Rect *const to_verify,
    const int width,
    const int height )
```

Auteur

Ange Despert

Cette fonction est appellée par les fonction de déplacement de frame tel que [next_frame_x](#) pour s'assurer qu'il n'y a aucune erreur.

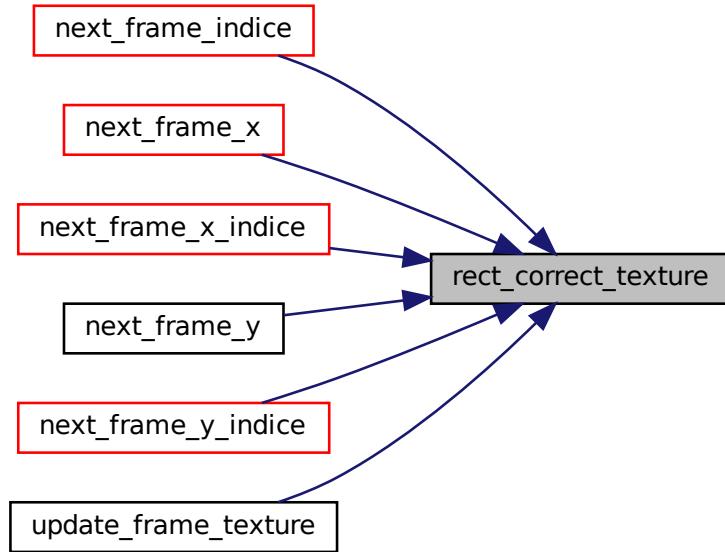
Paramètres

<i>to_verify</i>	La structure SDL_Rect à vérifier
<i>width</i>	La longueur de la texture
<i>height</i>	La largeur de la texture

Renvoie

Un booléen

Voici le graphe des appels de cette fonction :



6.1.2.59 rect_ecran_to_rect_map()

```
void rect_ecran_to_rect_map (
    SDL_Rect * ecran,
    SDL_Rect * r_map,
    int x,
    int y )
```

6.1.2.60 rects_egal_x()

```
bool rects_egal_x (
    const SDL_Rect *const r1,
    SDL_Rect const *const r2 )
```

Auteur

Ange Despert

Cette fonction permet de savoir si deux rectangles sont égaux mais en prenant seulement en compte l'axe x. Cette fonction est utilisée par la fonction [la fonction de déplacement de personnage](#).

Paramètres

<i>r1</i>	Le premier rectangle à comparer
<i>r2</i>	Le deuxième rectangle à comparer

Renvoie

Un booléen <Définition du type booléen

Voici le graphe des appelants de cette fonction :



6.1.2.61 rects_egal_y()

```
bool rects_egal_y (
    const SDL_Rect *const r1,
    SDL_Rect const *const r2 )
```

Auteur

Ange Despert

Cette fonction permet de savoir si deux rectangles sont égaux mais en prenant seulement en compte l'axe y. Cette fonction est utilisée par la fonction [la fonction de déplacement de personnage](#).

Paramètres

<i>r1</i>	Le premier rectangle à comparer
<i>r2</i>	Le deuxième rectangle à comparer

Renvoie

Un booléen <Définition du type booléen

Voici le graphe des appelants de cette fonction :



6.1.2.62 text_copier_position()

```
void text_copier_position (
    t_aff * a_modifier,
    const t_aff *const original )
```

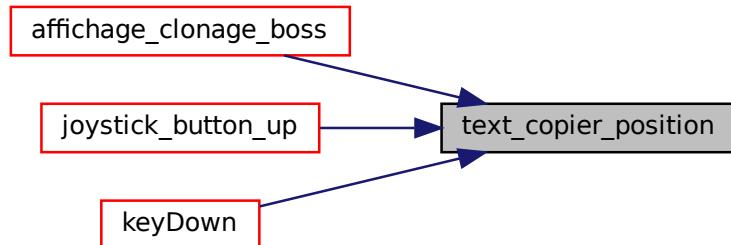
Auteur

Max Descomps

Paramètres

<i>a_modifier</i>	La texture dont on veut modifier la position
<i>original</i>	La texture dont on veut copier la position

Voici le graphe des appelants de cette fonction :



6.1.2.63 update_frame_texture()

```
err_t update_frame_texture (
    t_aff * texture,
    const int x,
    const int y )
```

Voici le graphe d'appel pour cette fonction :



6.1.3 Documentation des variables

6.1.3.1 bloquer

```
t_aff* bloquer = NULL
```

La texture de l'animation de blocage

6.1.3.2 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à [NB_FPS](#)

6.1.3.3 fenetre_finale

```
t_aff* fenetre_finale = NULL
```

La texture de la fenêtre de jeu finale sans l'interface

6.1.3.4 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.1.3.5 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

6.1.3.6 heal

```
t_aff* heal = NULL
```

La texture de l'animation de heal

6.1.3.7 liste_animations

```
list* liste_animations = NULL
```

La liste des animation à jouer

6.1.3.8 listeDeTextures

```
list* listeDeTextures
```

La liste de toutes les textures qui ont été créées. Sert à leur destruction

6.1.3.9 multiplicateur_x

```
float multiplicateur_x
```

Multiplicateur qui dépend de la résolution (longueur), vaut 1 pour 1920

6.1.3.10 multiplicateur_y

```
float multiplicateur_y
```

Multiplicateur qui dépend de la résolution (largeur), vaut 1 pour 1080

6.1.3.11 tx

```
SDL_Rect tx
```

Rectangle servant au déplacement du personnage principal en x

6.1.3.12 ty

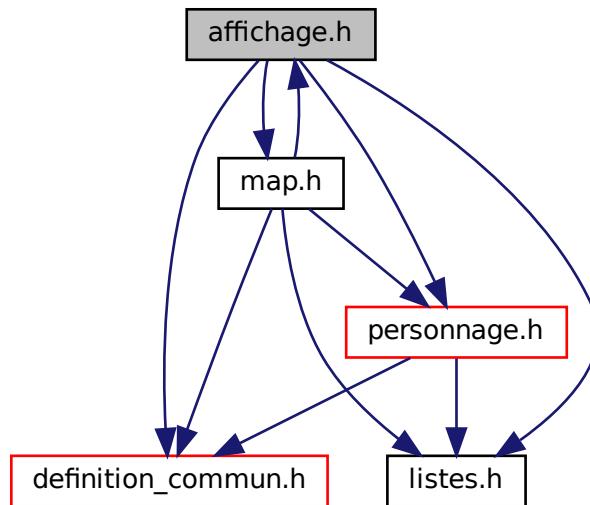
```
SDL_Rect ty
```

Rectangle servant au déplacement du personnage principal en y

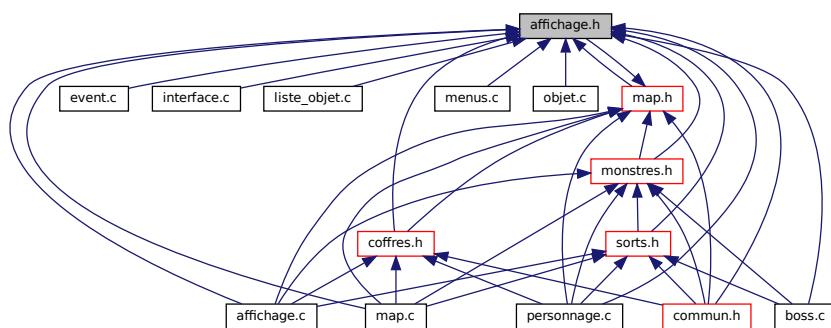
6.2 Référence du fichier affichage.h

Fichier contenant les définitions et les fonctions liées au module affichage.

Graphe des dépendances par inclusion de affichage.h:



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



Classes

- struct **t_aff**
Structure qui permet l'affichage d'une texture à l'écran de manière précise.
- struct **t_l_aff**
Structure contenant la liste des textures créées par le programme.

Macros

```
— #define NB_FPS 60
— #define NB_SPRITE_JOUEUR 5
— #define N_T_MARCHER "ressources/sprite/marcher.bmp"
— #define N_T_ATTAQUE "ressources/sprite/attaque.bmp"
— #define N_T_ATTAQUE_CHARGE "ressources/sprite/attaque_charge.bmp"
— #define N_T_CHARGER "ressources/sprite/charger.bmp"
— #define N_T_MARCHER_BOUCLIER "ressources/sprite/marcher_bouclier.bmp"
— #define N_T_MARCHER2 "ressources/sprite/marcher_green.bmp"
— #define N_T_ATTAQUE2 "ressources/sprite/attaque_green.bmp"
— #define N_T_ATTAQUE_CHARGE2 "ressources/sprite/attaque_charge_green.bmp"
— #define N_T_CHARGER2 "ressources/sprite/charger_green.bmp"
— #define N_T_MARCHER_BOUCLIER2 "ressources/sprite/marcher_bouclier_green.bmp"
— #define LONGUEUR_ENTITE 48
— #define LARGEUR_ENTITE 48
```

Énumérations

```
— enum t_texture_perso {
    TEXT_MARCHER, TEXT_ATTAQUE, TEXT_ATTAQUE_CHARGE, TEXT_CHARGER,
    TEXT_MARCHER_BOUCLIER }
```

Type énuméré renseignant sur la texture personnage à utiliser.

Fonctions

```
— void detruire_texture (t_aff **texture)
    Fonction qui détruit une structure d'affichage de texture passée en paramètre.
— void detruire_liste_textures (t_l_aff **l_texture)
    Fonction qui détruit une liste de textures.
— _Bool rect_correct_texture (const SDL_Rect *const to_verify, const int width, const int height)
    Fonction qui détermine si la structure SDL_Rect ne dépasse pas la taille de la texture.
— t_aff * creer_texture (const char *nom_fichier, const int taille_t_x, const int taille_t_y, const int x, const int y,
    const float multiplicateur_taille)
    Fonction qui renvoie, charge une texture et la prépare à se faire afficher.
— err_t afficher_texture (t_aff *texture, SDL_Renderer *rendu)
    Affiche la texture donnée en paramètre à l'écran.
— void next_frame_y (t_aff *texture)
    Fonction qui positionne la texture au sprite d'après sur l'axe des y.
— void next_frame_x (t_aff *texture)
    Fonction qui positionne la texture au sprite d'après sur l'axe des x.
— err_t next_frame_indice (t_aff *texture, const unsigned int x, const unsigned int y)
    Fonction qui positionne la texture au n-ème sprite sur l'axe des x.
— err_t next_frame_x_indice (t_aff *texture, const unsigned int indice)
    Fonction qui positionne la texture au n-ème sprite sur l'axe des x.
— err_t next_frame_y_indice (t_aff *texture, const unsigned int indice)
    Fonction qui positionne la texture au n-ème sprite sur l'axe des y.
— void * ajout_texture_liste (void *)
    Fonction qui ajoute une texture dans une liste générique.
— err_t afficher_buffer (list *buffer, SDL_Renderer *rendu)
    Fonction qui affiche les textures contenues dans la liste en paramètre.
— void placer_texture_centre (t_aff *texture, int x, int y)
    Déplace la texture pour que son centre soit au centre de l'écran.
— void placer_rect_origine (SDL_Rect *r, int x, int y)
    Déplace un rectangle depuis l'origine de l'écran.
— void placer_texture_origine (t_aff *texture, int x, int y)
    Déplace l'origine de la texture aux coordonnées données.
— void placer_rect_haut_droit (SDL_Rect *r, int x, int y)
    Place un rectangle en haut à droite de l'écran puis le replace à partir de cette origine.
— void placer_texture_haut_droit (t_aff *texture, int x, int y)
    La texture est déplacée vers la droite et vers le haut.
— void placer_texture_bas_gauche (t_aff *texture, int x, int y)
    La texture est déplacée vers le coin inférieur gauche de l'écran.
— void placer_texture_bas_droit (t_aff *texture, int x, int y)
    La texture est déplacée vers le coin inférieur droit de la fenêtre.
```

```

— void modif_affichage_rect (t_aff *texture, SDL_Rect r)
    Modifie le rectangle qui définit la zone de l'écran qui sera utilisée pour le rendu de la texture.
— t_l_aff * init_textures_joueur (joueur_t *j, int num_j)
    Fonction qui créer et renvoie une liste de textures pour le personnage (joueur)
— t_aff * init_texture_joueur (t_l_aff *textures_joueur, joueur_t *joueur)
    Fonction qui renvoie la texture de départ du personnage (joueur)
— t_aff * next_frame_joueur (joueur_t *j)
— _Bool déplacement_x_pers (t_map *map, joueur_t **joueurs, unsigned short int nb_joueurs, int x, lobjet_t *objets)
    Permet de déplacer le personnage de x unités horizontales sur la map.
— _Bool déplacement_y_pers (t_map *map, joueur_t **joueurs, unsigned short int nb_joueurs, int y, lobjet_t *objets)
    Permet de déplacer le personnage principal de y unités verticales sur la map.
— _Bool déplacement_y_joueur_secondaire (t_map *map, joueur_t *joueur, int y, SDL_Rect *r, lobjet_t *objets)
    Permet de déplacer un joueur secondaire de y unités verticales sur la map.
— _Bool déplacement_x_joueur_secondaire (t_map *map, joueur_t *joueur, int x, SDL_Rect *r, lobjet_t *objets)
    Permet de déplacer un joueur secondaire de x unités horizontales sur la map.
— void def_texture_taille (t_aff *a_modifier, const int longueur, const int largeur)
    Fonction qui permet de définir exactement la taille de la texture.
— void text_copier_position (t_aff *a_modifier, const t_aff *const original)
    Fonction qui permet de placer 2 textures aux mêmes endroit à l'écran.
— void rect_centre_x (SDL_Rect *rectangle)
    Fonction qui permet de placer un rectangle au centre de l'écran sur l'axe des x.
— void rect_centre_y (SDL_Rect *rectangle)
    Fonction qui permet de placer un rectangle au centre de l'écran sur l'axe des y.
— void rect_centre (SDL_Rect *rectangle)
    Fonction qui permet de placer un rectangle au centre de l'écran.
— _Bool rects_egal_x (const SDL_Rect *const r1, SDL_Rect const *const r2)
    Fonction qui permet de savoir si deux rectangles sont égaux sur l'axe x.
— _Bool rects_egal_y (const SDL_Rect *const r1, SDL_Rect const *const r2)
    Fonction qui permet de savoir si deux rectangles sont égaux sur l'axe y.
— SDL_Color color (Uint8 r, Uint8 g, Uint8 b, Uint8 a)
    Fonction qui permet de choisir une couleur SDL.
— void info_texture (t_aff *texture)
    Affiche des informations en console sur une texture.
— int current_frame_x (t_aff *texture)
    Fonction qui donne l'indice sur l'axe des abscisses actuelle de la texture.
— int current_frame_y (t_aff *texture)
    Fonction qui donne l'indice sur l'axe des ordonnées actuelle de la texture.
— void afficher_monstres (list *liste_monstre, joueur_t *joueur)
    Fonction qui affiche les monstres.
— void afficher_sorts (list *liste_sorts, joueur_t *joueur)
    Fonction qui affiche les sorts.
— void placer_texture (t_aff *texture, int x, int y)
    Place une texture sur l'écran.
— void init_animations (void)
    Initialise les textures des animations.
— t_aff * next_frame_animation (joueur_t *joueur)
    Fait évoluer les animations en jeu.
— void lister_animations (joueur_t **joueurs, list *animations)
    Liste les animations en jeu.
— void afficher_coffres (list *liste_coffre)
    Fonction qui affiche la texture des coffres.
— void rect_centre_rect_x (SDL_Rect *rectangle, SDL_Rect *rectangle_centre)
    Fonction qui permet de placer un rectangle au centre d'un autre sur l'axe des x.
— void rect_centre_rect_y (SDL_Rect *rectangle, SDL_Rect *rectangle_centre)
    Fonction qui permet de placer un rectangle au centre d'un autre sur l'axe des y.
— void rect_centre_rect (SDL_Rect *rectangle, SDL_Rect *rectangle_centre)
    Fonction qui permet de placer un rectangle au centre d'un autre.
— void afficher_animations (list *animations)
    Fonction qui affiche la texture des animations.
— _Bool déplacement_x_entité (t_map *m, t_aff *texture, int x, SDL_Rect *r)
    Fonction qui permet le déplacement d'une entité
— _Bool déplacement_y_entité (t_map *m, t_aff *texture, int y, SDL_Rect *r)
    Fonction qui permet le déplacement d'une entité
— void detruire_collision_dans_liste (list *liste_collisions, SDL_Rect *collision)
    Fonction qui détruit une collision donnée dans une liste de collisions.

```

- `SDL_Point get_rect_center (const SDL_Rect *const r)`
Renvoie les coordonnées du centre du rectangle.
- `SDL_Point get_rect_center_coord (const SDL_Rect *const r)`
Renvoie les coordonnées du centre du rectangle.
- `void place_rect_center_from_point (SDL_Rect *r, SDL_Point p)`
Fonction qui permet de placer le centre du rectangle donné en paramètre à un point précis.

Variables

- `list * listeDeTextures`
- `long int compteur`
- `SDL_Rect tx`
- `SDL_Rect ty`
- `float multiplicateur_x`
- `float multiplicateur_y`
- `t_aff * heal`
- `t_aff * bloquer`
- `t_aff * fenetre_finale`
- `list * liste_animations`

6.2.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.2.2 Documentation des macros

6.2.2.1 LARGEUR_ENTITE

```
#define LARGEUR_ENTITE 48
```

La largeur d'une frame d'un sprite d'une entité

6.2.2.2 LONGUEUR_ENTITE

```
#define LONGUEUR_ENTITE 48
```

La longueur d'une frame d'un sprite d'une entité

6.2.2.3 N_T_ATTAQUE

```
#define N_T_ATTAQUE "ressources/sprite/attaque.bmp"
```

La texture du personnage qui attaque

6.2.2.4 N_T_ATTAQUE2

```
#define N_T_ATTAQUE2 "ressources/sprite/attaque_green.bmp"
```

La texture du personnage qui attaque

6.2.2.5 N_T_ATTAQUE_CHARGEE

```
#define N_T_ATTAQUE_CHARGEE "ressources/sprite/attaque_chargee.bmp"
```

La texture du personnage qui fait son attaque chargée

6.2.2.6 N_T_ATTAQUE_CHARGEE2

```
#define N_T_ATTAQUE_CHARGEE2 "ressources/sprite/attaque_chargee_green.bmp"
```

La texture du personnage qui fait son attaque chargée

6.2.2.7 N_T_CHARGER

```
#define N_T_CHARGER "ressources/sprite/charger.bmp"
```

La texture du personnage qui charge son attaque chargée

6.2.2.8 N_T_CHARGER2

```
#define N_T_CHARGER2 "ressources/sprite/charger_green.bmp"
```

La texture du personnage qui charge son attaque chargée

6.2.2.9 N_T_MARCHER

```
#define N_T_MARCHER "ressources/sprite/marcher.bmp"
```

La texture du personnage qui marche

6.2.2.10 N_T_MARCHER2

```
#define N_T_MARCHER2 "ressources/sprite/marcher_green.bmp"
```

La texture du personnage qui marche

6.2.2.11 N_T_MARCHER_BOUCLIER

```
#define N_T_MARCHER_BOUCLIER "ressources/sprite/marcher_bouclier.bmp"
```

La texture du personnage qui marche avec son bouclier équipé

6.2.2.12 N_T_MARCHER_BOUCLIER2

```
#define N_T_MARCHER_BOUCLIER2 "ressources/sprite/marcher_bouclier_green.bmp"
```

La texture du personnage qui marche avec son bouclier équipé

6.2.2.13 NB_FPS

```
#define NB_FPS 60
```

Le nombre maximum de FPS

6.2.2.14 NB_SPRITE_JOUEUR

```
#define NB_SPRITE_JOUEUR 5
```

Le nombre de sprites différents du joueur

6.2.3 Documentation du type de l'énumération

6.2.3.1 t_texture_perso

```
enum t_texture_perso
```

Auteur

Antoine Bruneau

Valeurs énumérées

TEXT_MARCHER	La texture du personnage qui marche
TEXT_ATTAQUE	La texture du personnage qui attaque
TEXT_ATTAQUE_CHARGE	La texture du personnage qui fait une attaque chargée
TEXT_CHARGER	La texture du personnage qui charge son attaque
TEXT_MARCHER_BOUCLIER	La texture du personnage qui marche avec son bouclier équipé

6.2.4 Documentation des fonctions

6.2.4.1 afficher_animations()

```
void afficher_animations (
    list * animations )
```

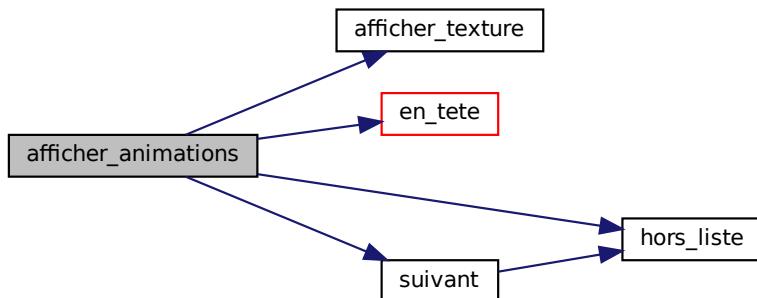
Auteur

Max Descomps

Paramètres

<i>animations</i>	Une liste d'animations
-------------------	------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.2 afficher_buffer()

```
err_t afficher_buffer (
    list * buffer,
    SDL_Renderer * rendu )
```

Auteur

Ange Despert

Les premiers éléments seront en arrière plan et les derniers seront en 1er plan.

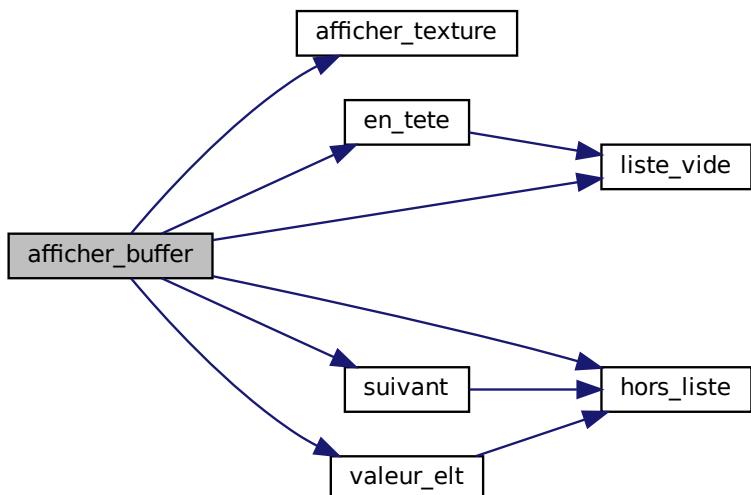
Paramètres

<i>buffer</i>	La liste de textures que l'on veut afficher
<i>rendu</i>	Le rendu sur lequel on veut afficher ces textures

Renvoie

Une valeur différente à 0 lors d'une erreur

Voici le graphe d'appel pour cette fonction :



6.2.4.3 afficher_coffres()

```
void afficher_coffres (
    list * liste_coffre )
```

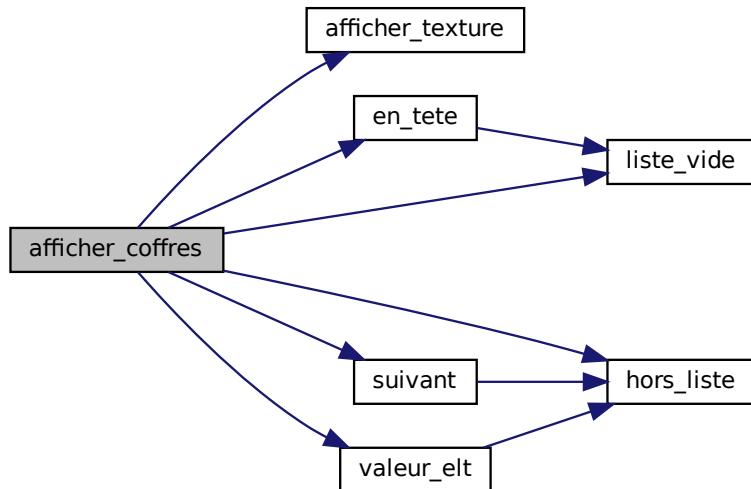
Auteur

Max Descomps

Paramètres

<i>liste_coffre</i>	Une liste de coffres
---------------------	----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



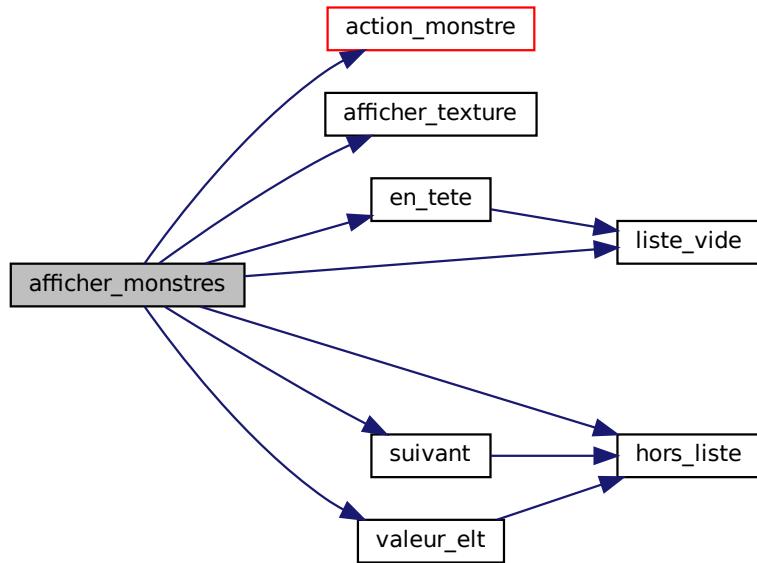
6.2.4.4 afficher_monstres()

```
void afficher_monstres (
    list * liste_monstre,
    joueur_t * joueur )
```

Paramètres

<i>liste_monstre</i>	une liste de monstres
<i>joueur</i>	le joueur qui influe sur les monstres

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



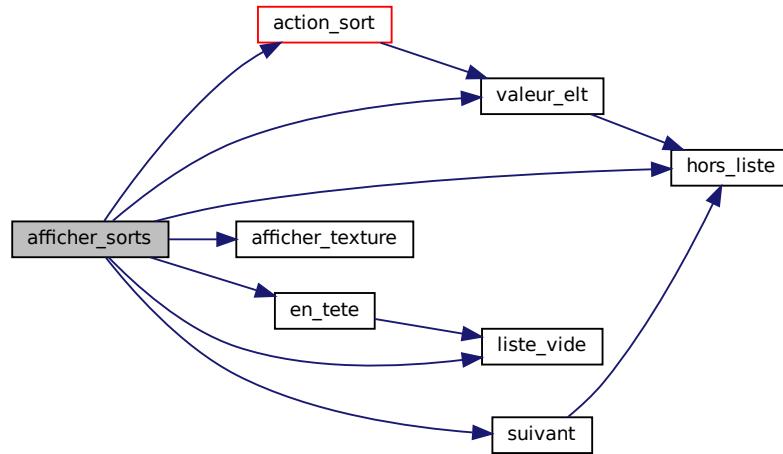
6.2.4.5 afficher_sorts()

```
void afficher_sorts (
    list * liste_sorts,
    joueur_t * joueur )
```

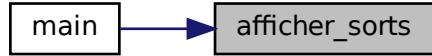
Paramètres

<i>liste_sorts</i>	une liste de sorts
<i>joueur</i>	le joueur qui influe sur les sorts

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.6 afficher_texture()

```
err_t afficher_texture (
    t_aff * texture,
    SDL_Renderer * rendu )
```

Auteur

Ange Despert

Cette fonction permet d'afficher à l'écran une texture utilisant le type personnalisé `t_aff` à l'écran avec le rendu donné en paramètre.

Rien ne sera afficher si la texture donnée en entrée est égale à NULL.

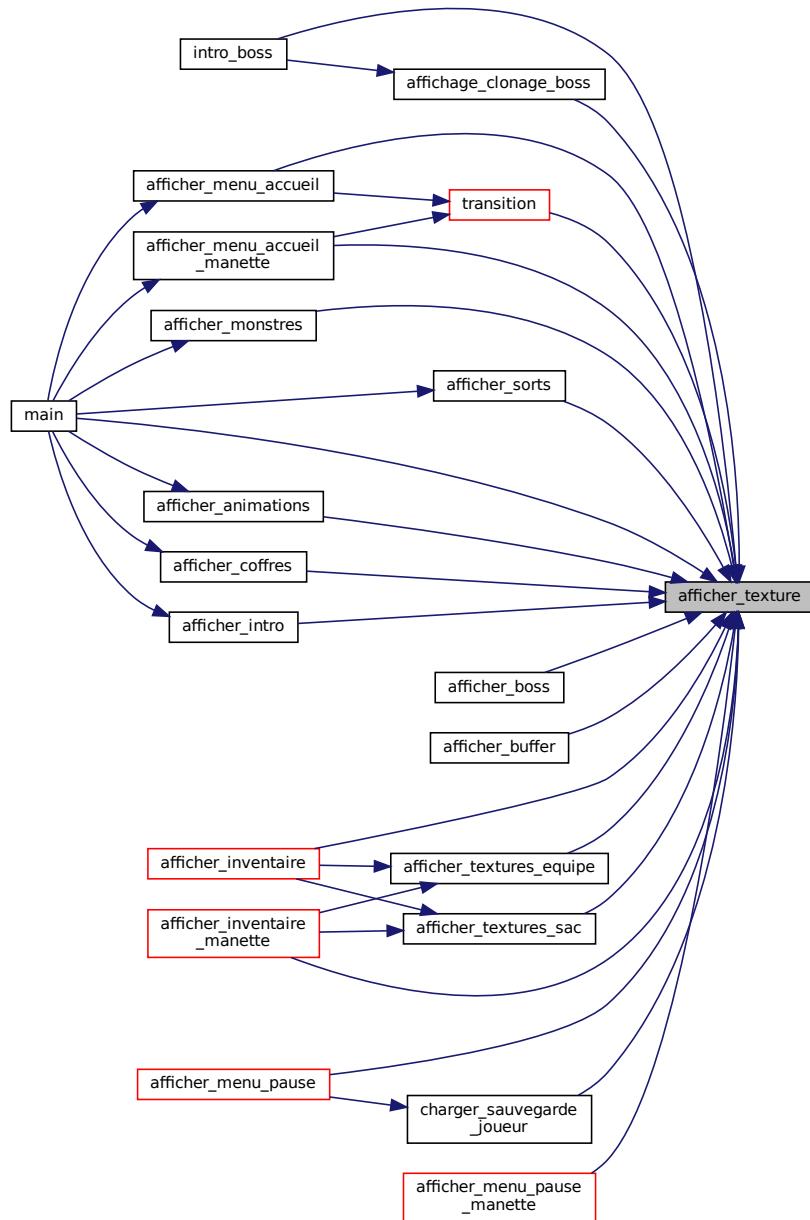
Paramètres

<code>texture</code>	La texture à afficher
<code>rendu</code>	Le rendu sur lequel afficher la texture à l'écran

Renvoie

0 s'il n'y a pas eu d'erreur sinon un entier négatif

Voici le graphe des appels de cette fonction :



6.2.4.7 ajout_text_liste()

```
void* ajout_text_liste (
    void * t )
```

Auteur

Ange Despert

Cette fonction sert juste à rérerencier les textures dans la [liste de textures](#).

Paramètres

<i>t</i>	une texture
----------	-------------

Renvoie

void * l'element à ajouter à la liste

Voici le graphe des appels de cette fonction :



6.2.4.8 color()

```
SDL_Color color (
    Uint8 r,
    Uint8 g,
    Uint8 b,
    Uint8 a )
```

Auteur

Max Descomps

Paramètres

<i>r</i>	niveau de rouge
<i>g</i>	niveau de vert
<i>b</i>	niveau de bleu
<i>a</i>	niveau d'opacité

Renvoie

SDL_Color une couleur SDL

Voici le graphe des appels de cette fonction :



6.2.4.9 créer_texture()

```
t_aff* créer_texture (
    const char * nom_fichier,
    const int taille_t_x,
    const int taille_t_y,
    const int x,
    const int y,
    const float multiplicateur_taille )
```

Auteur

Ange Despert

Fonction qui permet de créer un texture sous forme d'un type personnalisé [t_aff](#) qui contient pleins d'informations sur la texture.

6.2.4.10 Déroulement

Le fichier de la texture est chargé dans une surface.

On va converir cette surface en texture.

On va remplir les informations de la structure [t_aff](#) avec les information en entrée et des informations obtenues via requêtes SDL.

Bien entendu, on testera pour être sur qu'il n'y a aucune erreur.

6.2.4.11 Lors d'une erreur

Contrairement à la plupart des fonctions qui gèrent les textures, celle-ci affichera sûrement un warning à l'écran si l'on ne peut pas créer la texture. Il est donc important de ce rendre compte que cette fonction ne ferme pas le programme lors d'une erreur. C'est donc au développeur de décider si une impossibilité de créer une texture est une raison de fermer le programme.

6.2.4.12 A noter

Si l'on donne -1 et -1 pour les paramètres taille_t_x et taille_t_y le rectangle `frame_anim` de la structure `t_aff` sera NULL.

Il est donc important de prendre cela en compte pour ne pas causer des erreurs de segmentation.

Donner NULL en nom de texture ne la créera pas mais créera tout les autres attributs de la structure `t_aff`.

Donner 0 en multiplicateur de taille agrandira la texture pour qu'elle face la taille de l'écran.

Paramètres

<i>nom_fichier</i>	Le nom du fichier contenant la texture
<i>taille_t_x</i>	La longueur de la texture à montrer
<i>taille_t_y</i>	La largeur de la texture à montrer
<i>x</i>	La coordonnée x où afficher la texture à l'écran
<i>y</i>	La coordonnée y où afficher la texture à l'écran
<i>multiplicateur_taille</i>	Une valeur par laquelle multiplier la taille de la texture

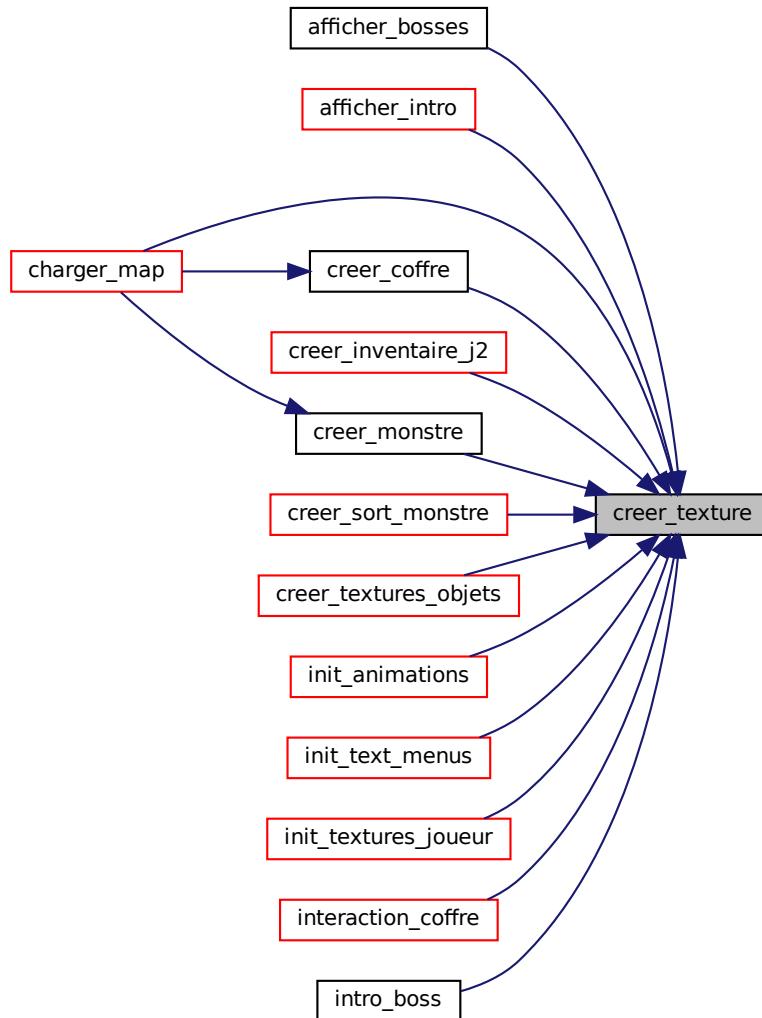
Renvoie

*t_aff** Une structure qui permet l'affichage de la texture à l'écran ou NULL s'il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.13 current_frame_x()

```
int current_frame_x (
    t_aff * texture )
```

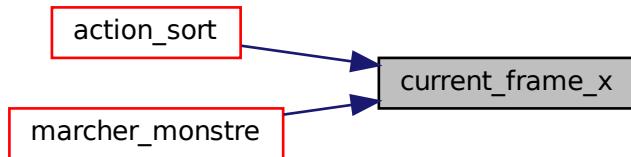
Paramètres

<i>texture</i>	une texture
----------------	-------------

Renvoie

un entier correspondant à l'indice

Voici le graphe des appelants de cette fonction :



6.2.4.14 current_frame_y()

```
int current_frame_y (  
    t_aff * texture )
```

Paramètres

<i>texture</i>	une texture
----------------	-------------

Renvoie

un entier correspondant à l'indice

Voici le graphe des appelants de cette fonction :



6.2.4.15 def_texture_taille()

```
void def_texture_taille (   
    t_aff * a_modifier,  
    const int longueur,  
    const int largeur )
```

Auteur

Ange Despert

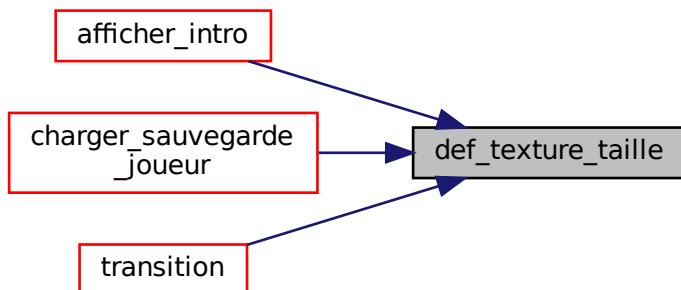
Obsolète L'utilisation de cette fonction n'a plus trop de sens étant donné que le moteur gère automatiquement la taille des textures.

Fonction qui permet de modifier le rectangle de la structure `t_aff : aff_fenetre` avec de nouvelles valeurs.

Paramètres

<code>a_modifie</code>	La texture à modifier
<code>longueur</code>	La nouvelle longueur en pixel à appliquer
<code>largeur</code>	La nouvelle largeur en pixel à appliquer

Voici le graphe des appelants de cette fonction :



6.2.4.16 `deplacement_x_entite()`

```
_Bool deplacement_x_entite (
    t_map * m,
    t_aff * texture,
    int x,
    SDL_Rect * r )
```

Auteur

Ange Despert

Cette fonction permet à une entité de se déplacer sur l'axe x.

Cette fonction gère les collisions et empêchera l'entité de sortir des limites de la map.

Cette dernière prend également les collisions définies dans la liste des collisions de la map : [liste_collisions](#).

Il est à noté que pour éviter les déplacement trop rapides la fonction utilise l'entier `duree_frame_anim` qui permet d'empêcher le déplacement tout les x frames.

On peut récupérer l'élément que l'entité a touché en regardant l'élément courant de la [liste de collisions](#) de la map.

Paramètres

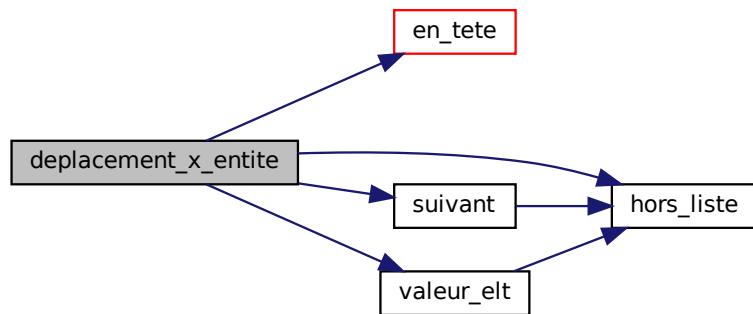
<i>m</i>	La map actuelle
<i>texture</i>	La texture de l'entité que l'on veut bouger
<i>x</i>	La nouvelle coordonnée du rectangle de l'entité
<i>r</i>	Le rectangle représentant la zone de collision de l'entité

Renvoie

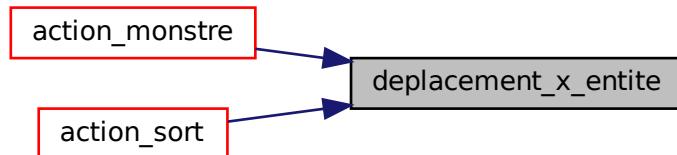
vrai : Si l'entité a réussi à se déplacer

faux : Si l'entité n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.17 deplacement_x_joueur_secondaire()

```

_Bool deplacement_x_joueur_secondaire (
    t_map * map,
    joueur_t * joueur,
    int x,
    SDL_Rect * r,
    lobjet_t * objets )
  
```

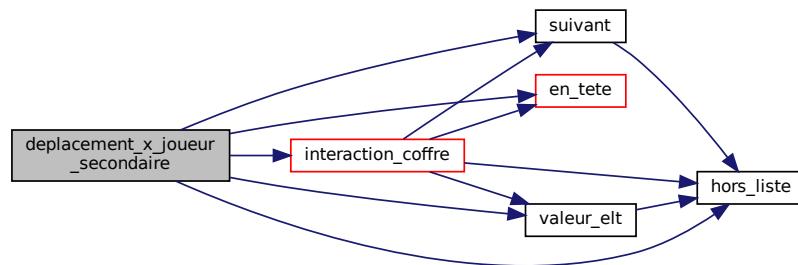
Paramètres

<i>map</i>	La map sur laquelle le personnage se déplace
<i>joueur</i>	Le joueur qui se déplace
<i>x</i>	Le nombre d'unités de déplacements
<i>r</i>	La zone de collision du joueur
<i>objets</i>	Les objets du jeu

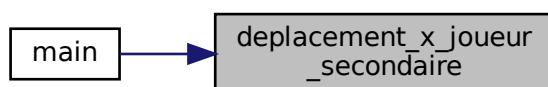
Renvoie

vrai : Si le joueur a réussi à se déplacer
faux : Si le joueur n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.18 deplacement_x_pers()

```

_Bool deplacement_x_pers (
    t_map * map,
    joueur_t ** joueurs,
    unsigned short int nb_joueurs,
    int x,
    lobjet_t * objets )

```

Auteur

Ange Despert

Cette fonction utilise un rectangle `tx` pour savoir quand elle doit bouger le personnage ou bien déplacer la camera. Elle prendra en compte les collisions de la [liste de collisions de la map](#). Elle empêche le personnage de sortir des bordures de la map.

Paramètres

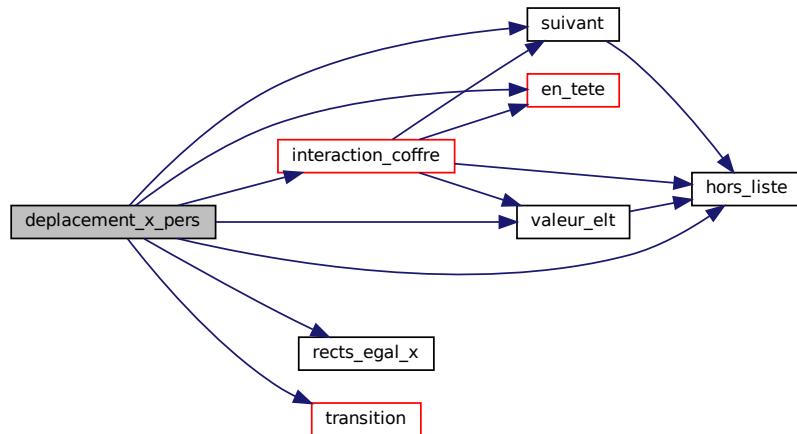
<code>map</code>	La map sur laquelle le personnage se déplace
<code>joueurs</code>	Les joueurs en jeu
<code>nb_joueurs</code>	Le nombre de joueurs en jeu
<code>x</code>	Le nombre d'unités de déplacements
<code>objets</code>	Les objets du jeu

Renvoie

vrai : Si le joueur s'est téléporté

faux : Si le joueur ne s'est pas téléporté <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.19 deplacement_y_entite()

```
_Bool deplacement_y_entite (
    t_map * m,
    t_aff * texture,
    int y,
    SDL_Rect * r )
```

Auteur

Ange Despert

Cette fonction permet à une entité de se déplacer sur l'axe y.

Cette fonction gère les collisions et empêchera l'entité de sortir des limites de la map.

Cette dernière prend également les collisions définies dans la liste des collisions de la map : [liste_collisions](#).

Il est à noté que pour éviter les déplacement trop rapides la fonction utilise l'entier [duree_frame_anim](#) qui permet d'empêcher le déplacement tout les x frames.

On peut récupérer l'élément que l'entité a touché en regardant l'élément courant de la [liste de collisions](#) de la map.

Paramètres

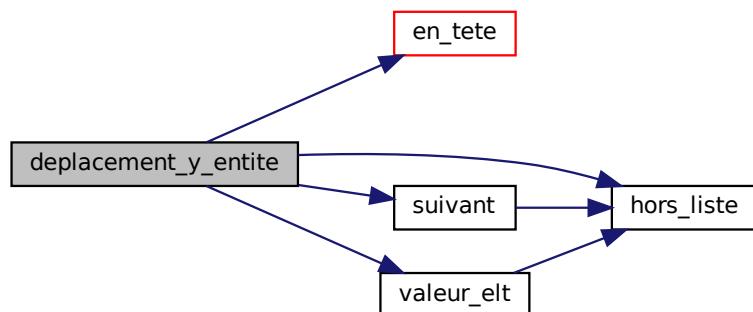
<i>m</i>	La map actuelle
<i>texture</i>	La texture de l'entité que l'on veut bouger
<i>y</i>	La nouvelle coordonnée du rectangle de l'entité
<i>r</i>	Le rectangle représentant la zone de collision de l'entité

Renvoie

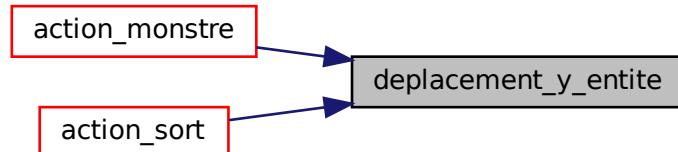
vrai : Si l'entité a réussi à se déplacer

faux : Si l'entité n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.20 deplacement_y_joueur_secondaire()

```
_Bool deplacement_y_joueur_secondaire (
    t_map * map,
    joueur_t * joueur,
    int y,
    SDL_Rect * r,
    lobjet_t * objets )
```

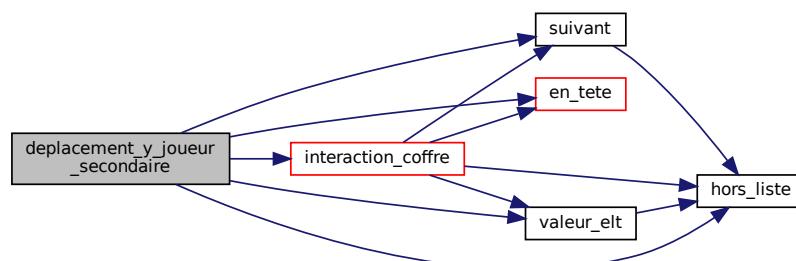
Paramètres

<i>map</i>	La map sur laquelle le personnage se déplace
<i>joueur</i>	Le joueur qui se déplace
<i>y</i>	Le nombre d'unités de déplacements
<i>r</i>	La zone de collision du joueur
<i>objets</i>	Les objets du jeu

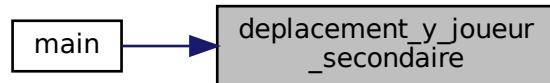
Renvoie

vrai : Si le joueur a réussi à se déplacer
faux : Si le joueur n'a pas pu se déplacer <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.21 `deplacement_y_pers()`

```
_Bool deplacement_y_pers (
    t_map * map,
    joueur_t ** joueurs,
    unsigned short int nb_joueurs,
    int y,
    lobjet_t * objets )
```

Auteur

Ange Despert

Cette fonction utilise un rectangle `ty` pour savoir quand elle doit bouger le personnage ou bien déplacer la camera. Elle prendra en compte les collisions de la [liste de collisions de la map](#).
Elle empêche le personnage de sortir des bordures de la map.

Paramètres

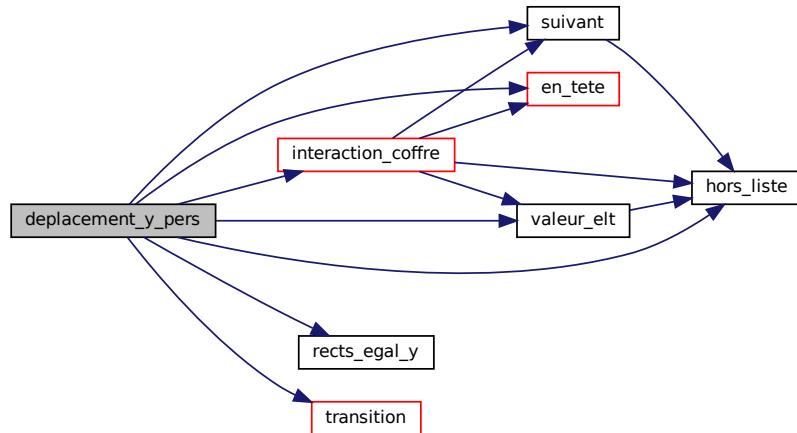
<code>map</code>	La map sur laquelle le personnage se déplace
<code>joueurs</code>	Les joueurs en jeu
<code>nb_joueurs</code>	Le nombre de joueurs en jeu
<code>y</code>	Le nombre d'unités de déplacements
<code>objets</code>	Les objets du jeu

Renvoie

vrai : Si le joueur s'est téléporté

faux : Si le joueur ne s'est pas téléporté <Définition du type booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.22 deplacer_rect_haut_droit()

```

void deplacer_rect_haut_droit (
    SDL_Rect * r,
    int x,
    int y )
  
```

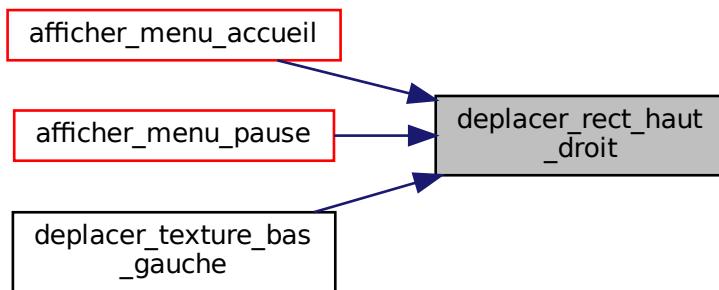
Auteur

Ange Despert

Paramètres

<i>r</i>	Le rectangle à placer
<i>x</i>	La coordonnée x du rectangle depuis la nouvelle origine.
<i>y</i>	La coordonnée y du rectangle depuis la nouvelle origine.

Voici le graphe des appelants de cette fonction :

**6.2.4.23 deplacer_rect_origine()**

```
void deplacer_rect_origine (
    SDL_Rect * r,
    int x,
    int y )
```

Auteur

Ange Despert

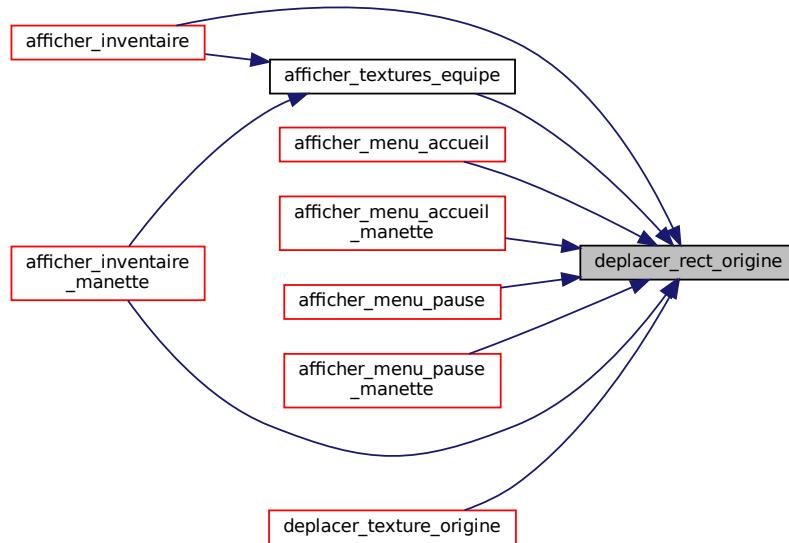
Paramètres

<i>r</i>	Le rectangle à déplacer
<i>x</i>	La position horizontale du rectangle
<i>y</i>	La position verticale du rectangle

Renvoie

Une valeur différente à 0 lors d'une erreur

Voici le graphe des appels de cette fonction :



6.2.4.24 `deplacer_texture_bas_droit()`

```
void deplacer_texture_bas_droit (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	Coordonnée x du coin supérieur gauche de la texture.
<i>y</i>	Coordonnée y du coin supérieur gauche de la texture.

6.2.4.25 `deplacer_texture_bas_gauche()`

```
void deplacer_texture_bas_gauche (
```

```
t_aff * texture,
int x,
int y )
```

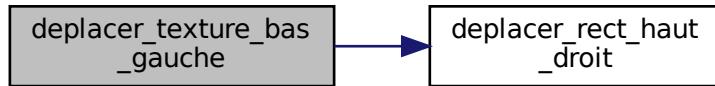
Auteur

Ange Despert

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	La coordonnée x du coin supérieur gauche de la texture.
<i>y</i>	Coordonnée y du coin supérieur gauche de la texture.

Voici le graphe d'appel pour cette fonction :

**6.2.4.26 deplacer_texture_centre()**

```
void deplacer_texture_centre (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.

Utilisez [place_rect_center_from_point](#) à l'aide du rectangle [frame_anim](#) de la variable [fenetre_finale](#).

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	La coordonnée x du centre de la texture.
<i>y</i>	La coordonnée y du centre de la texture.

Voici le graphe d'appel pour cette fonction :



6.2.4.27 `deplacer_texture_haut_droit()`

```
void deplacer_texture_haut_droit (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Paramètres

<i>texture</i>	la texture à déplacer
<i>x</i>	La coordonnée x du coin supérieur gauche de la texture.
<i>y</i>	Coordonnée y du coin supérieur gauche du rectangle.

6.2.4.28 `deplacer_texture_origine()`

```
void deplacer_texture_origine (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Ange Despert

Paramètres

<i>texture</i>	La texture à déplacer
<i>x</i>	La coordonnée x de l'origine de la texture.
<i>y</i>	La coordonnée y de l'origine de la texture.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



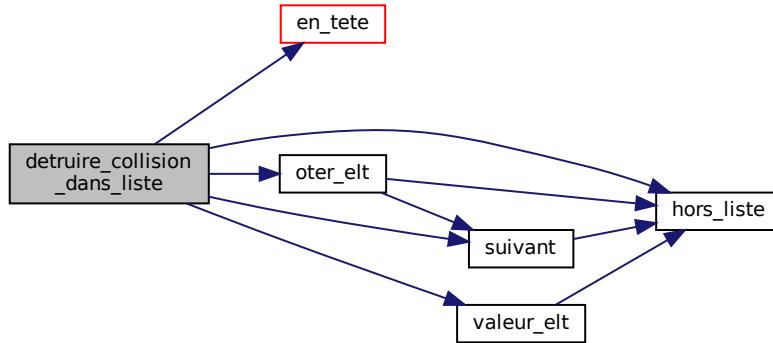
6.2.4.29 detruire_collision_dans_liste()

```
void detruire_collision_dans_liste (
    list * liste_collisions,
    SDL_Rect * collision )
```

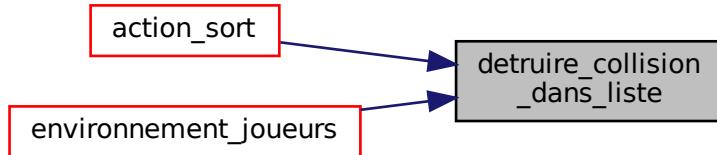
Paramètres

<i>liste_collisions</i>	Une liste de collisions
<i>collision</i>	La collision à détruire dans la liste

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.30 detruire_liste_textures()

```
void detruire_liste_textures (
    t_l_aff ** l_texture )
```

Auteur

Ange Despert

Paramètres

<i>l_texture</i>	L'adresse de la liste de texture à détruire
------------------	---

6.2.4.31 detruire_texture()

```
void detruire_texture (
    t_aff ** texture )
```

La liste des textures créées

Auteur

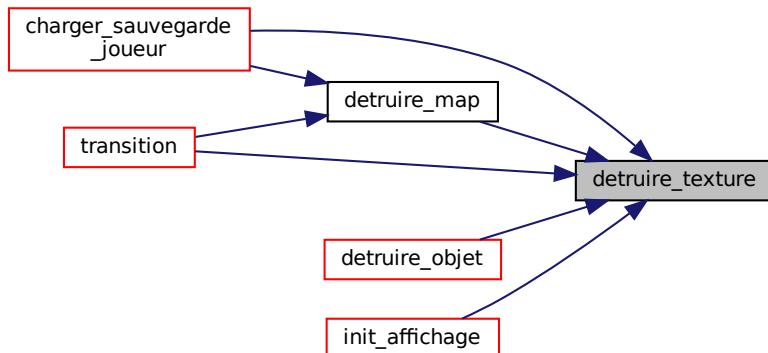
Despert Ange

Libère la mémoire allouée à la texture et la mets à NULL.

Paramètres

<i>texture</i>	L'adresse du pointeur sur la structure à détruire
----------------	---

Voici le graphe des appels de cette fonction :



6.2.4.32 get_rect_center()

```
SDL_Point get_rect_center (
    const SDL_Rect *const r )
```

Cette fonction contrairement à la fonction [get_rect_center_coord](#) donne les coordonnées strictes du centre.

Auteur

Ange Despert

Paramètres

<i>r</i>	Le rectangle dont on veut les coordonnées du milieu
----------	---

Renvoie

Les coordonnées du milieu du rectangle

Voici le graphe des appelants de cette fonction :



6.2.4.33 get_rect_center_coord()

```
SDL_Point get_rect_center_coord (
    const SDL_Rect *const r )
```

Cette fonction contrairement à la fonction [get_rect_center](#) donne les coordonnées du centre en prenant en compte les coordonnées actuelles du rectangle.

Cette fonction est donc faite pour être utilisée conjointement à la fonction [place_rect_center_from_point\(SDL_Rect *r, SDL_Point p\)](#)

Auteur

Ange Despert

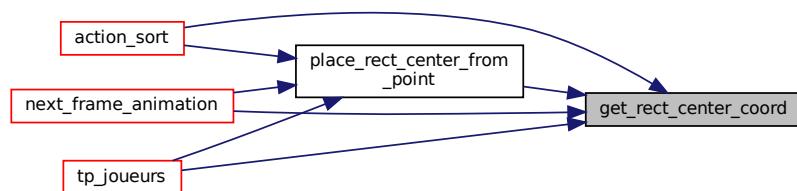
Paramètres

<i>r</i>	Le rectangle dont on veut les coordonnées du milieu.
----------	--

Renvoie

Les coordonnées du milieu du rectangle.

Voici le graphe des appelants de cette fonction :



6.2.4.34 info_texture()

```
void info_texture (
    t_aff * texture )
```

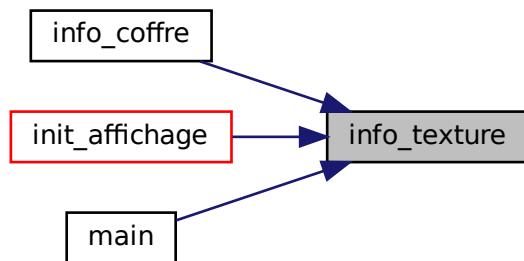
Auteur

Max Descomps

Paramètres

<i>texture</i>	la texture sur laquelle on se renseigne
----------------	---

Voici le graphe des appelants de cette fonction :



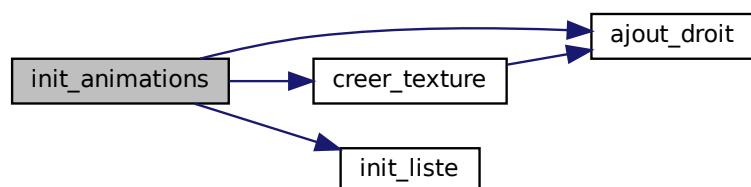
6.2.4.35 init_animations()

```
void init_animations ( )
```

Auteur

Max Descomps

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.36 init_texture_joueur()

```
t_aff* init_texture_joueur (
    t_l_aff * textures_joueur,
    joueur_t * joueur )
```

Auteur

Antoine Bruneau

Paramètres

<i>textures_joueur</i>	Liste de textures personnage
<i>joueur</i>	Joueur dont on veut la texture de départ

Renvoie

*t_aff** Une textures personnage

6.2.4.37 init_textures_joueur()

```
t_l_aff* init_textures_joueur (
    joueur_t * j,
    int num_j )
```

Auteur

Antoine Bruneau

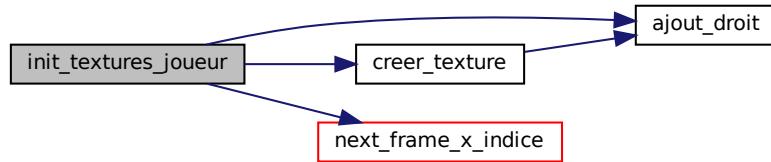
Paramètres

<i>j</i>	Le joueur dont on initialise les textures
<i>num_j</i>	Indice du joueur dans le tableau des joueurs

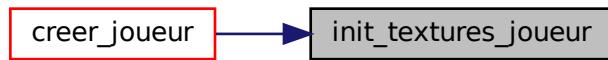
Renvoie

`t_l_aff*` Une liste de textures

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.38 lister_animations()

```
void lister_animations (
    joueur_t ** joueurs,
    list * animations )
```

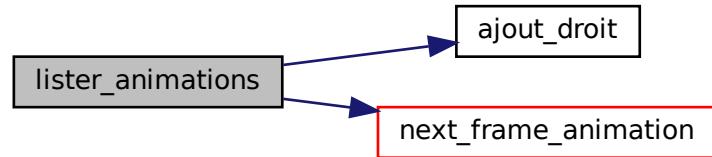
Auteur

Max Descomps

Paramètres

<code>joueurs</code>	Les joueurs sur lesquels placer les animation
<code>animations</code>	Liste regroupant les animations trouvées

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.39 modif_affichage_rect()

```
void modif_affichage_rect (
    t_aff * texture,
    SDL_Rect r )
```

Auteur

Ange Despert

Paramètres

<code>texture</code>	La texture à modifier.
<code>r</code>	Le rectangle à appliquer.

6.2.4.40 next_frame_animation()

```
t_aff* next_frame_animation (
    joueur_t * joueur )
```

Auteur

Max Descomps

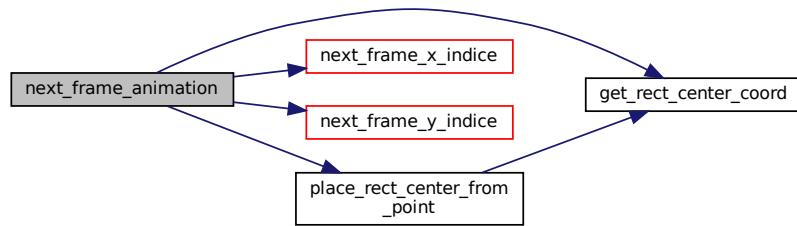
Paramètres

<code>joueur</code>	Le joueur sur lequel placer l'animation
---------------------	---

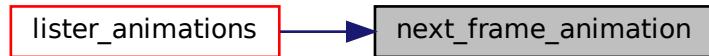
Renvoie

La texture de l'animation

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

**6.2.4.41 next_frame_indice()**

```

err_t next_frame_indice (
    t_aff * texture,
    const unsigned int x,
    const unsigned int y )
  
```

Auteur

Despert Ange

Fonction qui permet de choisir le sprite à un certain indice de l'image.

Cette fonction est conçue pour être utiliser avec une texture qui contient plusieurs sprites de même taille.

La fonction utilisera la taille du `rectangle` charger d'afficher qu'une partie d'une image.

Cette fonction est là pour éviter deux appels de fonctions, en l'occurrence `next_frame_x_indice` et `next_frame_y_indice`

Paramètres

<i>texture</i>	une texture joueur
<i>x</i>	qui correspond au n-ème sprite sur l'axe des x ou l'on souhaite positionner la texture
<i>y</i>	qui correspond au n-ème sprite sur l'axe des y ou l'on souhaite positionner la texture

Renvoie

err_t un entier pour savoir si il y a eu une erreur

Voici le graphe d'appel pour cette fonction :

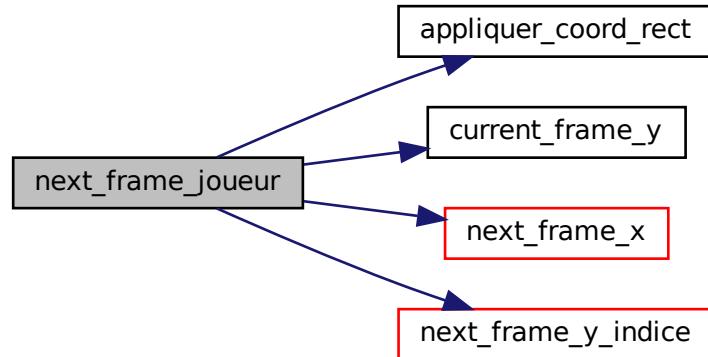


Voici le graphe des appelants de cette fonction :

**6.2.4.42 next_frame_joueur()**

```
t_aff* next_frame_joueur (
    joueur_t * j )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.43 next_frame_x()

```
void next_frame_x (
    t_aff * texture )
```

Auteur

Ange Despert

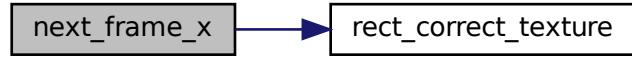
Fonction qui passe au sprite suivant dans l'axe x. Cette fonction fera une boucle complète de l'image, il n'y a donc pas à s'inquiéter d'arriver à la fin de l'image.

Comme les fonction à indice, cette fonction fait appel au rectangle [frame_anim](#) de la structure [d'affichage](#).

Paramètres

<i>texture*</i>	une texture joueur
-----------------	--------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.44 next_frame_x_indice()

```
err_t next_frame_x_indice (
    t_aff * texture,
    const unsigned int indice )
```

Auteur

Ange Despert

Fonction qui permet de choisir le sprite à un certain indice de l'image.

Cette fonction est conçue pour être utilisée avec une texture qui contient plusieurs sprites de même taille.

La fonction utilisera la taille du [rectangle](#) chargé d'afficher qu'une partie d'une image.

Paramètres

<i>texture*</i>	une texture joueur
<i>indice</i>	unsigned int qui correspond au n-ème sprite sur l'axe des x ou l'on souhaite positionner la texture

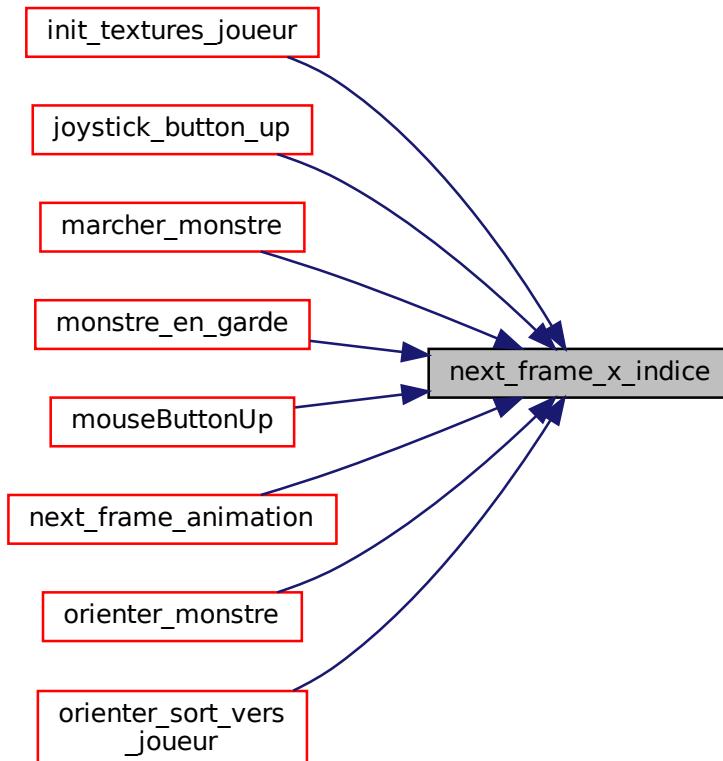
Renvoie

`err_t` un entier pour savoir si il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.45 `next_frame_y()`

```
void next_frame_y (
    t_aff * texture )
```

Auteur

Ange Despert

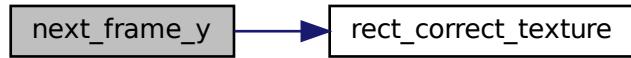
Fonction qui passe au sprite suivant dans l'axe y. Cette fonction fera une boucle complète de l'image, il n'y a donc pas à s'inquiéter d'arriver à la fin de l'image.

Comme les fonction à indice, cette fonction fait appel au rectangle `frame_anim` de la structure [d'affichage](#).

Paramètres

<i>texture*</i>	une texture joueur
-----------------	--------------------

Voici le graphe d'appel pour cette fonction :

**6.2.4.46 next_frame_y_indice()**

```
err_t next_frame_y_indice (
    t_aff * texture,
    const unsigned int indice )
```

Paramètres

<i>texture</i>	Une texture joueur
<i>indice</i>	Correspond au n-ème sprite sur l'axe des y ou l'on souhaite positionner la texture

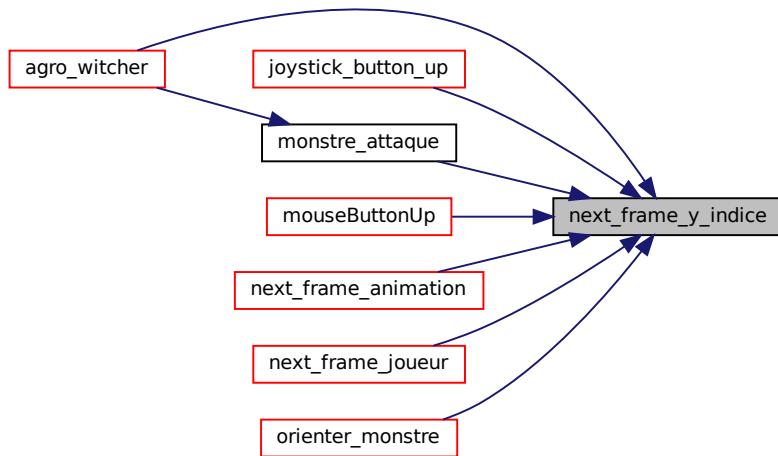
Renvoie

`err_t` un entier pour savoir si il y a eu une erreur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.47 place_rect_center_from_point()

```
void place_rect_center_from_point (
    SDL_Rect * r,
    SDL_Point p )
```

Le but de cette fonction est de pouvoir placer plusieurs rectangles au même endroit peut importe leur taille. C'est fonction est donc très puissante si elle est utilisée conjointement à la fonction [get_rect_center_coord\(const SDL_Rect *const r\)](#)

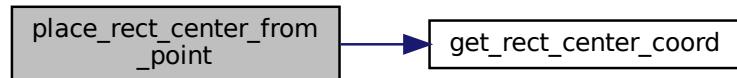
Auteur

Ange Despert

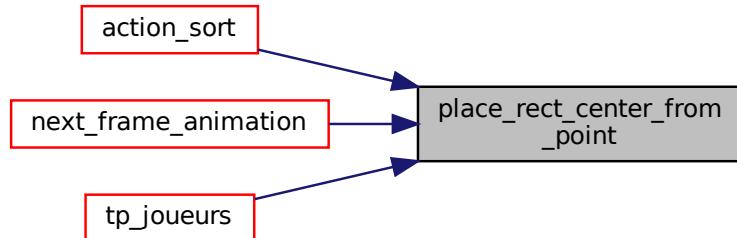
Paramètres

<i>r</i>	Le rectangle que l'on veut déplacer
<i>p</i>	Le point où on veut placer le milieu du rectangle

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.48 placer_texture()

```
void placer_texture (
    t_aff * texture,
    int x,
    int y )
```

Auteur

Max Descomps

Paramètres

<i>texture</i>	Texture à placer
<i>x</i>	Position horizontale
<i>y</i>	Position verticale

Voici le graphe des appelants de cette fonction :



6.2.4.49 rect_centre()

```
void rect_centre (
    SDL_Rect * rectangle )
```

Auteur

Ange Despert

Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.
Utilisez [place_rect_center_from_point](#) à l'aide du rectangle `frame_anim` de la variable `fenetre_finale`.

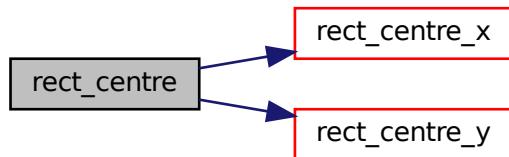
Paramètres

<i>rectangle</i>	Le rectangle à placer
------------------	-----------------------

Renvoie

Un booléen

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.2.4.50 rect_centre_rect()

```
void rect_centre_rect (
    SDL_Rect * rectangle,
    SDL_Rect * rectangle_centre )
```

Auteur

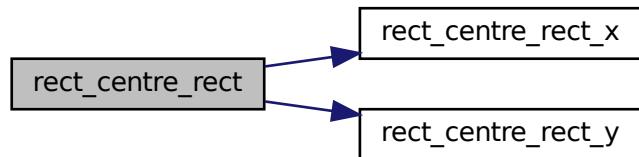
Ange Despert

Fonctionnement similaire à l'utilisation conjointe des fonctions [place_rect_center_from_point](#) et [get_rect_center_coord](#)

Paramètres

<i>rectangle</i>	Le rectangle à centrer
<i>rectangle_centre</i>	Le rectangle dans lequel en centre un autre

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.51 rect_centre_rect_x()

```
void rect_centre_rect_x (
    SDL_Rect * rectangle,
    SDL_Rect * rectangle_centre )
```

Auteur

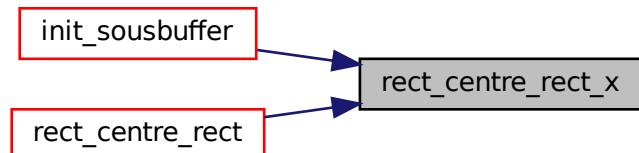
Ange Despert

Fonction qui fonctionne de la même manière que la fonction [rect_centre_rect](#) mais permet de seulement centrer l'axe x.

Paramètres

<i>rectangle</i>	Le rectangle à centrer
<i>rectangle_centre</i>	Le rectangle dans lequel en centre un autre

Voici le graphe des appelants de cette fonction :



6.2.4.52 rect_centre_rect_y()

```
void rect_centre_rect_y (
```

```
SDL_Rect * rectangle,
SDL_Rect * rectangle_centre )
```

Auteur

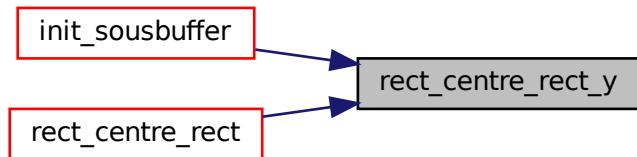
Ange Despert

Fonction qui fonctionne de la même manière que la fonction [rect_centre_rect](#) mais permet de seulement centrer l'axe y.

Paramètres

<i>rectangle</i>	Le rectangle à centrer
<i>rectangle_centre</i>	Le rectangle dans lequel en centre un autre

Voici le graphe des appels de cette fonction :

**6.2.4.53 rect_centre_x()**

```
void rect_centre_x (
    SDL_Rect * rectangle )
```

Auteur

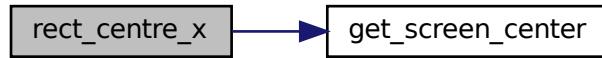
Ange Despert

Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.
Utilisez [place_rect_center_from_point](#) à l'aide du rectangle [frame_anim](#) de la variable [fenetre_finale](#).

Paramètres

<i>rectangle</i>	Le rectangle à placer
------------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.54 rect_centre_y()

```
void rect_centre_y (  
    SDL_Rect * rectangle )
```

Auteur

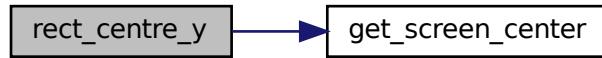
Ange Despert

Obsolète Cette fonction ne devrait plus être appelée car elle aurait un comportement imprévisible.
Utilisez [place_rect_center_from_point](#) à l'aide du rectangle [frame_anim](#) de la variable [fenetre_finale](#).

Paramètres

<i>rectangle</i>	Le rectangle à placer
------------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.2.4.55 rect_correct_texture()

```
_Bool rect_correct_texture (
    const SDL_Rect *const to_verify,
    const int width,
    const int height )
```

Auteur

Ange Despert

Cette fonction est appellée par les fonction de déplacement de frame tel que [next_frame_x](#) pour s'assurer qu'il n'y a aucune erreur.

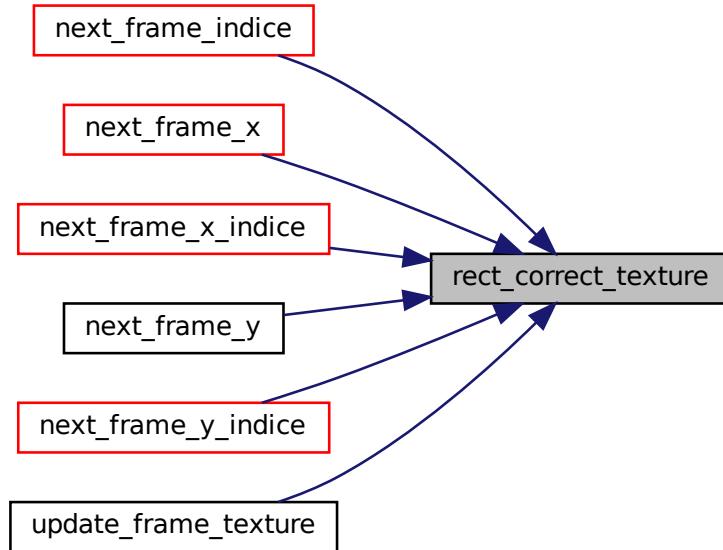
Paramètres

<i>to_verify</i>	La structure <code>SDL_Rect</code> à vérifier
<i>width</i>	La longueur de la texture
<i>height</i>	La largeur de la texture

Renvoie

Un booléen

Voici le graphe des appels de cette fonction :



6.2.4.56 rects_egal_x()

```
_Bool rects_egal_x (
    const SDL_Rect *const r1,
    SDL_Rect const *const r2 )
```

Auteur

Ange Despert

Cette fonction permet de savoir si deux rectangles sont égaux mais en prenant seulement en compte l'axe x.
Cette fonction est utilisée par la fonction [la fonction de déplacement de personnage](#).

Paramètres

<i>r1</i>	Le premier rectangle à comparer
<i>r2</i>	Le deuxième rectangle à comparer

Renvoie

Un booléen <Définition du type booléen

Voici le graphe des appelants de cette fonction :



6.2.4.57 rects_egal_y()

```
_Bool rects_egal_y (
    const SDL_Rect *const r1,
    SDL_Rect const *const r2 )
```

Auteur

Ange Despert

Cette fonction permet de savoir si deux rectangles sont égaux mais en prenant seulement en compte l'axe y. Cette fonction est utilisée par la fonction [la fonction de déplacement de personnage](#).

Paramètres

r1	Le premier rectangle à comparer
r2	Le deuxième rectangle à comparer

Renvoie

Un booléen <Définition du type booléen

Voici le graphe des appelants de cette fonction :



6.2.4.58 `text_copier_position()`

```
void text_copier_position (
    t_aff * a_modifier,
    const t_aff *const original )
```

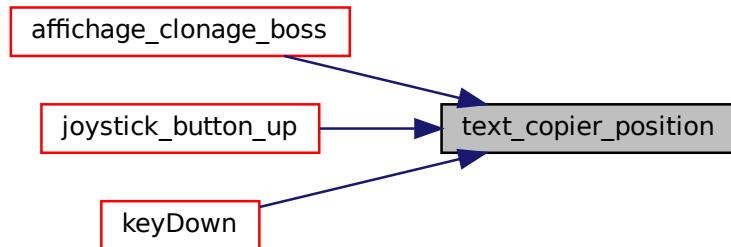
Auteur

Max Descomps

Paramètres

<i>a_modifier</i>	La texture dont on veut modifier la position
<i>original</i>	La texture dont on veut copier la position

Voici le graphe des appels de cette fonction :



6.2.5 Documentation des variables

6.2.5.1 `bloquer`

```
t_aff* bloquer
```

La texture de l'animation de blocage

6.2.5.2 `compteur`

```
long int compteur
```

Un compteur d'ips qui va de 0 à `NB_FPS`

6.2.5.3 fenetre_finale

```
t_aff* fenetre_finale
```

La texture de la fenêtre de jeu finale sans l'interface

6.2.5.4 heal

```
t_aff* heal
```

La texture de l'animation de heal

6.2.5.5 liste_animations

```
list* liste_animations
```

La liste des animation à jouer

6.2.5.6 listeDeTextures

```
list* listeDeTextures
```

La liste de toutes les textures qui ont été créées. Sert à leur destruction

6.2.5.7 multiplicateur_x

```
float multiplicateur_x
```

Multiplicateur qui dépend de la résolution (longueur), vaut 1 pour 1920

6.2.5.8 multiplicateur_y

```
float multiplicateur_y
```

Multiplicateur qui dépend de la résolution (largeur), vaut 1 pour 1080

6.2.5.9 tx

```
SDL_Rect tx
```

Rectangle servant au déplacement du personnage principal en x

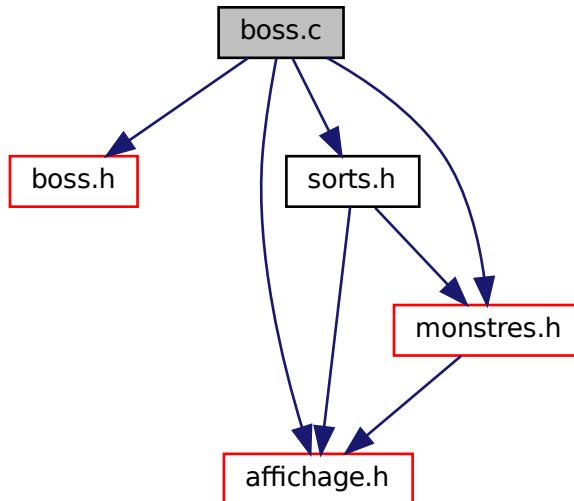
6.2.5.10 ty

```
SDL_Rect ty
```

Rectangle servant au déplacement du personnage principal en y

6.3 Référence du fichier boss.c

Graphe des dépendances par inclusion de boss.c:



Fonctions

- void `deplacement_boss_aleatoire (boss_t *boss)`
- boss_t * `creer_boss_clone (boss_t *boss, t_aff *texture)`
- void `affichage_chargement_attaque_boss (t_aff *boule_1, t_aff *boule_2, int phase)`
 animation lancement d'un sort
- void `affichage_clonage_boss (t_aff *boss, t_aff *clone_1, t_aff *clone_2, int phase)`
- void `intro_boss (boss_t *boss, boss_t *clone_1, boss_t *clone_2)`
- void `afficher_boss (boss_t *boss)`
- void `afficher_bosses (boss_t *boss[3])`

6.3.1 Documentation des fonctions

6.3.1.1 `affichage_chargement_attaque_boss()`

```

void affichage_chargement_attaque_boss (
    t_aff * boule_1,
    t_aff * boule_2,
    int phase )
    
```

Auteur

Bruneau Antoine

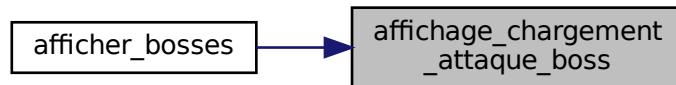
Paramètres

<i>sort</i>	le sort à détruire
-------------	--------------------

Voici le graphe d'appel pour cette fonction :



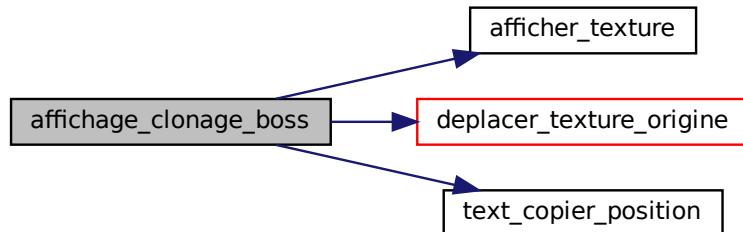
Voici le graphe des appelants de cette fonction :



6.3.1.2 affichage_clonage_boss()

```
void affichage_clonage_boss (
    t_aff * boss,
    t_aff * clone_1,
    t_aff * clone_2,
    int phase )
```

Voici le graphe d'appel pour cette fonction :



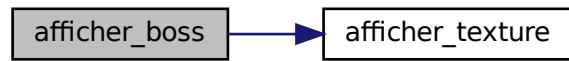
Voici le graphe des appelants de cette fonction :



6.3.1.3 afficher_boss()

```
void afficher_boss ( boss_t * boss )
```

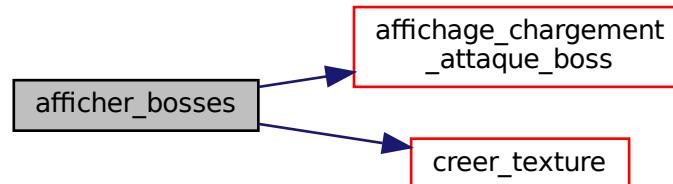
Voici le graphe d'appel pour cette fonction :



6.3.1.4 afficher_bosses()

```
void afficher_bosses ( boss_t * boss[3] )
```

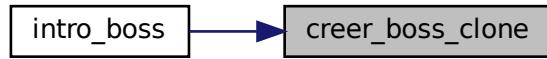
Voici le graphe d'appel pour cette fonction :



6.3.1.5 creer_boss_clone()

```
boss_t* creer_boss_clone (
    boss_t * boss,
    t_aff * texture )
```

Voici le graphe des appels de cette fonction :



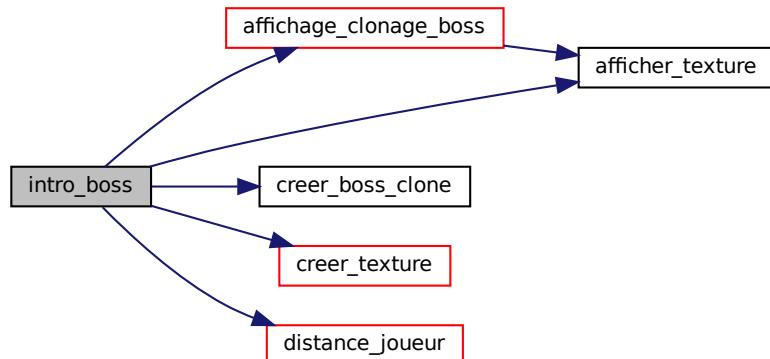
6.3.1.6 deplacement_boss_aleatoire()

```
void deplacement_boss_aleatoire (
    boss_t * boss )
```

6.3.1.7 intro_boss()

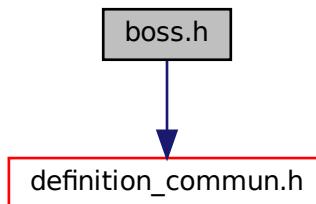
```
void intro_boss (
    boss_t * boss,
    boss_t * clone_1,
    boss_t * clone_2 )
```

Voici le graphe d'appel pour cette fonction :

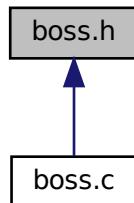


6.4 Référence du fichier boss.h

Graphe des dépendances par inclusion de boss.h:



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



Classes

- struct `boss_s`

Macros

- #define `CHEMIN_BOSS` "ressources/sprite/boss.bmp"
- #define `DUREE_CLONAGE` 60

Énumérations

- enum `type_boss_1_t` { `PRINCIPAL`, `CLONE` }
- enum `action_boss_1_t` {
 `AVANT_ATTAQUE`, `AVANT_DISPARITION`, `DISPARITION`, `APPARITION`,
 `DEPLACEMENT`, `ATTAQUE`, `MORT`, `CLONAGE`,
 `SOMMEIL`, `BLESSE` }

6.4.1 Description détaillée

Auteur

Bruneau Antoine (Antoine.Brunneau.Etu@univ-lemans.fr)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.4.2 Documentation des macros

6.4.2.1 CHEMIN_BOSS

```
#define CHEMIN_BOSS "ressources/sprite/boss.bmp"
```

6.4.2.2 DUREE_CLONAGE

```
#define DUREE_CLONAGE 60
```

6.4.3 Documentation du type de l'énumération

6.4.3.1 action_boss_1_t

```
enum action_boss_1_t
```

Valeurs énumérées

AVANT_ATTAQUE	
AVANT_DISPARITION	
DISPARITION	
APPARITION	
DEPLACEMENT	
ATTAQUE	
Généré par Doxygen	MORT
	CLONAGE
	SOMMEIL
	BLESSE

6.4.3.2 type_boss_1_t

enum `type_boss_1_t`

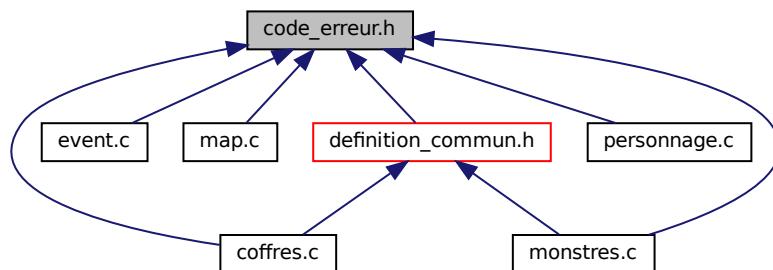
Valeurs énumérées

PRINCIPAL	le boss
CLONE	un clone du boss

6.5 Référence du fichier code_erreur.h

Fichier contenant les codes d'erreur du programme.

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



Macros

- `#define erreur(msg, code_erreur, ...)`
Macro qui permet de facilement afficher une erreur.
- `#define warning(msg, code_erreur, ...)`
Macro qui permet de facilement afficher un Warning.

Définitions de type

- `typedef int err_t`

Énumérations

```
— enum types_erreur {
    AUCUNE_ERREUR, SDL_ERREUR, ERREUR_TEXTURE, OUT_OF_MEM,
    BUFFER_EMPTY, ERREUR_FICHIER, ERREUR_FONCTION, ERREUR_MAP,
    ERREUR_INIT, ERREUR	SDL_IMG, ERREUR	SDL_TTF, ERREUR	SDL_MIX,
    ERREUR	SDL_AUDIO, ERREUR	SDL_TIMER, ERREUR	SDL_MOUSE, ERREUR	SDL_JOYSTICK,
    ERREUR	SDL_RENDER, ERREUR	SDL_WINDOW, ERREUR	SDL_MUSIC, ERREUR	SDL_SURFACE,
    ERREUR	SDL_PIXEL, ERREUR	SDL_MOUSEMOTION, ERREUR	SDL_MOUSEBUTTON, ERREUR	SDL_KEYBOARD,
    ERREUR	SDL_SCANCODE, ERREUR	SDL_QUIT, ERREUR	SDL_EVENT, ERREUR	SDL_TIMER_START,
    ERREUR	SDL_TIMER_STOP, ERREUR_LISTE, ERR_CREATION_REPERTOIRE_SAUVAGEARDE,
    ERR_RECTANGLE_TOO_BIG,
    ERREUR_JSON_CLE_NON_TROUVEE }
```

L'énumération des codes d'erreur du programme.

6.5.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr) Ce fichier contient une enumeration des codes d'erreur du programme, permettant ainsi une modification simple et une gestion plus facile des erreurs.

Version

1.0

Date

05/04/22

Copyright

Copyright (c) 2022

6.5.2 Documentation des macros

6.5.2.1 erreur

```
#define erreur(
    msg,
    code_erreur,
    ... )
```

Valeur :

```
{
    char *msp = malloc(sizeof(char) * (500));
    char *mspbis = malloc(sizeof(char) * (500));
    sprintf(msp, msg, #__VA_ARGS__);
    SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "%s\nErreur : 0x%X\n", msp, code_erreur);
    sprintf(mspbis, "%s\nErreur : 0x%X\n", msp, code_erreur);
    SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR, "Erreur", mspbis, NULL);
    free(msp);
    free(mspbis);
    fermer_programme(code_erreur);
}
```

Auteur

Ange Despert

Attention, cette macro quitte le programme.

6.5.2.2 warning

```
#define warning(
    msg,
    code_erreur,
    ...
)
```

Valeur :

```
{
    char *msp = malloc(sizeof(char) * (500));
    char *mspbis = malloc(sizeof(char) * (500));
    sprintf(msp, msg, ##__VA_ARGS__);
    SDL_LogWarn(SDL_LOG_CATEGORY_APPLICATION, "%s\nErreur : 0x%X\n", msp, code_erreur);
    sprintf(mspbis, "%s\nErreur : 0x%X\n", msp, code_erreur);
    SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_WARNING, "Attention", mspbis, NULL);
    free(msp);
    free(mspbis);
}
```

Auteur

Ange Despert

6.5.3 Documentation des définitions de type

6.5.3.1 err_t

```
typedef int err_t
```

6.5.4 Documentation du type de l'énumération

6.5.4.1 types_erreur

```
enum types_erreur
```

Cela permet une gestion plus simple des erreurs. L'utilisateur peut donc connaitre aisement le code d'erreur et le message correspondant. Permettant au developpeur de répondre à l'erreur en question.

Valeurs énumérées

AUCUNE_ERREUR	Aucune erreur
SDL_ERREUR	Une erreur liée à la SDL
ERREUR_TEXTURE	Une erreur liée à une texture
OUT_OF_MEM	Une erreur liée à l'allocation mémoire
BUFFER_EMPTY	Le buffer est vide
ERREUR_FICHIER	Une erreur liée au fichier (introuvable ou écriture ou lecture impossible)
ERREUR_FONCTION	Valeur renvoyée par une fonction qui échoue
ERREUR_MAP	Une erreur liée à la map

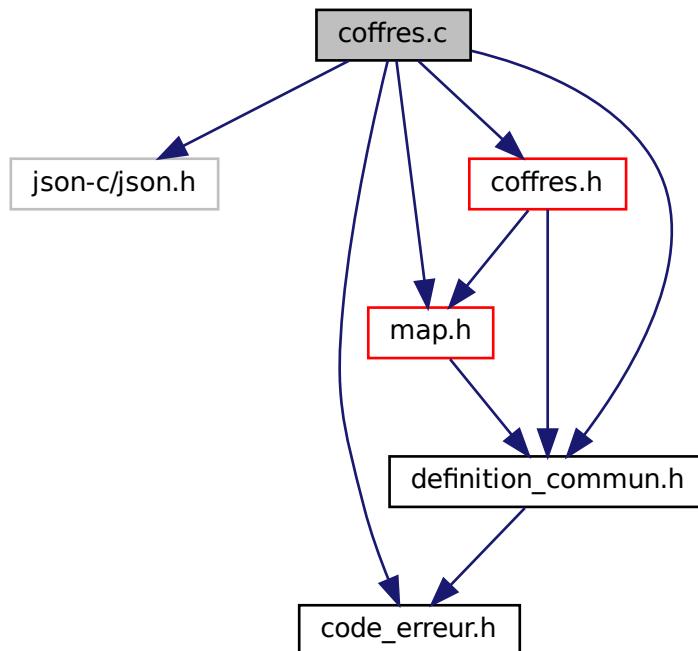
Valeurs énumérées

ERREUR_INIT	Une erreur liée à l'initialisation du programme
ERREUR SDL_IMG	Une erreur liée à l'initialisation de SDL_image
ERREUR SDL_TTF	Une erreur liée à l'initialisation de SDL_ttf
ERREUR SDL_MIX	Une erreur liée à l'initialisation de SDL_mixer
ERREUR SDL_AUDIO	Une erreur liée à l'initialisation de SDL_audio
ERREUR SDL_TIMER	Une erreur liée à l'initialisation de SDL_timer
ERREUR SDL_MOUSE	Une erreur liée à l'initialisation de SDL_mouse
ERREUR SDL_JOYSTICK	Une erreur liée à l'initialisation de SDL_joystick
ERREUR SDL_RENDER	Une erreur liée à l'initialisation du rendu
ERREUR SDL_WINDOW	Une erreur liée à l'initialisation de la fenêtre
ERREUR SDL_MUSIC	Une erreur liée à l'initialisation de la musique
ERREUR SDL_SURFACE	Une erreur liée à l'initialisation de la surface
ERREUR SDL_PIXEL	Une erreur liée à l'initialisation du pixel
ERREUR SDL_MOUSEMOTION	Une erreur liée aux mouvements de la souris
ERREUR SDL_MOUSEBUTTON	Une erreur liée aux boutons de la souris
ERREUR SDL_KEYBOARD	Une erreur liée aux touches du clavier
ERREUR SDL_SCANCODE	Une erreur liée aux scancode du clavier
ERREUR SDL_QUIT	Une erreur liée au clic sur la croix
ERREUR SDL_EVENT	Une erreur liée aux événements
ERREUR SDL_TIMER_START	Une erreur liée au démarrage du timer
ERREUR SDL_TIMER_STOP	Une erreur liée à l'arrêt du timer
ERREUR_LISTE	Une erreur liée à une liste
ERR_CREATION_REPERTOIRE_SAUVAGEARDE	Une erreur liée à la création du répertoire de sauvegarde
ERR_RECTANGLE_TOO_BIG	Une erreur liée au rectangle trop grand
ERREUR_JSON_CLE_NON_TROUVEE	Une Erreur liée à l'impossibilité de trouver une clé dans un fichier JSON

6.6 Référence du fichier coffres.c

Fichier contenant toutes les fonctions concernant les coffres.

Graphe des dépendances par inclusion de coffres.c:



Fonctions

- `void charger_base_coffre (char *chemin_fichier, liste_base_coffres_t **liste_base_coffres)`
Fonction qui recopie les informations d'un fichier json pour les insérer dans la structure liste_base_coffres.
- `type_coffre_t nom_coffre_to_type_coffre (const char *nom_coffre)`
Convertit une chaîne de caractères en type de coffre.
- `coffre_t * creer_coffre (int id_cle, int id_loot, liste_base_coffres_t *liste_base_coffres, const char *const nom_coffre, int x, int y, t_map *map)`
Fonction qui crée et initialise un coffre sur une carte.
- `void info_coffre (coffre_t *coffre)`
Affiche les informations sur un coffre dans la console.
- `t_direction_1 inverser_direction (t_direction_1 direction)`
Inverse une direction.
- `void interaction_coffre (SDL_Rect *coffre_rect, joueur_t *joueur, lobjet_t *objets)`
Gère les interactions du joueur avec un coffre lors d'une collision.

Variables

- `liste_base_coffres_t * liste_base_coffres = NULL`

6.6.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

03/04/2022

Copyright

Copyright (c) 2022

6.6.2 Documentation des fonctions

6.6.2.1 charger_base_coffre()

```
void charger_base_coffre (
    char * chemin_fichier,
    liste_base_coffres_t ** liste_base_coffres )
```

Auteur

Max Descomps

Paramètres

<i>chemin_fichier</i>	Le nom du fichier à lire
<i>liste_base_coffres</i>	La base dans laquelle enregistrer les coffres

Voici le graphe des appels de cette fonction :



6.6.2.2 creer_coffre()

```
coffre_t* creer_coffre (
    int id_cle,
    int id_loot,
```

```

liste_base_coffres_t * liste_base_coffres,
const char *const nom_coffre,
int x,
int y,
t_map * map )

```

Auteur

Max Descomps

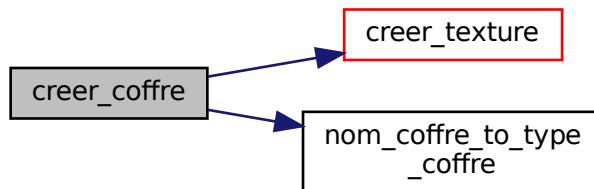
Paramètres

<i>id_cle</i>	L'identificateur de l'objet de quête nécessaire pour ouvrir le coffre, sinon 0
<i>id_loot</i>	L'identificateur de l'objet obtenu en ouvrant le coffre, sinon 0
<i>liste_base_coffres</i>	Les coffres de base
<i>nom_coffre</i>	Le nom du coffre à créer
<i>x</i>	La position en abscisse du coffre sur la map
<i>y</i>	La position en ordonnée du coffre sur la map
<i>map</i>	La carte dans laquelle mettre le coffre

Renvoie

Instance nouvellement allouée du type [coffre_t](#) contenant les informations du coffre ou NULL

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.6.2.3 info_coffre()

```
void info_coffre (
    coffre_t * coffre )
```

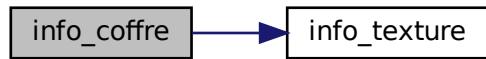
Auteur

Max Descomps

Paramètres

<i>coffre</i>	Le coffre sur lequel on se renseigne
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



6.6.2.4 interaction_coffre()

```
void interaction_coffre (
    SDL_Rect * coffre_rect,
    joueur_t * joueur,
    lobjet_t * objets )
```

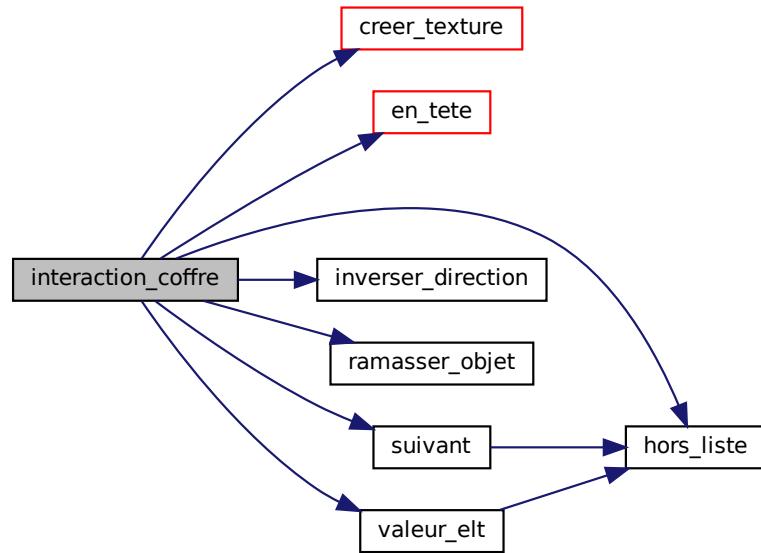
Auteur

Max Descomps

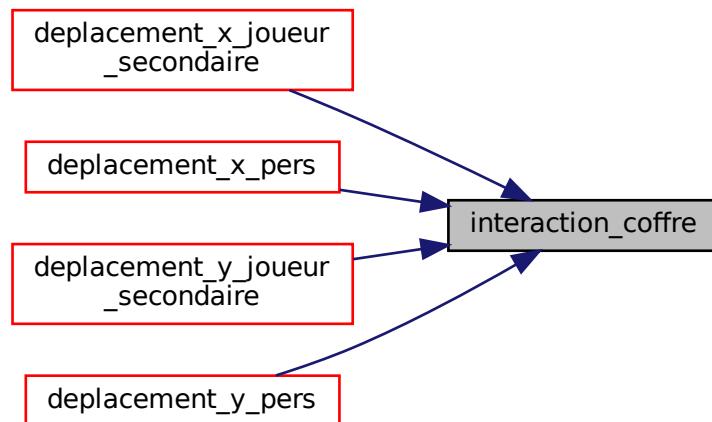
Paramètres

<i>coffre_rect</i>	Les coordonnées du prochain pas du personnage
<i>joueur</i>	Le joueur activant le coffre
<i>objets</i>	Les objets du jeu

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.6.2.5 inverser_direction()

```
t_direction_1 inverser_direction (
    t_direction_1 direction )
```

Auteur

Max Descomps

Paramètres

<i>direction</i>	La direction à inverser
------------------	-------------------------

Renvoie

Un type t_direction_1

Voici le graphe des appelants de cette fonction :



6.6.2.6 nom_coffre_to_type_coffre()

```
type_coffre_t nom_coffre_to_type_coffre (
    const char * nom_coffre )
```

Auteur

Max Descomps

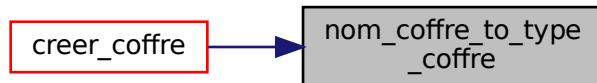
Paramètres

<i>nom_coffre</i>	La chaîne de caractères à convertir
-------------------	-------------------------------------

Renvoie

Une valeur du type type_coffre_t représentant le type de coffre

Voici le graphe des appels de cette fonction :



6.6.3 Documentation des variables

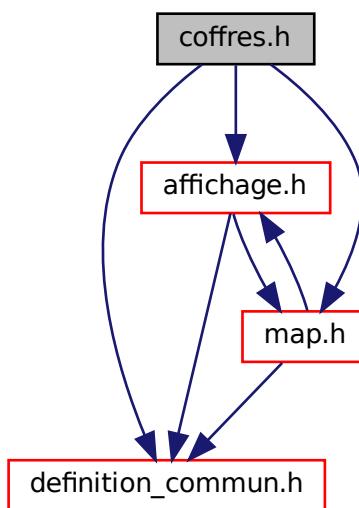
6.6.3.1 liste_base_coffres

```
liste_base_coffres_t* liste_base_coffres = NULL
```

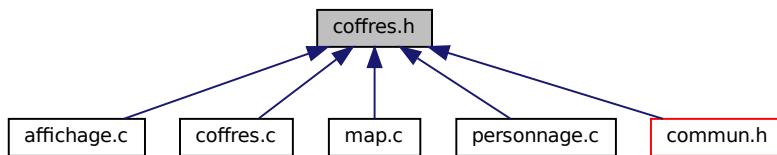
6.7 Référence du fichier coffres.h

Fonctions concernant les coffres.

Graphe des dépendances par inclusion de coffres.h:



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



Classes

- struct `coffre_s`
- struct `base_coffre_s`
Structure contenant les propriétés du coffre importé
- struct `liste_base_coffres_s`
Structure contenant un tableau avec tous les coffres possibles du jeu.

Macros

- #define `COFFRE_FACE_OUVERT` "ressources/sprite/coffrefaceouvert.bmp"
- #define `COFFRE_PROFIL_OUVERT` "ressources/sprite/coffreprofilouvert.bmp"

Énumérations

- enum `type_coffre_t` {
 `PROFIL_FERME`, `PROFIL_OUVERT`, `FACE_FERME`, `FACE_OUVERT`,
 `COFFRE_INCONNU`
}
- enum `etat_coffre_t` { `OUVERT`, `FERME` }

Fonctions

- void `charger_base_coffre` (char *chemin_fichier, liste_base_coffres_t **`liste_base_coffres`)
Fonction qui recopie les informations d'un fichier json pour les insérer dans la structure liste_base_coffres.
- `coffre_t * creer_coffre` (int id_cle, int id_loot, liste_base_coffres_t *`liste_base_coffres`, const char *const nom_coffre, int x, int y, t_map *`map`)
Fonction qui créer et initialise un coffre sur une carte.
- `type_coffre_t nom_coffre_to_type_coffre` (const char *nom_coffre)
Convertit une chaîne de caractères en type de coffre.
- void `info_coffre` (`coffre_t *coffre`)
Affiche les informations sur un coffre dans la console.
- void `interaction_coffre` (SDL_Rect *coffre_rect, joueur_t *joueur, lobjet_t *objets)
Gère les interactions du joueur avec un coffre lors d'une collision.
- `t_direction_1 inverser_direction` (`t_direction_1 direction`)
Inverse une direction.

Variables

- liste_base_coffres_t * `liste_base_coffres`

6.7.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

04/03/2022

Copyright

Copyright (c) 2022

6.7.2 Documentation des macros

6.7.2.1 COFFRE_FACE_OUVERT

```
#define COFFRE_FACE_OUVERT "ressources/sprite/coffrefaceouvert.bmp"
```

6.7.2.2 COFFRE_PROFIL_OUVERT

```
#define COFFRE_PROFIL_OUVERT "ressources/sprite/coffreprofilouvert.bmp"
```

6.7.3 Documentation du type de l'énumération

6.7.3.1 etat_coffre_t

```
enum etat_coffre_t
```

Type enum indiquant si un coffre a été ouvert par un joueur

Valeurs énumérées

OUVERT	Coffre déjà ouvert
FERME	Coffre jamais ouvert

6.7.3.2 type_coffre_t

enum `type_coffre_t`

Type enum renseignant sur le type d'un coffre

Valeurs énumérées

<code>PROFIL_FERME</code>	Modèle de coffre fermé vu de profil
<code>PROFIL_OUVERT</code>	Modèle de coffre ouvert vu de profil
<code>FACE_FERME</code>	Modèle de coffre fermé vu de face
<code>FACE_OUVERT</code>	Modèle de coffre ouvert vu de face
<code>COFFRE_INCONNU</code>	Modèle de coffre inconnu du programme

6.7.4 Documentation des fonctions

6.7.4.1 charger_base_coffre()

```
void charger_base_coffre (
    char * chemin_fichier,
    liste_base_coffres_t ** liste_base_coffres )
```

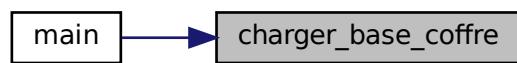
Auteur

Max Descomps

Paramètres

<code>chemin_fichier</code>	Le nom du fichier à lire
<code>liste_base_coffres</code>	La base dans laquelle enregistrer les coffres

Voici le graphe des appelants de cette fonction :



6.7.4.2 creer_coffre()

```
coffre_t* creer_coffre (
    int id_cle,
    int id_loot,
    liste_base_coffres_t * liste_base_coffres,
    const char *const nom_coffre,
    int x,
    int y,
    t_map * map )
```

Auteur

Max Descomps

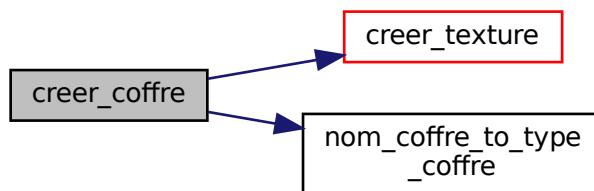
Paramètres

<i>id_cle</i>	L'identificateur de l'objet de quête nécessaire pour ouvrir le coffre, sinon 0
<i>id_loot</i>	L'identificateur de l'objet obtenu en ouvrant le coffre, sinon 0
<i>liste_base_coffres</i>	Les coffres de base
<i>nom_coffre</i>	Le nom du coffre à créer
<i>x</i>	La position en abscisse du coffre sur la map
<i>y</i>	La position en ordonnée du coffre sur la map
<i>map</i>	La carte dans laquelle mettre le coffre

Renvoie

Instance nouvellement allouée du type `coffre_t` contenant les informations du coffre ou NULL

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.7.4.3 info_coffre()

```
void info_coffre (
    coffre_t * coffre )
```

Auteur

Max Descomps

Paramètres

<i>coffre</i>	Le coffre sur lequel on se renseigne
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



6.7.4.4 interaction_coffre()

```
void interaction_coffre (
    SDL_Rect * coffre_rect,
    joueur_t * joueur,
    lobjet_t * objets )
```

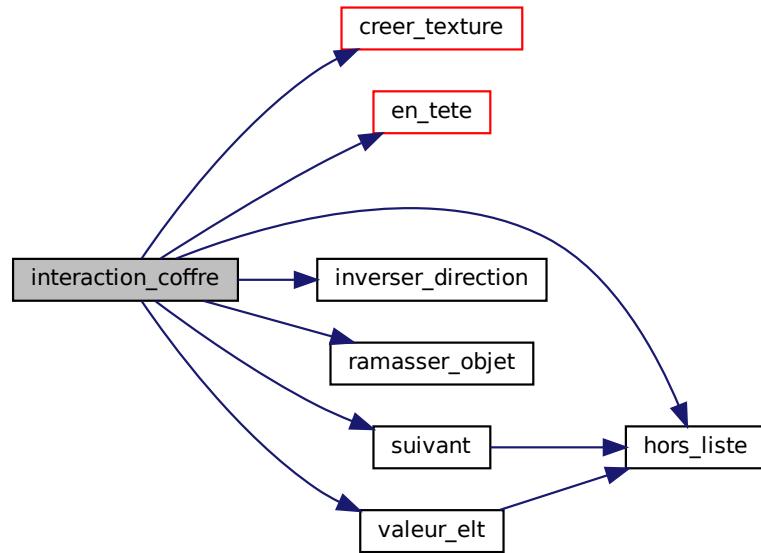
Auteur

Max Descomps

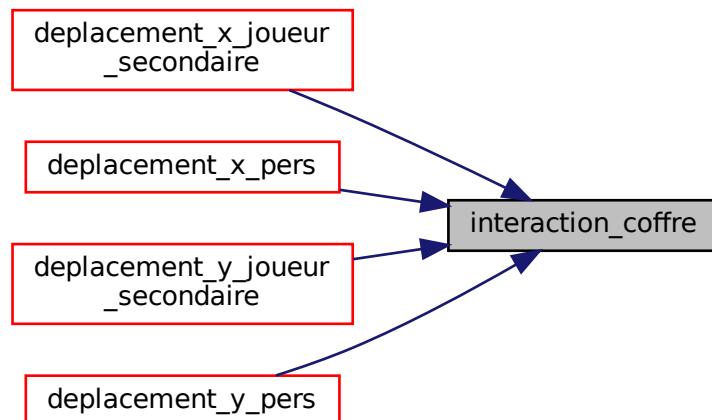
Paramètres

<i>coffre_rect</i>	Les coordonnées du prochain pas du personnage
<i>joueur</i>	Le joueur activant le coffre
<i>objets</i>	Les objets du jeu

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.7.4.5 inverser_direction()

```
t_direction_1 inverser_direction (
    t_direction_1 direction )
```

Auteur

Max Descomps

Paramètres

<i>direction</i>	La direction à inverser
------------------	-------------------------

Renvoie

Un type t_direction_1

Voici le graphe des appelants de cette fonction :



6.7.4.6 nom_coffre_to_type_coffre()

```
type_coffre_t nom_coffre_to_type_coffre (
    const char * nom_coffre )
```

Auteur

Max Descomps

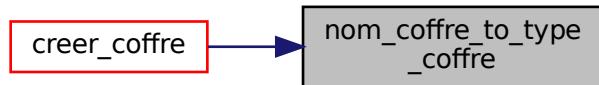
Paramètres

<i>nom_coffre</i>	La chaîne de caractères à convertir
-------------------	-------------------------------------

Renvoie

Une valeur du type type_coffre_t représentant le type de coffre

Voici le graphe des appelants de cette fonction :



6.7.5 Documentation des variables

6.7.5.1 liste_base_coffres

```
liste_base_coffres_t* liste_base_coffres
```

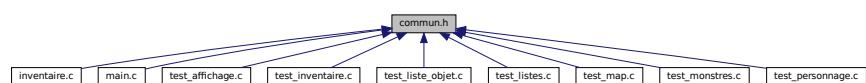
6.8 Référence du fichier commun.h

Fichier contenant les headers nécessaires au programme principal.

Graphe des dépendances par inclusion de commun.h:



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



Fonctions

- void **init** (void)
Fonction qui initialise le Programme.

6.8.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

1.0

Date

05/04/22

Copyright

Copyright (c) 2022

6.8.2 Documentation des fonctions

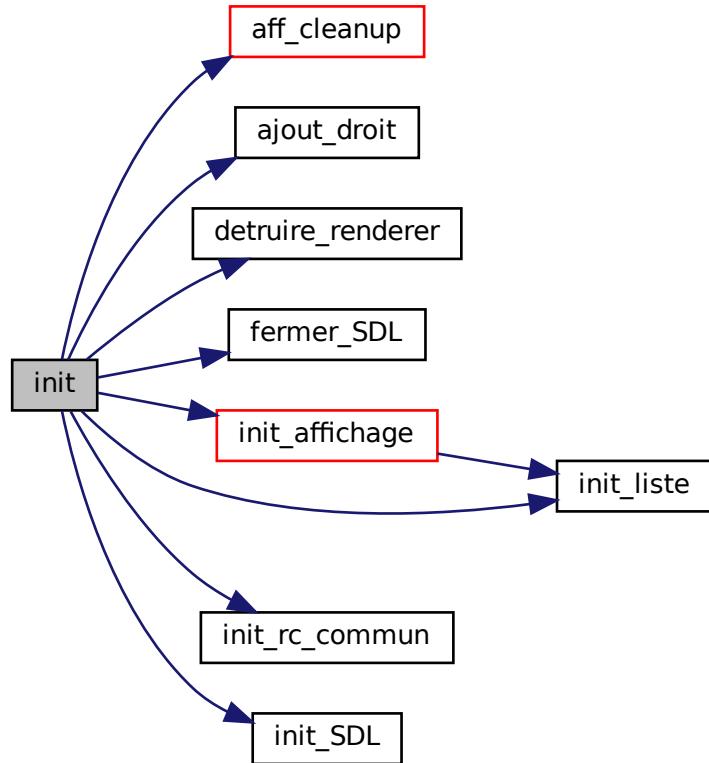
6.8.2.1 init()

```
void init ( )
```

Auteur

Ange Despert

Voici le graphe d'appel pour cette fonction :



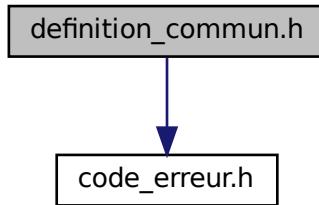
Voici le graphe des appelants de cette fonction :



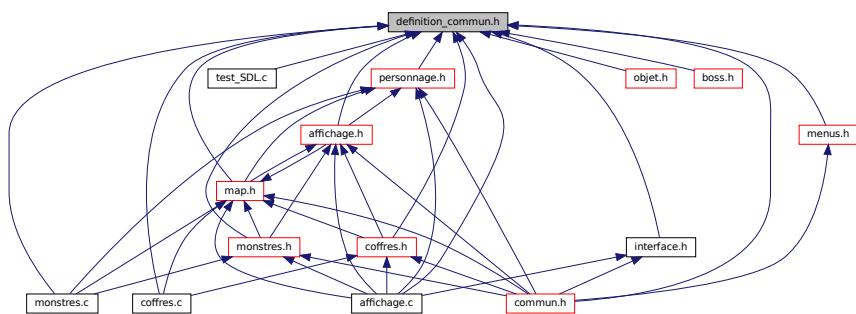
6.9 Référence du fichier definition_commun.h

Contient toutes les définitions communes à tout les fichiers.

Graphe des dépendances par inclusion de definition_commun.h:



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



Classes

- struct point

Macros

- #define bool _Bool
- #define vrai 1
- #define faux 0
- #define SAVE_PATH "Bloody_Sanada"

Définitions de type

- typedef unsigned char byte

Énumérations

- enum t_direction_1 { NORD_1, EST_1, SUD_1, OUEST_1 }

Structure de quatre points cardinaux.
- enum t_direction_2 {

NORD_2, NORD_EST_2, EST_2, SUD_EST_2,

SUD_2, SUD_OUEST_2, OUEST_2, NORD_OUEST_2 }

Structure de huit points cardinaux.

Fonctions

- void `fermer_programme` (int `code_erreur`)
Fonction qui appelle les fonctions pour terminer le programme.

Variables

- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`
- `SDL_Window` * `fenetre_Principale`
- `SDL_Renderer` * `rendu_principal`
- _Bool `running`

6.9.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

1.0

Date

05/04/22

Copyright

Copyright (c) 2022

6.9.2 Documentation des macros

6.9.2.1 bool

```
#define bool _Bool
```

Définition du type booléen

6.9.2.2 faux

```
#define faux 0
```

Valeur faux

6.9.2.3 SAVE_PATH

```
#define SAVE_PATH "Bloody_Sanada"
```

L'emplacement de la sauvegarde

6.9.2.4 vrai

```
#define vrai 1
```

Valeur vrai

6.9.3 Documentation des définitions de type

6.9.3.1 byte

```
typedef unsigned char byte
```

Type servant pour des triggers faisant la taille d'un octet

6.9.4 Documentation du type de l'énumération

6.9.4.1 t_direction_1

```
enum t_direction_1
```

Valeurs énumérées

NORD_1	Point cardinal nord
EST_1	Point cardinal est
SUD_1	Point cardinal sud
OUEST_1	Point cardinal ouest

6.9.4.2 t_direction_2

```
enum t_direction_2
```

Valeurs énumérées

NORD_2	Point cardinal nord
NORD_EST_2	Point cardinal nord-est
EST_2	Point cardinal est
SUD_EST_2	Point cardinal sud-est
SUD_2	Point cardinal sud
SUD_OUEST_2	Point cardinal sud-ouest
OUEST_2	Point cardinal ouest
NORD_OUEST_2	Point cardinal nord-ouest

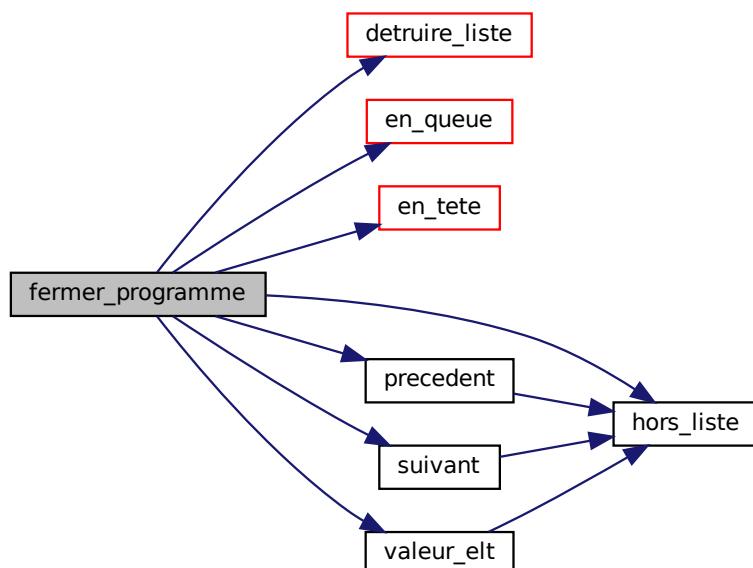
6.9.5 Documentation des fonctions**6.9.5.1 fermer_programme()**

```
void fermer_programme (
    int code_erreur )
```

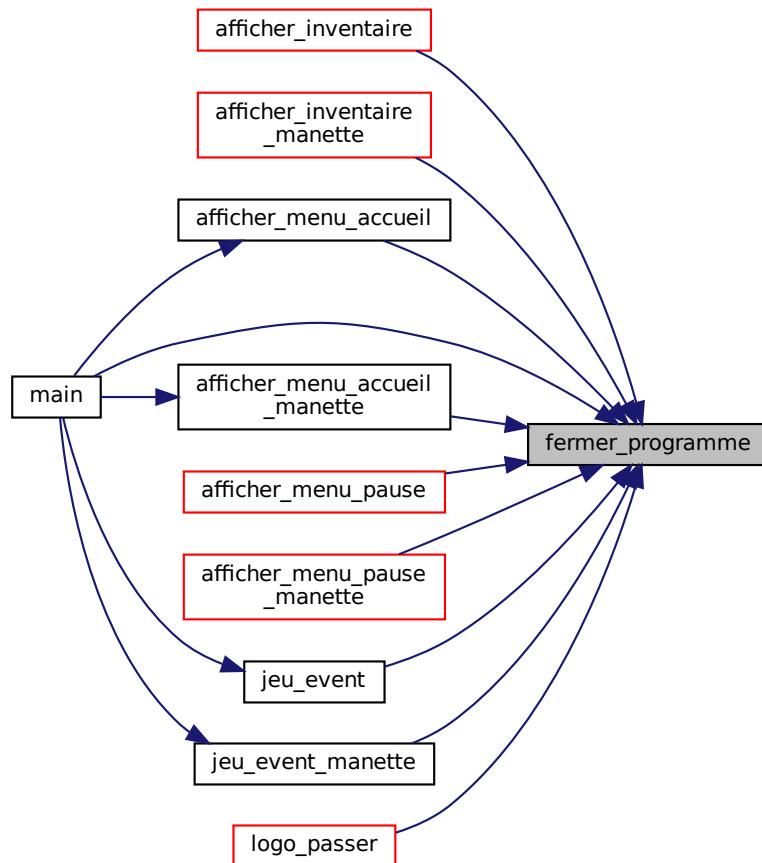
Auteur

Ange Despert

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.9.6 Documentation des variables

6.9.6.1 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.9.6.2 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

6.9.6.3 fenetre_Principale

`SDL_Window* fenetre_Principale`

Pointeur sur la fenêtre principale du programme

6.9.6.4 rendu_principal

`SDL_Renderer* rendu_principal`

Le rendu principal du programme

6.9.6.5 running

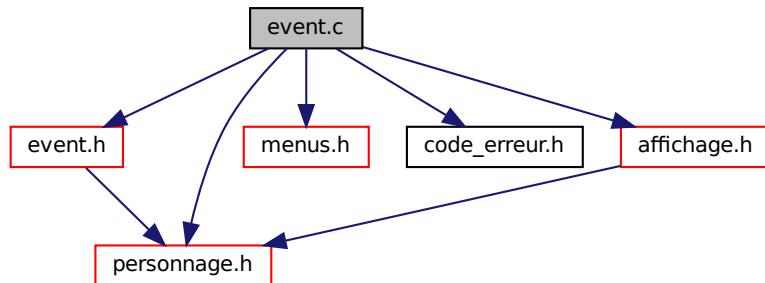
`_Bool running`

<Définition du type booléen Booléen qui permet de savoir si le programme doit s'arrêter

6.10 Référence du fichier event.c

Fonctions de gestion des événements du jeu.

Graphe des dépendances par inclusion de event.c:



Fonctions

- static void `keyDown` (SDL_KeyboardEvent *ev, joueur_t **joueurs, char *f_src_objet)
 — static void `keyUp` (SDL_KeyboardEvent *ev, joueur_t **joueurs)
Fonction qui gère les événements quand une touche du clavier est relâchée par un joueur.
- static void `joystick_button_down` (SDL_JoyButtonEvent *ev, joueur_t **j)
Fonction qui gère les événements lorsque l'on appuie sur un bouton de la manette.
- static void `joystick_button_up` (SDL_JoyButtonEvent *ev, joueur_t **j)
Fonction qui gère les événements lorsque l'on relâche un bouton de la manette.
- static void `joystick_stick` (joueur_t **j)
Fonction qui gère les déplacements au joystick.
- static void `mouseButtonDown` (SDL_MouseButtonEvent *ev, joueur_t **joueurs)
Fonction qui gère les événements quand un bouton de la souris est pressée par un joueur.
- static void `mouseButtonUp` (SDL_MouseButtonEvent *ev, joueur_t **joueurs)
Fonction qui gère les événements quand un bouton de la souris est relâchée par un joueur.
- void `jeu_event_manette` (joueur_t **joueurs)
Fonction qui gère les événements de la manette.
- void `jeu_event` (joueur_t **joueurs, char *f_src_obj)
Fonction qui gère les événements.
- `bool logo_passer` (void)
Fonction qui gère les événements pendant l'affichage de l'introduction.

Variables

- `SDL_GameController * manette`

6.10.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

0.1

Date

05/04/2022

Copyright

Copyright (c) 2022

6.10.2 Documentation des fonctions

6.10.2.1 jeu_event()

```
void jeu_event (  
    joueur_t ** joueurs,  
    char * f_src_obj )
```

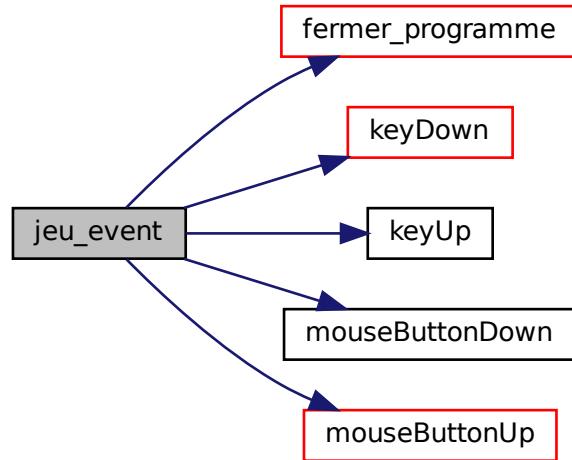
Auteur

Despert Ange

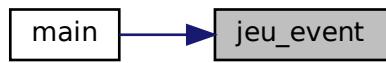
Paramètres

<code>joueurs</code>	Joueurs pouvant provoquer l'événement
<code>f_src_obj</code>	Fichier source des objets du jeu utile à certains événements

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.10.2.2 jeu_event_manette()

```
void jeu_event_manette (joueur_t ** joueurs )
```

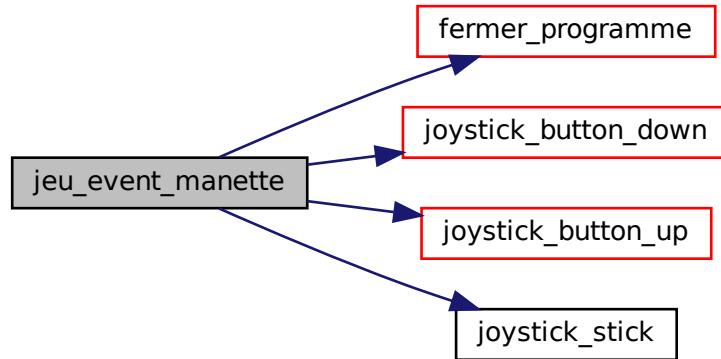
Auteur

Ange Despert

Paramètres

joueurs	Les joueurs existants
---------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.10.2.3 joystick_button_down()

```

static void joystick_button_down (
    SDL_JoyButtonEvent * ev,
    joueur_t ** j ) [static]
  
```

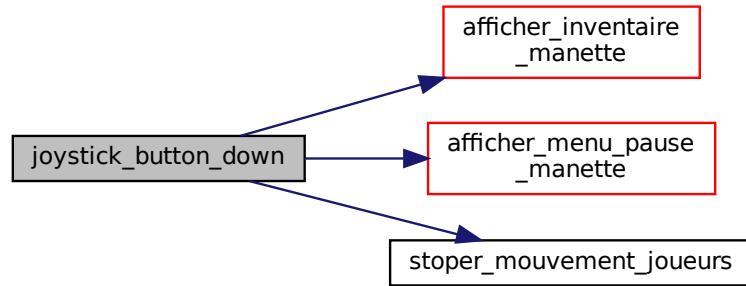
Auteur

Ange Despert

Paramètres

<i>ev</i>	L'événement détecté
<i>j</i>	Les joueurs qui existent

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.10.2.4 joystick_button_up()

```

static void joystick_button_up (
    SDL_JoyButtonEvent * ev,
    joueur_t ** j ) [static]
  
```

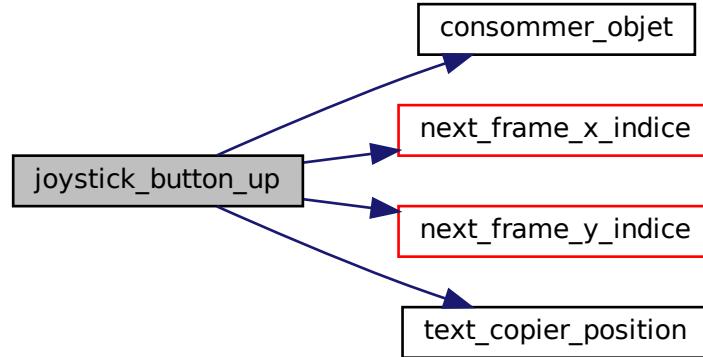
Auteur

Ange Despert

Paramètres

<code>ev</code>	L'événement détecté
<code>j</code>	Les joueurs qui existent

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.10.2.5 joystick_stick()

```
static void joystick_stick (
    joueur_t ** j ) [static]
```

Auteur

Ange Despert

Cette fonction récupère les coordonnées du joystick de la manette.
 Elle crée ensuite une zone morte pour éviter les problèmes de "drift".
 Elle divise ensuite le cercle du joystick en 4 parties pour les 4 directions possibles.
 Et enfin, permet le déplacement du joueur.

Paramètres

<code>j</code>	Les joueurs qui existent
----------------	--------------------------

Voici le graphe des appelants de cette fonction :

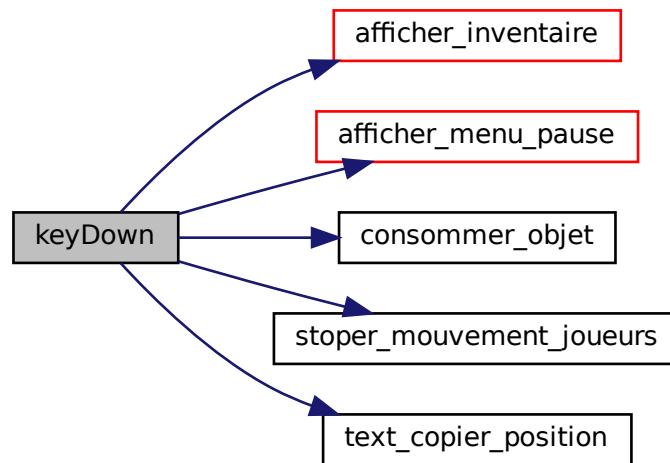


6.10.2.6 keyDown()

```

static void keyDown (
    SDL_KeyboardEvent * ev,
    joueur_t ** joueurs,
    char * f_src_objet ) [static]
  
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.10.2.7 keyUp()

```
void keyUp (
    SDL_KeyboardEvent * ev,
    joueur_t ** joueurs ) [static]
```

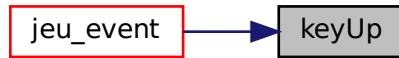
Auteur

Antoine Bruneau

Paramètres

<i>ev</i>	Structure contenant l'événement
<i>joueurs</i>	Joueurs pouvant provoquer l'événement

Voici le graphe des appelants de cette fonction :



6.10.2.8 logo_passer()

```
bool logo_passer (
    void )
```

Auteur

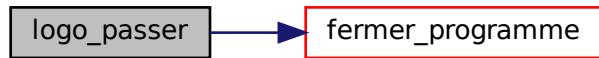
Ange Despert

Cette fonction permet de passer l'introduction avec une touche mais gère encore les événements tels que `SDL_QUIT`.

Renvoie

Si l'on doit passer le logo

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.10.2.9 mouseButtonDown()

```
void mouseButtonDown (
    SDL_MouseButtonEvent * ev,
    joueur_t ** joueurs ) [static]
```

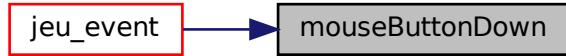
Auteur

Antoine Bruneau

Paramètres

<i>ev</i>	Structure contenant l'événement
<i>joueurs</i>	Joueurs pouvant provoquer l'événement

Voici le graphe des appelants de cette fonction :



6.10.2.10 mouseButtonUp()

```
void mouseButtonUp (
    SDL_MouseButtonEvent * ev,
    joueur_t ** joueurs ) [static]
```

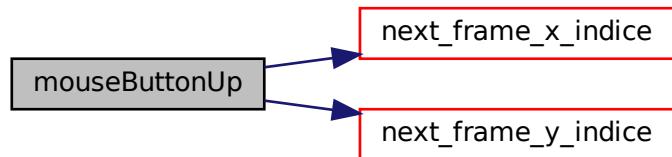
Auteur

Antoine Bruneau

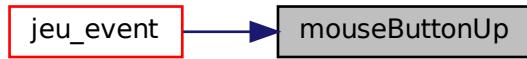
Paramètres

<i>ev</i>	Structure contenant l'événement
<i>joueurs</i>	Joueurs pouvant provoquer l'événement

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.10.3 Documentation des variables

6.10.3.1 manette

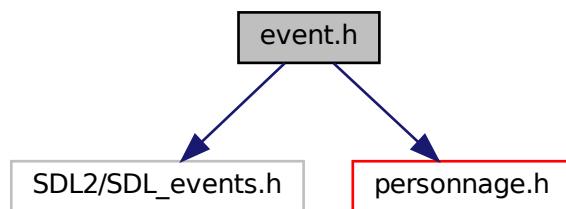
```
SDL_GameController* manette
```

La manette du J1

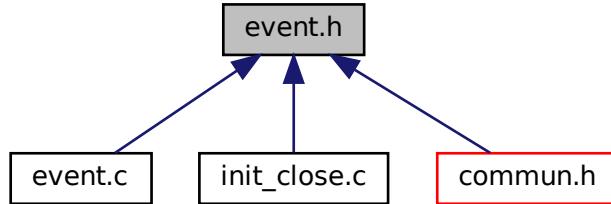
6.11 Référence du fichier event.h

Définitions relatives aux événements.

Graphe des dépendances par inclusion de event.h:



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



Macros

- #define TOUCHE_HAUT SDLK_z
- #define TOUCHE_BAS SDLK_s
- #define TOUCHE_GAUCHE SDLK_q
- #define TOUCHE_DROITE SDLK_d
- #define TOUCHE_TAB SDLK_TAB
- #define TOUCHE_ECHAP SDLK_ESCAPE
- #define TOUCHE_CONSUMMABLE SDLK_e

Fonctions

- _Bool logo_passer (void)
Fonction qui gère les événements pendant l'affichage de l'introduction.
- void jeu_event (joueur_t **joueurs, char *f_src_obj)
Fonction qui gère les événements.
- void jeu_event_manette (joueur_t **joueurs)
Fonction qui gère les événements de la manette.

Variables

- SDL_GameController * manette

6.11.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

0.1

Date

03/02/2022

Copyright

Copyright (c) 2022

6.11.2 Documentation des macros

6.11.2.1 TOUCHE_BAS

```
#define TOUCHE_BAS SDLK_s
```

6.11.2.2 TOUCHE_CONSUMMABLE

```
#define TOUCHE_CONSUMMABLE SDLK_e
```

6.11.2.3 TOUCHE_DROITE

```
#define TOUCHE_DROITE SDLK_d
```

6.11.2.4 TOUCHE_ECHAP

```
#define TOUCHE_ECHAP SDLK_ESCAPE
```

6.11.2.5 TOUCHE_GAUCHE

```
#define TOUCHE_GAUCHE SDLK_q
```

6.11.2.6 TOUCHE_HAUT

```
#define TOUCHE_HAUT SDLK_z
```

6.11.2.7 TOUCHE_TAB

```
#define TOUCHE_TAB SDLK_TAB
```

6.11.3 Documentation des fonctions

6.11.3.1 jeu_event()

```
void jeu_event (
    joueur_t ** joueurs,
    char * f_src_obj )
```

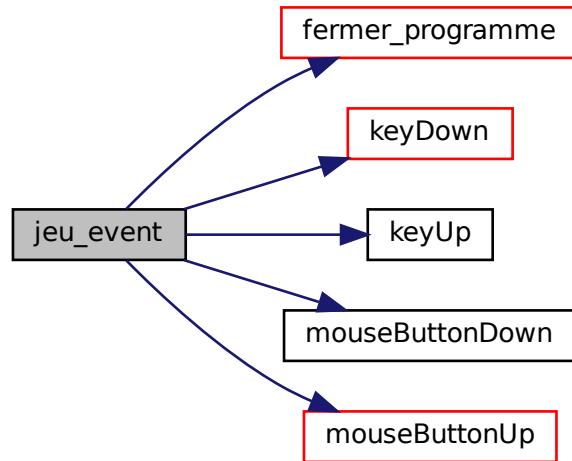
Auteur

Despert Ange

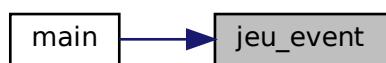
Paramètres

<i>joueurs</i>	Joueurs pouvant provoquer l'événement
<i>f_src_obj</i>	Fichier source des objets du jeu utile à certains événements

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.11.3.2 jeu_event_manette()

```
void jeu_event_manette (
    joueur_t ** joueurs )
```

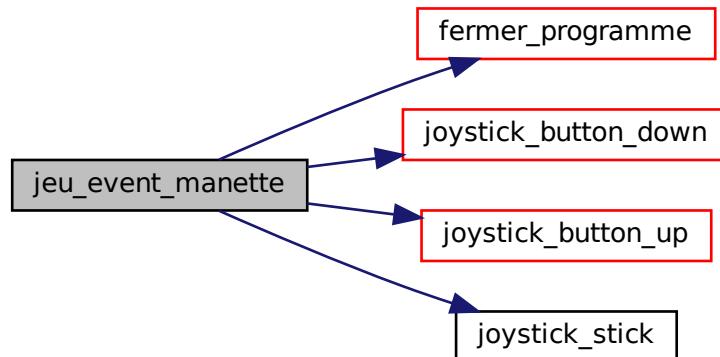
Auteur

Ange Despert

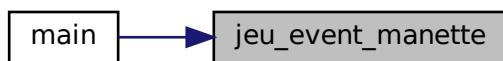
Paramètres

<i>joueurs</i>	Les joueurs existants
----------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.11.3.3 logo_passer()

```
_Bool logo_passer (
    void )
```

Auteur

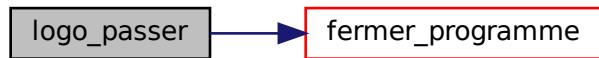
Ange Despert

Cette fonction permet de passer l'introduction avec une touche mais gère encore les événements tels que `SDL_QUIT`.

Renvoie

Si l'on doit passer le logo

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.11.4 Documentation des variables

6.11.4.1 manette

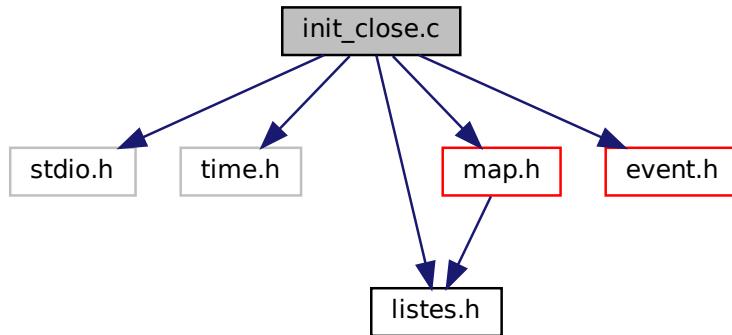
```
SDL_GameController* manette
```

La manette du J1

6.12 Référence du fichier init_close.c

Contient toutes les fonctions pour initialiser le programme.

Graphe des dépendances par inclusion de init_close.c:



Fonctions

- `void fermer_programme (int code_erreur)`
Fonction qui appelle les fonctions pour terminer le programme.
- `static void fermer SDL (void)`
Fonction qui détruit la fenêtre principale et ferme la SDL.
- `static void detruire_renderer (void)`
- `static void init SDL ()`
Fonction qui démarre la SDL et créer la fenêtre principale.
- `static void init_rc_commun (void)`
- `void aff_cleanup (void)`
Fonction ajoutée à la liste de atexit() afin de libérer toute la mémoire allouée.
- `void int_affichage ()`
- `void init ()`
Fonction qui initialise le Programme.

Variables

- `SDL_Window * fenetre_Principale = NULL`
- `SDL_Renderer * rendu_principal = NULL`
- `SDL_Window * fenetre_sous_rendu = NULL`
- `SDL_Renderer * sous_rendu = NULL`
- `bool running = vrai`
- `SDL_Rect * hors_hitbox = NULL`
- `list * f_close = NULL`

6.12.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

0.2

Date

27/03/22

Copyright

Copyright (c) 2022

6.12.2 Documentation des fonctions

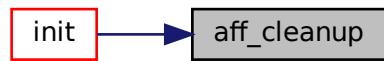
6.12.2.1 aff_cleanup()

```
void aff_cleanup (
    void )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.12.2.2 detruire_renderer()

```
static void detruire_renderer (
    void ) [static]
```

Voici le graphe des appelants de cette fonction :



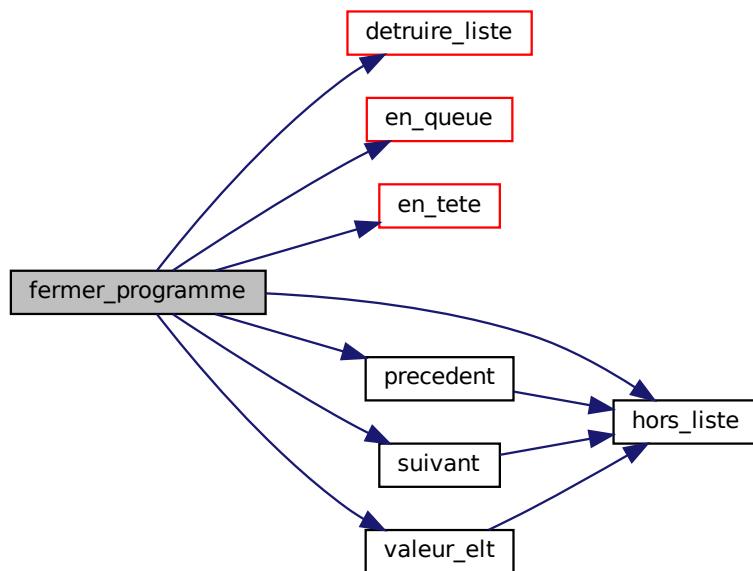
6.12.2.3 fermer_programme()

```
void fermer_programme (
    int code_erreur )
```

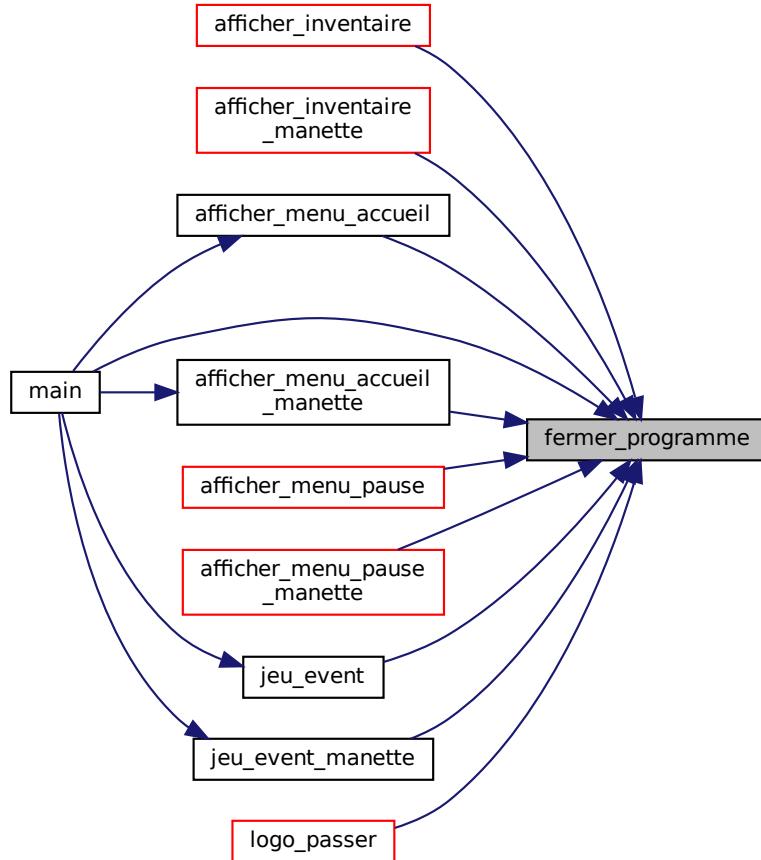
Auteur

Ange Despert

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



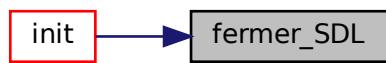
6.12.2.4 fermer SDL()

```
void fermer(SDL (
    void ) [static]
```

Auteur

Ange Despert

Voici le graphe des appelants de cette fonction :



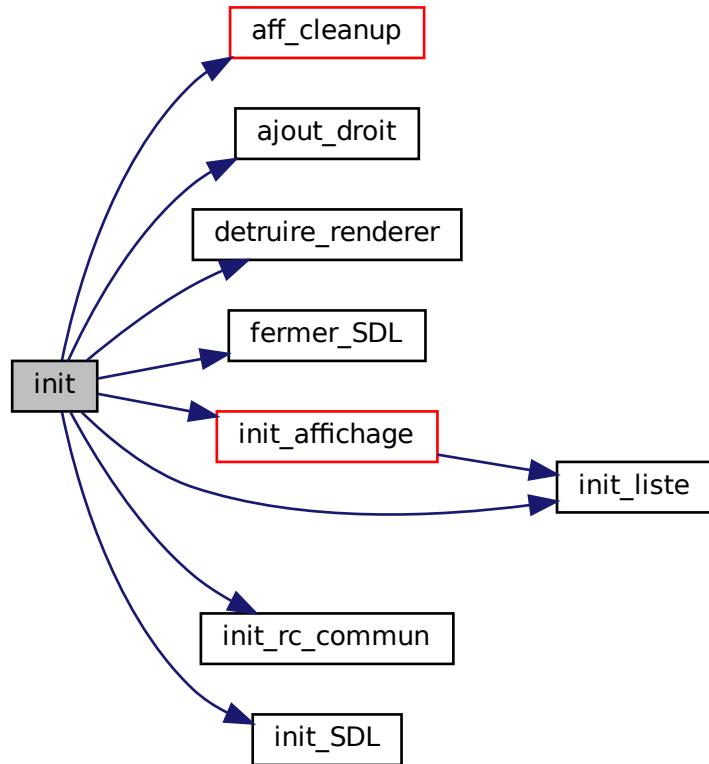
6.12.2.5 init()

```
void init ( )
```

Auteur

Ange Despert

Voici le graphe d'appel pour cette fonction :



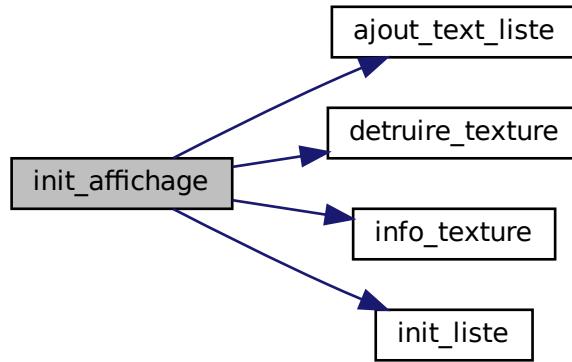
Voici le graphe des appels de cette fonction :



6.12.2.6 init_affichage()

```
void init_affichage ( )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.12.2.7 init_rc_commun()

```
static void init_rc_commun (
    void ) [static]
```

Voici le graphe des appelants de cette fonction :



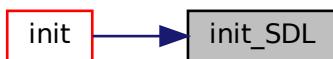
6.12.2.8 init SDL()

```
void init SDL ( ) [static]
```

Auteur

Ange Despert

Voici le graphe des appelants de cette fonction :



6.12.3 Documentation des variables

6.12.3.1 f_close

```
list* f_close = NULL
```

Liste des fonctions à appeler lors de la fermeture du programme

6.12.3.2 fenetre_Principale

```
SDL_Window* fenetre_Principale = NULL
```

Pointeur sur la fenêtre principale du programme

6.12.3.3 fenetre_sous_rendu

```
SDL_Window* fenetre_sous_rendu = NULL
```

6.12.3.4 hors_hitbox

```
SDL_Rect* hors_hitbox = NULL
```

6.12.3.5 rendu_principal

```
SDL_Renderer* rendu_principal = NULL
```

Le rendu principal du programme

6.12.3.6 running

```
bool running = vrai
```

<Définition du type booléen Booléen qui permet de savoir si le programme doit s'arrêter

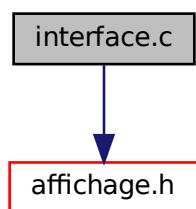
6.12.3.7 sous_rendu

```
SDL_Renderer* sous_rendu = NULL
```

6.13 Référence du fichier interface.c

Fonctions liées à l'interface joueur.

Graphe des dépendances par inclusion de interface.c:



Fonctions

- void [RenderHPBar](#) (int x, int y, int w, int h, float pourcent, SDL_Color vie, SDL_Color jauge)
Fonction qui affiche une barre de vie horizontale.

6.13.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.13.2 Documentation des fonctions

6.13.2.1 RenderHPBar()

```
void RenderHPBar (
    int x,
    int y,
    int w,
    int h,
    float pourcent,
    SDL_Color vie,
    SDL_Color jauge )
```

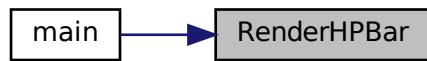
Auteur

Max Descomps

Paramètres

<i>x</i>	position horizontale de la barre
<i>y</i>	position verticale de la barre
<i>w</i>	largeur de la barre
<i>h</i>	hauteur de la barre
<i>pourcent</i>	taux de points de vie
<i>vie</i>	couleur de la barre de vie
<i>jauge</i>	couleur de la barre vide

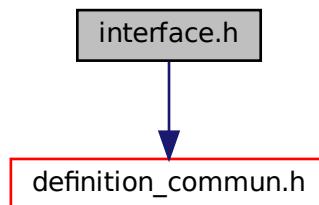
Voici le graphe des appels de cette fonction :



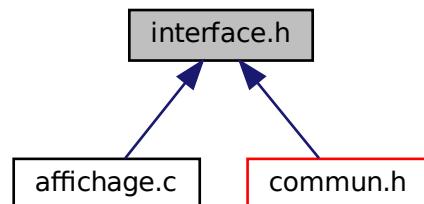
6.14 Référence du fichier interface.h

Définition des fonctions relatives à l'interface joueur.

Graphe des dépendances par inclusion de interface.h:



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



Fonctions

- void **RenderHPBar** (int x, int y, int w, int h, float pourcent, **SDL_Color** vie, **SDL_Color** jauge)
Fonction qui affiche une barre de vie horizontale.

6.14.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

1.0

Date

28/03/2022

Copyright

Copyright (c) 2022

6.14.2 Documentation des fonctions

6.14.2.1 RenderHPBar()

```
void RenderHPBar (
    int x,
    int y,
    int w,
    int h,
    float pourcent,
    SDL_Color vie,
    SDL_Color jauge )
```

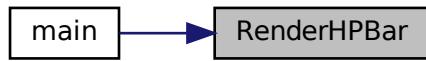
Auteur

Max Descomps

Paramètres

<i>x</i>	position horizontale de la barre
<i>y</i>	position verticale de la barre
<i>w</i>	largeur de la barre
<i>h</i>	hauteur de la barre
<i>pourcent</i>	taux de points de vie
<i>vie</i>	couleur de la barre de vie
<i>jauge</i>	couleur de la barre vide

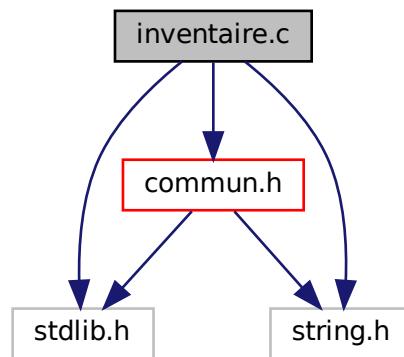
Voici le graphe des appelants de cette fonction :



6.15 Référence du fichier inventaire.c

Fonctions liées au module inventaire.

Graphe des dépendances par inclusion de inventaire.c:



Fonctions

- void **change_statistiques** (joueur_t *joueur)

Fonction qui permet de modifier les statistiques du joueur selon ses objets équipés.
- void **equiper_objet** (joueur_t *joueur, objet_t **objet)

Fonction qui permet de passer un objet du sac(non-équipé) à un emplacement d'équipement(équipé) dans l'inventaire.
- void **desequiper** (joueur_t *joueur, objet_t **objet)

Fonction qui permet de passer un objet d'un emplacement d'équipement(équipé) au sac(non-équipé) dans l'inventaire.
- inventaire_t * **creer_inventaire** (char *fichier_src)
- void **destruire_inventaire** (inventaire_t **inventaire)

Détruit un inventaire.
- void **ramasser_objet** (objet_t *objet, inventaire_t *inventaire)

Fonction qui permet de mettre un objet trouvé dans le sac(inventaire)
- void **tout_ramasser** (lobjet_t *objets, inventaire_t *inventaire)

Met tous les objets du jeu dans l'inventaire (sac)
- void **equiper_sac_slot** (joueur_t *joueur, int slot)

Équipe le joueur de l'objet contenu dans un slot de l'inventaire (sac)
- void **desequiper_slot** (joueur_t *joueur, int slot)

Déséquipe le joueur de l'objet contenu dans un slot de l'inventaire (equipe)
- void **consommer_objet** (joueur_t *joueur)

Utilise le consommable équipé pour soigner le joueur.

6.15.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.3

Date

29/03/2022

Copyright

Copyright (c) 2022

6.15.2 Documentation des fonctions

6.15.2.1 `changement_statistiques()`

```
void changement_statistiques (
    joueur_t * joueur )
```

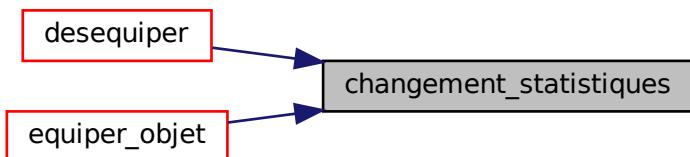
Auteur

Max Descomps

Paramètres

<code>joueur</code>	Joueur dont on change les statistiques
---------------------	--

Voici le graphe des appelants de cette fonction :



6.15.2.2 consommer_objet()

```
void consommer_objet (
    joueur_t * joueur )
```

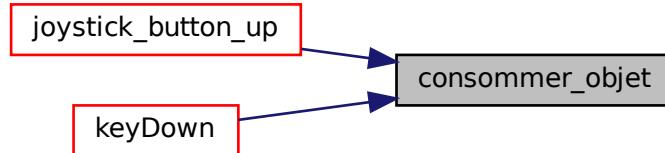
Auteur

Max Descomps

Paramètres

<i>joueur</i>	Le joueur qui consomme un objet
---------------	---------------------------------

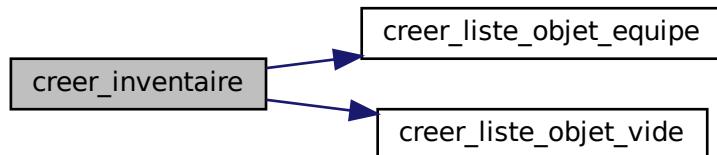
Voici le graphe des appels de cette fonction :



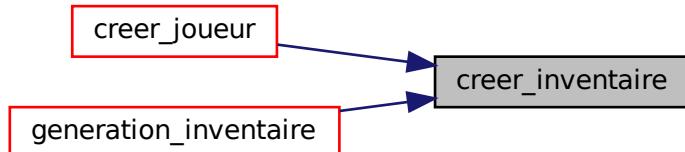
6.15.2.3 creer_inventaire()

```
inventaire_t* creer_inventaire (
    char * fichier_src )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.15.2.4 desequiper()

```
void desequiper (
    joueur_t * joueur,
    objet_t ** objet )
```

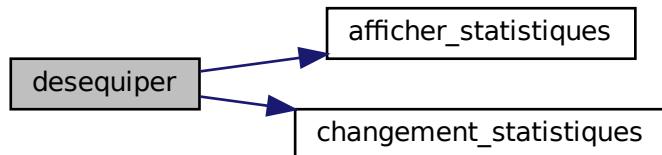
Auteur

Max Descomps

Paramètres

<i>joueur</i>	Joueur à déséquiper
<i>objet</i>	Objet à déséquiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.15.2.5 desequiper_slot()

```
void desequiper_slot (
    joueur_t * joueur,
    int slot )
```

Auteur

Max Descomps

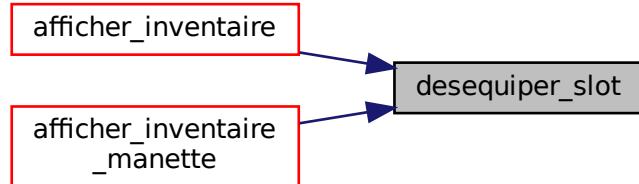
Paramètres

<i>slot</i>	Le slot de l'objet à déséquiper
<i>joueur</i>	Le joueur à déséquiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.15.2.6 detruire_inventaire()

```
void detruire_inventaire (
    inventaire_t ** inventaire )
```

Auteur

Max Descomps

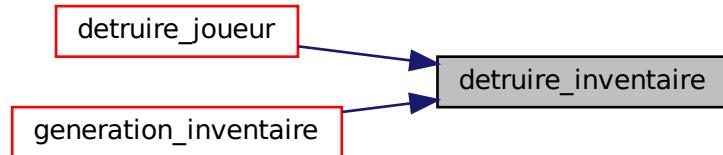
Paramètres

<i>inventaire</i>	Adresse du pointeur sur inventaire_t
-------------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.15.2.7 equiper_objet()

```
void equiper_objet (
    joueur_t * joueur,
    objet_t ** objet )
```

Auteur

Max Descomps

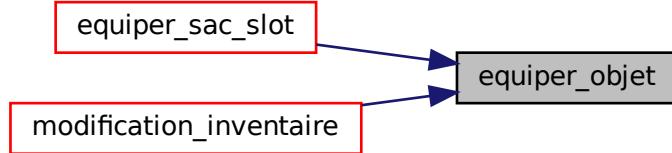
Paramètres

<i>joueur</i>	Joueur à équiper
<i>objet</i>	Objet à équiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.15.2.8 equiper_sac_slot()

```
void equiper_sac_slot (
    joueur_t * joueur,
    int slot )
```

Auteur

Max Descomps

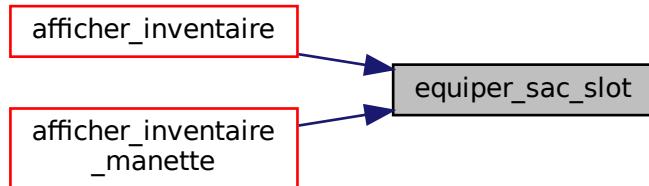
Paramètres

<i>slot</i>	Le slot de l'objet à équiper
<i>joueur</i>	Le joueur à équiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.15.2.9 ramasser_objet()

```
void ramasser_objet (
    objet_t * objet,
    inventaire_t * inventaire )
```

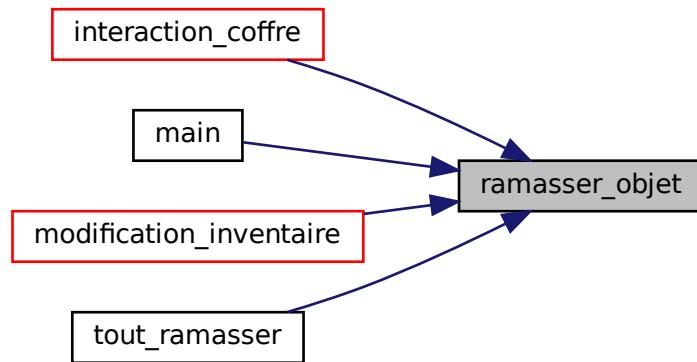
Auteur

Max Descomps

Paramètres

<i>objet</i>	Objet trouvé
<i>inventaire</i>	L'inventaire dans lequel mettre l'objet

Voici le graphe des appelants de cette fonction :



6.15.2.10 tout_ramasser()

```
void tout_ramasser (
    lobjet_t * objets,
    inventaire_t * inventaire )
```

Auteur

Max Descomps

Paramètres

<i>objets</i>	Liste d'objets du jeu
<i>inventaire</i>	Inventaire du joueur

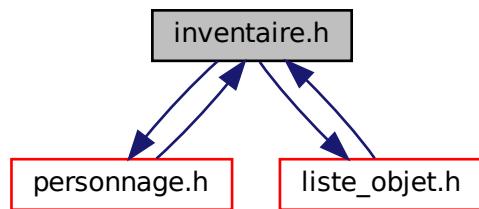
Voici le graphe d'appel pour cette fonction :



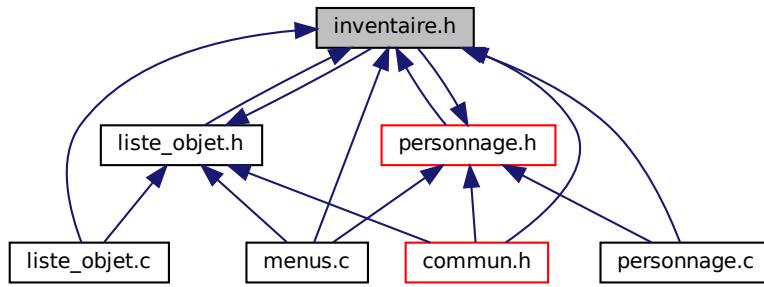
6.16 Référence du fichier inventaire.h

Définition des fonctions relatives au module inventaire.

Graphe des dépendances par inclusion de inventaire.h:



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



Classes

- struct [inventaire_t](#)

Macros

- #define [CAPACITE_SAC](#) 10

Fonctions

- [inventaire_t * creer_inventaire \(\)](#)
Créé un inventaire.
- void [changement_statistiques \(joueur_t *joueur\)](#)
Fonction qui permet de modifier les statistiques du joueur selon ses objets équipés.
- void [equiper_objet \(joueur_t *joueur, objet_t **objet\)](#)
Fonction qui permet de passer un objet du sac(non-équipé) à un emplacement d'équipement(équipé) dans l'inventaire.
- void [desequiper \(joueur_t *joueur, objet_t **objet\)](#)
Fonction qui permet de passer un objet d'un emplacement d'équipement(équipé) au sac(non-équipé) dans l'inventaire.
- void [ramasser_objet \(objet_t *objet, inventaire_t *inventaire\)](#)
Fonction qui permet de mettre un objet trouvé dans le sac(inventaire)
- void [tout_ramasser \(lobjet_t *objets, inventaire_t *inventaire\)](#)
Met tous les objets du jeu dans l'inventaire (sac)
- void [detruire_inventaire \(inventaire_t **inventaire\)](#)
Détruit un inventaire.
- void [equiper_sac_slot \(joueur_t *joueur, int slot\)](#)
Équipe le joueur de l'objet contenu dans un slot de l'inventaire (sac)
- void [desequiper_slot \(joueur_t *joueur, int slot\)](#)
Déséquipe le joueur de l'objet contenu dans un slot de l'inventaire (equipe)
- void [consommer_objet \(joueur_t *joueur\)](#)
Utilise le consommable équipé pour soigner le joueur.

6.16.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

1.0

Date

28/03/2022

Copyright

Copyright (c) 2022

6.16.2 Documentation des macros

6.16.2.1 CAPACITE_SAC

```
#define CAPACITE_SAC 10
```

La capacité maximum du sac dans l'inventaire en nombre d'objets

6.16.3 Documentation des fonctions

6.16.3.1 changement_statistiques()

```
void changement_statistiques (
    joueur_t * joueur )
```

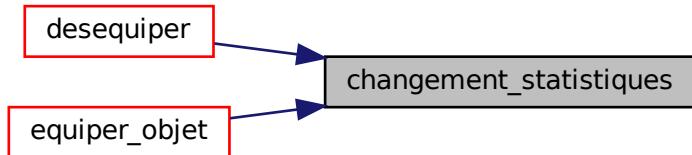
Auteur

Max Descomps

Paramètres

<i>joueur</i>	Joueur dont on change les statistiques
---------------	--

Voici le graphe des appelants de cette fonction :



6.16.3.2 consommer_objet()

```
void consommer_objet (joueur_t * joueur )
```

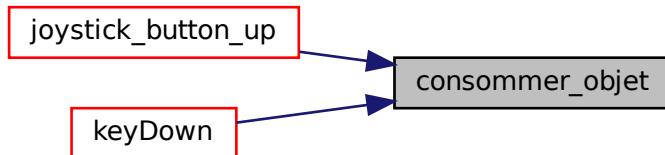
Auteur

Max Descomps

Paramètres

<i>joueur</i>	Le joueur qui consomme un objet
---------------	---------------------------------

Voici le graphe des appelants de cette fonction :



6.16.3.3 creer_inventaire()

```
inventaire_t* creer_inventaire ( )
```

Auteur

Max Descomps

Renvoie

Instance nouvellement allouée du type [inventaire_t](#) ou NULL

6.16.3.4 desequiper()

```
void desequiper (
    joueur_t * joueur,
    objet_t ** objet )
```

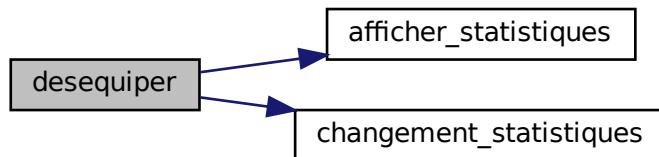
Auteur

Max Descomps

Paramètres

<i>joueur</i>	Joueur à déséquiper
<i>objet</i>	Objet à déséquiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.16.3.5 desequiper_slot()

```
void desequiper_slot (
    joueur_t * joueur,
    int slot )
```

Auteur

Max Descomps

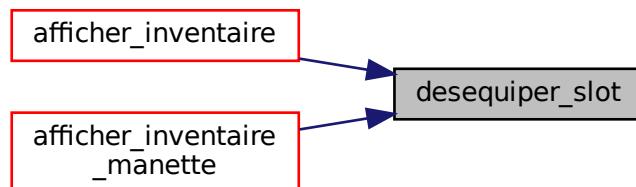
Paramètres

<i>slot</i>	Le slot de l'objet à déséquiper
<i>joueur</i>	Le joueur à déséquiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.16.3.6 detruire_inventaire()

```
void detruire_inventaire (
    inventaire_t ** inventaire )
```

Auteur

Max Descomps

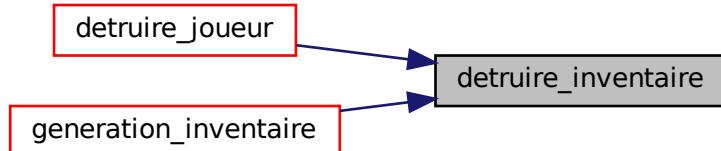
Paramètres

<i>inventaire</i>	Adresse du pointeur sur inventaire_t
-------------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.16.3.7 équiper_objet()

```
void équiper_objet (
    joueur_t * joueur,
    objet_t ** objet )
```

Auteur

Max Descomps

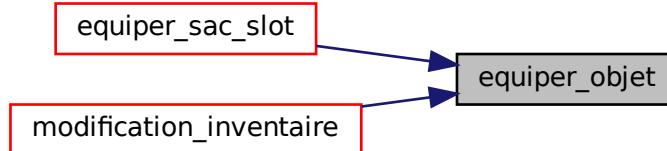
Paramètres

<i>joueur</i>	Joueur à équiper
<i>objet</i>	Objet à équiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.16.3.8 equiper_sac_slot()

```
void equiper_sac_slot (
    joueur_t * joueur,
    int slot )
```

Auteur

Max Descomps

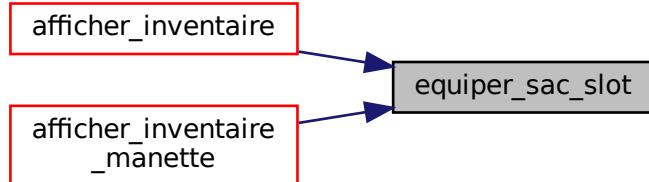
Paramètres

<i>slot</i>	Le slot de l'objet à équiper
<i>joueur</i>	Le joueur à équiper

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.16.3.9 ramasser_objet()

```
void ramasser_objet (
    objet_t * objet,
    inventaire_t * inventaire )
```

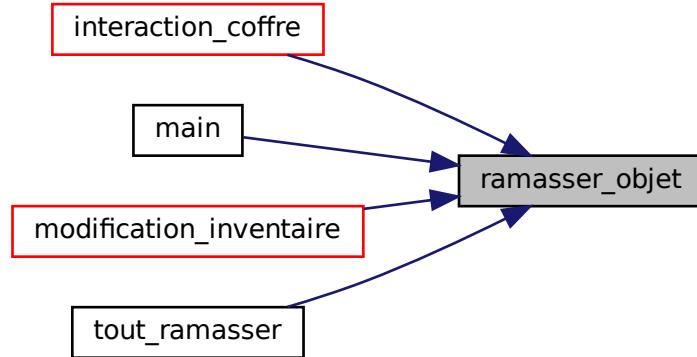
Auteur

Max Descomps

Paramètres

<i>objet</i>	Objet trouvé
<i>inventaire</i>	L'inventaire dans lequel mettre l'objet

Voici le graphe des appelants de cette fonction :



6.16.3.10 tout_ramasser()

```
void tout_ramasser (
    lobjet_t * objets,
    inventaire_t * inventaire )
```

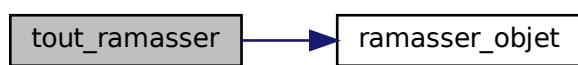
Auteur

Max Descomps

Paramètres

<i>objets</i>	Liste d'objets du jeu
<i>inventaire</i>	Inventaire du joueur

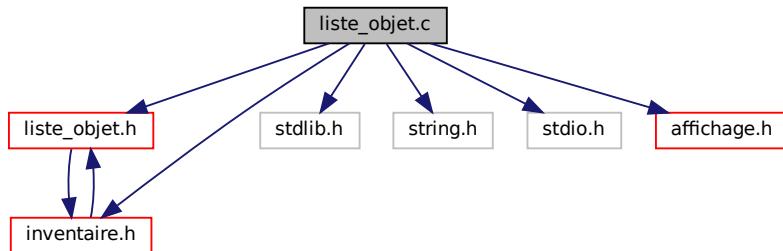
Voici le graphe d'appel pour cette fonction :



6.17 Référence du fichier liste_objet.c

Fonctions relatives aux structures contenant les objets.

Graphe des dépendances par inclusion de liste_objet.c:



Fonctions

- `lobjet_t * creer_liste_objet (char *fichier_src)`
Créé une liste d'objets et la remplit à l'aide du fichier des objets du jeu.
- `lobjet_t * creer_liste_objet_vide (char *fichier_src)`
Créé une liste d'objets vide.
- `lobjet_t * creer_liste_objet_equipe ()`
Créé une liste d'objets vide contenant les objets équipés.
- `void detruire_liste_objet (lobjet_t **liste_obj)`
Détruit une liste d'objets.
- `void effacer_liste_objet (lobjet_t **liste_obj)`
Efface une liste d'objets sans libérer la mémoire allouée aux objets qu'elle contient.
- `void afficher_liste_objet (lobjet_t *const liste_obj)`
Affiche une liste d'objets dans la console.
- `void placer_objet_sac (objet_t *objet, int slot)`
Place la texture d'un objet contenu dans une liste d'objet dans le sac depuis le menu inventaire selon son numéro de slot.
- `void afficher_textures_sac (inventaire_t *const inventaire)`
Affiche les textures des objets du sac dans le menu inventaire.
- `void afficher_textures_equipe (inventaire_t *const inventaire)`
Affiche les textures des objets équipés dans le menu inventaire.
- `void creer_textures_objets (lobjet_t *liste_obj)`
Créer toutes les textures d'une structure `lobjet_t`.

Variables

- `lobjet_t * objets = NULL`

6.17.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.17.2 Documentation des fonctions

6.17.2.1 afficher_liste_objet()

```
void afficher_liste_objet (
    lobjet_t *const liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	La liste d'objets à afficher
------------------	------------------------------

Voici le graphe d'appel pour cette fonction :



6.17.2.2 afficher_textures_equipe()

```
void afficher_textures_equipe (
    inventaire_t *const inventaire )
```

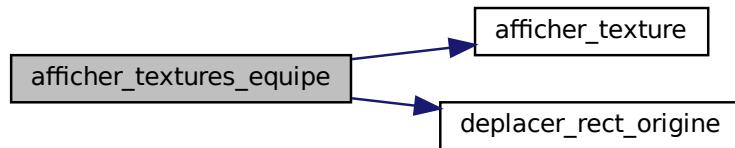
Auteur

Max Descomps

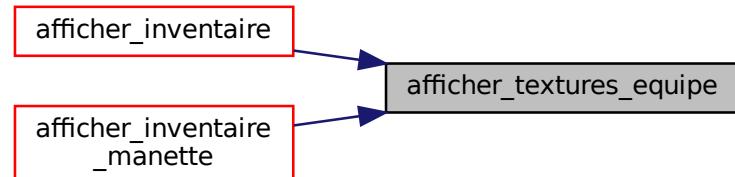
Paramètres

<i>inventaire</i>	L'inventaire contenant les objets équipés
-------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**6.17.2.3 afficher_textures_sac()**

```
void afficher_textures_sac (
    inventaire_t *const inventaire )
```

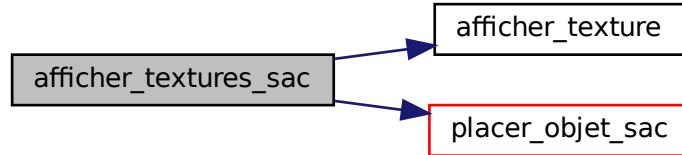
Auteur

Max Descomps

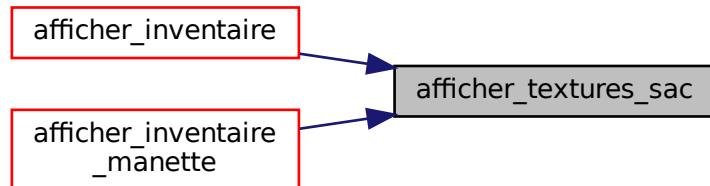
Paramètres

<i>inventaire</i>	L'inventaire contenant le sac
-------------------	-------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.17.2.4 creer_liste_objet()

```
lobjet_t* creer_liste_objet (  
    char * fichier_src )
```

Auteur

Max Descomps

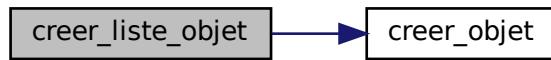
Paramètres

fichier_src	fichier source contenant la liste des objets
-------------	--

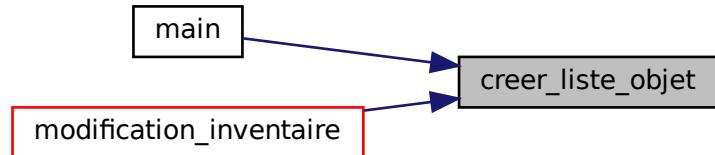
Renvoie

Instance nouvellement allouée du type `lobjet_t` ou NULL

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.17.2.5 `creer_liste_objet_equipe()`

```
lobjet_t* creer_liste_objet_equipe ( )
```

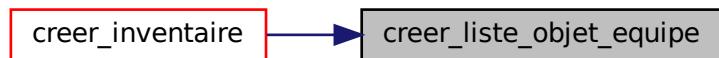
Auteur

Max Descomps

Renvoie

Instance nouvellement allouée du type `lobjet_t` ou NULL

Voici le graphe des appels de cette fonction :



6.17.2.6 creer_liste_objet_vide()

```
lobjet_t* creer_liste_objet_vide (  
    char * fichier_src )
```

Auteur

Max Descomps

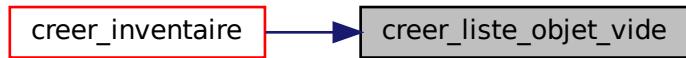
Paramètres

<i>fichier_src</i>	fichier source contenant la liste des objets pour en connaître le nombre
--------------------	--

Renvoie

Instance nouvellement allouée du type `lobjet_t` ou NULL

Voici le graphe des appels de cette fonction :



6.17.2.7 creer_textures_objets()

```
void creer_textures_objets (   
    lobjet_t * liste_obj )
```

Auteur

Max Descomps

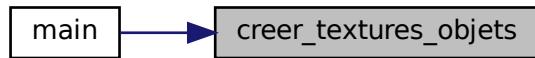
Paramètres

<i>liste_obj</i>	Pointeur sur la structure <code>lobjet_t</code>
------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.17.2.8 detruire_liste_objet()

```
void detruire_liste_objet (
    lobjet_t ** liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	La liste d'objets à détruire
------------------	------------------------------

Voici le graphe d'appel pour cette fonction :



6.17.2.9 effacer_liste_objet()

```
void effacer_liste_objet (
    lobjet_t ** liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	La liste d'objets à effacer
------------------	-----------------------------

Voici le graphe des appels de cette fonction :



6.17.2.10 placer_objet_sac()

```
void placer_objet_sac (
    objet_t * objet,
    int slot )
```

Auteur

Max Descomps

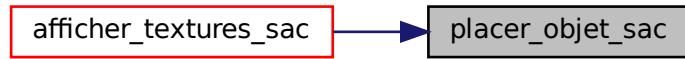
Paramètres

<i>objet</i>	L'objet dont on place la texture dans le sac
<i>slot</i>	le numéro de slot de l'objet

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.17.3 Documentation des variables

6.17.3.1 objets

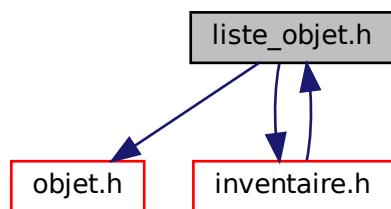
```
lobjet_t* objets = NULL
```

La liste des objets du jeu

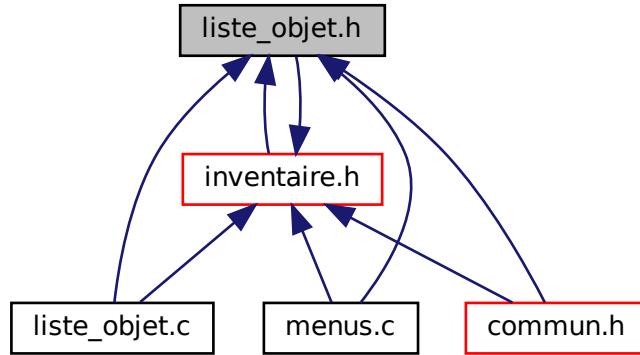
6.18 Référence du fichier liste_objet.h

Définition relatives aux structures contenant les objets.

Graphe des dépendances par inclusion de liste_objet.h:



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



Classes

- struct `lobjet_t`
Définition de la structure `lobjet`.

Fonctions

- `lobjet_t * creer_liste_objet (char *fichier_src)`
Créé une liste d'objets et la remplit à l'aide du fichier des objets du jeu.
- `lobjet_t * creer_liste_objet_vide (char *fichier_src)`
Créé une liste d'objets vide.
- `lobjet_t * creer_liste_objet_equipe ()`
Créé une liste d'objets vide contenant les objets équipés.
- `void detruire_liste_objet (lobjet_t **liste_obj)`
Détruit une liste d'objets.
- `void afficher_liste_objet (lobjet_t *const liste_obj)`
Affiche une liste d'objets dans la console.
- `void effacer_liste_objet (lobjet_t **liste_obj)`
Efface une liste d'objets sans libérer la mémoire allouée aux objets qu'elle contient.
- `void placer_objet_sac (objet_t *objet, int slot)`
Place la texture d'un objet contenu dans une liste d'objet dans le sac depuis le menu inventaire selon son numéro de slot.
- `void afficher_textures_sac (inventaire_t *const inventaire)`
Affiche les textures des objets du sac dans le menu inventaire.
- `void afficher_textures_equipe (inventaire_t *const inventaire)`
Affiche les textures des objets équipés dans le menu inventaire.
- `void creer_textures_objets (lobjet_t *liste_obj)`
Creer toutes les textures d'une structure `lobjet_t`.

Variables

- `lobjet_t * objets`

6.18.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

1.0

Date

28/03/2022

Copyright

Copyright (c) 2022

6.18.2 Documentation des fonctions

6.18.2.1 afficher_liste_objet()

```
void afficher_liste_objet (
    lobjet_t *const liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	La liste d'objets à afficher
------------------	------------------------------

Voici le graphe d'appel pour cette fonction :



6.18.2.2 afficher_textures_equipe()

```
void afficher_textures_equipe (
    inventaire_t *const inventaire )
```

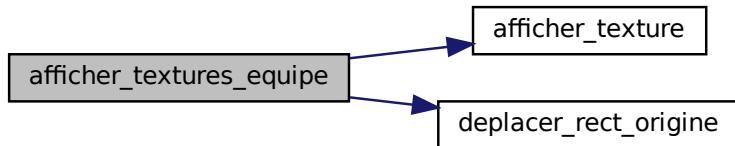
Auteur

Max Descomps

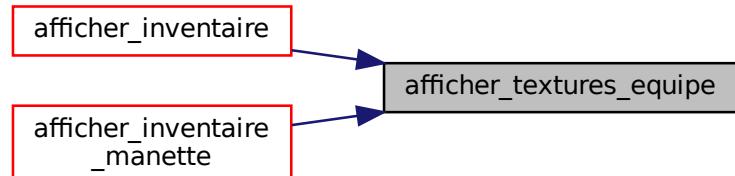
Paramètres

<i>inventaire</i>	L'inventaire contenant les objets équipés
-------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.18.2.3 afficher_textures_sac()

```
void afficher_textures_sac (
    inventaire_t *const inventaire )
```

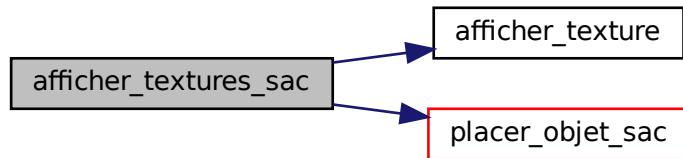
Auteur

Max Descomps

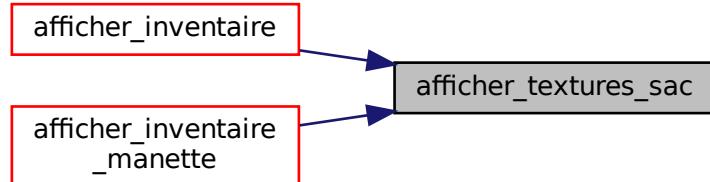
Paramètres

<i>inventaire</i>	L'inventaire contenant le sac
-------------------	-------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.18.2.4 `creer_liste_objet()`

```
lobjet_t* creer_liste_objet (
    char * fichier_src )
```

Auteur

Max Descomps

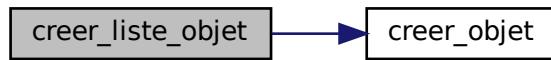
Paramètres

<i>fichier_src</i>	fichier source contenant la liste des objets
--------------------	--

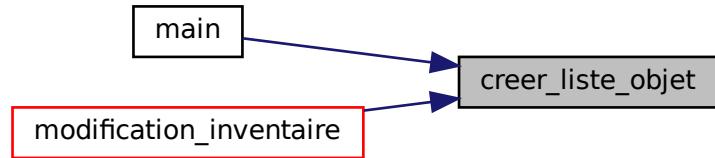
Renvoie

Instance nouvellement allouée du type `lobjet_t` ou NULL

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.18.2.5 `creer_liste_objet_equipe()`

```
lobjet_t* creer_liste_objet_equipe ( )
```

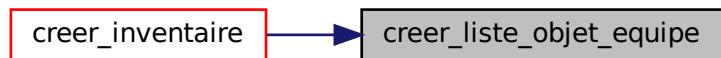
Auteur

Max Descomps

Renvoie

Instance nouvellement allouée du type `lobjet_t` ou NULL

Voici le graphe des appels de cette fonction :



6.18.2.6 creer_liste_objet_vide()

```
lobjet_t* creer_liste_objet_vide (
    char * fichier_src )
```

Auteur

Max Descomps

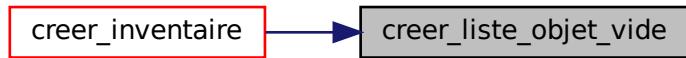
Paramètres

<i>fichier_src</i>	fichier source contenant la liste des objets pour en connaître le nombre
--------------------	--

Renvoie

Instance nouvellement allouée du type `lobjet_t` ou NULL

Voici le graphe des appelants de cette fonction :



6.18.2.7 creer_textures_objets()

```
void creer_textures_objets (
    lobjet_t * liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	Pointeur sur la structure <code>lobjet_t</code>
------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.18.2.8 detruire_liste_objet()

```
void detruire_liste_objet (
    lobjet_t ** liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	La liste d'objets à détruire
------------------	------------------------------

Voici le graphe d'appel pour cette fonction :



6.18.2.9 effacer_liste_objet()

```
void effacer_liste_objet (
    lobjet_t ** liste_obj )
```

Auteur

Max Descomps

Paramètres

<i>liste_obj</i>	La liste d'objets à effacer
------------------	-----------------------------

Voici le graphe des appels de cette fonction :



6.18.2.10 placer_objet_sac()

```
void placer_objet_sac (
    objet_t * objet,
    int slot )
```

Auteur

Max Descomps

Paramètres

<i>objet</i>	L'objet dont on place la texture dans le sac
<i>slot</i>	le numéro de slot de l'objet

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.18.3 Documentation des variables

6.18.3.1 objets

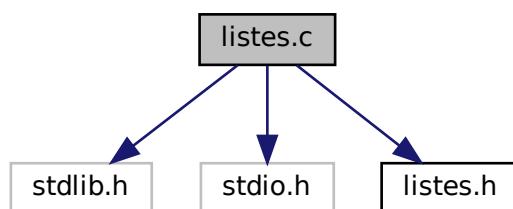
`lobjet_t* objets`

La liste des objets du jeu

6.19 Référence du fichier listes.c

Fonctions relatives au type de structure liste générique.

Graphe des dépendances par inclusion de listes.c:



Fonctions

- `list * init_liste (void *(*fonction_ajout)(void *), void(*fonction_suppresion)(void *), void(*fonction_← affichage)(void *))`
Fonction qui permet de créer une liste.
- `_Bool liste_vide (const list *const to_verify)`
Fonction booléenne qui permet de savoir si la liste est vide.
- `_Bool hors_liste (const list *const to_verify)`
Fonction booléenne qui permet de savoir si ll'on est actuellement hors de la liste.
- `void en_tete (list *mylist)`
Fonction qui permet de se placer en tête de la liste.
- `void en_queue (list *mylist)`
Fonction qui permet de se placer en queue de la liste.
- `void suivant (list *mylist)`
Fonction qui permet de passer à l'élément suivant dans la liste.
- `void precedent (list *mylist)`
Fonction qui permet de passer à l'élément suivant dans la liste.
- `void * valeur_elt (const list *const mylist)`
Fonction qui renvoie l'élément courant.
- `void modif_elt (list *mylist, void *v)`
Fonction qui permet de modifier l'élément courant.
- `void oter_elt (list *mylist)`
Fonction qui permet de supprimer un élément de la liste.
- `void ajout_droit (list *mylist, void *v)`
Fonction qui permet d'ajouter un élément à droite de l'élément courant.
- `void ajout_gauche (list *mylist, void *v)`
Fonction qui permet d'ajouter un élément à gauche de l'élément courant.
- `unsigned int taille_liste (const list *const mylist)`
Fonction qui renvoie le nombre d'éléments dans la liste.
- `void vider_liste (list *mylist)`
Fonction qui supprime tous les éléments de la liste.
- `void detruire_liste (list **liste)`
Détruit une liste en détruisant tous ses éléments.
- `void afficher_liste (list *liste)`
Affiche une liste d'objets génériques.
- `_Bool selectionner_element (list *liste, void *element, _Bool(*f_egalite)(void *, void *))`
Fonction qui permet de sélectionner l'élément courant à partir d'un élément précis.

6.19.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
 Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.19.2 Documentation des fonctions

6.19.2.1 afficher_liste()

```
void afficher_liste (
    list * liste )
```

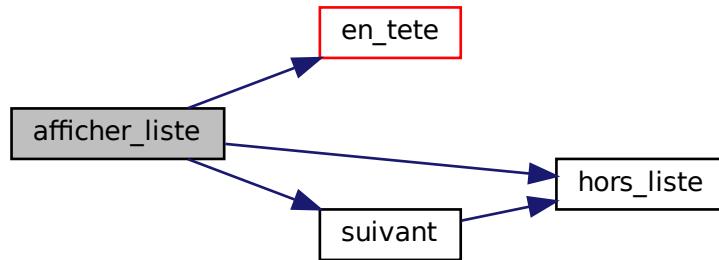
Auteur

Max Descomps

Paramètres

<i>liste</i>	Pointeur sur la liste
--------------	-----------------------

Voici le graphe d'appel pour cette fonction :



6.19.2.2 ajout_droit()

```
void ajout_droit (
    list * mylist,
    void * v )
```

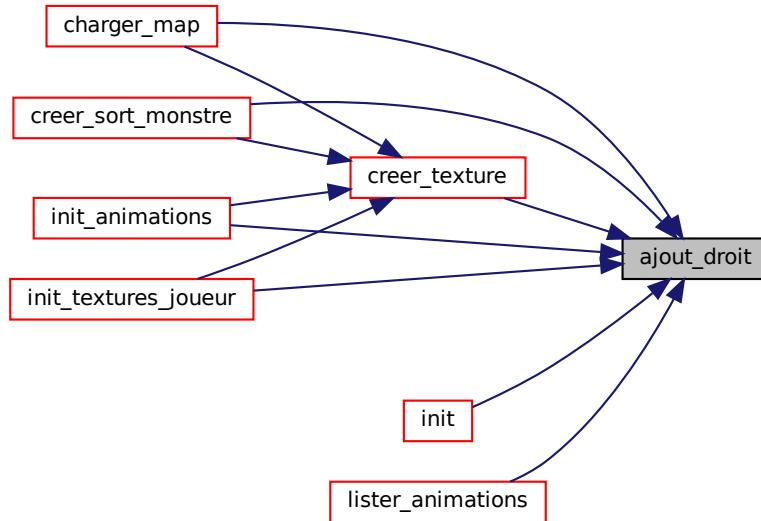
Auteur

Ange Despert

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

Voici le graphe des appelants de cette fonction :



6.19.2.3 ajout_gauche()

```
void ajout_gauche (
    list * mylist,
    void * v )
```

Auteur

Ange Despert

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

6.19.2.4 detruire_liste()

```
void detruire_liste (
    list ** liste )
```

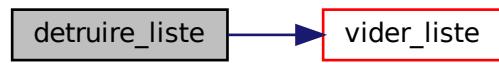
Auteur

Ange Despert

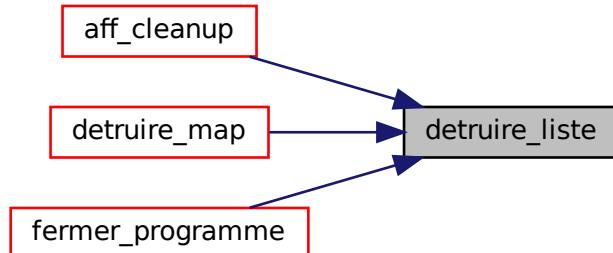
Paramètres

<i>liste</i>	Adresse de la liste
--------------	---------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**6.19.2.5 en_queue()**

```
void en_queue (
    list * mylist )
```

Auteur

Ange Despert

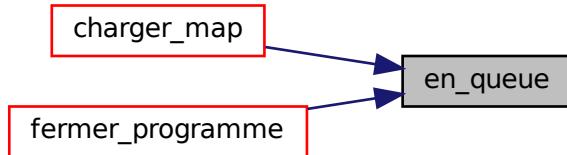
Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.19.2.6 en_tete()

```
void en_tete (list * mylist )
```

Auteur

Ange Despert

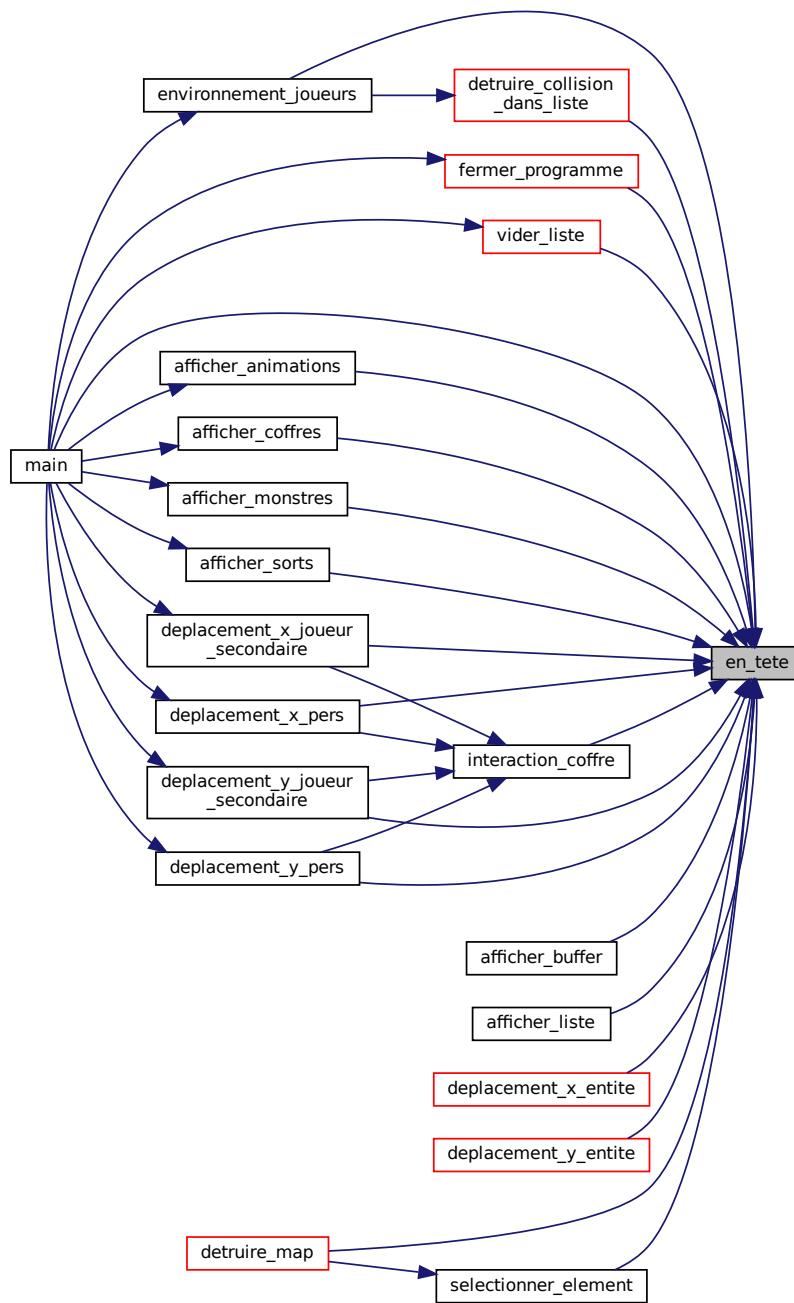
Paramètres

<code>mylist</code>	La liste dans laquelle on se déplace
---------------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.19.2.7 hors_liste()

```
_Bool hors_liste (
    const list *const to_verify )
```

Auteur

Ange Despert

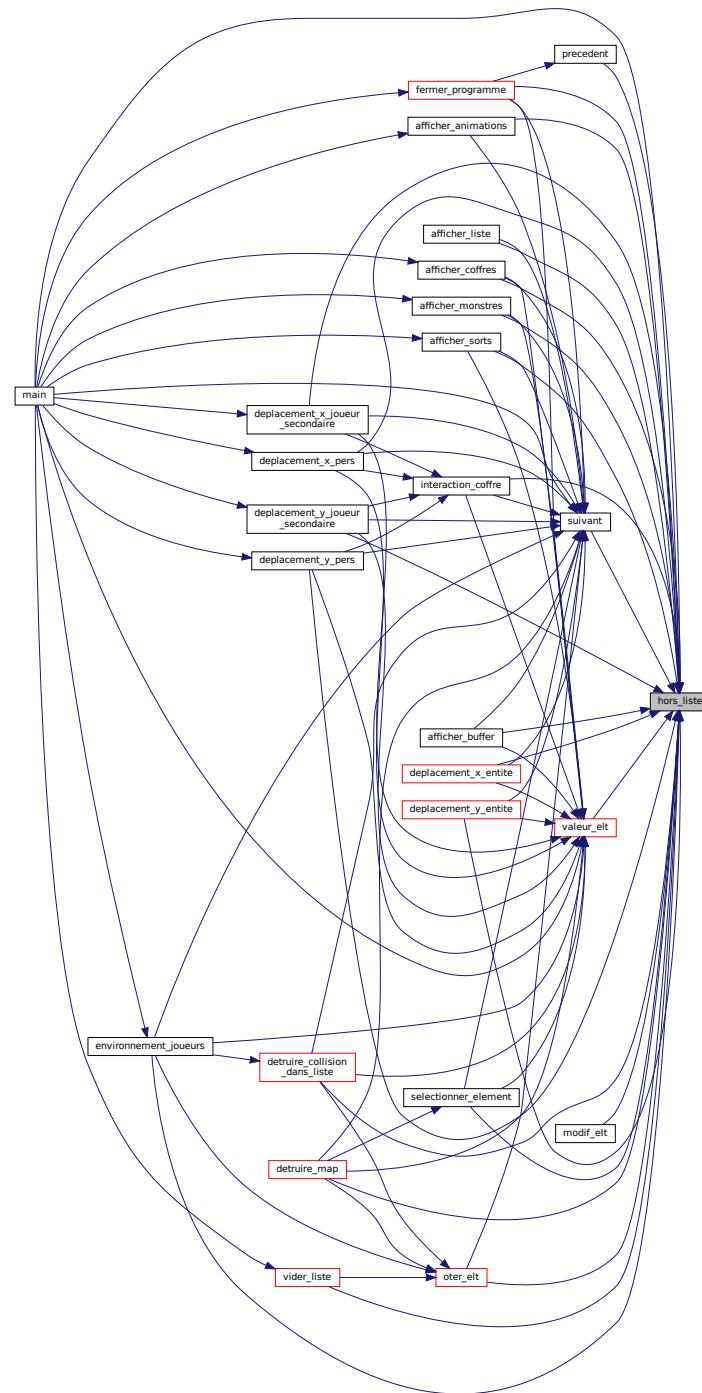
Paramètres

<i>to_verify</i>	La liste que l'on doit vérifier
------------------	---------------------------------

Renvoie

Vrai si on se trouve en dehors de la liste, faux sinon

Voici le graphe des appels de cette fonction :



6.19.2.8 init_liste()

```
list* init_liste (
    void *(*)(void *) fonction_ajout,
    void(*)(void *) f_suppression,
    void(*)(void *) fonction_affichage )
```

Auteurs

Ange Despert
Max Descomps

On peut préciser des fonction pour l'insertion et la supréssion des objets.

Mais on ne peut pas avoir une fonction d'insertion et aucune fonction de supréssion et vice-versa.

Par défaut la liste fonctionne par référencement, pour cela il suffit de ne pas fournir de fonctions dans l'appel de la fonction.

On attend des éléments dynamiques dans la liste

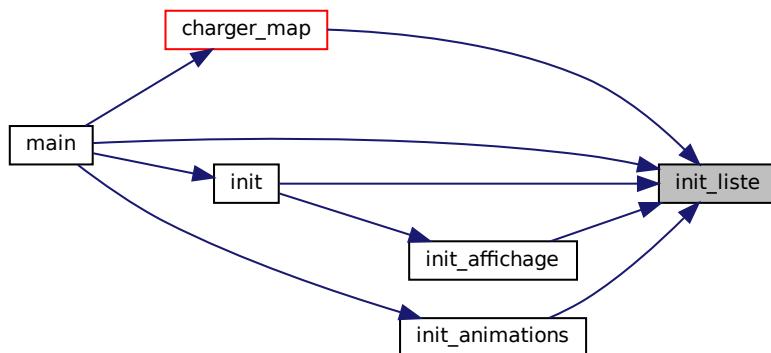
Paramètres

<i>fonction_ajout</i>	La fonction qui permet d'intertion des objets, NULL sinon
<i>f_suppresion</i>	La fonction qui permet la suppression des objets, NULL sinon
<i>fonction_affichage</i>	La fonction qui permet d'afficher les objets de la liste, NULL sinon

Renvoie

La liste qui vient d'être crée, NULL s'il y a eu une erreur

Voici le graphe des appels de cette fonction :

**6.19.2.9 liste_vide()**

```
_Bool liste_vide (
    const list *const to_verify )
```

Auteur

Ange Despert

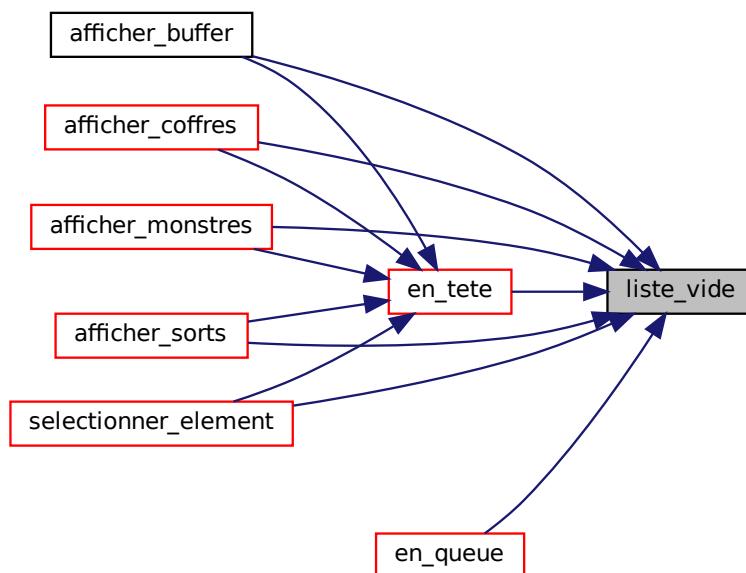
Paramètres

<i>to_verify</i>	La liste que l'on doit vérifier
------------------	---------------------------------

Renvoie

Vrai si la liste est vide, faux sinon

Voici le graphe des appels de cette fonction :

**6.19.2.10 modif_elt()**

```
void modif_elt (
    list * mylist,
    void * v )
```

Auteur

Ange Despert

ATTENTION : si les éléments sont placés dans la liste par référencement cela détruira l'élément précédent !

Paramètres

<i>mylist</i>	La liste donc on veut modifier l'élément courant
<i>v</i>	L'élément à mettre à la place de l'ancien

Voici le graphe d'appel pour cette fonction :



6.19.2.11 oter_elt()

```
void oter_elt (
    list * mylist )
```

Auteur

Ange Despert

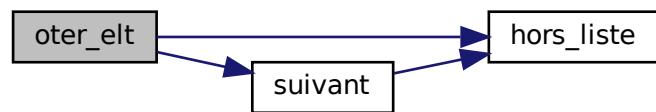
ATTENTION : si les éléments sont placés dans la liste par référencement cela détruira l'élément (ne pas y accéder après) !

On attend des éléments dynamiques dans la liste.

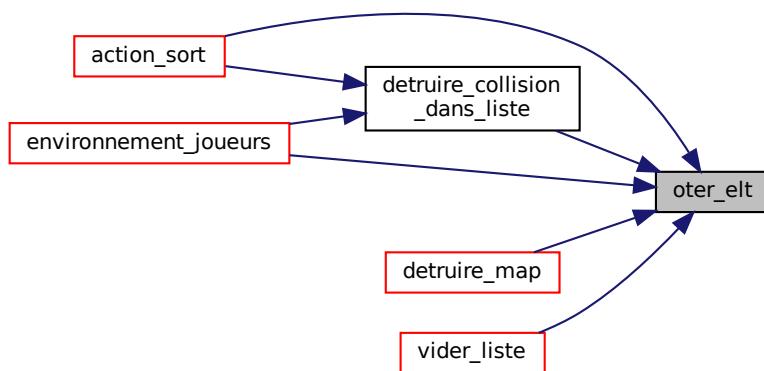
Paramètres

<i>mylist</i>	La liste dont on veut oter l'élément
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.19.2.12 precedent()

```
void precedent (
    list * mylist )
```

Auteur

Ange Despert

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.19.2.13 selectionner_element()

```
_Bool selectionner_element (
    list * liste,
    void * element,
    _Bool(*)(void *, void *) f_egalite )
```

Auteur

Ange Despert

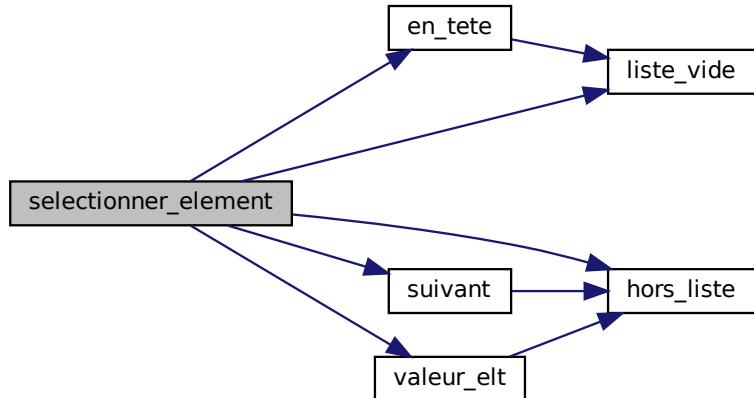
Paramètres

<i>liste</i>	La liste dans laquelle on veut faire la recherche
<i>element</i>	L'élément que l'on veut rechercher
<i>f_egalite</i>	Une fonction de comparaison qui renvoie vrai si les deux valeurs sont égales ou NULL si on veut comparer les adresses

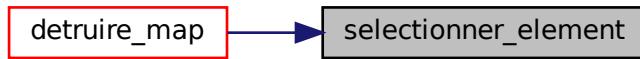
Renvoie

Vrai si l'élément à été trouvé

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.19.2.14 suivant()

```
void suivant (
    list * mylist )
```

Auteur

Ange Despert

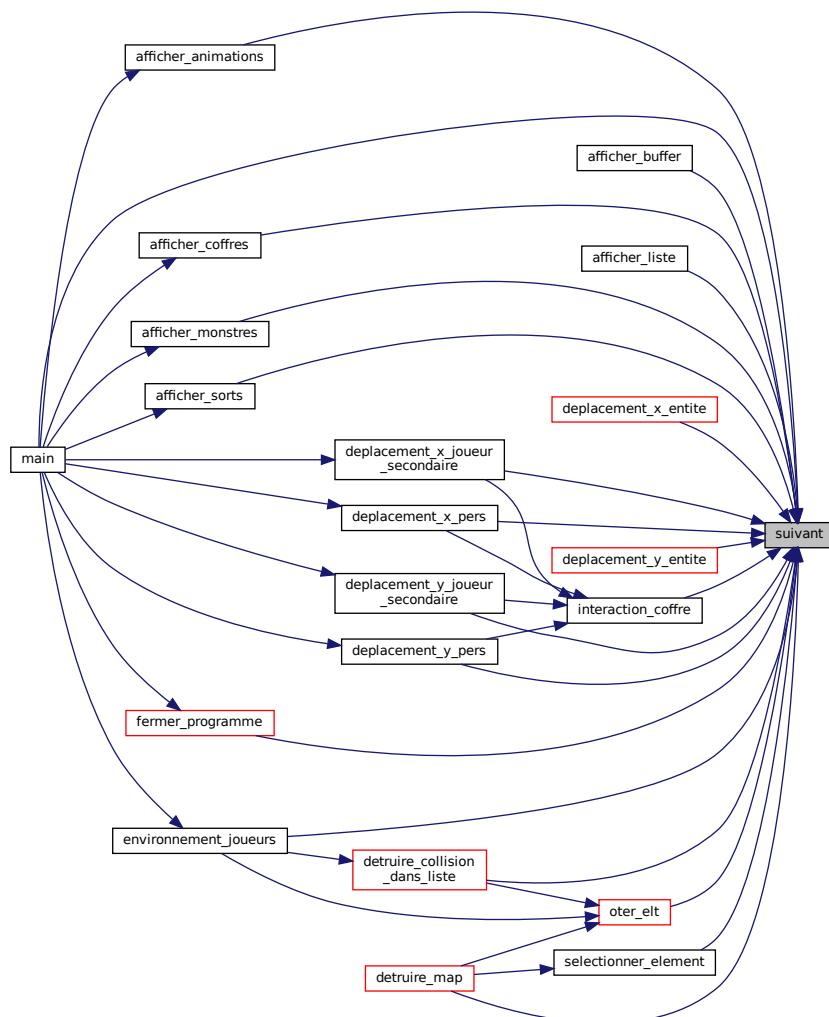
Paramètres

<code>mylist</code>	La liste dans laquelle on se déplace
---------------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.19.2.15 taille_liste()

```

unsigned int taille_liste (
    const list *const mylist )
  
```

Auteur

Ange Despert

 Paramètres

<i>mylist</i>	La liste dont on veut connaître le nombre d'éléments
---------------	--

 Renvoie

Le nombre d'éléments dans la liste

 6.19.2.16 valeur_elt()

```
void* valeur_elt (
    const list *const mylist )
```

 Auteur

Ange Despert

 Paramètres

<i>mylist</i>	La liste dont on veut l'élément
---------------	---------------------------------

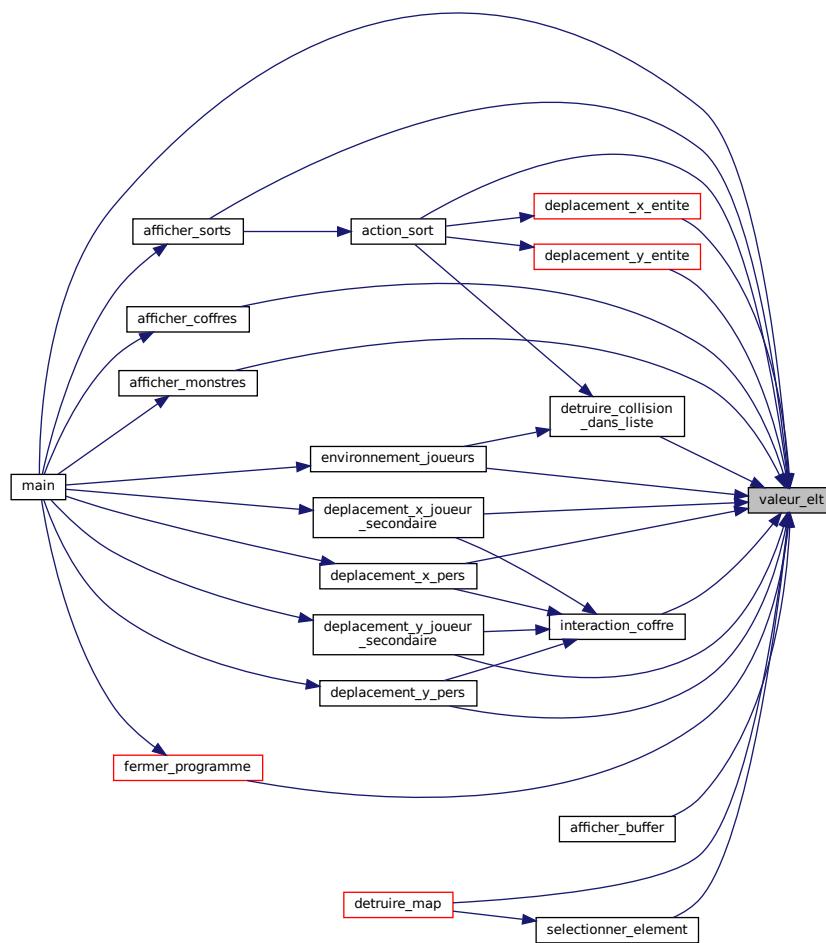
 Renvoie

L'élément que l'on convoite

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.19.2.17 vider_liste()

```
void vider_liste (
    list * mylist )
```

Auteur

Ange Despert

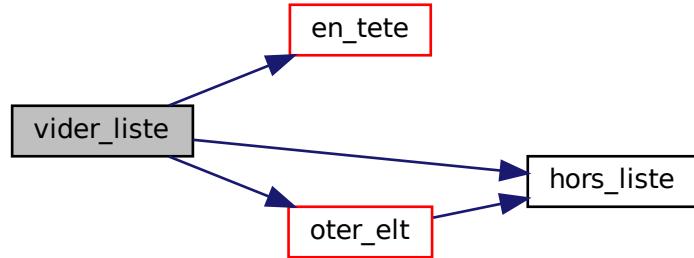
ATTENTION : si les éléments sont insérés par référencement, ils seront tous détruits !

On attend des éléments dynamiques dans la liste.

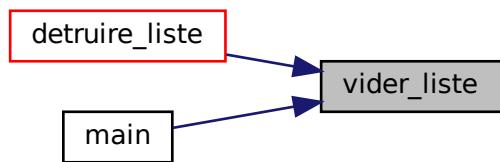
Paramètres

<i>mylist</i>	La liste que l'on veut vider
---------------	------------------------------

Voici le graphe d'appel pour cette fonction :



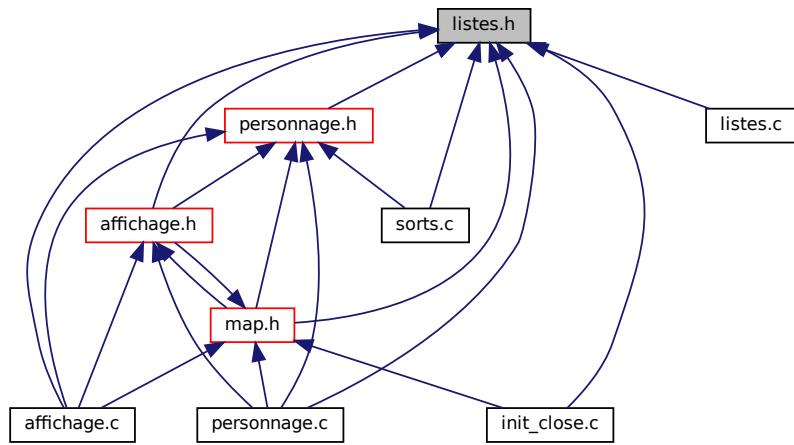
Voici le graphe des appellants de cette fonction :



6.20 Référence du fichier listes.h

Fonction relatives au type de structure liste générique.

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



Classes

- struct **element**
Définition du type **element** : un élément d'une liste.
- struct **list**
Définition du type **list** : une liste générique.

Fonctions

- **list * init_liste** (void *(*fonction_ajout)(void *), void(*f_suppresion)(void *), void(*fonction_affichage)(void *))
Fonction qui permet de créer une liste.
- **_Bool liste_vide** (const **list** *const to_verify)
Fonction booléenne qui permet de savoir si la liste est vide.
- **_Bool hors_liste** (const **list** *const to_verify)
Fonction booléenne qui permet de savoir si l'on est actuellement hors de la liste.
- **void en_tete** (**list** *mylist)
Fonction qui permet de se placer en tête de la liste.
- **void en_queue** (**list** *mylist)
Fonction qui permet de se placer en queue de la liste.
- **void suivant** (**list** *mylist)
Fonction qui permet de passer à l'élément suivant dans la liste.
- **void précédent** (**list** *mylist)
Fonction qui permet de passer à l'élément suivant dans la liste.
- **void * valeur_elt** (const **list** *const mylist)
Fonction qui renvoie l'élément courant.
- **void modif_elt** (**list** *mylist, void *)
Fonction qui permet de modifier l'élément courant.
- **void oter_elt** (**list** *mylist)
Fonction qui permet de supprimer un élément de la liste.
- **void ajout_droit** (**list** *mylist, void *)
Fonction qui permet d'ajouter un élément à droite de l'élément courant.
- **void ajout_gauche** (**list** *mylist, void *)
Fonction qui permet d'ajouter un élément à gauche de l'élément courant.
- **unsigned int taille_liste** (const **list** *const mylist)
Fonction qui renvoie le nombre d'éléments dans la liste.
- **void vider_liste** (**list** *mylist)
Fonction qui supprime tous les éléments de la liste.
- **void detruire_liste** (**list** **liste)
Détruit une liste en détruisant tout ses éléments.

- void **afficher_liste** (*list *liste*)
Affiche une liste d'objets génériques.
- _Bool **selectionner_element** (*list *liste, void *element, _Bool(*f_egalite)(void *, void *)*)
Fonction qui permet de sélectionner l'élément courant à partir d'un élément précis.

6.20.1 Description détaillée

Auteurs

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

2.0

Date

03/04/2022

Copyright

Copyright (c) 2022

6.20.2 Documentation des fonctions

6.20.2.1 afficher_liste()

```
void afficher_liste (
    list * liste )
```

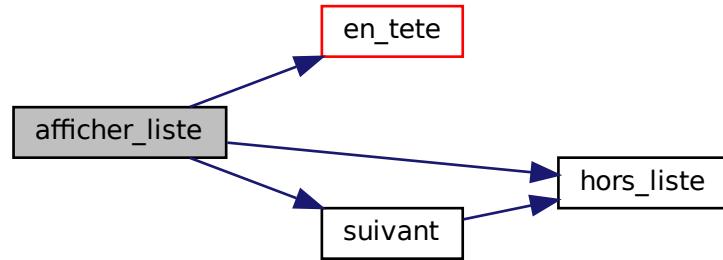
Auteur

Max Descomps

Paramètres

<i>liste</i>	Pointeur sur la liste
--------------	-----------------------

Voici le graphe d'appel pour cette fonction :



6.20.2.2 ajout_droit()

```
void ajout_droit (
    list * mylist,
    void * v )
```

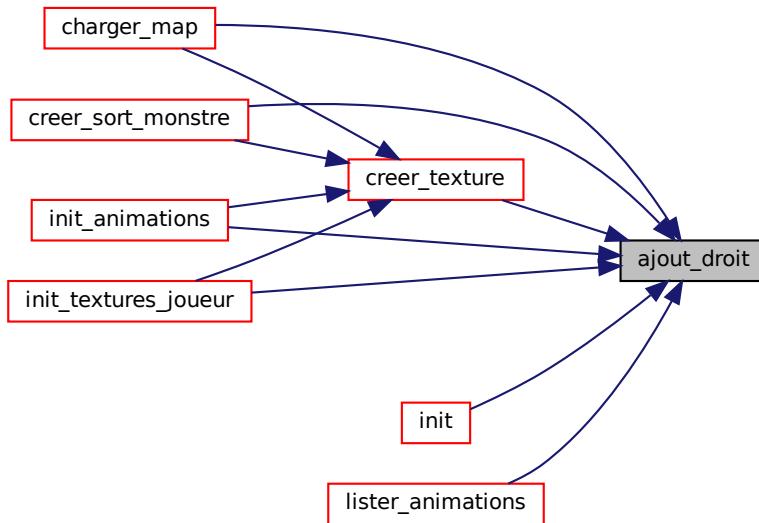
Auteur

Ange Despert

Paramètres

<code>mylist</code>	La liste où on veut ajouter l'élément
<code>v</code>	L'élément que l'on veut ajouter

Voici le graphe des appelants de cette fonction :



6.20.2.3 ajout_gauche()

```
void ajout_gauche (
    list * mylist,
    void * v )
```

Auteur

Ange Despert

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

6.20.2.4 detruire_liste()

```
void detruire_liste (
    list ** liste )
```

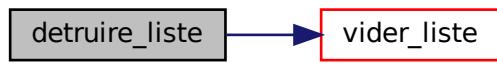
Auteur

Ange Despert

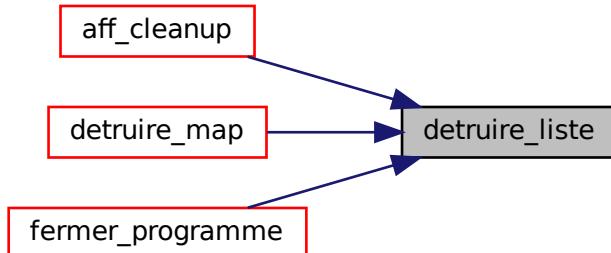
Paramètres

<i>liste</i>	Adresse de la liste
--------------	---------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**6.20.2.5 en_queue()**

```
void en_queue (list * mylist )
```

Auteur

Ange Despert

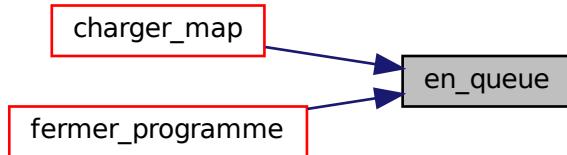
Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.20.2.6 en_tete()

```
void en_tete (list * mylist )
```

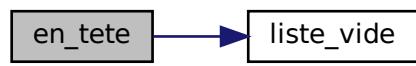
Auteur

Ange Despert

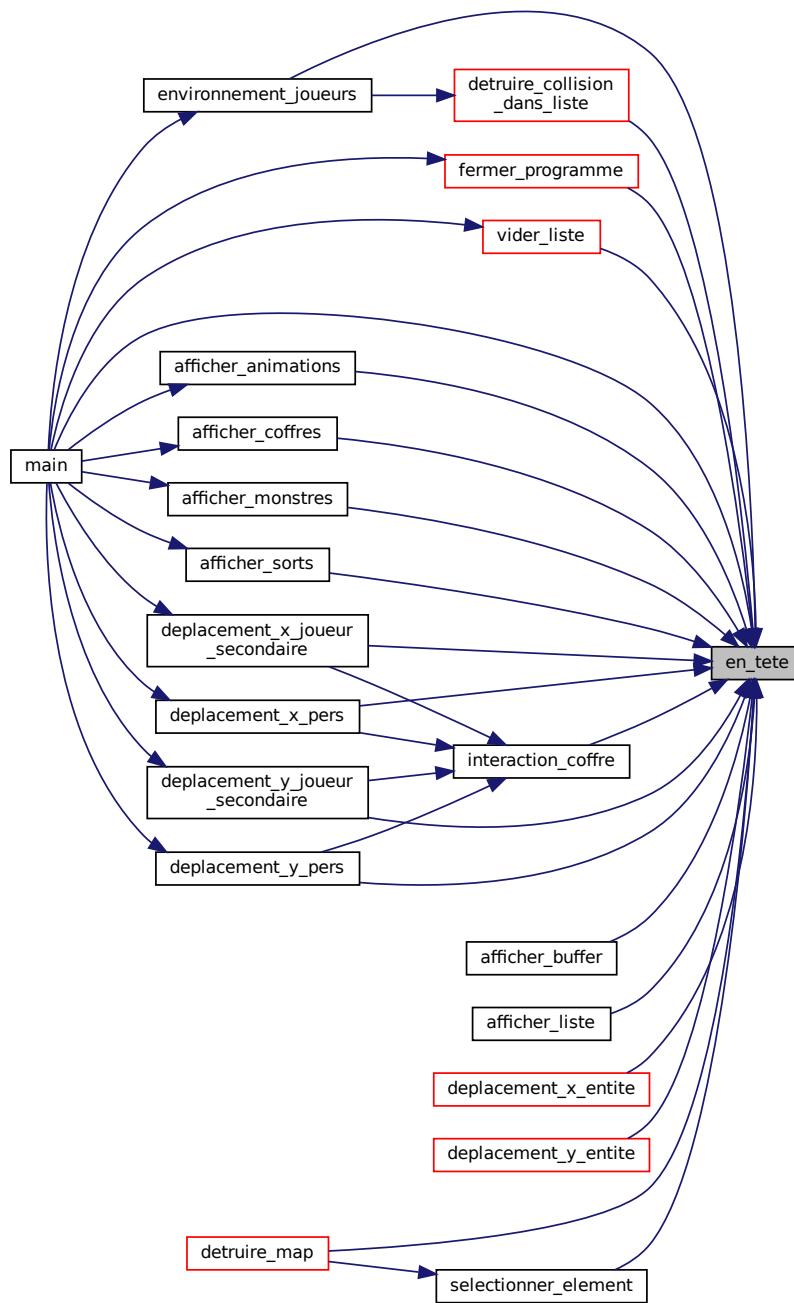
Paramètres

<code>mylist</code>	La liste dans laquelle on se déplace
---------------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.20.2.7 hors_liste()

```
_Bool hors_liste (
    const list *const to_verify )
```

Auteur

Ange Despert

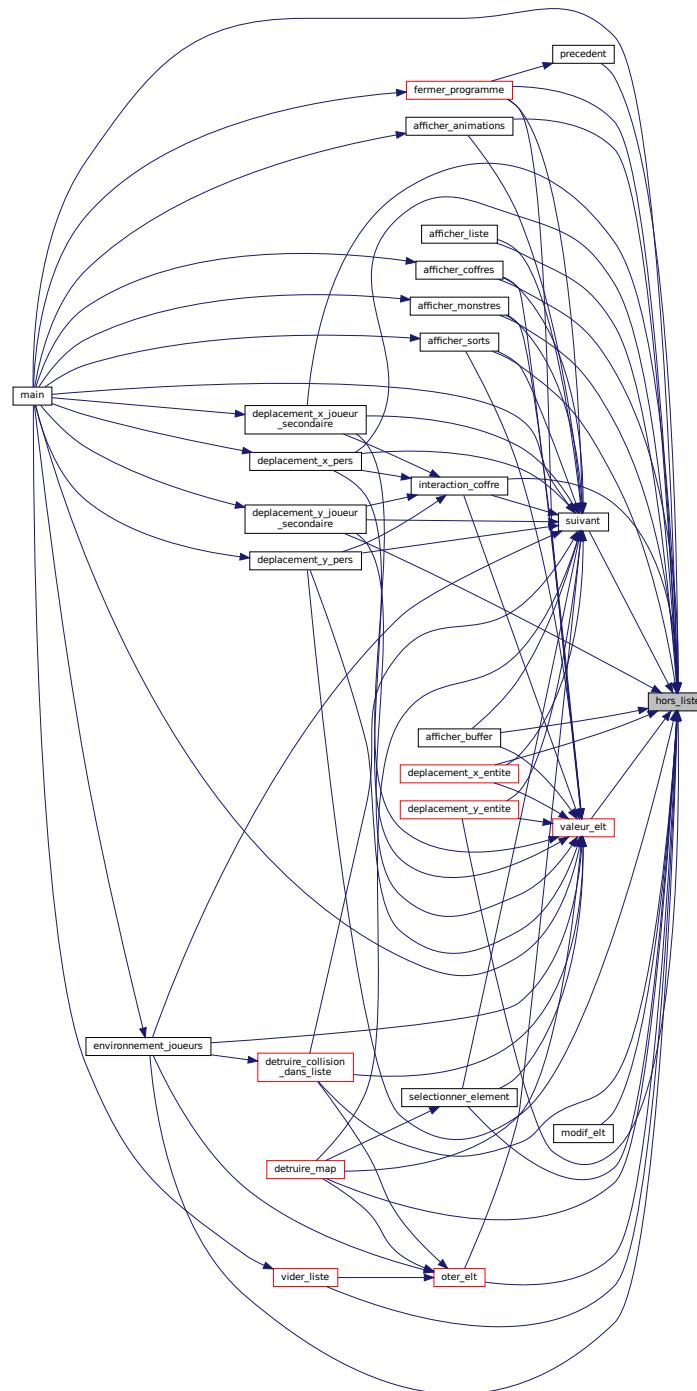
Paramètres

<i>to_verify</i>	La liste que l'on doit vérifier
------------------	---------------------------------

Renvoie

Vrai si on se trouve en dehors de la liste, faux sinon

Voici le graphe des appels de cette fonction :



6.20.2.8 init_liste()

```

list* init_liste (
    void *(*)(void *) fonction_ajout,
    void(*)(void *) f_suppression,
    void(*)(void *) fonction_affichage )

```

Auteurs

Ange Despert
Max Descomps

On peut préciser des fonction pour l'insertion et la supréssion des objets.

Mais on ne peut pas avoir une fonction d'insertion et aucune fonction de supréssion et vice-versa.

Par défaut la liste fonctionne par référencement, pour cela il suffit de ne pas fournir de fonctions dans l'appel de la fonction.

On attend des éléments dynamiques dans la liste

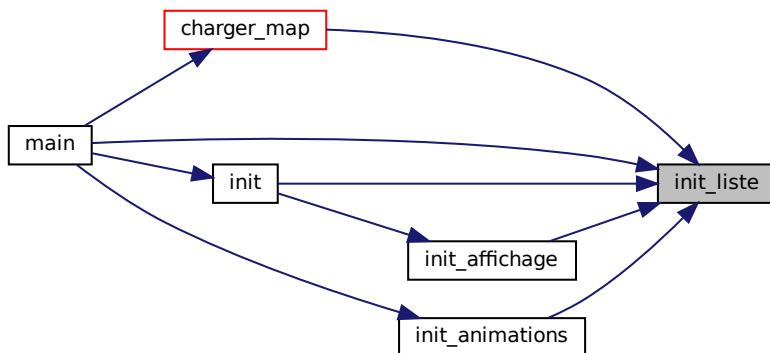
Paramètres

<i>fonction_ajout</i>	La fonction qui permet d'intertion des objets, NULL sinon
<i>f_suppresion</i>	La fonction qui permet la suppression des objets, NULL sinon
<i>fonction_affichage</i>	La fonction qui permet d'afficher les objets de la liste, NULL sinon

Renvoie

La liste qui vient d'être crée, NULL s'il y a eu une erreur

Voici le graphe des appels de cette fonction :

**6.20.2.9 liste_vide()**

```
_Bool liste_vide (
    const list *const to_verify )
```

Auteur

Ange Despert

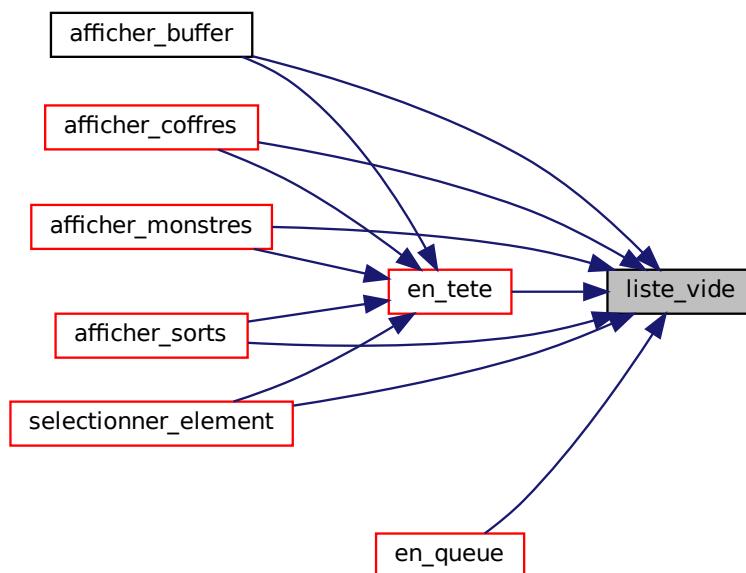
Paramètres

<i>to_verify</i>	La liste que l'on doit vérifier
------------------	---------------------------------

Renvoie

Vrai si la liste est vide, faux sinon

Voici le graphe des appels de cette fonction :

**6.20.2.10 modif_elt()**

```
void modif_elt (
    list * mylist,
    void * v )
```

Auteur

Ange Despert

ATTENTION : si les éléments sont placés dans la liste par référencement cela détruira l'élément précédent !

Paramètres

<i>mylist</i>	La liste donc on veut modifier l'élément courant
<i>v</i>	L'élément à mettre à la place de l'ancien

Voici le graphe d'appel pour cette fonction :



6.20.2.11 oter_elt()

```
void oter_elt (
    list * mylist )
```

Auteur

Ange Despert

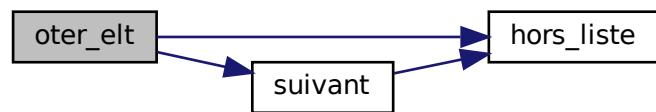
ATTENTION : si les éléments sont placés dans la liste par référencement cela détruira l'élément (ne pas y accéder après) !

On attend des éléments dynamiques dans la liste.

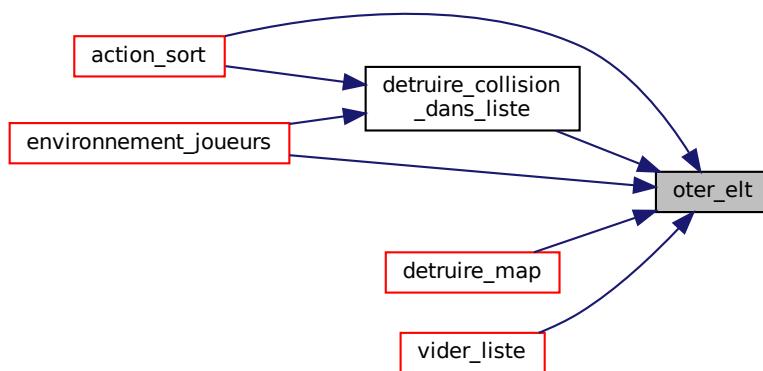
Paramètres

<i>mylist</i>	La liste dont on veut oter l'élément
---------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.20.2.12 precedent()

```
void precedent (list * mylist )
```

Auteur

Ange Despert

Paramètres

<code>mylist</code>	La liste dans laquelle on se déplace
---------------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.20.2.13 selectionner_element()

```
_Bool selectionner_element (
    list * liste,
    void * element,
    _Bool(*)(void *, void *) f_egalite )
```

Auteur

Ange Despert

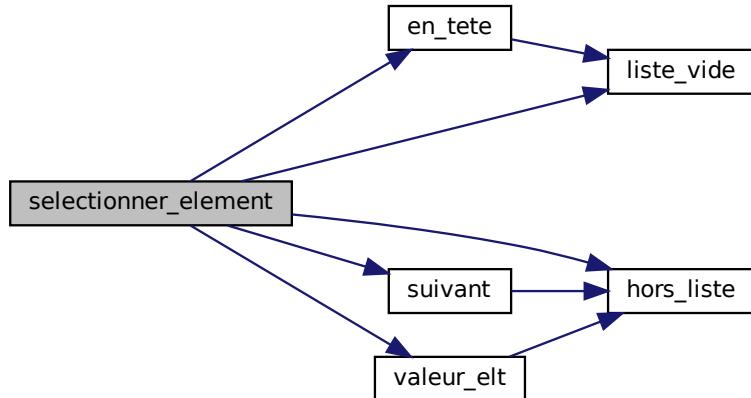
Paramètres

<i>liste</i>	La liste dans laquelle on veut faire la recherche
<i>element</i>	L'élément que l'on veut rechercher
<i>f_egalite</i>	Une fonction de comparaison qui renvoie vrai si les deux valeurs sont égales ou NULL si on veut comparer les adresses

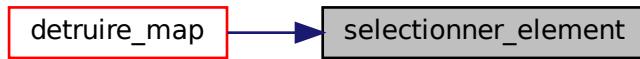
Renvoie

Vrai si l'élément à été trouvé

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.20.2.14 suivant()

```
void suivant (
    list * mylist )
```

Auteur

Ange Despert

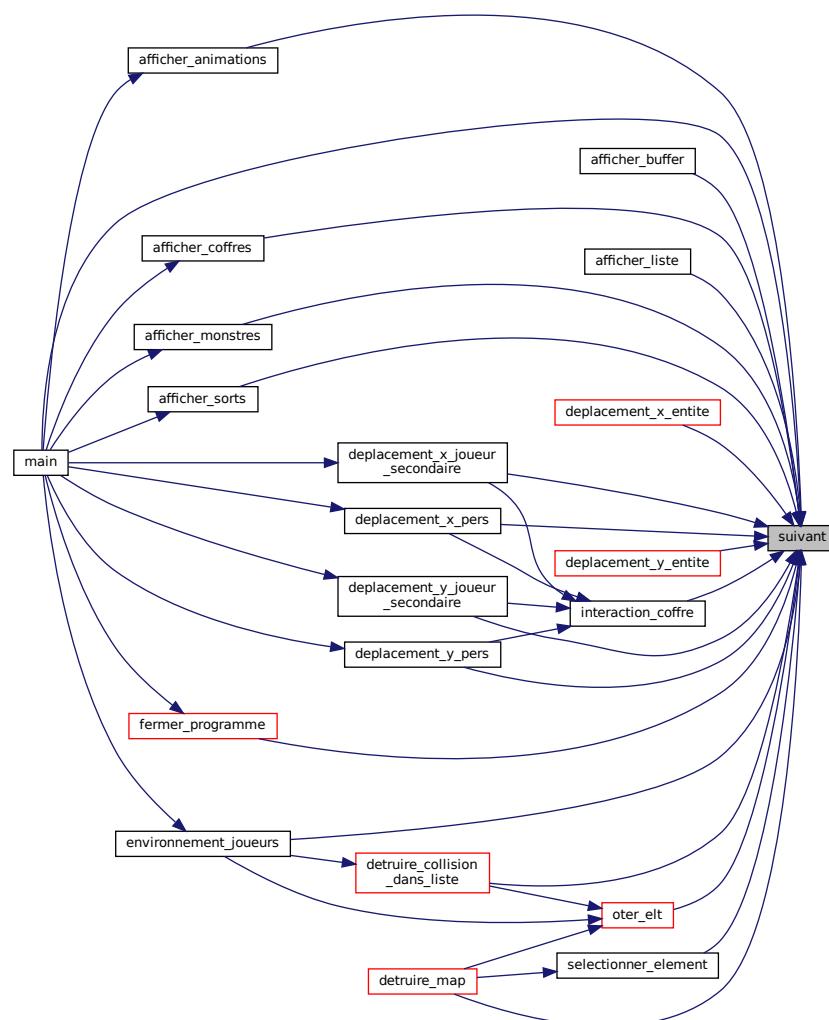
Paramètres

<code>mylist</code>	La liste dans laquelle on se déplace
---------------------	--------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.20.2.15 taille_liste()

```

unsigned int taille_liste (
    const list *const mylist )
  
```

Auteur

Ange Despert

 Paramètres

<i>mylist</i>	La liste dont on veut connaître le nombre d'éléments
---------------	--

 Renvoie

Le nombre d'éléments dans la liste

 6.20.2.16 valeur_elt()

```
void* valeur_elt (
    const list *const mylist )
```

 Auteur

Ange Despert

 Paramètres

<i>mylist</i>	La liste dont on veut l'élément
---------------	---------------------------------

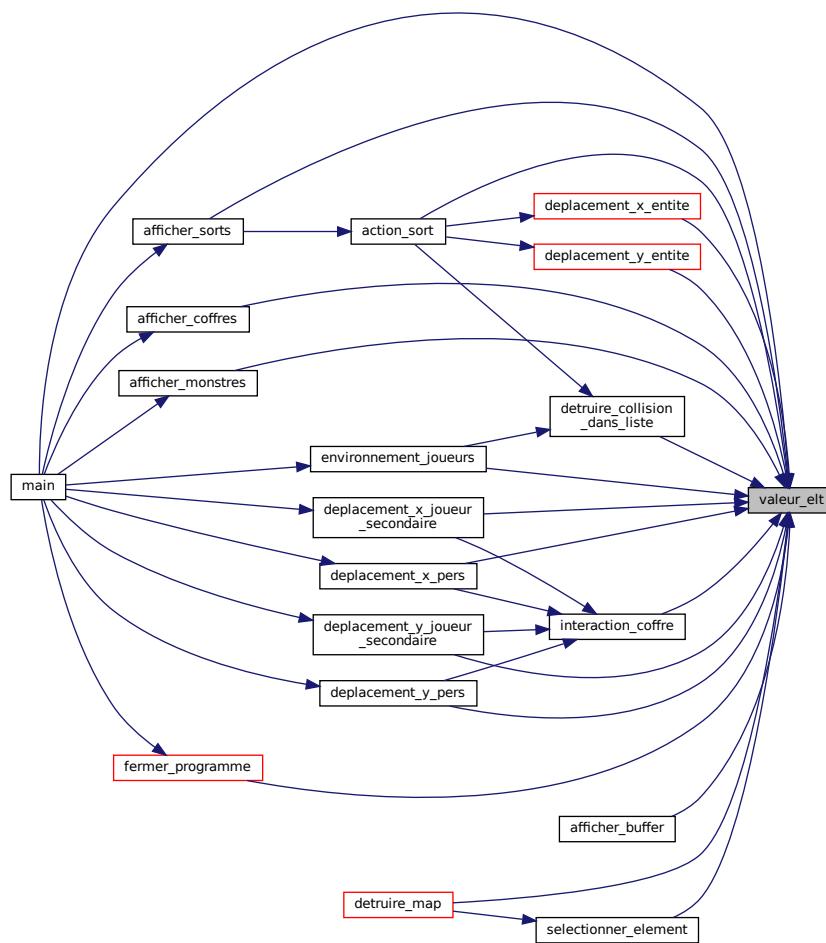
 Renvoie

L'élément que l'on convoite

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.20.2.17 vider_liste()

```
void vider_liste (
    list * mylist )
```

Auteur

Ange Despert

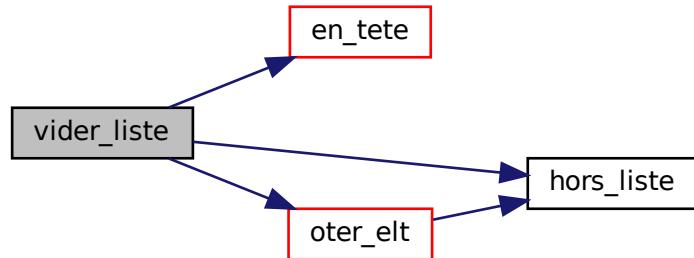
ATTENTION : si les éléments sont insérés par référencement, ils seront tous détruits !

On attend des éléments dynamiques dans la liste.

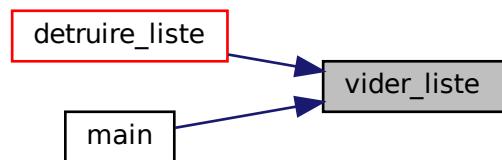
Paramètres

<i>mylist</i>	La liste que l'on veut vider
---------------	------------------------------

Voici le graphe d'appel pour cette fonction :



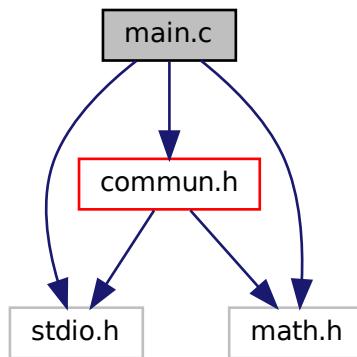
Voici le graphe des appellants de cette fonction :



6.21 Référence du fichier main.c

Programme principal du jeu.

Graphe des dépendances par inclusion de main.c:



Macros

— #define `SDL_MAIN_HANDLED`

Fonctions

— void `afficher_intro` (void)
— int `main` (int argc, char **argv)

6.21.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.21.2 Documentation des macros

6.21.2.1 SDL_MAIN_HANDLED

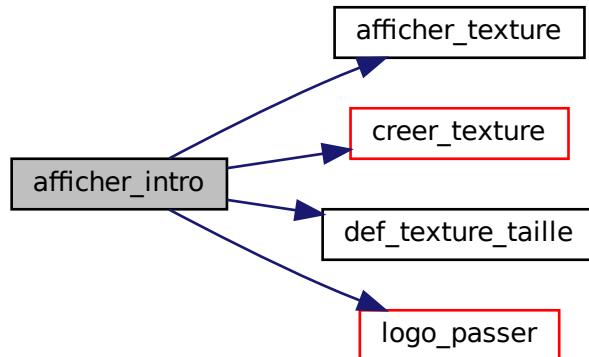
```
#define SDL_MAIN_HANDLED
```

6.21.3 Documentation des fonctions

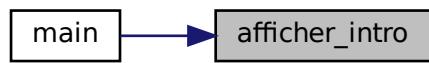
6.21.3.1 afficher_intro()

```
void afficher_intro (
    void )
```

Voici le graphe d'appel pour cette fonction :



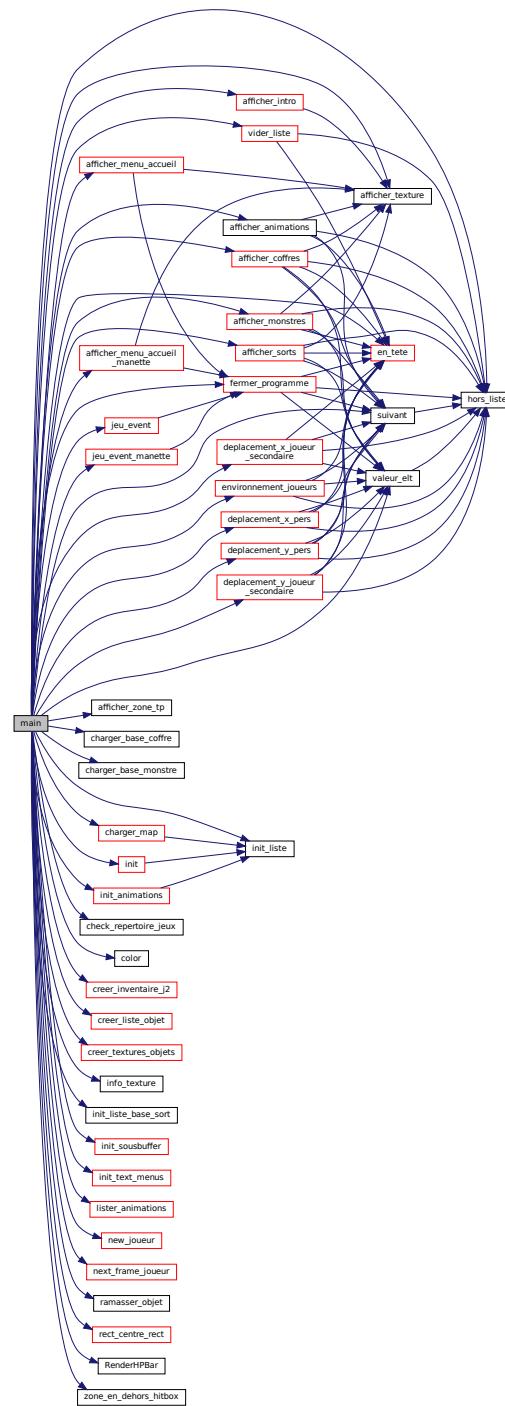
Voici le graphe des appels de cette fonction :



6.21.3.2 main()

```
int main (
    int argc,
    char ** argv )
```

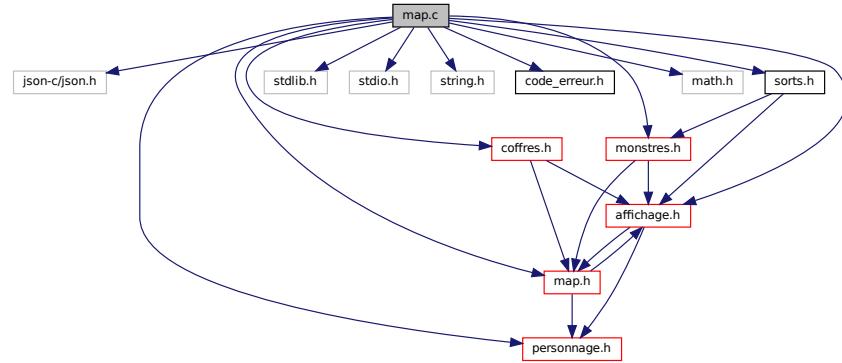
<Si l'écran doit apparaître avec un fonduVoici le graphe d'appel pour cette fonction :



6.22 Référence du fichier map.c

Fonctions de gestion de la map.

Graphe des dépendances par inclusion de map.c:



Fonctions

- `void init_sousbuffer (t_map *map, joueur_t *joueur)`
Fonction qui initialise les textures qui serviront de buffer d'affichage au moteur du jeu.
- `SDL_Rect taille_ecran_cases ()`
- `t_map * charger_map (const char *const nom_map)`
Fonction qui récupère les informations stockées dans le fichier don le nom est donne en entrée.
- `t_aff * texture_map (const t_map *map)`
Fonction qui renvoie la texture de la map.
- `void detruire_map (t_map **map, joueur_t *joueurs[], unsigned short int nb_joueurs)`
Fonction qui permet de détruire une map.
- `void transition (t_map **actuelle, unsigned int num_map, joueur_t **joueurs, unsigned short int nb_joueurs, unsigned int new_x, unsigned int new_y)`
- `void tp_joueurs (t_map *map, unsigned int x, unsigned int y, joueur_t **joueurs, unsigned short int nb_joueurs)`
- `void afficher_zone_tp (zone_tp *z)`
- `bool hors_map_monstre (SDL_Rect *collision, t_map *map)`
Fonction qui regarde si l'entité est hors_map.

Variables

- `t_map * map`

6.22.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.22.2 Documentation des fonctions

6.22.2.1 afficher_zone_tp()

```
void afficher_zone_tp (
    zone_tp * z )
```

Voici le graphe des appels de cette fonction :



6.22.2.2 charger_map()

```
t_map* charger_map (
    const char *const nom_map )
```

Auteur

Ange Despert

Fonction qui fait appel à la librairie JSON-C pour charger les informations de la map au format JSON.

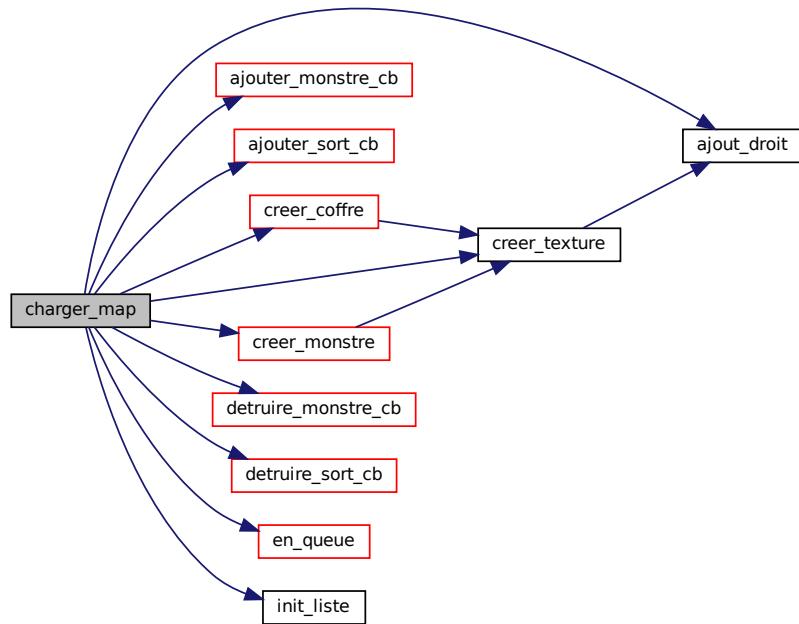
Paramètres

<i>nom_map</i>	Le nom du fichier qui contient les informations sur la map
----------------	--

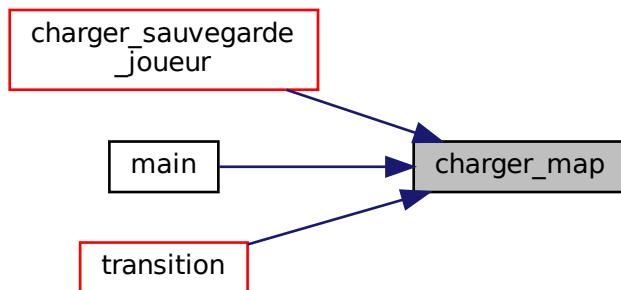
Renvoie

Une map initialisée avec toutes les informations dedans

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.22.2.3 detruire_map()

```
void detruire_map (
    t_map ** map,
    joueur_t * joueurs[],
    unsigned short int nb_joueurs )
```

Auteur

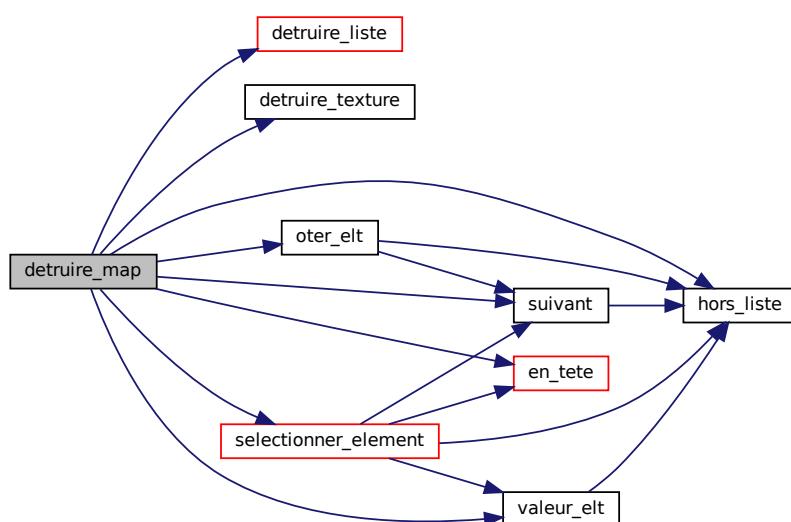
Ange Despert

Cette fonction libérera toute la mémoire allouée à la map. Elle détruira également les éléments des listes de la map. En ce qui concerne la liste de collision, étant donné que certains rectangle ne sont pas dynamiques, la fonction va d'abord s'assurer qu'ils peuvent être supprimer.

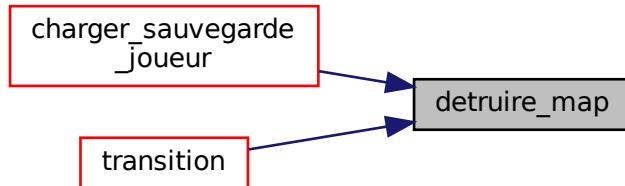
Paramètres

<i>map</i>	L'adresse de la map que l'on veut détruire
<i>joueurs</i>	Les joueurs qui existent
<i>nb_joueurs</i>	Le nombre de joueurs qui existent

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.22.2.4 hors_map_monstre()

```
bool hors_map_monstre (
    SDL_Rect * collision,
    t_map * map )
```

Auteur

Bruneau Antoine

Paramètres

<i>collision</i>	la collision de l'entité
<i>map</i>	la map

Renvoie

bool 1 si l'entité est en dehors de la map, 0 sinon

6.22.2.5 init_sousbuffer()

```
void init_sousbuffer (
    t_map * map,
    joueur_t * joueur )
```

Auteur

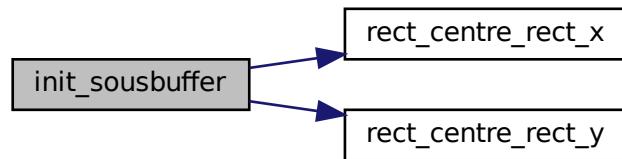
Ange Despert

On va initialiser 2 sous buffer : l'un pour le [plan principal](#) et l'un pour le [1er plan](#).

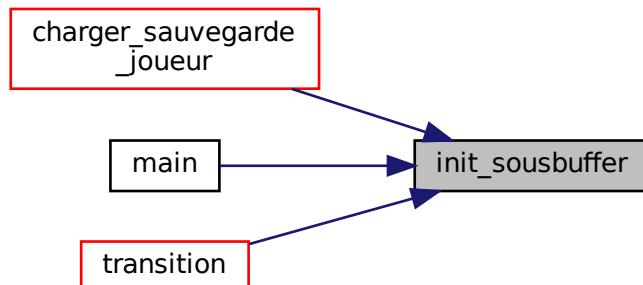
Paramètres

<i>map</i>	La mat qui contiendra l'un des sous buffer
<i>joueur</i>	Joueur autour duquel se fait la fenêtre d'affichage de la carte

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.22.2.6 taille_ecran_cases()

```
SDL_Rect taille_ecran_cases ( )
```

6.22.2.7 texture_map()

```
t_aff* texture_map (
    const t_map * map )
```

Auteur

Ange Despert

On va directement chercher cette information dans la structure [de la map](#).

Paramètres

<i>map</i>	La map dont on veut la texture
------------	--------------------------------

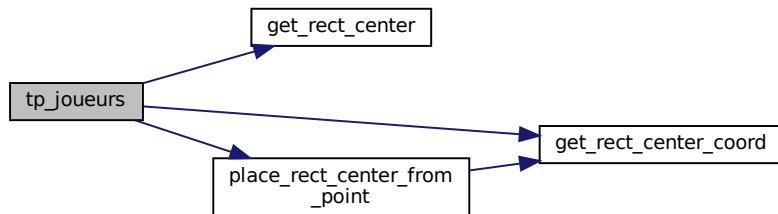
Renvoie

La texture de la map

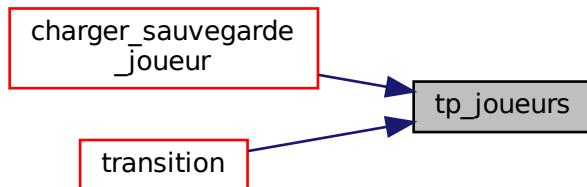
6.22.2.8 tp_joueurs()

```
void tp_joueurs (
    t_map * map,
    unsigned int x,
    unsigned int y,
    joueur_t ** joueurs,
    unsigned short int nb_joueurs )
```

Voici le graphe d'appel pour cette fonction :



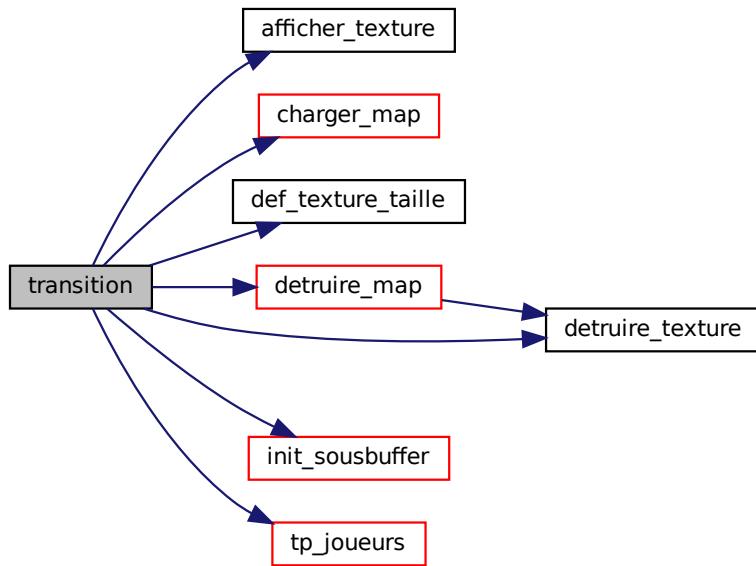
Voici le graphe des appels de cette fonction :



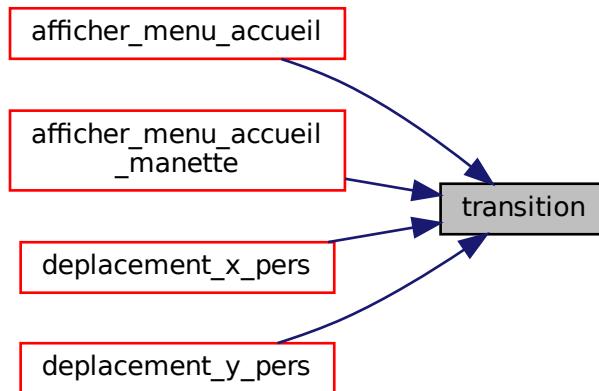
6.22.2.9 transition()

```
void transition (
    t_map ** actuelle,
    unsigned int num_map,
    joueur_t ** joueurs,
    unsigned short int nb_joueurs,
    unsigned int new_x,
    unsigned int new_y )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.22.3 Documentation des variables

6.22.3.1 map

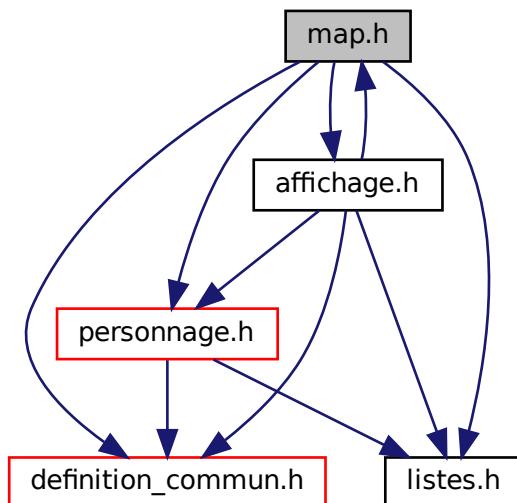
`t_map* map`

La map courante

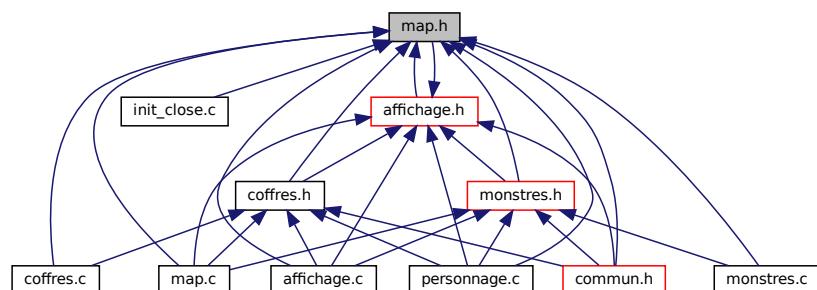
6.23 Référence du fichier map.h

Définitions des fonctions de gestion de la map.

Graphe des dépendances par inclusion de map.h:



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



Classes

- struct `zone_tp`
Structure représentant une zone de tp.
- struct `t_map`
Structure représentant une map.

Macros

- #define `TAILLE_CASE` 16

Fonctions

- `t_map * charger_map (const char *const nom_map)`
Fonction qui récupère les informations stockées dans le fichier don le nom est donné en entrée.
- void `init_sousbuffer (t_map *map, joueur_t *joueur)`
Fonction qui initialise les textures qui serviront de buffer d'affichage au moteur du jeu.
- `t_aff * texture_map (const t_map *map)`
Fonction qui renvoie la texture de la map.
- void `transition (t_map **actuelle, unsigned int num_map, joueur_t *joueurs[], unsigned short int nb_joueurs, unsigned int new_x, unsigned int new_y)`
Fonction qui permet de faire la transition avec une autre map.
- void `tp_joueurs (t_map *map, unsigned int x, unsigned int y, joueur_t *joueurs[], unsigned short int nb_joueurs)`
Fonction qui permet de téléporter des joueurs à un certaine coordonnée.
- void `détruire_map (t_map **map, joueur_t *joueurs[], unsigned short int nb_joueurs)`
Fonction qui permet de détruire une map.

Variables

- `t_map * map`

6.23.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)

Version

1.0

Date

05/04/2022

Copyright

Copyright (c) 2022

6.23.2 Documentation des macros

6.23.2.1 TAILLE_CASE

```
#define TAILLE_CASE 16
```

6.23.3 Documentation des fonctions

6.23.3.1 charger_map()

```
t_map* charger_map (
    const char *const nom_map )
```

Auteur

Ange Despert

Fonction qui fait appel à la librairie JSON-C pour charger les informations de la map au format JSON.

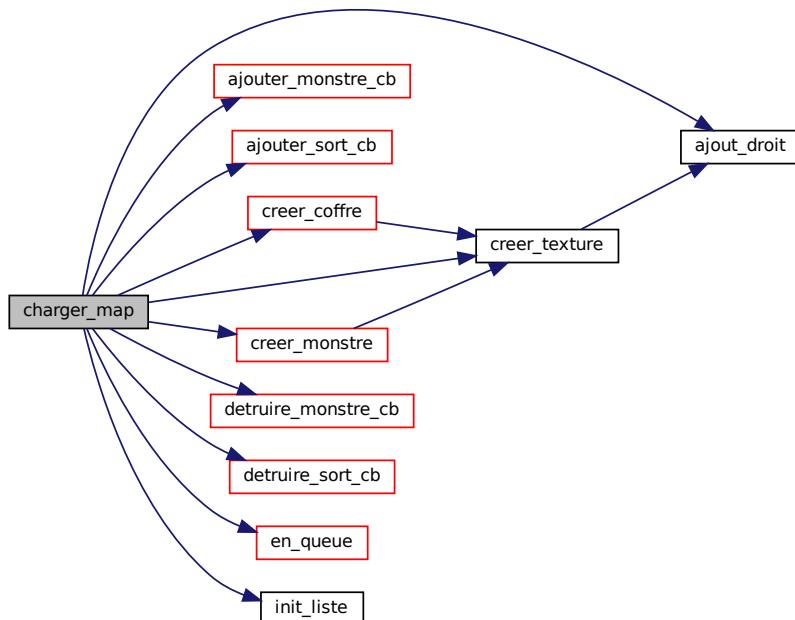
Paramètres

<i>nom_map</i>	Le nom du fichier qui contient les informations sur la map
----------------	--

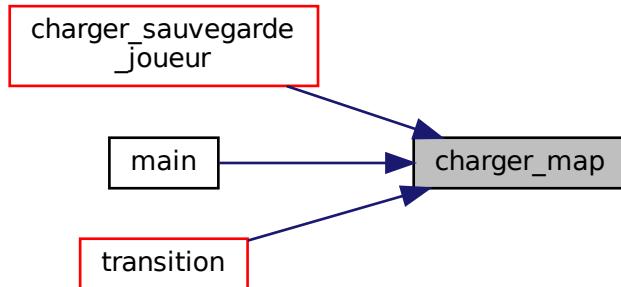
Renvoie

Une map initialisée avec toutes les informations dedans

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.23.3.2 detruire_map()

```
void detruire_map (
    t_map ** map,
    joueur_t * joueurs[],
    unsigned short int nb_joueurs )
```

Auteur

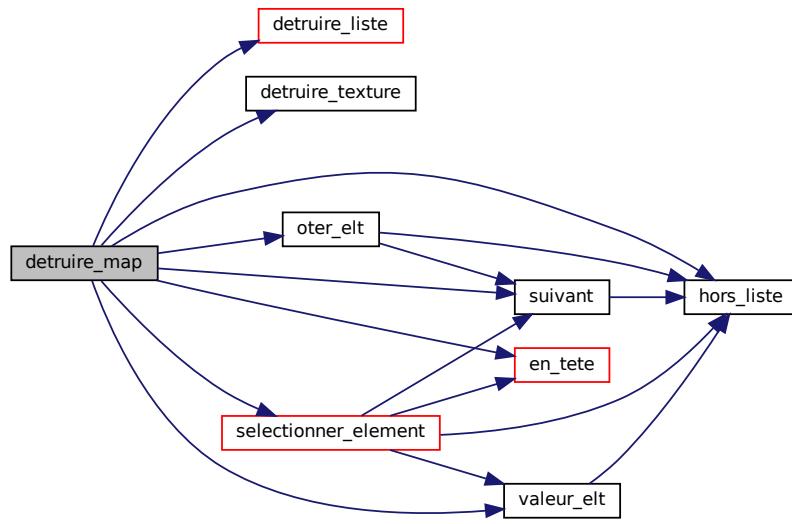
Ange Despert

Cette fonction libérera toute la mémoire allouée à la map. Elle détruira également les éléments des listes de la map. En ce qui concerne la liste de collision, étant donné que certains rectangle ne sont pas dynamiques, la fonction va d'abord s'assurer qu'ils peuvent être supprimer.

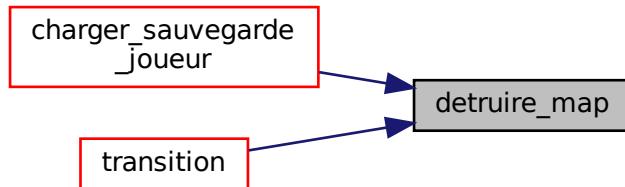
Paramètres

<i>map</i>	L'adresse de la map que l'on veut détruire
<i>joueurs</i>	Les joueurs qui existent
<i>nb_joueurs</i>	Le nombre de joueurs qui existent

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.23.3.3 init_sousbuffer()

```
void init_sousbuffer (
    t_map * map,
    joueur_t * joueur )
```

Auteur

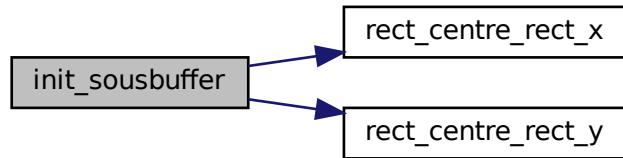
Ange Despert

On va initialiser 2 sous buffer : l'un pour le [plan principal](#) et l'un pour le [1er plan](#).

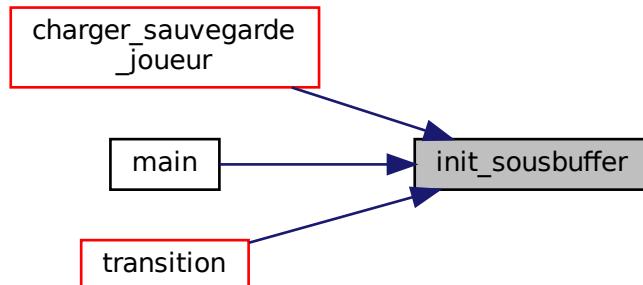
Paramètres

<i>map</i>	La mat qui contiendra l'un des sous buffer
<i>joueur</i>	Joueur autour duquel se fait la fenêtre d'affichage de la carte

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.23.3.4 `texture_map()`

```
t_aff* texture_map (
    const t_map * map )
```

Auteur

Ange Despert

On va directement chercher cette information dans la structure [de la map](#).

Paramètres

<i>map</i>	La map dont on veut la texture
------------	--------------------------------

Renvoie

La texture de la map

6.23.3.5 tp_joueurs()

```
void tp_joueurs (
    t_map * map,
    unsigned int x,
    unsigned int y,
    joueur_t * joueurs[],
    unsigned short int nb_joueurs )
```

Auteur

Ange Despert

Cette fonction va placer les joueurs aux coordonnées voulues. Elle centrera la caméra au niveau du personnage principal si cela est possible.

Paramètres

<i>map</i>	La map où l'on téléporte les joueurs
<i>x</i>	La nouvelle coordonnée x des joueurs
<i>y</i>	La nouvelle coordonnée y des joueurs
<i>joueurs</i>	Les joueurs qui existent
<i>nb_joueurs</i>	Le nombre de joueurs qui existent

6.23.3.6 transition()

```
void transition (
    t_map ** actuelle,
    unsigned int num_map,
    joueur_t * joueurs[],
    unsigned short int nb_joueurs,
    unsigned int new_x,
    unsigned int new_y )
```

Auteur

Ange Despert

Cette fonction va détruire la map actuelle, charger la nouvelle et déplacer les personnages à leur nouvel emplacement donné en paramètre.

Paramètres

<i>actuelle</i>	L'adresse de la map actuelle
<i>num_map</i>	L'identifiant de la map que l'on veut accéder
<i>joueurs</i>	Les joueurs qui existent
<i>nb_joueurs</i>	Le nombre de joueurs existants
<i>new_x</i>	La nouvelle coordonnée x des joueurs
<i>new_y</i>	La nouvelle coordonnée y des joueurs

6.23.4 Documentation des variables

6.23.4.1 map

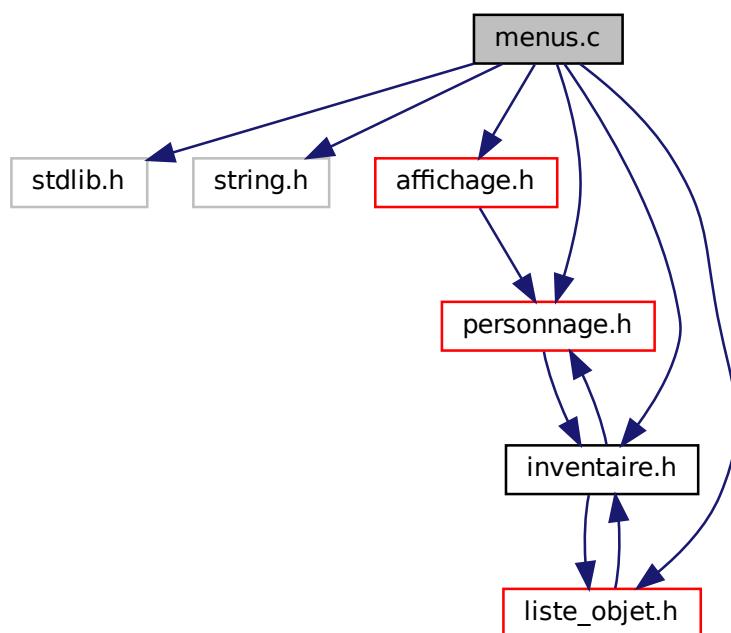
```
t_map* map
```

La map courante

6.24 Référence du fichier menus.c

Fonctions des menus du jeu.

Graphe des dépendances par inclusion de menus.c:



Fonctions

- void [afficher_menu_pause](#) (joueur_t *joueur[], char *f_src_objet, unsigned short int nb_joueurs)

Affiche le menu pause et gère les interactions avec le joueur.
- void [afficher_inventaire](#) (joueur_t *joueur, SDL_KeyCode touche_inventaire)

Affiche le menu inventaire et gère les interactions avec le joueur.
- void [afficher_menu_accueil](#) (int *nb_joueur)

Affiche le menu d'accueil et gère les interactions avec le joueur.
- void [init_text_menus](#) (int nb_joueur)

Créer l'inventaire du joueur 2.
- static int [draw_rect_epaisseur](#) (const SDL_Rect *a_dessiner, SDL_Renderer *rendu, unsigned int epaisseur)

Fonction qui permet d'afficher un rectangle à l'écran avec un épaisseur choisie.
- void [afficher_menu_pause_manette](#) (joueur_t *joueur)

Affiche le menu pause et gère les interactions avec le joueur.
- void [afficher_menu_accueil_manette](#) (int *nb_joueur)

Affiche le menu d'accueil et gère les interactions avec le joueur.
- void [afficher_inventaire_manette](#) (joueur_t *joueur)

Affiche l'inventaire et gère les interactions avec l'utilisateur.

Variables

- t_aff * text_pause = NULL
- t_aff * text_inventaire1 = NULL
- t_aff * text_inventaire2 = NULL
- t_aff * text_accueil = NULL

6.24.1 Description détaillée

Auteurs

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
 Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

1.0

Date

05/04/2022

Copyright

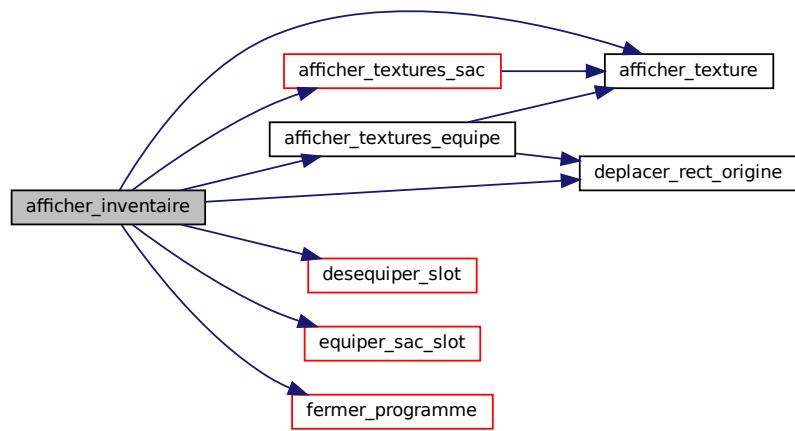
Copyright (c) 2022

6.24.2 Documentation des fonctions

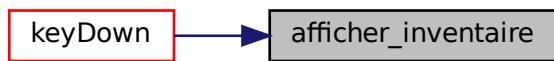
6.24.2.1 afficher_inventaire()

```
void afficher_inventaire (  
    joueur_t * joueur,  
    SDL_KeyCode touche_inventaire )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.24.2.2 afficher_inventaire_manette()

```
void afficher_inventaire_manette (
```

Auteur

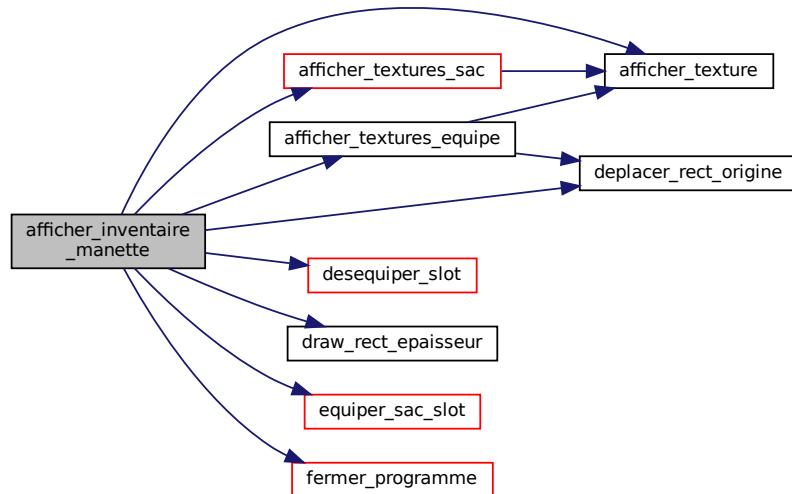
Ange Despert

Fonctionnement similaire à la fonction `afficher_inventaire` mais fonctionne à l'aide d'une manette.

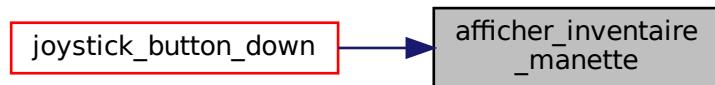
Paramètres

<i>joueur</i>	Joueur auquel appartient l'inventaire
---------------	---------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.24.2.3 afficher_menu_accueil()

```
void afficher_menu_accueil (
    int * nb_joueur )
```

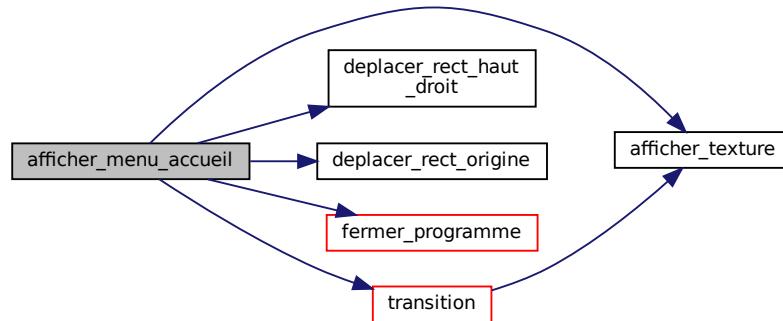
Auteur

Max Descomps

Paramètres

<i>nb_joueur</i>	Le nombre de joueurs souhaitant commencer une partie
------------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.24.2.4 afficher_menu_accueil_manette()

```
void afficher_menu_accueil_manette (
    int * nb_joueur )
```

Auteur

Ange Despert

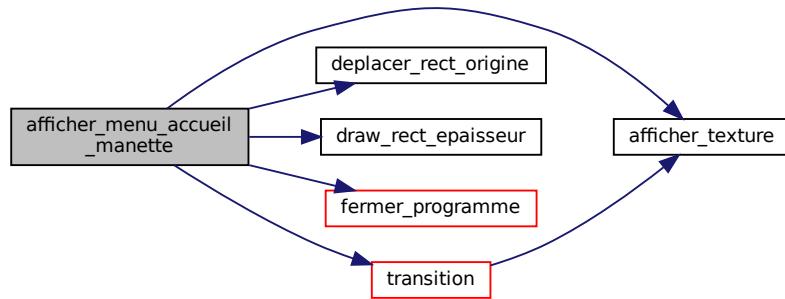
Fonction qui est similaire à la fonction [afficher_menu_accueil](#) mais permet l'utilisation de la manette.

A faire Ajout du multijoueur.

Paramètres

<code>nb_joueur</code>	Le nombre de joueurs souhaitant commencer une partie (renvoie toujours 1)
------------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

**6.24.2.5 afficher_menu_pause()**

```
void afficher_menu_pause (
    joueur_t * joueur[],
    char * f_src_objet,
    unsigned short int nb_joueurs )
```

Auteur

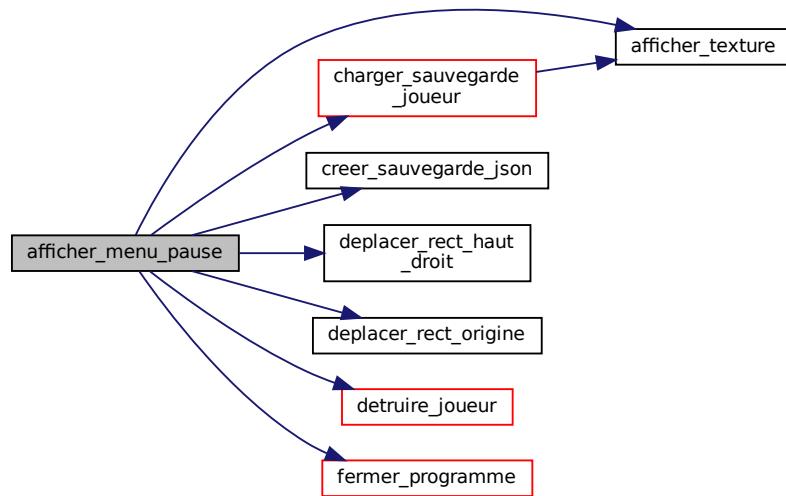
Ange Despert

Cette fonction affiche le menu pause et gère le clic des boutons à la souris.

Paramètres

<code>joueur</code>	Les joueurs qui existent
<code>f_src_objet</code>	Liste des objets du jeu utilisée dans certaines fonctionnalités du menu pause
<code>nb_joueurs</code>	Le nombre de joueurs qui existent

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.24.2.6 afficher_menu_pause_manette()

```
void afficher_menu_pause_manette (
    joueur_t * joueur )
```

Auteur

Ange Despert

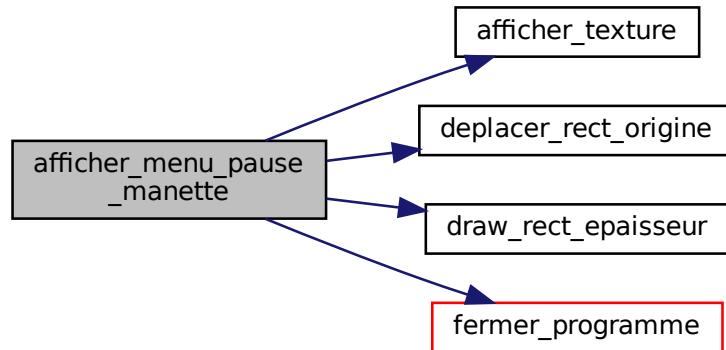
Cette fonction affiche le menu pause et fonctionne de façon similaire à la fonction [afficher_menu_pause](#) mais les sélections sont faites à la manette.

A faire Permettre la sauvegarde et le chargement de celle-ci avec la manette.

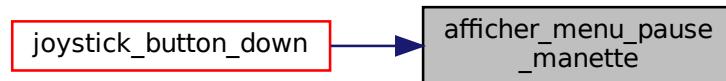
Paramètres

<i>joueur</i>	Le joueur principal
---------------	---------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



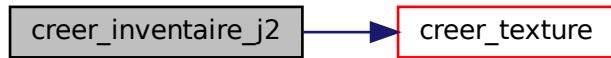
6.24.2.7 `creer_inventaire_j2()`

```
void creer_inventaire_j2( )
```

Auteur

Max Descomps

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.24.2.8 draw_rect_epaisseur()

```

static int draw_rect_epaisseur (
    const SDL_Rect * a_dessiner,
    SDL_Renderer * rendu,
    unsigned int epaisseur ) [static]
  
```

Auteur

Ange Despert

Cette fonction permet d'afficher des rectangles plus épais nécessaires à l'affichage de la sélection avec manette.

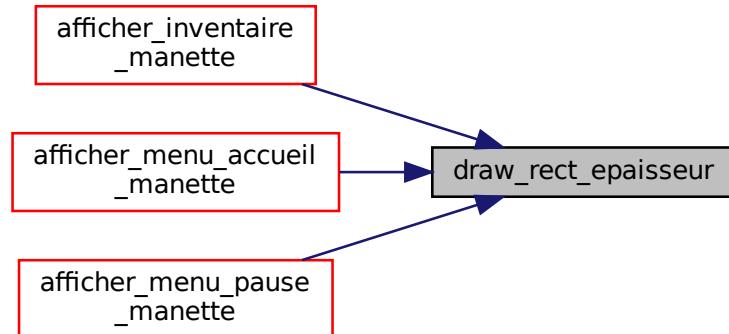
Paramètres

<i>a_dessiner</i>	Le rectangle que l'on veut dessiner à l'écran
<i>rendu</i>	Le rendu sur lequel on veut le dessiner
<i>epaisseur</i>	L'épaissir du trait

Renvoie

0 s'il n'y a pas eu d'erreur, appeler `SDL_GetError()` pour plus d'informations

Voici le graphe des appelants de cette fonction :



6.24.2.9 init_text_menus()

```
void init_text_menus (
    int nb_joueur )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.24.3 Documentation des variables

6.24.3.1 text_accueil

```
t_aff* text_accueil = NULL
```

La texture du menu d'accueil

6.24.3.2 text_inventaire1

```
t_aff* text_inventaire1 = NULL
```

La texture de l'inventaire du J1

6.24.3.3 text_inventaire2

```
t_aff* text_inventaire2 = NULL
```

La texture de l'inventaire du J2

6.24.3.4 text_pause

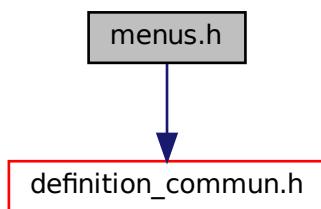
```
t_aff* text_pause = NULL
```

La texture du menu de pause

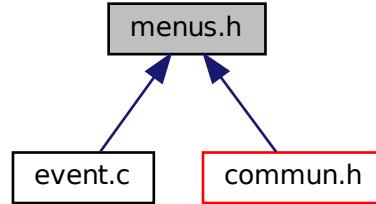
6.25 Référence du fichier menus.h

Définitions relatives à la gestion des menus du jeu.

Graphe des dépendances par inclusion de menus.h:



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



Fonctions

- void `afficher_menu_pause` (joueur_t *joueur[], char *f_src_objet, unsigned short int nb_joueurs)
Affiche le menu pause et gère les interactions avec le joueur.
- void `afficher_menu_pause_manette` (joueur_t *joueur)
Affiche le menu pause et gère les interactions avec le joueur.
- void `afficher_menu_accueil` (int *nb_joueur)
Affiche le menu d'accueil et gère les interactions avec le joueur.
- void `afficher_menu_accueil_manette` (int *nb_joueur)
Affiche le menu d'accueil et gère les interactions avec le joueur.
- void `afficher_inventaire` (joueur_t *joueur, SDL_KeyCode touche_inventaire)
- void `afficher_inventaire_manette` (joueur_t *joueur)
Affiche l'inventaire et gère les interactions avec l'utilisateur.
- void `init_text_menus` (void)
Créer les textures des menus.
- void `creer_inventaire_j2` (void)
Créer l'inventaire du joueur 2.

Variables

- t_aff * text_pause
- t_aff * text_inventaire1
- t_aff * text_inventaire2
- t_aff * text_accueil

6.25.1 Description détaillée

Auteurs

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
Max Descomps

Version

1.0

Date

05/04/2022

Copyright

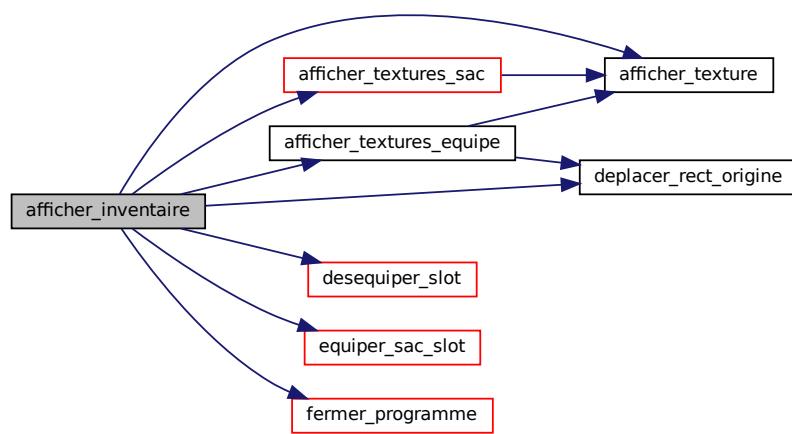
Copyright (c) 2022

6.25.2 Documentation des fonctions

6.25.2.1 afficher_inventaire()

```
void afficher_inventaire (
    joueur_t * joueur,
    SDL_KeyCode touche_inventaire )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.25.2.2 afficher_inventaire_manette()

```
void afficher_inventaire_manette (
    joueur_t * joueur )
```

Auteur

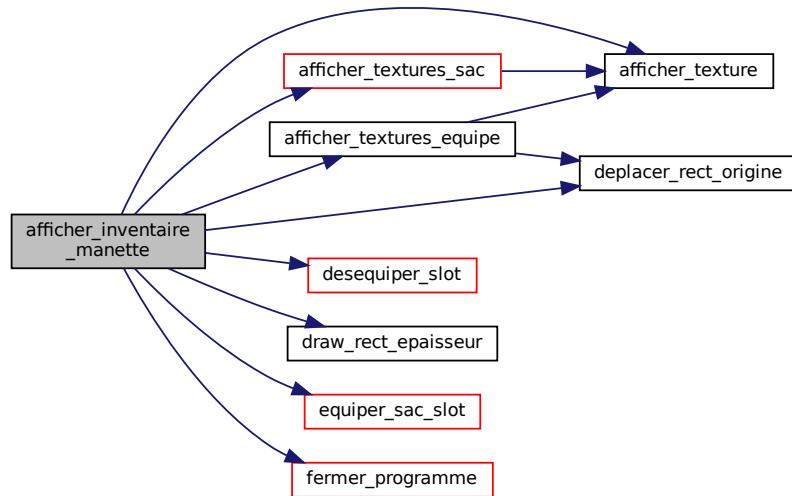
Ange Despert

Fonctionnement similaire à la fonction [afficher_inventaire](#) mais fonctionne à l'aide d'une manette.

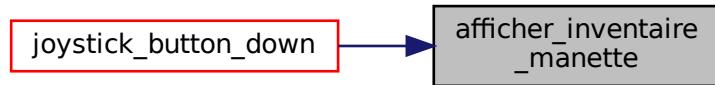
Paramètres

<i>joueur</i>	Joueur auquel appartient l'inventaire
---------------	---------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.25.2.3 afficher_menu_accueil()

```
void afficher_menu_accueil (
    int * nb_joueur )
```

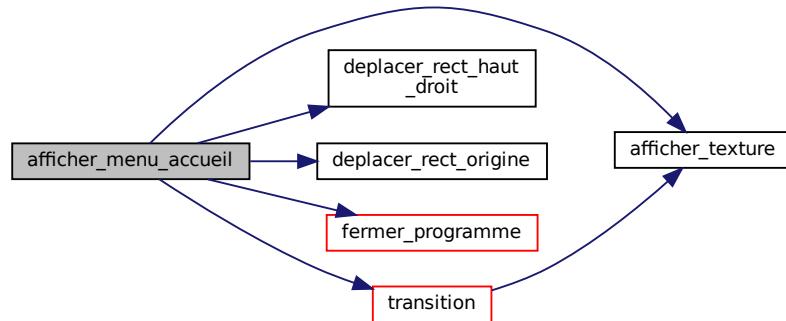
Auteur

Max Descomps

Paramètres

<i>nb_joueur</i>	Le nombre de joueurs souhaitant commencer une partie
------------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.25.2.4 afficher_menu_accueil_manette()

```
void afficher_menu_accueil_manette (
    int * nb_joueur )
```

Auteur

Ange Despert

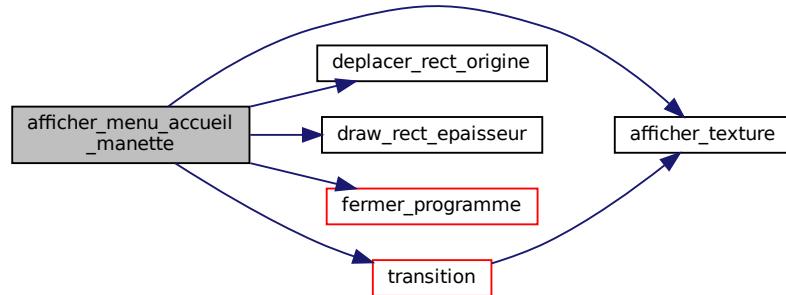
Fonction qui est similaire à la fonction [afficher_menu_accueil](#) mais permet l'utilisation de la manette.

A faire Ajout du multijoueur.

Paramètres

<code>nb_joueur</code>	Le nombre de joueurs souhaitant commencer une partie (renvoie toujours 1)
------------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

**6.25.2.5 afficher_menu_pause()**

```
void afficher_menu_pause (
    joueur_t * joueur[],
    char * f_src_objet,
    unsigned short int nb_joueurs )
```

Auteur

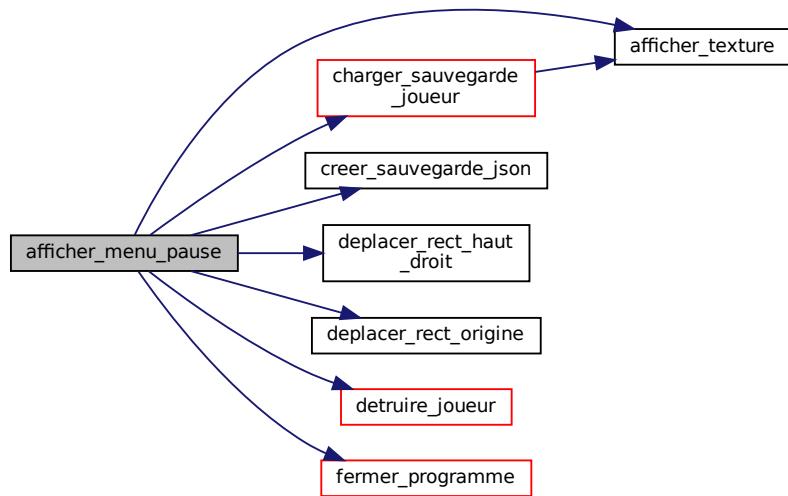
Ange Despert

Cette fonction affiche le menu pause et gère le clic des boutons à la souris.

Paramètres

<code>joueur</code>	Les joueurs qui existent
<code>f_src_objet</code>	Liste des objets du jeu utilisée dans certaines fonctionnalités du menu pause
<code>nb_joueurs</code>	Le nombre de joueurs qui existent

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.25.2.6 afficher_menu_pause_manette()

```
void afficher_menu_pause_manette (
    joueur_t * joueur )
```

Auteur

Ange Despert

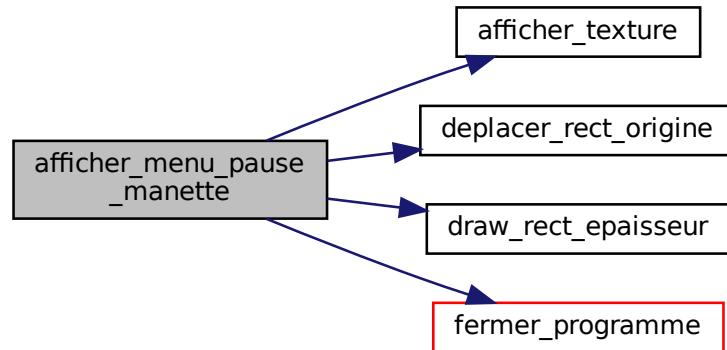
Cette fonction affiche le menu pause et fonctionne de façon similaire à la fonction [afficher_menu_pause](#) mais les sélections sont faites à la manette.

A faire Permettre la sauvegarde et le chargement de celle-ci avec la manette.

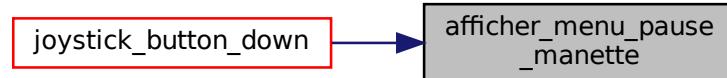
Paramètres

<i>joueur</i>	Le joueur principal
---------------	---------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



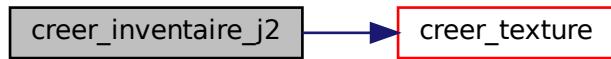
6.25.2.7 `creer_inventaire_j2()`

```
void creer_inventaire_j2( )
```

Auteur

Max Descomps

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.25.2.8 init_text_menus()

```
void init_text_menus (
    void )
```

Auteur

Max Descomps

6.25.3 Documentation des variables

6.25.3.1 text_accueil

t_aff* text_accueil

La texture du menu d'accueil

6.25.3.2 text_inventaire1

`t_aff* text_inventaire1`

La texture de l'inventaire du J1

6.25.3.3 text_inventaire2

`t_aff* text_inventaire2`

La texture de l'inventaire du J2

6.25.3.4 text_pause

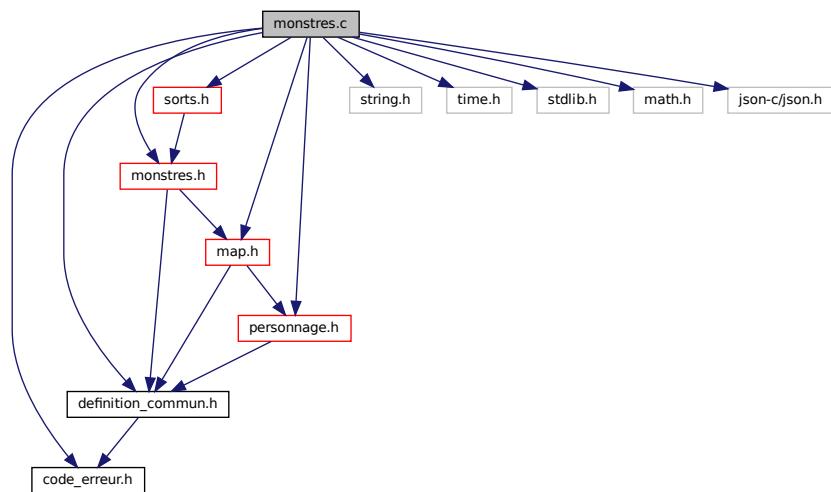
`t_aff* text_pause`

La texture du menu de pause

6.26 Référence du fichier monstres.c

Fichier contenant toutes les fonctions concernant les monstres.

Graphe des dépendances par inclusion de monstres.c:



Fonctions

- void **détruire_monstre** (*monstre_t* ***monstre*)
- void **détruire_monstre_cb** (void **monstre*)
 - Fonction de "call back" qui détruit une structure monstre.*
- *monstre_t* * **ajouter_monstre** (*monstre_t* **monstre*)
- void * **ajouter_monstre_cb** (void **monstre*)
 - Fonction de "call back" qui retourne une structure monstre.*
- void **détruire_liste_base_monstres** (*liste_base_monstres_t* ***liste_base_monstres*)
- *monstre_t* * **creer_monstre** (*liste_base_monstres_t* **liste_base_monstres*, const char *const nom_monstre, int x, int y, *t_map* **map*)
- *type_monstre_t* nom_monstre_to_type_monstre (const char *const nom_monstre)
 - Convertit une chaîne de caractères en type de monstre.*
- void **marcher_monstre** (*monstre_t* **monstre*)
 - Fonction qui met à jour la texture du monstre pour ses déplacements.*
- void **orienter_monstre** (*monstre_t* **monstre*)
 - Fonction qui met à jour la texture du monstre par rapport à son orientation.*
- void **orienter_monstre_vers_joueur** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour la texture du monstre pour qu'il soit orienté vers le joueur.*
- void **monstre_en_garde** (*monstre_t* **monstre*)
 - Fonction qui met à jour la texture du monstre sur la position de garde (immobile)*
- void **monstre_attaque** (*monstre_t* **monstre*)
 - Fonction qui met à jour la texture du monstre sur la position d'attaque.*
- void **fuir_joueur** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour la texture du monstre de sorte à ce qu'il fuit le joueur.*
- void **rush_joueur** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour la texture du monstre de sorte à ce qu'il se déplace vers le joueur.*
- void **agro_witcher** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour la texture du monstre "witcher" lorsqu'il est dans sa phase d'attaque.*
- void **agro_knight** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour la texture du monstre "knight" lorsqu'il est dans sa phase d'attaque (reboucle la même action)*
- void **agro_monstre** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour la texture d'un monstre lorsqu'il est dans sa phase d'attaque.*
- void **ronde_monstre** (*monstre_t* **monstre*)
 - Fonction qui met à jour la texture d'un monstre lorsqu'il est dans sa phase passive.*
- void **action_monstre** (*monstre_t* **monstre*, *joueur_t* **joueur*)
 - Fonction qui met à jour le sprite d'un monstre en fonction d'un joueur.*
- void **charger_base_monstre** (char *chemin_fichier, *liste_base_monstres_t* ***liste_base_monstres*)
 - Fonction qui recopie les informations d'un fichier json pour les insérer dans la structure liste_base_monstres.*

Variables

- *liste_base_monstres_t* * *liste_base_monstres* = NULL

6.26.1 Description détaillée

Auteur

Antoine Bruneau (Antoine.Bruneau.Etu@univ-lemans.fr)

Version

0.1

Date

03/03/2022

Copyright

Copyright (c) 2022

6.26.2 Documentation des fonctions

6.26.2.1 action_monstre()

```
void action_monstre (
    monstre_t * monstre,
    joueur_t * joueur )
```

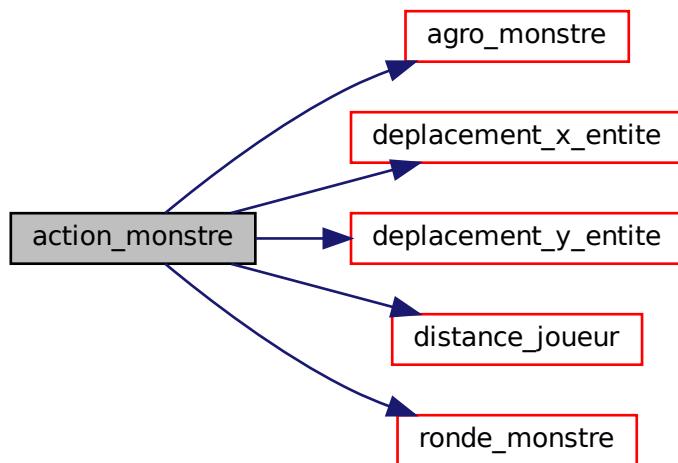
Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à mettre à jour
<i>joueur</i>	le joueur qui provoque l'action du monstre

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.2 agro_knight()

```
void agro_knight (
    monstre_t * monstre,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à mettre à jour
<i>joueur</i>	le joueur à attaquer

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.3 agro_monstre()

```
void agro_monstre (
    monstre_t * monstre,
    joueur_t * joueur )
```

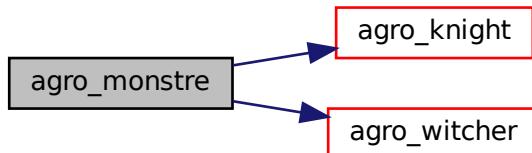
Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à mettre à jour
<i>joueur</i>	le joueur à attaquer

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.4 agro_witcher()

```
void agro_witcher (
    monstre_t * monstre,
    joueur_t * joueur )
```

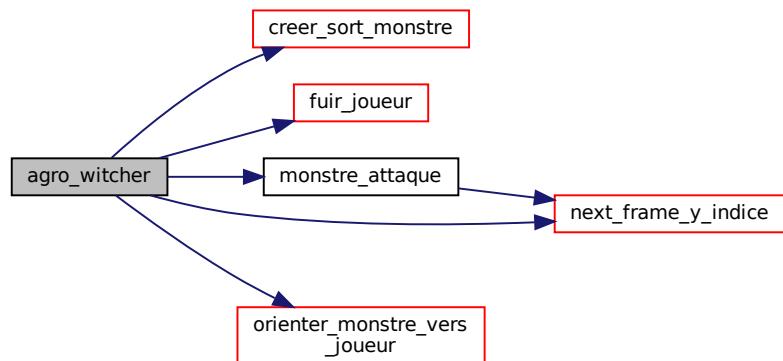
Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à mettre à jour
<i>joueur</i>	le joueur à attaquer

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.5 ajouter_monstre()

```
monstre_t* ajouter_monstre (
    monstre_t * monstre )
```

Voici le graphe des appels de cette fonction :



6.26.2.6 ajouter_monstre_cb()

```
void * ajouter_monstre_cb (
    void * monstre )
```

Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à retourner
----------------	------------------------

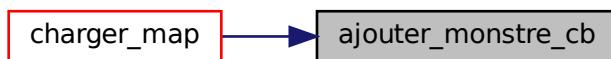
Renvoie

void* un pointeur générique sur une structure monstre

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.26.2.7 charger_base_monstre()

```
void charger_base_monstre (
    char * chemin_fichier,
    liste_base_monstres_t ** liste_base_monstres )
```

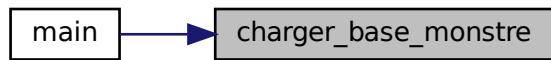
Auteur

Bruneau Antoine

Paramètres

<i>chemin_fichier</i>	nom du fichier à lire
<i>liste_base_monstres</i>	tableau dans lequel enregistrer les monstres

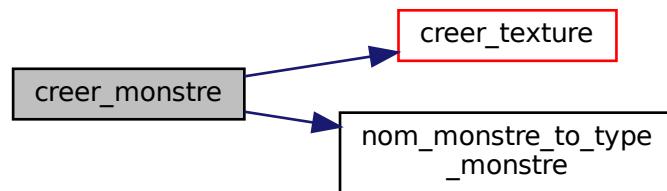
Voici le graphe des appels de cette fonction :



6.26.2.8 creer_monstre()

```
monstre_t* creer_monstre (
    liste_base_monstres_t * liste_base_monstres,
    const char *const nom_monstre,
    int x,
    int y,
    t_map * map )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.9 detruire_liste_base_monstres()

```
void detruire_liste_base_monstres (
    liste_base_monstres_t ** liste_base_monstres )
```

6.26.2.10 detruire_monstre()

```
void detruire_monstre (
    monstre_t ** monstre )
```

Voici le graphe des appelants de cette fonction :



6.26.2.11 detruire_monstre_cb()

```
void detruire_monstre_cb (
    void * monstre )
```

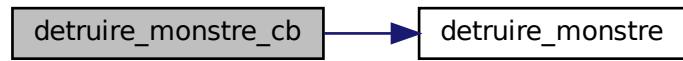
Auteur

Bruneau Antoine

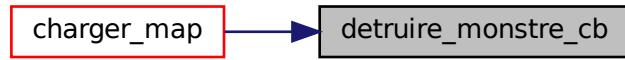
Paramètres

<i>monstre</i>	le monstre à détruire
----------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

**6.26.2.12 fuir_joueur()**

```
void fuir_joueur (
    monstre_t * monstre,
    joueur_t * joueur )
```

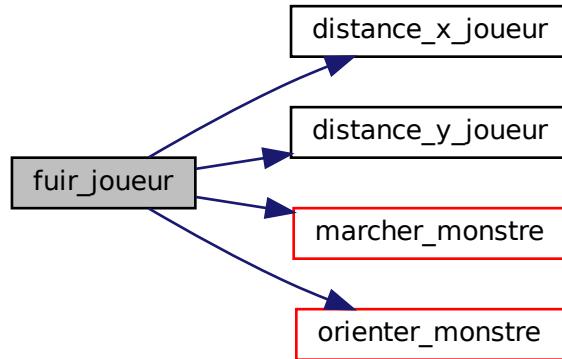
Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à mettre à jour
<i>joueur</i>	le joueur à fuire

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.26.2.13 marcher_monstre()

```
void marcher_monstre (
    monstre_t * monstre )
```

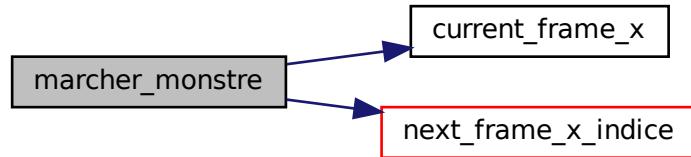
Auteur

Bruneau Antoine

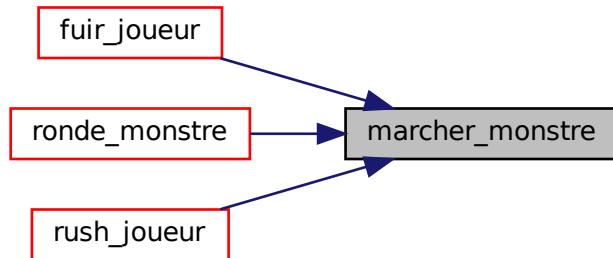
Paramètres

<code>monstre</code>	le monstre à mettre à jour
----------------------	----------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.26.2.14 monstre_attaque()

```
void monstre_attaque (
    monstre_t * monstre )
```

Auteur

Bruneau Antoine

Paramètres

<code>monstre</code>	le monstre à mettre à jour
----------------------	----------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.15 monstre_en_garde()

```
void monstre_en_garde (
    monstre_t * monstre )
```

Auteur

Bruneau Antoine

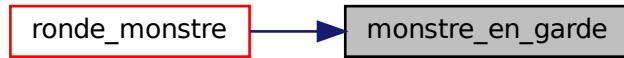
Paramètres

<i>monstre</i>	le monstre à mettre à jour
----------------	----------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.16 nom_monstre_to_type_monstre()

```
type_monstre_t nom_monstre_to_type_monstre (
    const char *const nom_monstre )
```

Auteur

Bruneau Antoine

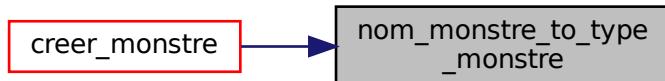
Paramètres

<i>nom_monstre</i>	La chaîne de caractères à convertir
--------------------	-------------------------------------

Renvoie

type_monstre_t le type de monstre

Voici le graphe des appelants de cette fonction :



6.26.2.17 orienter_monstre()

```
void orienter_monstre (
    monstre_t * monstre )
```

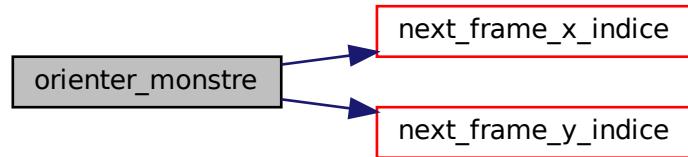
Auteur

Bruneau Antoine

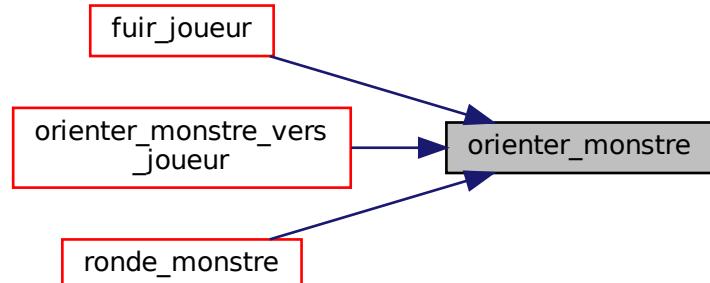
Paramètres

<i>monstre</i>	le monstre à mettre à jour
----------------	----------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.18 orienter_monstre_vers_joueur()

```
void orienter_monstre_vers_joueur (
    monstre_t * monstre,
    joueur_t * joueur )
```

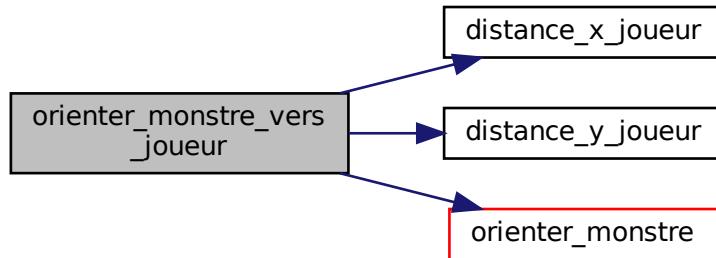
Auteur

Bruneau Antoine

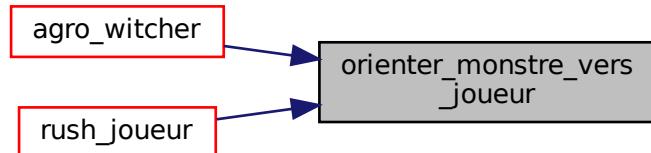
Paramètres

<i>monstre</i>	le monstre à mettre à jour
<i>joueur</i>	le joueur ciblé

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

**6.26.2.19 ronde_monstre()**

```
void ronde_monstre (
    monstre_t * monstre )
```

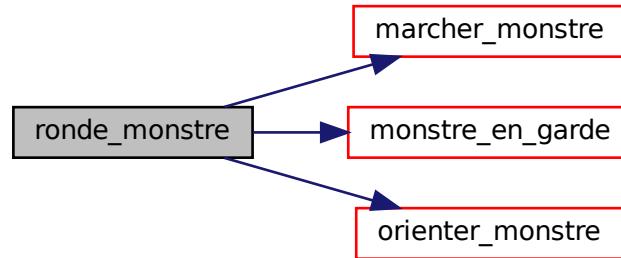
Auteur

Bruneau Antoine

Paramètres

<i>monstre</i>	le monstre à mettre à jour
----------------	----------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.26.2.20 `rush_joueur()`

```
void rush_joueur (
    monstre_t * monstre,
    joueur_t * joueur )
```

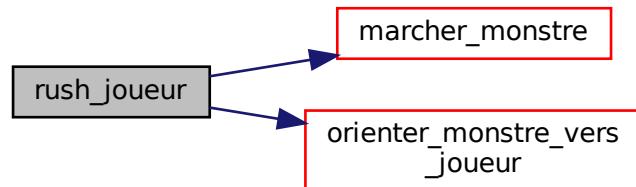
Auteur

Bruneau Antoine

Paramètres

<code>monstre</code>	le monstre à mettre à jour
<code>joueur</code>	le joueur à atteindre

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.26.3 Documentation des variables

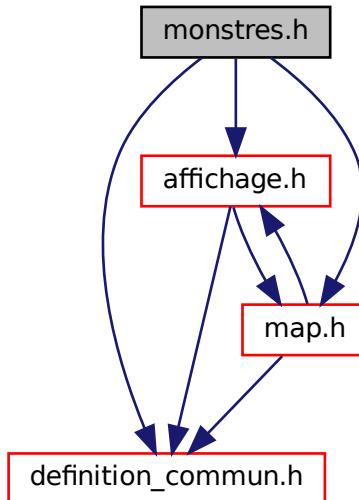
6.26.3.1 liste_base_monstres

```
liste_base_monstres_t* liste_base_monstres = NULL
```

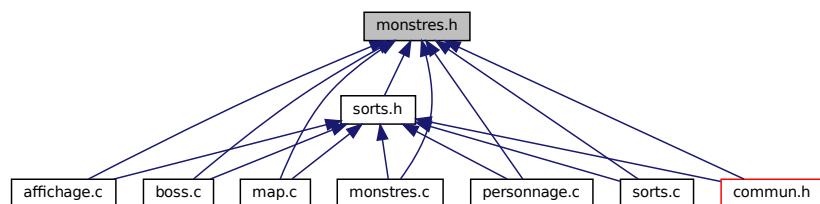
6.27 Référence du fichier monstres.h

Fonctions concernant les monstres.

Graphe des dépendances par inclusion de monstres.h:



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



Classes

- struct [monstre_s](#)
- struct [base_monstre_s](#)
- struct [liste_base_monstres_t](#)

Macros

- #define [DISTANCE_AGRO](#) 130 /* distance à partir de laquelle le joueur n'est pas détecté par les monstres*/
- #define [DUREE_MONSTRE_MARCHER](#) 60
- #define [DUREE_MONSTRE_EN_GARDE](#) 80
- #define [DUREE_RUSH_OU_FUITE](#) 40
- #define [DUREE_MONSTRE_ATTAQUE](#) 50
- #define [DUREE_MONSTRE_BLESSE](#) 20
- #define [DUREE_MONSTRE_PAUSE](#) 15

Énumérations

- enum `type_monstre_t` { `WITCHER`, `KNIGHT`, `TYPE_MONSTRE_INCONNU` }
L'énumération des types de monstre.
- enum `action_monstre_t` {
 `MONSTRE_MARCHER`, `MONSTRE_EN_GARDE`, `MONSTRE_ATTAQUE`, `RUSH_OU_FUITE`,
 `MONSTRE_BLESSE`, `MONSTRE_PAUSE` }
L'énumération des actions du monstre.

Fonctions

- void * `ajouter_monstre_cb` (void *monstre)
Fonction de "call back" qui retourne une structure monstre.
- void `détruire_monstre_cb` (void *monstre)
Fonction de "call back" qui détruit une structure monstre.
- `monstre_t *` `creer_monstre` (`liste_base_monstres_t *liste_base_monstres`, const char *const nom_monstre,
 int x, int y, `t_map *map`)
- void `détruire_liste_base_monstres` (`liste_base_monstres_t **liste_base_monstres`)
- void `charger_base_monstre` (char *chemin_fichier, `liste_base_monstres_t **liste_base_monstres`)
Fonction qui recopie les informations d'un fichier json pour les insérer dans la structure liste_base_monstres.
- void `action_monstre` (`monstre_t *monstre`, joueur_t *joueur)
Fonction qui met à jour le sprite d'un monstre en fonction d'un joueur.
- `type_monstre_t nom_monstre_to_type_monstre` (const char *const nom_monstre)
Convertit une chaîne de caractères en type de monstre.

Variables

- `liste_base_monstres_t * liste_base_monstres`

6.27.1 Description détaillée

Auteur

Antoine Bruneau (Antoine.Bruneau.Etu@univ-lemans.fr)

Version

0.1

Date

01/02/2022

Copyright

Copyright (c) 2022

6.27.2 Documentation des macros

6.27.2.1 DISTANCE_AGRO

```
#define DISTANCE_AGRO 130 /* distance à partir de laquelle le joueur n'est pas détecté par les monstres*/
```

6.27.2.2 DUREE_MONSTRE_ATTAQUE

```
#define DUREE_MONSTRE_ATTAQUE 50
```

6.27.2.3 DUREE_MONSTRE_BLESSE

```
#define DUREE_MONSTRE_BLESSE 20
```

6.27.2.4 DUREE_MONSTRE_EN_GARDE

```
#define DUREE_MONSTRE_EN_GARDE 80
```

6.27.2.5 DUREE_MONSTRE_MARCHER

```
#define DUREE_MONSTRE_MARCHER 60
```

6.27.2.6 DUREE_MONSTRE_PAUSE

```
#define DUREE_MONSTRE_PAUSE 15
```

6.27.2.7 DUREE_RUSH_OU_FUITE

```
#define DUREE_RUSH_OU_FUITE 40
```

6.27.3 Documentation du type de l'énumération

6.27.3.1 action_monstre_t

```
enum action_monstre_t
```

Cela permet de savoir dans quelle état est le monstre

Valeurs énumérées

MONSTRE_MARCHER	le monstre marche normalement
MONSTRE_EN_GARDE	le monstre est immobile
MONSTRE_ATTAQUE	le monstre attaque
RUSH_OU_FUITE	le monstre va vers le joueur ou le fuit
MONSTRE_BLESSE	le monstre prend des dégâts
MONSTRE_PAUSE	le monstre se stop (après avoir blessé un joueur)

6.27.3.2 type_monstre_t

```
enum type_monstre_t
```

Cela permet de différencier les monstres pour réaliser des actions particulières en fonction de celui-ci.

Valeurs énumérées

WITCHER	
KNIGHT	
TYPE_MONSTRE_INCONNU	

6.27.4 Documentation des fonctions

6.27.4.1 action_monstre()

```
void action_monstre (
    monstre_t * monstre,
    joueur_t * joueur )
```

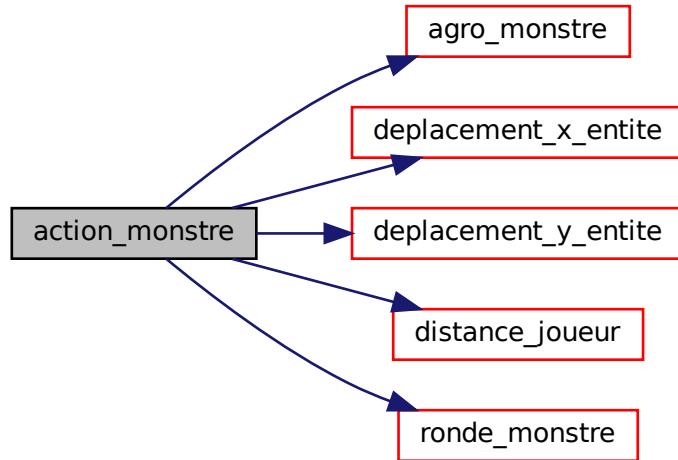
Auteur

Bruneau Antoine

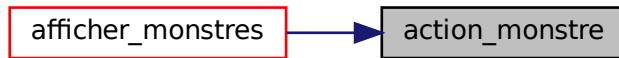
Paramètres

monstre	le monstre à mettre à jour
joueur	le joueur qui provoque l'action du monstre

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.27.4.2 ajouter_monstre_cb()

```
void* ajouter_monstre_cb (
    void * monstre )
```

Auteur

Bruneau Antoine

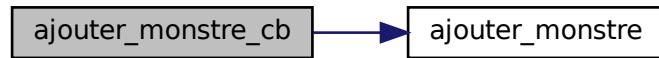
Paramètres

<code>monstre</code>	le monstre à retourner
----------------------	------------------------

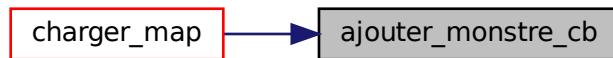
Renvoie

`void* un pointeur générique sur une structure monstre`

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.27.4.3 `charger_base_monstre()`

```
void charger_base_monstre (
    char * chemin_fichier,
    liste_base_monstres_t ** liste_base_monstres )
```

Auteur

Bruneau Antoine

Paramètres

<code>chemin_fichier</code>	nom du fichier à lire
<code>liste_base_monstres</code>	tableau dans lequel enregistrer les monstres

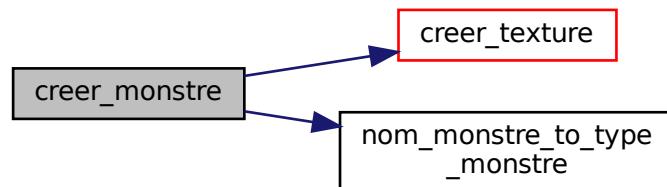
Voici le graphe des appelants de cette fonction :



6.27.4.4 creer_monstre()

```
monstre_t* creer_monstre (
    liste_base_monstres_t * liste_base_monstres,
    const char *const nom_monstre,
    int x,
    int y,
    t_map * map )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.27.4.5 detruire_liste_base_monstres()

```
void detruire_liste_base_monstres (
    liste_base_monstres_t ** liste_base_monstres )
```

6.27.4.6 detruire_monstre_cb()

```
void detruire_monstre_cb (
    void * monstre )
```

Auteur

Bruneau Antoine

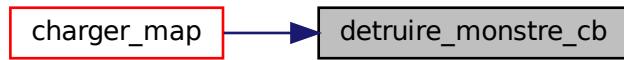
Paramètres

<i>monstre</i>	le monstre à détruire
----------------	-----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.27.4.7 nom_monstre_to_type_monstre()

```
type_monstre_t nom_monstre_to_type_monstre (
    const char *const nom_monstre )
```

Auteur

Bruneau Antoine

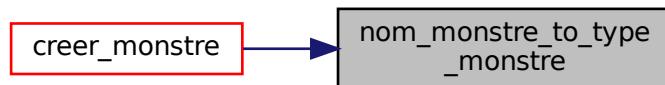
Paramètres

<i>nom_monstre</i>	La chaîne de caractères à convertir
--------------------	-------------------------------------

Renvoie

`type_monstre_t` le type de monstre

Voici le graphe des appels de cette fonction :



6.27.5 Documentation des variables

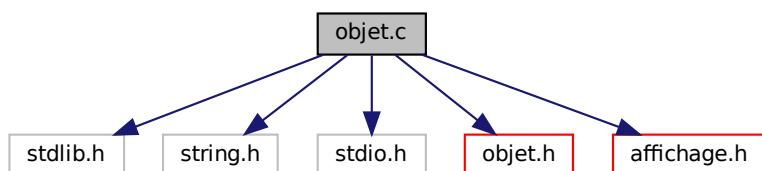
6.27.5.1 liste_base_monstres

`liste_base_monstres_t* liste_base_monstres`

6.28 Référence du fichier objet.c

Fichier contenant toutes les fonctions concernant les objets.

Graphe des dépendances par inclusion de `objet.c`:



Fonctions

- `objet_t * creer_objet` (const int id, const char *const texture_src, const `t_item` type, const char *nom, const short int niveau, const int att, const int def, const int vit)
Créé un objet du jeu.
- `void afficher_objet (objet_t *obj)`
Affiche les caractéristiques d'un objet dans la console.
- `void detruire_objet (objet_t **obj)`
Libère la mémoire allouée à un objet.

6.28.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

24/02/2022

Copyright

Copyright (c) 2022

6.28.2 Documentation des fonctions

6.28.2.1 afficher_objet()

```
void afficher_objet (
    objet_t * obj )
```

Auteur

Max Descomps

Paramètres

<code>obj</code>	L'objet à afficher
------------------	--------------------

Voici le graphe des appelants de cette fonction :



6.28.2.2 creer_objet()

```
objet_t* creer_objet (
    const int id,
    const char *const texture_src,
    const t_item type,
    const char * nom,
    const short int niveau,
    const int att,
    const int def,
    const int vit )
```

Auteur

Max Descomps

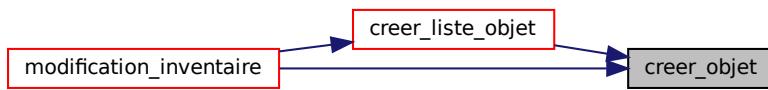
Paramètres

<i>id</i>	L'identificateur de l'objet
<i>texture_src</i>	Chemin vers l'image de l'objet
<i>type</i>	Type d'objet
<i>nom</i>	Nom de l'objet
<i>niveau</i>	Niveau nécessaire pour s'équiper de l'objet
<i>att</i>	Bonus d'attaque de l'objet
<i>def</i>	Bonus de défense de l'objet
<i>vit</i>	Bonus de vitesse de l'objet

Renvoie

Instance nouvellement allouée du type `objet_t` ou NULL

Voici le graphe des appelants de cette fonction :



6.28.2.3 `detruire_objet()`

```
void detruire_objet (
    objet_t ** obj )
```

Auteur

Max Descomps

Paramètres

<code>obj</code>	L'objet à libérer
------------------	-------------------

Voici le graphe d'appel pour cette fonction :



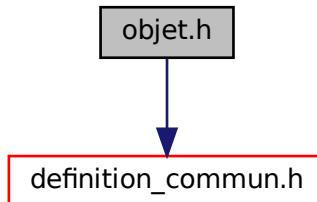
Voici le graphe des appelants de cette fonction :



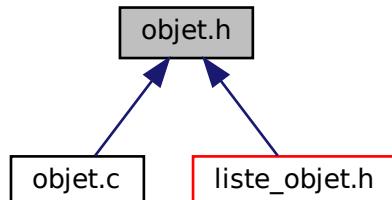
6.29 Référence du fichier objet.h

Définitions concernant les objets.

Graphe des dépendances par inclusion de objet.h:



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



Classes

- struct `objet_t`
Définition de la structure objet.

Macros

- #define `NB_TYPE_OBJ` 6

Énumérations

- enum `t_item` {
 `quete, arme, bouclier, protection,`
`amulette, consommable` }

Fonctions

- `objet_t * creer_objet` (const int id, const char *const texture_src, const `t_item` type, const char *nom, const short int niveau, const int att, const int def, const int vit)
Créé un objet du jeu.
- `void detruire_objet (objet_t **obj)`
Libère la mémoire allouée à un objet.
- `void afficher_objet (objet_t *obj)`
Affiche les caractéristiques d'un objet dans la console.

6.29.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.3

Date

05/04/2022

Copyright

Copyright (c) 2022

6.29.2 Documentation des macros

6.29.2.1 NB_TYPE_OBJ

```
#define NB_TYPE_OBJ 6
```

Nombre de types d'objets dans le jeu

6.29.3 Documentation du type de l'énumération

6.29.3.1 t_item

```
enum t_item
```

Types d'objets du jeu

Valeurs énumérées

quete	objet de quête
arme	objet de la catégorie arme
bouclier	objet de la catégorie bouclier
protection	objet de la catégorie protection
amulette	objet de la catégorie amulette
consommable	objet consommable

6.29.4 Documentation des fonctions**6.29.4.1 afficher_objet()**

```
void afficher_objet (
    objet_t * obj )
```

Auteur

Max Descomps

Paramètres

<i>obj</i>	L'objet à afficher
------------	--------------------

Voici le graphe des appelants de cette fonction :

**6.29.4.2 creer_objet()**

```
objet_t* creer_objet (
    const int id,
    const char *const texture_src,
    const t_item type,
    const char * nom,
```

```
const short int niveau,
const int att,
const int def,
const int vit )
```

Auteur

Max Descomps

Paramètres

<i>id</i>	L'identificateur de l'objet
<i>texture_src</i>	Chemin vers l'image de l'objet
<i>type</i>	Type d'objet
<i>nom</i>	Nom de l'objet
<i>niveau</i>	Niveau nécessaire pour s'équiper de l'objet
<i>att</i>	Bonus d'attaque de l'objet
<i>def</i>	Bonus de défense de l'objet
<i>vit</i>	Bonus de vitesse de l'objet

Renvoie

Instance nouvellement allouée du type [objet_t](#) ou NULL

Voici le graphe des appelants de cette fonction :

**6.29.4.3 detruire_objet()**

```
void detruire_objet (
    objet\_t ** obj )
```

Auteur

Max Descomps

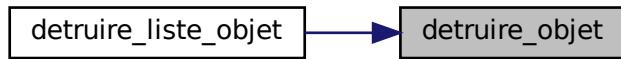
Paramètres

<i>obj</i>	L'objet à libérer
------------	-------------------

Voici le graphe d'appel pour cette fonction :



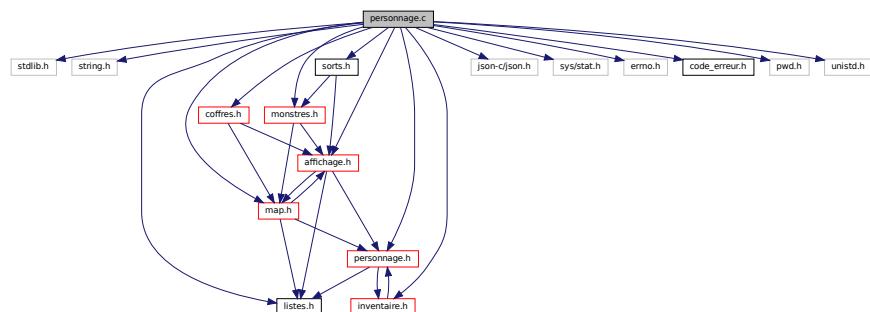
Voici le graphe des appels de cette fonction :



6.30 Référence du fichier personnage.c

Fichier contenant toutes les fonctions concernant le personnage.

Graphe des dépendances par inclusion de personnage.c:



Fonctions

- static void `copy` (const `byte` *origin, `byte` *out, `size_t` size)
- void `check_reperoire_jeux` ()

Fonction gère le répertoire de jeux.
- void `creer_sauvegarde_json` (`joueur_t` *j)

Fonction qui créer les sauvegardes du jeu.
- `bool sauv_existe` (`char` *nom_sauv)
- `joueur_t` * `charger_sauvegarde_joueur` (`char` *nom_sauv, `char` *f_src_obj, `joueur_t` *joueurs[], `unsigned short int` nb_joueurs)

- *Fonction qui charge une sauvegarde au format JSON.*
— joueur_t * **new_joueur** (const char *nom, int num_j, char *f_src_obj)
Fonction de création d'un joueur correspondant au modèle standard du jeu.
- joueur_t * **creer_joueur** (const char *nom, const int niveau, const int xp, const int maxPdv, const int pdv, const int attaque, const int defense, const int vitesse, const byte trig[TAILLE_TRIGGER], const t_direction_1 orient, const bool bouclier_equipe, const int num_j, char *f_src_obj)
— void **détruire_joueur** (joueur_t *)
Fonction qui détruit un joueur.
- void **maj_statistiques** (joueur_t *perso)
Fonction qui met à jour les statistiques d'un joueur lors d'un passage de niveau.
- void **afficher_statistiques** (joueur_t *perso)
Fonction qui affiche les statistiques d'un joueur dans la console.
- void **levelup** (joueur_t *perso)
Fonction qui gère le passage de niveau d'un joueur.
- void **gain_xp** (joueur_t *perso)
Fonction qui gère les effets d'un gain d'expérience.
- SDL_Rect * **zone_en_dehors_hitbox** (SDL_Rect *hitbox, SDL_Rect *sprite, t_direction_2 orient)
— SDL_bool **entite_suit_attaque** (SDL_Rect *monstre_hitbox, joueur_t *joueur)
— SDL_bool **entite_en_collision** (SDL_Rect *entite_1, SDL_Rect *entite_2, t_direction_1 *cote_entite_1, t_direction_1 *cote_entite_2)
— void **environnement_joueurs** (list *liste_monstres, list *liste_sorts, list *liste_coffres, joueur_t **joueurs, int nb_joueur)
Fonction qui gère les effets de l'environnement sur le joueur.
- void **stoper_mouvement_joueurs** (joueur_t **joueurs)
Stop le mouvement des joueurs en jeu.
- int **distance_x_joueur** (SDL_Rect collision, joueur_t *joueur)
Renvoie la distance séparant le joueur d'une entité définie par sa collision sur l'axe des abscisses.
- int **distance_y_joueur** (SDL_Rect collision, joueur_t *joueur)
Renvoie la distance séparant le joueur d'une entité définie par sa collision sur l'axe des ordonnées.
- int **distance_joueur** (SDL_Rect collision, joueur_t *joueur)
Renvoie la distance séparant le joueur d'une entité définie par sa collision.

Variables

- char **save_path** [500]

6.30.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
 Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.30.2 Documentation des fonctions

6.30.2.1 afficher_statistiques()

```
void afficher_statistiques (
    joueur_t * perso )
```

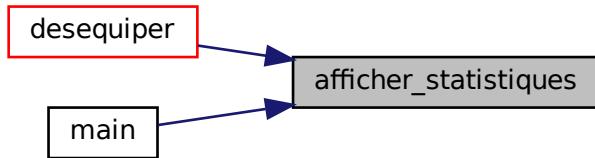
Auteur

Rafael Doneau
Max Descomps

Paramètres

<i>perso</i>	Le joueur sur lequel on se renseigne
--------------	--------------------------------------

Voici le graphe des appelants de cette fonction :



6.30.2.2 charger_sauvegarde_joueur()

```
joueur_t* charger_sauvegarde_joueur (
    char * nom_sauv,
    char * f_src_obj,
    joueur_t * joueurs[],
    unsigned short int nb_joueurs )
```

Auteur

Ange Despert

Cette fonction va récupérer les informations dans la sauvegarde au format JSON.

Il va ensuite detruire les joueurs et la carte, pour ensuite les recréer avec les informations qui correspondent à la sauvegarde puis téléporter le joueur aux coordonnées voulues.

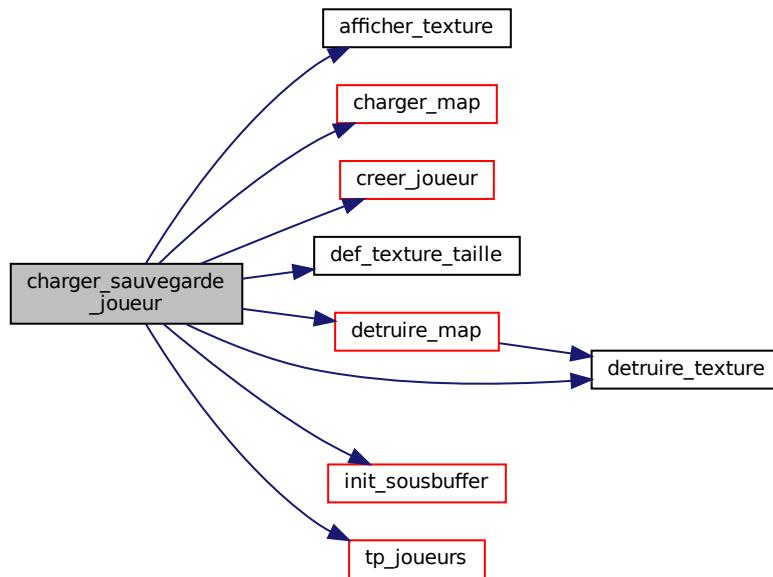
Paramètres

<i>nom_sauv</i>	Chemin complet du fichier de sauvegarde
<i>f_src_obj</i>	Le fichier source contenant les objets du jeu
<i>joueurs</i>	Les joueurs existants
<i>nb_joueurs</i>	Le nombre de joueurs existants

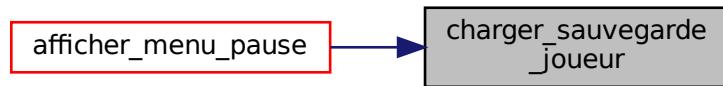
Renvoie

joueur_t*

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.30.2.3 check_repertoire_jeux()

```
void check_repertoire_jeux ( )
```

Auteur

Ange Despert

Cette fonction vérifie si le répertoire de jeux existe (emplacement différent selon les OS).
Puis le créer s'il n'existe pas. Voici le graphe des appelants de cette fonction :



6.30.2.4 copy()

```
static void copy (
    const byte * origin,
    byte * out,
    size_t size ) [static]
```

Voici le graphe des appelants de cette fonction :

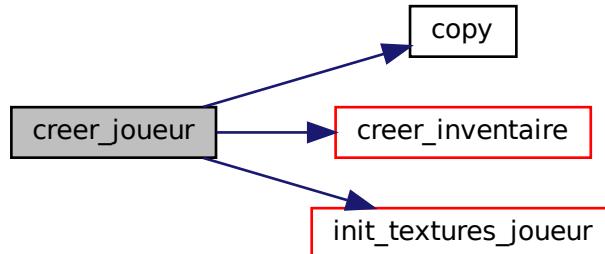


6.30.2.5 creer_joueur()

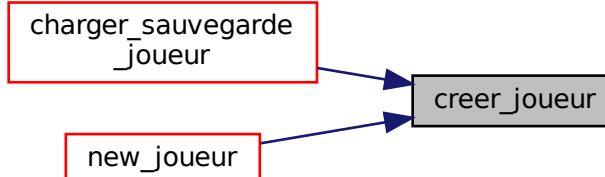
```
joueur_t* creer_joueur (
    const char * nom,
    const int niveau,
    const int xp,
    const int maxPdv,
    const int pdv,
    const int attaque,
    const int defense,
    const int vitesse,
    const byte trig[TAILLE_TRIGGER],
    const t_direction_1 orient,
```

```
const bool bouclier_equipe,
const int num_j,
char * f_src_obj )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.30.2.6 créer_sauvegarde_json()

```
void créer_sauvegarde_json (
    joueur_t * j )
```

Auteur

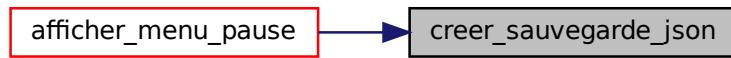
Ange Despert

Cette fonction va créer une sauvegarde dans le répertoire de sauvegarde au format JSON contenant toutes les informations à conserver sur le joueur.

Paramètres

<i>j</i>	Le joueur qui sauvegarde
----------	--------------------------

Voici le graphe des appelants de cette fonction :

**6.30.2.7 detruire_joueur()**

```
void detruire_joueur (
    joueur_t * j )
```

Auteur

Max Descomps

Paramètres

<i>j</i>	Le joueur à détruire
----------	----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.30.2.8 distance_joueur()

```
int distance_joueur (
    SDL_Rect collision,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

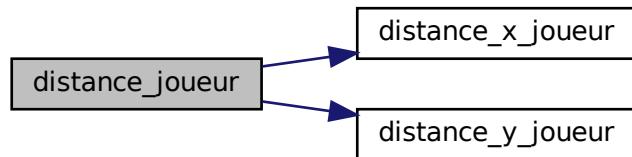
Paramètres

<i>collision</i>	la zone de collision de l'entité
<i>joueur</i>	le joueur

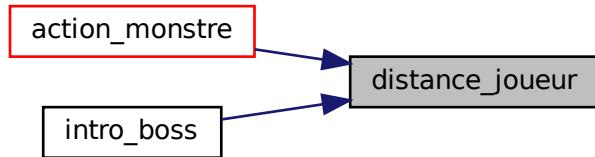
Renvoie

int la distance entre l'entité et le joueur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.30.2.9 `distance_x_joueur()`

```
int distance_x_joueur (
    SDL_Rect collision,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

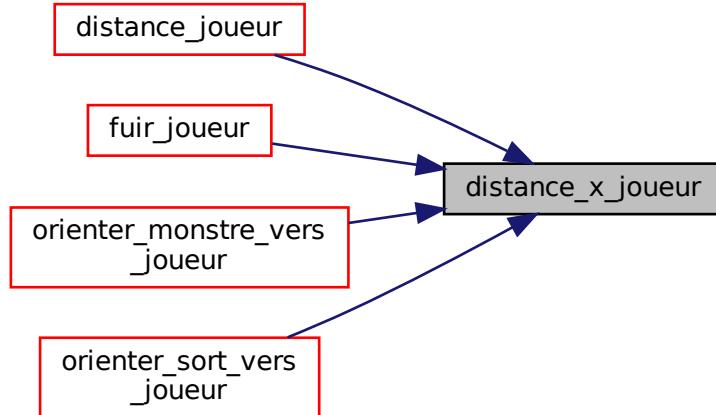
Paramètres

<code>collision</code>	la zone de collision de l'entité
<code>joueur</code>	le joueur

Renvoie

int la distance entre l'entité et le joueur sur l'axe des abscisses

Voici le graphe des appelants de cette fonction :



6.30.2.10 `distance_y_joueur()`

```
int distance_y_joueur (
    SDL_Rect collision,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

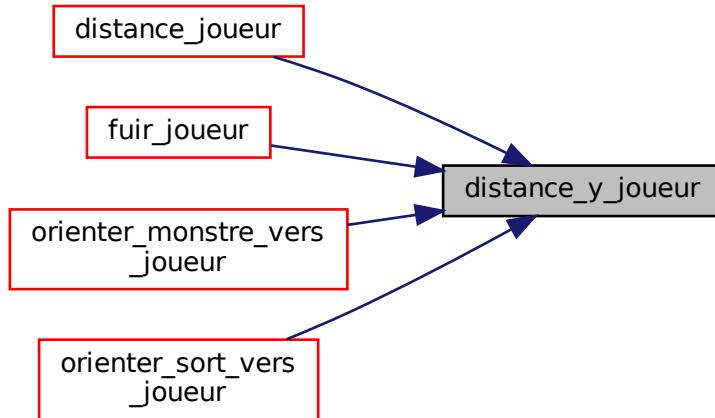
Paramètres

<code>collision</code>	la zone de collision de l'entité
<code>joueur</code>	le joueur

Renvoie

int la distance entre l'entité et le joueur sur l'axe des ordonnées

Voici le graphe des appelants de cette fonction :



6.30.2.11 entite_en_collision()

```

SDL_bool entite_en_collision (
    SDL_Rect * entite_1,
    SDL_Rect * entite_2,
    t_direction_1 * cote_entite_1,
    t_direction_1 * cote_entite_2 )
  
```

Voici le graphe des appelants de cette fonction :



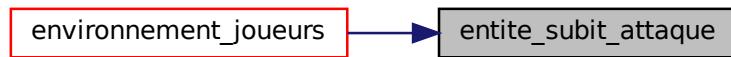
6.30.2.12 entite_subit_attaque()

```
SDL_bool entite_subit_attaque (
    SDL_Rect * monstre_hitbox,
    joueur_t * joueur )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



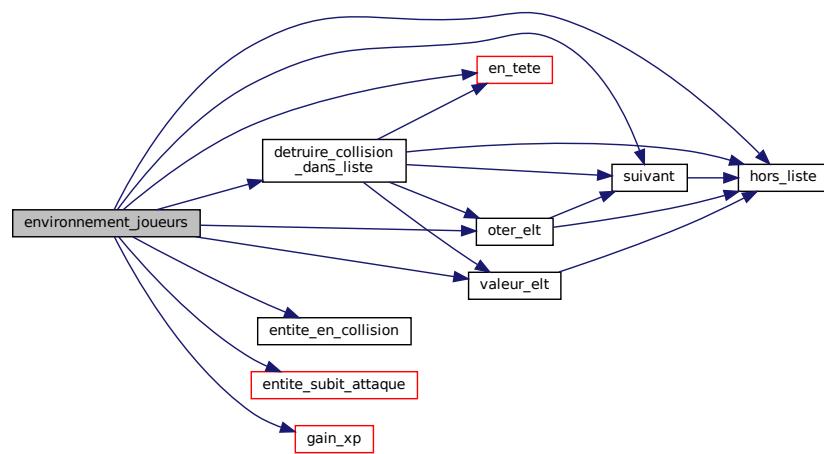
6.30.2.13 environnement_joueurs()

```
void environnement_joueurs (
    list * liste_monstres,
    list * liste_sorts,
    list * liste_coffres,
    joueur_t ** joueurs,
    int nb_joueur )
```

Paramètres

<i>liste_monstres</i>	La liste des monstres du jeu
<i>liste_sorts</i>	La liste des sorts du jeu
<i>liste_coffres</i>	La liste des coffres du jeu
<i>joueurs</i>	Les joueurs qui interagissent avec l'environnement du jeu
<i>nb_joueur</i>	Le nombre de joueurs en jeu

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.30.2.14 gain_xp()

```
void gain_xp (
    joueur_t * perso )
```

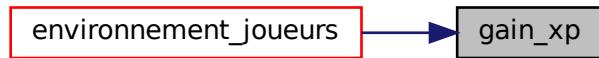
Paramètres

<i>perso</i>	Le joueur qui gagne de l'expérience
--------------	-------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.30.2.15 levelup()

```
void levelup (
    joueur_t * perso )
```

Paramètres

<i>perso</i>	Le joueur qui passe un niveau
--------------	-------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



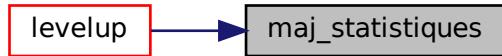
6.30.2.16 maj_statistiques()

```
void maj_statistiques (
    joueur_t * perso )
```

Paramètres

<i>perso</i>	Le joueur qui passe un niveau
--------------	-------------------------------

Voici le graphe des appelants de cette fonction :



6.30.2.17 new_joueur()

```
joueur_t* new_joueur (
    const char * nom,
    int num_j,
    char * f_src_obj )
```

Auteur

Ange Despert

Paramètres

<i>nom</i>	Le nom du joueur
<i>num_j</i>	La place du joueur dans le tableau des joueurs
<i>f_src_obj</i>	Le fichier source des objets du jeu

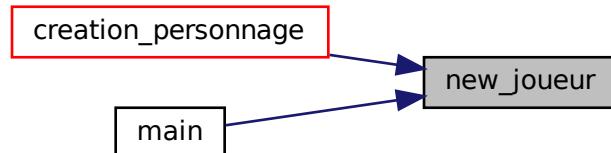
Renvoie

Instance nouvellement allouée du type joueur_t contenant les informations du joueur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.30.2.18 sauv_existe()

```
bool sauv_existe (
    char * nom_sauv )
```

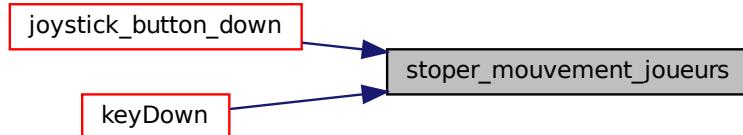
6.30.2.19 stoper_mouvement_joueurs()

```
void stoper_mouvement_joueurs (
    joueur_t ** joueurs )
```

Paramètres

<i>joueurs</i>	Tableau des joueurs en jeu
----------------	----------------------------

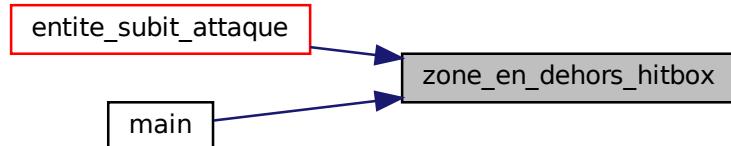
Voici le graphe des appelants de cette fonction :



6.30.2.20 zone_en_dehors_hitbox()

```
SDL_Rect* zone_en_dehors_hitbox (
    SDL_Rect * hitbox,
    SDL_Rect * sprite,
    t_direction_2 orient )
```

Voici le graphe des appelants de cette fonction :



6.30.3 Documentation des variables

6.30.3.1 save_path

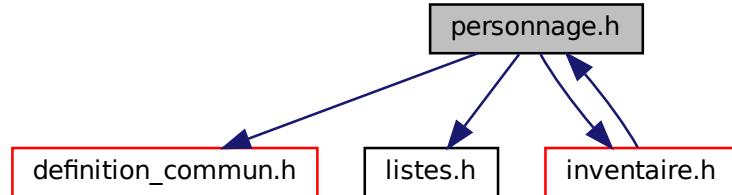
```
char save_path[500]
```

Le répertoire complet de sauvegarde du jeu

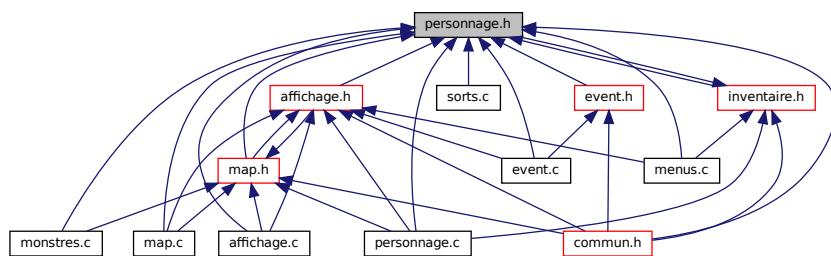
6.31 Référence du fichier personnage.h

Fichier contenant toutes les définitions concernant le personnage.

Graphe des dépendances par inclusion de personnage.h:



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



Classes

- struct `statut_s`
Structure contenant les éléments nécessaires au choix de l'affichage des sprites du personnage.
- struct `joueur_t`
Structure non manipulable hors des fonctions du personnage contenant les informations sur le joueur.

Macros

- #define `DUREE_ATTAQUE_OU_CHARGE` 12
- #define `DUREE_ATTAQUE` 4
- #define `DUREE_ATTAQUE_CHARGE` 10
- #define `DUREE_BLOQUER` 14
- #define `DUREE_JOUEUR_BLESSE` 12
- #define `DUREE_SOIN` 25
- #define `TAILLE_PERSONNAGE` 16 /*La taille du personnage en pixels*/
- #define `TAILLE_TRIGGER` 200

Définitions de type

- typedef unsigned char `byte`

Énumérations

```
— enum action_t {
    RIEN, ATTAQUE, ATTAQUE_CHARGE, CHARGER,
    BLOQUER, ATTAQUE_OU_CHARGER, J_BLESSE, SOIN }
```

Fonctions

```
— void stoper_mouvement_joueurs (joueur_t **joueurs)
    Stop le mouvement des joueurs en jeu.
— joueur_t * creer_joueur (const char *nom, const int niveau, const int xp, const int maxPdv, const int pdv,
    const int attaque, const int defense, const int vitesse, const byte trig[200], const t_direction_1 orient, const
    _Bool bouclier_equipe, const int num_j, char *f_src_obj)
    Creer un joueur.
— joueur_t * new_joueur (const char *nom, int num_j, char *f_src_obj)
    Fonction de création d'un joueur correspondant au modèle standard du jeu.
— void detruire_joueur (joueur_t *j)
    Fonction qui détruit un joueur.
— joueur_t * charger_sauvegarde_joueur (char *nom_sauv, char *f_src_obj, joueur_t *joueurs[], unsigned
    short int nb_joueurs)
    Fonction qui charge une sauvegarde au format JSON.
— void maj_statistiques (joueur_t *perso)
    Fonction qui met à jour les statistiques d'un joueur lors d'un passage de niveau.
— void afficher_statistiques (joueur_t *perso)
    Fonction qui affiche les statistiques d'un joueur dans la console.
— void levelup (joueur_t *perso)
    Fonction qui gère le passage de niveau d'un joueur.
— void gain_xp (joueur_t *perso)
    Fonction qui gère les effets d'un gain d'expérience.
— void creer_sauvegarde_json (joueur_t *j)
    Fonction qui créer les sauvegardes du jeu.
— void check_reperoire_jeux ()
    Fonction gère le réperoire de jeux.
— void environnement_joueurs (list *liste_monstres, list *liste_sorts, list *liste_coffres, joueur_t **joueurs, int
    nb_joueur)
    Fonction qui gère les effets de l'environnement sur le joueur.
— int distance_x_joueur (SDL_Rect collision, joueur_t *joueur)
    Renvoie la distance séparant le joueur d'une entité définit par sa collision sur l'axe des abscisses.
— int distance_y_joueur (SDL_Rect collision, joueur_t *joueur)
    Renvoie la distance séparant le joueur d'une entité définit par sa collision sur l'axe des ordonnées.
— int distance_joueur (SDL_Rect collision, joueur_t *joueur)
    Renvoie la distance séparant le joueur d'une entité définit par sa collision.
— SDL_Rect * zone_en_dehors_hitbox (SDL_Rect *hitbox, SDL_Rect *sprite, t_direction_2 orientation)
```

Variables

```
— char save_path [500]
```

6.31.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
 Max Descomps (Max.Descomps.Etu@univ-lemans.fr)
 Antoine Bruneau (Antoine.Bruneau.Etu@univ-lemans.fr)

Version

0.2

Date

28/03/2022

Copyright

Copyright (c) 2022

6.31.2 Documentation des macros

6.31.2.1 DUREE_ATTAQUE

```
#define DUREE_ATTAQUE 4
```

6.31.2.2 DUREE_ATTAQUE_CHARGEÉE

```
#define DUREE_ATTAQUE_CHARGEÉE 10
```

6.31.2.3 DUREE_ATTAQUE_OU_CHARGEÉE

```
#define DUREE_ATTAQUE_OU_CHARGEÉE 12
```

6.31.2.4 DUREE_BLOQUER

```
#define DUREE_BLOQUER 14
```

6.31.2.5 DUREE_JOUEUR_BLESSE

```
#define DUREE_JOUEUR_BLESSE 12
```

6.31.2.6 DUREE_SOIN

```
#define DUREE_SOIN 25
```

Nombre de sprites dans le spritesheet de soin

6.31.2.7 TAILLE_PERSONNAGE

```
#define TAILLE_PERSONNAGE 16 /*La taille du personnage en pixels*/
```

6.31.2.8 TAILLE_TRIGGER

```
#define TAILLE_TRIGGER 200
```

6.31.3 Documentation des définitions de type

6.31.3.1 byte

```
typedef unsigned char byte
```

6.31.4 Documentation du type de l'énumération

6.31.4.1 action_t

```
enum action_t
```

Valeurs énumérées

RIEN	Aucune action
ATTAQUE	Action d'attaque
ATTAQUE_CHARGEE	Action d'attaque chargée
CHARGER	Action de charge
BLOQUER	Action de blocage
ATTAQUE_OU_CHARGER	Action d'attaque ou de charge
J_BLESSE	Action de blessure
SOIN	Action de soin

6.31.5 Documentation des fonctions

6.31.5.1 afficher_statistiques()

```
void afficher_statistiques (
    joueur_t * perso )
```

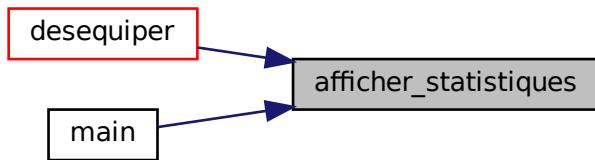
Auteur

Rafael Doneau
Max Descomps

Paramètres

<i>perso</i>	Le joueur sur lequel on se renseigne
--------------	--------------------------------------

Voici le graphe des appels de cette fonction :



6.31.5.2 charger_sauvegarde_joueur()

```
joueur_t* charger_sauvegarde_joueur (
    char * nom_sauv,
    char * f_src_obj,
    joueur_t * joueurs[],
    unsigned short int nb_joueurs )
```

Auteur

Ange Despert

Cette fonction va récupérer les informations dans la sauvegarde au format JSON.

Il va ensuite détruire les joueurs et la carte, pour ensuite les recréer avec les informations qui correspondent à la sauvegarde puis téléporter le joueur aux coordonnées voulues.

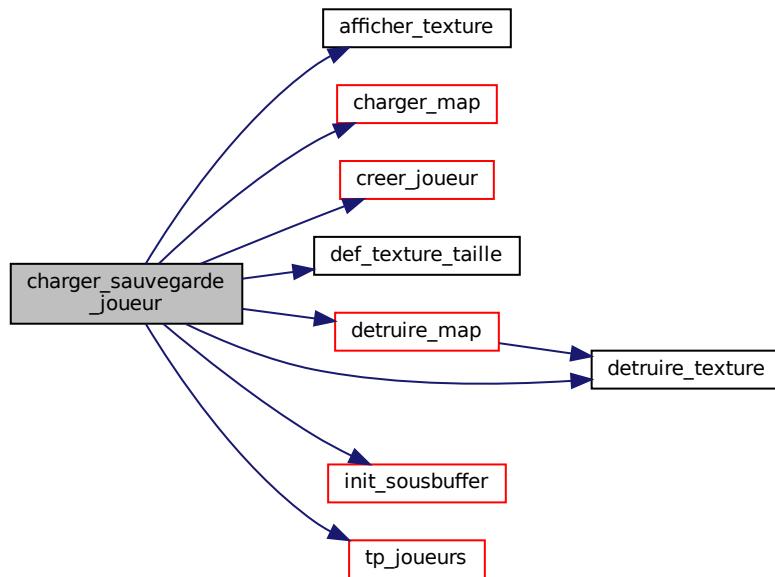
Paramètres

<i>nom_sauv</i>	Chemin complet du fichier de sauvegarde
<i>f_src_obj</i>	Le fichier source contenant les objets du jeu
<i>joueurs</i>	Les joueurs existants
<i>nb_joueurs</i>	Le nombre de joueurs existants

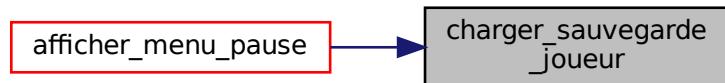
Renvoie

`joueur_t*`

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.31.5.3 check_repertoire_jeux()

```
void check_repertoire_jeux( )
```

Auteur

Ange Despert

Cette fonction vérifie si le répertoire de jeux existe (emplacement différent selon les OS). Puis le créer s'il n'existe pas. Voici le graphe des appelants de cette fonction :

**6.31.5.4 creer_joueur()**

```
joueur_t* creer_joueur (
    const char * nom,
    const int niveau,
    const int xp,
    const int maxPdv,
    const int pdv,
    const int attaque,
    const int defense,
    const int vitesse,
    const byte trig[200],
    const t_direction_1 orient,
    const _Bool bouclier_equipe,
    const int num_j,
    char * f_src_obj )
```

Auteurs

Max Descomps

Ange Despert

Paramètres

<i>nom</i>	Le nom du joueur
<i>niveau</i>	Le niveau du joueur
<i>xp</i>	L'expérience du joueur
<i>maxPdv</i>	Le nombre maximum de points de vie du joueur
<i>pdv</i>	Le nombre de points de vie du joueur
<i>attaque</i>	L'attaque du joueur
<i>defense</i>	La défense du joueur
<i>vitesse</i>	La vitesse du joueur
<i>trig</i>	Le trigger d'information sur le joueur
<i>orient</i>	L'orientation spatiale du joueur
<i>bouclier_equipe</i>	La possession d'un bouclier équipé par le joueur
<i>num_j</i> Généré par Doxygen	La place du joueur dans le tableau des joueurs
<i>f_src_obj</i>	Le fichier source des objets du jeu

Renvoie

Instance nouvellement allouée du type joueur_t contenant les informations du joueur

Paramètres

<i>bouclier_equipe</i>	Définition du type booléen
------------------------	----------------------------

6.31.5.5 creer_sauvegarde_json()

```
void creer_sauvegarde_json (
    joueur_t * j )
```

Auteur

Ange Despert

Cette fonction va créer une sauvegarde dans le répertoire de sauvegarde au format JSON contenant toutes les informations à conserver sur le joueur.

Paramètres

<i>j</i>	Le joueur qui sauvegarde
----------	--------------------------

Voici le graphe des appels de cette fonction :

**6.31.5.6 detruire_joueur()**

```
void detruire_joueur (
    joueur_t * j )
```

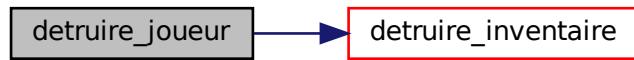
Auteur

Max Descomps

Paramètres

<i>j</i>	Le joueur à détruire
----------	----------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**6.31.5.7 distance_joueur()**

```
int distance_joueur (
    SDL_Rect collision,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

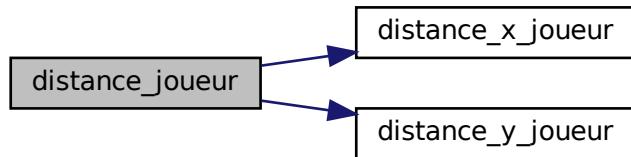
Paramètres

<i>collision</i>	la zone de collision de l'entité
<i>joueur</i>	le joueur

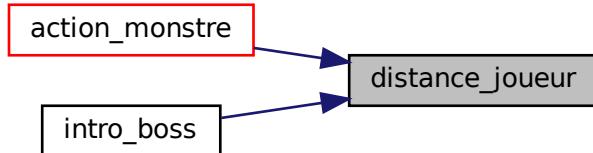
Renvoie

```
int la distance entre l'entité et le joueur
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

**6.31.5.8 distance_x_joueur()**

```
int distance_x_joueur (
    SDL_Rect collision,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

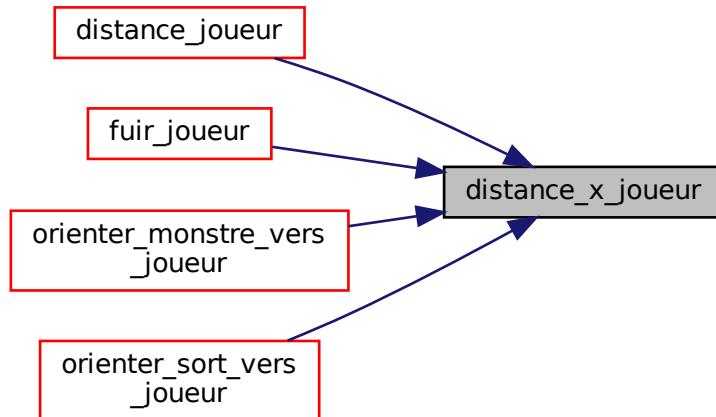
Paramètres

<i>collision</i>	la zone de collision de l'entité
<i>joueur</i>	le joueur

Renvoie

int la distance entre l'entité et le joueur sur l'axe des abscisses

Voici le graphe des appelants de cette fonction :



6.31.5.9 `distance_y_joueur()`

```
int distance_y_joueur (
    SDL_Rect collision,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

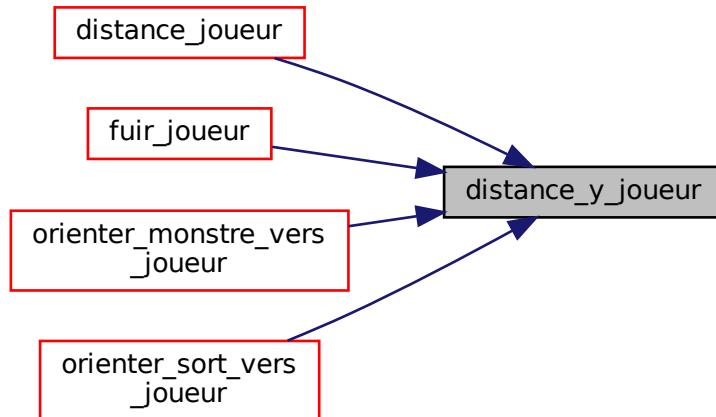
Paramètres

<code>collision</code>	la zone de collision de l'entité
<code>joueur</code>	le joueur

Renvoie

int la distance entre l'entité et le joueur sur l'axe des ordonnées

Voici le graphe des appelants de cette fonction :



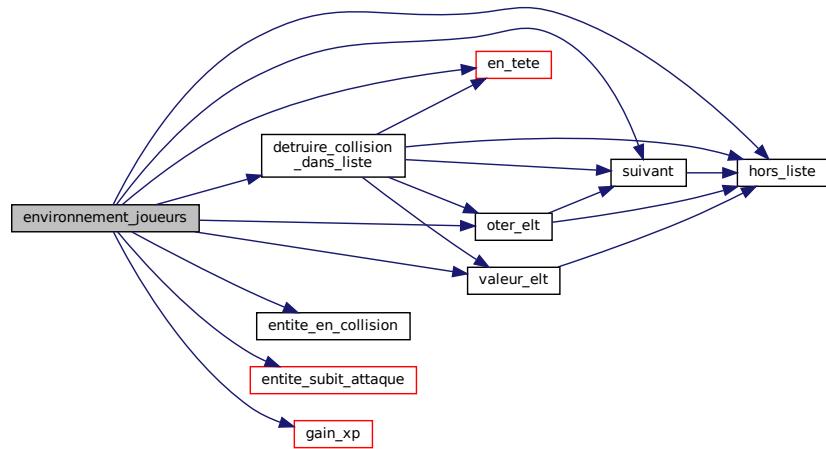
6.31.5.10 environnement_joueurs()

```
void environnement_joueurs (
    list * liste_monstres,
    list * liste_sorts,
    list * liste_coffres,
    joueur_t ** joueurs,
    int nb_joueur )
```

Paramètres

<i>liste_monstres</i>	La liste des monstres du jeu
<i>liste_sorts</i>	La liste des sorts du jeu
<i>liste_coffres</i>	La liste des coffres du jeu
<i>joueurs</i>	Les joueurs qui interagissent avec l'environnement du jeu
<i>nb_joueur</i>	Le nombre de joueurs en jeu

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



6.31.5.11 gain_xp()

```
void gain_xp (
    joueur_t * perso )
```

Paramètres

<code>perso</code>	Le joueur qui gagne de l'expérience
--------------------	-------------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.31.5.12 levelup()

```
void levelup (
    joueur_t * perso )
```

Paramètres

<i>perso</i>	Le joueur qui passe un niveau
--------------	-------------------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



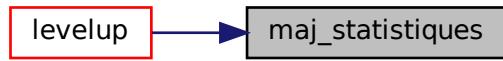
6.31.5.13 maj_statistiques()

```
void maj_statistiques (
    joueur_t * perso )
```

Paramètres

<i>perso</i>	Le joueur qui passe un niveau
--------------	-------------------------------

Voici le graphe des appelants de cette fonction :



6.31.5.14 new_joueur()

```
joueur_t* new_joueur (
    const char * nom,
    int num_j,
    char * f_src_obj )
```

Auteur

Ange Despert

Paramètres

<i>nom</i>	Le nom du joueur
<i>num_j</i>	La place du joueur dans le tableau des joueurs
<i>f_src_obj</i>	Le fichier source des objets du jeu

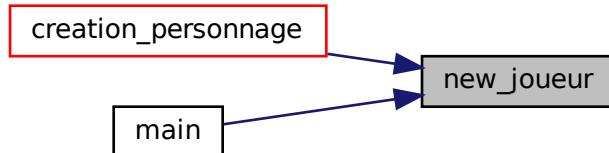
Renvoie

Instance nouvellement allouée du type joueur_t contenant les informations du joueur

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

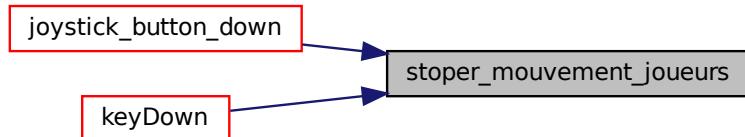
**6.31.5.15 stoper_mouvement_joueurs()**

```
void stoper_mouvement_joueurs (
    joueur_t ** joueurs )
```

Paramètres

<i>joueurs</i>	Tableau des joueurs en jeu
----------------	----------------------------

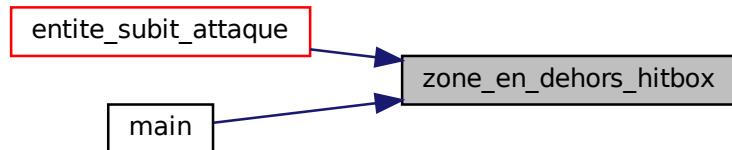
Voici le graphe des appelants de cette fonction :



6.31.5.16 zone_en_dehors_hitbox()

```
SDL_Rect* zone_en_dehors_hitbox (
    SDL_Rect * hitbox,
    SDL_Rect * sprite,
    t_direction_2 orientation )
```

Voici le graphe des appelants de cette fonction :



6.31.6 Documentation des variables

6.31.6.1 save_path

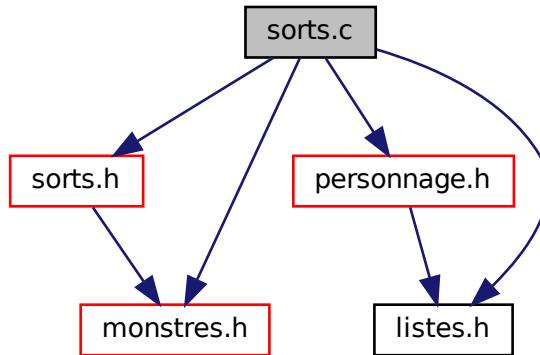
```
char save_path[500]
```

Le répertoire complet de sauvegarde du jeu

6.32 Référence du fichier sorts.c

Fichier contenant toutes les fonctions concernant les sorts.

Graphe des dépendances par inclusion de sorts.c:



Fonctions

- void `détruire_sort (sort_t **sort)`
- void `détruire_sort_cb (void *sort)`
Fonction de "call back" qui détruit une structure sort.
- `sort_t * ajouter_sort (sort_t *sort)`
- void * `ajouter_sort_cb (void *sort)`
Fonction de "call back" qui retourne une structure sort.
- void `init_liste_base_sort (liste_base_monstres_t *liste_base_monstres)`
Fonction qui initialise le tableau de modèles de sort.
- void `creer_sort_monstre (monstre_t *monstre, joueur_t *joueur)`
Fonction qui créer un sort orienté vers le joueur.
- void `orienter_sort_vers_joueur (monstre_t *monstre, sort_t *sort, joueur_t *joueur)`
Fonction qui met à jour la texture sort de façon à ce qu'elle soit orienté vers le joueur.
- void `action_sort (sort_t *sort, joueur_t joueur[2])`

Variables

- base_sort_t `liste_base_sort [3]`

6.32.1 Description détaillée

Auteur

Antoine Bruneau (Antoine.Bruneau.Etu@univ-lemans.fr)

Version

0.1

Date

22/03/2022

Copyright

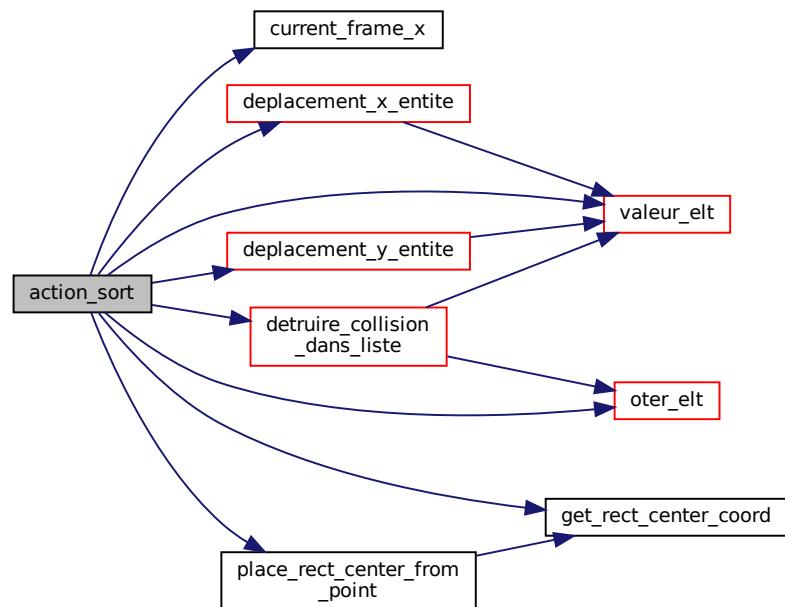
Copyright (c) 2022

6.32.2 Documentation des fonctions

6.32.2.1 action_sort()

```
void action_sort (
    sort_t * sort,
    joueur_t joueur[2] )
```

Voici le graphe d'appel pour cette fonction :



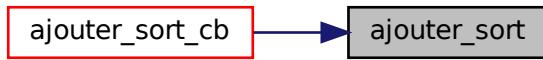
Voici le graphe des appels de cette fonction :



6.32.2.2 ajouter_sort()

```
sort_t* ajouter_sort (
    sort_t * sort )
```

Voici le graphe des appelants de cette fonction :



6.32.2.3 ajouter_sort_cb()

```
void * ajouter_sort_cb (
    void * sort )
```

Auteur

Bruneau Antoine

Paramètres

sort	le sort à retourner
------	---------------------

Renvoie

void* un pointeur générique sur une structure sort

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.32.2.4 creer_sort_monstre()

```
void creer_sort_monstre (
    monstre_t * monstre,
    joueur_t * joueur )
```

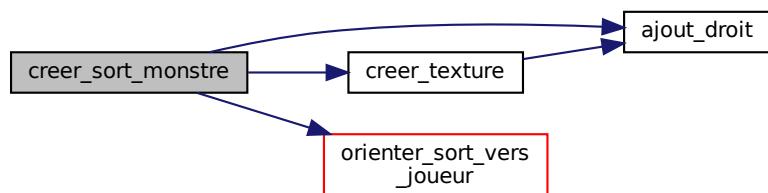
Auteur

Bruneau Antoine

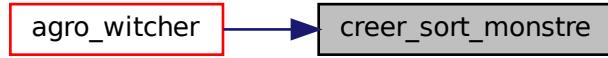
Paramètres

<i>monstre</i>	le monstre qui à créé le sort
<i>joueur</i>	le joueur qui est ciblé par le sort

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.32.2.5 detruire_sort()

```
void detruire_sort (
    sort_t ** sort )
```

Voici le graphe des appelants de cette fonction :



6.32.2.6 detruire_sort_cb()

```
void detruire_sort_cb (
    void * sort )
```

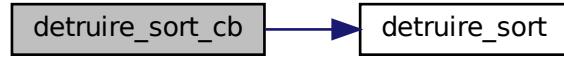
Auteur

Bruneau Antoine

Paramètres

<code>sort</code>	le sort à détruire
-------------------	--------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.32.2.7 init_liste_base_sort()

```
void init_liste_base_sort (
    liste_base_monstres_t * liste_base_monstres )
```

Auteur

Bruneau Antoine

Paramètres

<i>liste_base_monstres</i>	le tableau de modèles de sort
----------------------------	-------------------------------

Voici le graphe des appelants de cette fonction :



6.32.2.8 orienter_sort_vers_joueur()

```
void orienter_sort_vers_joueur (
    monstre_t * monstre,
    sort_t * sort,
    joueur_t * joueur )
```

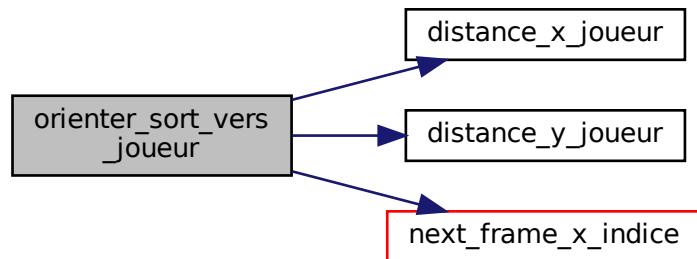
Auteur

Bruneau Antoine

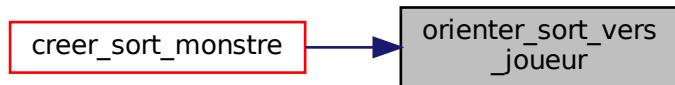
Paramètres

<i>monstre</i>	le monstre qui lance le sort
<i>sort</i>	le sort à orienter
<i>joueur</i>	le joueur cible

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.32.3 Documentation des variables

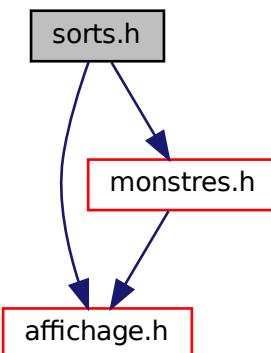
6.32.3.1 liste_base_sort

```
base_sort_t liste_base_sort[3]
```

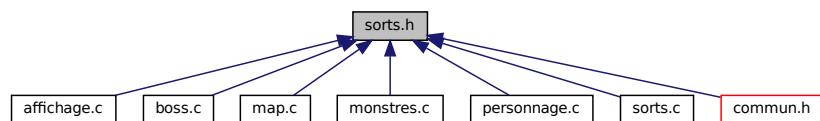
6.33 Référence du fichier sorts.h

Fichier contenant toutes les définitions concernant les sorts.

Graphe des dépendances par inclusion de sorts.h:



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



Classes

- struct `sort_s`
- struct `base_sort_s`

Macros

- #define `PATH_SPELL_BOSS` "ressources/sprite/spell_boss.bmp"
- #define `PATH_SPELL_WITCHER` "ressources/sprite/spell_witcher.bmp"
- #define `CONVERTIR_RADIANT_DEGREE` 57.3

Énumérations

- enum `type_sort_t` { `SP_WITCHER`, `SP_BOSS_BALL`, `SP_BOSS_GROUP`, `SP_BOSS_SPLIT` }
L'énumération des types de sort.

Fonctions

- void `détruire_sort_cb` (void *sort)
Fonction de "call back" qui détruit une structure sort.
- void * `ajouter_sort_cb` (void *sort)
Fonction de "call back" qui retourne une structure sort.
- void `init_liste_base_sort` (`liste_base_monstres_t` *`liste_base_monstres`)
Fonction qui initialise le tableau de modèles de sort.
- void `creer_sort_monstre` (`monstre_t` *monstre, `joueur_t` *joueur)
Fonction qui crée un sort orienté vers le joueur.
- void `action_sort` (`sort_t` *sort, `joueur_t` *joueur)
Fonction qui gère le déplacement du sort sur la map.
- void `orienter_sort_vers_joueur` (`monstre_t` *monstre, `sort_t` *sort, `joueur_t` *joueur)
Fonction qui met à jour la texture sort de façon à ce qu'elle soit orienté vers le joueur.

Variables

- `base_sort_t liste_base_sort` [3]

6.33.1 Description détaillée

Auteur

Bruneau Antoine (`Antoine.Bruneau.Etu@univ-lemans.fr`)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.33.2 Documentation des macros

6.33.2.1 CONVERTIR_RADIANT_DEGREE

```
#define CONVERTIR_RADIANT_DEGREE 57.3
```

6.33.2.2 PATH_SPELL_BOSS

```
#define PATH_SPELL_BOSS "ressources/sprite/spell_boss.bmp"
```

6.33.2.3 PATH_SPELL_WITCHER

```
#define PATH_SPELL_WITCHER "ressources/sprite/spell_witcher.bmp"
```

6.33.3 Documentation du type de l'énumération

6.33.3.1 type_sort_t

```
enum type_sort_t
```

Cela permet de différencier les sorts pour réaliser des actions particulières en fonction de ce type.

Valeurs énumérées

SP_WITCHER	sort du Witcher
SP_BOSS_BALL	le sort "boule" du boss
SP_BOSS_GROUP	le sort "groupé" du boss
SP_BOSS_SPLIT	le sort "split" qui est créé à partir du sort "groupé" du boss

6.33.4 Documentation des fonctions

6.33.4.1 action_sort()

```
void action_sort (
    sort_t * sort,
    joueur_t * joueur )
```

Auteur

Bruneau Antoine

Paramètres

sort	le sort à déplacer
joueur	les joueurs qui peuvent être touchés par le sort

6.33.4.2 ajouter_sort_cb()

```
void* ajouter_sort_cb (
    void * sort )
```

Auteur

Bruneau Antoine

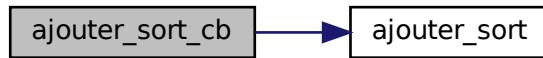
Paramètres

<i>sort</i>	le sort à retourner
-------------	---------------------

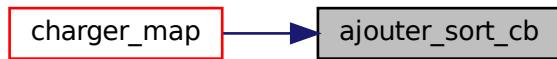
Renvoie

void* un pointeur générique sur une structure sort

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.33.4.3 creer_sort_monstre()

```
void creer_sort_monstre (
    monstre_t * monstre,
    joueur_t * joueur )
```

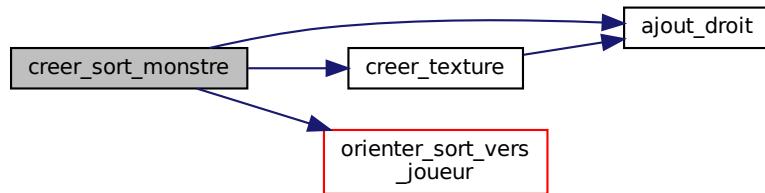
Auteur

Bruneau Antoine

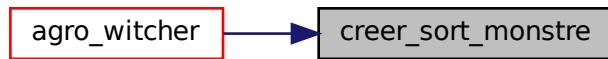
Paramètres

<i>monstre</i>	le monstre qui à créé le sort
<i>joueur</i>	le joueur qui est ciblé par le sort

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**6.33.4.4 détruire_sort_cb()**

```
void détruire_sort_cb (
    void * sort )
```

Auteur

Bruneau Antoine

Paramètres

<i>sort</i>	le sort à détruire
-------------	--------------------

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.33.4.5 init_liste_base_sort()

```
void init_liste_base_sort (
    liste_base_monstres_t * liste_base_monstres )
```

Auteur

Bruneau Antoine

Paramètres

<i>liste_base_monstres</i>	le tableau de modèles de sort
----------------------------	-------------------------------

Voici le graphe des appelants de cette fonction :



6.33.4.6 orienter_sort_vers_joueur()

```
void orienter_sort_vers_joueur (
    monstre_t * monstre,
    sort_t * sort,
    joueur_t * joueur )
```

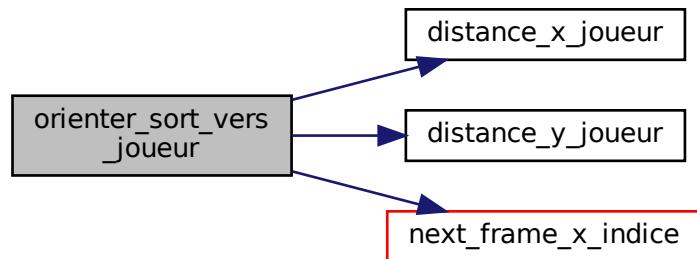
Auteur

Bruneau Antoine

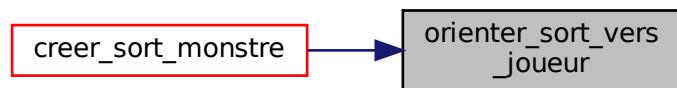
Paramètres

<i>monstre</i>	le monstre qui lance le sort
<i>sort</i>	le sort à orienter
<i>joueur</i>	le joueur cible

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.33.5 Documentation des variables

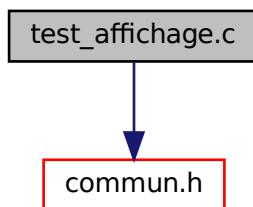
6.33.5.1 liste_base_sort

```
base_sort_t liste_base_sort[3]
```

6.34 Référence du fichier test_affichage.c

Programme de test du module affichage.

Graphe des dépendances par inclusion de test_affichage.c:



Fonctions

- int `main ()`

Variables

- long int `compteur`
- `t_map * test_map`
- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`

6.34.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.34.2 Documentation des fonctions

6.34.2.1 main()

```
int main ( )
```

6.34.3 Documentation des variables

6.34.3.1 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à [NB_FPS](#)

6.34.3.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.34.3.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

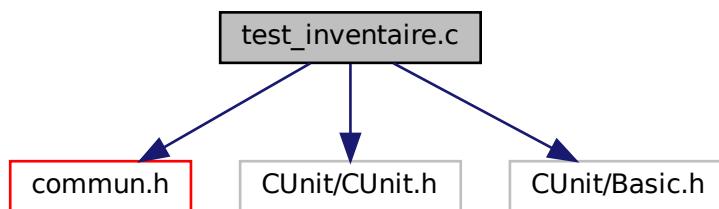
6.34.3.4 test_map

```
t_map* test_map
```

6.35 Référence du fichier test_inventaire.c

Programme de test du module inventaire.

Graphe des dépendances par inclusion de test_inventaire.c:



Fonctions

- int `init_suite` (void)
- int `clean_suite` (void)
- void `generation_inventaire` (void)
- void `modification_inventaire` (void)
- int `main` ()

Variables

- long int `compteur`
- t_map * `test_map`
- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`
- joueur_t * `perso_principal` = NULL
- objet_t * `objet_test` = NULL
- lobjet_t * `sac` = NULL
- lobjet_t * `equipe` = NULL

6.35.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.3

Date

02/04/2022

Copyright

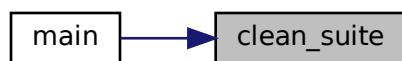
Copyright (c) 2022

6.35.2 Documentation des fonctions

6.35.2.1 clean_suite()

```
int clean_suite (
    void )
```

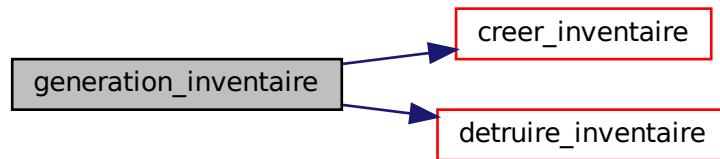
Voici le graphe des appels de cette fonction :



6.35.2.2 generation_inventaire()

```
void generation_inventaire (
    void )
```

Voici le graphe d'appel pour cette fonction :



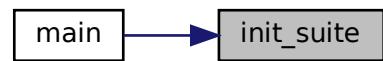
Voici le graphe des appelants de cette fonction :



6.35.2.3 init_suite()

```
int init_suite (
    void )
```

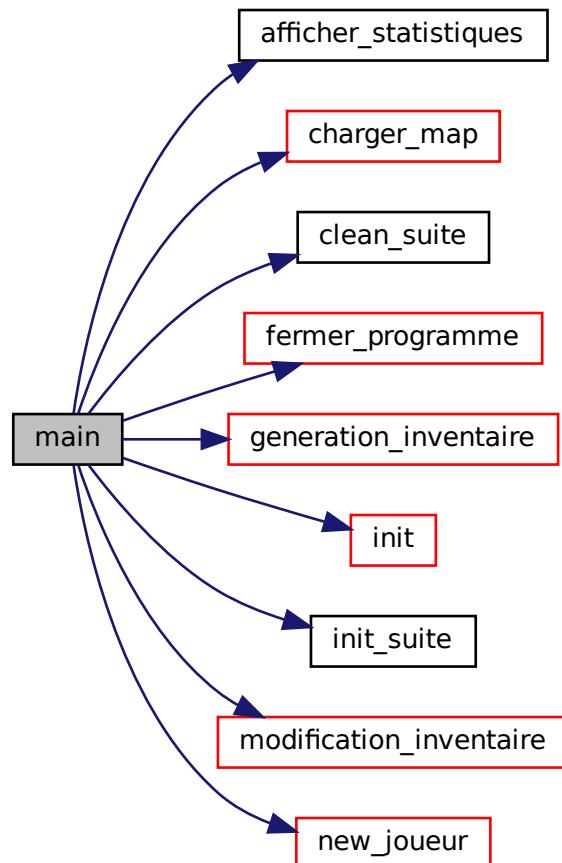
Voici le graphe des appelants de cette fonction :



6.35.2.4 main()

```
int main ( )
```

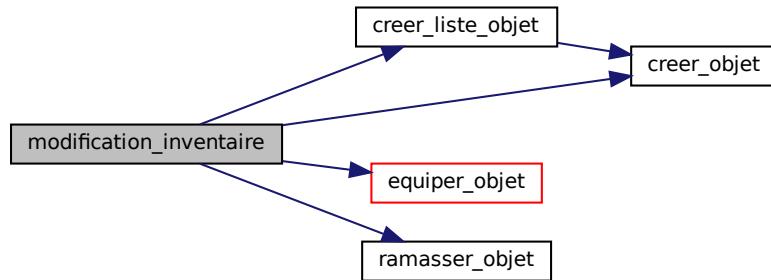
Voici le graphe d'appel pour cette fonction :



6.35.2.5 modification_inventaire()

```
void modification_inventaire (
    void )
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



6.35.3 Documentation des variables

6.35.3.1 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à `NB_FPS`

6.35.3.2 equipe

```
lobjet_t* equipe = NULL
```

6.35.3.3 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.35.3.4 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

6.35.3.5 objet_test

```
objet_t* objet_test = NULL
```

6.35.3.6 perso_principal

```
joueur_t* perso_principal = NULL
```

6.35.3.7 sac

```
objet_t* sac = NULL
```

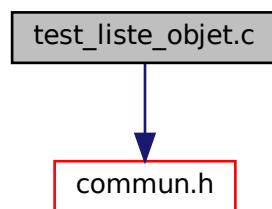
6.35.3.8 test_map

```
t_map* test_map
```

6.36 Référence du fichier test_liste_objet.c

Programme de test du module liste_objet.

Graphe des dépendances par inclusion de test_liste_objet.c:



Fonctions

— int `main ()`

Variables

— long int `compteur`
— t_map * `test_map`
— unsigned int `FENETRE_LONGUEUR`
— unsigned int `FENETRE_LARGEUR`

6.36.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.36.2 Documentation des fonctions

6.36.2.1 main()

```
int main ( )
```

6.36.3 Documentation des variables

6.36.3.1 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à `NB_FPS`

6.36.3.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.36.3.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

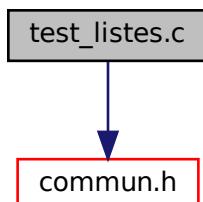
6.36.3.4 test_map

```
t_map* test_map
```

6.37 Référence du fichier test_listes.c

Programme de test du type de structure liste générique.

Graphe des dépendances par inclusion de test_listes.c:



Fonctions

- void `afficher_int` (int *nb)
- int `main` ()

Variables

- long int `compteur`
- t_map * `test_map`
- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`

6.37.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.37.2 Documentation des fonctions

6.37.2.1 afficher_int()

```
void afficher_int (
    int * nb )
```

6.37.2.2 main()

```
int main ( )
```

6.37.3 Documentation des variables

6.37.3.1 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à [NB_FPS](#)

6.37.3.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.37.3.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

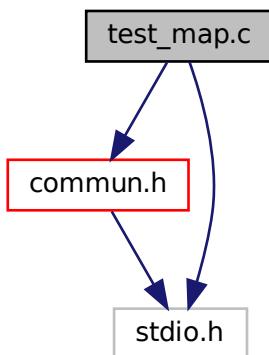
6.37.3.4 test_map

```
t_map* test_map
```

6.38 Référence du fichier test_map.c

Programme de test du module map.

Graphe des dépendances par inclusion de test_map.c:



Fonctions

- int `main` (int argc, char **argv)

Variables

- long int `compteur`
- t_map * `test_map`
- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`

6.38.1 Description détaillée

Auteur

Ange Despert (Ange.Despert.Etu@univ-lemans.fr)
Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

28/03/2022

Copyright

Copyright (c) 2022

6.38.2 Documentation des fonctions

6.38.2.1 main()

```
int main (
    int argc,
    char ** argv )
```

6.38.3 Documentation des variables

6.38.3.1 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à [NB_FPS](#)

6.38.3.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.38.3.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

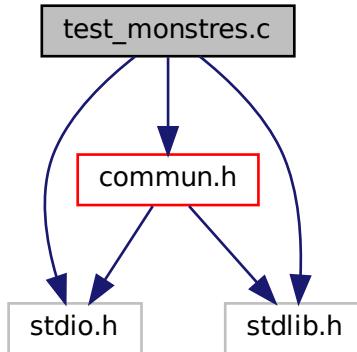
6.38.3.4 test_map

```
t_map* test_map
```

6.39 Référence du fichier test_monstres.c

Programme de test du module monstres.

Graphe des dépendances par inclusion de test_monstres.c:



Fonctions

- int `main ()`

6.39.1 Description détaillée

Auteur

Antoine Bruneau (Antoine.Bruneau.Etu@univ-lemans.fr)

Version

0.1

Date

20/03/2022

Copyright

Copyright (c) 2022

6.39.2 Documentation des fonctions

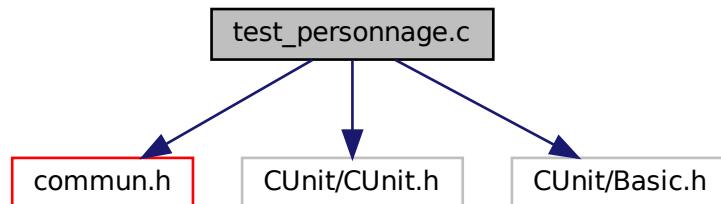
6.39.2.1 main()

```
int main ( )
```

6.40 Référence du fichier test_personnage.c

Programme de test du module personnage.

Graphe des dépendances par inclusion de test_personnage.c:



Fonctions

- int `init_suite` (void)
- int `clean_suite` (void)
- void `creation_personnage` (void)
- void `modification_personnage` (void)
- int `main` ()

Variables

- long int `compteur`
- t_map * `test_map`
- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`
- joueur_t * `perso_principal` = NULL

6.40.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.2

Date

02/04/2022

Copyright

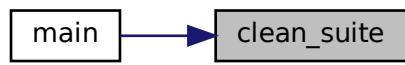
Copyright (c) 2022

6.40.2 Documentation des fonctions

6.40.2.1 clean_suite()

```
int clean_suite (
    void )
```

Voici le graphe des appelants de cette fonction :



6.40.2.2 creation_personnage()

```
void creation_personnage (
    void )
```

Voici le graphe d'appel pour cette fonction :



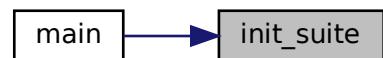
Voici le graphe des appels de cette fonction :



6.40.2.3 init_suite()

```
int init_suite (
    void )
```

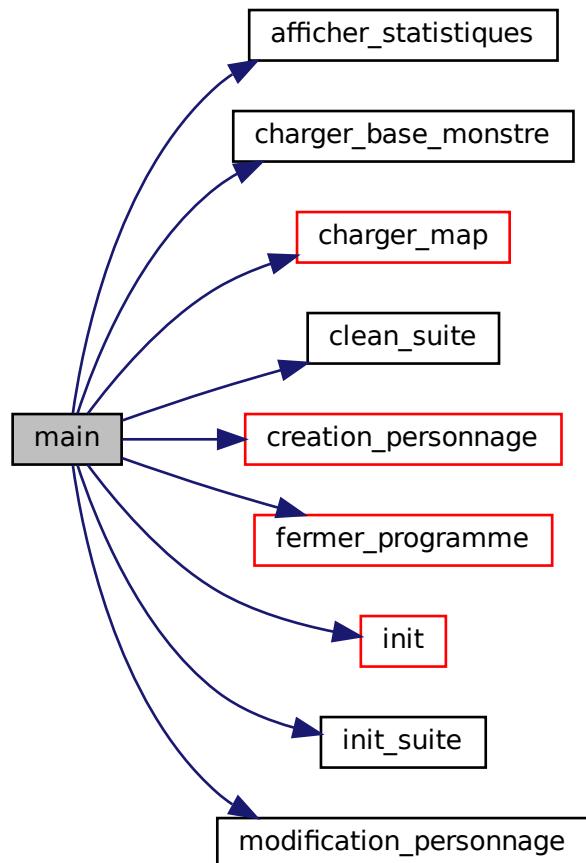
Voici le graphe des appels de cette fonction :



6.40.2.4 main()

```
int main ( )
```

Voici le graphe d'appel pour cette fonction :



6.40.2.5 modification_personnage()

```
void modification_personnage ( void )
```

Voici le graphe des appelants de cette fonction :



6.40.3 Documentation des variables

6.40.3.1 compteur

```
long int compteur
```

Un compteur d'ips qui va de 0 à [NB_FPS](#)

6.40.3.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.40.3.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

6.40.3.4 perso_principal

```
joueur_t* perso_principal = NULL
```

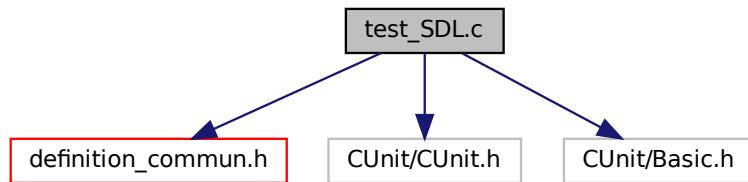
6.40.3.5 test_map

```
t_map* test_map
```

6.41 Référence du fichier test SDL.c

Programme de test des fonctions principales de la SDL.

Graphe des dépendances par inclusion de test(SDL.c):



Fonctions

- int `init_suite` (void)
- int `clean_suite` (void)
- void `test SDL_init` (void)
- void `test SDL_close` (void)
- int `main` ()

Variables

- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`
- `SDL_Window * fenetre_Principale`
- `SDL_Renderer * rendu_principal`

6.41.1 Description détaillée

Auteur

Max Descomps (Max.Descomps.Etu@univ-lemans.fr)

Version

0.1

Date

1/04/2022

Copyright

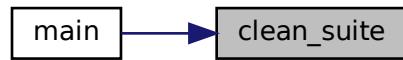
Copyright (c) 2022

6.41.2 Documentation des fonctions

6.41.2.1 clean_suite()

```
int clean_suite (
    void )
```

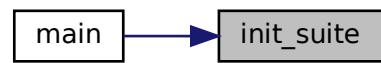
Voici le graphe des appelants de cette fonction :



6.41.2.2 init_suite()

```
int init_suite (
    void )
```

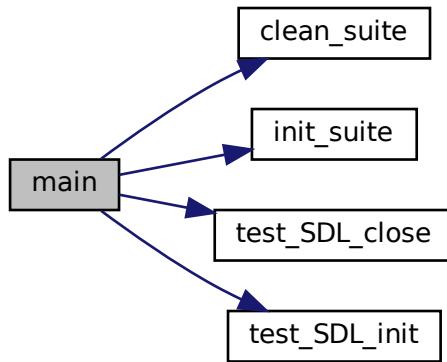
Voici le graphe des appelants de cette fonction :



6.41.2.3 main()

```
int main ( )
```

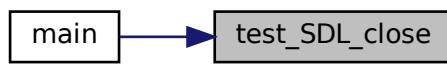
Voici le graphe d'appel pour cette fonction :



6.41.2.4 test SDL_close()

```
void test SDL close (
    void  )
```

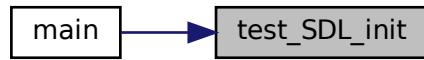
Voici le graphe des appels de cette fonction :



6.41.2.5 test SDL_init()

```
void test	SDL_init (
    void )
```

Voici le graphe des appels de cette fonction :



6.41.3 Documentation des variables

6.41.3.1 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

La largeur de la fenêtre obtenue par requête SDL

6.41.3.2 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

La longueur de la fenêtre obtenue par requête SDL

6.41.3.3 fenetre_Principale

```
SDL_Window* fenetre_Principale
```

Pointeur sur la fenêtre principale du programme

6.41.3.4 rendu_principal

```
SDL_Renderer* rendu_principal
```

Le rendu principal du programme

Index

action
 boss_s, 13
 monstre_s, 26
 statut_s, 32
action_boss_1_t
 boss.h, 157
action_monstre
 monstres.c, 326
 monstres.h, 345
action_monstre_t
 monstres.h, 344
action_sort
 sorts.c, 393
 sorts.h, 401
action_t
 personnage.h, 378
aff
 list, 21
aff_cleanup
 init_close.c, 201
aff_fenetre
 t_aff, 34
affichage.c, 41
 afficher_animations, 44
 afficher_buffer, 45
 afficher_coffres, 46
 afficher_monstres, 47
 afficher_sorts, 48
 afficher_texture, 49
 ajout_text_liste, 50
 appliquer_coord_rect, 51
 bloquer, 94
 color, 51
 compteur, 94
 creer_texture, 52
 current_frame_x, 54
 current_frame_y, 55
 def_texture_taille, 55
 deplacement_x_entite, 56
 deplacement_x_joueur_secondeaire, 57
 deplacement_x_pers, 58
 deplacement_y_entite, 59
 deplacement_y_joueur_secondeaire, 61
 deplacement_y_pers, 62
 deplacer_rect_haut_droit, 63
 deplacer_rect_origine, 64
 deplacer_texture_bas_droit, 65
 deplacer_texture_bas_gauche, 65
 deplacer_texture_centre, 66
 deplacer_texture_haut_droit, 67
 deplacer_texture_origine, 67
 detruire_collision_dans_liste, 68
 detruire_liste_textures, 69
 detruire_texture, 69
 fenetre_finale, 94
 FENETRE_LARGEUR, 94
 FENETRE_LONGUEUR, 94
 get_rect_center, 70
 get_rect_center_coord, 71
 get_screen_center, 71
 heal, 95
 info_texture, 72
 init_animations, 73
 init_texture_joueur, 73
 init_textures_joueur, 74
 liste_animations, 95
 listeDeTextures, 95
 lister_animations, 75
 modif_affichage_rect, 76
 multiplicateur_x, 95
 multiplicateur_y, 95
 next_frame_animation, 76
 next_frame_indice, 77
 next_frame_joueur, 78
 next_frame_x, 79
 next_frame_x_indice, 80
 next_frame_y, 81
 next_frame_y_indice, 82
 place_rect_center_from_point, 83
 placer_texture, 84
 point_in_rect, 85
 rect_centre, 85
 rect_centre_rect, 86
 rect_centre_rect_x, 87
 rect_centre_rect_y, 88
 rect_centre_x, 88
 rect_centre_y, 89
 rect_correct_texture, 90
 rect_ecran_to_rect_map, 91
 rects_egal_x, 91
 rects_egal_y, 92
 text_copier_position, 93
 tx, 95
 ty, 95
 update_frame_texture, 93
affichage.h, 96
 afficher_animations, 102
 afficher_buffer, 102

afficher_coffres, 103
 afficher_monstres, 104
 afficher_sorts, 105
 afficher_texture, 106
 ajout_text_liste, 107
 bloquer, 150
 color, 108
 compteur, 150
 creer_texture, 109
 current_frame_x, 112
 current_frame_y, 113
 def_texture_taille, 113
 deplacement_x_entite, 114
 deplacement_x_joueur_secondeaire, 115
 deplacement_x_pers, 116
 deplacement_y_entite, 117
 deplacement_y_joueur_secondeaire, 119
 deplacement_y_pers, 120
 déplacer_rect_haut_droit, 121
 déplacer_rect_origine, 122
 déplacer_texture_bas_droit, 123
 déplacer_texture_bas_gauche, 123
 déplacer_texture_centre, 124
 déplacer_texture_haut_droit, 125
 déplacer_texture_origine, 125
 detruire_collision_dans_liste, 126
 detruire_liste_textures, 127
 detruire_texture, 127
 fenetre_finale, 150
 get_rect_center, 128
 get_rect_center_coord, 129
 heal, 151
 info_texture, 129
 init_animations, 130
 init_texture_joueur, 131
 init_textures_joueur, 131
 LARGEUR_ENTITE, 99
 liste_animations, 151
 listeDeTextures, 151
 lister_animations, 132
 LONGUEUR_ENTITE, 99
 modif_affichage_rect, 133
 multiplicateur_x, 151
 multiplicateur_y, 151
 N_T_ATTAQUE, 99
 N_T_ATTAQUE2, 100
 N_T_ATTAQUE_CHARGEES, 100
 N_T_ATTAQUE_CHARGEES2, 100
 N_T_CHARGER, 100
 N_T_CHARGER2, 100
 N_T_MARCHER, 100
 N_T_MARCHER2, 100
 N_T_MARCHER_BOUCLE, 100
 N_T_MARCHER_BOUCLE2, 101
 NB_FPS, 101
 NB_SPRITE_JOUEUR, 101
 next_frame_animation, 133
 next_frame_indice, 134
 next_frame_joueur, 135
 next_frame_x, 136
 next_frame_x_indice, 137
 next_frame_y, 138
 next_frame_y_indice, 139
 place_rect_center_from_point, 140
 placer_texture, 141
 rect_centre, 142
 rect_centre_rect, 143
 rect_centre_rect_x, 144
 rect_centre_rect_y, 144
 rect_centre_x, 145
 rect_centre_y, 146
 rect_correct_texture, 147
 rects_egal_x, 148
 rects_egal_y, 149
 t_texture_perso, 101
 TEXT_ATTAQUE, 101
 TEXT_ATTAQUE_CHARGEES, 101
 TEXT_CHARGER, 101
 text_copier_position, 149
 TEXT_MARCHER, 101
 TEXT_MARCHER_BOUCLE, 101
 tx, 151
 ty, 151
 affichage_chargement_attaque_boss
 boss.c, 152
 affichage_clonage_boss
 boss.c, 153
 afficher_animations
 affichage.c, 44
 affichage.h, 102
 afficher_boss
 boss.c, 154
 afficher_bosses
 boss.c, 154
 afficher_buffer
 affichage.c, 45
 affichage.h, 102
 afficher_coffres
 affichage.c, 46
 affichage.h, 103
 afficher_int
 test_listes.c, 415
 afficher_intro
 main.c, 288
 afficher_inventaire
 menus.c, 306
 menus.h, 317
 afficher_inventaire_manette
 menus.c, 307
 menus.h, 317
 afficher_liste
 listes.c, 248
 listes.h, 268
 afficher_liste_objet
 liste_objet.c, 231
 liste_objet.h, 240

afficher_menu_accueil
 menus.c, 308
 menus.h, 318
afficher_menu_accueil_manette
 menus.c, 309
 menus.h, 319
afficher_menu_pause
 menus.c, 310
 menus.h, 320
afficher_menu_pause_manette
 menus.c, 311
 menus.h, 321
afficher_monstres
 affichage.c, 47
 affichage.h, 104
afficher_objet
 objet.c, 351
 objet.h, 356
afficher_sorts
 affichage.c, 48
 affichage.h, 105
afficher_statistiques
 personnage.c, 359
 personnage.h, 378
afficher_texture
 affichage.c, 49
 affichage.h, 106
afficher_textures_equipe
 liste_objet.c, 231
 liste_objet.h, 240
afficher_textures_sac
 liste_objet.c, 232
 liste_objet.h, 241
afficher_zone_tp
 map.c, 291
agro_knight
 monstres.c, 327
agro_monstre
 monstres.c, 328
agro_witcher
 monstres.c, 328
ajout
 list, 21
ajout_droit
 listes.c, 249
 listes.h, 269
ajout_gauche
 listes.c, 250
 listes.h, 270
ajout_text_liste
 affichage.c, 50
 affichage.h, 107
ajouter_monstre
 monstres.c, 329
ajouter_monstre_cb
 monstres.c, 330
 monstres.h, 346
ajouter_sort
 sorts.c, 393
 ajouter_sort_cb
 sorts.c, 394
 sorts.h, 402
 amulette
 objet.h, 356
 animation
 statut_s, 32
APPARITION
 boss.h, 157
appliquer_coord_rect
 affichage.c, 51
arme
 objet.h, 356
ATTAQUE
 boss.h, 157
 personnage.h, 378
attaque
 base_monstre_s, 10
 boss_s, 13
 joueur_t, 19
 monstre_s, 26
 objet_t, 28
attaque_actif
 joueur_t, 19
ATTAQUE_CHARGEES
 personnage.h, 378
ATTAQUE_OU_CHARGER
 personnage.h, 378
AUCUNE_ERREUR
 code_erreur.h, 160
AVANT_ATTAQUE
 boss.h, 157
AVANT_DISPARITION
 boss.h, 157
base_coffre_s, 9
 fichier_image, 9
 hitbox, 9
 nom_coffre, 9
base_monstre_s, 10
 attaque, 10
 fichier_image, 10
 gainXp, 10
 hitbox, 10
 nom_monstre, 10
 pdv, 11
 vitesse, 11
base_monstre_t, 11
base_sort_s, 11
 collision, 11
 degat, 12
 type, 12
BLESSE
 boss.h, 157
BLOQUER
 personnage.h, 378
bloquer
 affichage.c, 94

affichage.h, 150
 bool
 definition_commun.h, 180
 boss.c, 152
 affichage_chargement_attaque_boss, 152
 affichage_clonage_boss, 153
 afficher_boss, 154
 afficher_bosses, 154
 creer_boss_clone, 154
 deplacement_boss_aleatoire, 155
 intro_boss, 155
 boss.h, 156
 action_boss_1_t, 157
 APPARITION, 157
 ATTAQUE, 157
 AVANT_ATTAQUE, 157
 AVANT_DISPARITION, 157
 BLESSÉ, 157
 CHEMIN_BOSS, 157
 CLONAGE, 157
 CLONE, 158
 DEPLACEMENT, 157
 DISPARITION, 157
 DUREE_CLONAGE, 157
 MORT, 157
 PRINCIPAL, 158
 SOMMEIL, 157
 type_boss_1_t, 158
 boss_s, 12
 action, 13
 attaque, 13
 cible, 13
 collision, 13
 duree, 13
 pdv, 13
 texture, 13
 texture_temp, 13
 type, 14
 xp, 14
 bouclier
 objet.h, 356
 bouclier_equipe
 statut_s, 32
 BUFFER_EMPTY
 code_erreur.h, 160
 byte
 definition_commun.h, 181
 personnage.h, 378
 CAPACITE_SAC
 inventaire.h, 222
 cases_x
 t_map, 38
 cases_y
 t_map, 38
 changement_statistiques
 inventaire.c, 212
 inventaire.h, 222
 CHARGER
 personnage.h, 378
 charger_base_coffre
 coffres.c, 163
 coffres.h, 171
 charger_base_monstre
 monstres.c, 331
 monstres.h, 347
 charger_map
 map.c, 291
 map.h, 300
 charger_sauvegarde_joueur
 personnage.c, 360
 personnage.h, 379
 check_reperoire_jeux
 personnage.c, 361
 personnage.h, 380
 CHEMIN_BOSS
 boss.h, 157
 cible
 boss_s, 13
 clean_suite
 test_inventaire.c, 408
 test_personnage.c, 420
 test SDL.c, 425
 CLONAGE
 boss.h, 157
 CLONE
 boss.h, 158
 code_erreur.h, 158
 AUCUNE_ERREUR, 160
 BUFFER_EMPTY, 160
 ERR_CREATION_REPERTOIRE_SAUVAGEARDE, 161
 ERR_RECTANGLE_TOO_BIG, 161
 err_t, 160
 erreur, 159
 ERREUR_FICHIER, 160
 ERREUR_FONCTION, 160
 ERREUR_INIT, 161
 ERREUR_JSON_CLE_NON_TROUVEE, 161
 ERREUR_LISTE, 161
 ERREUR_MAP, 160
 ERREUR SDL_AUDIO, 161
 ERREUR SDL_EVENT, 161
 ERREUR SDL_IMG, 161
 ERREUR SDL_JOYSTICK, 161
 ERREUR SDL_KEYBOARD, 161
 ERREUR SDL_MIX, 161
 ERREUR SDL_MOUSE, 161
 ERREUR SDL_MOUSEBUTTON, 161
 ERREUR SDL_MOUSEMOTION, 161
 ERREUR SDL_MUSIC, 161
 ERREUR SDL_PIXEL, 161
 ERREUR SDL_QUIT, 161
 ERREUR SDL_RENDER, 161
 ERREUR SDL_SCANCODE, 161
 ERREUR SDL_SURFACE, 161
 ERREUR SDL_TIMER, 161

ERREUR SDL_TIMER_START, 161
ERREUR SDL_TIMER_STOP, 161
ERREUR SDL_TTF, 161
ERREUR SDL_WINDOW, 161
ERREUR_TEXTURE, 160
OUT_OF_MEM, 160
SDL_ERREUR, 160
types_erreur, 160
warning, 159
COFFRE_FACE_OUVERT
coffres.h, 170
COFFRE_INCONNU
coffres.h, 171
COFFRE_PROFIL_OUVERT
coffres.h, 170
coffre_s, 14
collision, 14
etat, 14
id_cle, 14
id_loot, 15
orientation, 15
texture, 15
type, 15
coffre_t, 15
coffres.c, 161
charger_base_coffre, 163
creer_coffre, 163
info_coffre, 164
interaction_coffre, 165
inverser_direction, 166
liste_base_coffres, 168
nom_coffre_to_type_coffre, 167
coffres.h, 168
charger_base_coffre, 171
COFFRE_FACE_OUVERT, 170
COFFRE_INCONNU, 171
COFFRE_PROFIL_OUVERT, 170
creer_coffre, 171
etat_coffre_t, 170
FACE_FERME, 171
FACE_OUVERT, 171
FERME, 170
info_coffre, 173
interaction_coffre, 173
inverser_direction, 174
liste_base_coffres, 176
nom_coffre_to_type_coffre, 175
OUVERT, 170
PROFIL_FERME, 171
PROFIL_OUVERT, 171
type_coffre_t, 171
collision
base_sort_s, 11
boss_s, 13
coffre_s, 14
monstre_s, 26
sort_s, 30
color
affichage.c, 51
affichage.h, 108
commun.h, 176
init, 177
compteur
affichage.c, 94
affichage.h, 150
test_affichage.c, 407
test_inventaire.c, 411
test_liste_objet.c, 413
test_listes.c, 415
test_map.c, 417
test_personnage.c, 423
compteur_frame_anim
t_aff, 35
consommable
objet.h, 356
consommer_objet
inventaire.c, 212
inventaire.h, 223
CONVERTIR_RADIANT_DEGREE
sorts.h, 400
copy
personnage.c, 362
creation_personnage
test_personnage.c, 420
creer_boss_clone
boss.c, 154
creer_coffre
coffres.c, 163
coffres.h, 171
creer_inventaire
inventaire.c, 213
inventaire.h, 223
creer_inventaire_j2
menus.c, 312
menus.h, 322
creer_joueur
personnage.c, 362
personnage.h, 381
creer_liste_objet
liste_objet.c, 233
liste_objet.h, 242
creer_liste_objet_equipe
liste_objet.c, 234
liste_objet.h, 243
creer_liste_objet_vide
liste_objet.c, 234
liste_objet.h, 243
creer_monstre
monstres.c, 331
monstres.h, 348
creer_objet
objet.c, 352
objet.h, 356
creer_sauvegarde_json
personnage.c, 363
personnage.h, 382

creer_sort_monstre
 sorts.c, 395
 sorts.h, 402
 creer_texture
 affichage.c, 52
 affichage.h, 109
 creer_textures_objets
 liste_objet.c, 235
 liste_objet.h, 244
 current_frame_x
 affichage.c, 54
 affichage.h, 112
 current_frame_y
 affichage.c, 55
 affichage.h, 113
 def_texture_taille
 affichage.c, 55
 affichage.h, 113
 defense
 joueur_t, 19
 objet_t, 28
 defense_actif
 joueur_t, 19
 definition_commun.h, 178
 bool, 180
 byte, 181
 EST_1, 181
 EST_2, 182
 faux, 180
 FENETRE_LARGEUR, 183
 FENETRE_LONGUEUR, 183
 fenetre_Principale, 183
 fermer_programme, 182
 NORD_1, 181
 NORD_2, 182
 NORD_EST_2, 182
 NORD_OUEST_2, 182
 OUEST_1, 181
 OUEST_2, 182
 rendu_principal, 184
 running, 184
 SAVE_PATH, 180
 SUD_1, 181
 SUD_2, 182
 SUD_EST_2, 182
 SUD_OUEST_2, 182
 t_direction_1, 181
 t_direction_2, 181
 vrai, 181
 degat
 base_sort_s, 12
 sort_s, 30
 del
 list, 22
 DEPLACEMENT
 boss.h, 157
 deplacement_boss_aleatoire
 boss.c, 155
 deplacement_x_entite
 affichage.c, 56
 affichage.h, 114
 deplacement_x_joueur_secondaire
 affichage.c, 57
 affichage.h, 115
 deplacement_x_pers
 affichage.c, 58
 affichage.h, 116
 deplacement_y_entite
 affichage.c, 59
 affichage.h, 117
 deplacement_y_joueur_secondaire
 affichage.c, 61
 affichage.h, 119
 deplacement_y_pers
 affichage.c, 62
 affichage.h, 120
 deplacer_rect_haut_droit
 affichage.c, 63
 affichage.h, 121
 deplacer_rect_origine
 affichage.c, 64
 affichage.h, 122
 deplacer_texture_bas_droit
 affichage.c, 65
 affichage.h, 123
 deplacer_texture_bas_gauche
 affichage.c, 65
 affichage.h, 123
 deplacer_texture_centre
 affichage.c, 66
 affichage.h, 124
 deplacer_texture_haut_droit
 affichage.c, 67
 affichage.h, 125
 deplacer_texture_origine
 affichage.c, 67
 affichage.h, 125
 desequiper
 inventaire.c, 214
 inventaire.h, 224
 desequiper_slot
 inventaire.c, 215
 inventaire.h, 224
 dest
 zone_tp, 40
 detruire_collision_dans_liste
 affichage.c, 68
 affichage.h, 126
 detruire_inventaire
 inventaire.c, 216
 inventaire.h, 225
 detruire_joueur
 personnage.c, 364
 personnage.h, 382
 detruire_liste
 listes.c, 250

listes.h, 270
detruire_liste_base_monstres
monstres.c, 332
monstres.h, 348
detruire_liste_objet
liste_objet.c, 236
liste_objet.h, 245
detruire_liste_textures
affichage.c, 69
affichage.h, 127
detruire_map
map.c, 292
map.h, 301
detruire_monstre
monstres.c, 332
detruire_monstre_cb
monstres.c, 332
monstres.h, 349
detruire_objet
objet.c, 353
objet.h, 357
detruire_renderer
init_close.c, 201
detruire_sort
sorts.c, 396
detruire_sort_cb
sorts.c, 396
sorts.h, 403
detruire_texture
affichage.c, 69
affichage.h, 127
DISPARITION
boss.h, 157
DISTANCE_AGRO
monstres.h, 343
distance_joueur
personnage.c, 365
personnage.h, 383
distance_x_joueur
personnage.c, 366
personnage.h, 384
distance_y_joueur
personnage.c, 367
personnage.h, 385
draw_rect_epaisseur
menus.c, 313
duree
boss_s, 13
monstre_s, 26
statut_s, 32
duree_anim
statut_s, 32
DUREE_ATTAQUE
personnage.h, 377
DUREE_ATTAQUE_CHARGE
personnage.h, 377
DUREE_ATTAQUE_OU_CHARGE
personnage.h, 377
DUREE_BLOQUER
personnage.h, 377
DUREE_CLONAGE
boss.h, 157
duree_frame_anim
t_aff, 35
DUREE_JOUEUR_BLESSE
personnage.h, 377
DUREE_MONSTRE_ATTAQUE
monstres.h, 344
DUREE_MONSTRE_BLESSE
monstres.h, 344
DUREE_MONSTRE_EN_GARDE
monstres.h, 344
DUREE_MONSTRE_MARCHER
monstres.h, 344
DUREE_MONSTRE_PAUSE
monstres.h, 344
DUREE_RUSH_OU_FUITE
monstres.h, 344
DUREE_SOIN
personnage.h, 377
ec
list, 22
effacer_liste_objet
liste_objet.c, 236
liste_objet.h, 245
element, 16
pred, 16
succ, 16
valeur, 16
en_mouvement
statut_s, 32
en_queue
listes.c, 251
listes.h, 271
en_tete
listes.c, 252
listes.h, 272
entite_en_collision
personnage.c, 368
entite_subit_attaque
personnage.c, 368
environnement_joueurs
personnage.c, 369
personnage.h, 386
equipe
inventaire_t, 17
test_inventaire.c, 411
equiper_objet
inventaire.c, 217
inventaire.h, 226
equiper_sac_slot
inventaire.c, 218
inventaire.h, 227
ERR_CREATION_REPERTOIRE_SAUVAGEARDE
code_erreur.h, 161
ERR_RECTANGLE_TOO_BIG

code_erreur.h, 161
err_t
 code_erreur.h, 160
erreur
 code_erreur.h, 159
ERREUR_FICHIER
 code_erreur.h, 160
ERREUR_FONCTION
 code_erreur.h, 160
ERREUR_INIT
 code_erreur.h, 161
ERREUR_JSON_CLE_NON_TROUVEE
 code_erreur.h, 161
ERREUR_LISTE
 code_erreur.h, 161
ERREUR_MAP
 code_erreur.h, 160
ERREUR SDL_AUDIO
 code_erreur.h, 161
ERREUR SDL_EVENT
 code_erreur.h, 161
ERREUR SDL_IMG
 code_erreur.h, 161
ERREUR SDL_JOYSTICK
 code_erreur.h, 161
ERREUR SDL_KEYBOARD
 code_erreur.h, 161
ERREUR SDL_MIX
 code_erreur.h, 161
ERREUR SDL_MOUSE
 code_erreur.h, 161
ERREUR SDL_MOUSEBUTTON
 code_erreur.h, 161
ERREUR SDL_MOUSEMOTION
 code_erreur.h, 161
ERREUR SDL_MUSIC
 code_erreur.h, 161
ERREUR SDL_PIXEL
 code_erreur.h, 161
ERREUR SDL_QUIT
 code_erreur.h, 161
ERREUR SDL_RENDER
 code_erreur.h, 161
ERREUR SDL_SCANCODE
 code_erreur.h, 161
ERREUR SDL_SURFACE
 code_erreur.h, 161
ERREUR SDL_TIMER
 code_erreur.h, 161
ERREUR SDL_TIMER_START
 code_erreur.h, 161
ERREUR SDL_TIMER_STOP
 code_erreur.h, 161
ERREUR SDL_TTF
 code_erreur.h, 161
ERREUR SDL_WINDOW
 code_erreur.h, 161
ERREUR_TEXTURE
 code_erreur.h, 160
EST_1
 definition_commun.h, 181
EST_2
 definition_commun.h, 182
etat
 coffre_s, 14
etat_coffre_t
 coffres.h, 170
event.c, 184
 jeu_event, 185
 jeu_event_manette, 186
 joystick_button_down, 187
 joystick_button_up, 188
 joystick_stick, 189
 keyDown, 190
 keyUp, 191
 logo_passer, 191
 manette, 194
 mouseButtonDown, 192
 mouseButtonUp, 193
event.h, 194
 jeu_event, 196
 jeu_event_manette, 197
 logo_passer, 198
 manette, 199
 TOUCHE_BAS, 196
 TOUCHE_CONSUMMABLE, 196
 TOUCHE_DROITE, 196
 TOUCHE_ECHAP, 196
 TOUCHE_GAUCHE, 196
 TOUCHE_HAUT, 196
 TOUCHE_TAB, 196
f_close
 init_close.c, 206
FACE_FERME
 coffres.h, 171
FACE_OUVERT
 coffres.h, 171
faux
 definition_commun.h, 180
fenetre_finale
 affichage.c, 94
 affichage.h, 150
FENETRE_LARGEUR
 affichage.c, 94
 definition_commun.h, 183
 test_affichage.c, 407
 test_inventaire.c, 411
 test_liste_objet.c, 413
 test_listes.c, 415
 test_map.c, 417
 test_personnage.c, 423
 test SDL.c, 427
FENETRE_LONGUEUR
 affichage.c, 94
 definition_commun.h, 183
 test_affichage.c, 407

test_inventaire.c, 411
test_liste_objet.c, 414
test_listes.c, 415
test_map.c, 417
test_personnage.c, 423
test SDL.c, 427
fenetre_Principale
 definition_commun.h, 183
 init_close.c, 206
 test SDL.c, 427
fenetre_sous_rendu
 init_close.c, 206
FERME
 coffres.h, 170
fermer_programme
 definition_commun.h, 182
 init_close.c, 201
fermer SDL
 init_close.c, 203
fichier_image
 base_coffre_s, 9
 base_monstre_s, 10
flag
 list, 22
frame_anim
 t_aff, 35
fuir_joueur
 monstres.c, 333

gain_xp
 personnage.c, 370
 personnage.h, 387
gainXp
 base_monstre_s, 10
 monstre_s, 26
generation_inventaire
 test_inventaire.c, 408
get_rect_center
 affichage.c, 70
 affichage.h, 128
get_rect_center_coord
 affichage.c, 71
 affichage.h, 129
get_screen_center
 affichage.c, 71

heal
 affichage.c, 95
 affichage.h, 151
height
 t_aff, 35
 t_map, 38
hitbox
 base_coffre_s, 9
 base_monstre_s, 10
hors_hitbox
 init_close.c, 206
hors_liste
 listes.c, 254

 listes.h, 274
 hors_map_monstre
 map.c, 294

 id
 objet_t, 28
 id_cle
 coffre_s, 14
 id_loot
 coffre_s, 15
 id_map
 t_map, 38
 zone_tp, 40
 info_coffre
 coffres.c, 164
 coffres.h, 173
 info_texture
 affichage.c, 72
 affichage.h, 129
 init
 commun.h, 177
 init_close.c, 204
 init_affichage
 init_close.c, 204
 init_animations
 affichage.c, 73
 affichage.h, 130
 init_close.c, 199
 aff_cleanup, 201
 detruire_renderer, 201
 f_close, 206
 fenetre_Principale, 206
 fenetre_sous_rendu, 206
 fermer_programme, 201
 fermer(SDL, 203
 hors_hitbox, 206
 init, 204
 init_affichage, 204
 init_rc_commun, 205
 init SDL, 205
 rendu_principal, 206
 running, 207
 sous_rendu, 207
 init_liste
 listes.c, 256
 listes.h, 276
 init_liste_base_sort
 sorts.c, 397
 sorts.h, 404
 init_rc_commun
 init_close.c, 205
init SDL
 init_close.c, 205
init_sousbuffer
 map.c, 294
 map.h, 302
init_suite
 test_inventaire.c, 409
 test_personnage.c, 421

test SDL.c, 425
 init_text_menus
 menus.c, 314
 menus.h, 323
 init_texture_joueur
 affichage.c, 73
 affichage.h, 131
 init_textures_joueur
 affichage.c, 74
 affichage.h, 131
 interaction_coffre
 coffres.c, 165
 coffres.h, 173
 interface.c, 207
 RenderHPBar, 208
 interface.h, 209
 RenderHPBar, 210
 intro_boss
 boss.c, 155
 inventaire
 joueur_t, 19
 inventaire.c, 211
 changement_statistiques, 212
 consommer_objet, 212
 creer_inventaire, 213
 desequiper, 214
 desequiper_slot, 215
 detruire_inventaire, 216
 equiper_objet, 217
 equiper_sac_slot, 218
 ramasser_objet, 219
 tout_ramasser, 220
 inventaire.h, 220
 CAPACITE_SAC, 222
 changement_statistiques, 222
 consommer_objet, 223
 creer_inventaire, 223
 desequiper, 224
 desequiper_slot, 224
 detruire_inventaire, 225
 equiper_objet, 226
 equiper_sac_slot, 227
 ramasser_objet, 228
 tout_ramasser, 229
 inventaire_t, 17, 18
 equipe, 17
 sac, 17
 inverser_direction
 coffres.c, 166
 coffres.h, 174

 J_BLESSE
 personnage.h, 378
 jeu_event
 event.c, 185
 event.h, 196
 jeu_event_manette
 event.c, 186
 event.h, 197

 joueur_t, 18
 attaque, 19
 attaque_actif, 19
 defense, 19
 defense_actif, 19
 inventaire, 19
 maxPdv, 19
 niveau, 20
 nom_pers, 20
 pdv, 20
 statut, 20
 textures_joueur, 20
 trigger, 20
 vitesse, 20
 vitesse_actif, 20
 xp, 21
 joystick_button_down
 event.c, 187
 joystick_button_up
 event.c, 188
 joystick_stick
 event.c, 189

 keyDown
 event.c, 190
 keyUp
 event.c, 191
 KNIGHT
 monstres.h, 345

 LARGEUR_ENTITE
 affichage.h, 99
 levelup
 personnage.c, 371
 personnage.h, 388
 list, 21
 aff, 21
 ajout, 21
 del, 22
 ec, 22
 flag, 22
 nb_elem, 22
 liste
 lobjet_t, 25
 t_l_aff, 36
 liste_animations
 affichage.c, 95
 affichage.h, 151
 liste_base_coffres
 coffres.c, 168
 coffres.h, 176
 liste_base_coffres_s, 22
 nb_coffre, 23
 tab, 23
 liste_base_monstres
 monstres.c, 341
 monstres.h, 350
 liste_base_monstres_t, 23, 24
 nb_monstre, 23

tab, 24
liste_base_sort
 sorts.c, 398
 sorts.h, 405
liste_coffres
 t_map, 38
liste_collisions
 t_map, 38
liste_monstres
 t_map, 38
liste_objet.c, 230
 afficher_liste_objet, 231
 afficher_textures_equipe, 231
 afficher_textures_sac, 232
 creer_liste_objet, 233
 creer_liste_objet_equipe, 234
 creer_liste_objet_vide, 234
 creer_textures_objets, 235
 detruire_liste_objet, 236
 effacer_liste_objet, 236
 objets, 238
 placer_objet_sac, 237
liste_objet.h, 238
 afficher_liste_objet, 240
 afficher_textures_equipe, 240
 afficher_textures_sac, 241
 creer_liste_objet, 242
 creer_liste_objet_equipe, 243
 creer_liste_objet_vide, 243
 creer_textures_objets, 244
 detruire_liste_objet, 245
 effacer_liste_objet, 245
 objets, 247
 placer_objet_sac, 246
liste_sorts
 t_map, 38
liste_vide
 listes.c, 257
 listes.h, 277
liste_zone_tp
 t_map, 39
listeDeTextures
 affichage.c, 95
 affichage.h, 151
lister_animations
 affichage.c, 75
 affichage.h, 132
listes.c, 247
 afficher_liste, 248
 ajout_droit, 249
 ajout_gauche, 250
 detruire_liste, 250
 en_queue, 251
 en_tete, 252
 hors_liste, 254
 init_liste, 256
 liste_vide, 257
 modif_elt, 258
 oter_elt, 259
 precedent, 260
 selectionner_element, 261
 suivant, 262
 taille_liste, 263
 valeur_elt, 264
 vider_liste, 265
listes.h, 266
 afficher_liste, 268
 ajout_droit, 269
 ajout_gauche, 270
 detruire_liste, 270
 en_queue, 271
 en_tete, 272
 hors_liste, 274
 init_liste, 276
 liste_vide, 277
 modif_elt, 278
 oter_elt, 279
 precedent, 280
 selectionner_element, 281
 suivant, 282
 taille_liste, 283
 valeur_elt, 284
 vider_liste, 285
lobjet_t, 24, 25
 liste, 25
 nb, 25
logo_passer
 event.c, 191
 event.h, 198
LONGUEUR_ENTITE
 affichage.h, 99
main
 main.c, 288
 test_affichage.c, 407
 test_inventaire.c, 409
 test_liste_objet.c, 413
 test_listes.c, 415
 test_map.c, 417
 test_monstres.c, 419
 test_personnage.c, 421
 test SDL.c, 425
main.c, 286
 afficher_intro, 288
 main, 288
 SDL_MAIN_HANDLED, 287
maj_statistiques
 personnage.c, 372
 personnage.h, 389
manette
 event.c, 194
 event.h, 199
map
 map.c, 298
 map.h, 305
map.c, 290
 afficher_zone_tp, 291

charger_map, 291
 detruire_map, 292
 hors_map_monstre, 294
 init_sousbuffer, 294
 map, 298
 taille_ecran_cases, 295
 texture_map, 295
 tp_joueurs, 296
 transition, 296
 map.h, 298
 charger_map, 300
 detruire_map, 301
 init_sousbuffer, 302
 map, 305
 TAILLE_CASE, 299
 texture_map, 303
 tp_joueurs, 304
 transition, 304
 marcher_monstre
 monstres.c, 334
 maxPdv
 joueur_t, 19
 menus.c, 305
 afficher_inventaire, 306
 afficher_inventaire_manette, 307
 afficher_menu_accueil, 308
 afficher_menu_accueil_manette, 309
 afficher_menu_pause, 310
 afficher_menu_pause_manette, 311
 creer_inventaire_j2, 312
 draw_rect_epaisseur, 313
 init_text_menus, 314
 text_accueil, 315
 text_inventaire1, 315
 text_inventaire2, 315
 text_pause, 315
 menus.h, 315
 afficher_inventaire, 317
 afficher_inventaire_manette, 317
 afficher_menu_accueil, 318
 afficher_menu_accueil_manette, 319
 afficher_menu_pause, 320
 afficher_menu_pause_manette, 321
 creer_inventaire_j2, 322
 init_text_menus, 323
 text_accueil, 323
 text_inventaire1, 323
 text_inventaire2, 324
 text_pause, 324
 modif_affichage_rect
 affichage.c, 76
 affichage.h, 133
 modif_elt
 listes.c, 258
 listes.h, 278
 modification_inventaire
 test_inventaire.c, 410
 modification_personnage
 test_personnage.c, 422
 MONSTRE_ATTAQUE
 monstres.h, 345
 monstre_attaque
 monstres.c, 335
 MONSTRE_BLESSE
 monstres.h, 345
 MONSTRE_EN_GARDE
 monstres.h, 345
 monstre_en_garde
 monstres.c, 336
 MONSTRE_MARCHER
 monstres.h, 345
 MONSTRE_PAUSE
 monstres.h, 345
 monstre_s, 26
 action, 26
 attaque, 26
 collision, 26
 duree, 26
 gainXp, 26
 orientation, 26
 pdv, 27
 texture, 27
 type, 27
 vitesse, 27
 monstre_t, 27
 monstres.c, 324
 action_monstre, 326
 agro_knight, 327
 agro_monstre, 328
 agro_witcher, 328
 ajouter_monstre, 329
 ajouter_monstre_cb, 330
 charger_base_monstre, 331
 creer_monstre, 331
 detruire_liste_base_monstres, 332
 detruire_monstre, 332
 detruire_monstre_cb, 332
 fuir_joueur, 333
 liste_base_monstres, 341
 marcher_monstre, 334
 monstre_attaque, 335
 monstre_en_garde, 336
 nom_monstre_to_type_monstre, 337
 orienter_monstre, 337
 orienter_monstre_vers_joueur, 338
 ronde_monstre, 339
 rush_joueur, 340
 monstres.h, 341
 action_monstre, 345
 action_monstre_t, 344
 ajouter_monstre_cb, 346
 charger_base_monstre, 347
 creer_monstre, 348
 detruire_liste_base_monstres, 348
 detruire_monstre_cb, 349
 DISTANCE_AGRO, 343

DUREE_MONSTRE_ATTAQUE, 344
DUREE_MONSTRE_BLESSE, 344
DUREE_MONSTRE_EN_GARDE, 344
DUREE_MONSTRE_MARCHER, 344
DUREE_MONSTRE_PAUSE, 344
DUREE_RUSH_OU_FUITE, 344
KNIGHT, 345
liste_base_monstres, 350
MONSTRE_ATTAQUE, 345
MONSTRE_BLESSE, 345
MONSTRE_EN_GARDE, 345
MONSTRE_MARCHER, 345
MONSTRE_PAUSE, 345
nom_monstre_to_type_monstre, 349
RUSH_OU_FUITE, 345
TYPE_MONSTRE_INCONNU, 345
type_monstre_t, 345
WITCHER, 345
MORT
 boss.h, 157
mouseButtonDown
 event.c, 192
mouseButtonUp
 event.c, 193
multipli_taille
 t_aff, 35
multiplicateur_x
 affichage.c, 95
 affichage.h, 151
multiplicateur_y
 affichage.c, 95
 affichage.h, 151
N_T_ATTAQUE
 affichage.h, 99
N_T_ATTAQUE2
 affichage.h, 100
N_T_ATTAQUE_CHARGEES
 affichage.h, 100
N_T_ATTAQUE_CHARGEES2
 affichage.h, 100
N_T_CHARGER
 affichage.h, 100
N_T_CHARGER2
 affichage.h, 100
N_T_MARCHER
 affichage.h, 100
N_T_MARCHER2
 affichage.h, 100
N_T_MARCHER_BOUCLEIERS
 affichage.h, 100
N_T_MARCHER_BOUCLEIERS2
 affichage.h, 101
nb
 lobjet_t, 25
nb_coffre
 liste_base_coffres_s, 23
nb_elem
 list, 22
NB_FPS
 affichage.h, 101
nb_monstre
 liste_base_monstres_t, 23
NB_SPRITE_JOUEUR
 affichage.h, 101
NB_TYPE_OBJ
 objet.h, 355
nb_valeurs
 t_l_aff, 36
new_joueur
 personnage.c, 372
 personnage.h, 389
next_frame_animation
 affichage.c, 76
 affichage.h, 133
next_frame_indice
 affichage.c, 77
 affichage.h, 134
next_frame_joueur
 affichage.c, 78
 affichage.h, 135
next_frame_x
 affichage.c, 79
 affichage.h, 136
next_frame_x_indice
 affichage.c, 80
 affichage.h, 137
next_frame_y
 affichage.c, 81
 affichage.h, 138
next_frame_y_indice
 affichage.c, 82
 affichage.h, 139
niveau
 joueur_t, 20
 objet_t, 28
nom
 objet_t, 28
nom_coffre
 base_coffre_s, 9
nom_coffre_to_type_coffre
 coffres.c, 167
 coffres.h, 175
nom_monstre
 base_monstre_s, 10
nom_monstre_to_type_monstre
 monstres.c, 337
 monstres.h, 349
nom_pers
 joueur_t, 20
NORD_1
 definition_commun.h, 181
NORD_2
 definition_commun.h, 182
NORD_EST_2
 definition_commun.h, 182
NORD_OUEST_2

definition_commun.h, 182
objet.c, 350
 afficher_objet, 351
 creer_objet, 352
 detruire_objet, 353
objet.h, 354
 afficher_objet, 356
 amulette, 356
 arme, 356
 bouclier, 356
 consommable, 356
 creer_objet, 356
 detruire_objet, 357
NB_TYPE_OBJ, 355
 protection, 356
 quete, 356
 t_item, 355
objet_t, 28, 29
 attaque, 28
 defense, 28
 id, 28
 niveau, 28
 nom, 28
 texture, 29
 texture_src, 29
 type, 29
 vitesse, 29
objet_test
 test_inventaire.c, 412
objets
 liste_objet.c, 238
 liste_objet.h, 247
orient_att
 statut_s, 32
orient_dep
 statut_s, 33
orientation
 coffre_s, 15
 monstre_s, 26
orienter_monstre
 monstres.c, 337
orienter_monstre_vers_joueur
 monstres.c, 338
orienter_sort_vers_joueur
 sorts.c, 397
 sorts.h, 404
oter_elt
 listes.c, 259
 listes.h, 279
QUEST_1
 definition_commun.h, 181
QUEST_2
 definition_commun.h, 182
OUT_OF_MEM
 code_erreur.h, 160
OUVERT
 coffres.h, 170
PATH_SPELL_BOSS
 sorts.h, 400
PATH_SPELL_WITCHER
 sorts.h, 401
pdv
 base_monstre_s, 11
 boss_s, 13
 joueur_t, 20
 monstre_s, 27
perso_principal
 test_inventaire.c, 412
 test_personnage.c, 423
personnage.c, 358
 afficher_statistiques, 359
 charger_sauvegarde_joueur, 360
 check_repertoire_jeux, 361
 copy, 362
 creer_joueur, 362
 creer_sauvegarde_json, 363
 detruire_joueur, 364
 distance_joueur, 365
 distance_x_joueur, 366
 distance_y_joueur, 367
 entite_en_collision, 368
 entite_subit_attaque, 368
 environnement_joueurs, 369
 gain_xp, 370
 levelup, 371
 maj_statistiques, 372
 new_joueur, 372
 sauv_existe, 373
 save_path, 374
 stoper_mouvement_joueurs, 373
 zone_en_dehors_hitbox, 374
personnage.h, 375
 action_t, 378
 afficher_statistiques, 378
 ATTAQUE, 378
 ATTAQUE_CHARGEES, 378
 ATTAQUE_OU_CHARGER, 378
 BLOQUER, 378
 byte, 378
 CHARGER, 378
 charger_sauvegarde_joueur, 379
 check_repertoire_jeux, 380
 creer_joueur, 381
 creer_sauvegarde_json, 382
 detruire_joueur, 382
 distance_joueur, 383
 distance_x_joueur, 384
 distance_y_joueur, 385
 DUREE_ATTAQUE, 377
 DUREE_ATTAQUE_CHARGEES, 377
 DUREE_ATTAQUE_OU_CHARGEES, 377
 DUREE_BLOQUER, 377
 DUREE_JOUEUR_BLESSE, 377
 DUREE_SOIN, 377
 environnement_joueurs, 386

gain_xp, 387
J_BLESSE, 378
levelup, 388
maj_statistiques, 389
new_joueur, 389
RIEN, 378
save_path, 391
SOIN, 378
stoper_mouvement_joueurs, 390
TAILLE_PERSONNAGE, 377
TAILLE_TRIGGER, 378
zone_en_dehors_hitbox, 391
place_rect_center_from_point
 affichage.c, 83
 affichage.h, 140
placer_objet_sac
 liste_objet.c, 237
 liste_objet.h, 246
placer_texture
 affichage.c, 84
 affichage.h, 141
point, 30
 x, 30
 y, 30
point_in_rect
 affichage.c, 85
precedent
 listes.c, 260
 listes.h, 280
pred
 element, 16
PRINCIPAL
 boss.h, 158
PROFIL_FERME
 coffres.h, 171
PROFIL_OUVERT
 coffres.h, 171
protection
 objet.h, 356
quete
 objet.h, 356
ramasser_objet
 inventaire.c, 219
 inventaire.h, 228
rect_centre
 affichage.c, 85
 affichage.h, 142
rect_centre_rect
 affichage.c, 86
 affichage.h, 143
rect_centre_rect_x
 affichage.c, 87
 affichage.h, 144
rect_centre_rect_y
 affichage.c, 88
 affichage.h, 144
rect_centre_x
 affichage.c, 88
 affichage.h, 144
rect_centre_y
 affichage.c, 89
 affichage.h, 146
rect_correct_texture
 affichage.c, 90
 affichage.h, 147
rect_ecran_to_rect_map
 affichage.c, 91
rects_egal_x
 affichage.c, 91
 affichage.h, 148
rects_egal_y
 affichage.c, 92
 affichage.h, 149
RenderHPBar
 interface.c, 208
 interface.h, 210
rendu_principal
 definition_commun.h, 184
 init_close.c, 206
 test SDL.c, 427
RIEN
 personnage.h, 378
ronde_monstre
 monstres.c, 339
running
 definition_commun.h, 184
 init_close.c, 207
rush_joueur
 monstres.c, 340
RUSH_OU_FUITE
 monstres.h, 345
sac
 inventaire_t, 17
 test_inventaire.c, 412
sauv_existe
 personnage.c, 373
SAVE_PATH
 definition_commun.h, 180
save_path
 personnage.c, 374
 personnage.h, 391
SDL_ERREUR
 code_erreur.h, 160
SDL_MAIN_HANDLED
 main.c, 287
selectionner_element
 listes.c, 261
 listes.h, 281
SOIN
 personnage.h, 378
SOMMEIL
 boss.h, 157
sort_s, 30
 collision, 30
 degat, 30

texture, 31
 type, 31
sort_t, 31
sorts.c, 392
 action_sort, 393
 ajouter_sort, 393
 ajouter_sort_cb, 394
 creer_sort_monstre, 395
 detruire_sort, 396
 detruire_sort_cb, 396
 init_liste_base_sort, 397
 liste_base_sort, 398
 orienter_sort_vers_joueur, 397
sorts.h, 399
 action_sort, 401
 ajouter_sort_cb, 402
 CONVERTIR_RADIAN_DEGREE, 400
 creer_sort_monstre, 402
 detruire_sort_cb, 403
 init_liste_base_sort, 404
 liste_base_sort, 405
 orienter_sort_vers_joueur, 404
 PATH_SPELL_BOSS, 400
 PATH_SPELL_WITCHER, 401
 SP_BOSS_BALL, 401
 SP_BOSS_GROUP, 401
 SP_BOSS_SPLIT, 401
 SP_WITCHER, 401
 type_sort_t, 401
sous_rendu
 init_close.c, 207
 SP_BOSS_BALL
 sorts.h, 401
 SP_BOSS_GROUP
 sorts.h, 401
 SP_BOSS_SPLIT
 sorts.h, 401
 SP_WITCHER
 sorts.h, 401
statut
 joueur_t, 20
statut_s, 31
 action, 32
 animation, 32
 bouclier_equipe, 32
 duree, 32
 duree_anim, 32
 en_mouvement, 32
 orient_att, 32
 orient_dep, 33
 texture_prec, 33
 vrai_zone_collision, 33
 zone_colision, 33
stoper_mouvement_joueurs
 personnage.c, 373
 personnage.h, 390
struct, 33
succ

element, 16
SUD_1
 definition_commun.h, 181
SUD_2
 definition_commun.h, 182
SUD_EST_2
 definition_commun.h, 182
SUD_OUEST_2
 definition_commun.h, 182
suivant
 listes.c, 262
 listes.h, 282
t_aff, 34
 aff_fenetre, 34
 compteur_frame_anim, 35
 duree_frame_anim, 35
 frame_anim, 35
 height, 35
 multipli_taille, 35
 texture, 35
 width, 35
t_direction_1
 definition_commun.h, 181
t_direction_2
 definition_commun.h, 181
t_item
 objet.h, 355
t_l_aff, 36
 liste, 36
 nb_valeurs, 36
t_map, 37
 cases_x, 38
 cases_y, 38
 height, 38
 id_map, 38
 liste_coffres, 38
 liste_collisions, 38
 liste_monstres, 38
 liste_sorts, 38
 liste_zone_tp, 39
 taille_case, 39
 text_map, 39
 text_sol, 39
 texture_superposition, 39
 width, 39
t_texture_perso
 affichage.h, 101
tab
 liste_base_coffres_s, 23
 liste_base_monstres_t, 24
TAILLE_CASE
 map.h, 299
taille_case
 t_map, 39
taille_ecran_cases
 map.c, 295
taille_liste
 listes.c, 263

listes.h, 283
TAILLE_PERSONNAGE
 personnage.h, 377
TAILLE_TRIGGER
 personnage.h, 378
test_affichage.c, 406
 compteur, 407
 FENETRE_LARGEUR, 407
 FENETRE_LONGUEUR, 407
 main, 407
 test_map, 407
test_inventaire.c, 407
 clean_suite, 408
 compteur, 411
 equipe, 411
 FENETRE_LARGEUR, 411
 FENETRE_LONGUEUR, 411
 generation_inventaire, 408
 init_suite, 409
 main, 409
 modification_inventaire, 410
 objet_test, 412
 perso_principal, 412
 sac, 412
 test_map, 412
test_liste_objet.c, 412
 compteur, 413
 FENETRE_LARGEUR, 413
 FENETRE_LONGUEUR, 414
 main, 413
 test_map, 414
test_listes.c, 414
 afficher_int, 415
 compteur, 415
 FENETRE_LARGEUR, 415
 FENETRE_LONGUEUR, 415
 main, 415
 test_map, 416
test_map
 test_affichage.c, 407
 test_inventaire.c, 412
 test_liste_objet.c, 414
 test_listes.c, 416
 test_map.c, 418
 test_personnage.c, 423
test_map.c, 416
 compteur, 417
 FENETRE_LARGEUR, 417
 FENETRE_LONGUEUR, 417
 main, 417
 test_map, 418
test_monstres.c, 418
 main, 419
test_personnage.c, 419
 clean_suite, 420
 compteur, 423
 creation_personnage, 420
 FENETRE_LARGEUR, 423
 FENETRE_LONGUEUR, 423
 init_suite, 421
 main, 421
 modification_personnage, 422
 perso_principal, 423
 test_map, 423
test SDL.c, 424
 clean_suite, 425
 FENETRE_LARGEUR, 427
 FENETRE_LONGUEUR, 427
 fenetre_Principale, 427
 init_suite, 425
 main, 425
 rendu_principal, 427
 test SDL_close, 426
 test SDL_init, 426
test SDL_close
 test SDL.c, 426
test SDL_init
 test SDL.c, 426
text_accueil
 menus.c, 315
 menus.h, 323
TEXT_ATTAQUE
 affichage.h, 101
TEXT_ATTAQUE_CHARGEE
 affichage.h, 101
TEXT_CHARGER
 affichage.h, 101
text_copier_position
 affichage.c, 93
 affichage.h, 149
text_inventaire1
 menus.c, 315
 menus.h, 323
text_inventaire2
 menus.c, 315
 menus.h, 324
text_map
 t_map, 39
TEXT_MARCHER
 affichage.h, 101
TEXT_MARCHER_BOUCLIER
 affichage.h, 101
text_pause
 menus.c, 315
 menus.h, 324
text_sol
 t_map, 39
texture
 boss_s, 13
 coffre_s, 15
 monstre_s, 27
 objet_t, 29
 sort_s, 31
 t_aff, 35
texture_map
 map.c, 295

map.h, 303
 texture_prec
 statut_s, 33
 texture_src
 objet_t, 29
 texture_superposition
 t_map, 39
 texture_temp
 boss_s, 13
 textures_joueur
 joueur_t, 20
 TOUCHE_BAS
 event.h, 196
 TOUCHE_CONSUMMABLE
 event.h, 196
 TOUCHE_DROITE
 event.h, 196
 TOUCHE_ECHAP
 event.h, 196
 TOUCHE_GAUCHE
 event.h, 196
 TOUCHE_HAUT
 event.h, 196
 TOUCHE_TAB
 event.h, 196
 tout_ramasser
 inventaire.c, 220
 inventaire.h, 229
 tp_joueurs
 map.c, 296
 map.h, 304
 transition
 map.c, 296
 map.h, 304
 trigger
 joueur_t, 20
 tx
 affichage.c, 95
 affichage.h, 151
 ty
 affichage.c, 95
 affichage.h, 151
 type
 base_sort_s, 12
 boss_s, 14
 coffre_s, 15
 monstre_s, 27
 objet_t, 29
 sort_s, 31
 type_boss_1_t
 boss.h, 158
 type_coffre_t
 coffres.h, 171
 TYPE_MONSTRE_INCONNU
 monstres.h, 345
 type_monstre_t
 monstres.h, 345
 type_sort_t
 sorts.h, 401
 types_erreur
 code_erreur.h, 160
 update_frame_texture
 affichage.c, 93
 valeur
 element, 16
 valeur_elt
 listes.c, 264
 listes.h, 284
 vider_liste
 listes.c, 265
 listes.h, 285
 vitesse
 base_monstre_s, 11
 joueur_t, 20
 monstre_s, 27
 objet_t, 29
 vitesse_actif
 joueur_t, 20
 vrai
 definition_commun.h, 181
 vrai_zone_collision
 statut_s, 33
 warning
 code_erreur.h, 159
 width
 t_aff, 35
 t_map, 39
 WITCHER
 monstres.h, 345
 x
 point, 30
 xp
 boss_s, 14
 joueur_t, 21
 y
 point, 30
 zone
 zone_tp, 40
 zone_colision
 statut_s, 33
 zone_en_dehors_hitbox
 personnage.c, 374
 personnage.h, 391
 zone_tp, 40
 dest, 40
 id_map, 40
 zone, 40