

Bloody Sanada

Généré par Doxygen 1.8.17

1 Index des classes	1
1.1 Liste des classes	1
2 Index des fichiers	3
2.1 Liste des fichiers	3
3 Documentation des classes	5
3.1 Référence de la structure base_monstre_s	5
3.1.1 Documentation des données membres	5
3.1.1.1 attaque	5
3.1.1.2 fichier_image	5
3.1.1.3 gainXp	5
3.1.1.4 nom_monstre	6
3.1.1.5 pdv	6
3.1.1.6 vitesse	6
3.2 Référence de la structure element	6
3.2.1 Documentation des données membres	6
3.2.1.1 pred	6
3.2.1.2 succ	6
3.2.1.3 valeur	7
3.3 Référence de la structure inventaire_s	7
3.3.1 Documentation des données membres	7
3.3.1.1 equipe	7
3.3.1.2 sac	7
3.4 Référence de la structure joueur_s	7
3.4.1 Documentation des données membres	8
3.4.1.1 attaque	8
3.4.1.2 defense	8
3.4.1.3 maxPdv	8
3.4.1.4 niveau	8
3.4.1.5 nom_pers	8
3.4.1.6 pdv	9
3.4.1.7 statut	9
3.4.1.8 textures_joueur	9
3.4.1.9 trigger	9
3.4.1.10 vitesse	9
3.4.1.11 xp	9
3.5 Référence de la structure list	9
3.5.1 Documentation des données membres	10
3.5.1.1 aff	10
3.5.1.2 ajout	10
3.5.1.3 del	10
3.5.1.4 ec	10

3.5.1.5 flag	10
3.5.1.6 nb_elem	10
3.6 Référence de la structure liste_base_monstres_s	11
3.6.1 Documentation des données membres	11
3.6.1.1 nb_monstre	11
3.6.1.2 tab	11
3.7 Référence de la structure lobjet_s	11
3.7.1 Documentation des données membres	11
3.7.1.1 liste	12
3.7.1.2 nb	12
3.8 Référence de la structure monstre_s	12
3.8.1 Documentation des données membres	12
3.8.1.1 attaque	12
3.8.1.2 gainXp	12
3.8.1.3 nom_monstre	13
3.8.1.4 orientation	13
3.8.1.5 pdv	13
3.8.1.6 position	13
3.8.1.7 texture	13
3.8.1.8 vitesse	13
3.9 Référence de la structure objet_s	13
3.9.1 Documentation des données membres	14
3.9.1.1 attaque	14
3.9.1.2 defense	14
3.9.1.3 niveau	14
3.9.1.4 nom	14
3.9.1.5 texture	14
3.9.1.6 texture_src	15
3.9.1.7 type	15
3.9.1.8 vitesse	15
3.10 Référence de la structure point	15
3.10.1 Documentation des données membres	15
3.10.1.1 x	15
3.10.1.2 y	15
3.11 Référence de la structure s_aff	16
3.11.1 Description détaillée	16
3.11.2 Documentation des données membres	16
3.11.2.1 aff_fenetre	16
3.11.2.2 duree_frame_anim	16
3.11.2.3 frame_anim	16
3.11.2.4 height	16
3.11.2.5 multipli_taille	17

3.11.2.6 texture	17
3.11.2.7 width	17
3.12 Référence de la structure s_l_aff	17
3.12.1 Description détaillée	17
3.12.2 Documentation des données membres	17
3.12.2.1 liste	17
3.12.2.2 nb_valeurs	18
3.13 Référence de la structure statut_s	18
3.13.1 Documentation des données membres	18
3.13.1.1 action	18
3.13.1.2 bouclier_equipe	18
3.13.1.3 duree	18
3.13.1.4 en_mouvement	18
3.13.1.5 orientation	19
3.13.1.6 zone_colision	19
3.14 Référence de la structure struct	19
3.14.1 Description détaillée	19
3.15 Référence de la structure t_map	20
3.15.1 Description détaillée	20
3.15.2 Documentation des données membres	20
3.15.2.1 height	20
3.15.2.2 liste_monstres	20
3.15.2.3 text_map	20
3.15.2.4 unite_dep_x	21
3.15.2.5 unite_dep_y	21
3.15.2.6 width	21
3.16 Référence de la structure zone_tp	21
3.16.1 Description détaillée	21
3.16.2 Documentation des données membres	21
3.16.2.1 dest	22
3.16.2.2 id_map	22
3.16.2.3 p1	22
3.16.2.4 p2	22
4 Documentation des fichiers	23
4.1 Référence du fichier affichage.c	23
4.2 Référence du fichier affichage.h	23
4.2.1 Description détaillée	25
4.2.2 Documentation des macros	25
4.2.2.1 LARGEUR_PERSONNAGE	25
4.2.2.2 LONGUEUR_PERSONNAGE	25
4.2.2.3 N_T_ATAQUE	25

4.2.2.4 N_T_ATTAQUE_CHARGEE	25
4.2.2.5 N_T_CHARGER	26
4.2.2.6 N_T_MARCHER	26
4.2.2.7 N_T_MARCHER_BOUCLIER	26
4.2.2.8 NB_FPS	26
4.2.2.9 NB_SPRITE_JOUEUR	26
4.2.3 Documentation des définitions de type	26
4.2.3.1 t_aff	26
4.2.3.2 t_l_aff	26
4.2.4 Documentation du type de l'énumération	26
4.2.4.1 t_texture_perso	26
4.2.5 Documentation des fonctions	27
4.2.5.1 afficher_buffer()	27
4.2.5.2 afficher_texture()	27
4.2.5.3 ajout_text_liste()	28
4.2.5.4 color()	28
4.2.5.5 creer_texture()	28
4.2.5.6 def_texture_taille()	29
4.2.5.7 deplacement_x_pers()	29
4.2.5.8 deplacement_y_pers()	29
4.2.5.9 deplacer_rect_haut_droit()	30
4.2.5.10 deplacer_rect_origine()	30
4.2.5.11 deplacer_texture_bas_droit()	30
4.2.5.12 deplacer_texture_bas_gauche()	30
4.2.5.13 deplacer_texture_centre()	31
4.2.5.14 deplacer_texture_haut_droit()	31
4.2.5.15 deplacer_texture_origine()	31
4.2.5.16 detruire_liste_textures()	32
4.2.5.17 detruire_texture()	32
4.2.5.18 info_texture()	32
4.2.5.19 init_texture_joueur()	33
4.2.5.20 init_textures_joueur()	33
4.2.5.21 modif_affichage_rect()	33
4.2.5.22 next_frame_joueur()	33
4.2.5.23 next_frame_x()	34
4.2.5.24 next_frame_x_indice()	34
4.2.5.25 next_frame_y()	34
4.2.5.26 next_frame_y_indice()	34
4.2.5.27 rect_centre()	35
4.2.5.28 rect_centre_x()	35
4.2.5.29 rect_centre_y()	35
4.2.5.30 rect_correct_texture()	35

4.2.5.31	rects_egal_x()	36
4.2.5.32	rects_egal_y()	36
4.2.5.33	text_copier_position()	36
4.2.6	Documentation des variables	36
4.2.6.1	buffer_affichage	36
4.2.6.2	compteur	37
4.2.6.3	listeDeTextures	37
4.2.6.4	multiplicateur_x	37
4.2.6.5	multiplicateur_y	37
4.2.6.6	tx	37
4.2.6.7	ty	37
4.3	Référence du fichier code_erreur.h	37
4.3.1	Description détaillée	38
4.3.2	Documentation du type de l'énumération	38
4.3.2.1	types_erreur	38
4.4	Référence du fichier commun.h	39
4.5	Référence du fichier definition_commun.h	40
4.5.1	Description détaillée	41
4.5.2	Documentation des macros	41
4.5.2.1	bool	41
4.5.2.2	faux	41
4.5.2.3	SAVE_PATH	41
4.5.2.4	vrai	41
4.5.3	Documentation des définitions de type	41
4.5.3.1	byte	42
4.5.3.2	err_t	42
4.5.3.3	point	42
4.5.4	Documentation du type de l'énumération	42
4.5.4.1	t_direction	42
4.5.5	Documentation des fonctions	42
4.5.5.1	fermer_programme()	42
4.5.6	Documentation des variables	43
4.5.6.1	FENETRE_LARGEUR	43
4.5.6.2	FENETRE_LONGUEUR	43
4.5.6.3	fenetre_Principale	43
4.5.6.4	rendu_principal	43
4.5.6.5	running	43
4.6	Référence du fichier definition_ressources.h	43
4.6.1	Documentation des macros	43
4.6.1.1	PLAYER_HEIGHT	44
4.6.1.2	PLAYER_WITH	44
4.7	Référence du fichier event.c	44

4.7.1 Description détaillée	44
4.7.2 Documentation des fonctions	45
4.7.2.1 jeu_event()	45
4.7.2.2 keyDown()	45
4.7.2.3 keyUp()	45
4.7.2.4 logo_passer()	46
4.7.2.5 mouseButtonDown()	46
4.7.2.6 mouseButtonUp()	46
4.8 Référence du fichier event.h	46
4.8.1 Description détaillée	47
4.8.2 Documentation des macros	47
4.8.2.1 TOUCHE_BAS	47
4.8.2.2 TOUCHE_DROITE	47
4.8.2.3 TOUCHE_ECHAP	48
4.8.2.4 TOUCHE_GAUCHE	48
4.8.2.5 TOUCHE_HAUT	48
4.8.2.6 TOUCHE_TAB	48
4.8.3 Documentation des fonctions	48
4.8.3.1 jeu_event()	48
4.8.3.2 logo_passer()	48
4.9 Référence du fichier fonctions.h	49
4.9.1 Description détaillée	49
4.9.2 Documentation des fonctions	49
4.9.2.1 init()	49
4.9.2.2 init_event()	50
4.9.2.3 terminate_init()	50
4.10 Référence du fichier init_close.c	50
4.10.1 Documentation des fonctions	50
4.10.1.1 aff_cleanup()	51
4.10.1.2 detruire_renderer()	51
4.10.1.3 fermer_programme()	51
4.10.1.4 fermer_SDL()	51
4.10.1.5 init()	51
4.10.1.6 init_affichage()	52
4.10.1.7 init_rc_commun()	52
4.10.1.8 init_SDL()	52
4.10.1.9 init_sousbuffer()	52
4.10.2 Documentation des variables	52
4.10.2.1 f_close	52
4.10.2.2 fenetre_Principale	52
4.10.2.3 fenetre_sous_rendu	53
4.10.2.4 rendu_principal	53

4.10.2.5 running	53
4.10.2.6 sous_rendu	53
4.11 Référence du fichier interface.c	53
4.11.1 Documentation des fonctions	53
4.11.1.1 RenderHPBar()	53
4.12 Référence du fichier interface.h	54
4.12.1 Description détaillée	54
4.12.2 Documentation des fonctions	54
4.12.2.1 RenderHPBar()	54
4.13 Référence du fichier inventaire.c	55
4.13.1 Documentation des fonctions	55
4.13.1.1 changement_statistiques()	55
4.13.1.2 creer_inventaire()	56
4.13.1.3 desequiper()	56
4.13.1.4 detruire_inventaire()	56
4.13.1.5 equiper_objet()	56
4.13.1.6 ramasser_objet()	57
4.14 Référence du fichier inventaire.h	57
4.14.1 Description détaillée	58
4.14.2 Documentation des définitions de type	58
4.14.2.1 inventaire_t	58
4.14.3 Documentation des fonctions	58
4.14.3.1 changement_statistiques()	58
4.14.3.2 creer_inventaire()	58
4.14.3.3 desequiper()	59
4.14.3.4 detruire_inventaire()	59
4.14.3.5 equiper_objet()	59
4.14.3.6 ramasser_objet()	60
4.15 Référence du fichier liste_objet.c	60
4.15.1 Documentation des fonctions	60
4.15.1.1 afficher_liste_objet()	60
4.15.1.2 creer_liste_objet()	60
4.15.1.3 creer_liste_objet_equipe()	61
4.15.1.4 creer_liste_objet_vide()	61
4.15.1.5 detruire_liste_objet()	61
4.15.1.6 effacer_liste_objet()	61
4.16 Référence du fichier liste_objet.h	61
4.16.1 Description détaillée	62
4.16.2 Documentation des définitions de type	62
4.16.2.1 lobjet_t	62
4.16.3 Documentation des fonctions	62
4.16.3.1 afficher_liste_objet()	62

4.16.3.2	creer_liste_objet()	62
4.16.3.3	creer_liste_objet_equipe()	63
4.16.3.4	creer_liste_objet_vide()	63
4.16.3.5	destruire_liste_objet()	63
4.16.3.6	effacer_liste_objet()	63
4.17	Référence du fichier listes.c	63
4.17.1	Documentation des définitions de type	64
4.17.1.1	list	64
4.17.1.2	t_element	64
4.17.2	Documentation des fonctions	64
4.17.2.1	afficher_liste()	64
4.17.2.2	ajout_droit()	65
4.17.2.3	ajout_gauche()	65
4.17.2.4	destruire_liste()	65
4.17.2.5	en_queue()	65
4.17.2.6	en_tete()	66
4.17.2.7	hors_liste()	66
4.17.2.8	init_liste()	66
4.17.2.9	liste_vide()	67
4.17.2.10	modif_elt()	67
4.17.2.11	oter_elt()	68
4.17.2.12	precedent()	68
4.17.2.13	suivant()	68
4.17.2.14	taille_liste()	69
4.17.2.15	valeur_elt()	69
4.17.2.16	vider_liste()	69
4.18	Référence du fichier listes.h	70
4.18.1	Documentation des définitions de type	70
4.18.1.1	list	70
4.18.2	Documentation des fonctions	70
4.18.2.1	afficher_liste()	70
4.18.2.2	ajout_droit()	71
4.18.2.3	ajout_gauche()	71
4.18.2.4	destruire_liste()	71
4.18.2.5	en_queue()	72
4.18.2.6	en_tete()	72
4.18.2.7	hors_liste()	72
4.18.2.8	init_liste()	72
4.18.2.9	liste_vide()	73
4.18.2.10	modif_elt()	73
4.18.2.11	oter_elt()	74
4.18.2.12	precedent()	74

4.18.2.13	suivant()	74
4.18.2.14	taille_liste()	75
4.18.2.15	valeur_elt()	75
4.18.2.16	vider_liste()	75
4.19	Référence du fichier main.c	76
4.19.1	Documentation des macros	76
4.19.1.1	SDL_MAIN_HANDLED	76
4.19.2	Documentation des fonctions	76
4.19.2.1	afficher_intro()	76
4.19.2.2	main()	77
4.20	Référence du fichier map.c	77
4.20.1	Documentation des fonctions	77
4.20.1.1	charger_f_map()	77
4.20.1.2	charger_s_map()	78
4.20.1.3	taille_ecran_cases()	78
4.20.1.4	texture_map()	78
4.20.2	Documentation des variables	78
4.20.2.1	map	79
4.21	Référence du fichier map.h	79
4.21.1	Description détaillée	80
4.21.2	Documentation des macros	80
4.21.2.1	TAILLE_CASE	80
4.21.3	Documentation des définitions de type	80
4.21.3.1	t_aff	80
4.21.4	Documentation des fonctions	80
4.21.4.1	charger_f_map()	80
4.21.4.2	charger_s_map()	81
4.21.4.3	texture_map()	81
4.21.5	Documentation des variables	81
4.21.5.1	map	82
4.22	Référence du fichier menus.c	82
4.22.1	Documentation des fonctions	82
4.22.1.1	afficher_inventaire()	82
4.22.1.2	afficher_menu_pause()	82
4.23	Référence du fichier menus.h	82
4.23.1	Documentation du type de l'énumération	83
4.23.1.1	menus_t	83
4.23.2	Documentation des fonctions	83
4.23.2.1	afficher_inventaire()	83
4.23.2.2	afficher_menu_pause()	83
4.23.3	Documentation des variables	83
4.23.3.1	menus	84

4.24	Référence du fichier monstres.c	84
4.24.1	Documentation des fonctions	84
4.24.1.1	charger_monstres()	84
4.24.1.2	destruire_liste_base_monstres()	84
4.24.1.3	destruire_monstre()	85
4.25	Référence du fichier monstres.h	85
4.25.1	Documentation des définitions de type	85
4.25.1.1	base_monstre_t	85
4.25.1.2	liste_base_monstres_t	85
4.25.1.3	monstre_t	86
4.25.1.4	position_t	86
4.25.2	Documentation des fonctions	86
4.25.2.1	charger_monstres()	86
4.25.2.2	destruire_liste_base_monstres()	86
4.25.2.3	destruire_monstre()	86
4.26	Référence du fichier objet.c	87
4.26.1	Description détaillée	87
4.26.2	Documentation des fonctions	87
4.26.2.1	afficher_objet()	87
4.26.2.2	creer_objet()	88
4.26.2.3	destruire_objet()	88
4.27	Référence du fichier objet.h	88
4.27.1	Description détaillée	89
4.27.2	Documentation des définitions de type	89
4.27.2.1	objet_t	89
4.27.3	Documentation du type de l'énumération	89
4.27.3.1	t_item	89
4.27.4	Documentation des fonctions	90
4.27.4.1	afficher_objet()	90
4.27.4.2	creer_objet()	90
4.27.4.3	destruire_objet()	90
4.27.4.4	destruire_texture()	90
4.27.4.5	init_objet()	91
4.28	Référence du fichier personnage.c	91
4.28.1	Description détaillée	91
4.28.2	Documentation des fonctions	92
4.28.2.1	afficher_statistiques()	92
4.28.2.2	caracteristiques()	92
4.28.2.3	charger_sauvegarde_joueur()	92
4.28.2.4	check_repertoire_jeux()	92
4.28.2.5	copy()	92
4.28.2.6	creer_joueur()	93

4.28.2.7 creer_sauvegarde_json()	93
4.28.2.8 detruire_joueur()	93
4.28.2.9 gain_xp()	93
4.28.2.10 levelup()	93
4.28.2.11 new_joueur()	93
4.28.2.12 sauv_existe()	94
4.28.3 Documentation des variables	94
4.28.3.1 perso_principal	94
4.28.3.2 save_path	94
4.29 Référence du fichier personnage.h	94
4.29.1 Description détaillée	95
4.29.2 Documentation des macros	95
4.29.2.1 DUREE_ATTAQUE	95
4.29.2.2 DUREE_ATTAQUE_CHARGEE	96
4.29.2.3 DUREE_ATTAQUE_OU_CHARGEE	96
4.29.2.4 DUREE_BLOQUER	96
4.29.2.5 TAILLE_PERSONNAGE	96
4.29.2.6 TAILLE_TRIGGER	96
4.29.3 Documentation des définitions de type	96
4.29.3.1 byte	96
4.29.3.2 joueur_t	96
4.29.3.3 statut_t	97
4.29.3.4 t_l_aff	97
4.29.4 Documentation du type de l'énumération	97
4.29.4.1 action_t	97
4.29.5 Documentation des fonctions	97
4.29.5.1 afficher_statistiques()	97
4.29.5.2 caracteristiques()	97
4.29.5.3 charger_sauvegarde_joueur()	98
4.29.5.4 check_repertoire_jeux()	98
4.29.5.5 creer_joueur()	98
4.29.5.6 creer_sauvegarde_json()	98
4.29.5.7 detruire_joueur()	98
4.29.5.8 gain_xp()	98
4.29.5.9 levelup()	99
4.29.5.10 new_joueur()	99
4.29.6 Documentation des variables	99
4.29.6.1 perso_principal	99
4.29.6.2 save_path	99
4.30 Référence du fichier test_affichage.c	99
4.30.1 Documentation des fonctions	99
4.30.1.1 main()	100

4.30.2 Documentation des variables	100
4.30.2.1 compteur	100
4.30.2.2 FENETRE_LARGEUR	100
4.30.2.3 FENETRE_LONGUEUR	100
4.30.2.4 test_map	100
4.31 Référence du fichier test_inventaire.c	100
4.31.1 Documentation des fonctions	101
4.31.1.1 main()	101
4.31.2 Documentation des variables	101
4.31.2.1 compteur	101
4.31.2.2 FENETRE_LARGEUR	101
4.31.2.3 FENETRE_LONGUEUR	101
4.31.2.4 test_map	101
4.32 Référence du fichier test_liste_objet.c	101
4.32.1 Documentation des fonctions	102
4.32.1.1 main()	102
4.32.2 Documentation des variables	102
4.32.2.1 compteur	102
4.32.2.2 FENETRE_LARGEUR	102
4.32.2.3 FENETRE_LONGUEUR	102
4.32.2.4 fenetre_Principale	103
4.32.2.5 rendu_principal	103
4.32.2.6 running	103
4.32.2.7 test_map	103
4.33 Référence du fichier test_listes.c	103
4.33.1 Documentation des fonctions	103
4.33.1.1 afficher_int()	104
4.33.1.2 main()	104
4.33.2 Documentation des variables	104
4.33.2.1 compteur	104
4.33.2.2 FENETRE_LARGEUR	104
4.33.2.3 FENETRE_LONGUEUR	104
4.33.2.4 test_map	104
4.34 Référence du fichier test_map.c	104
4.34.1 Documentation des fonctions	105
4.34.1.1 main()	105
4.34.2 Documentation des variables	105
4.34.2.1 compteur	105
4.34.2.2 FENETRE_LARGEUR	105
4.34.2.3 FENETRE_LONGUEUR	105
4.34.2.4 test_map	105
4.35 Référence du fichier test_monstres.c	106

4.35.1 Documentation des fonctions	106
4.35.1.1 main()	106
4.36 Référence du fichier test_personnage.c	106
4.36.1 Documentation des fonctions	106
4.36.1.1 main()	106
4.36.2 Documentation des variables	107
4.36.2.1 compteur	107
4.36.2.2 FENETRE_LARGEUR	107
4.36.2.3 FENETRE_LONGUEUR	107
4.36.2.4 test_map	107
Index	109

Chapitre 1

Index des classes

1.1 Liste des classes

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

base_monstre_s	5
element	6
inventaire_s	7
joueur_s	7
list	9
liste_base_monstres_s	11
lobjet_s	11
monstre_s	12
objet_s	13
point	15
s_aff	
Structure qui permet l'affichage d'une texture à l'écran de manière précise	16
s_l_aff	
Structure contenant la liste des textures créées par le programme	17
statut_s	18
struct	
Structure de liste d'objets	19
t_map	
Structure représentant une map	20
zone_tp	
Structure représentant une zone de tp	21

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

affichage.c	
Fichier contenant toutes les fonctions relatives à l'affichage	23
affichage.h	23
code_erreur.h	
Fichier contenant les codes d'erreur du programme	37
commun.h	39
definition_commun.h	
Contient toutes les définitions communes à tout les fichiers	40
definition_ressources.h	43
event.c	
Fichier qui appelle différentes fonctions en fonction du déclenchement d'événements	44
event.h	46
fonctions.h	
Fichier qui contient les définitions de toutes les fonctions	49
init_close.c	50
interface.c	53
interface.h	54
inventaire.c	55
inventaire.h	57
liste_objet.c	60
liste_objet.h	
Fichier contenant toutes les définitions concernant les listes d'objets	61
listes.c	63
listes.h	70
main.c	76
map.c	77
map.h	
Le fichier contient les définitions des fonctions de gestion de la map	79
menus.c	82
menus.h	82
monstres.c	84
monstres.h	85
objet.c	
Fichier contenant toutes les fonctions concernant les objets	87
objet.h	
Fichier contenant toutes les définitions concernant les objets	88

personnage.c	
Fichier contenant toutes les fonctions concernant le personnage	91
personnage.h	
Fichier contenant toutes les définitions concernant le personnage	94
test_affichage.c	99
test_inventaire.c	100
test_liste_objet.c	101
test_listes.c	103
test_map.c	104
test_monstres.c	106
test_personnage.c	106

Chapitre 3

Documentation des classes

3.1 Référence de la structure base_monstre_s

```
#include <monstres.h>
```

Attributs publics

- char fichier_image [20]
- char nom_monstre [20]
- int pdv
- int attaque
- float vitesse
- int gainXp

3.1.1 Documentation des données membres

3.1.1.1 attaque

```
int base_monstre_s::attaque
```

attaque

3.1.1.2 fichier_image

```
char base_monstre_s::fichier_image[20]
```

nom fichier image

3.1.1.3 gainXp

```
int base_monstre_s::gainXp
```

gain d'xp pour le joueur

3.1.1.4 nom_monstre

```
char base_monstre_s::nom_monstre[20]
```

nom du monstre

3.1.1.5 pdv

```
int base_monstre_s::pdv
```

points de vie

3.1.1.6 vitesse

```
float base_monstre_s::vitesse
```

vitesse de déplacement

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

— [monstres.h](#)

3.2 Référence de la structure element

Graphe de collaboration de element:

Attributs publics

- void * [valeur](#)
- [struct element](#) * [pred](#)
- [struct element](#) * [succ](#)

3.2.1 Documentation des données membres

3.2.1.1 pred

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

3.2.1.2 succ

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

3.2.1.3 valeur

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

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

— [listes.c](#)

3.3 Référence de la structure inventaire_s

```
#include <inventaire.h>
```

Graphe de collaboration de inventaire_s:

Attributs publics

— [lobjet_t](#) * [equipe](#)
— [lobjet_t](#) * [sac](#)

3.3.1 Documentation des données membres

3.3.1.1 equipe

```
lobjet\_t* inventaire_s::equipe
```

3.3.1.2 sac

```
lobjet\_t* inventaire_s::sac
```

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

— [inventaire.h](#)

3.4 Référence de la structure joueur_s

```
#include <personnage.h>
```

Graphe de collaboration de joueur_s:

Attributs publics

- char * [nom_pers](#)
- short int [niveau](#)
- int [xp](#)
- byte * [trigger](#)
- int [maxPdv](#)
- int [pdv](#)
- int [attaque](#)
- int [defense](#)
- int [vitesse](#)
- [statut_t](#) * [statut](#)
- [t_l_aff](#) * [textures_joueur](#)

3.4.1 Documentation des données membres

3.4.1.1 attaque

```
int joueur_s::attaque
```

attaque du joueur

3.4.1.2 defense

```
int joueur_s::defense
```

defense du joueur

3.4.1.3 maxPdv

```
int joueur_s::maxPdv
```

TODO : créer un type énuméré map

3.4.1.4 niveau

```
short int joueur_s::niveau
```

Le niveau du joueur

3.4.1.5 nom_pers

```
char* joueur_s::nom_pers
```

Le nom du personnage

3.4.1.6 pdv

```
int joueur_s::pdv
```

3.4.1.7 statut

```
statut_t* joueur_s::statut
```

statut du joueur

3.4.1.8 textures_joueur

```
t_l_aff* joueur_s::textures_joueur
```

Tableau contenant toutes les textures du joueur

3.4.1.9 trigger

```
byte* joueur_s::trigger
```

Une variable contenant des triggers logiques concernant le personnage

3.4.1.10 vitesse

```
int joueur_s::vitesse
```

vitesse de déplacement du joueur

3.4.1.11 xp

```
int joueur_s::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](#)

3.5 Référence de la structure list

Graphe de collaboration de list:

Attributs publics

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

3.5.1 Documentation des données membres

3.5.1.1 aff

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

3.5.1.2 ajout

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

3.5.1.3 del

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

3.5.1.4 ec

```
t\_element* list::ec
```

3.5.1.5 flag

```
t\_element* list::flag
```

3.5.1.6 nb_elem

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

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

- [listes.c](#)

3.6 Référence de la structure liste_base_monstres_s

```
#include <monstres.h>
```

Graphe de collaboration de liste_base_monstres_s:

Attributs publics

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

3.6.1 Documentation des données membres

3.6.1.1 nb_monstre

```
int liste_base_monstres_s::nb_monstre
```

3.6.1.2 tab

```
base\_monstre\_t** liste_base_monstres_s::tab
```

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

- [monstres.h](#)

3.7 Référence de la structure lobjet_s

```
#include <liste_objet.h>
```

Graphe de collaboration de lobjet_s:

Attributs publics

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

3.7.1 Documentation des données membres

3.7.1.1 liste

```
objet_t** lobjet_s::liste
```

3.7.1.2 nb

```
int lobjet_s::nb
```

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

— [liste_objet.h](#)

3.8 Référence de la structure monstre_s

```
#include <monstres.h>
```

Graphe de collaboration de monstre_s:

Attributs publics

- char [nom_monstre](#) [20]
- int [pdv](#)
- int [attaque](#)
- float [vitesse](#)
- [position_t](#) [position](#)
- int [gainXp](#)
- [t_direction](#) [orientation](#)
- [t_aff](#) * [texture](#)

3.8.1 Documentation des données membres

3.8.1.1 attaque

```
int monstre_s::attaque
```

attaque

3.8.1.2 gainXp

```
int monstre_s::gainXp
```

gain d'xp pour le joueur

3.8.1.3 nom_monstre

```
char monstre_s::nom_monstre[20]
```

nom du monstre

3.8.1.4 orientation

```
t_direction monstre_s::orientation
```

orientation

3.8.1.5 pdv

```
int monstre_s::pdv
```

points de vie

3.8.1.6 position

```
position_t monstre_s::position
```

coordonnées

3.8.1.7 texture

```
t_aff* monstre_s::texture
```

texture

3.8.1.8 vitesse

```
float monstre_s::vitesse
```

vitesse de déplacement

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

— [monstres.h](#)

3.9 Référence de la structure objet_s

```
#include <objet.h>
```

Graphe de collaboration de objet_s:

Attributs publics

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

3.9.1 Documentation des données membres

3.9.1.1 attaque

```
int objet_s::attaque
```

modificateur d'attaque de l'objet

3.9.1.2 defense

```
int objet_s::defense
```

modificateur de defense de l'objet

3.9.1.3 niveau

```
short int objet_s::niveau
```

Le niveau necessaire pour équiper l'objet

3.9.1.4 nom

```
char* objet_s::nom
```

Le nom de l'objet

3.9.1.5 texture

```
t_aff* objet_s::texture
```

Image de l'objet

3.9.1.6 texture_src

```
char* objet_s::texture_src
```

3.9.1.7 type

```
t_item objet_s::type
```

Le type d'objet permet de contrôler sa bonne utilisation

3.9.1.8 vitesse

```
int objet_s::vitesse
```

modificateur de vitesse de l'objet

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

— [objet.h](#)

3.10 Référence de la structure point

```
#include <definition_commun.h>
```

Attributs publics

— int [x](#)
— int [y](#)

3.10.1 Documentation des données membres

3.10.1.1 x

```
int point::x
```

3.10.1.2 y

```
int point::y
```

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

— [definition_commun.h](#)

3.11 Référence de la structure s_aff

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

```
#include <affichage.h>
```

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](#)

3.11.1 Description détaillée

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

3.11.2 Documentation des données membres

3.11.2.1 aff_fenetre

```
SDL_Rect* s_aff::aff_fenetre
```

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

3.11.2.2 duree_frame_anim

```
unsigned int s_aff::duree_frame_anim
```

3.11.2.3 frame_anim

```
SDL_Rect* s_aff::frame_anim
```

Désigne la zone de la texture à afficher

3.11.2.4 height

```
int s_aff::height
```


3.11.2.5 `multipli_taille`

```
float s_aff::multipli_taille
```

Sauvegarde du multiplicateur de taille de la texture

3.11.2.6 `texture`

```
SDL_Texture* s_aff::texture
```

Texture utilisée

3.11.2.7 `width`

```
int s_aff::width
```

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

— [affichage.h](#)

3.12 Référence de la structure `s_l_aff`

Structure contenant la liste des textures créées par le programme.

```
#include <affichage.h>
```

Graphe de collaboration de `s_l_aff`:

Attributs publics

- [t_aff](#) ** [liste](#)
- unsigned int [nb_valeurs](#)

3.12.1 Description détaillée

Structure contenant la liste des textures créées par le programme.

3.12.2 Documentation des données membres

3.12.2.1 `liste`

```
t\_aff** s_l_aff::liste
```

3.12.2.2 nb_valeurs

```
unsigned int s_l_aff::nb_valeurs
```

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

— [affichage.h](#)

3.13 Référence de la structure statut_s

```
#include <personnage.h>
```

Attributs publics

- [bool](#) en_mouvement
- [t_direction](#) orientation
- [bool](#) bouclier_equipe
- [int](#) duree
- [action_t](#) action
- [SDL_Rect](#) zone_colision

3.13.1 Documentation des données membres

3.13.1.1 action

```
action_t statut_s::action
```

l'action du personnage

3.13.1.2 bouclier_equipe

```
bool statut_s::bouclier_equipe
```

personnage à un bouclier d'équipé

3.13.1.3 duree

```
int statut_s::duree
```

duree de l'action à réaliser

3.13.1.4 en_mouvement

```
bool statut_s::en_mouvement
```

personnage en mouvement

3.13.1.5 orientation

`t_direction` statut_s::orientation

orientation du personnage

3.13.1.6 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](#)

3.14 Référence de la structure struct

Structure de liste d'objets.

3.14.1 Description détaillée

Structure de liste d'objets.

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

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

Structure objet.

Structure contenant un tableau avec tous les monstres différent que l'on peut utiliser dans le jeu.

Structure contenant les propriétés du monstre importé

Structure contenant les propriétés du monstre en jeu.

Structure regroupant les coordonnées.

Auteur

Descomps Max
Bruneau Antoine
Despert Ange

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

— [liste_objet.h](#)

3.15 Référence de la structure t_map

Structure représentant une map.

```
#include <map.h>
```

Graphe de collaboration de t_map:

Attributs publics

- [t_aff](#) * [text_map](#)
- unsigned int [width](#)
- unsigned int [height](#)
- [list](#) * [liste_monstres](#)
- int [unite_dep_x](#)
- int [unite_dep_y](#)

3.15.1 Description détaillée

Structure représentant une map.

Auteur

Ange Despert

3.15.2 Documentation des données membres

3.15.2.1 height

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

La largeur et la hauteur de la map

3.15.2.2 liste_monstres

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

3.15.2.3 text_map

```
t\_aff* t_map::text_map
```

La texture de la map

3.15.2.4 unite_dep_x

```
int t_map::unite_dep_x
```

L'unité de déplacement en x

3.15.2.5 unite_dep_y

```
int t_map::unite_dep_y
```

L'unité de déplacement en y

3.15.2.6 width

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

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

— [map.h](#)

3.16 Référence de la structure zone_tp

Structure représentant une zone de tp.

```
#include <map.h>
```

Graphe de collaboration de zone_tp:

Attributs publics

- [point p1](#)
- [point p2](#)
- unsigned int [id_map](#)
- [point dest](#)

3.16.1 Description détaillée

Structure représentant une zone de tp.

Il s'agit ici d'identifier une zone de colision 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

3.16.2 Documentation des données membres

3.16.2.1 dest

`point zone_tp::dest`

Les coordonnées du point d'apparition sur la map

3.16.2.2 id_map

`unsigned int zone_tp::id_map`

l'id de la map de destination

3.16.2.3 p1

`point zone_tp::p1`

3.16.2.4 p2

`point zone_tp::p2`

Rectangle représentant la zone de tp

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

— [map.h](#)

Chapitre 4

Documentation des fichiers

4.1 Référence du fichier affichage.c

Fichier contenant toutes les fonctions relatives à l'affichage.

```
#include <affichage.h>
#include <listes.h>
#include <code_erreur.h>
#include <personnage.h>
#include <math.h>
#include <map.h>
#include <definition_commun.h>
```

Graphe des dépendances par inclusion de affichage.c:

4.2 Référence du fichier affichage.h

```
#include "SDL2/SDL.h"
#include "definition_commun.h"
#include "listes.h"
#include "map.h"
#include "personnage.h"
```

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

Classes

- struct [s_aff](#)
Structure qui permet l'affichage d'une texture à l'écran de manière précise.
- struct [s_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_chargee.bmp"
- #define [N_T_CHARGER](#) "ressources/sprite/charger.bmp"
- #define [N_T_MARCHER_BOUCIER](#) "ressources/sprite/marcher_bouclier.bmp"
- #define [LONGUEUR_PERSONNAGE](#) 48
- #define [LARGEUR_PERSONNAGE](#) 48

Définitions de type

- typedef `struct s_aff` `t_aff`
 - typedef `struct s_l_aff` `t_l_aff`
- Structure contenant la liste des textures créées par le programme.*

Énumérations

- enum `t_texture_perso` {
`TEXT_MARCHER`, `TEXT_ATTAQUE`, `TEXT_ATTAQUE_CHARGEE`, `TEXT_CHARGER`,
`TEXT_MARCHER_BOUCLIER` }

Fonctions

- void `destruire_texture` (`t_aff` **texture)
Fonction qui détruit une structure d'affichage de texture passée en paramètre.
- void `destruire_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 crée une texture à partir d'un fichier.
- void `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.
- void `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.
- void `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_text_liste` (void *t)
Fonction qui ajoute une texture à la liste.
- void `err_t_afficher_buffer` (list *buffer, SDL_Renderer *rendu)
- void `deplacer_texture_centre` (`t_aff` *texture, int x, int y)
- void `deplacer_rect_origine` (SDL_Rect *r, int x, int y)
- void `deplacer_texture_origine` (`t_aff` *texture, int x, int y)
- void `deplacer_rect_haut_droit` (SDL_Rect *r, int x, int y)
- void `deplacer_texture_haut_droit` (`t_aff` *texture, int x, int y)
- void `deplacer_texture_bas_gauche` (`t_aff` *texture, int x, int y)
- void `deplacer_texture_bas_droit` (`t_aff` *texture, int x, int y)
- void `modif_affichage_rect` (`t_aff` *texture, SDL_Rect r)
- void `deplacement_x_pers` (`t_map` *m, `joueur_t` *j, int x)
- `t_l_aff` * `init_textures_joueur` (`joueur_t` *j)
- `t_aff` * `init_texture_joueur` (`t_l_aff` *textures_joueur)
Fonction qui renvoie la texture de départ du personnage (joueur)
- `t_aff` * `next_frame_joueur` (`joueur_t` *j)
- void `deplacement_y_pers` (`t_map` *m, `joueur_t` *j, int y)
- void `def_texture_taille` (`t_aff` *a_modifier, const int longueur, const int largeur)
- void `text_copier_position` (`t_aff` *a_modifier, const `t_aff` *const original)
- void `rect_centre_x` (SDL_Rect *rectangle)
- void `rect_centre_y` (SDL_Rect *rectangle)
- void `rect_centre` (SDL_Rect *rectangle)
- bool `rects_egal_x` (const SDL_Rect *const r1, const SDL_Rect *const r2)
- bool `rects_egal_y` (const SDL_Rect *const r1, const SDL_Rect *const r2)
- 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 sur une texture.

Variables

- list * `listeDeTextures`
- list * `buffer_affichage`
- long int `compteur`
- SDL_Rect `tx`
- SDL_Rect `ty`
- float `multiplicateur_x`
- float `multiplicateur_y`

4.2.1 Description détaillée

Auteur

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

Version

0.1

Date

12/03/2022

Copyright

Copyright (c) 2022

4.2.2 Documentation des macros

4.2.2.1 LARGEUR_PERSONNAGE

```
#define LARGEUR_PERSONNAGE 48
```

4.2.2.2 LONGUEUR_PERSONNAGE

```
#define LONGUEUR_PERSONNAGE 48
```

4.2.2.3 N_T_ATTAQUE

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

4.2.2.4 N_T_ATTAQUE_CHARGE

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

4.2.2.5 N_T_CHARGER

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

4.2.2.6 N_T_MARCHER

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

4.2.2.7 N_T_MARCHER_BOUCILIER

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

4.2.2.8 NB_FPS

```
#define NB_FPS 60
```

4.2.2.9 NB_SPRITE_JOUEUR

```
#define NB_SPRITE_JOUEUR 5
```

4.2.3 Documentation des définitions de type

4.2.3.1 t_aff

```
typedef struct s_aff t_aff
```

4.2.3.2 t_l_aff

```
typedef struct s_l_aff t_l_aff
```

Structure contenant la liste des textures créées par le programme.

4.2.4 Documentation du type de l'énumération

4.2.4.1 t_texture_perso

```
enum t_texture_perso
```

Valeurs énumérées

TEXT_MARCHER	
TEXT_ATTAQUE	
TEXT_ATTAQUE_CHARGEE	
TEXT_CHARGER	
TEXT_MARCHER_BOUCLIER	

4.2.5 Documentation des fonctions

4.2.5.1 `afficher_buffer()`

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

Fonction qui affiche les textures contenues dans la liste en paramètre.

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

4.2.5.2 `afficher_texture()`

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

Affiche la texture donnée en paramètre à l'écran.

Paramètres

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

Renvoie

0 s'il n'y a pas eu d'erreur

4.2.5.3 ajout_text_liste()

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

4.2.5.4 color()

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

Fonction qui permet de choisir une couleur SDL.

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

4.2.5.5 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 )
```

4.2.5.6 `def_texture_taille()`

```
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 à afficher sur l'écran

Paramètres

<i>a_modifier</i>	La texture à modifier
<i>longueur</i>	La nouvelle longueur en pixel à appliquer
<i>largeur</i>	La nouvelle largeur en pixel à appliquer

4.2.5.7 `deplacement_x_pers()`

```
void deplacement_x_pers (
    t_map * m,
    joueur_t * j,
    int x )
```

Permet de déplacer le personnage de x unités sur la map

Paramètres

<i>map</i>	La map sur laquelle le personnage se déplace
<i>pers</i>	La texture du personnage elle même
<i>x</i>	Le nombre d'unités de déplacements

4.2.5.8 `deplacement_y_pers()`

```
void deplacement_y_pers (
    t_map * m,
    joueur_t * j,
    int y )
```

Permet de déplacer le personnage de y unités sur la map

Paramètres

<i>map</i>	La map sur laquelle le personnage se déplace
<i>pers</i>	La texture du personnage elle même
<i>y</i>	Le nombre d'unités de déplacements

4.2.5.9 `deplacer_rect_haut_droit()`

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

4.2.5.10 `deplacer_rect_origine()`

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

4.2.5.11 `deplacer_texture_bas_droit()`

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

La texture est déplacée vers le coin inférieur droit de la fenêtre

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.

4.2.5.12 `deplacer_texture_bas_gauche()`

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

La texture est déplacée vers le coin inférieur gauche de l'écran

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.

4.2.5.13 `deplacer_texture_centre()`

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

Déplacez la texture pour que son centre soit au centre de l'écran

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.

4.2.5.14 `deplacer_texture_haut_droit()`

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

La texture est déplacée vers la droite et vers le haut

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.

4.2.5.15 `deplacer_texture_origine()`

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

Déplacez l'origine de la texture aux coordonnées données.

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.

4.2.5.16 détruire_liste_textures()

```
void détruire_liste_textures (
    t_l_aff ** l_texture )
```

Fonction qui détruit une liste de textures.

Paramètres

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

4.2.5.17 détruire_texture()

```
void détruire_texture (
    t_aff ** texture )
```

Fonction qui détruit une structure d'affichage de texture passée en paramètre.

La liste des textures créées

Auteur

Despert Ange

Paramètres

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

4.2.5.18 info_texture()

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

Affiche des informations sur une texture.

Paramètres

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

4.2.5.19 `init_texture_joueur()`

```
t_aff* init_texture_joueur (
    t_l_aff * textures_joueur )
```

Fonction qui renvoie la texture de départ du personnage (joueur)

Auteur

Antoine Bruneau

Paramètres

<i>t_l_aff*</i>	la liste de textures personnage
-----------------	---------------------------------

Renvoie

*t_aff** Une textures personnage

4.2.5.20 `init_textures_joueur()`

```
t_l_aff* init_textures_joueur (
    joueur_t * j )
```

4.2.5.21 `modif_affichage_rect()`

```
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

Paramètres

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

4.2.5.22 `next_frame_joueur()`

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

4.2.5.23 next_frame_x()

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

Fonction qui positionne la texture au sprite d'après sur l'axe des x.

Paramètres

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

4.2.5.24 next_frame_x_indice()

```
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.

Paramètres

<i>t_aff*</i>	une texture joueur
<i>const</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

4.2.5.25 next_frame_y()

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

Fonction qui positionne la texture au sprite d'après sur l'axe des y.

Paramètres

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

4.2.5.26 next_frame_y_indice()

```
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.

Paramètres

<i>t_aff*</i>	une texture joueur
<i>const</i>	unsigned int 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

4.2.5.27 rect_centre()

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

4.2.5.28 rect_centre_x()

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

4.2.5.29 rect_centre_y()

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

4.2.5.30 rect_correct_texture()

```
_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.

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

vrai Si tout est OK, faux sinon.

4.2.5.31 `rects_egal_x()`

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

4.2.5.32 `rects_egal_y()`

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

4.2.5.33 `text_copier_position()`

```
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

Paramètres

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

4.2.6 Documentation des variables

4.2.6.1 `buffer_affichage`

```
list* buffer_affichage
```

4.2.6.2 compteur

```
long int compteur
```

4.2.6.3 listeDeTextures

```
list* listeDeTextures
```

4.2.6.4 multiplicateur_x

```
float multiplicateur_x
```

4.2.6.5 multiplicateur_y

```
float multiplicateur_y
```

4.2.6.6 tx

```
SDL_Rect tx
```

4.2.6.7 ty

```
SDL_Rect ty
```

4.3 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 :

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_SAUVEGARDE,  
    ERR_RECTANGLE_TOO_BIG }  
    L'énumération des codes d'erreur du programme.
```

4.3.1 Description détaillée

Fichier contenant les codes d'erreur du programme.

Auteur

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

Version

0.1

Date

02/02/2022

Copyright

Copyright (c) 2022

Ce fichier contient une enumeration des codes d'erreur du programme, permettant ainsi une modification simple et une gestion plus facile des erreurs.

4.3.2 Documentation du type de l'énumération

4.3.2.1 types_erreur

```
enum types_erreur
```

L'énumération des codes d'erreur du programme.

Cela permet une gestion plus simple des erreurs. L'utilisateur peut donc connaitre aisément 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
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 scancodes 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_SAUVEGARDE	Une erreur liée à la création du répertoire de sauvegarde
ERR_RECTANGLE_TOO_BIG	Une erreur liée au rectangle trop grand

4.4 Référence du fichier commun.h

```
#include "code_erreur.h"
#include "definition_commun.h"
#include "fonctions.h"
#include "monstres.h"
#include "personnage.h"
#include "affichage.h"
#include "liste_objet.h"
#include "event.h"
#include "inventaire.h"
#include "menus.h"
```

```
#include "map.h"
#include "interface.h"
```

Graphe des dépendances par inclusion de `commun.h`: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

4.5 Référence du fichier `definition_commun.h`

Contient toutes les définitions communes à tout les fichiers.

```
#include "SDL2/SDL.h"
```

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`
— typedef int `err_t`
— typedef struct `point` `point`

Énumérations

— enum `t_direction` { `NORD`, `EST`, `SUD`, `OUEST` }

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`

4.5.1 Description détaillée

Contient toutes les définitions communes à tout les fichiers.

Auteur

Ange Despert

Version

0.1

Date

20/01/22

4.5.2 Documentation des macros

4.5.2.1 bool

```
#define bool _Bool
```

4.5.2.2 faux

```
#define faux 0
```

4.5.2.3 SAVE_PATH

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

4.5.2.4 vrai

```
#define vrai 1
```

4.5.3 Documentation des définitions de type

4.5.3.1 byte

```
typedef unsigned char byte
```

4.5.3.2 err_t

```
typedef int err_t
```

4.5.3.3 point

```
typedef struct point point
```

4.5.4 Documentation du type de l'énumération

4.5.4.1 t_direction

```
enum t_direction
```

Valeurs énumérées

NORD	
EST	
SUD	
OUEST	

4.5.5 Documentation des fonctions

4.5.5.1 fermer_programme()

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

Fonction qui appelle les fonctions pour terminer le programme.

Auteur

Ange Despert

4.5.6 Documentation des variables

4.5.6.1 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

4.5.6.2 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

4.5.6.3 fenetre_Principale

```
SDL_Window* fenetre_Principale
```

4.5.6.4 rendu_principal

```
SDL_Renderer* rendu_principal
```

Pointeur vers fenêtre globale pointant sur la fenêtre principale du programme

4.5.6.5 running

```
bool running
```

4.6 Référence du fichier definition_ressources.h

Macros

- #define [PLAYER_HEIGHT](#) 21
- #define [PLAYER_WITH](#) 14

4.6.1 Documentation des macros

4.6.1.1 PLAYER_HEIGHT

```
#define PLAYER_HEIGHT 21
```

4.6.1.2 PLAYER_WITH

```
#define PLAYER_WITH 14
```

4.7 Référence du fichier event.c

Fichier qui appelle différentes fonctions en fonction du déclenchement d'événements.

```
#include <commun.h>
#include <event.h>
#include <personnage.h>
Graphe des dépendances par inclusion de event.c:
```

Fonctions

- static void [keyDown](#) (SDL_KeyboardEvent *ev)
Fonction qui gère les événements quand une touche du clavier est pressée.
- static void [keyUp](#) (SDL_KeyboardEvent *ev)
Fonction qui gère les événements quand une touche du clavier est relachée.
- static void [mouseButtonDown](#) (SDL_MouseButtonEvent *ev)
Fonction qui gère les événements quand un bouton de la souris est pressée.
- static void [mouseButtonUp](#) (SDL_MouseButtonEvent *ev)
Fonction qui gère les événements quand un bouton de la souris est relachée.
- void [jeu_event](#) (void)
Fonction qui gère les événements.
- bool [logo_passer](#) (void)

4.7.1 Description détaillée

Fichier qui appelle différentes fonctions en fonction du déclenchement d'événements.

Auteur

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

Version

0.1

Date

02/02/2022

Copyright

Copyright (c) 2022

4.7.2 Documentation des fonctions

4.7.2.1 jeu_event()

```
void jeu_event (
    void )
```

Fonction qui gère les événements.

Auteur

Despert Ange

4.7.2.2 keyDown()

```
void keyDown (
    SDL_KeyboardEvent * ev ) [static]
```

Fonction qui gère les événements quand une touche du clavier est pressée.

Auteur

Antoine Bruneau

Paramètres

<i>SDL_KeyboardEvent*</i>	une structure contenant l'évènement
---------------------------	-------------------------------------

4.7.2.3 keyUp()

```
void keyUp (
    SDL_KeyboardEvent * ev ) [static]
```

Fonction qui gère les événements quand une touche du clavier est relachée.

Auteur

Antoine Bruneau

Paramètres

<i>SDL_KeyboardEvent*</i>	une structure contenant l'évènement
---------------------------	-------------------------------------

4.7.2.4 logo_passer()

```
bool logo_passer (
    void )
```

4.7.2.5 mouseButtonDown()

```
void mouseButtonDown (
    SDL_MouseButtonEvent * ev ) [static]
```

Fonction qui gère les événements quand un bouton de la souris est pressée.

Auteur

Antoine Bruneau

Paramètres

<i>SDL_KeyboardEvent*</i>	une structure contenant l'évènement
---------------------------	-------------------------------------

4.7.2.6 mouseButtonUp()

```
void mouseButtonUp (
    SDL_MouseButtonEvent * ev ) [static]
```

Fonction qui gère les événements quand un bouton de la souris est relachée.

Auteur

Antoine Bruneau

Paramètres

<i>SDL_KeyboardEvent*</i>	une structure contenant l'évènement
---------------------------	-------------------------------------

4.8 Référence du fichier event.h

```
#include "SDL2/SDL_events.h"
```

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
```

Fonctions

```
— _Bool logo_passer (void)  
— void jeu_event (void)  
    Fonction qui gère les événements.
```

4.8.1 Description détaillée

Auteur

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

Version

0.1

Date

03/02/2022

Copyright

Copyright (c) 2022

4.8.2 Documentation des macros

4.8.2.1 TOUCHE_BAS

```
#define TOUCHE_BAS SDLK_s
```

4.8.2.2 TOUCHE_DROITE

```
#define TOUCHE_DROITE SDLK_d
```

4.8.2.3 TOUCHE_ECHAP

```
#define TOUCHE_ECHAP SDLK_ESCAPE
```

4.8.2.4 TOUCHE_GAUCHE

```
#define TOUCHE_GAUCHE SDLK_q
```

4.8.2.5 TOUCHE_HAUT

```
#define TOUCHE_HAUT SDLK_z
```

4.8.2.6 TOUCHE_TAB

```
#define TOUCHE_TAB SDLK_TAB
```

4.8.3 Documentation des fonctions

4.8.3.1 jeu_event()

```
void jeu_event (
    void )
```

Fonction qui gère les événements.

Auteur

Despert Ange

4.8.3.2 logo_passer()

```
_Bool logo_passer (
    void )
```


4.9 Référence du fichier fonctions.h

Fichier qui contient les définitions de toutes les fonctions.

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

Fonctions

- void `init` (void)
Fonction qui initialise le Programme.
- void `terminate_init` (void)
- void `init_event` (void)
Fonction qui initialise les fonctions liées aux événements.

4.9.1 Description détaillée

Fichier qui contient les définitions de toutes les fonctions.

Auteur

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

Version

0.1

Date

02/02/2022

Copyright

Copyright (c) 2022

4.9.2 Documentation des fonctions

4.9.2.1 `init()`

```
void init ( )
```

Fonction qui initialise le Programme.

Auteur

Ange Despert

4.9.2.2 init_event()

```
init_event (
    void )
```

Fonction qui initialise les fonctions liées aux événements.

4.9.2.3 terminate_init()

```
void terminate_init (
    void )
```

4.10 Référence du fichier init_close.c

```
#include <commun.h>
#include <stdio.h>
#include <listes.h>
```

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 [destruire_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 [init_affichage](#) ()
- SDL_Texture * [init_sousbuffer](#) (t_map *map)
- 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
- list * [f_close](#) = NULL

4.10.1 Documentation des fonctions

4.10.1.1 `aff_cleanup()`

```
void aff_cleanup (
    void )
```

Fonction ajoutée à la liste de `atexit()` afin de libérer toute la mémoire allouée.

4.10.1.2 `detruire_renderer()`

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

4.10.1.3 `fermer_programme()`

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

Fonction qui appelle les fonctions pour terminer le programme.

Auteur

Ange Despert

4.10.1.4 `fermer_SDL()`

```
void fermer_SDL (
    void ) [static]
```

Fonction qui détruit la fenêtre principale et ferme la SDL.

Auteur

Ange Despert

4.10.1.5 `init()`

```
void init ( )
```

Fonction qui initialise le Programme.

Auteur

Ange Despert

4.10.1.6 init_affichage()

```
void init_affichage ( )
```

4.10.1.7 init_rc_commun()

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

4.10.1.8 init_SDL()

```
void init_SDL ( ) [static]
```

Fonction qui démarre la SDL et créer la fenêtre principale.

Auteur

Ange Despert

4.10.1.9 init_sousbuffer()

```
SDL_Texture* init_sousbuffer (
    t_map * map )
```

4.10.2 Documentation des variables

4.10.2.1 f_close

```
list* f_close = NULL
```

Liste des fonctions à appeler lors de la fermeture du programme

4.10.2.2 fenetre_Principale

```
SDL_Window* fenetre_Principale = NULL
```

4.10.2.3 fenetre_sous_rendu

```
SDL_Window* fenetre_sous_rendu = NULL
```

4.10.2.4 rendu_principal

```
SDL_Renderer* rendu_principal = NULL
```

Pointeur vers fenêtre globale pointant sur la fenêtre principale du programme

4.10.2.5 running

```
bool running = vrai
```

4.10.2.6 sous_rendu

```
SDL_Renderer* sous_rendu = NULL
```

4.11 Référence du fichier interface.c

```
#include <affichage.h>
```

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.

4.11.1 Documentation des fonctions

4.11.1.1 RenderHPBar()

```
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.

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>vie</i>	couleur de la barre de vie
<i>jauge</i>	couleur de la barre vide

4.12 Référence du fichier interface.h

```
#include "definition_commun.h"
```

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.

4.12.1 Description détaillée

Auteur

Descomps Max

Version

1.0

Date

26/03/2022

Copyright

Copyright (c) 2022

4.12.2 Documentation des fonctions

4.12.2.1 RenderHPBar()

```
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.

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>vie</i>	couleur de la barre de vie
<i>jauge</i>	couleur de la barre vide

4.13 Référence du fichier inventaire.c

```
#include <stdlib.h>
#include <commun.h>
#include <string.h>
```

Graphe des dépendances par inclusion de inventaire.c:

Fonctions

- void `changement_statistiques` (`joueur_t *j`, `lobjet_t *equipe`)
Fonction qui permet de modifier les statistiques du joueur selon une liste d'objets.
- void `equiper_objet` (`joueur_t *j`, `objet_t **objet`, `inventaire_t *inventaire`)
Fonction qui permet de passer un objet du sac(non-équipé) à un emplacement d'équipement(équipé) dans l'inventaire.
- void `desequiper` (`joueur_t *j`, `objet_t **objet`, `inventaire_t *inventaire`)
- `inventaire_t *` `creer_inventaire` ()
Fonction qui permet de creer un inventaire.
- void `detruire_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)

4.13.1 Documentation des fonctions

4.13.1.1 `changement_statistiques()`

```
void changement_statistiques (
    joueur_t * j,
    lobjet_t * liste )
```

Fonction qui permet de modifier les statistiques du joueur selon une liste d'objets.

Paramètres

<i>j</i>	Joueur
<i>equipe</i>	la liste d'objets

4.13.1.2 creer_inventaire()

```
inventaire_t * creer_inventaire ( )
```

Fonction qui permet de creer un inventaire.

Renvoie

Un pointeur sur l'inventaire

4.13.1.3 desequiper()

```
void desequiper (
    joueur_t * j,
    objet_t ** objet,
    inventaire_t * inventaire )
```

4.13.1.4 detruire_inventaire()

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

Détruit un inventaire.

Paramètres

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

4.13.1.5 equiper_objet()

```
void equiper_objet (
    joueur_t * j,
    objet_t ** objet,
    inventaire_t * inventaire )
```

Fonction qui permet de passer un objet du sac(non-équipé) à un emplacement d'équipement(équipé) dans l'inventaire.

Paramètres

<i>j</i>	Joueur à équiper
<i>objet</i>	Objet à équiper
<i>inventaire</i>	Inventaire du joueur

4.13.1.6 ramasser_objet()

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

Fonction qui permet de mettre un objet trouvé dans le sac(inventaire)

Paramètres

<i>objet</i>	Objet trouvé
<i>inventaire</i>	Inventaire du joueur

4.14 Référence du fichier inventaire.h

```
#include "commun.h"
```

Graphe des dépendances par inclusion de inventaire.h: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

— struct [inventaire_s](#)

Définitions de type

— typedef [struct inventaire_s](#) [inventaire_t](#)

Fonctions

- [inventaire_t](#) * [creer_inventaire](#) ()
Fonction qui permet de creer un inventaire.
- void [changement_statistiques](#) ([joueur_t](#) *j, [lobjet_t](#) *equipe)
Fonction qui permet de modifier les statistiques du joueur selon une liste d'objets.
- void [equiper_objet](#) ([joueur_t](#) *j, [objet_t](#) **objet, [inventaire_t](#) *inventaire)
Fonction qui permet de passer un objet du sac(non-équipé) à un emplacement d'équipement(équipé) dans l'inventaire.
- void [desequiper](#) ([joueur_t](#) *j, [objet_t](#) **objet, [inventaire_t](#) *inventaire)
- void [ramasser_objet](#) ([objet_t](#) *objet, [inventaire_t](#) *inventaire)
Fonction qui permet de mettre un objet trouvé dans le sac(inventaire)
- void [destruire_inventaire](#) ([inventaire_t](#) **inventaire)
Détruit un inventaire.

4.14.1 Description détaillée

Auteur

Descomps Max Doneau Rafael

Version

0.2

Date

26/03/2022

Copyright

Copyright (c) 2022

4.14.2 Documentation des définitions de type

4.14.2.1 inventaire_t

```
typedef struct inventaire_s inventaire_t
```

4.14.3 Documentation des fonctions

4.14.3.1 changement_statistiques()

```
void changement_statistiques (
    joueur_t * j,
    lobjet_t * equipe )
```

Fonction qui permet de modifier les statistiques du joueur selon une liste d'objets.

Paramètres

<i>j</i>	Joueur
<i>equipe</i>	la liste d'objets

4.14.3.2 creer_inventaire()

```
inventaire_t* creer_inventaire ( )
```

Fonction qui permet de creer un inventaire.

Renvoie

Un pointeur sur l'inventaire

4.14.3.3 desequiper()

```
void desequiper (
    joueur_t * j,
    objet_t ** objet,
    inventaire_t * inventaire )
```

4.14.3.4 detruire_inventaire()

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

Détruit un inventaire.

Paramètres

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

4.14.3.5 equiper_objet()

```
void equiper_objet (
    joueur_t * j,
    objet_t ** objet,
    inventaire_t * inventaire )
```

Fonction qui permet de passer un objet du sac(non-équipé) à un emplacement d'équipement(équipé) dans l'inventaire.

Paramètres

<i>j</i>	Joueur à équiper
<i>objet</i>	Objet à équiper
<i>inventaire</i>	Inventaire du joueur

4.14.3.6 ramasser_objet()

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

Fonction qui permet de mettre un objet trouvé dans le sac(inventaire)

Paramètres

<i>objet</i>	Objet trouvé
<i>inventaire</i>	Inventaire du joueur

4.15 Référence du fichier liste_objet.c

```
#include <liste_objet.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

Graphe des dépendances par inclusion de liste_objet.c:

Fonctions

- `lobjet_t * creer_liste_objet ()`
- `lobjet_t * creer_liste_objet_vide ()`
- `lobjet_t * creer_liste_objet_equipe ()`
- `void detruire_liste_objet (lobjet_t **liste_obj)`
- `void effacer_liste_objet (lobjet_t **liste_obj)`
- `void afficher_liste_objet (lobjet_t *const liste_obj)`

4.15.1 Documentation des fonctions

4.15.1.1 afficher_liste_objet()

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

4.15.1.2 creer_liste_objet()

```
lobjet_t* creer_liste_objet ( )
```

4.15.1.3 creer_liste_objet_equipe()

```
lobjet_t* creer_liste_objet_equipe ( )
```

4.15.1.4 creer_liste_objet_vide()

```
lobjet_t* creer_liste_objet_vide ( )
```

4.15.1.5 detruire_liste_objet()

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

4.15.1.6 effacer_liste_objet()

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

4.16 Référence du fichier liste_objet.h

Fichier contenant toutes les définitions concernant les listes d'objets.

```
#include "objet.h"
```

Graphe des dépendances par inclusion de liste_objet.h: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

— struct [lobjet_s](#)

Définitions de type

— typedef [struct lobjet_s lobjet_t](#)

Fonctions

- [lobjet_t * creer_liste_objet](#) (void)
- [lobjet_t * creer_liste_objet_vide](#) (void)
- void [detruire_liste_objet](#) ([lobjet_t **liste_obj](#))
- void [afficher_liste_objet](#) ([lobjet_t *const liste_obj](#))
- [lobjet_t * creer_liste_objet_equipe](#) ()
- void [effacer_liste_objet](#) ([lobjet_t **liste_obj](#))

4.16.1 Description détaillée

Fichier contenant toutes les définitions concernant les listes d'objets.

Auteur

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

Version

0.1

Date

24/02/2022

Copyright

Copyright (c) 2022

4.16.2 Documentation des définitions de type

4.16.2.1 lobjet_t

```
typedef struct lobjet_s lobjet_t
```

4.16.3 Documentation des fonctions

4.16.3.1 afficher_liste_objet()

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

4.16.3.2 creer_liste_objet()

```
lobjet_t* creer_liste_objet (
    void )
```

4.16.3.3 `creer_liste_objet_equipe()`

```
lobjet_t* creer_liste_objet_equipe ( )
```

4.16.3.4 `creer_liste_objet_vide()`

```
lobjet_t* creer_liste_objet_vide (
    void )
```

4.16.3.5 `detruire_liste_objet()`

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

4.16.3.6 `effacer_liste_objet()`

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

4.17 Référence du fichier listes.c

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

Graphe des dépendances par inclusion de listes.c:

Classes

- struct `element`
- struct `list`

Définitions de type

- typedef struct `element` `t_element`
- typedef struct `list` `list`

Fonctions

- `list * init_liste` (void *(*fonction_ajout)(void *), void(*fonction_suppression)(void *), void(*fonction_↵
affichage)(void *))
 - `_Bool liste_vide` (const `list` *const to_verify)
 - `_Bool hors_liste` (const `list` *const to_verify)
 - void `en_tete` (`list` *mylist)
 - void `en_queue` (`list` *mylist)
 - void `suivant` (`list` *mylist)
 - void `precedent` (`list` *mylist)
 - void * `valeur_elt` (const `list` *const mylist)
 - void `modif_elt` (`list` *mylist, void *v)
 - void `oter_elt` (`list` *mylist)
 - void `ajout_droit` (`list` *mylist, void *v)
 - void `ajout_gauche` (`list` *mylist, void *v)
 - unsigned int `taille_liste` (const `list` *const mylist)
 - void `vider_liste` (`list` *mylist)
 - void `detruire_liste` (`list` **liste)
 - void `afficher_liste` (`list` *liste)
- Détruit une structure liste.*
Affiche une liste d'objets génériques.

4.17.1 Documentation des définitions de type

4.17.1.1 list

```
typedef struct list list
```

4.17.1.2 t_element

```
typedef struct element t_element
```

4.17.2 Documentation des fonctions

4.17.2.1 afficher_liste()

```
void afficher_liste (
    list * liste )
```

Affiche une liste d'objets génériques.

Paramètres

<i>liste</i>	Pointeur sur la liste
--------------	-----------------------

4.17.2.2 ajout_droit()

```
void ajout_droit (
    list * mylist,
    void * v )
```

Fonction qui permet d'ajouter un élément à droite de l'élément courant

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

4.17.2.3 ajout_gauche()

```
void ajout_gauche (
    list * mylist,
    void * v )
```

Fonction qui permet d'ajouter un élément à gauche de l'élément courant

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

4.17.2.4 detruire_liste()

```
void detruire_liste (
    list ** liste )
```

Détruit une structure liste.

Paramètres

<i>liste</i>	Adresse du pointeur sur la liste
--------------	----------------------------------

4.17.2.5 en_queue()

```
void en_queue (
```

```
list * mylist )
```

Fonction qui permet de se placer en queue de la liste.

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.17.2.6 en_tete()

```
void en_tete (  
    list * mylist )
```

Fonction qui permet de se placer en tête de la liste.

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.17.2.7 hors_liste()

```
_Bool hors_liste (  
    const list *const to_verify )
```

Fonction booléenne qui permet de savoir si l'on est actuellement hors de la liste.

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

4.17.2.8 init_liste()

```
list* init_liste (  
    void (*)(void *) fonction_ajout,  
    void (*)(void *) f_suppresion,  
    void (*)(void *) fonction_affichage )
```

Fonction qui permet de créer une liste

On peut préciser des fonction pour l'insertion et la suppression des objets.

Mais on ne peut pas avoir une fonction d'insertion et aucune fonction de suppression 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

Renvoie

La liste qui vient d'être créée, NULL s'il y a eu une erreur

4.17.2.9 liste_vide()

```
_Bool liste_vide (
    const list *const to_verify )
```

Fonction booléenne qui permet de savoir si la liste est vide.

Paramètres

<i>to_verify</i>	La liste que l'on doit vérifier
------------------	---------------------------------

Renvoie

Vrai si la liste est vide, faux sinon

4.17.2.10 modif_elt()

```
void modif_elt (
    list * mylist,
    void * v )
```

Fonction qui permet de modifier l'élément courant.

ATTENTION : si les élément 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

4.17.2.11 oter_elt()

```
void oter_elt (
    list * mylist )
```

Fonction qui permet de supprimer un élément de la liste.

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
---------------	--------------------------------------

4.17.2.12 precedent()

```
void precedent (
    list * mylist )
```

\Fonction qui permet de passer à l'élément suivant dans la liste

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.17.2.13 suivant()

```
void suivant (
    list * mylist )
```

Fonction qui permet de passer à l'élément suivant dans la liste

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.17.2.14 `taille_liste()`

```
unsigned int taille_liste (
    const list *const mylist )
```

Fonction qui renvoie le nombre d'éléments dans la liste

Paramètres

<i>mylist</i>	La liste dont on veut connaître le nombre d'éléments
---------------	--

Renvoie

Le nombre d'éléments dans la liste

4.17.2.15 `valeur_elt()`

```
void* valeur_elt (
    const list *const mylist )
```

Fonction qui renvoie l'élément courant

Paramètres

<i>mylist</i>	La liste dont on veut l'élément
---------------	---------------------------------

Renvoie

L'élément que l'on convoite

4.17.2.16 `vider_liste()`

```
void vider_liste (
    list * mylist )
```

Fonction qui supprime tout les éléments de la liste.

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
---------------	------------------------------

4.18 Référence du fichier listes.h

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

Définitions de type

- typedef `struct list list`
Définition du type liste.

Fonctions

- `list * init_liste` (void (*)(*fonction_ajout)(void *), void (*)(*f_suppresion)(void *), void (*)(*fonction_affichage)(void *))
- `_Bool liste_vide` (const `list` *const to_verify)
- `_Bool hors_liste` (const `list` *const to_verify)
- void `en_tete` (`list` *mylist)
- void `en_queueue` (`list` *mylist)
- void `suivant` (`list` *mylist)
- void `precedent` (`list` *mylist)
- void * `valeur_elt` (const `list` *const mylist)
- void `modif_elt` (`list` *mylist, void *v)
- void `oter_elt` (`list` *mylist)
- void `ajout_droit` (`list` *mylist, void *v)
- void `ajout_gauche` (`list` *mylist, void *v)
- unsigned int `taille_liste` (const `list` *const mylist)
- void `vider_liste` (`list` *mylist)
- void `detruire_liste` (`list` **liste)
Détruit une structure liste.
- void `afficher_liste` (`list` *liste)
Affiche une liste d'objets génériques.

4.18.1 Documentation des définitions de type

4.18.1.1 list

```
typedef struct list list
```

Définition du type liste.

4.18.2 Documentation des fonctions

4.18.2.1 afficher_liste()

```
void afficher_liste (
    list * liste )
```

Affiche une liste d'objets génériques.

Paramètres

<i>liste</i>	Pointeur sur la liste
--------------	-----------------------

4.18.2.2 ajout_droit()

```
void ajout_droit (
    list * mylist,
    void * v )
```

Fonction qui permet d'ajouter un élément à droite de l'élément courant

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

4.18.2.3 ajout_gauche()

```
void ajout_gauche (
    list * mylist,
    void * v )
```

Fonction qui permet d'ajouter un élément à gauche de l'élément courant

Paramètres

<i>mylist</i>	La liste où on veut ajouter l'élément
<i>v</i>	L'élément que l'on veut ajouter

4.18.2.4 detruire_liste()

```
void detruire_liste (
    list ** liste )
```

Détruit une structure liste.

Paramètres

<i>liste</i>	Adresse du pointeur sur la liste
--------------	----------------------------------

4.18.2.5 en_queue()

```
void en_queue (
    list * mylist )
```

Fonction qui permet de se placer en queue de la liste.

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.18.2.6 en_tete()

```
void en_tete (
    list * mylist )
```

Fonction qui permet de se placer en tête de la liste.

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.18.2.7 hors_liste()

```
_Bool hors_liste (
    const list *const to_verify )
```

Fonction booléenne qui permet de savoir si l'on est actuellement hors de la liste.

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

4.18.2.8 init_liste()

```
list* init_liste (
    void (*)(void *) fonction_ajout,
```



```
void(*) (void *) f_suppresion,  
void(*) (void *) fonction_affichage )
```

Fonction qui permet de créer une liste

On peut préciser des fonction pour l'insertion et la suppression des objets.

Mais on ne peut pas avoir une fonction d'insertion et aucune fonction de suppression 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

Renvoie

La liste qui vient d'être créée, NULL s'il y a eu une erreur

4.18.2.9 liste_vide()

```
_Bool liste_vide (  
    const list *const to_verify )
```

Fonction booléenne qui permet de savoir si la liste est vide.

Paramètres

<i>to_verify</i>	La liste que l'on doit vérifier
------------------	---------------------------------

Renvoie

Vrai si la liste est vide, faux sinon

4.18.2.10 modif_elt()

```
void modif_elt (  
    list * mylist,  
    void * v )
```

Fonction qui permet de modifier l'élément courant.

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

4.18.2.11 oter_elt()

```
void oter_elt (
    list * mylist )
```

Fonction qui permet de supprimer un élément de la liste.

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
---------------	--------------------------------------

4.18.2.12 precedent()

```
void precedent (
    list * mylist )
```

\Fonction qui permet de passer à l'élément suivant dans la liste

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.18.2.13 suivant()

```
void suivant (
    list * mylist )
```

Fonction qui permet de passer à l'élément suivant dans la liste

Paramètres

<i>mylist</i>	La liste dans laquelle on se déplace
---------------	--------------------------------------

4.18.2.14 `taille_liste()`

```
unsigned int taille_liste (
    const list *const mylist )
```

Fonction qui renvoie le nombre d'éléments dans la liste

Paramètres

<i>mylist</i>	La liste dont on veut connaître le nombre d'éléments
---------------	--

Renvoie

Le nombre d'éléments dans la liste

4.18.2.15 `valeur_elt()`

```
void* valeur_elt (
    const list *const mylist )
```

Fonction qui renvoie l'élément courant

Paramètres

<i>mylist</i>	La liste dont on veut l'élément
---------------	---------------------------------

Renvoie

L'élément que l'on convoite

4.18.2.16 `vider_liste()`

```
void vider_liste (
    list * mylist )
```

Fonction qui supprime tout les éléments de la liste.

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
---------------	------------------------------

4.19 Référence du fichier main.c

```
#include <commun.h>
#include <stdio.h>
#include <math.h>
#include <map.h>
```

Graphe des dépendances par inclusion de main.c:

Macros

— #define [SDL_MAIN_HANDLED](#)

Fonctions

— void [afficher_intro](#) (void)
— int [main](#) (int argc, char **argv)

4.19.1 Documentation des macros

4.19.1.1 SDL_MAIN_HANDLED

```
#define SDL_MAIN_HANDLED
```

4.19.2 Documentation des fonctions

4.19.2.1 afficher_intro()

```
void afficher_intro (
    void )
```

4.19.2.2 main()

```
int main (
    int argc,
    char ** argv )
```

4.20 Référence du fichier map.c

```
#include <json-c/json.h>
#include <affichage.h>
#include <map.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <code_erreur.h>
#include <math.h>
```

Graphe des dépendances par inclusion de map.c:

Fonctions

- SDL_Rect [taille_ecran_cases](#) ()
- char * [charger_f_map](#) (const char *const nom_map)
Fonction qui charge le contenu du fichier dont le nom est donné en paramètre dans un buffer de caractères.
- [t_map](#) * [charger_s_map](#) (char *buffer)
Fonction qui récupère les informations stockées dans le buffer en entrée.
- [t_aff](#) * [texture_map](#) (const [t_map](#) *map)
Fonction qui renvoie la texture de la map.

Variables

- [t_map](#) * map

4.20.1 Documentation des fonctions

4.20.1.1 charger_f_map()

```
char* charger_f_map (
    const char *const nom_map )
```

Fonction qui charge le contenu du fichier dont le nom est donné en paramètre dans un buffer de caractères.

Paramètres

<i>nom_map</i>	Le nom du fichier map à charger
----------------	---------------------------------

Renvoie

Un buffer de caractères contenant l'intégralité du fichier

4.20.1.2 charger_s_map()

```
t_map* charger_s_map (
    char * buffer )
```

Fonction qui récupère les informations stockées dans le buffer en entrée.

Paramètres

<i>buffer</i>	Le buffer qui contient les informations
---------------	---

Renvoie

Une map initialisée avec toutes les informations dedans;

4.20.1.3 taille_ecran_cases()

```
SDL_Rect taille_ecran_cases ( )
```

4.20.1.4 texture_map()

```
t_aff* texture_map (
    const t_map * map )
```

Fonction qui renvoie la texture de la map.

Paramètres

<i>map</i>	La map dont on veut la texture
------------	--------------------------------

Renvoie

La texture de la map

4.20.2 Documentation des variables

4.20.2.1 map

`t_map*` map

La map courante

4.21 Référence du fichier map.h

Le fichier contient les définitions des fonctions de gestion de la map.

```
#include "definition_commun.h"
#include "listes.h"
```

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

Définitions de type

- typedef `struct s_aff t_aff`
Structure de texture.

Fonctions

- `char *` `charger_f_map` (`const char *``const nom_map`)
Fonction qui charge le contenu du fichier dont le nom est donné en paramètre dans un buffer de caractères.
- `t_map *` `charger_s_map` (`char *``buffer`)
Fonction qui récupère les informations stockées dans le buffer en entrée.
- `t_aff *` `texture_map` (`const t_map *``map`)
Fonction qui renvoie la texture de la map.

Variables

- `t_map *` `map`

4.21.1 Description détaillée

Le fichier contient les définitions des fonctions de gestion de la map.

Auteur

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

Version

1.0

Date

27/03/2022

Copyright

Copyright (c) 2022

4.21.2 Documentation des macros

4.21.2.1 TAILLE_CASE

```
#define TAILLE_CASE 16
```

4.21.3 Documentation des définitions de type

4.21.3.1 t_aff

```
typedef struct s_aff t_aff
```

Structure de texture.

4.21.4 Documentation des fonctions

4.21.4.1 charger_f_map()

```
char* charger_f_map (
    const char *const nom_map )
```

Fonction qui charge le contenu du fichier dont le nom est donné en paramètre dans un buffer de caractères.

Paramètres

<i>nom_map</i>	Le nom du fichier map à charger
----------------	---------------------------------

Renvoie

Un buffer de caractères contenant l'intégralité du fichier

4.21.4.2 charger_s_map()

```
t_map* charger_s_map (
    char * buffer )
```

Fonction qui récupère les informations stockées dans le buffer en entrée.

Paramètres

<i>buffer</i>	Le buffer qui contient les informations
---------------	---

Renvoie

Une map initialisée avec toutes les informations dedans;

4.21.4.3 texture_map()

```
t_aff* texture_map (
    const t_map * map )
```

Fonction qui renvoie la texture de la map.

Paramètres

<i>map</i>	La map dont on veut la texture
------------	--------------------------------

Renvoie

La texture de la map

4.21.5 Documentation des variables

4.21.5.1 map

`t_map*` map

La map courante

4.22 Référence du fichier menus.c

```
#include <stdlib.h>
#include <string.h>
#include <affichage.h>
#include <personnage.h>
```

Graphe des dépendances par inclusion de menus.c:

Fonctions

- void `afficher_menu_pause` ()
- void `afficher_inventaire` ()

4.22.1 Documentation des fonctions

4.22.1.1 `afficher_inventaire()`

```
void afficher_inventaire ( )
```

4.22.1.2 `afficher_menu_pause()`

```
void afficher_menu_pause ( )
```

4.23 Référence du fichier menus.h

```
#include "definition_commun.h"
```

Graphe des dépendances par inclusion de menus.h: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Énumérations

- enum `menus_t` { `JEU`, `PAUSE`, `PRINCIPAL`, `INVENTAIRE` }

Fonctions

- void `afficher_menu_pause` ()
- void `afficher_inventaire` ()

Variables

- `menus_t` `menus`

4.23.1 Documentation du type de l'énumération

4.23.1.1 `menus_t`

enum `menus_t`

Valeurs énumérées

JEU	
PAUSE	
PRINCIPAL	
INVENTAIRE	

4.23.2 Documentation des fonctions

4.23.2.1 `afficher_inventaire()`

```
void afficher_inventaire ( )
```

4.23.2.2 `afficher_menu_pause()`

```
void afficher_menu_pause ( )
```

4.23.3 Documentation des variables

4.23.3.1 menus

`menus_t` menus

4.24 Référence du fichier monstres.c

```
#include <monstres.h>
```

```
#include <string.h>
```

Graphe des dépendances par inclusion de monstres.c:

Fonctions

- void `détruire_monstre` (`monstre_t **monstre`)
- void `détruire_liste_base_monstres` (`liste_base_monstres_t **liste_base_monstres`)
- `liste_base_monstres_t * charger_monstres` (`char *nom_fichier`)
Fonction qui recopie les informations d'un fichier pour les insérer dans une structure `liste_base_monstres_t`.

4.24.1 Documentation des fonctions

4.24.1.1 charger_monstres()

```
liste_base_monstres_t * charger_monstres (  
    char * nom_fichier )
```

Fonction qui recopie les informations d'un fichier pour les insérer dans une structure `liste_base_monstres_t`.

Paramètres

<i>nom_fichier</i>	nom du fichier à lire
--------------------	-----------------------

Renvoie

`liste_base_monstres_t*` une structure contenant la liste des monstres

4.24.1.2 détruire_liste_base_monstres()

```
void détruire_liste_base_monstres (  
    liste_base_monstres_t ** liste_base_monstres )
```

4.24.1.3 détruire_monstre()

```
void détruire_monstre (
    monstre_t ** monstre )
```

4.25 Référence du fichier monstres.h

```
#include "definition_commun.h"
#include <affichage.h>
```

Graphes 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_s`

Définitions de type

- typedef `point position_t`
- typedef `struct monstre_s monstre_t`
- typedef `struct base_monstre_s base_monstre_t`
- typedef `struct liste_base_monstres_s liste_base_monstres_t`

Fonctions

- void `détruire_liste_base_monstres` (`liste_base_monstres_t **liste_base_monstres`)
- void `détruire_monstre` (`monstre_t **monstre`)
- `liste_base_monstres_t * charger_monstres` (`char *nom_fichier`)
Fonction qui recopie les informations d'un fichier pour les insérer dans une structure `liste_base_monstres_t`.

4.25.1 Documentation des définitions de type

4.25.1.1 base_monstre_t

```
typedef struct base_monstre_s base_monstre_t
```

4.25.1.2 liste_base_monstres_t

```
typedef struct liste_base_monstres_s liste_base_monstres_t
```

4.25.1.3 monstre_t

```
typedef struct monstre_s monstre_t
```

4.25.1.4 position_t

```
typedef point position_t
```

4.25.2 Documentation des fonctions

4.25.2.1 charger_monstres()

```
liste_base_monstres_t* charger_monstres (
    char * nom_fichier )
```

Fonction qui recopie les informations d'un fichier pour les insérer dans une structure liste_base_monstres_t.

Paramètres

<i>nom_fichier</i>	nom du fichier à lire
--------------------	-----------------------

Renvoie

liste_base_monstres_t* une structure contenant la liste des monstres

4.25.2.2 detruire_liste_base_monstres()

```
void detruire_liste_base_monstres (
    liste_base_monstres_t ** liste_base_monstres )
```

4.25.2.3 detruire_monstre()

```
void detruire_monstre (
    monstre_t ** monstre )
```

4.26 Référence du fichier objet.c

Fichier contenant toutes les fonctions concernant les objets.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <objet.h>
```

Graphe des dépendances par inclusion de objet.c:

Fonctions

- `objet_t * creer_objet` (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)
- void `afficher_objet` (`objet_t` *obj)
- void `detruire_objet` (`objet_t` **obj)

4.26.1 Description détaillée

Fichier contenant toutes les fonctions concernant les objets.

Auteur

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

Version

0.2

Date

24/02/2022

Copyright

Copyright (c) 2022

4.26.2 Documentation des fonctions

4.26.2.1 afficher_objet()

```
void afficher_objet (
    objet_t * obj )
```

4.26.2.2 creer_objet()

```
objet_t* creer_objet (
    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 )
```

4.26.2.3 detruire_objet()

```
void detruire_objet (
    objet_t ** obj )
```

4.27 Référence du fichier objet.h

Fichier contenant toutes les définitions concernant les objets.

```
#include "definition_commun.h"
#include "affichage.h"
```

Graphe des dépendances par inclusion de objet.h: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

— struct [objet_s](#)

Définitions de type

— typedef [struct objet_s](#) [objet_t](#)

Énumérations

— enum [t_item](#) {
[quete](#), [arme](#), [bouclier](#), [protection](#),
[amulette](#), [consommable](#) }

Fonctions

— void [detruire_texture](#) ([t_aff](#) **texture)
Fonction qui détruit une structure d'affichage de texture passée en paramètre.
 — [objet_t](#) * [creer_objet](#) (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)
 — void [detruire_objet](#) ([objet_t](#) **obj)
 — [objet_t](#) ** [init_objet](#) (void)
 — void [afficher_objet](#) ([objet_t](#) *obj)

4.27.1 Description détaillée

Fichier contenant toutes les définitions concernant les objets.

Auteur

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

Version

0.3

Date

24/02/2022

Copyright

Copyright (c) 2022

4.27.2 Documentation des définitions de type

4.27.2.1 objet_t

```
typedef struct objet_s objet_t
```

4.27.3 Documentation du type de l'énumération

4.27.3.1 t_item

```
enum t_item
```

Valeurs énumérées

quete	
arme	
bouclier	
protection	
amulette	
consommable	

4.27.4 Documentation des fonctions

4.27.4.1 afficher_objet()

```
void afficher_objet (
    objet_t * obj )
```

4.27.4.2 creer_objet()

```
objet_t* creer_objet (
    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 )
```

4.27.4.3 detruire_objet()

```
void detruire_objet (
    objet_t ** obj )
```

4.27.4.4 detruire_texture()

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

Fonction qui détruit une structure d'affichage de texture passée en paramètre.

Auteur

Despert Ange

Paramètres

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

4.27.4.5 init_objet()

```
objet_t** init_objet (
    void )
```

4.28 Référence du fichier personnage.c

Fichier contenant toutes les fonctions concernant le personnage.

```
#include <stdlib.h>
#include <string.h>
#include <personnage.h>
#include <affichage.h>
#include <json-c/json.h>
#include <sys/stat.h>
#include <errno.h>
#include <code_erreur.h>
#include <pwd.h>
#include <unistd.h>
```

Graphe des dépendances par inclusion de personnage.c:

Fonctions

- void `copy` (const byte *origin, byte *out, size_t size)
- void `check_repertoire_jeux` ()
- void `creer_sauvegarde_json` (joueur_t *j)
- bool `sauv_existe` (char *nom_sauv)
- joueur_t * `charger_sauvegarde_joueur` (char *nom_sauv)
- joueur_t * `new_joueur` (const char *nom)
- 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 orientation, const bool bouclier_equipe)
- void `destruire_joueur` (joueur_t *j)
- joueur_t * `caracteristiques` (joueur_t *perso)
- void `afficher_statistiques` (joueur_t *perso)
- joueur_t * `levelup` (joueur_t *perso)
- joueur_t * `gain_xp` (joueur_t *perso)

Variables

- char `save_path` [500]
- joueur_t * `perso_principal`

4.28.1 Description détaillée

Fichier contenant toutes les fonctions concernant le personnage.

Auteur

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

Version

0.1

Date

01/02/2022

Copyright

Copyright (c) 2022

4.28.2 Documentation des fonctions

4.28.2.1 afficher_statistiques()

```
void afficher_statistiques (
    joueur_t * perso )
```

4.28.2.2 caracteristiques()

```
joueur_t* caracteristiques (
    joueur_t * perso )
```

4.28.2.3 charger_sauvegarde_joueur()

```
joueur_t* charger_sauvegarde_joueur (
    char * nom_sauv )
```

4.28.2.4 check_repertoire_jeux()

```
void check_repertoire_jeux ( )
```

4.28.2.5 copy()

```
void copy (
    const byte * origin,
    byte * out,
    size_t size )
```

4.28.2.6 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 orientation,
    const bool bouclier_equipe )
```

4.28.2.7 creer_sauvegarde_json()

```
void creer_sauvegarde_json (
    joueur_t * j )
```

4.28.2.8 detruire_joueur()

```
void detruire_joueur (
    joueur_t * j )
```

4.28.2.9 gain_xp()

```
joueur_t* gain_xp (
    joueur_t * perso )
```

4.28.2.10 levelup()

```
joueur_t* levelup (
    joueur_t * perso )
```

4.28.2.11 new_joueur()

```
joueur_t* new_joueur (
    const char * nom )
```

4.28.2.12 sauv_existe()

```
bool sauv_existe (
    char * nom_sauv )
```

4.28.3 Documentation des variables

4.28.3.1 perso_principal

```
joueur_t* perso_principal
```

4.28.3.2 save_path

```
char save_path[500]
```

4.29 Référence du fichier personnage.h

Fichier contenant toutes les définitions concernant le personnage.

```
#include "definition_commun.h"
```

Grappe des dépendances par inclusion de personnage.h: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

- struct [statut_s](#)
- struct [joueur_s](#)

Macros

- #define [DUREE_ATTAQUE_OU_CHARGE](#) 4
- #define [DUREE_ATTAQUE](#) 6
- #define [DUREE_ATTAQUE_CHARGE](#) 10
- #define [DUREE_BLOQUER](#) 3
- #define [TAILLE_PERSONNAGE](#) 16 /*La taille du personnage en pixels*/
- #define [TAILLE_TRIGGER](#) 200

Définitions de type

- typedef struct [s_l_aff](#) [t_l_aff](#)
- typedef struct [statut_s](#) [statut_t](#)
- typedef unsigned char [byte](#)
- typedef struct [joueur_s](#) [joueur_t](#)

Énumérations

- enum `action_t` {
 `RIEN`, `ATTAQUE`, `ATTAQUE_CHARGEE`, `CHARGER`,
 `BLOQUER`, `ATTAQUE_OU_CHARGER` }

Fonctions

- `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 orientation, const bool bouclier_equipe)
- `joueur_t * new_joueur` (const char *nom)
- void `destruire_joueur` (joueur_t *j)
- `joueur_t * charger_sauvegarde_joueur` (char *nom_sauv)
- `joueur_t * caracteristiques` (joueur_t *perso)
- void `afficher_statistiques` (joueur_t *perso)
- `joueur_t * levelup` (joueur_t *perso)
- `joueur_t * gain_xp` (joueur_t *perso)
- void `creer_sauvegarde_json` (joueur_t *j)
- void `check_repertoire_jeux` ()

Variables

- `joueur_t * perso_principal`
- char `save_path` [500]

4.29.1 Description détaillée

Fichier contenant toutes les définitions concernant le personnage.

Auteur

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

Version

0.1

Date

01/02/2022

Copyright

Copyright (c) 2022

4.29.2 Documentation des macros

4.29.2.1 DUREE_ATTAQUE

```
#define DUREE_ATTAQUE 6
```

4.29.2.2 DUREE_ATTAQUE_CHARGEE

```
#define DUREE_ATTAQUE_CHARGEE 10
```

4.29.2.3 DUREE_ATTAQUE_OU_CHARGEE

```
#define DUREE_ATTAQUE_OU_CHARGEE 4
```

4.29.2.4 DUREE_BLOQUER

```
#define DUREE_BLOQUER 3
```

4.29.2.5 TAILLE_PERSONNAGE

```
#define TAILLE_PERSONNAGE 16 /*La taille du personnage en pixels*/
```

4.29.2.6 TAILLE_TRIGGER

```
#define TAILLE_TRIGGER 200
```

4.29.3 Documentation des définitions de type

4.29.3.1 byte

```
typedef unsigned char byte
```

4.29.3.2 joueur_t

```
typedef struct joueur_s joueur_t
```


4.29.3.3 statut_t

```
typedef struct statut_s statut_t
```

4.29.3.4 t_l_aff

```
typedef struct s_l_aff t_l_aff
```

4.29.4 Documentation du type de l'énumération

4.29.4.1 action_t

```
enum action_t
```

Valeurs énumérées

RIEN	
ATTAQUE	
ATTAQUE_CHARGEE	
CHARGER	
BLOQUER	
ATTAQUE_OU_CHARGER	

4.29.5 Documentation des fonctions

4.29.5.1 afficher_statistiques()

```
void afficher_statistiques (
    joueur_t * perso )
```

4.29.5.2 caracteristiques()

```
joueur_t* caracteristiques (
    joueur_t * perso )
```

4.29.5.3 charger_sauvegarde_joueur()

```
joueur_t* charger_sauvegarde_joueur (
    char * nom_sauv )
```

4.29.5.4 check_repertoire_jeux()

```
void check_repertoire_jeux ( )
```

4.29.5.5 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 orientation,
    const bool bouclier_equipe )
```

4.29.5.6 creer_sauvegarde_json()

```
void creer_sauvegarde_json (
    joueur_t * j )
```

4.29.5.7 detruire_joueur()

```
void detruire_joueur (
    joueur_t * j )
```

4.29.5.8 gain_xp()

```
joueur_t* gain_xp (
    joueur_t * perso )
```

4.29.5.9 levelup()

```
joueur_t* levelup (
    joueur_t * perso )
```

4.29.5.10 new_joueur()

```
joueur_t* new_joueur (
    const char * nom )
```

4.29.6 Documentation des variables

4.29.6.1 perso_principal

```
joueur_t* perso_principal
```

4.29.6.2 save_path

```
char save_path[500]
```

4.30 Référence du fichier test_affichage.c

```
#include <commun.h>
```

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`

4.30.1 Documentation des fonctions

4.30.1.1 main()

```
int main ( )
```

4.30.2 Documentation des variables

4.30.2.1 compteur

```
long int compteur
```

4.30.2.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

4.30.2.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

4.30.2.4 test_map

```
t_map* test_map
```

4.31 Référence du fichier test_inventaire.c

```
#include <commun.h>
#include <personnage.h>
#include <objet.h>
#include <inventaire.h>
```

Graphe des dépendances par inclusion de test_inventaire.c:

Fonctions

— int [main](#) ()

Variables

- long int `compteur`
- `t_map` * `test_map`
- unsigned int `FENETRE_LONGUEUR`
- unsigned int `FENETRE_LARGEUR`

4.31.1 Documentation des fonctions

4.31.1.1 `main()`

```
int main ( )
```

4.31.2 Documentation des variables

4.31.2.1 `compteur`

```
long int compteur
```

4.31.2.2 `FENETRE_LARGEUR`

```
unsigned int FENETRE_LARGEUR
```

4.31.2.3 `FENETRE_LONGUEUR`

```
unsigned int FENETRE_LONGUEUR
```

4.31.2.4 `test_map`

```
t_map* test_map
```

4.32 Référence du fichier test_liste_objet.c

```
#include <liste_objet.h>
```

Graphe des dépendances par inclusion de test_liste_objet.c:

Fonctions

— int `main` ()

Variables

— SDL_Window * `fenetre_Principale` = NULL
— SDL_Renderer * `rendu_principal` = NULL
— bool `running` = vrai
— long int `compteur`
— t_map * `test_map`
— unsigned int `FENETRE_LONGUEUR`
— unsigned int `FENETRE_LARGEUR`

4.32.1 Documentation des fonctions

4.32.1.1 `main()`

```
int main ( )
```

4.32.2 Documentation des variables

4.32.2.1 `compteur`

```
long int compteur
```

4.32.2.2 `FENETRE_LARGEUR`

```
unsigned int FENETRE_LARGEUR
```

4.32.2.3 `FENETRE_LONGUEUR`

```
unsigned int FENETRE_LONGUEUR
```

4.32.2.4 fenetre_Principale

```
SDL_Window* fenetre_Principale = NULL
```

4.32.2.5 rendu_principal

```
SDL_Renderer* rendu_principal = NULL
```

Pointeur vers fenêtre globale pointant sur la fenêtre principale du programme

4.32.2.6 running

```
bool running = vrai
```

4.32.2.7 test_map

```
t_map* test_map
```

4.33 Référence du fichier test_listes.c

```
#include <commun.h>
#include <listes.h>
#include <affichage.h>
```

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`

4.33.1 Documentation des fonctions

4.33.1.1 afficher_int()

```
void afficher_int (
    int * nb )
```

4.33.1.2 main()

```
int main ( )
```

4.33.2 Documentation des variables

4.33.2.1 compteur

```
long int compteur
```

4.33.2.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

4.33.2.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

4.33.2.4 test_map

```
t_map* test_map
```

4.34 Référence du fichier test_map.c

```
#include <fonctions.h>
#include <event.h>
#include <affichage.h>
#include <definition_commun.h>
#include <map.h>
#include <stdio.h>
```

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`

4.34.1 Documentation des fonctions

4.34.1.1 `main()`

```
int main (  
    int argc,  
    char ** argv )
```

4.34.2 Documentation des variables

4.34.2.1 `compteur`

```
long int compteur
```

4.34.2.2 `FENETRE_LARGEUR`

```
unsigned int FENETRE_LARGEUR
```

4.34.2.3 `FENETRE_LONGUEUR`

```
unsigned int FENETRE_LONGUEUR
```

4.34.2.4 `test_map`

```
t_map* test_map
```

4.35 Référence du fichier test_monstres.c

```
#include <commun.h>
#include <stdio.h>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de test_monstres.c:

Fonctions

— int `main` ()

4.35.1 Documentation des fonctions

4.35.1.1 `main()`

```
int main ( )
```

4.36 Référence du fichier test_personnage.c

```
#include <commun.h>
#include <personnage.h>
```

Graphe des dépendances par inclusion de test_personnage.c:

Fonctions

— int `main` ()

Variables

— long int `compteur`
— `t_map` * `test_map`
— unsigned int `FENETRE_LONGUEUR`
— unsigned int `FENETRE_LARGEUR`

4.36.1 Documentation des fonctions

4.36.1.1 `main()`

```
int main ( )
```

4.36.2 Documentation des variables

4.36.2.1 compteur

```
long int compteur
```

4.36.2.2 FENETRE_LARGEUR

```
unsigned int FENETRE_LARGEUR
```

4.36.2.3 FENETRE_LONGUEUR

```
unsigned int FENETRE_LONGUEUR
```

4.36.2.4 test_map

```
t_map* test_map
```


Index

action
 statut_s, 18
action_t
 personnage.h, 97
aff
 list, 10
aff_cleanup
 init_close.c, 50
aff_fenetre
 s_aff, 16
affichage.c, 23
affichage.h, 23
 afficher_buffer, 27
 afficher_texture, 27
 ajout_text_liste, 28
 buffer_affichage, 36
 color, 28
 compteur, 36
 creer_texture, 28
 def_texture_taille, 28
 deplacement_x_pers, 29
 deplacement_y_pers, 29
 deplacer_rect_haut_droit, 30
 deplacer_rect_origine, 30
 deplacer_texture_bas_droit, 30
 deplacer_texture_bas_gauche, 30
 deplacer_texture_centre, 31
 deplacer_texture_haut_droit, 31
 deplacer_texture_origine, 31
 detruire_liste_textures, 32
 detruire_texture, 32
 info_texture, 32
 init_texture_joueur, 32
 init_textures_joueur, 33
 LARGEUR_PERSONNAGE, 25
 listeDeTextures, 37
 LONGUEUR_PERSONNAGE, 25
 modif_affichage_rect, 33
 multiplicateur_x, 37
 multiplicateur_y, 37
 N_T_ATTAQUE, 25
 N_T_ATTAQUE_CHARGEES, 25
 N_T_CHARGER, 25
 N_T_MARCHER, 26
 N_T_MARCHER_BOUCIER, 26
 NB_FPS, 26
 NB_SPRITE_JOUEUR, 26
 next_frame_joueur, 33
 next_frame_x, 33
 next_frame_x_indice, 34
 next_frame_y, 34
 next_frame_y_indice, 34
 rect_centre, 35
 rect_centre_x, 35
 rect_centre_y, 35
 rect_correct_texture, 35
 rects_egal_x, 36
 rects_egal_y, 36
 t_aff, 26
 t_l_aff, 26
 t_texture_perso, 26
 TEXT_ATTAQUE, 27
 TEXT_ATTAQUE_CHARGEES, 27
 TEXT_CHARGER, 27
 text_copier_position, 36
 TEXT_MARCHER, 27
 TEXT_MARCHER_BOUCIER, 27
 tx, 37
 ty, 37
afficher_buffer
 affichage.h, 27
afficher_int
 test_listes.c, 103
afficher_intro
 main.c, 76
afficher_inventaire
 menus.c, 82
 menus.h, 83
afficher_liste
 listes.c, 64
 listes.h, 70
afficher_liste_objet
 liste_objet.c, 60
 liste_objet.h, 62
afficher_menu_pause
 menus.c, 82
 menus.h, 83
afficher_objet
 objet.c, 87
 objet.h, 90
afficher_statistiques
 personnage.c, 92
 personnage.h, 97
afficher_texture
 affichage.h, 27
ajout
 list, 10
ajout_droit

- listes.c, [65](#)
- listes.h, [71](#)
- ajout_gauche
 - listes.c, [65](#)
 - listes.h, [71](#)
- ajout_text_liste
 - affichage.h, [28](#)
- amulette
 - objet.h, [89](#)
- arme
 - objet.h, [89](#)
- ATTAQUE
 - personnage.h, [97](#)
- attaque
 - base_monstre_s, [5](#)
 - joueur_s, [8](#)
 - monstre_s, [12](#)
 - objet_s, [14](#)
- ATTAQUE_CHARGEES
 - personnage.h, [97](#)
- ATTAQUE_OU_CHARGER
 - personnage.h, [97](#)
- AUCUNE_ERREUR
 - code_erreur.h, [39](#)
- base_monstre_s, [5](#)
 - attaque, [5](#)
 - fichier_image, [5](#)
 - gainXp, [5](#)
 - nom_monstre, [5](#)
 - pdv, [6](#)
 - vitesse, [6](#)
- base_monstre_t
 - monstres.h, [85](#)
- BLOQUER
 - personnage.h, [97](#)
- bool
 - definition_commun.h, [41](#)
- bouclier
 - objet.h, [89](#)
- bouclier_equipe
 - statut_s, [18](#)
- buffer_affichage
 - affichage.h, [36](#)
- BUFFER_EMPTY
 - code_erreur.h, [39](#)
- byte
 - definition_commun.h, [41](#)
 - personnage.h, [96](#)
- caracteristiques
 - personnage.c, [92](#)
 - personnage.h, [97](#)
- changement_statistiques
 - inventaire.c, [55](#)
 - inventaire.h, [58](#)
- CHARGER
 - personnage.h, [97](#)
- charger_f_map
 - map.c, [77](#)
 - map.h, [80](#)
- charger_monstres
 - monstres.c, [84](#)
 - monstres.h, [86](#)
- charger_s_map
 - map.c, [78](#)
 - map.h, [81](#)
- charger_sauvegarde_joueur
 - personnage.c, [92](#)
 - personnage.h, [97](#)
- check_repertoire_jeux
 - personnage.c, [92](#)
 - personnage.h, [98](#)
- code_erreur.h, [37](#)
 - AUCUNE_ERREUR, [39](#)
 - BUFFER_EMPTY, [39](#)
 - ERR_CREATION_REPERTOIRE_SAUVEGARDE, [39](#)
 - ERR_RECTANGLE_TOO_BIG, [39](#)
 - ERREUR_FICHER, [39](#)
 - ERREUR_FONCTION, [39](#)
 - ERREUR_INIT, [39](#)
 - ERREUR_LISTE, [39](#)
 - ERREUR_MAP, [39](#)
 - ERREUR_SDL_AUDIO, [39](#)
 - ERREUR_SDL_EVENT, [39](#)
 - ERREUR_SDL_IMG, [39](#)
 - ERREUR_SDL_JOYSTICK, [39](#)
 - ERREUR_SDL_KEYBOARD, [39](#)
 - ERREUR_SDL_MIX, [39](#)
 - ERREUR_SDL_MOUSE, [39](#)
 - ERREUR_SDL_MOUSEBUTTON, [39](#)
 - ERREUR_SDL_MOUSEMOTION, [39](#)
 - ERREUR_SDL_MUSIC, [39](#)
 - ERREUR_SDL_PIXEL, [39](#)
 - ERREUR_SDL_QUIT, [39](#)
 - ERREUR_SDL_RENDER, [39](#)
 - ERREUR_SDL_SCANCODE, [39](#)
 - ERREUR_SDL_SURFACE, [39](#)
 - ERREUR_SDL_TIMER, [39](#)
 - ERREUR_SDL_TIMER_START, [39](#)
 - ERREUR_SDL_TIMER_STOP, [39](#)
 - ERREUR_SDL_TTF, [39](#)
 - ERREUR_SDL_WINDOW, [39](#)
 - ERREUR_TEXTURE, [39](#)
 - OUT_OF_MEM, [39](#)
 - SDL_ERREUR, [39](#)
 - types_erreur, [38](#)
- color
 - affichage.h, [28](#)
- commun.h, [39](#)
- compteur
 - affichage.h, [36](#)
 - test_affichage.c, [100](#)
 - test_inventaire.c, [101](#)
 - test_liste_objet.c, [102](#)
 - test_listes.c, [104](#)

- test_map.c, 105
- test_personnage.c, 107
- consommable
 - objet.h, 89
- copy
 - personnage.c, 92
- creer_inventaire
 - inventaire.c, 56
 - inventaire.h, 58
- creer_joueur
 - personnage.c, 92
 - personnage.h, 98
- creer_liste_objet
 - liste_objet.c, 60
 - liste_objet.h, 62
- creer_liste_objet_equipe
 - liste_objet.c, 60
 - liste_objet.h, 62
- creer_liste_objet_vide
 - liste_objet.c, 61
 - liste_objet.h, 63
- creer_objet
 - objet.c, 87
 - objet.h, 90
- creer_sauvegarde_json
 - personnage.c, 93
 - personnage.h, 98
- creer_texture
 - affichage.h, 28
- def_texture_taille
 - affichage.h, 28
- defense
 - joueur_s, 8
 - objet_s, 14
- definition_commun.h, 40
 - bool, 41
 - byte, 41
 - err_t, 42
 - EST, 42
 - faux, 41
 - FENETRE_LARGEUR, 43
 - FENETRE_LONGUEUR, 43
 - fenetre_Principale, 43
 - fermer_programme, 42
 - NORD, 42
 - OUEST, 42
 - point, 42
 - rendu_principal, 43
 - running, 43
 - SAVE_PATH, 41
 - SUD, 42
 - t_direction, 42
 - vrai, 41
- definition_ressources.h, 43
 - PLAYER_HEIGHT, 43
 - PLAYER_WITH, 44
- del
 - list, 10
- deplacement_x_pers
 - affichage.h, 29
- deplacement_y_pers
 - affichage.h, 29
- deplacer_rect_haut_droit
 - affichage.h, 30
- deplacer_rect_origine
 - affichage.h, 30
- deplacer_texture_bas_droit
 - affichage.h, 30
- deplacer_texture_bas_gauche
 - affichage.h, 30
- deplacer_texture_centre
 - affichage.h, 31
- deplacer_texture_haut_droit
 - affichage.h, 31
- deplacer_texture_origine
 - affichage.h, 31
- desequiper
 - inventaire.c, 56
 - inventaire.h, 59
- dest
 - zone_tp, 21
- detruire_inventaire
 - inventaire.c, 56
 - inventaire.h, 59
- detruire_joueur
 - personnage.c, 93
 - personnage.h, 98
- detruire_liste
 - listes.c, 65
 - listes.h, 71
- detruire_liste_base_monstres
 - monstres.c, 84
 - monstres.h, 86
- detruire_liste_objet
 - liste_objet.c, 61
 - liste_objet.h, 63
- detruire_liste_textures
 - affichage.h, 32
- detruire_monstre
 - monstres.c, 84
 - monstres.h, 86
- detruire_objet
 - objet.c, 88
 - objet.h, 90
- detruire_renderer
 - init_close.c, 51
- detruire_texture
 - affichage.h, 32
 - objet.h, 90
- duree
 - statut_s, 18
- DUREE_ATTAQUE
 - personnage.h, 95
- DUREE_ATTAQUE_CHARGE
 - personnage.h, 95
- DUREE_ATTAQUE_OU_CHARGE

- personnage.h, 96
- DUREE_BLOQUER
 - personnage.h, 96
- duree_frame_anim
 - s_aff, 16
- ec
 - list, 10
- effacer_liste_objet
 - liste_objet.c, 61
 - liste_objet.h, 63
- element, 6
 - pred, 6
 - succ, 6
 - valeur, 6
- en_mouvement
 - statut_s, 18
- en_queue
 - listes.c, 65
 - listes.h, 72
- en_tete
 - listes.c, 66
 - listes.h, 72
- equipe
 - inventaire_s, 7
- equiper_objet
 - inventaire.c, 56
 - inventaire.h, 59
- ERR_CREATION_REPERTOIRE_SAUVEGARDE
 - code_erreur.h, 39
- ERR_RECTANGLE_TOO_BIG
 - code_erreur.h, 39
- err_t
 - definition_commun.h, 42
- ERREUR_FICHER
 - code_erreur.h, 39
- ERREUR_FONCTION
 - code_erreur.h, 39
- ERREUR_INIT
 - code_erreur.h, 39
- ERREUR_LISTE
 - code_erreur.h, 39
- ERREUR_MAP
 - code_erreur.h, 39
- ERREUR_SDL_AUDIO
 - code_erreur.h, 39
- ERREUR_SDL_EVENT
 - code_erreur.h, 39
- ERREUR_SDL_IMG
 - code_erreur.h, 39
- ERREUR_SDL_JOYSTICK
 - code_erreur.h, 39
- ERREUR_SDL_KEYBOARD
 - code_erreur.h, 39
- ERREUR_SDL_MIX
 - code_erreur.h, 39
- ERREUR_SDL_MOUSE
 - code_erreur.h, 39
- ERREUR_SDL_MOUSEBUTTON
 - code_erreur.h, 39
- ERREUR_SDL_MOUSEMOTION
 - code_erreur.h, 39
- ERREUR_SDL_MUSIC
 - code_erreur.h, 39
- ERREUR_SDL_PIXEL
 - code_erreur.h, 39
- ERREUR_SDL_QUIT
 - code_erreur.h, 39
- ERREUR_SDL_RENDER
 - code_erreur.h, 39
- ERREUR_SDL_SCANCODE
 - code_erreur.h, 39
- ERREUR_SDL_SURFACE
 - code_erreur.h, 39
- ERREUR_SDL_TIMER
 - code_erreur.h, 39
- ERREUR_SDL_TIMER_START
 - code_erreur.h, 39
- ERREUR_SDL_TIMER_STOP
 - code_erreur.h, 39
- ERREUR_SDL_TTF
 - code_erreur.h, 39
- ERREUR_SDL_WINDOW
 - code_erreur.h, 39
- ERREUR_TEXTURE
 - code_erreur.h, 39
- EST
 - definition_commun.h, 42
- event.c, 44
 - jeu_event, 45
 - keyDown, 45
 - keyUp, 45
 - logo_passer, 46
 - mouseButtonDown, 46
 - mouseButtonUp, 46
- event.h, 46
 - jeu_event, 48
 - logo_passer, 48
 - TOUCHE_BAS, 47
 - TOUCHE_DROITE, 47
 - TOUCHE_ECHAP, 47
 - TOUCHE_GAUCHE, 48
 - TOUCHE_HAUT, 48
 - TOUCHE_TAB, 48
- f_close
 - init_close.c, 52
- faux
 - definition_commun.h, 41
- FENETRE_LARGEUR
 - definition_commun.h, 43
 - test_affichage.c, 100
 - test_inventaire.c, 101
 - test_liste_objet.c, 102
 - test_listes.c, 104
 - test_map.c, 105
 - test_personnage.c, 107
- FENETRE_LONGUEUR

- definition_commun.h, 43
- test_affichage.c, 100
- test_inventaire.c, 101
- test_liste_objet.c, 102
- test_listes.c, 104
- test_map.c, 105
- test_personnage.c, 107
- fenetre_Principale
 - definition_commun.h, 43
 - init_close.c, 52
 - test_liste_objet.c, 102
- fenetre_sous_rendu
 - init_close.c, 52
- fermer_programme
 - definition_commun.h, 42
 - init_close.c, 51
- fermer_SDL
 - init_close.c, 51
- fichier_image
 - base_monstre_s, 5
- flag
 - list, 10
- fonctions.h, 49
 - init, 49
 - init_event, 49
 - terminate_init, 50
- frame_anim
 - s_aff, 16
- gain_xp
 - personnage.c, 93
 - personnage.h, 98
- gainXp
 - base_monstre_s, 5
 - monstre_s, 12
- height
 - s_aff, 16
 - t_map, 20
- hors_liste
 - listes.c, 66
 - listes.h, 72
- id_map
 - zone_tp, 22
- info_texture
 - affichage.h, 32
- init
 - fonctions.h, 49
 - init_close.c, 51
- init_affichage
 - init_close.c, 51
- init_close.c, 50
 - aff_cleanup, 50
 - destruire_render, 51
 - f_close, 52
 - fenetre_Principale, 52
 - fenetre_sous_rendu, 52
 - fermer_programme, 51
 - fermer_SDL, 51
 - init, 51
 - init_affichage, 51
 - init_rc_commun, 52
 - init_SDL, 52
 - init_sousbuffer, 52
 - rendu_principal, 53
 - running, 53
 - sous_rendu, 53
- init_event
 - fonctions.h, 49
- init_liste
 - listes.c, 66
 - listes.h, 72
- init_objet
 - objet.h, 90
- init_rc_commun
 - init_close.c, 52
- init_SDL
 - init_close.c, 52
- init_sousbuffer
 - init_close.c, 52
- init_texture_joueur
 - affichage.h, 32
- init_textures_joueur
 - affichage.h, 33
- interface.c, 53
 - RenderHPBar, 53
- interface.h, 54
 - RenderHPBar, 54
- INVENTAIRE
 - menus.h, 83
- inventaire.c, 55
 - changement_statistiques, 55
 - creer_inventaire, 56
 - desequiper, 56
 - destruire_inventaire, 56
 - equiper_objet, 56
 - ramasser_objet, 57
- inventaire.h, 57
 - changement_statistiques, 58
 - creer_inventaire, 58
 - desequiper, 59
 - destruire_inventaire, 59
 - equiper_objet, 59
 - inventaire_t, 58
 - ramasser_objet, 59
- inventaire_s, 7
 - equipe, 7
 - sac, 7
- inventaire_t
 - inventaire.h, 58
- JEU
 - menus.h, 83
- jeu_event
 - event.c, 45
 - event.h, 48
- joueur_s, 7

- attaque, 8
- defense, 8
- maxPdv, 8
- niveau, 8
- nom_pers, 8
- pdv, 8
- statut, 9
- textures_joueur, 9
- trigger, 9
- vitesse, 9
- xp, 9
- joueur_t
 - personnage.h, 96
- keyDown
 - event.c, 45
- keyUp
 - event.c, 45
- LARGEUR_PERSONNAGE
 - affichage.h, 25
- levelup
 - personnage.c, 93
 - personnage.h, 98
- list, 9
 - aff, 10
 - ajout, 10
 - del, 10
 - ec, 10
 - flag, 10
 - listes.c, 64
 - listes.h, 70
 - nb_elem, 10
- liste
 - lobjet_s, 11
 - s_l_aff, 17
- liste_base_monstres_s, 11
 - nb_monstre, 11
 - tab, 11
- liste_base_monstres_t
 - monstres.h, 85
- liste_monstres
 - t_map, 20
- liste_objet.c, 60
 - afficher_liste_objet, 60
 - creer_liste_objet, 60
 - creer_liste_objet_equipe, 60
 - creer_liste_objet_vide, 61
 - destruire_liste_objet, 61
 - effacer_liste_objet, 61
- liste_objet.h, 61
 - afficher_liste_objet, 62
 - creer_liste_objet, 62
 - creer_liste_objet_equipe, 62
 - creer_liste_objet_vide, 63
 - destruire_liste_objet, 63
 - effacer_liste_objet, 63
 - lobjet_t, 62
- liste_vide
 - listes.c, 67
 - listes.h, 73
- listeDeTextures
 - affichage.h, 37
- listes.c, 63
 - afficher_liste, 64
 - ajout_droit, 65
 - ajout_gauche, 65
 - destruire_liste, 65
 - en_queue, 65
 - en_tete, 66
 - hors_liste, 66
 - init_liste, 66
 - list, 64
 - liste_vide, 67
 - modif_elt, 67
 - oter_elt, 68
 - precedent, 68
 - suivant, 68
 - t_element, 64
 - taille_liste, 68
 - valeur_elt, 69
 - vider_liste, 69
- listes.h, 70
 - afficher_liste, 70
 - ajout_droit, 71
 - ajout_gauche, 71
 - destruire_liste, 71
 - en_queue, 72
 - en_tete, 72
 - hors_liste, 72
 - init_liste, 72
 - list, 70
 - liste_vide, 73
 - modif_elt, 73
 - oter_elt, 74
 - precedent, 74
 - suivant, 74
 - taille_liste, 75
 - valeur_elt, 75
 - vider_liste, 75
- lobjet_s, 11
 - liste, 11
 - nb, 12
- lobjet_t
 - liste_objet.h, 62
- logo_passer
 - event.c, 46
 - event.h, 48
- LONGUEUR_PERSONNAGE
 - affichage.h, 25
- main
 - main.c, 76
 - test_affichage.c, 99
 - test_inventaire.c, 101
 - test_liste_objet.c, 102
 - test_listes.c, 104
 - test_map.c, 105

- test_monstres.c, 106
- test_personnage.c, 106
- main.c, 76
 - afficher_intro, 76
 - main, 76
 - SDL_MAIN_HANDLED, 76
- map
 - map.c, 78
 - map.h, 81
- map.c, 77
 - charger_f_map, 77
 - charger_s_map, 78
 - map, 78
 - taille_ecran_cases, 78
 - texture_map, 78
- map.h, 79
 - charger_f_map, 80
 - charger_s_map, 81
 - map, 81
 - t_aff, 80
 - TAILLE_CASE, 80
 - texture_map, 81
- maxPdv
 - joueur_s, 8
- menus
 - menus.h, 83
- menus.c, 82
 - afficher_inventaire, 82
 - afficher_menu_pause, 82
- menus.h, 82
 - afficher_inventaire, 83
 - afficher_menu_pause, 83
 - INVENTAIRE, 83
 - JEU, 83
 - menus, 83
 - menus_t, 83
 - PAUSE, 83
 - PRINCIPAL, 83
- menus_t
 - menus.h, 83
- modif_affichage_rect
 - affichage.h, 33
- modif_elt
 - listes.c, 67
 - listes.h, 73
- monstre_s, 12
 - attaque, 12
 - gainXp, 12
 - nom_monstre, 12
 - orientation, 13
 - pdv, 13
 - position, 13
 - texture, 13
 - vitesse, 13
- monstre_t
 - monstres.h, 85
- monstres.c, 84
 - charger_monstres, 84
 - détruire_liste_base_monstres, 84
 - détruire_monstre, 84
- monstres.h, 85
 - base_monstre_t, 85
 - charger_monstres, 86
 - détruire_liste_base_monstres, 86
 - détruire_monstre, 86
 - liste_base_monstres_t, 85
 - monstre_t, 85
 - position_t, 86
- mouseButtonDown
 - event.c, 46
- mouseButtonUp
 - event.c, 46
- multipli_taille
 - s_aff, 16
- multiplicateur_x
 - affichage.h, 37
- multiplicateur_y
 - affichage.h, 37
- N_T_ATTAQUE
 - affichage.h, 25
- N_T_ATTAQUE_CHARGEE
 - affichage.h, 25
- N_T_CHARGER
 - affichage.h, 25
- N_T_MARCHER
 - affichage.h, 26
- N_T_MARCHER_BOUCLIER
 - affichage.h, 26
- nb
 - lobjet_s, 12
- nb_elem
 - list, 10
- NB_FPS
 - affichage.h, 26
- nb_monstre
 - liste_base_monstres_s, 11
- NB_SPRITE_JOUEUR
 - affichage.h, 26
- nb_valeurs
 - s_l_aff, 17
- new_joueur
 - personnage.c, 93
 - personnage.h, 99
- next_frame_joueur
 - affichage.h, 33
- next_frame_x
 - affichage.h, 33
- next_frame_x_indice
 - affichage.h, 34
- next_frame_y
 - affichage.h, 34
- next_frame_y_indice
 - affichage.h, 34
- niveau
 - joueur_s, 8
 - objet_s, 14

- nom
 - objet_s, 14
- nom_monstre
 - base_monstre_s, 5
 - monstre_s, 12
- nom_pers
 - joueur_s, 8
- NORD
 - definition_commun.h, 42
- objet.c, 87
 - afficher_objet, 87
 - creer_objet, 87
 - destruire_objet, 88
- objet.h, 88
 - afficher_objet, 90
 - amulette, 89
 - arme, 89
 - bouclier, 89
 - consommable, 89
 - creer_objet, 90
 - destruire_objet, 90
 - destruire_texture, 90
 - init_objet, 90
 - objet_t, 89
 - protection, 89
 - quete, 89
 - t_item, 89
- objet_s, 13
 - attaque, 14
 - defense, 14
 - niveau, 14
 - nom, 14
 - texture, 14
 - texture_src, 14
 - type, 15
 - vitesse, 15
- objet_t
 - objet.h, 89
- orientation
 - monstre_s, 13
 - statut_s, 18
- oter_elt
 - listes.c, 68
 - listes.h, 74
- OUEST
 - definition_commun.h, 42
- OUT_OF_MEM
 - code_erreur.h, 39
- p1
 - zone_tp, 22
- p2
 - zone_tp, 22
- PAUSE
 - menus.h, 83
- pdv
 - base_monstre_s, 6
 - joueur_s, 8
- monstre_s, 13
- perso_principal
 - personnage.c, 94
 - personnage.h, 99
- personnage.c, 91
 - afficher_statistiques, 92
 - caracteristiques, 92
 - charger_sauvegarde_joueur, 92
 - check_repertoire_jeux, 92
 - copy, 92
 - creer_joueur, 92
 - creer_sauvegarde_json, 93
 - destruire_joueur, 93
 - gain_xp, 93
 - levelup, 93
 - new_joueur, 93
 - perso_principal, 94
 - sauv_existe, 93
 - save_path, 94
- personnage.h, 94
 - action_t, 97
 - afficher_statistiques, 97
 - ATTAQUE, 97
 - ATTAQUE_CHARGEES, 97
 - ATTAQUE_OU_CHARGER, 97
 - BLOQUER, 97
 - byte, 96
 - caracteristiques, 97
 - CHARGER, 97
 - charger_sauvegarde_joueur, 97
 - check_repertoire_jeux, 98
 - creer_joueur, 98
 - creer_sauvegarde_json, 98
 - destruire_joueur, 98
 - DUREE_ATTAQUE, 95
 - DUREE_ATTAQUE_CHARGEES, 95
 - DUREE_ATTAQUE_OU_CHARGEES, 96
 - DUREE_BLOQUER, 96
 - gain_xp, 98
 - joueur_t, 96
 - levelup, 98
 - new_joueur, 99
 - perso_principal, 99
 - RIEN, 97
 - save_path, 99
 - statut_t, 96
 - t_l_aff, 97
 - TAILLE_PERSONNAGE, 96
 - TAILLE_TRIGGER, 96
- PLAYER_HEIGHT
 - definition_ressources.h, 43
- PLAYER_WITH
 - definition_ressources.h, 44
- point, 15
 - definition_commun.h, 42
 - x, 15
 - y, 15
- position

- monstre_s, 13
- position_t
 - monstres.h, 86
- precedent
 - listes.c, 68
 - listes.h, 74
- pred
 - element, 6
- PRINCIPAL
 - menus.h, 83
- protection
 - objet.h, 89
- quete
 - objet.h, 89
- ramasser_objet
 - inventaire.c, 57
 - inventaire.h, 59
- rect_centre
 - affichage.h, 35
- rect_centre_x
 - affichage.h, 35
- rect_centre_y
 - affichage.h, 35
- rect_correct_texture
 - affichage.h, 35
- rects_egal_x
 - affichage.h, 36
- rects_egal_y
 - affichage.h, 36
- RenderHPBar
 - interface.c, 53
 - interface.h, 54
- rendu_principal
 - definition_commun.h, 43
 - init_close.c, 53
 - test_liste_objet.c, 103
- RIEN
 - personnage.h, 97
- running
 - definition_commun.h, 43
 - init_close.c, 53
 - test_liste_objet.c, 103
- s_aff, 16
 - aff_fenetre, 16
 - duree_frame_anim, 16
 - frame_anim, 16
 - height, 16
 - multipli_taille, 16
 - texture, 17
 - width, 17
- s_l_aff, 17
 - liste, 17
 - nb_valeurs, 17
- sac
 - inventaire_s, 7
- sauv_existe
 - personnage.c, 93
- SAVE_PATH
 - definition_commun.h, 41
- save_path
 - personnage.c, 94
 - personnage.h, 99
- SDL_ERREUR
 - code_erreur.h, 39
- SDL_MAIN_HANDLED
 - main.c, 76
- sous_rendu
 - init_close.c, 53
- statut
 - joueur_s, 9
- statut_s, 18
 - action, 18
 - bouclier_equipe, 18
 - duree, 18
 - en_mouvement, 18
 - orientation, 18
 - zone_colision, 19
- statut_t
 - personnage.h, 96
- struct, 19
- succ
 - element, 6
- SUD
 - definition_commun.h, 42
- suivant
 - listes.c, 68
 - listes.h, 74
- t_aff
 - affichage.h, 26
 - map.h, 80
- t_direction
 - definition_commun.h, 42
- t_element
 - listes.c, 64
- t_item
 - objet.h, 89
- t_l_aff
 - affichage.h, 26
 - personnage.h, 97
- t_map, 20
 - height, 20
 - liste_monstres, 20
 - text_map, 20
 - unite_dep_x, 20
 - unite_dep_y, 21
 - width, 21
- t_texture_perso
 - affichage.h, 26
- tab
 - liste_base_monstres_s, 11
- TAILLE_CASE
 - map.h, 80
- taille_ecran_cases
 - map.c, 78

taille_liste
 listes.c, 68
 listes.h, 75
TAILLE_PERSONNAGE
 personnage.h, 96
TAILLE_TRIGGER
 personnage.h, 96
terminate_init
 fonctions.h, 50
test_affichage.c, 99
 compteur, 100
 FENETRE_LARGEUR, 100
 FENETRE_LONGUEUR, 100
 main, 99
 test_map, 100
test_inventaire.c, 100
 compteur, 101
 FENETRE_LARGEUR, 101
 FENETRE_LONGUEUR, 101
 main, 101
 test_map, 101
test_liste_objet.c, 101
 compteur, 102
 FENETRE_LARGEUR, 102
 FENETRE_LONGUEUR, 102
 fenetre_Principale, 102
 main, 102
 rendu_principal, 103
 running, 103
 test_map, 103
test_listes.c, 103
 afficher_int, 103
 compteur, 104
 FENETRE_LARGEUR, 104
 FENETRE_LONGUEUR, 104
 main, 104
 test_map, 104
test_map
 test_affichage.c, 100
 test_inventaire.c, 101
 test_liste_objet.c, 103
 test_listes.c, 104
 test_map.c, 105
 test_personnage.c, 107
test_map.c, 104
 compteur, 105
 FENETRE_LARGEUR, 105
 FENETRE_LONGUEUR, 105
 main, 105
 test_map, 105
test_monstres.c, 106
 main, 106
test_personnage.c, 106
 compteur, 107
 FENETRE_LARGEUR, 107
 FENETRE_LONGUEUR, 107
 main, 106
 test_map, 107
TEXT_ATAQUE
 affichage.h, 27
TEXT_ATAQUE_CHARGE
 affichage.h, 27
TEXT_CHARGER
 affichage.h, 27
text_copier_position
 affichage.h, 36
text_map
 t_map, 20
TEXT_MARCHER
 affichage.h, 27
TEXT_MARCHER_BOUC
 affichage.h, 27
texture
 monstre_s, 13
 objet_s, 14
 s_aff, 17
texture_map
 map.c, 78
 map.h, 81
texture_src
 objet_s, 14
textures_joueur
 joueur_s, 9
TOUCHE_BAS
 event.h, 47
TOUCHE_DROITE
 event.h, 47
TOUCHE_ECHAP
 event.h, 47
TOUCHE_GAUCHE
 event.h, 48
TOUCHE_HAUT
 event.h, 48
TOUCHE_TAB
 event.h, 48
trigger
 joueur_s, 9
tx
 affichage.h, 37
ty
 affichage.h, 37
type
 objet_s, 15
types_erreur
 code_erreur.h, 38
unite_dep_x
 t_map, 20
unite_dep_y
 t_map, 21
valeur
 element, 6
valeur_elt
 listes.c, 69
 listes.h, 75
vider_liste

- listes.c, [69](#)
- listes.h, [75](#)
- vitesse
 - base_monstre_s, [6](#)
 - joueur_s, [9](#)
 - monstre_s, [13](#)
 - objet_s, [15](#)
- vrai
 - definition_commun.h, [41](#)
- width
 - s_aff, [17](#)
 - t_map, [21](#)
- x
 - point, [15](#)
- xp
 - joueur_s, [9](#)
- y
 - point, [15](#)
- zone_collision
 - statut_s, [19](#)
- zone_tp, [21](#)
 - dest, [21](#)
 - id_map, [22](#)
 - p1, [22](#)
 - p2, [22](#)