

Bataille-Navale

v.1.0

Généré par Doxygen 1.8.1.1

Lundi Juin 25 2012 09 :42 :35

Table des matières

1	Projet Bataille-Navale	1
2	Installation et Compilation	3
2.1	Installation	3
2.1.1	Windows	3
2.1.2	MacOS X	3
2.2	Compilation	3
2.2.1	Windows	3
2.2.2	MacOS X	4
3	Présentation Générale	5
3.1	Comment Jouer ?	5
3.1.1	Nouvelle Partie	5
3.1.2	Charger une Partie	5
3.1.3	Écran de jeu	5
3.2	Explications générales	6
3.3	Explications techniques	6
4	Index des structures de données	7
4.1	Structures de données	7
5	Index des fichiers	9
5.1	Liste des fichiers	9
6	Documentation des structures de données	11
6.1	Référence de la structure CaseGrille	11
6.1.1	Description détaillée	11
6.1.2	Documentation des champs	11
6.1.2.1	couleur	11
6.1.2.2	etatCase	11
6.1.2.3	idBateauOccupe	11
6.2	Référence de la structure Cellule	11
6.2.1	Documentation des champs	12

6.2.1.1	Info	12
6.2.1.2	Lien	12
6.3	Référence de la structure ChampSaisie	12
6.3.1	Description détaillée	12
6.3.2	Documentation des champs	12
6.3.2.1	abscisse	12
6.3.2.2	chaine	12
6.3.2.3	longMax	12
6.3.2.4	onFocus	12
6.3.2.5	ordonnee	13
6.3.2.6	tailleTexte	13
6.4	Référence de la structure Coord	13
6.4.1	Description détaillée	13
6.4.2	Documentation des champs	13
6.4.2.1	noCol	13
6.4.2.2	noLin	13
6.5	Référence de la structure Couleur	13
6.6	Référence de la structure Coup	14
6.7	Référence de la structure CPSPProcessSerNum	14
6.8	Référence de la structure Grille	14
6.8.1	Description détaillée	14
6.8.2	Documentation des champs	14
6.8.2.1	abscisse	14
6.8.2.2	Matrice	14
6.8.2.3	NbCol	14
6.8.2.4	NbLin	15
6.8.2.5	ordonnee	15
6.9	Référence de la structure Image	15
6.9.1	Description détaillée	15
6.9.2	Documentation des champs	15
6.9.2.1	abscisse	15
6.9.2.2	hauteur	15
6.9.2.3	longueur	15
6.9.2.4	ordonnee	15
6.9.2.5	zoneImage	15
6.10	Référence de la structure Joueur	16
6.10.1	Documentation des champs	16
6.10.1.1	mesBateaux	16
6.10.1.2	nomJ	16
6.11	Référence de la structure Ligne	16

6.11.1 Description détaillée	16
6.12 Référence de la classe <code>NSApplication</code>	16
6.13 Référence de la classe <code>NSApplication(SDL_Missing_Methods)</code>	16
6.14 Référence de la classe <code>NSApplication(SDLApplication)</code>	16
6.15 Référence de la classe <code>NSString</code>	17
6.16 Référence de la structure <code>Rectangle</code>	17
6.16.1 Description détaillée	17
6.16.2 Documentation des champs	17
6.16.2.1 abscisse	17
6.16.2.2 couleur	17
6.16.2.3 hauteur	17
6.16.2.4 largeur	17
6.16.2.5 ordonnee	17
6.16.2.6 zoneRectangle	17
6.17 Référence de la structure <code>Score</code>	18
6.17.1 Description détaillée	18
6.17.2 Documentation des champs	18
6.17.2.1 nom	18
6.17.2.2 score	18
6.18 Référence de la structure <code>SDL_Bouton</code>	18
6.18.1 Description détaillée	18
6.18.2 Documentation des champs	18
6.18.2.1 abscisse	18
6.18.2.2 largCarac	19
6.18.2.3 longTexte	19
6.18.2.4 ordonnee	19
6.18.2.5 tailleTexte	19
6.18.2.6 texte	19
6.19 Référence de la structure <code>SDL_MsgBox</code>	19
6.20 Référence de la classe <code>SDLMain</code>	19
6.21 Référence de la structure <code>TBateau</code>	19
6.21.1 Description détaillée	20
6.21.2 Documentation des champs	20
6.21.2.1 estPlace	20
6.21.2.2 etat	20
6.21.2.3 idBateau	20
6.21.2.4 position	20
6.22 Référence de la structure <code>TInfoBateau</code>	20
6.22.1 Description détaillée	20
6.22.2 Documentation des champs	20

6.22.2.1	couleur	20
6.22.2.2	nomBateau	20
6.22.2.3	type	21
6.23	Référence de la structure Tparam	21
6.23.1	Description détaillée	21
6.23.2	Documentation des champs	21
6.23.2.1	bateauxJoueur	21
6.23.2.2	bateauxMachine	21
6.23.2.3	nombreInstanceBateaux	21
6.24	Référence de la structure TPartie	21
6.24.1	Description détaillée	22
6.24.2	Documentation des champs	22
6.24.2.1	grille	22
6.24.2.2	grilleMachine	22
6.24.2.3	joueur	22
6.24.2.4	machine	22
6.24.2.5	parametres	22
6.24.2.6	pileCoups	22
6.24.2.7	scorePlayer	22
6.25	Référence de la structure TPosition	22
6.25.1	Description détaillée	23
6.25.2	Documentation des champs	23
6.25.2.1	direction	23
6.25.2.2	x	23
6.25.2.3	y	23
6.26	Référence de la structure TSensBat	23
6.26.1	Documentation des champs	23
6.26.1.1	libSens	23
6.26.1.2	sensBat	23
6.27	Référence de la structure TtypeBat	23
6.27.1	Description détaillée	24
6.27.2	Documentation des champs	24
6.27.2.1	nomType	24
6.27.2.2	typeBat	24
7	Documentation des fichiers	25
7.1	Référence du fichier src/ctrl/EcransDivers.h	25
7.1.1	Description détaillée	25
7.2	Référence du fichier src/ctrl/FichierDebug.h	25
7.2.1	Description détaillée	25

7.2.2	Documentation des fonctions	26
7.2.2.1	debug	26
7.2.2.2	destruire_debug	26
7.2.2.3	dgAttention	26
7.2.2.4	dgErreur	26
7.2.2.5	dgFatal	27
7.2.2.6	dglInfo	27
7.2.2.7	dgSDL	27
7.2.2.8	init_debug	27
7.3	Référence du fichier src/ctrl/FichierMeilleursScores.h	27
7.3.1	Description détaillée	28
7.3.2	Documentation des fonctions	28
7.3.2.1	ajouterScore	28
7.3.2.2	enregistrerTabScore	28
7.3.2.3	getMeilleursScoresFichier	28
7.3.2.4	ouvrirFichierMeilleursScores	28
7.3.2.5	placeScoreTableau	29
7.4	Référence du fichier src/ctrl/FichierSauvRes.h	29
7.4.1	Description détaillée	29
7.4.2	Documentation des fonctions	29
7.4.2.1	restaurerBateaux	29
7.4.2.2	restaurerCoups	30
7.4.2.3	restaurerGrilles	30
7.4.2.4	restaurerParam	30
7.4.2.5	restaurerPartie	30
7.4.2.6	sauvegardeBateaux	31
7.4.2.7	sauvegardeCoups	31
7.4.2.8	sauvegardeGrille	31
7.4.2.9	sauvegardeParam	31
7.4.2.10	sauvegardePartie	32
7.5	Référence du fichier src/ctrl/Jeu.h	32
7.5.1	Description détaillée	32
7.5.2	Documentation des fonctions	32
7.5.2.1	changerSensBat	32
7.5.2.2	coordAleat	33
7.5.2.3	ecranJeu	33
7.5.2.4	jeu	33
7.5.2.5	menuPause	33
7.5.2.6	menuPlacementChoixBat	33
7.5.2.7	menuPlacementGrille	33

7.5.2.8	placementAleatBat	33
7.5.2.9	placementBatValide	34
7.6	Référence du fichier src/ctrl/Menu.h	34
7.6.1	Description détaillée	34
7.6.2	Documentation des fonctions	34
7.6.2.1	afficherMenuAccueil	34
7.6.2.2	afficherMenuRacine	34
7.6.2.3	menuNouvellePartie	35
7.6.2.4	menuParam	35
7.7	Référence du fichier src/ctrl/UtilsModel.h	35
7.7.1	Description détaillée	35
7.8	Référence du fichier src/ctrl/UtilsPoliceEcriture.h	35
7.8.1	Description détaillée	35
7.8.2	Documentation des fonctions	36
7.8.2.1	chargerPoliceEcriture	36
7.9	Référence du fichier src/ctrl/UtilsSDL.h	36
7.9.1	Description détaillée	36
7.9.2	Documentation des fonctions	36
7.9.2.1	arreterSDL	36
7.9.2.2	demarrerSDL	36
7.10	Référence du fichier src/model/Bateau.h	37
7.10.1	Description détaillée	38
7.10.2	Documentation du type de l'énumération	38
7.10.2.1	ESens	38
7.10.2.2	ETypeBat	38
7.10.3	Documentation des fonctions	38
7.10.3.1	creerBateau	38
7.10.3.2	estCoule	38
7.10.3.3	estPlacable	38
7.10.3.4	etatBateau	39
7.10.3.5	getBateauFromId	39
7.10.3.6	getIdBat	39
7.10.3.7	getPosBateau	39
7.10.3.8	getPosXBateau	40
7.10.3.9	getPosYBateau	40
7.10.3.10	getSensBateau	40
7.10.3.11	getTypeBateau	40
7.10.3.12	libererBateau	40
7.10.3.13	setPosBat	41
7.10.3.14	toucherBateau	41

7.11	Référence du fichier src/model/ChampSaisie.h	41
7.11.1	Description détaillée	42
7.11.2	Documentation du type de l'énumération	42
7.11.2.1	EtatChamp	42
7.11.3	Documentation des fonctions	42
7.11.3.1	ajouterCharFin	42
7.11.3.2	chainePleine	43
7.11.3.3	changeFocus	43
7.11.3.4	creerChamp	43
7.11.3.5	initTexte	43
7.11.3.6	libererChamp	44
7.11.3.7	supprimerDernierChar	44
7.12	Référence du fichier src/model/Couleurs.h	44
7.12.1	Description détaillée	44
7.12.2	Documentation des fonctions	45
7.12.2.1	getChar	45
7.12.2.2	getColor	45
7.12.2.3	getCouleurFromNum	45
7.12.2.4	getNbCouleurs	45
7.12.2.5	getNom	45
7.12.2.6	getNumFromColor	45
7.12.2.7	lettreToCouleur	45
7.13	Référence du fichier src/model/Coups.h	46
7.13.1	Description détaillée	46
7.13.2	Documentation des fonctions	46
7.13.2.1	creerCoup	46
7.14	Référence du fichier src/model/Grille.h	46
7.14.1	Description détaillée	47
7.14.2	Documentation du type de l'énumération	48
7.14.2.1	EtatCase	48
7.14.3	Documentation des fonctions	48
7.14.3.1	consulter	48
7.14.3.2	creerGrille	48
7.14.3.3	effacerGrille	48
7.14.3.4	getIdBateauSurCase	49
7.14.3.5	getNbCol	49
7.14.3.6	getNbLin	49
7.14.3.7	libererGrille	49
7.14.3.8	setEtatCase	49
7.15	Référence du fichier src/model/Joueur.h	50

7.15.1	Description détaillée	50
7.15.2	Documentation des macros	50
7.15.2.1	KLGNOMJ	50
7.15.3	Documentation des fonctions	51
7.15.3.1	creerJoueur	51
7.15.3.2	getNomJoueur	51
7.15.3.3	getTypeJoueur	51
7.15.3.4	libererJoueur	51
7.16	Référence du fichier src/model/Parametre.h	51
7.16.1	Description détaillée	52
7.16.2	Documentation des fonctions	52
7.16.2.1	chargerParam	52
7.16.2.2	getBNom	53
7.16.2.3	getCouleur	53
7.16.2.4	getInfoBateau	53
7.16.2.5	getNbBat	53
7.16.2.6	getNBInstances	53
7.16.2.7	getNbInstancesType	54
7.16.2.8	getNumBat	54
7.16.2.9	getType	54
7.16.2.10	libererParam	54
7.16.2.11	memParam	55
7.16.2.12	newTParam	55
7.16.2.13	resetInfoBateau	55
7.16.2.14	retierInfoBateauxType	55
7.16.2.15	setlemeInfoBateauTParam	55
7.16.2.16	setInfoBateau	56
7.17	Référence du fichier src/model/Partie.h	56
7.17.1	Description détaillée	56
7.17.2	Documentation des fonctions	57
7.17.2.1	annulerDernierCoup	57
7.17.2.2	initialiser	57
7.17.2.3	jouerUnCoup	57
7.17.2.4	libererPartie	57
7.17.2.5	partie_JHumain	57
7.17.2.6	partieEstFinie	57
7.17.3	Documentation des variables	58
7.17.3.1	globalPartie	58
7.18	Référence du fichier src/model/PileCoup.h	58
7.18.1	Description détaillée	58

7.18.2	Documentation des fonctions	58
7.18.2.1	creerPile	58
7.18.2.2	longueurPile	59
7.19	Référence du fichier src/model/Random.h	59
7.19.1	Description détaillée	59
7.19.2	Documentation des fonctions	59
7.19.2.1	choixMotHasard	59
7.19.2.2	initRandom	59
7.19.2.3	nombreAleatoire	60
7.20	Référence du fichier src/model/Score.h	60
7.20.1	Description détaillée	60
7.20.2	Documentation des fonctions	60
7.20.2.1	creerScore	60
7.20.2.2	creerScoreP	61
7.20.2.3	getNomScore	61
7.20.2.4	getScore	61
7.20.2.5	libererScore	61
7.20.2.6	setNomScore	61
7.20.2.7	setScore	61
7.21	Référence du fichier src/model/SDLMsgBox.h	62
7.21.1	Description détaillée	62
7.22	Référence du fichier src/test/model/TestBateau.h	62
7.22.1	Description détaillée	63
7.22.2	Documentation des fonctions	63
7.22.2.1	testCreerBateau	63
7.22.2.2	testEstCoule	63
7.22.2.3	testToucheBateau	63
7.23	Référence du fichier src/test/model/TestParam.h	63
7.23.1	Description détaillée	64
7.24	Référence du fichier src/test/Test.h	64
7.24.1	Description détaillée	64
7.25	Référence du fichier src/test/view/TestVue.h	64
7.25.1	Description détaillée	64
7.26	Référence du fichier src/view/IncludeSDL.h	65
7.26.1	Description détaillée	65
7.26.2	Documentation des macros	65
7.26.2.1	FONT_REP	65
7.26.2.2	RESSOURCES_REP	65
7.27	Référence du fichier src/view/SDLButton.h	65
7.27.1	Description détaillée	66

7.27.2	Documentation des fonctions	66
7.27.2.1	afficherBouton	66
7.27.2.2	clicSurBouton	66
7.27.2.3	creerBouton	67
7.27.2.4	libererBouton	67
7.28	Référence du fichier src/view/SDLImage.h	67
7.28.1	Description détaillée	67
7.28.2	Documentation des fonctions	67
7.28.2.1	afficherImage	68
7.28.2.2	clicSurImage	68
7.28.2.3	creerImage	68
7.28.2.4	creerSDLImage	68
7.28.2.5	libererImage	68
7.29	Référence du fichier src/view/SDLRectangle.h	69
7.29.1	Description détaillée	69
7.29.2	Documentation des fonctions	69
7.29.2.1	afficherRectangle	69
7.29.2.2	clicSurRectangle	69
7.29.2.3	creerRectangle	69
7.29.2.4	incrCouleurRectangle	70
7.29.2.5	libererRectangle	70
7.30	Référence du fichier src/view/VueBateau.h	70
7.30.1	Description détaillée	70
7.31	Référence du fichier src/view/VueChampSaisie.h	70
7.31.1	Description détaillée	71
7.31.2	Documentation des fonctions	71
7.31.2.1	afficherChamp	71
7.31.2.2	clicSurChamp	71
7.31.2.3	editerChamp	71
7.32	Référence du fichier src/view/VueGrille.h	72
7.32.1	Description détaillée	72
7.32.2	Documentation des fonctions	72
7.32.2.1	afficherGrille	72
7.32.2.2	clicCaseGrille	72
7.32.2.3	clicDansGrille	73
7.32.2.4	updateGrille	73
7.33	Référence du fichier src/view/VueParam.h	73
7.33.1	Description détaillée	73
7.33.2	Documentation des fonctions	73
7.33.2.1	afficherParamTest	73

7.34 Référence du fichier src/view/VueRegles.h	74
7.34.1 Description détaillée	74

Chapitre 1

Projet Bataille-Navale

Bienvenue dans la Bataille-Navale !

Voici quelques informations qui pourraient vous être utiles :

- [Présentation du projet](#)
- [Informations d'installation et de compilation](#)

Chapitre 2

Installation et Compilation

2.1 Installation

2.1.1 Windows

Ce programme utilise les bibliothèques SDL 1.2.15, SDL_image et SDL_ttf.

Toutes les DLL nécessaires à Windows sont incluses. Et doivent être placées dans le même dossier que l'exécutable.

2.1.2 MacOS X

Le programme se présente sous la forme d'un fichier .app et du dossier des ressources. Les deux doivent être placés dans le même répertoire.

2.2 Compilation

2.2.1 Windows

Pour pouvoir compiler, vous devez télécharger des fichiers spécifiques.

Attention, procédure pour Code : :Blocks sur Windows :

– SDL :

1. **Télécharger SDL**
2. Extraire l'archive dans un dossier (SDL-1.2.15 par défaut)
3. Déplacez les fichiers de SDL-1.2.15\include\SDL dans SDL-1.2.15\include
4. Déplacez le dossier SDL-1.2.15 dans le répertoire d'installation de Code : :Blocks
5. Créez un projet SDL dans Code : :Blocks
6. Une fois à la fenêtre "Global Variable Editor", dans le champ "base", chercher le dossier <répertoire installation Code : :Blocks>\SDL-1.2.15
7. Ignorez les avertissements ou les fenêtres, continuer normalement.

– SDL_image :

1. **Télécharger SDL_image**
2. Dézippez tout
3. SDL_image.h va dans <répertoire installation Code : :Blocks>\SDL-1.2.15\include
4. SDL_image.lib va dans <répertoire installation Code : :Blocks>\SDL-1.2.15\lib

5. Dans Code : :Blocks allez dans les paramètres du linker
 6. Ajoutez le fichier SDL_image.lib
- SDL_ttf :
1. [Télécharger SDL_ttf](#)
 2. SDL_ttf.h va dans <répertoire installation Code : :Blocks>\SDL-1.2.15\include
 3. SDL_ttf.lib va dans <répertoire installation Code : :Blocks>\SDL-1.2.15\lib
 4. Ajoutez le .lib aux paramètres du linker comme pour SDL_image

Le programme a besoin de toutes les DLL fournies avec ce code source, elles devront également être fournies avec l'exécutable.

2.2.2 MacOS X

Procédure pour XCode :

À venir

Si vous rencontrez des difficultés, écrivez-moi : aurelienbertron[AT]gmail[DOT]com

Chapitre 3

Présentation Générale

Vous voici dans la Bataille-Navale, développée par deux étudiants en 1ère année de DUT Informatique à l'IUT de Blagnac. Nous allons essayer de vous présenter brièvement ce projet.

3.1 Comment Jouer ?

Avant de pouvoir vous éclater à couler les navires de l'ordinateur, vous aurez quelques actions à effectuer.

Le Menu principal vous propose plusieurs possibilités de jeu : Nouvelle Partie ou Chargement

3.1.1 Nouvelle Partie

Vous devez tout d'abord choisir combien de bateaux de chaque type vous aurez. Pour des raisons techniques, il n'est pas possible d'avoir plus de 6 bateaux de chaque type, mais vous pourrez tout autant vous amuser ! Il vous faudra ensuite cliquer sur le bouton "Plus de paramètres" pour choisir le nom et la couleur de chaque bateau. Cliquez sur le carré de couleur pour les faire défiler, vous ne pourrez laisser la couleur à Blanc et le nom à "Nom :".

De retour à l'écran de paramétrage, vous pourrez enregistrer ces paramètres pour pouvoir les réutiliser plus tard, ou les passer à vos amis. Les paramètres sont dans le répertoire ressources/ et paramUser.dat contient les paramètres que vous enregistrez. paramOrigin contient des paramètres par défaut qu'on vous déconseille de changer (ils permettent de jouer rapidement avec de beaux bateaux bien nommés). Vous pourrez d'ailleurs charger tels ou tels paramètres dans l'écran de paramétrage.

Vient ensuite le placement des bateaux. Vous devrez sélectionner chaque bateau en cliquant sur son nom et le placer dans la grille. Vous pouvez changer son sens en cliquant sur le bouton "Sens". Si tout se passe bien, le bouton "OK" devrait apparaître. Une fois un bateau placé, vous ne pourrez pas revenir en arrière. Une fois que tous les bateaux sont placés, vous êtes amenés à l'écran de jeu.

3.1.2 Charger une Partie

En sélectionnant cette option, vous serez redirigé directement sur l'écran de jeu avec les paramètres contenus dans le fichier (ressources/saves/partieUser.dat).

3.1.3 Écran de jeu

L'écran de jeu est sans doute la partie la plus intéressante du jeu. Vous avez à gauche votre grille et à droite celle de la machine. Votre mission sera de cliquer dans la grille de la machine pour lui porter des coups. Une croix bleue signifie "Manqué" et une croix rouge signifie "Touché". Lorsqu'un bateau est coulé, il est entièrement coloré en rouge foncé. À tout moment vous pouvez annuler le dernier coup en cliquant sur le bouton correspondant.

En appuyant sur Échap vous accédez au menu Pause. Le menu Pause vous permet d'enregistrer la partie à un instant donné pour pouvoir la reprendre plus tard. Vous pouvez également quitter la partie (Attention, toute évolution non sauvegardée sera perdue).

Lorsque la partie est terminée, vous ne pouvez plus jouer ni annuler un coup et êtes redirigé vers un écran de fin correspondant au résultat de la partie.

3.2 Explications générales

Ce programme a entièrement été développé en C à l'aide de la bibliothèque SDL. Il est donc en mode graphique fenêtré avec prise en charge du clavier et de la souris. L'utilisateur navigue dans le jeu au travers de différents écrans et à l'aide de plusieurs objets d'interface.

Les boutons par exemple sont représentés par des rectangles gris et sont cliquables ou non selon les circonstances. Souvent la touche Escape (Échap) est utilisable pour revenir à l'écran précédent.

3.3 Explications techniques

La majeure partie du temps passé l'est dans l'attente d'un événement (clavier ou souris). Ainsi l'affichage d'un écran va toujours se diviser en plusieurs phases :

1. Déclaration des variables locales
2. Affectation des variables locales
3. Entrée dans une boucle "infinie", affichage des éléments et attente d'un événement
4. Analyse de l'événement et sortie de la boucle (sinon on revient au point précédent)
5. Libération de la mémoire

Chapitre 4

Index des structures de données

4.1 Structures de données

Liste des structures de données avec une brève description :

CaseGrille	
Contient les informations d'une case	11
Cellule	11
ChampSaisie	
Champ de saisie	12
Coord	
Coordonnées dans la grille	13
Couleur	13
Coup	14
CPSPProcessSerNum	14
Grille	
Matrice	14
Image	
Outil de dessin d'image	15
Joueur	16
Ligne	
Tableau dynamique de cases	16
NSApplication	16
NSApplication(SDL_Missing_Methods)	16
NSApplication(SDLApplication)	16
NSString	17
Rectangle	
Outil de dessin de rectangle	17
Score	
Contient un score avec le nom du joueur	18
SDL_Bouton	
Outil de dessin de boutons	18
SDL_MsgBox	19
SDLMain	19
TBateau	
Caractéristiques du bateau	19
TInfoBateau	
Contient les informations sur un bateau	20
Tparam	
Les paramètres d'une partie	21
TPartie	
Structure reprÈsentant une partie	21

TPosition	
Position du bateau dans la grille	22
TSensBat	23
TtypeBat	
Types de bateaux et noms	23

Chapitre 5

Index des fichiers

5.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

SDLMain.h	??
src/ctrl/ EcransDivers.h	
Contrôleur Ecran Divers	25
src/ctrl/ FichierDebug.h	
Contrôleur Débogage	25
src/ctrl/ FichierMeilleursScores.h	
Controlleur Meilleurs Scores	27
src/ctrl/ FichierSauvRes.h	
Contrôleur du fichier de sauvegarde	29
src/ctrl/ Jeu.h	
Contrôleur des écrans de jeu	32
src/ctrl/ Menu.h	
Controleur des écrans du menu	34
src/ctrl/ UtilsModel.h	
Contrôleur Outils utiles au développement des modèles	35
src/ctrl/ UtilsPoliceEcriture.h	
Contrôleur Utilitaire Police Ecriture	35
src/ctrl/ UtilsSDL.h	
Contrôleur des utilitaires SDL	36
src/model/ Bateau.h	
Modèle Bateau	37
src/model/ ChampSaisie.h	
Modèle Champ Saisie	41
src/model/ Couleurs.h	
Modèle Couleurs	44
src/model/ Coups.h	
Modèle Coup	46
src/model/ Grille.h	
Modèle Grille	46
src/model/ Joueur.h	
Modèle Joueur	50
src/model/ Parametre.h	
Modèle Paramètres	51
src/model/ Partie.h	
Modèle Partie	56
src/model/ PileCoup.h	
Modèle Pile Coups	58

src/model/ Random.h	
Modèle Aléatoire Headers	59
src/model/ Score.h	
Modèle Score	60
src/model/ SDLMsgBox.h	
Modèle de boîtes de messages	62
src/test/ Test.h	
Test	64
src/test/model/ TestBateau.h	
Test Modèle Bateau	62
src/test/model/ TestParam.h	
Test Modèle Paramètres	63
src/test/view/ TestVue.h	
Test Vue	64
src/view/ IncludeSDL.h	
Vue Inclusion de la SDL	65
src/view/ SDLButton.h	
Vue des bouton SDL	65
src/view/ SDLImage.h	
Vue des images SDL	67
src/view/ SDLRectangle.h	
Vue Rectangle SDL	69
src/view/ VueBateau.h	
Vue affichage des bateaux	70
src/view/ VueChampSaisie.h	
Vue Champ Saisies	70
src/view/ VueGrille.h	
Vue Grille	72
src/view/ VueParam.h	
Vue Paramètres	73
src/view/ VueRegles.h	
Vue Regles	74
src/view/ VueSDLMsgBox.h	??
src/view/ VueUtilsSDL.h	??

Chapitre 6

Documentation des structures de données

6.1 Référence de la structure CaseGrille

Contient les informations d'une case.

```
#include <Grille.h>
```

Champs de données

- [EtatCase](#) `etatCase`
- `int` [couleur](#)
- `int` [idBateauOccupe](#)

6.1.1 Description détaillée

Contient les informations d'une case.

6.1.2 Documentation des champs

6.1.2.1 `int` [couleur](#)

Numéro de la couleur dans la table des couleurs

6.1.2.2 [EtatCase](#) `etatCase`

État de la case

6.1.2.3 `int` [idBateauOccupe](#)

Id du bateau qui occupe la case.

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

- `src/model/`[Grille.h](#)

6.2 Référence de la structure Cellule

Champs de données

- [Coup](#) * [Info](#)

– struct [Cellule](#) * [Lien](#)

6.2.1 Documentation des champs

6.2.1.1 Coup* Info

Pointeur vers coups

6.2.1.2 struct [Cellule](#)* [Lien](#)

Pointeur vers la cellule suivante

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

– src/model/[PileCoup.h](#)

6.3 Référence de la structure ChampSaisie

Champ de saisie.

```
#include <ChampSaisie.h>
```

Champs de données

- char * [chaine](#)
- int [longMax](#)
- int [tailleTexte](#)
- int [largCarac](#)
- int [abscisse](#)
- int [ordonnee](#)
- [EtatChamp](#) [onFocus](#)

6.3.1 Description détaillée

Champ de saisie.

6.3.2 Documentation des champs

6.3.2.1 int [abscisse](#)

Abscisse du champ dans l'écran

6.3.2.2 char* [chaine](#)

Chaine tapée dans le champ

6.3.2.3 int [longMax](#)

Longueur maximum de la chaine

6.3.2.4 [EtatChamp](#) [onFocus](#)

Vaut CHAMP_ACTIF si le champ est actif (mode édition) et CHAMP_INACTIF sinon

6.3.2.5 int ordonnee

Ordonnee du champ dans l'écran

6.3.2.6 int tailleTexte

Taille de la police de texte

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

– src/model/[ChampSaisie.h](#)

6.4 Référence de la structure Coord

Coordonnées dans la grille.

```
#include <Grille.h>
```

Champs de données

- int [noLin](#)
- int [noCol](#)

6.4.1 Description détaillée

Coordonnées dans la grille.

Cette structure représente les coordonnées dans la grille. Il est important de différencier coordonnées dans la grille et coordonnées dans l'écran. Dans le cas de la bataille navale, les numéros de ligne sont normalement des lettres mais il n'est pas nécessaire de les traiter comme tels, car tout est transparent pour l'utilisateur (saisie à la souris)

6.4.2 Documentation des champs

6.4.2.1 int noCol

Numéro de colonne de la grille

6.4.2.2 int noLin

Numéro de ligne de la grille

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

– src/model/[Grille.h](#)

6.5 Référence de la structure Couleur

Champs de données

- char **lettre**
- char **nom** [KCOULEURS_LGNOMCOUL+1]
- SDL_Color **rgb**

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

– src/model/[Couleurs.h](#)

6.6 Référence de la structure Coup

Champs de données

- ETypeJoueur **type**
- **Coord** **coordTir**

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

- src/model/[Coups.h](#)

6.7 Référence de la structure CPSPProcessSerNum

Attributs protégés

- UInt32 **lo**
- UInt32 **hi**

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

- SDLMain.m

6.8 Référence de la structure Grille

Matrice.

```
#include <Grille.h>
```

Champs de données

- **Ligne** * **Matrice**
- int **NbLin**
- int **NbCol**
- int **abscisse**
- int **ordonnee**

6.8.1 Description détaillée

Matrice.

6.8.2 Documentation des champs

6.8.2.1 int abscisse

Abscisse de la grille à l'écran

6.8.2.2 **Ligne*** **Matrice**

Tableau dynamique de lignes

6.8.2.3 int NbCol

Nombre de colonnes de la matrice

6.8.2.4 int NbLin

Nombre de lignes de la matrice

6.8.2.5 int ordonnee

Ordonnée de la grille à l'écran

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

– src/model/[Grille.h](#)

6.9 Référence de la structure Image

Outil de dessin d'image.

```
#include <SDLImage.h>
```

Champs de données

- int [abscisse](#)
- int [ordonnee](#)
- int [longueur](#)
- int [hauteur](#)
- SDL_Surface * [zoneImage](#)

6.9.1 Description détaillée

Outil de dessin d'image.

6.9.2 Documentation des champs

6.9.2.1 int abscisse

Abscisse de l'image

6.9.2.2 int hauteur

Hauteur de l'image

6.9.2.3 int longueur

Longueur de l'image

6.9.2.4 int ordonnee

Ordonnée de l'image

6.9.2.5 SDL_Surface* zoneImage

Surface de stockage de l'image

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

– src/view/[SDLImage.h](#)

6.10 Référence de la structure Joueur

Champs de données

- ETypeJoueur **type**
- char **nomJ** [KLGNOMJ]
- **TBateau** ** **mesBateaux**

6.10.1 Documentation des champs

6.10.1.1 TBateau** mesBateaux

Nom du joueur

6.10.1.2 char nomJ[KLGNOMJ]

Type de joueur

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

- src/model/[Joueur.h](#)

6.11 Référence de la structure Ligne

Tableau dynamique de cases.

```
#include <Grille.h>
```

6.11.1 Description détaillée

Tableau dynamique de cases.

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

- src/model/[Grille.h](#)

6.12 Référence de la classe NSApplication

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

- SDLMain.m

6.13 Référence de la classe NSApplication(SDL_Missing_Methods)

Fonctions membres publiques

- (void) - **setAppleMenu** :

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

- SDLMain.m

6.14 Référence de la classe NSApplication(SDLApplication)

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

- SDLMain.m

6.15 Référence de la classe NSString

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

- SDLMain.m

6.16 Référence de la structure Rectangle

Outil de dessin de rectangle.

```
#include <SDLRectangle.h>
```

Champs de données

- int [abscisse](#)
- int [ordonnee](#)
- int [largeur](#)
- int [hauteur](#)
- int [couleur](#)
- SDL_Surface * [zoneRectangle](#)

6.16.1 Description détaillée

Outil de dessin de rectangle.

6.16.2 Documentation des champs

6.16.2.1 int abscisse

Abscisse du rectangle

6.16.2.2 int couleur

Indice de la couleur (voir table des couleurs)

6.16.2.3 int hauteur

Hauteur du rectangle

6.16.2.4 int largeur

Largeur du rectangle

6.16.2.5 int ordonnee

Ordonnée du rectangle

6.16.2.6 SDL_Surface* zoneRectangle

Surface de stockage du rectangle

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

- src/view/[SDLRectangle.h](#)

6.17 Référence de la structure Score

Contient un score avec le nom du joueur.

```
#include <Score.h>
```

Champs de données

- char [nom](#) [KLG NOMJ]
- int [score](#)

6.17.1 Description détaillée

Contient un score avec le nom du joueur.

6.17.2 Documentation des champs

6.17.2.1 char nom[KLG NOMJ]

Le nom du joueur qui à réalisé le score

6.17.2.2 int score

Le score du joueur

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

- src/model/[Score.h](#)

6.18 Référence de la structure SDL_Bouton

Outil de dessin de boutons.

```
#include <SDLButton.h>
```

Champs de données

- int [abscisse](#)
- int [ordonnee](#)
- char [texte](#) [SDL_BOUTON_KLONGMAX]
- int [tailleTexte](#)
- int [longTexte](#)
- int [largCarac](#)

6.18.1 Description détaillée

Outil de dessin de boutons.

6.18.2 Documentation des champs

6.18.2.1 int abscisse

Abscisse du bouton

6.18.2.2 int largCarac

Largeur d'un caractère

6.18.2.3 int longTexte

Longueur du texte

6.18.2.4 int ordonnee

Ordonnée du bouton

6.18.2.5 int tailleTexte

Taille du texte

6.18.2.6 char texte[SDL_BOUTON_KLONGMAX]

Texte du bouton

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

- src/view/[SDLButton.h](#)

6.19 Référence de la structure SDL_MsgBox

Champs de données

- int **abscisse**
- int **ordonnee**
- int **largeur**
- int **hauteur**
- char * **texte**

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

- src/model/[SDLMsgBox.h](#)

6.20 Référence de la classe SDLMain

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

- SDLMain.h

6.21 Référence de la structure TBateau

Caractéristiques du bateau.

```
#include <Bateau.h>
```

Champs de données

- int [idBateau](#)
- [TPosition](#) [position](#)
- [EEtat](#) [etat](#) [[KTAILLEMAXBAT](#)]
- int [estPlace](#)

6.21.1 Description détaillée

Caractéristiques du bateau.

6.21.2 Documentation des champs

6.21.2.1 int estPlace

Vaut 1 si le bateau est placé sur sa grille et 0 sinon

6.21.2.2 EEtat etat[KTAILLEMAXBAT]

Tableau d'État du bateau, renseigne l'État de chaque case du bateau

6.21.2.3 int idBateau

NumÈro du bateau (voir paramètres)

6.21.2.4 TPosition position

Position du bateau dans la grille

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

– src/model/[Bateau.h](#)

6.22 Référence de la structure TInfoBateau

Contient les informations sur un bateau.

```
#include <Joueur.h>
```

Champs de données

- int [couleur](#)
- [ETypeBat](#) type
- char [nomBateau](#) [K_LGNOM]

6.22.1 Description détaillée

Contient les informations sur un bateau.

Ces informations seront stockées dans un tableau (voir [Tparam](#))

6.22.2 Documentation des champs

6.22.2.1 int couleur

Indice dans la table des couleurs

6.22.2.2 char nomBateau[K_LGNOM]

Nom du bateau

6.22.2.3 ETypeBat type

Type du bateau

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

– src/model/[Parametre.h](#)

6.23 Référence de la structure Tparam

Les paramètres d'une partie.

```
#include <Parametre.h>
```

Champs de données

- int * [nombreInstanceBateaux](#)
- [TInfoBateau](#) * [bateauxJoueur](#)
- [TInfoBateau](#) * [bateauxMachine](#)

6.23.1 Description détaillée

Les paramètres d'une partie.

Ces paramètres seront ceux écrits dans un fichier.

6.23.2 Documentation des champs

6.23.2.1 TInfoBateau* bateauxJoueur

Tableau dynamique des bateaux du joueur. Il est à noter que les id déclarés sont fait dans par taille de bateau croissante.

6.23.2.2 TInfoBateau* bateauxMachine

Tableau dynamique des bateaux de la machine

6.23.2.3 int* nombreInstanceBateaux

Tableau d'entiers : Nombre d'instances de chaque type de bateau

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

– src/model/[Parametre.h](#)

6.24 Référence de la structure TPartie

Structure représentant une partie.

```
#include <Partie.h>
```

Champs de données

- [Joueur](#) * [joueur](#)
- [Joueur](#) * [machine](#)
- [Tparam](#) * [parametres](#)

- [Pile pileCoups](#)
- [Grille * grille](#)
- [Grille * grilleMachine](#)
- [int scorePlayer](#)

6.24.1 Description détaillée

Structure représentant une partie.

Définit le type Partie. Ce type correspond à une partie de bataille navale.

6.24.2 Documentation des champs

6.24.2.1 Grille* grille

[Grille](#) de l'humain

6.24.2.2 Grille* grilleMachine

[Grille](#) de l'IA

6.24.2.3 Joueur* joueur

[Joueur](#) humain

6.24.2.4 Joueur* machine

[Joueur](#) IA

6.24.2.5 Tparam* parametres

Paramètres de la partie

6.24.2.6 Pile pileCoups

Pile de coups

6.24.2.7 int scorePlayer

[Score](#) du joueur

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

- [src/model/Partie.h](#)

6.25 Référence de la structure TPosition

Position du bateau dans la grille.

```
#include <Bateau.h>
```

Champs de données

- ESens direction
- int x
- int y

6.25.1 Description détaillée

Position du bateau dans la grille.

6.25.2 Documentation des champs

6.25.2.1 ESens direction

Sens/Direction du bateau

6.25.2.2 int x

Num colonne dans la grille (point en haut et gauche du bateau)

6.25.2.3 int y

Num ligne dans la grille (point en haut et gauche du bateau)

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

- src/model/Bateau.h

6.26 Référence de la structure TSensBat

Champs de données

- ESens sensBat
- char * libSens

6.26.1 Documentation des champs

6.26.1.1 char* libSens

Libellé du sens

6.26.1.2 ESens sensBat

Sens du bateau

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

- src/model/Bateau.h

6.27 Référence de la structure TtypeBat

Types de bateaux et noms.

```
#include <Bateau.h>
```

Champs de données

- [ETypeBat typeBat](#)
- `char *` [nomType](#)

6.27.1 Description détaillée

Types de bateaux et noms.

DÉfinit le type du bateau et le nom du type (vise à remplacer ETypeBat)

6.27.2 Documentation des champs

6.27.2.1 `char* nomType`

Nom du type

6.27.2.2 `ETypeBat typeBat`

Type du bateau

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

- `src/model/`[Bateau.h](#)

Chapitre 7

Documentation des fichiers

7.1 Référence du fichier src/ctrl/EcransDivers.h

Contrôleur Ecran Divers.

Fonctions

- void **ecranVictoire** (void)
- void **ecranPerte** (void)

7.1.1 Description détaillée

Contrôleur Ecran Divers.

Auteur

Aurélien Bertron

Date

03 juin 2012 Contient les déclaration du module Debug. Ces fonction servent à afficher divers écrans de l'application (gagné, perdu,...)

7.2 Référence du fichier src/ctrl/FichierDebug.h

Contrôleur Débogage.

Fonctions

- int **init_debug** ()
- int **destruire_debug** ()
- int **dgSDL** (const char message[])
- int **dgInfo** (const char message[])
- int **dgAttention** (const char message[])
- int **dgErreur** (const char message[])
- int **dgFatal** (const char message[])
- int **debug** (const char prefixe[], const char message[])

7.2.1 Description détaillée

Contrôleur Débogage.

Auteur

Benoît Sauvère

Date

03 juin 2012 Contient les déclarations du module Debug. Ces fonction servent à écrire facilement des messages pour indiquer le déroulement de certaines opérations. Un fichier texte est généré au fur et à mesure.

7.2.2 Documentation des fonctions**7.2.2.1 int debug (const char *prefixe*[], const char *message*[])**

Insère dans le fichier de debug une entrée avec le prefixe designé.

Paramètres

<i>prefixe</i>	Le prefixe du message.
<i>message</i>	Le message a inserer.

Renvoie

0 si tout est OK. 1 sinon.

7.2.2.2 int detruire_debug ()

Termine le debug et enregistre dans le fichier

Renvoie

0 si tout est OK. 1 en cas d'erreur.

7.2.2.3 int dgAttention (const char *message*[])

Insère dans le fichier de debug une entrée de type "Attention".

Paramètres

<i>message</i>	Le message a inserer.
----------------	-----------------------

Renvoie

0 si tout est OK. 1 sinon.

7.2.2.4 int dgErreur (const char *message*[])

Insère dans le fichier de debug une entrée de type "Erreur".

Paramètres

<i>message</i>	Le message a inserer.
----------------	-----------------------

Renvoie

0 si tout est OK. 1 sinon.

7.2.2.5 int dgFatal (const char *message*[])

Insère dans le fichier de debug une entrée de type "Erreur fatale".

Paramètres

<i>message</i>	Le message à insérer.
----------------	-----------------------

Renvoie

0 si tout est OK. 1 sinon.

7.2.2.6 int dgInfo (const char *message*[])

Insère dans le fichier de debug une entrée de type "Information".

Paramètres

<i>message</i>	Le message à insérer.
----------------	-----------------------

Renvoie

0 si tout est OK. 1 sinon.

7.2.2.7 int dgSDL (const char *message*[])

Insère dans le fichier de debug une entrée de type "SDL".

Paramètres

<i>message</i>	Le message à insérer.
----------------	-----------------------

Renvoie

0 si tout est OK. 1 sinon.

7.2.2.8 int init_debug ()

Initialise les fonction de débogage.

Renvoie

0 si tout est OK. 1 si le fichier n'a pas pu être créé.

7.3 Référence du fichier src/ctrl/FichierMeilleursScores.h

Controlleur Meilleurs Scores.

```
#include "../model/Joueur.h"
#include "../model/Score.h"
#include <stdio.h>
```

Macros

- #define **FIC_MEILLEURSSCORES** "ressources/bestScores.dat"
- #define **MAX_MEILLEURS_SCORES** 10

Fonctions

- int [placeScoreTableau](#) ([Score](#) *tabScores[], [Score](#) *score)
- void [ajouterScore](#) ([Score](#) *score)
- [Score](#) ** [getMeilleursScoresFichier](#) ()
- FILE * [ouvrirFichierMeilleursScores](#) (const char modeOuverture[])
- void [enregistrerTabScore](#) ([Score](#) *tabScores[])

7.3.1 Description détaillée

Controlleur Meilleurs Scores.

Auteur

Benoît Sauvère

Date

19 juin 2012 Contient les déclarations des fonction de gestions des meilleurs scores.

7.3.2 Documentation des fonctions

7.3.2.1 void ajouterScore ([Score](#) * *score*)

Ajoute un score dans le fichier de meilleurs scores.

Paramètres

<i>in</i>	<i>score</i>	Un pointeur sur le score que l'on veut ajouter.
-----------	--------------	---

7.3.2.2 void enregistrerTabScore ([Score](#) * *tabScores*[])

Enregistre le tableau des scores dans le fichier des meilleurs scores.

Paramètres

<i>in</i>	<i>tabScores</i>	Le tableau de pointeur de Score à écrire
-----------	------------------	--

7.3.2.3 [Score](#)** [getMeilleursScoresFichier](#) ()

Récupère un tableau de scores de MAX_MEILLEURS_SCORES structures [Score](#).

Renvoie

Un tableau de pointeurs de scores de MAX_MEILLEURS_SCORES cases. Si il n'y a pas assez de cases, les cases vides seront initialisée à -1 et "".

7.3.2.4 FILE* [ouvrirFichierMeilleursScores](#) (const char *modeOuverture*[])

Ouvre le fichier contenant les meilleurs score en mode binaire.

Paramètres

<i>in</i>	<i>modeOuverture</i>	Le mode d'ouverture dans lequel on veut ouvrir le fichier des meilleurs scores
-----------	----------------------	--

Renvoie

Un fichier pointant sur le fichiers meilleurs scores ouvert en mode lecture/écriture/binaire.

7.3.2.5 int placeScoreTableau (Score * tabScores[], Score * score)

Détermine l'index d'un score dans un tableau de scores ;

Paramètres

<i>in</i>	<i>tabScore</i>	Un tableau de pointeurs de Score de MAX_MEILLEURS_SCORES cases.
<i>in</i>	<i>score</i>	Un pointeur sur le score dont on veut déterminer l'index dans le tableau des scores.

Renvoie

L'index du score dans tabScore. Retourne -1 si le score n'a pas sa place dans le tableau.

7.4 Référence du fichier src/ctrl/FichierSauvRes.h

Contrôleur du fichier de sauvegarde.

```
#include "../model/Partie.h"
```

Fonctions

- int [sauvegardePartie](#) (TPartie *partie, const char nomSauv[])
- int [sauvegardeBateaux](#) (TPartie *partie, FILE *fichier)
- int [sauvegardeGrille](#) (Grille *grille, FILE *fichier)
- int [sauvegardeCoups](#) (TPartie *partie, FILE *fichier)
- int [sauvegardeParam](#) (TPartie *partie, FILE *fichier)
- TPartie * [restaurerPartie](#) (const char nomSauv[])
- int [restaurerBateaux](#) (TPartie *partie, FILE *fichier)
- int [restaurerGrilles](#) (TPartie *partie, FILE *fichier)
- int [restaurerCoups](#) (TPartie *partie, FILE *fichier)
- int [restaurerParam](#) (TPartie *partie, FILE *fichier)

7.4.1 Description détaillée

Contrôleur du fichier de sauvegarde.

Auteur

Benoît Sauvère

Date

19 juin 2012 Contient les déclarations des fonctions utilisées pour la sauvegarde et la restauration d'une partie.

7.4.2 Documentation des fonctions

7.4.2.1 int restaurerBateaux (TPartie * partie, FILE * fichier)

Restaure les bateaux dans la structure de type [TPartie](#) à partir du flux fichier.

Paramètres

out	<i>partie</i>	La partie à modifier.
in	<i>fichier</i>	Le flux où lire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.2 int restaurerCoups (TPartie * *partie*, FILE * *fichier*)

Restaure les coups dans la pile de la structure de type TPartie à partir du flux fichier.

Paramètres

out	<i>partie</i>	La partie à modifier.
in	<i>fichier</i>	Le flux où lire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.3 int restaurerGrilles (TPartie * *partie*, FILE * *fichier*)

Restaure les grilles dans la structure TPartie passée en paramètre à partir du flux fichier.

Paramètres

out	<i>partie</i>	La partie à remplir
in	<i>fichier</i>	Le flux où lire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.4 int restaurerParam (TPartie * *partie*, FILE * *fichier*)

Restaure les paramètres dans la structure de type TPartie à partir du flux fichier.

Paramètres

out	<i>partie</i>	La partie à modifier.
in	<i>fichier</i>	Le flux où lire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.5 TPartie* restaurerPartie (const char *nomSauv*[])

Créer une partie à partir des données contenu dans le fichier de sauvegarde désigné.

Paramètres

in	<i>nomSauv</i>	Le nom du fichier de sauvegarde (dans le dossier saves)
----	----------------	---

Renvoie

Une structure de type **TPartie** avec les données du fichier de sauvegarde.

7.4.2.6 int sauvegardeBateaux (TPartie * partie, FILE * fichier)

Sauvegarde les bateaux de la partie (Voir la documentation pour la structure du fichier de sauvegarde)

Paramètres

in	<i>partie</i>	La partie contenant les bateaux à sauvegarder
in	<i>fichier</i>	Le flux où écrire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.7 int sauvegardeCoups (TPartie * partie, FILE * fichier)

Sauvegarde les coups de la pile de la partie (Voir la documentation pour la structure du fichier de sauvegarde)

Paramètres

in	<i>partie</i>	La partie contenant les coups à sauvegarder
in	<i>fichier</i>	Le flux où écrire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.8 int sauvegardeGrille (Grille * grille, FILE * fichier)

Sauvegarde les grilles de la partie (Voir la documentation pour la structure du fichier de sauvegarde)

Paramètres

in	<i>grille</i>	Un pointeur sur la grille à sauvegarder.
in	<i>fichier</i>	Le flux où écrire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.9 int sauvegardeParam (TPartie * partie, FILE * fichier)

Sauvegarde les paramètres de la partie (Voir la documentation pour la structure du fichier de sauvegarde)

Paramètres

in	<i>partie</i>	La partie contenant les paramètres à sauvegarder
in	<i>fichier</i>	Le flux où écrire les données

Renvoie

1 = pas d'erreur, autre = codeErreur

7.4.2.10 int sauvegardePartie (TPartie * partie, const char nomSauv[])

Sauvegarde la partie passée en paramètre dans le fichier désigné. (Voir la documentation pour la structure du fichier de sauvegarde)

Paramètres

in	partie	La partie à sauvegarder
in	nomSauv	Le nom du fichier (sauvegardé dans le fichier de sauvegarde)

Renvoie

1 = pas d'erreur, autre = codeErreur

7.5 Référence du fichier src/ctrl/Jeu.h

Contrôleur des écrans de jeu.

```
#include "../model/Parametre.h"
#include "../model/Joueur.h"
```

Fonctions

- int jeu (Tparam *pParam)
- int menuPlacementChoixBat (void)
- int menuPlacementGrille (TBateau *pBat)
- int ecranJeu (void)
- int menuPause (void)
- int changerSensBat (int pSensBat)
- int placementBatValide (Joueur *pJoueur)
- void placementAleatBat (Joueur *pJoueur, Grille *pGrille)
- Coord coordAleat (Grille *pGrille)

7.5.1 Description détaillée

Contrôleur des écrans de jeu.

Auteur

Aurélien Bertron

Date

28 avril 2012 Contient les déclaration des contrôleurs des écrans de jeu

7.5.2 Documentation des fonctions**7.5.2.1 int changerSensBat (int pSensBat)****Paramètres**

in	pSensBat	L'indice dans la table des sens de bateaux
----	----------	--

Renvoie

Le sens opposé à celui passé en paramètre (1 pour 0 et 0 pour 1) Inverse le sens d'un bateau

7.5.2.2 Coord coordAleat (Grille * pGrille)**Paramètres**

<i>in</i>	<i>pGrille</i>	Une grille
-----------	----------------	------------

Renvoie

Les coordonnées déterminées Détermine aléatoirement un couple de coordonnées dans la grille pGrille

7.5.2.3 int ecranJeu (void)**Renvoie**

L'état de la partie (voir fonction partieEstFinie) Ecran principal de jeu avec affichage des grilles, et gestions des actions du joueur (jouer un coup, annuler, mise en pause)

7.5.2.4 int jeu (Tparam * pParam)**Paramètres**

<i>in</i>	<i>pParam</i>	Les parametres de la partie, definis a l'ecran des parametres
-----------	---------------	---

Renvoie

-1 si la machine a gagne, 1 si l'humain a gagné et 0 sinon Initialise la partie et gere l'enchainement des ecrans de jeu

7.5.2.5 int menuPause (void)**Renvoie**

Le choix de l'utilisateur (1 :continuer, 2 :sauvegarder, 3 :quitter) Affiche le menu de pause

7.5.2.6 int menuPlacementChoixBat (void)**Renvoie**

1 si la partie est prete et 0 sinon Premier ecran du placement des bateaux avec liste des bateaux

7.5.2.7 int menuPlacementGrille (TBateau * pBat)**Paramètres**

<i>in, out</i>	<i>pBat</i>	Le bateau a placer
----------------	-------------	--------------------

Renvoie

1 si le bateau est bien place et 0 sinon Affiche la grille pour placer le bateau pBat

7.5.2.8 void placementAleatBat (Joueur * pJoueur, Grille * pGrille)

Paramètres

in, out	<i>pJoueur</i>	Un joueur
in, out	<i>pGrille</i>	La grille où placer les bateaux Place les bateaux d'un joueur pJoueur aléatoirement dans la grille pGrille

7.5.2.9 int placementBatValide (Joueur * pJoueur)

Paramètres

in	<i>pJoueur</i>	Un joueur
----	----------------	-----------

Renvoie

1 si valide et 0 sinon Détermine si tous les bateaux d'un joueur pJoueur sont bien placés

7.6 Référence du fichier src/ctrl/Menu.h

Controleur des écrans du menu.

```
#include "../model/ChampSaisie.h"
#include "../model/Parametre.h"
```

Fonctions

- void [afficherMenuAccueil](#) (void)
- int [afficherMenuRacine](#) (void)
- int [menuNouvellePartie](#) (Tparam *parametre)
- void [menuParam](#) (Tparam *parametre)

7.6.1 Description détaillée

Controleur des écrans du menu.

Auteur

Aurélien Bertron

Date

28 avril 2012 Contient les déclarations des controleurs des ecrans de menu

7.6.2 Documentation des fonctions

7.6.2.1 void afficherMenuAccueil (void)

Affiche le menu de presentation

7.6.2.2 int afficherMenuRacine (void)

Renvoie

Le choix du menu a charger Affiche le menu de choix. L'utilisateur a le choix entre plusieurs actions comme debuter une nouvelle partie, afficher les meilleurs scores, etc.

7.6.2.3 int menuNouvellePartie (Tparam * *parametre*)

Paramètres

in, out	<i>parametre</i>	Les parametres a modifier
---------	------------------	---------------------------

Renvoie

1 si la partie peut etre lancee et 0 si l'on doit retourner au menu precedent. Affiche le premier menu de saisie des parametres de la partie, avec la possibilite de charger des parametres

7.6.2.4 void menuParam (Tparam * *parametre*)

Paramètres

in, out	<i>parametre</i>	Les parametres a modifier Affiche le second menu de saisie des parametres, choix des noms et des couleurs des bateaux
---------	------------------	---

7.7 Référence du fichier src/ctrl/UtilsModel.h

Contrôleur Outils utiles au développement des modèles.

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

Fonctions

– FILE * **ouvrirFichierRessources** (const char *nomFic, const char *mode)

7.7.1 Description détaillée

Contrôleur Outils utiles au développement des modèles.

Auteur

Aurélien Bertron

Date

19 juin 2012 Controleur des outils utiles à la gestion des modèles.

7.8 Référence du fichier src/ctrl/UtilsPoliceEcriture.h

Contrôleur Utilitaire Police Ecriture.

```
#include "../view/IncludeSDL.h"
```

Fonctions

– TTF_Font * **chargerPoliceEcriture** (const char pChemin[], int pTailleEcriture)

7.8.1 Description détaillée

Contrôleur Utilitaire Police Ecriture.

Auteur

Benoit Sauvère

Date

03 juin 2012 Contient les déclarations pour le module des utilitaires de chargement des polices d'écriture.

7.8.2 Documentation des fonctions

7.8.2.1 TTF_Font* chargerPoliceEcriture (const char pChemin[], int pTailleEcriture)

Charge une police d'Écriture contenue dans le fichier des polices d'Écriture

Paramètres

<i>pChemin</i>	Le chemin de la police dans le dossier contenant les polices d'Écritures
<i>pTailleEcriture</i>	Un entier contenant la taille de la police.

Renvoie

Une structure TTF_Font contenant la police chargée.

7.9 Référence du fichier src/ctrl/UtilsSDL.h

Contrôleur des utilitaires SDL.

```
#include "../view/IncludeSDL.h"
```

Fonctions

- SDL_Surface * [demarrerSDL](#) (int width, int height, char *titreFenetre)
- void [arreterSDL](#) (void)

7.9.1 Description détaillée

Contrôleur des utilitaires SDL.

Auteur

Aurélien Bertron

Date

13 mai 2012 Les fonctions d'activation et de destruction de la librairie SDL.

7.9.2 Documentation des fonctions

7.9.2.1 void arreterSDL (void)

Arrête tous les modules SDL À ne pas oublier à la fin du programme

7.9.2.2 SDL_Surface* demarrerSDL (int width, int height, char * titreFenetre)

Permet d'initialiser tous les modules SDL Dans tout le code, la surface renvoyée par cette fonction peut-être retrouvée avec SDL_GetVideoSurface()

Paramètres

in	width	Largeur de la fenêtre
in	height	Hauteur de la fenêtre

Renvoi

La surface de l'écran

7.10 Référence du fichier src/model/Bateau.h

Modèle Bateau.

```
#include "../model/Grille.h"
```

Structures de données

- struct **TTypeBat**
Types de bateaux et noms.
- struct **TSensBat**
- struct **TPosition**
Position du bateau dans la grille.
- struct **TBateau**
Caractéristiques du bateau.

Macros

- #define **KTAILLEMAXBAT** 5
Taille maximale d'un bateau.
- #define **KLONGMAXNOMTYPE** 20
Longueur maximale du nom d'un type.

Énumérations

- enum **ETypeBat** {
 VOILIER = 1, **REMORQUEUR** = 2, **CARGOT** = 3, **SOUSMARIN** = 4,
 PORTEAVION = 5, **NONE** = 0 }
Types de bateaux.
- enum **ESens** { **HORIZONTAL**, **VERTICAL** }
Sens du bateau.
- enum **EEtat** { **INTACT**, **TOUCHE**, **COULE** }
État d'une case de bateau Est utilisÉ dans un tableau dont le nombre d'ElÉments est Égal à la taille du bateau.

Fonctions

- **TBateau** * **creerBateau** ()
- int **getIdBat** (**TBateau** *pBat)
RÉcupère l'id du bateau.
- **TBateau** * **getBateauFromId** (int idBateau)
RÉcupère un pointeur sur le bateau d'ÉsignÉ par l'id.
- void **toucherBateau** (**TBateau** *bat, int posTouch)
- int **etatBateau** (**TBateau** *bat)
- int **estCoule** (**TBateau** *bat)
- **TPosition** **getPosBateau** (**TBateau** *bat)
- int **getPosXBateau** (**TBateau** *bat)
- int **getPosYBateau** (**TBateau** *bat)
- **ESens** **getSensBateau** (**TBateau** *bat)
- **ETypeBat** **getTypeBateau** (**TBateau** *bat)
- void **setPosBat** (**TBateau** *pBat, **ESens** pSens, int pAbs, int pOrd)
- int **estPlacable** (**TBateau** *bat, **Grille** *grille)
- void **libererBateau** (**TBateau** *bat)

Variables

- const [TTypeBat](#) **tabTypesBat** [[KTAILLEMAXBAT](#)]
- const [TSensBat](#) **tabSensBat** [2]

7.10.1 Description détaillée

Modèle Bateau.

Auteur

Benoît Sauvère

Date

13 mai 2012 Contient les déclarations du module Bateau.

7.10.2 Documentation du type de l'énumération

7.10.2.1 enum ESens

Sens du bateau.

Définit le sens du bateau dans la grille

7.10.2.2 enum ETypeBat

Types de bateaux.

Définit à la fois le type et la taille du bateau

7.10.3 Documentation des fonctions

7.10.3.1 TBateau* creerBateau ()

Crée un bateau.

Renvoie

Une structure Bateau correctement initialisée et vide.

7.10.3.2 int estCoule (TBateau * bat)

Vérifie si le bateau est coulé.

Paramètres

<i>bat</i>	Un pointeur sur une structure TBateau
------------	---

Renvoie

1 si le bateau est coulé, 0 si il est en vie

7.10.3.3 int estPlacable (TBateau * bat, Grille * grille)

Determine si un bateau est placable ou non.

Paramètres

<i>in</i>	<i>bat</i>	Le bateau en question.
<i>in</i>	<i>grille</i>	La grille qui d'Êsire contenir le bateau.

Renvoie

1 si le bateau est plaÁable, 0 sinon.

7.10.3.4 int etatBateau (TBateau * bat)

Renvoie le nombre de coups necessaires pour couler le bateau (0 si le bateau est coulÉ). Utile ?

Paramètres

<i>bat</i>	un pointeur sur le bateau en question.
------------	--

Renvoie

Une valeur de l'ÉnumÉration EEtat.

7.10.3.5 TBateau* getBateauFromId (int idBateau)

RÉcupère un pointeur sur le bateau d'ÊsignÉ par l'id.

Paramètres

<i>in</i>	<i>idBateau</i>	L'id du bateau d'ÊsirÉ
-----------	-----------------	------------------------

Renvoie

Un pointeur sur le bateau d'ÊsirÉ

7.10.3.6 int getIdBat (TBateau * pBat)

RÉcupère l'id du bateau.

Paramètres

<i>in</i>	<i>pBat</i>	Le bateau voulu
-----------	-------------	-----------------

Renvoie

Le numero du bateau

7.10.3.7 TPosition getPosBateau (TBateau * bat)

Retourne une structure contenant la position d'un bateau.

Paramètres

<i>bat</i>	Un pointeur sur le bateau en question.
------------	--

Renvoie

Une structure de type [TPosition](#).

7.10.3.8 int getPosXBateau (TBateau * bat)

Retourne la position en X (latitude) d'un bateau.

Paramètres

<i>bat</i>	Un pointeur sur le bateau en question.
------------	--

Renvoie

un entier.

7.10.3.9 int getPosYBateau (TBateau * bat)

Retourne la position en Y (longituede) d'un bateau.

Paramètres

<i>bat</i>	Un pointeur sur le bateau en question.
------------	--

Renvoie

un entier.

7.10.3.10 ESens getSensBateau (TBateau * bat)

Retourne le sens d'un bateau.

Paramètres

<i>bat</i>	Un pointeur sur le bateau en question.
------------	--

Renvoie

Un Élément de l'enumération ESens.

7.10.3.11 ETypeBat getTypeBateau (TBateau * bat)

Retourne le type d'un bateau.

Paramètres

<i>bat</i>	Un pointeur sur le bateau en question.
------------	--

Renvoie

Un Élément de l'enumération ETypeBat.

7.10.3.12 void libererBateau (TBateau * bat)

Libère le bateau en mémoire

Paramètres

<i>in</i>	<i>bat</i>	Un pointeur sur le bateau
-----------	------------	---------------------------

7.10.3.13 void setPosBat (TBateau * *pBat*, ESens *pSens*, int *pAbs*, int *pOrd*)

Configure la position d'un bateau

Paramètres

in	<i>pBat</i>	Le bateau a configurer
in	<i>pSens</i>	Sens du bateau
in	<i>pAbs</i>	Abscisse du bateau
in	<i>pOrd</i>	Ordonnee du bateau

7.10.3.14 void toucherBateau (TBateau * *bat*, int *posTouch*)

Cette fonction marque une case d'un bateau donnée comme touchée.

Paramètres

	<i>bat</i>	Le pointeur sur le bateau en question
	<i>posTouch</i>	Le rang de la case touchée (cases de 1 à type).

7.11 Référence du fichier src/model/ChampSaisie.h

Modèle Champ Saisie.

Structures de données

- struct [ChampSaisie](#)
Champ de saisie.

Macros

- #define [KESP_VERT](#) 5
Espacement vertical du texte avec le bord du champ.
- #define [KESP_HORI](#) 5
Espacement horizontal du texte avec le bord du champ.
- #define [KCOULTXT_R](#) 0
Couleur du texte RGB R.
- #define [KCOULTXT_G](#) 0
Couleur du texte RGB G.
- #define [KCOULTXT_B](#) 0
Couleur du texte RGB B.
- #define [KCOULNORM_R](#) 200
Couleur du champ normal RGB R.
- #define [KCOULNORM_G](#) 207
Couleur du champ normal RGB G.
- #define [KCOULNORM_B](#) 212
Couleur du champ normal RGB B.
- #define [KCOULEDIT_R](#) 255
Couleur du champ édition RGB R.
- #define [KCOULEDIT_G](#) 255
Couleur du champ édition RGB G.
- #define [KCOULEDIT_B](#) 255
Couleur du champ édition RGB B.

Énumérations

- enum `EtatChamp` { `CHAMP_ACTIF`, `CHAMP_INACTIF` }
Etat d'un champ.

Fonctions

- `ChampSaisie` * `creerChamp` (int longMax, int taille, int abscisse, int ordonnee)
Initialise le champ.
- void `initTexte` (`ChampSaisie` *champ, const char *chaine)
Initialise le contenu du champ.
- int `chainePleine` (const `ChampSaisie` *champ)
Définit si le champ est plein.
- void `changeFocus` (`ChampSaisie` *champ, `EtatChamp` etat)
Change l'état du champ.
- char * `supprimerDernierChar` (char *chaine)
Supprime le dernier caractère d'une chaine.
- char * `ajouterCharFin` (char *chaine, char charEnt)
Ajoute un caractère à la fin de la chaine.
- void `libererChamp` (`ChampSaisie` *champ)
Libère le champ en mémoire.

7.11.1 Description détaillée

Modèle Champ Saisie.

Auteur

Aurélien Bertron

Date

29 avril 2012 Contient les types et en-têtes des fonctions du module de champs de saisie. Ce module implémente la gestion du champ de saisie. Il permet à l'utilisateur de saisir une chaîne de caractères.

7.11.2 Documentation du type de l'énumération

7.11.2.1 enum EtatChamp

Etat d'un champ.

Constantes définissant l'activation ou non du champ.

Valeurs énumérées :

CHAMP_ACTIF Champ en mode édition

CHAMP_INACTIF Champ en mode hors-édition

7.11.3 Documentation des fonctions

7.11.3.1 char* ajouterCharFin (char * chaine, char charEnt)

Ajoute un caractère à la fin de la chaîne.

Paramètres

in, out	chaine	
in	charEnt	Caractère à insérer

Renvoie

La chaine modifiée

La chaine ne doit pas être pleine.

7.11.3.2 int chainePleine (const ChampSaisie * champ)

Définit si le champ est plein.

Paramètres

in	champ	
----	-------	--

Renvoie

1 si le champ est plein et 0 sinon

Cette fonction teste si la chaine du champ n'est pas de la longueur maximale spécifiée à la création du champ.

7.11.3.3 void changeFocus (ChampSaisie * champ, EtatChamp etat)

Change l'état du champ.

Paramètres

in, out	champ	
in	etat	Nouvel état du champ

Met le champ à l'état spécifié (activé ou non)

7.11.3.4 ChampSaisie* creerChamp (int longMax, int taille, int abscisse, int ordonnee)

Initialise le champ.

Paramètres

in	longMax	Longueur maximale du champ
in	taille	Taille de la police
in	abscisse	Abscisse du champ dans l'écran
in	ordonnee	Ordonnée du champ dans l'écran

Renvoie

Un champ de saisie initialisé et NULL en cas d'erreur

Par défaut, le champ est inactif et vide.

7.11.3.5 void initTexte (ChampSaisie * champ, const char * chaine)

Initialise le contenu du champ.

Paramètres

in, out	champ	
in	chaine	Chaine à insérer dans le champ

Initialise le contenu du champ avec une chaine de caractères.

7.11.3.6 void libererChamp (ChampSaisie * champ)

Libère le champ en mémoire.

Paramètres

in	champ	Champ à libérer
----	-------	-----------------

À ne pas oublier à la fin du programme

7.11.3.7 char* supprimerDernierChar (char * chaine)

Supprime le dernier caractère d'une chaine.

Paramètres

in, out	chaine	
---------	--------	--

Renvoie

La chaine modifiée

7.12 Référence du fichier src/model/Couleurs.h

Modèle Couleurs.

```
#include "../view/IncludeSDL.h"
```

Structures de données

– struct [Couleur](#)

Macros

– #define **KCOULEURS_NBCOULMAX** 8
– #define **KCOULEURS_LGNOMCOUL** 10

Fonctions

– int [getNbCouleurs](#) (void)
– [Couleur lettreToCouleur](#) (char pLettre)
– [Couleur getCouleurFromNum](#) (int pl)
– SDL_Color [getColor](#) ([Couleur](#) pCouleur)
– char [getChar](#) ([Couleur](#) pCouleur)
– void [getNom](#) ([Couleur](#) pCouleur, char pNom[])
– int [getNumFromColor](#) ([Couleur](#) color)

Variables

– const [Couleur](#) **tableCouleurs** [KCOULEURS_NBCOULMAX]

7.12.1 Description détaillée

Modèle Couleurs.

Auteur

Aurélien Bertron, Benoît Sauvère

Date

18 avril 2012 Contient les déclaration du module Couleurs

7.12.2 Documentation des fonctions**7.12.2.1 char getChar (Couleur pCouleur)**

Fonction d'accès au champ lettre de la structure couleur.

7.12.2.2 SDL_Color getColor (Couleur pCouleur)

Fonction d'accès au champ rgb de la structure couleur.

7.12.2.3 Couleur getCouleurFromNum (int pl)

Retourne la couleur associée à l'indice pl dans le tableau des couleurs. Le tableau doit avoir au moins pl+1 éléments.

Paramètres

in	pl	L'indice de la couleur
----	----	------------------------

Renvoie

La couleur recherchée

7.12.2.4 int getNbCouleurs (void)**Renvoie**

Le nombre de couleurs gérées

7.12.2.5 void getNom (Couleur pCouleur, char pNom[])

Fonction d'accès au champ nom de la structure couleur.

7.12.2.6 int getNumFromColor (Couleur color)

Recupère l'index d'une couleur dans le tableau des couleurs depuis la structure color.

Paramètres

color	La couleur en question
-------	------------------------

Renvoie

L'index de la couleur dans le tableau des couleurs

7.12.2.7 Couleur lettreToCouleur (char pLettre)

Retourne la couleur correspondant à la lettre entrée. pLettre doit exister dans le tableau de couleurs.

Paramètres

<i>in</i>	<i>pLettre</i>	La lettre de la couleur
-----------	----------------	-------------------------

Renvoie

La couleur recherchée

7.13 Référence du fichier src/model/Coups.h

Modèle [Coup](#).

```
#include "../model/Joueur.h"
```

Structures de données

– struct [Coup](#)

Fonctions

– [Coup](#) * [creerCoup](#) (int *estJoueur*, [Coord](#) *pos*)

7.13.1 Description détaillée

Modèle [Coup](#).

Auteur

Benoît Sauvère

Date

19 juin 2012 Contient les déclarations du module [Coup](#), représentant un coup d'un joueur.

7.13.2 Documentation des fonctions

7.13.2.1 [Coup](#)* [creerCoup](#) (int *estJoueur*, [Coord](#) *pos*)

Créer une structure contenant les informations sur un tir.

Paramètres

<i>in</i>	<i>estJoueur</i>	1 = tir du joueur, sinon tir de la machine
<i>in</i>	<i>pos</i>	Les coordonnées du tir.

Renvoie

Un pointeur sur le coups crée.

7.14 Référence du fichier src/model/Grille.h

Modèle [Grille](#).

Structures de données

- struct [CaseGrille](#)
Contient les informations d'une case.
- struct [Grille](#)
Matrice.
- struct [Coord](#)
Coordonnées dans la grille.

Macros

- #define [KLARGGRILLE](#) 10
Largeur de la grille.
- #define [KHAUTGRILLE](#) 10
Hauteur de la grille.
- #define [KLARGCASE](#) 30
Largeur d'une case.
- #define [KHAUTEURCASE](#) 30
Hauteur d'une case.
- #define [KTAILLEPOLICE](#) 30
Taille de police.
- #define [KESP_CASE_VERT](#) 5
Espace vertical entre les cases.
- #define [KESP_CASE_HORI](#) 5
Espace horizontal entre les cases.
- #define [KIDCOULDEFAUT](#) 0
Indice de la couleur par défaut de la grille.

Définitions de type

- typedef [CaseGrille](#) * **Ligne**

Énumérations

- enum [EtatCase](#) { [GRILLE_CASE_NORMAL](#), [GRILLE_CASE_TOUCHE](#), [GRILLE_CASE_COULE](#), [GRILLE_CASE_EAU](#) }
Définit l'aspect de la case.

Fonctions

- [Grille](#) * [creerGrille](#) (int nbLin, int nbCol)
Constructeur de grille.
- int [getNbLin](#) ([Grille](#) *pGrille)
- int [getNbCol](#) ([Grille](#) *pGrille)
- [CaseGrille](#) * [consulter](#) ([Grille](#) *grille, [Coord](#) coord)
Récupère une case.
- int [getIdBateauSurCase](#) ([Grille](#) *grille, [Coord](#) coord)
Récupère l'id d'un bateau.
- [Grille](#) * [setEtatCase](#) ([Grille](#) *grille, [Coord](#) coord, [EtatCase](#) etat)
Définit l'état d'une case.
- [Grille](#) * [effacerGrille](#) ([Grille](#) *grille)
Efface la grille.
- void [libererGrille](#) ([Grille](#) *grille)
Libère la grille en mémoire.

7.14.1 Description détaillée

Modèle [Grille](#).

Auteur

Aurelien Bertron, Benoît Sauvère

Date

21 avril 2012 Contient les types et en-tetes des fonctions du module de grille

7.14.2 Documentation du type de l'énumération**7.14.2.1 enum EtatCase**

Definit l'aspect de la case.

Valeurs énumérées :

GRILLE_CASE_NORMAL Fond case normal
GRILLE_CASE_TOUCHE Ajout d'un signal "touche"
GRILLE_CASE_COULE Ajout d'un signal "coule"
GRILLE_CASE_EAU Ajout d'un signal "a l'eau"

7.14.3 Documentation des fonctions**7.14.3.1 CaseGrille* consulter (Grille * grille, Coord coord)**

Récupère une case.

Paramètres

<i>in</i>	<i>grille</i>	
<i>in</i>	<i>coord</i>	Coordonnées dans la grille de la case

Renvoie

La case aux coordonnées indiquées

Récupère une case de la grille. Attention, ligne et colonne doivent être cohérents avec les dimensions de la grille

7.14.3.2 Grille* creerGrille (int nbLin, int nbCol)

Constructeur de grille.

Paramètres

<i>in</i>	<i>nbLin</i>	Nombre de lignes de la grille
<i>in</i>	<i>nbCol</i>	Nombre de colonnes de la grille

Renvoie

Une grille initialisée ou NULL en cas d'erreur

Ce constructeur initialise une grille de nbLin lignes et nbCol colonnes. Attention, nbLin et nbCol doivent être strictement supérieurs à zéro.

7.14.3.3 Grille* effacerGrille (Grille * grille)

Efface la grille.

Paramètres

in	<i>grille</i>	
----	---------------	--

Renvoie

La grille modifiée

Efface la grille en la remettant à l'état normal Attention, la grille doit être initialisée

7.14.3.4 int getIdBateauSurCase (Grille * *grille*, Coord *coord*)

Récupère l'id d'un bateau.

Paramètres

in	<i>grille</i>	
in	<i>coord</i>	Coordonnées dans la grille de la case ou est le bateau

Renvoie

L'id du bateau qui occupe la case. -1 si pas de bateaux.

Récupère l'id du bateau occupant une case.

7.14.3.5 int getNbCol (Grille * *pGrille*)

Renvoie

Le nombre de colonnes de la grille

Paramètres

in	<i>pGrille</i>	
----	----------------	--

7.14.3.6 int getNbLin (Grille * *pGrille*)

Renvoie

Le nombre de lignes de la grille

Paramètres

in	<i>pGrille</i>	
----	----------------	--

7.14.3.7 void libererGrille (Grille * *grille*)

Libère la grille en mémoire.

Paramètres

in	<i>grille</i>	Attention, la grille doit être initialisée
----	---------------	--

7.14.3.8 Grille* setEtatCase (Grille * *grille*, Coord *coord*, EtatCase *etat*)

Définit l'état d'une case.

Paramètres

<i>in</i>	<i>grille</i>	
<i>in</i>	<i>coord</i>	Coordonnées de l'élément à insérer dans la grille
<i>in</i>	<i>etat</i>	État de la case à modifier

Renvoie

La grille modifiée

Définit l'état d'une case dans la grille. Attention la grille doit être initialisée

7.15 Référence du fichier src/model/Joueur.h

Modèle [Joueur](#).

```
#include "../model/Bateau.h"
```

Structures de données

– struct [Joueur](#)

Macros

– #define [KLGNO MJ](#) 25

Énumérations

– enum **ETypeJoueur** { **HUMAIN**, **MACHINE**, **UNDEF** }

Fonctions

– [Joueur](#) * [creerJoueur](#) (void)
– int [getTypeJoueur](#) (const [Joueur](#) *pJoueur)
– char * [getNomJoueur](#) ([Joueur](#) *pJoueur)
– void [libererJoueur](#) ([Joueur](#) *pJoueur)

7.15.1 Description détaillée

Modèle [Joueur](#).

Auteur

Aurélien Bertron, Benoît Sauvère

Date

18 avril 2012 Contient les déclaration du module [Joueur](#).

7.15.2 Documentation des macros

7.15.2.1 #define KLGNO MJ 25

Taille maximale du nom du nom d'un joueur

7.15.3 Documentation des fonctions

7.15.3.1 Joueur* creerJoueur (void)

Initialise un joueur

Renvoie

Un joueur initialisé

7.15.3.2 char* getNomJoueur (Joueur * pJoueur)

Récupère le nom d'un joueur

Paramètres

in	pJoueur	Le joueur voulu
----	---------	-----------------

Renvoie

Le nom de pJoueur

7.15.3.3 int getTypeJoueur (const Joueur * pJoueur)

Récupère le type d'un joueur

Paramètres

in	pJoueur	Le joueur voulu
----	---------	-----------------

Renvoie

Le type de pJoueur

7.15.3.4 void libererJoueur (Joueur * pJoueur)

Libère le joueur en mémoire

Paramètres

in	pJoueur	Le joueur à libérer
----	---------	---------------------

7.16 Référence du fichier src/model/Parametre.h

Modèle Paramètres.

```
#include "../model/Bateau.h"
#include "../model/Couleurs.h"
#include <stdio.h>
#include <stdlib.h>
```

Structures de données

- struct [TInfoBateau](#)
Contient les informations sur un bateau.
- struct [Tparam](#)

Les paramètres d'une partie.

Macros

- #define `K_NBTYPBATEAUX` `KTAILLEMAXBAT`
Nombre de types de bateaux.
- #define `K_LGNOM` 25
Longueur maximale du nom du bateau.

Fonctions

- int `getCouleur` (const `TInfoBateau` *pB)
RÉcupère la couleur du bateau.
- `ETypeBat` `getType` (const `TInfoBateau` *pB)
Donne le type de l'info bateau pB.
- void `getBNom` (const `TInfoBateau` *pB, char pNom[])
Donne le nom du bateau.
- void `setInfoBateau` (`TInfoBateau` *pB, char pNom[], int pCouleur, `ETypeBat` pType)
Affecte les infos pNom, pCouleur et pType à l'info bateau pB.
- `Tparam` * `newTParam` (int *pNbInstances)
Constructeur de Tparam.
- int * `getNBInstances` (const `Tparam` *pParam)
Donne le nombre d'instances de chaque bateau pour un joueur.
- int `getNumBat` (`ETypeBat` pTypeBat, int pNumBatType, `Tparam` *pParam)
- `TInfoBateau` * `getInfoBateau` (int pNum, const `Tparam` *pParam)
Donne les informations sur le pNum eme bateau des paramètres de la partie.
- int `getNbInstancesType` (const `Tparam` *pParam, `ETypeBat` pType)
Donne le nombre d'instances d'un type de bateau pour un joueur.
- int `getNbBat` (const `Tparam` *pParam)
Donne le nombre total de bateaux pour chaque joueur.
- void `resetInfoBateau` (`Tparam` *pParam)
- void `chargerParam` (FILE *pDesc, `Tparam` *pParam)
Lit les paramètres de la partie dans un descripteur de fichier pDesc.
- void `memParam` (const `Tparam` *pParam, FILE *pDesc)
Sauve les paramètres de la partie pParam dans un fichier.
- int `infoBateauValide` (const `Tparam` *pParam)
- void `libererParam` (`Tparam` *pParam)
- void `setlemInfoBateauTParam` (int pIdBateau, `Tparam` *pP, const char pNom[], int pCouleur, `ETypeBat` pType)
Configure un bateau selon son numÉro.
- void `retierInfoBateauxType` (`Tparam` *param, int nb, `ETypeBat` type)

7.16.1 Description détaillée

Modèle Paramètres.

Auteur

Benoît Sauvère, Aurélien Bertron

Date

19 mai 2012 Le module parametre d'une partie permet de charger et sauver ces paramètres dans un fichier.

7.16.2 Documentation des fonctions

7.16.2.1 void chargerParam (FILE * pDesc, Tparam * pParam)

Lit les paramètres de la partie dans un descripteur de fichier pDesc.

Paramètres

in, out	<i>pDesc</i>	Un descripteur de fichier
out	<i>pParam</i>	Les paramètres de la partie

pDesc doit être ouvert en lecture

7.16.2.2 void getBNom (const TInfoBateau * *pB*, char *pNom*[])

Donne le nom du bateau.

Paramètres

in	<i>pB</i>	Un pointeur sur les informations du bateau
out	<i>pNom</i>	Une chaîne de caractères contenant le nom

7.16.2.3 int getCouleur (const TInfoBateau * *pB*)

Récupère la couleur du bateau.

Paramètres

in	<i>pB</i>	Un pointeur sur les informations du bateau
----	-----------	--

Renvoie

Le numéro de la couleur

Retourne l'index de la couleur dans le tableau d'anglais.

7.16.2.4 TInfoBateau* getInfoBateau (int *pNum*, const Tparam * *pParam*)

Donne les informations sur le pNum eme bateau des parametres de la partie.

Paramètres

in	<i>pNum</i>	Le numéro du bateau
in	<i>pParam</i>	Les paramètres de la partie

Renvoie

Les informations du bateau

Les id des bateaux vont de 0 à m pour le joueur et de m+1 à n pour la machine Attention, pNum doit correspondre à un bateau existant

7.16.2.5 int getNbBat (const Tparam * *pParam*)

Donne le nombre total de bateaux pour chaque joueur.

Paramètres

in	<i>pParam</i>	Les paramètres de la partie
----	---------------	-----------------------------

Renvoie

Nombre total de bateaux pour chaque joueur

7.16.2.6 int* getNBInstances (const Tparam * *pParam*)

Donne le nombre d'instances de chaque bateau pour un joueur.

Paramètres

<i>in</i>	<i>pParam</i>	Les paramètres de la partie
-----------	---------------	-----------------------------

Renvoie

Un pointeur sur le premier élément d'un tableau d'entiers

7.16.2.7 int getNbInstancesType (const Tparam * pParam, ETypeBat pType)

Donne le nombre d'instances d'un type de bateau pour un joueur.

Paramètres

<i>in</i>	<i>pParam</i>	Les paramètres de la partie
<i>in</i>	<i>pType</i>	Le type de bateau

Renvoie

Un nombre

7.16.2.8 int getNumBat (ETypeBat pTypeBat, int pNumBatType, Tparam * pParam)

Retourne l'id qu'aurait un bateau avec ces caractéristiques

Paramètres

<i>in</i>	<i>pTypeBat</i>	Le type du bateau
<i>in</i>	<i>pNumBatType</i>	La position du bateau par rapport aux bateaux du même type
<i>in</i>	<i>pParam</i>	Les paramètres de la partie (contient les TInfoBateaux

Renvoie

Le numéro du bateau

7.16.2.9 ETypeBat getType (const TInfoBateau * pB)

Donne le type de l'info bateau pB.

Paramètres

<i>in</i>	<i>pB</i>	Un pointeur sur les informations du bateau
-----------	-----------	--

Renvoie

Le type du bateau

7.16.2.10 void libererParam (Tparam * param)

Détruit la structure paramètre ainsi que toutes les ressources qu'elle contient.

Paramètres

<i>param</i>	Le paramètre à libérer
--------------	------------------------

7.16.2.11 void memParam (const Tparam * pParam, FILE * pDesc)

Sauve les paramètres de la partie pParam dans un fichier.

Paramètres

in	pParam	Les paramètres de la partie
in, out	pDesc	Un descripteur de fichier

pDesc est un descripteur de fichier ouvert en écriture

7.16.2.12 Tparam* newTParam (int * pNbInstances)

Constructeur de [Tparam](#).

Paramètres

in	pNbInstances	Nombre d'instances de bateaux pour chaque type (tableau dynamique d'entiers)
----	--------------	--

Renvoie

Des paramètres initialisés

Attention, ‡ appeler avant toute manipulation de paramètres (même chargerParam)

7.16.2.13 void resetInfoBateau (Tparam * pParam)

Configure le nom de tous les bateaux à "Nom :" et la couleur à Blanc

Paramètres

in, out	pParam	Les paramètres de la partie
---------	--------	-----------------------------

7.16.2.14 void retierInfoBateauxType (Tparam * param, int nb, ETypeBat type)

Retire nb bateau(x) de la liste d'un type donné.

Paramètres

param	La structure parametre ‡ modifier
nb	Le nombre de TInfoBateau du type ‡ supprimer
type	Le type de bateau ‡ supprimer

7.16.2.15 void setlemInfoBateauTParam (int pIdBateau, Tparam * pP, const char pNom[], int pCouleur, ETypeBat pType)

Configure un bateau selon son numéro.

Paramètres

in	pIdBateau	Le numéro du bateau voulu
out	pP	Les paramètres de la partie
in	pNom	Le nom du bateau
in	pCouleur	La couleur du bateau
in	pType	Le type du bateau

7.16.2.16 void setInfoBateau (TInfoBateau * *pB*, char *pNom*[], int *pCouleur*, ETypeBat *pType*)

Affecte les infos *pNom*, *pCouleur* et *pType* à l'info bateau *pB*.

Paramètres

out	<i>pB</i>	Un pointeur sur les informations du bateau
in	<i>pNom</i>	Le nom du bateau
in	<i>pCouleur</i>	La couleur du bateau
in	<i>pType</i>	Le type du bateau

7.17 Référence du fichier src/model/Partie.h

Modèle Partie.

```
#include "../model/Joueur.h"
#include "../model/Parametre.h"
#include "../model/PileCoup.h"
#include "../model/Grille.h"
```

Structures de données

- struct TPartie
Structure reprÉsantant une partie.

Fonctions

- Joueur * partie_JHumain ()
- Joueur * partie_JMachine ()
- Tparam * partie_Param ()
- Pile partie_PileCoups ()
- Grille * partie_Grille ()
- Grille * partie_GrilleMachine ()
- int partie_Score ()
- TPartie * initialiser (Tparam *param)
- int jouerUnCoup (TPartie *partie, Coord cible, int estJoueur)
- int partieEstFinie (TPartie *partie)
- void annulerDernierCoup (TPartie *partie)
- void libererPartie (TPartie *partie)

Variables

- TPartie * globalPartie

7.17.1 Description détaillée

Modèle Partie.

Auteur

Benoît Sauvère

Date

19 juin 2012 Contient les déclarations du module de gestion de la partie.

7.17.2 Documentation des fonctions

7.17.2.1 void annulerDernierCoup (TPartie * *partie*)

Annule le dernier coups jouÉ (aussi bien par la machine que par le joueur).

Paramètres

out	<i>partie</i>	La partie dans laquelle on veut annuler le dernier coups.
-----	---------------	---

7.17.2.2 TPartie* initialiser (Tparam * *param*)

Cette fonction prÉpare une structure pour qu'elle soit jouable.

Paramètres

in	<i>param</i>	Les paramÈtres à appliquer à la partie
----	--------------	--

Renvoie

Un pointeur sur la partie prÉparÉE

7.17.2.3 int jouerUnCoup (TPartie * *partie*, Coord *cible*, int *estJoueur*)

Cette fonction rÉalise un tir (aussi bien pour la machine que pour le joueur).

Paramètres

in	<i>partie</i>	La partie concernÉE.
in	<i>cible</i>	Les coordonnÉes oÙ l'on tire.
in	<i>estJoueur</i>	BoolÉen indiquant si c'est un coup pour le joueur ou non.

Renvoie

Retourne le rÉsultat de l'action (1 = touchÉ, 0 = ratÉ)

7.17.2.4 void libererPartie (TPartie * *partie*)

LibÈre les ressources liÉes à la partie.

Paramètres

in	<i>partie</i>	La partie à libÈrer.
----	---------------	----------------------

7.17.2.5 Joueur* partie_JHumain ()

Getters de la structure [TPartie](#)

7.17.2.6 int partieEstFinie (TPartie * *partie*)

DÉtermine si la partie est finie ou non et indique un Éventuel vainqueur.

Paramètres

in	<i>partie</i>	La partie en question
----	---------------	-----------------------

Renvoie

0 = partie toujours en cours, 1 = le joueur \neq gagnÉ, -1 = la machine \neq gagnÉ

7.17.3 Documentation des variables

7.17.3.1 TPartie* globalPartie

Variable globale contenant les données de la partie. Cette variable contient l'ensemble des données de la partie et peut-être utilisée depuis n'importe quelle partie du programme.

7.18 Référence du fichier src/model/PileCoup.h

Modèle Pile Coups.

```
#include "../model/Grille.h"
#include "../model/Joueur.h"
#include "../model/Coups.h"
```

Structures de données

– struct [Cellule](#)

Définitions de type

– typedef struct [Cellule](#) * **Pile**

Fonctions

- [Pile](#) **creerPile** (void)
- int **pileVide** ([Pile](#) pPile)
- [Pile](#) **empiler** ([Pile](#) pPile, [Coup](#) *pElem)
- [Pile](#) **depiler** ([Pile](#) pPile)
- [Coup](#) * **sommet** ([Pile](#) pPile)
- int **longueurPile** ([Pile](#) pile)

7.18.1 Description détaillée

Modèle Pile Coups.

Auteur

Aurélien Bertron

Date

21 mai 2012 Contient les déclarations du module de pile de coups.

7.18.2 Documentation des fonctions

7.18.2.1 Pile creerPile (void)

Initialise une pile

Renvoie

Retourne une pile initialisee

7.18.2.2 int longueurPile (Pile pile)

Détermine la longueur de la pile.

Paramètres

in	<i>pile</i>	Un strucutre Pile
----	-------------	-------------------

Renvoie

Un entier contenant la longueur de la pile.

7.19 Référence du fichier src/model/Random.h

Modèle Aléatoire Headers.

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

Fonctions

- void [initRandom](#) (void)
- int [nombreAleatoire](#) (int pNbMin, int pNbMax)
- void [choixMotHasard](#) (char *pMot, FILE *pDesc, int longMax)

7.19.1 Description détaillée

Modèle Aléatoire Headers.

Auteur

Aurélien Bertron

Date

10 juin 2012 Contient les déclarations du module d'aléatoire

7.19.2 Documentation des fonctions**7.19.2.1 void choixMotHasard (char * pMot, FILE * pDesc, int longMax)**

Choisit au hasard une ligne dans un fichier Le fichier doit être ouvert en lecture

Paramètres

out	<i>pMot</i>	Le mot pioché
in, out	<i>pDesc</i>	Le descripteur du fichier ouvert en lecture
in	<i>longMax</i>	Longueur maximale d'une ligne du fichier

7.19.2.2 void initRandom (void)

Initialise le générateur de nombres aléatoires À n'appeler qu'une seule fois en début de programme

7.19.2.3 int nombreAleatoire (int *pNbMin*, int *pNbMax*)

Génère un nombre aléatoirement dans un intervalle donné

Paramètres

in	<i>pNbMin</i>	Borne inférieure de l'intervalle
in	<i>pNbMax</i>	Borne supérieure de l'intervalle

Renvoie

Le nombre généré

7.20 Référence du fichier src/model/Score.h

Modèle [Score](#).

```
#include "../model/joueur.h"
```

Structures de données

- struct [Score](#)
Contient un score avec le nom du joueur.

Fonctions

- [Score](#) * [creerScore](#) ()
- [Score](#) * [creerScoreP](#) (int score, const char nom[])
- int [getScore](#) ([Score](#) *score)
- char * [getNomScore](#) ([Score](#) *score)
- void [setScore](#) ([Score](#) *score, int nouvScore)
- void [setNomScore](#) ([Score](#) *score, const char nouvNom[])
- void [libererScore](#) ([Score](#) *score)

7.20.1 Description détaillée

Modèle [Score](#).

Auteur

Benoît Sauvère

Date

14 juin 2012 Contient les déclarations de gestion des structures [Score](#).

7.20.2 Documentation des fonctions

7.20.2.1 [Score](#)* [creerScore](#) ()

Constructeur du module [Score](#).

Renvoie

Un pointeur sur un score correctement alloué.

7.20.2.2 Score* creerScoreP (int score, const char nom[])

Constructeur du module [Score](#) avec paramétrage.

Paramètres

in	score	Valeur du score
in	nom	Nom du joueur

Renvoie

Un pointeur sur un score correctement alloué et initialisé.

7.20.2.3 char* getNomScore (Score * score)

Retourne le nom du joueur qui à réalisé le score.

Paramètres

in	score	Un pointeur sur le score dont on veut le nom du joueur.
----	-------	---

7.20.2.4 int getScore (Score * score)

Retourne le score d'un joueur.

Paramètres

in	score	Un pointeur sur le score dont on veut la valeur.
----	-------	--

7.20.2.5 void libererScore (Score * score)

Libère les ressources liées à un score.

Paramètres

in	score	Un pointeur sur le score à libérer.
----	-------	-------------------------------------

7.20.2.6 void setNomScore (Score * score, const char nouvNom[])

Modifie le score d'une structure score.

Paramètres

out	score	Un pointeur sur le score dont on veut modifier le score.
in	nouvNom	Une chaîne de caractères contenant le nouveau nom.

7.20.2.7 void setScore (Score * score, int nouvScore)

Modifie le score d'une structure score.

Paramètres

out	score	Un pointeur sur le score dont on veut modifier le score.
in	nouvScore	La nouvelle valeur du score.

7.21 Référence du fichier src/model/SDLMsgBox.h

Modèle de boîtes de messages.

Structures de données

- struct `SDL_MsgBox`

Macros

- #define `KPADDING` 10

Fonctions

- `SDL_MsgBox * creerMsgBox` (int pAbs, int pOrd, int pLarg, int pHaut)
- void `setMsg` (`SDL_MsgBox *pMBox`, char *pTexte)
- int `nbLinMsg` (char *pTexte)
- int `longLinMax` (char *pTexte)
- void `libererMsgBox` (`SDL_MsgBox *pMBox`)

7.21.1 Description détaillée

Modèle de boîtes de messages. Vue utilitaires SDL.

Vue des boîtes de messages.

Auteur

Aurélien Bertron

Date

19 juin 2012 Contient les déclarations de gestion du modèle de boites de messages.

Auteur

Aurélien Bertron

Date

19 juin 2012 Contient les déclarations des fonctions utilisées pour l’affichage des MsgBox.

Auteur

Aurélien Bertron

Date

19 juin 2012 Contient les déclarations de fonctions utilisées pour simplifier l’utilisation de la SDL.

7.22 Référence du fichier src/test/model/TestBateau.h

Test Modèle Bateau.

```
#include "../model/Bateau.h"
```

Fonctions

- int testEstCoule ()
- int testCreerBateau ()
- int testToucheBateau ()

7.22.1 Description détaillée

Test Modèle Bateau.

Auteur

Benoît Sauvère

Date

13 mai 2012 Contient les déclarations des fonctions de test du modèle Bateau.

7.22.2 Documentation des fonctions

7.22.2.1 int testCreerBateau ()

Test unitaire de la fonction estCoule.

Renvoie

1 si tout les test sont passÉ. 0 Si echec.

7.22.2.2 int testEstCoule ()

Test unitaire de la fonction estCoule.

Renvoie

1 si tout les test sont passÉ. 0 Si echec.

7.22.2.3 int testToucheBateau ()

Test unitaire de la fonction estCoule.

Renvoie

1 si tout les test sont passÉ. 0 Si echec.

Test unitaire de la fonction toucheBateau.

Renvoie

1 si tout les test sont passÉ. 0 Si echec.

7.23 Référence du fichier src/test/model/TestParam.h

Test Modèle Paramètres.

```
#include "../model/Parametre.h"
```

Fonctions

- void **testParam** (void)
- void **controleurParametreVersionTest** (Tparam *param)

7.23.1 Description détaillée

Test Modèle Paramètres.

Auteur

Benoît Sauvère

Date

13 mai 2012 Contient les déclarations des fonctions de test du module Paramètres

7.24 Référence du fichier src/test/Test.h

Test.

Fonctions

- void **menuTest** (void)

7.24.1 Description détaillée

Test.

Auteur

Benoît Sauvère

Date

13 mai 2012 Contient les déclarations du module central de tests.

7.25 Référence du fichier src/test/view/TestVue.h

Test Vue.

Fonctions

- void **menuTestVue** (void)

7.25.1 Description détaillée

Test Vue.

Auteur

Benoît Sauvere

Date

13 mai 2012 Contient les déclarations des fonctions de tests de la vue.

7.26 Référence du fichier src/view/IncludeSDL.h

Vue Inclusion de la SDL.

```
#include <SDL/SDL.h>
#include <SDL_image/SDL_image.h>
#include <SDL_ttf/SDL_ttf.h>
```

Macros

- #define **RESSOURCES_REP** "ressources/"
- #define **FONT_REP** "ressources/Fonts/"
- #define **IMG_REP** "ressources/Images/"

7.26.1 Description détaillée

Vue Inclusion de la SDL.

Auteur

Aurélien Bertron

Date

19 juin 2012 Contient les inclusions nécessaires à l'utilisation de la librairie SDL, ainsi que les chemins vers plusieurs dossiers de ressources.

7.26.2 Documentation des macros

7.26.2.1 #define FONT_REP "ressources/Fonts/"

Définis où se trouvent les images du programme

7.26.2.2 #define RESSOURCES_REP "ressources/"

<Définis où se trouvent les ressources du programme Définis où se trouvent les polices d'écriture du programme

7.27 Référence du fichier src/view/SDLButton.h

Vue des bouton SDL.

```
#include "../view/IncludeSDL.h"
#include "../view/VueUtilsSDL.h"
```

Structures de données

- struct **SDL_Bouton**
Outil de dessin de boutons.

Macros

- #define **SDL_BOUTON_KLONGMAX** 30
Longueur maximum du texte du bouton.

- #define `SDL_BOUTON_KESP_VERT` 5
Espacement vertical du texte avec le bord du bouton.
- #define `SDL_BOUTON_KESP_HORI` 15
Espacement horizontal du texte avec le bord du bouton.
- #define `SDL_BOUTON_KCOULTXT_R` 0
Couleur du texte RGB R.
- #define `SDL_BOUTON_KCOULTXT_G` 0
Couleur du texte RGB G.
- #define `SDL_BOUTON_KCOULTXT_B` 0
Couleur du texte RGB B.
- #define `SDL_BOUTON_KCOUL_R` 200
Couleur du bouton normal RGB R.
- #define `SDL_BOUTON_KCOUL_G` 207
Couleur du bouton normal RGB G.
- #define `SDL_BOUTON_KCOUL_B` 212
Couleur du bouton normal RGB B.

Fonctions

- `SDL_Bouton` * `creerBouton` (char *pTexte, SDL_Rect *pCoord, int pTailleTexte)
- void `afficherBouton` (`SDL_Bouton` *pBouton)
- int `clicSurBouton` (`SDL_Bouton` *pBouton, SDL_Rect *positionClic)
- void `libererBouton` (`SDL_Bouton` *pBouton)

7.27.1 Description détaillée

Vue des bouton SDL.

Auteur

Aurélien Bertron

Date

28 mai 2012 Contient les déclarations du module Bouton SDL

7.27.2 Documentation des fonctions

7.27.2.1 void afficherBouton (`SDL_Bouton` * *pBouton*)

Affiche un bouton

Paramètres

in	<i>pBouton</i>	Un bouton
----	----------------	-----------

7.27.2.2 int clicSurBouton (`SDL_Bouton` * *pBouton*, SDL_Rect * *positionClic*)

Détermine si un clic est sur le bouton ou non

Paramètres

in	<i>pBouton</i>	Un bouton
in	<i>positionClic</i>	La position du clic

Renvoie

1 si le clic est sur le bouton et 0 sinon

7.27.2.3 **SDL_Bouton*** creerBouton (char * *pTexte*, SDL_Rect * *pCoord*, int *pTailleTexte*)

Crée un bouton.

Paramètres

<i>pTexte</i>	Le texte du bouton.
<i>pCoord</i>	Un pointeur sur les coordonnÉes
<i>pTailleTexte</i>	Un entier contenant la taille du texte.

Renvoie

Une strucutre [SDL_Bouton](#) correctement initialisÉe.

7.27.2.4 void libererBouton (SDL_Bouton * *pBouton*)

Libère le bouton en mémoire

Paramètres

in	<i>pBouton</i>	Un bouton
----	----------------	-----------

7.28 Référence du fichier src/view/SDLImage.h

Vue des images SDL.

```
#include "../view/IncludeSDL.h"
```

Structures de données

- struct [Image](#)
Outil de dessin d'image.

Fonctions

- [Image](#) * [creerImage](#) (char *pChemin, int pAbscisse, int pOrdonnee)
- void [afficherImage](#) ([Image](#) *pImage)
- int [clicSurImage](#) ([Image](#) *pImage, SDL_Rect *pPosClic)
- void [libererImage](#) ([Image](#) *pImage)
- SDL_Surface SDLCALL * [creerSDLImage](#) (char chemin[])

7.28.1 Description détaillée

Vue des images SDL.

Auteur

Aurélien Bertron

Date

29 mai 2012 Contient les déclarations du module [Image](#) SDL

7.28.2 Documentation des fonctions

7.28.2.1 void afficherImage (Image * *pImage*)

Affiche une image

Paramètres

<i>in</i>	<i>pImage</i>	Une image
-----------	---------------	-----------

7.28.2.2 int clicSurImage (Image * *pImage*, SDL_Rect * *pPosClic*)

Détermine si un clic est sur l'image ou non

Paramètres

<i>in</i>	<i>pImage</i>	Une image
<i>in</i>	<i>pPosClic</i>	La position du clic

Renvoie

1 si le clic est sur l'image et 0 sinon

7.28.2.3 Image* creerImage (char * *pChemin*, int *pAbscisse*, int *pOrdonnee*)

Crée une structure [Image](#).

Paramètres

<i>pChemin</i>	Le chemin de l'image à l'intérieur du dossier image
<i>pAbscisse</i>	Un entier contenant l'abscisse où placer l'image
<i>pOrdonnee</i>	Un entier contenant l'ordonnée où placer l'image

Renvoie

Une structure [Image](#) correctement créée.

7.28.2.4 SDL_Surface SDLCALL* creerSDLImage (char *chemin*[])

Cette fonction charge une image du dossier [Image](#) dans une structure SDL_Surface

Paramètres

<i>pChemin</i>	Le chemin de l'image à l'intérieur du dossier image
----------------	---

Renvoie

Une structure SDL_Surface correctement créée.

7.28.2.5 void libererImage (Image * *pImage*)

Libère l'image en mémoire

Paramètres

<i>in</i>	<i>pImage</i>	Une image
-----------	---------------	-----------

7.29 Référence du fichier src/view/SDLRectangle.h

Vue [Rectangle](#) SDL.

```
#include "../view/IncludeSDL.h"
```

Structures de données

- struct [Rectangle](#)
Outil de dessin de rectangle.

Fonctions

- [Rectangle](#) * [creerRectangle](#) (int pAbs, int pOrd, int pLarg, int pHaut)
- void [afficherRectangle](#) ([Rectangle](#) *pRect)
- int [clicSurRectangle](#) ([Rectangle](#) *pRect, SDL_Rect *pPosClic)
- void [incrCouleurRectangle](#) ([Rectangle](#) *pRect)
- void [libererRectangle](#) ([Rectangle](#) *pRect)

7.29.1 Description détaillée

Vue [Rectangle](#) SDL.

Auteur

Aurélien Bertron

Date

7 juin 2012 Contient les déclarations du module [Rectangle](#) SDL

7.29.2 Documentation des fonctions

7.29.2.1 void afficherRectangle ([Rectangle](#) * *pRect*)

Afficher le rectangle

Paramètres

in	<i>pRect</i>	Un rectangle
----	--------------	--------------

7.29.2.2 int clicSurRectangle ([Rectangle](#) * *pRect*, SDL_Rect * *pPosClic*)

Détermine si un clic est sur le rectangle ou non

Paramètres

in	<i>pRect</i>	Un rectangle
in	<i>pPosClic</i>	La position du clic

Renvoie

1 si le clic est sur le rectangle et 0 sinon

7.29.2.3 [Rectangle](#)* [creerRectangle](#) (int *pAbs*, int *pOrd*, int *pLarg*, int *pHaut*)

Crée un rectangle de couleur blanche

Paramètres

<i>in</i>	<i>pAbs</i>	Abscisse du rectangle
<i>in</i>	<i>pOrd</i>	Ordonnée du rectangle
<i>in</i>	<i>pLarg</i>	Largeur du rectangle
<i>in</i>	<i>pHaut</i>	Hauteur du rectangle

Renvoie

Un rectangle initialisé

7.29.2.4 void incrCouleurRectangle (Rectangle * *pRect*)

Incrémente la couleur du rectangle (voir table des couleurs)

Paramètres

<i>in</i>	<i>pRect</i>	Un rectangle
-----------	--------------	--------------

7.29.2.5 void libererRectangle (Rectangle * *pRect*)

Libère le rectangle en mémoire

Paramètres

<i>in</i>	<i>pRect</i>	Un rectangle
-----------	--------------	--------------

7.30 Référence du fichier src/view/VueBateau.h

Vue affichage des bateaux.

```
#include "../model/Grille.h"
#include "../model/Bateau.h"
```

Fonctions

– *Grille* * **insertBateau** (*Grille* *grille, *TBateau* *bat)

7.30.1 Description détaillée

Vue affichage des bateaux.

Auteur

Aurélien Bertron

Date

19 juin 2012 Contient les déclarations des fonctions utilisées pour la sauvegarde et la restauration d'une partie.

7.31 Référence du fichier src/view/VueChampSaisie.h

Vue Champ Saisies.

```
#include "../view/IncludeSDL.h"
#include "../model/ChampSaisie.h"
```

Fonctions

- void **afficherChamp** (**ChampSaisie** *champ)
Affiche le champ.
- void **editerChamp** (**ChampSaisie** *champ)
Passe le champ en mode édition.
- int **clicSurChamp** (**ChampSaisie** *champ, SDL_Rect *positionClic)
Détermine si le clic est sur le champ.

7.31.1 Description détaillée

Vue Champ Saisies.

Auteur

Aurélien Bertron

Date

29 avril 2012 Contient les en-têtes des fonctions d'entrée/sortie du module de champs de saisie. Ce module implémente la gestion du champ de saisie. Il permet à l'utilisateur de saisir une chaîne de caractères.

7.31.2 Documentation des fonctions

7.31.2.1 void afficherChamp (ChampSaisie * champ)

Affiche le champ.

Paramètres

in	champ	Affiche le champ aux coordonnées données lors de la création.
----	-------	---

7.31.2.2 int clicSurChamp (ChampSaisie * champ, SDL_Rect * positionClic)

Détermine si le clic est sur le champ.

Paramètres

in	champ	
in	positionClic	Coordonnées du clic dans l'écran

Renvoie

1 si le clic est sur le champ et 0 sinon

7.31.2.3 void editerChamp (ChampSaisie * champ)

Passe le champ en mode édition.

Paramètres

in	champ	Passe le champ en mode édition (possibilité d'ajouter des caractères et d'en supprimer). Pour sortir du mode édition, il est nécessaire de cliquer hors du champ.
----	-------	---

7.32 Référence du fichier src/view/VueGrille.h

Vue [Grille](#).

```
#include "../view/IncludeSDL.h"
#include "../model/Grille.h"
```

Fonctions

- void [afficherGrille](#) ([Grille](#) *grille, int abscisse, int ordonnee)
Affiche la grille à l'écran.
- void [updateGrille](#) ([Grille](#) *grille, [Coord](#) coord)
Met à jour la grille.
- [Coord clicCaseGrille](#) ([Grille](#) *grille, [SDL_Rect](#) *positionClic)
Coordonnées d'un clic dans la grille.
- int [clicDansGrille](#) ([Grille](#) *grille, [SDL_Rect](#) *positionClic)
Détermine si un clic est dans la grille.

7.32.1 Description détaillée

Vue [Grille](#).

Auteur

Aurélien Bertron

Date

21 avril 2012 Contient les en-têtes des fonctions d'entrée-sortie du module de grille

7.32.2 Documentation des fonctions

7.32.2.1 void [afficherGrille](#) ([Grille](#) * grille, int abscisse, int ordonnee)

Affiche la grille à l'écran.

Paramètres

in	<i>grille</i>	Grille à afficher
in	<i>abscisse</i>	Abscisse de la grille sur l'écran
in	<i>ordonnee</i>	Ordonnée de la grille sur l'écran

Affiche la grille à l'écran aux abscisse et ordonnee données. Attention la grille doit être initialisée.

7.32.2.2 [Coord clicCaseGrille](#) ([Grille](#) * grille, [SDL_Rect](#) * positionClic)

Coordonnées d'un clic dans la grille.

Paramètres

in	<i>grille</i>	Grille cliquée
in	<i>positionClic</i>	Position du clic sur l'écran (type défini par la SDL)

Renvoie

Les coordonnées du clic dans la grille si le clic est dans la grille et des coordonnées nulles sinon

7.32.2.3 int clicDansGrille (Grille * grille, SDL_Rect * positionClic)

Détermine si un clic est dans la grille.

Paramètres

in	grille	Grille où l'on cherche le clic
in	positionClic	Position du clic sur l'écran (type défini par la SDL)

Renvoie

1 si le clic est dans la grille et 0 sinon

7.32.2.4 void updateGrille (Grille * grille, Coord coord)

Met à jour la grille.

Paramètres

in	grille	Grille à mettre à jour
in	coord	Coordonnées de la case à mettre à jour

Met à jour l'affichage de la grille à une case donnée. Attention la grille doit être initialisée et les coordonnées doivent correspondre aux dimensions de la grille

7.33 Référence du fichier src/view/VueParam.h

Vue Paramètres.

```
#include "../model/Parametre.h"
```

Fonctions

- void [afficherParamTest](#) (Tparam *param)
Affiche les paramètres dans la version de test (en ligne de commande)

7.33.1 Description détaillée

Vue Paramètres.

Auteur

Aurélien Bertron

Date

19 mai 2012 Ce module permet de gérer l'affichage/saisie des paramètres d'une partie

7.33.2 Documentation des fonctions

7.33.2.1 void afficherParamTest (Tparam * param)

Affiche les paramètres dans la version de test (en ligne de commande)

Paramètres

<i>in</i>	<i>param</i>	Les paramètres de la partie
-----------	--------------	-----------------------------

7.34 Référence du fichier src/view/VueRegles.h

Vue Regles.

Fonctions

– void **afficherRegles** (void)

7.34.1 Description détaillée

Vue Regles.

Auteur

Aurélien Bertron

Date

19 juin 2012 Contient les déclarations du module d’affichage des règles

Index

abscisse
 ChampSaisie, 12
 Grille, 14
 Image, 15
 Rectangle, 17
 SDL_Bouton, 18

afficherBouton
 SDLButton.h, 66

afficherChamp
 VueChampSaisie.h, 71

afficherGrille
 VueGrille.h, 72

afficherImage
 SDLImage.h, 67

afficherMenuAccueil
 Menu.h, 34

afficherMenuRacine
 Menu.h, 34

afficherParamTest
 VueParam.h, 73

afficherRectangle
 SDLRectangle.h, 69

ajouterCharFin
 ChampSaisie.h, 42

ajouterScore
 FichierMeilleursScores.h, 28

annulerDernierCoup
 Partie.h, 57

arreterSDL
 UtilsSDL.h, 36

Bateau.h
 creerBateau, 38
 ESens, 38
 ETypeBat, 38
 estCoule, 38
 estPlacable, 38
 etatBateau, 39
 getBateauFromId, 39
 getIdBat, 39
 getPosBateau, 39
 getPosXBateau, 39
 getPosYBateau, 40
 getSensBateau, 40
 getTypeBateau, 40
 libererBateau, 40
 setPosBat, 40
 toucherBateau, 41

bateauxJoueur
 Tparam, 21

bateauxMachine
 Tparam, 21

CHAMP_ACTIF
 ChampSaisie.h, 42

CHAMP_INACTIF
 ChampSaisie.h, 42

CPSPProcessSerNum, 14

CaseGrille, 11
 couleur, 11
 etatCase, 11
 idBateauOccupe, 11

Cellule, 11
 Info, 12
 Lien, 12

chaîne
 ChampSaisie, 12

chaînePleine
 ChampSaisie.h, 43

ChampSaisie.h
 CHAMP_ACTIF, 42
 CHAMP_INACTIF, 42

ChampSaisie, 12
 abscisse, 12
 chaîne, 12
 longMax, 12
 onFocus, 12
 ordonnee, 12
 tailleTexte, 13

ChampSaisie.h
 ajouterCharFin, 42
 chaînePleine, 43
 changeFocus, 43
 creerChamp, 43
 EtatChamp, 42
 initTexte, 43
 libererChamp, 43
 supprimerDernierChar, 44

changeFocus
 ChampSaisie.h, 43

changerSensBat
 Jeu.h, 32

chargerParam
 Parametre.h, 52

chargerPoliceEcriture
 UtilsPoliceEcriture.h, 36

choixMotHasard
 Random.h, 59

clicCaseGrille
 VueGrille.h, 72

- clicDansGrille
 - VueGrille.h, 72
- clicSurBouton
 - SDLButton.h, 66
- clicSurChamp
 - VueChampSaisie.h, 71
- clicSurImage
 - SDLImage.h, 68
- clicSurRectangle
 - SDLRectangle.h, 69
- consulter
 - Grille.h, 48
- Coord, 13
 - noCol, 13
 - noLin, 13
- coordAleat
 - Jeu.h, 33
- Couleur, 13
- couleur
 - CaseGrille, 11
 - Rectangle, 17
 - TInfoBateau, 20
- Couleurs.h
 - getChar, 45
 - getColor, 45
 - getCouleurFromNum, 45
 - getNbCouleurs, 45
 - getNom, 45
 - getNumFromColor, 45
 - lettreToCouleur, 45
- Coup, 14
- Coups.h
 - creerCoup, 46
- creerBateau
 - Bateau.h, 38
- creerBouton
 - SDLButton.h, 66
- creerChamp
 - ChampSaisie.h, 43
- creerCoup
 - Coups.h, 46
- creerGrille
 - Grille.h, 48
- creerImage
 - SDLImage.h, 68
- creerJoueur
 - Joueur.h, 51
- creerPile
 - PileCoup.h, 58
- creerRectangle
 - SDLRectangle.h, 69
- creerSDLImage
 - SDLImage.h, 68
- creerScore
 - Score.h, 60
- creerScoreP
 - Score.h, 60
- debug
 - FichierDebug.h, 26
- demarrerSDL
 - UtilsSDL.h, 36
- detruire_debug
 - FichierDebug.h, 26
- dgAttention
 - FichierDebug.h, 26
- dgErreur
 - FichierDebug.h, 26
- dgFatal
 - FichierDebug.h, 27
- dgInfo
 - FichierDebug.h, 27
- dgSDL
 - FichierDebug.h, 27
- direction
 - TPosition, 23
- ESens
 - Bateau.h, 38
- ETypeBat
 - Bateau.h, 38
- ecranJeu
 - Jeu.h, 33
- editerChamp
 - VueChampSaisie.h, 71
- effacerGrille
 - Grille.h, 48
- enregistrerTabScore
 - FichierMeilleursScores.h, 28
- estCoule
 - Bateau.h, 38
- estPlacable
 - Bateau.h, 38
- estPlace
 - TBateau, 20
- etat
 - TBateau, 20
- etatBateau
 - Bateau.h, 39
- EtatCase
 - Grille.h, 48
- etatCase
 - CaseGrille, 11
- EtatChamp
 - ChampSaisie.h, 42
- FONT_REP
 - IncludeSDL.h, 65
- FichierDebug.h
 - debug, 26
 - detruire_debug, 26
 - dgAttention, 26
 - dgErreur, 26
 - dgFatal, 27
 - dgInfo, 27
 - dgSDL, 27
 - init_debug, 27
- FichierMeilleursScores.h

- ajouterScore, 28
- enregistrerTabScore, 28
- getMeilleursScoresFichier, 28
- ouvrirFichierMeilleursScores, 28
- placeScoreTableau, 29
- FichierSauvRes.h
 - restaurerBateaux, 29
 - restaurerCoups, 30
 - restaurerGrilles, 30
 - restaurerParam, 30
 - restaurerPartie, 30
 - sauvegardeBateaux, 31
 - sauvegardeCoups, 31
 - sauvegardeGrille, 31
 - sauvegardeParam, 31
 - sauvegardePartie, 32
- GRILLE_CASE_COULE
 - Grille.h, 48
- GRILLE_CASE_EAU
 - Grille.h, 48
- GRILLE_CASE_NORMAL
 - Grille.h, 48
- GRILLE_CASE_TOUCHE
 - Grille.h, 48
- getBNom
 - Parametre.h, 53
- getBateauFromId
 - Bateau.h, 39
- getChar
 - Couleurs.h, 45
- getColor
 - Couleurs.h, 45
- getCouleur
 - Parametre.h, 53
- getCouleurFromNum
 - Couleurs.h, 45
- getIdBat
 - Bateau.h, 39
- getIdBateauSurCase
 - Grille.h, 49
- getInfoBateau
 - Parametre.h, 53
- getMeilleursScoresFichier
 - FichierMeilleursScores.h, 28
- getNBInstances
 - Parametre.h, 53
- getNbBat
 - Parametre.h, 53
- getNbCol
 - Grille.h, 49
- getNbCouleurs
 - Couleurs.h, 45
- getNbInstancesType
 - Parametre.h, 54
- getNbLin
 - Grille.h, 49
- getNom
 - Couleurs.h, 45
- getNomJoueur
 - Joueur.h, 51
- getNomScore
 - Score.h, 61
- getNumBat
 - Parametre.h, 54
- getNumFromColor
 - Couleurs.h, 45
- getPosBateau
 - Bateau.h, 39
- getPosXBateau
 - Bateau.h, 39
- getPosYBateau
 - Bateau.h, 40
- getScore
 - Score.h, 61
- getSensBateau
 - Bateau.h, 40
- getType
 - Parametre.h, 54
- getTypeBateau
 - Bateau.h, 40
- getTypeJoueur
 - Joueur.h, 51
- globalPartie
 - Partie.h, 58
- Grille, 14
 - abscisse, 14
 - Matrice, 14
 - NbCol, 14
 - NbLin, 14
 - ordonnee, 15
- grille
 - TPartie, 22
- Grille.h
 - GRILLE_CASE_COULE, 48
 - GRILLE_CASE_EAU, 48
 - GRILLE_CASE_NORMAL, 48
 - GRILLE_CASE_TOUCHE, 48
- Grille.h
 - consulter, 48
 - creerGrille, 48
 - effacerGrille, 48
 - EtatCase, 48
 - getIdBateauSurCase, 49
 - getNbCol, 49
 - getNbLin, 49
 - libererGrille, 49
 - setEtatCase, 49
- grilleMachine
 - TPartie, 22
- hauteur
 - Image, 15
 - Rectangle, 17
- idBateau
 - TBateau, 20
- idBateauOccupe

- CaseGrille, 11
- Image, 15
 - abscisse, 15
 - hauteur, 15
 - longueur, 15
 - ordonnee, 15
 - zoneImage, 15
- IncludeSDL.h
 - FONT_REP, 65
 - RESSOURCES_REP, 65
- incrCouleurRectangle
 - SDLRectangle.h, 70
- Info
 - Cellule, 12
- init_debug
 - FichierDebug.h, 27
- initRandom
 - Random.h, 59
- initTexte
 - ChampSaisie.h, 43
- initialiser
 - Partie.h, 57
- jeu
 - Jeu.h, 33
- Jeu.h
 - changerSensBat, 32
 - coordAleat, 33
 - ecranJeu, 33
 - jeu, 33
 - menuPause, 33
 - menuPlacementChoixBat, 33
 - menuPlacementGrille, 33
 - placementAleatBat, 33
 - placementBatValide, 34
- jouerUnCoup
 - Partie.h, 57
- Joueur, 16
 - mesBateaux, 16
 - nomJ, 16
- joueur
 - TPartie, 22
- Joueur.h
 - creerJoueur, 51
 - getNomJoueur, 51
 - getTypeJoueur, 51
 - KLGNOMJ, 50
 - libererJoueur, 51
- KLGNOMJ
 - Joueur.h, 50
- largCarac
 - SDL_Bouton, 18
- largeur
 - Rectangle, 17
- lettreToCouleur
 - Couleurs.h, 45
- libSens
 - TSensBat, 23
- libererBateau
 - Bateau.h, 40
- libererBouton
 - SDLButton.h, 67
- libererChamp
 - ChampSaisie.h, 43
- libererGrille
 - Grille.h, 49
- libererImage
 - SDLImage.h, 68
- libererJoueur
 - Joueur.h, 51
- libererParam
 - Parametre.h, 54
- libererPartie
 - Partie.h, 57
- libererRectangle
 - SDLRectangle.h, 70
- libererScore
 - Score.h, 61
- Lien
 - Cellule, 12
- Ligne, 16
- longMax
 - ChampSaisie, 12
- longTexte
 - SDL_Bouton, 19
- longueur
 - Image, 15
- longueurPile
 - PileCoup.h, 59
- machine
 - TPartie, 22
- Matrice
 - Grille, 14
- memParam
 - Parametre.h, 54
- Menu.h
 - afficherMenuAccueil, 34
 - afficherMenuRacine, 34
 - menuNouvellePartie, 34
 - menuParam, 35
- menuNouvellePartie
 - Menu.h, 34
- menuParam
 - Menu.h, 35
- menuPause
 - Jeu.h, 33
- menuPlacementChoixBat
 - Jeu.h, 33
- menuPlacementGrille
 - Jeu.h, 33
- mesBateaux
 - Joueur, 16
- NSApplication, 16
- NSApplication(SDL_Missing_Methods), 16

NSApplication(SDLApplication), 16
NSString, 17
NbCol
 Grille, 14
NbLin
 Grille, 14
newTParam
 Parametre.h, 55
noCol
 Coord, 13
noLin
 Coord, 13
nom
 Score, 18
nomBateau
 TInfoBateau, 20
nomJ
 Joueur, 16
nomType
 TtypeBat, 24
nombreAleatoire
 Random.h, 59
nombreInstanceBateaux
 Tparam, 21

onFocus
 ChampSaisie, 12
ordonnee
 ChampSaisie, 12
 Grille, 15
 Image, 15
 Rectangle, 17
 SDL_Bouton, 19
ouvrirFichierMeilleursScores
 FichierMeilleursScores.h, 28

Parametre.h
 chargerParam, 52
 getBNom, 53
 getCouleur, 53
 getInfoBateau, 53
 getNBInstances, 53
 getNbBat, 53
 getNbInstancesType, 54
 getNumBat, 54
 getType, 54
 libererParam, 54
 memParam, 54
 newTParam, 55
 resetInfoBateau, 55
 retierInfoBateauxType, 55
 setlemInfoBateauTParam, 55
 setInfoBateau, 55
parametres
 TPartie, 22
Partie.h
 annulerDernierCoup, 57
 globalPartie, 58
 initialiser, 57
 jouerUnCoup, 57
 libererPartie, 57
 partie_JHumain, 57
 partieEstFinie, 57
partie_JHumain
 Partie.h, 57
partieEstFinie
 Partie.h, 57
PileCoup.h
 creerPile, 58
 longueurPile, 59
pileCoups
 TPartie, 22
placeScoreTableau
 FichierMeilleursScores.h, 29
placementAleatBat
 Jeu.h, 33
placementBatValide
 Jeu.h, 34
position
 TBateau, 20

RESSOURCES_REP
 IncludeSDL.h, 65
Random.h
 choixMotHasard, 59
 initRandom, 59
 nombreAleatoire, 59
Rectangle, 17
 abscisse, 17
 couleur, 17
 hauteur, 17
 largeur, 17
 ordonnee, 17
 zoneRectangle, 17
resetInfoBateau
 Parametre.h, 55
restaurerBateaux
 FichierSauvRes.h, 29
restaurerCoups
 FichierSauvRes.h, 30
restaurerGrilles
 FichierSauvRes.h, 30
restaurerParam
 FichierSauvRes.h, 30
restaurerPartie
 FichierSauvRes.h, 30
retierInfoBateauxType
 Parametre.h, 55

SDL_Bouton, 18
 abscisse, 18
 largCarac, 18
 longTexte, 19
 ordonnee, 19
 tailleTexte, 19
 texte, 19
SDL_MsgBox, 19
SDLButton.h

- afficherBouton, 66
- clicSurBouton, 66
- creerBouton, 66
- libererBouton, 67
- SDLImage.h
 - afficherImage, 67
 - clicSurImage, 68
 - creerImage, 68
 - creerSDLImage, 68
 - libererImage, 68
- SDLMain, 19
- SDLRectangle.h
 - afficherRectangle, 69
 - clicSurRectangle, 69
 - creerRectangle, 69
 - incrCouleurRectangle, 70
 - libererRectangle, 70
- sauvegardeBateaux
 - FichierSauvRes.h, 31
- sauvegardeCoups
 - FichierSauvRes.h, 31
- sauvegardeGrille
 - FichierSauvRes.h, 31
- sauvegardeParam
 - FichierSauvRes.h, 31
- sauvegardePartie
 - FichierSauvRes.h, 32
- Score, 18
 - nom, 18
 - score, 18
- score
 - Score, 18
- Score.h
 - creerScore, 60
 - creerScoreP, 60
 - getNomScore, 61
 - getScore, 61
 - libererScore, 61
 - setNomScore, 61
 - setScore, 61
- scorePlayer
 - TPartie, 22
- sensBat
 - TSensBat, 23
- setEtatCase
 - Grille.h, 49
- setlemelInfoBateauTParam
 - Parametre.h, 55
- setInfoBateau
 - Parametre.h, 55
- setNomScore
 - Score.h, 61
- setPosBat
 - Bateau.h, 40
- setScore
 - Score.h, 61
- src/ctrl/EcransDivers.h, 25
- src/ctrl/FichierDebug.h, 25
- src/ctrl/FichierMeilleursScores.h, 27
- src/ctrl/FichierSauvRes.h, 29
- src/ctrl/Jeu.h, 32
- src/ctrl/Menu.h, 34
- src/ctrl/UtilsModel.h, 35
- src/ctrl/UtilsPoliceEcriture.h, 35
- src/ctrl/UtilsSDL.h, 36
- src/model/Bateau.h, 37
- src/model/ChampSaisie.h, 41
- src/model/Couleurs.h, 44
- src/model/Coups.h, 46
- src/model/Grille.h, 46
- src/model/Joueur.h, 50
- src/model/Parametre.h, 51
- src/model/Partie.h, 56
- src/model/PileCoup.h, 58
- src/model/Random.h, 59
- src/model/SDLMsgBox.h, 62
- src/model/Score.h, 60
- src/test/Test.h, 64
- src/test/model/TestBateau.h, 62
- src/test/model/TestParam.h, 63
- src/test/view/TestVue.h, 64
- src/view/IncludeSDL.h, 65
- src/view/SDLButton.h, 65
- src/view/SDLImage.h, 67
- src/view/SDLRectangle.h, 69
- src/view/VueBateau.h, 70
- src/view/VueChampSaisie.h, 70
- src/view/VueGrille.h, 72
- src/view/VueParam.h, 73
- src/view/VueRegles.h, 74
- supprimerDernierChar
 - ChampSaisie.h, 44
- TBateau, 19
 - estPlace, 20
 - etat, 20
 - idBateau, 20
 - position, 20
- TInfoBateau, 20
 - couleur, 20
 - nomBateau, 20
 - type, 20
- TPartie, 21
 - grille, 22
 - grilleMachine, 22
 - joueur, 22
 - machine, 22
 - parametres, 22
 - pileCoups, 22
 - scorePlayer, 22
- TPosition, 22
 - direction, 23
 - x, 23
 - y, 23
- TSensBat, 23
 - libSens, 23
 - sensBat, 23

- tailleTexte
 - ChampSaisie, [13](#)
 - SDL_Bouton, [19](#)
- TestBateau.h
 - testCreerBateau, [63](#)
 - testEstCoule, [63](#)
 - testToucheBateau, [63](#)
- testCreerBateau
 - TestBateau.h, [63](#)
- testEstCoule
 - TestBateau.h, [63](#)
- testToucheBateau
 - TestBateau.h, [63](#)
- texte
 - SDL_Bouton, [19](#)
- toucherBateau
 - Bateau.h, [41](#)
- Tparam, [21](#)
 - bateauxJoueur, [21](#)
 - bateauxMachine, [21](#)
 - nombreInstanceBateaux, [21](#)
- TtypeBat, [23](#)
 - nomType, [24](#)
 - typeBat, [24](#)
- type
 - TInfoBateau, [20](#)
- typeBat
 - TtypeBat, [24](#)
- updateGrille
 - VueGrille.h, [73](#)
- UtilsPoliceEcriture.h
 - chargerPoliceEcriture, [36](#)
- UtilsSDL.h
 - arreterSDL, [36](#)
 - demarrerSDL, [36](#)
- VueChampSaisie.h
 - afficherChamp, [71](#)
 - clicSurChamp, [71](#)
 - editerChamp, [71](#)
- VueGrille.h
 - afficherGrille, [72](#)
 - clicCaseGrille, [72](#)
 - clicDansGrille, [72](#)
 - updateGrille, [73](#)
- VueParam.h
 - afficherParamTest, [73](#)
- x
 - TPosition, [23](#)
- y
 - TPosition, [23](#)
- zoneImage
 - Image, [15](#)
- zoneRectangle
 - Rectangle, [17](#)