Bataille Navale

Généré par Doxygen 1.8.11

Table des matières

Chapitre 1

Index hiérarchique

1.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 2

Index des classes

2.1 Liste des classes

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

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la classe Affichage

classe qui gère l'affichage d'une partie de Bataille Navale.

`#include <Affichage.hpp>`

Fonctions membres publiques

- std : :string demanderNom ()
  fonction qui demande le nom du joueur
- int demanderDimensionGrille (std : :string nomDeLaDimension)
  fonction qui demande la dimension de la grille à l'utilisateur
- char choixTypeJeu ()
  fonction qui demande à l'utilisateur à quel type de jeu il souhaite jouer.
- void afficherGrille (Grille ∗grille)
  fonction permettant d'afficher la grille
- char demanderPlacementCorrect ()
  fonction qui demande à l'utilisateur si le placement demandé correspondant à la position du bateau visible à l'écran
- int demanderCoordonneesBateau (char coordonnee)
  fonction qui demande à l'utilisateur les coordonnées du bateau à placer
- char demanderOrientationBateau ()
  fonction qui demande à l'utilisateur si le bateau à placer doit être parallèle à l'axe des abcisses ou à l'axe des ordonnées
- int demanderTailleBateau ()
  fonction qui demande à l'utilisateur la taille du bateau à placer
- int demanderCoordonneesBombe (char coordonnee)
  fonction qui demande à l'utilisateur les coordonnées d'une bombe à lancer à son adversaire

Fonctions membres publiques statiques

- static void afficherMessage (std : :string message)
  fonction qui affiche un message à l'écran
- static char proposerNouveauJeuOuSauvegarde ()
  fonction qui enregistre le choix de l'utiliteur : lui porposant de charger une sauvegarde, supprimer une sauvegarde ou faire une nouvelle partie
- static char choixSauvegarde (std : :string ∗liste)
  fonction qui affiche la liste des jeux à charger et enregistre le choix de l'utilisateur
- static char proposerSauvegarder ()
  fonction qui propose à l'utilisateur de sauvegarder
- static std : :string demanderNomSauvegarde ()
  fonction qui demande le nom associé à la sauvegarde
- static char menuPrincipal ()
  fonction qui affiche le menu principale de la Bataille Navale
- static void attendreJoueurSuivant ()
  fonction qui demande à l'utilisateur d'appuyer sur entrée pour changer de joueur.
- static void afficherGagnant (std : :string nom)
  fonction qui affiche le gagnant à l'écran

4.1.1 Description détaillée

classe qui gère l'affichage d'une partie de Bataille Navale.

4.1.2 Documentation des fonctions membres

4.1.2.1 public Affichage : :afficherGagnant (std : :string *nom*) [static]

fonction qui affiche le gagnant à l'écran

Paramètres

nom	: nom du gagnant à afficher

4.1.2.2 public Affichage : :afficherGrille (Grille ∗ *grille*)

fonction permettant d'afficher la grille

Paramètres

∗*grille*	: la grille à afficher

4.1.2.3 public Affichage : :afficherMessage (std : :string *message*) [static]

fonction qui affiche un message à l'écran

Paramètres

message	de caractère message à afficher

4.1.2.4 public Affichage : :choixSauvegarde (std : :string ∗ *liste*) [static]

fonction qui affiche la liste des jeux à charger et enregistre le choix de l'utilisateur

Paramètres

∗*liste*	liste des jeux à charger, sauvegardés précedemment.

Renvoie

choix : un caractère précisant le choix de l'utilisateur.

4.1.2.5 public Affichage : :choixTypeJeu ( )

fonction qui demande à l'utilisateur à quel type de jeu il souhaite jouer.

Renvoie

le choix du joueur

4.1.2.6 public Affichage : :demanderCoordonneesBateau (char *coordonnee*)

fonction qui demande à l'utilisateur les coordonnées du bateau à placer

Paramètres

coordonnee	: indique si on est dans le cas de xExtremite ou yExtremite

Renvoie

un entier représentant les coordonnées en question

4.1.2.7 public Affichage : :demanderCoordonneesBombe (char *coordonnee*)

fonction qui demande à l'utilisateur les coordonnées d'une bombe à lancer à son adversaire

Paramètres

coordonnee	indique si on est dans le cas de xExtremite ou yExtremite

Renvoie

un entier représentant les coordonnées en question

4.1.2.8 public Affichage : :demanderDimensionGrille (std : :string *nomDeLaDimension*)

fonction qui demande la dimension de la grille à l'utilisateur

Paramètres

nomDeLaDimension	: hauteur, ou largeur

Renvoie

un entier representant la dimension voulue par l'utilisateur

4.1.2.9 public Affichage : :demanderNom ( )

fonction qui demande le nom du joueur

Renvoie

un caractère exprimant la réponse de l'utilisateur

4.1.2.10 public Affichage : :demanderNomSauvegarde ( ) [static]

fonction qui demande le nom associé à la sauvegarde

Renvoie

un caractère res exprimant la réponse de l'utilisateur

4.1.2.11 public Affichage : :demanderOrientationBateau ( )

fonction qui demande à l'utilisateur si le bateau à placer doit être parallèle à l'axe des abcisses ou à l'axe des ordonnées

Renvoie

la réponse de l'utilisateur

4.1.2.12 public Affichage : :demanderPlacementCorrect ( )

fonction qui demande à l'utilisateur si le placement demandé correspondant à la position du bateau visible à l'écran

Renvoie

la réponse de l'utilisateur

4.1.2.13 public Affichage : :demanderTailleBateau ( )

fonction qui demande à l'utilisateur la taille du bateau à placer

Renvoie

la réponse de l'utilisateur

4.1.2.14 public Affichage : :menuPrincipal ( ) [static]

fonction qui affiche le menu principale de la Bataille Navale

Renvoie

le choix de l'utilisateur :1, 2, ou 3 tel que : 1)Joueur contre joueur 2) Joueur contre IA 3) IA contre IA q) quitter.

4.1.2.15 public Affichage : :proposerNouveauJeuOuSauvegarde ( ) [static]

fonction qui enregistre le choix de l'utiliteur : lui porposant de charger une sauvegarde, supprimer une sauvegarde ou faire une nouvelle partie

Renvoie

choix : un caractère 1, 2 ou 3. 1) Faire un nouveau jeu 2) Charger une sauvegarde ; 3) Supprimer une sauvegarde.

4.1.2.16 public Affichage : :proposerSauvegarder ( ) [static]

fonction qui propose à l'utilisateur de sauvegarder

Renvoie

choix : un caractère représentant la réponse de l'utilisateur.

La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Affichage.hpp
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Affichage.cpp

4.2 Référence de la classe Bateau

classe fixant des bateaux au début de la partie. L'extrémité d'un bateau est située en haut et à gauche. C'est à dire que l'éxtrémité d'un bateau situé en position verticale est en haut et l'éxtrémité d'un bateau situé à l'horizontale est à gauche.

`#include <Bateau.hpp>`

Fonctions membres publiques

- Bateau ()
  Constructeur de la classe Bateau qui met ses caractéristiques en ses valeurs par défaut.
- bool placerSurGrille (Grille *grille)
  fonction qui modifie la grille passée en paramètre afin de placer le bateau dessus.
- void retirerDeLaGrille (Grille *grille)
  fonction qui permet de supprimer un bateau de la grille
- void setTaille (int taille)
  fonction qui modifie l'attribut taille du bateau
- void setxExtremite (int xExtremite)
  fonction qui modifie l'attribut xExtremite du bateau
- void setyExtremite (int yExtremite)
  fonction qui modifie l'attribut yExtremite du bateau
- void setOrientation (char orientationInput)
  fonction qui modifie l'orientation du bateau. O pour une orientation horizontale et 1 pour la verticale.
- bool getOrientation ()
  fonction qui renvoie l'orientation du bateau
- int getTaille ()
  fonction renvoie la longueur du bateau.
- int getxExtremite ()
  fonction qui renvoie l'attribut xExtremite (coordonnée x d'extremité) du bateau
- int getyExtremite ()
  fonction qui renvoie l'attribut yExtremite (coordonnée y d'extremité) du bateau
- int ** getCoordonneesCompletes ()
  fonction renvoie tous les coordonnées en x et y occupées par le bateau

Attributs privés

- int taille
- int xExtremite
- int yExtremite
- bool orientation

4.2.1 Description détaillée

classe fixant des bateaux au début de la partie. L'extrémité d'un bateau est située en haut et à gauche. C'est à dire que l'éxtrémité d'un bateau situé en position verticale est en haut et l'éxtrémité d'un bateau situé à l'horizontale est à gauche.

4.2.2 Documentation des fonctions membres

4.2.2.1 public Bateau : :getCoordonneesCompletes ( )

fonction renvoie tous les coordonnées en x et y occupées par le bateau

Renvoie

cette fonction retourne les coordonnées entiers (x,y) du bateau ;

4.2.2.2 public Bateau : :getOrientation ( )

fonction qui renvoie l'orientation du bateau

Renvoie

un booléen représentant l'orientation du bateau.

4.2.2.3 public Bateau : :getTaille ( )

fonction renvoie la longueur du bateau.

Renvoie

la longueur du bateau.

4.2.2.4 public Bateau : :getxExtremite ( )

fonction qui renvoie l'attribut xExtremite (coordonnée x d'extremité) du bateau

Renvoie

le coordonnée x d'extremité du bateau

4.2.2.5 public Bateau : :getyExtremite ( )

fonction qui renvoie l'attribut yExtremite (coordonnée y d'extremité) du bateau

Renvoie

la coordonnée en y de l'extremite du bateau

4.2.2.6 public Bateau : :placerSurGrille (Grille * *grille*)

fonction qui modifie la grille passée en paramètre afin de placer le bateau dessus.

Paramètres

∗*grille*	: la grille sur laquelle on souhaite placer le bateau

Renvoie

true : si le placement a été réussi. false : s'il n'y avait pas de place pour positionner le bateau

4.2.2.7 public Bateau : :retirerDeLaGrille (Grille ∗ *grille*)

fonction qui permet de supprimer un bateau de la grille

Paramètres

∗*grille*	: la grille du joueur

4.2.2.8 public Bateau : :setOrientation (char *orientationInput*)

fonction qui modifie l'orientation du bateau. O pour une orientation horizontale et 1 pour la verticale.

Paramètres

orientationInput	: caractère h (pour horizontal) ou v (pour vertical) qui va être transformé en son booléen correspondant (0 pour h et 1 pour v).

4.2.2.9 public Bateau : :setTaille (int *taille*)

fonction qui modifie l'attribut taille du bateau

Paramètres

taille	: taille voulue du bateau

4.2.2.10 public Bateau : :setxExtremite (int *xExtremite*)

fonction qui modifie l'attribut xExtremite du bateau

Paramètres

xExtremite	la coordonnée voulue

4.2.2.11 public Bateau : :setyExtremite (int *yExtremite*)

fonction qui modifie l'attribut yExtremite du bateau

Paramètres

yExtremite	la coordonnée voulue

4.2.3 Documentation des données membres

4.2.3.1 bool Bateau : :orientation [private]

orientation du bateau 0 pour l'horizontal et 1 pour le vertical

4.2.3.2 int Bateau : :taille [private]

La taille du bateau

4.2.3.3 int Bateau : :xExtremite [private]

La position selon la coordonnée en x du bateau sur la grille.

4.2.3.4 int Bateau : :yExtremite [private]

La position selon la coordonnée en y du bateau sur la grille.

La documentation de cette classe a été générée à partir des fichiers suivants :

- /home/lucas/Documents/GitHub/Bataille-Navale/src/Bateau.hpp
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Bateau.cpp

4.3 Référence de la classe GestionSauvegarde

classe qui gère les sauvegardes.

```
#include <GestionSauvegarde.hpp>
```

Fonctions membres publiques

- GestionSauvegarde ()
  Constructeur de la classe GestionSauvegarde. Le nomFichier est mis à "sauvegarde.sav" par défaut.
- GestionSauvegarde (std : :string nomFichier)
  Constructeur de la classe GestionSauvegarde.
- bool nouvelleSauvegarde (bool ecraserFichierSiExisteDeja=false)
  Initialise une nouvelle sauvegarde.
- void ecrireNomJoueur (char numeroJoueur, std : :string nomJoueur)
  Ecrit le nom `nomJoueur` du joueur numero `numeroJoueur` dans la sauvegarde.
- void ecrireAttribut (std : :string attribut, std : :string valeur)
  Ecrit l'attribut `attribut` dans la sauvegarde.
- void ecrireGrille (std : :string nomGrille, char **grille, int largeur, int hauteur)
  Ecrit la grille `grille` dans la sauvegarde.
- std : :string lireNomJoueur (char numeroJoueur)
  Lit le nom du joueur numero `numeroJoueur` dans la sauvegarde.
- std : :string lireAttribut (std : :string attribut)
  Lit le fichier ligne par ligne, s'arrête lorsqu'on a trouvé l'attribut `attribut` et lit les données le concernant puis les renvoie. Si le fichier présente plusieurs attributs portant le même nom, alors seul le 1er attribut sera lu.
- char ** lireGrille (std : :string nomGrille)
  Récupère la grille dont le nom est `nomGrille` dans le fichier de sauvegarde.
- int lireHauteurGrille (std : :string nomGrille)
  Récupère la hauteur de la grille portant le nom `nomGrille`.
- int lireLargeurGrille (std : :string nomGrille)
  Récupère la largeur de la grille portant le nom `nomGrille`.

Fonctions membres publiques statiques

- static std : :string * getListeSauvegardes ()
  Récupere la liste des noms de fichier de sauvegardes stockée dans le fichier FICHIERLISTESAV.
- static void supprimerSauvegarde (std : :string nomFichier)
  Supprime la sauvegarde contenue dans le fichier portant le nom `nomFichier`. Cette fonction supprime donc le fichier `nomFichier` mais aussi son nom de la liste des sauvegardes située dans le fichier FICHIERLISTESAV.

Fonctions membres privées

- bool fichierExiste (std : :string nomFichier="")
  Permet de savoir si un fichier existe déjà.
- int sauvegardeDejaDansListe ()
  Donne la position de la sauvegarde dans la liste des sauvegardes si elle est présente dedant. Si elle n'est pas présente on renvoie -1.
- std : :string : :size_type debutDeLaChaineSansEspaces (std : :string chaine)
  Donne la position du premier caractère qui n'est pas un espace dans la chaine `chaine`.
- int positionAttribut (std : :string attribut, bool verifierEgale=true)
  Donne la position de l'attribut `attribut` dans le fichier de sauvegarde.
- char * parseLigneGrille (std : :string ligne, int taille)
  Transforme une ligne de type "1 2 3 4" en un tableau de caractère [1, 2, 3, 4]. Attention le tableau résultant n'est pas ['1', '2', '3', '4'].
- bool estNumerique (char c)
  Permet de savoir si le caractère `c` est un chiffre comme '0', '1', etc.
- int recupererHauteurGrille (int pos)
  Permet d'obtenir la hauteur de la grille située à la position `pos` dans le fichier.
- void remplacerLigne (int pos, std : :string nouvelleLigne, bool ajouterLaLigne=false)
  Remplace la ligne à la position `pos` dans le fichier. On réécrit le début du fichier dans un fichier temporaire jusqu'à la ligne en question. On écrit ensuite la nouvelle ligne dans le fichier temporaire. On réécrit la fin du fichier dans le fichier temporaire jusqu'à la fin du fichier. On supprime alors le fichier et on renomme le fichier temporaire pour qu'il ait le même nom que le fichier d'origine.
- void effacerGrille (int pos, int h)
  Efface la grille située à la position `pos` dans le fichier.

Attributs privés

- std : :string nomFichier

4.3.1 Description détaillée

classe qui gère les sauvegardes.

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 public GestionSauvegarde : :GestionSauvegarde (std : :string *nomFichier*)

Constructeur de la classe GestionSauvegarde.

Paramètres

nomFichier	Le nom de la sauvegarde.

4.3.3 Documentation des fonctions membres

4.3.3.1 private GestionSauvegarde : :debutDeLaChaineSansEspaces (std : :string *chaine*) [private]

Donne la position du premier caractère qui n'est pas un espace dans la chaine `chaine`.

Paramètres

chaine	La chaine que l'on veut analyser.

Renvoie

la position du premier caractère qui n'est pas un espace dans la chaine `chaine`.

4.3.3.2 public GestionSauvegarde : :ecrireAttribut (std : :string *attribut,* std : :string *valeur*)

Ecrit l'attribut `attribut` dans la sauvegarde.

Paramètres

attribut	le nom de l'attribut à écrire.
valeur	la valeur de l'attribut à écrire.

4.3.3.3 public GestionSauvegarde : :ecrireGrille (std : :string *nomGrille,* char ** *grille,* int *largeur,* int *hauteur*)

Ecrit la grille `grille` dans la sauvegarde.

Paramètres

nomGrille	le nom de la grille. Par exemple, "grille1" pour la grille du joueur1, "grilleTentatives1" pour la grille des tentatives du joueur1.
grille	la grille à écrire.
largeur	la largeur de la grille.
hauteur	la hauteur de la grille.

4.3.3.4 public GestionSauvegarde : :ecrireNomJoueur (char *numeroJoueur,* std : :string *nomJoueur*)

Ecrit le nom `nomJoueur` du joueur numero `numeroJoueur` dans la sauvegarde.

Paramètres

numeroJoueur	le numero du joueur (joueur 1, joueur 2). Doit être un caractère '1' ou '2' par exemple.
nomJoueur	le nom du joueur.

4.3.3.5 private GestionSauvegarde : :effacerGrille (int *pos,* int *h*) [private]

Efface la grille située à la position `pos` dans le fichier.

Paramètres

pos	la position de la grille dans le fichier.
h	la hauteur de la grille.

4.3.3.6 private GestionSauvegarde : :estNumerique (char *c*) [private]

Permet de savoir si le caractère `c` est un chiffre comme '0', '1', etc.

Paramètres

c	le caractère dont on veut connaitre la nature.

Renvoie

true si `c` est numérique, false sinon.

4.3.3.7 private GestionSauvegarde : :fichierExiste (std : :string *nomFichier =* " ") [private]

Permet de savoir si un fichier existe déjà.

Paramètres

nomFichier	Le nom du fichier dont on veut savoir s'il existe. S'il n'est pas donné, alors on utilisera GestionSauvegarde : :nomFichier.

Renvoie

true si le fichier existe, false sinon.

4.3.3.8 public GestionSauvegarde : :getListeSauvegardes ( ) [static]

Récupere la liste des noms de fichier de sauvegardes stockée dans le fichier FICHIERLISTESAV.

Renvoie

la liste des noms des fichiers de sauvegarde sous la forme d'un tableau de string.

4.3.3.9 public GestionSauvegarde : :lireAttribut (std : :string *attribut*)

Lit le fichier ligne par ligne, s'arrête lorsqu'on a trouvé l'attribut `attribut` et lit les données le concernant puis les renvoie. Si le fichier présente plusieurs attributs portant le même nom, alors seul le 1er attribut sera lu.

Paramètres

attribut	le nom de l'attribut que l'on va chercher dans le fichier de sauvegarde.

Renvoie

la valeur de l'attribut.

4.3.3.10 public GestionSauvegarde : :lireGrille (std : :string *nomGrille*)

Récupère la grille dont le nom est `nomGrille` dans le fichier de sauvegarde.

Paramètres

nomGrille	le nom de la grille que l'on va chercher dans le fichier. L'attribut que l'on va chercher sera alors [`nomGrille`].

Renvoie

La grille sous le format pour JeuBatailleNavale. C'est à dire que si on a lu un '1' on stockera l'entier 1 dans la grille.

4.3.3.11 public GestionSauvegarde : :lireHauteurGrille (std : :string *nomGrille*)

Récupère la hauteur de la grille portant le nom `nomGrille`.

Paramètres

nomGrille	le nom de la grille que l'on va chercher dans le fichier. L'attribut que l'on va chercher sera alors [nomGrille].

Renvoie

La hauteur de la grille en question.

4.3.3.12 public GestionSauvegarde : :lireLargeurGrille (std : :string *nomGrille*)

Récupère la largeur de la grille portant le nom nomGrille.

Paramètres

nomGrille	le nom de la grille que l'on va chercher dans le fichier. L'attribut que l'on va chercher sera alors [nomGrille].

Renvoie

La largeur de la grille en question.

4.3.3.13 public GestionSauvegarde : :lireNomJoueur (char *numeroJoueur*)

Lit le nom du joueur numero numeroJoueur dans la sauvegarde.

Paramètres

numeroJoueur	le numero du joueur dont on veut lire le nom. L'attribut que l'on lire sera alors "nomJoueur" +numeroJoueur.

Renvoie

le nom du joueur.

4.3.3.14 public GestionSauvegarde : :nouvelleSauvegarde (bool *ecraserFichierSiExisteDeja* = false)

Initialise une nouvelle sauvegarde.

Paramètres

ecraserFichierSiExisteDeja	si à false, on ne fait rien si le fichier existe déjà. Si à true, on écrase le fichier s'il existe déjà. Par défaut à false.

Renvoie

true si l'initialisation a réussi. False sinon.

4.3.3.15 private GestionSauvegarde : :parseLigneGrille (std : :string *ligne,* int *taille*) [private]

Transforme une ligne de type "1 2 3 4" en un tableau de caractère [1, 2, 3, 4]. Attention le tableau résultant n'est pas ['1', '2', '3', '4'].

Paramètres

ligne	La ligne à analyser.
taille	le nombre de caractères qui ne sont pas des ' ' dans la `ligne`. C'est donc la taille du tableau qui sera retourné.

Renvoie

Un tableau de char correspondant à `ligne`.

4.3.3.16 private GestionSauvegarde : :positionAttribut (std : :string *attribut,* bool *verifierEgale* = `true`) [private]

Donne la position de l'attribut `attribut` dans le fichier de sauvegarde.

Paramètres

attribut	L'attribut que l'on souhaite trouver.
verifierEgale	Si est placé à false, on cherche l'attribut sans savoir s'il est bien suivi du caractère "". S'il est à true, on ne trouve l'aatribut que s'il est suivi de '='. Par défaut `verifierEgale` est à true.

Renvoie

la position de l'attribut dans la sauvegarde. Retourne -1 s'il n'est pas présent.

4.3.3.17 private GestionSauvegarde : :recupererHauteurGrille (int *pos*) [private]

Permet d'obtenir la hauteur de la grille située à la position `pos` dans le fichier.

Paramètres

pos	la position de la grille dans le fichier.

Renvoie

la hauteur de la grille. -1 s'il y a eu une erreur.

4.3.3.18 private GestionSauvegarde : :remplacerLigne (int *pos,* std : :string *nouvelleLigne,* bool *ajouterLaLigne* = `false`) [private]

Remplace la ligne à la position `pos` dans le fichier. On réécrit le début du fichier dans un fichier temporaire jusqu'à la ligne en question. On écrit ensuite la nouvelle ligne dans le fichier temporaire. On réécrit la fin du fichier dans le fichier temporaire jusqu'à la fin du fichier. On supprime alors le fichier et on renomme le fichier temporaire pour qu'il ait le même nom que le fichier d'origine.

Paramètres

pos	la position de la ligne à remplacer.
nouvelleLigne	La ligne que l'on écrit à la place de l'ancienne
ajouterLaLigne	si à false, on remplace la ligne, si à true on ajoute la ligne. Par défaut à false.

4.3.3.19 private GestionSauvegarde : :sauvegardeDejaDansListe ( ) [private]

Donne la position de la sauvegarde dans la liste des sauvegardes si elle est présente dedant. Si elle n'est pas présente on renvoie -1.

Renvoie

La position de la sauvegarde dans la liste si elle est présente, -1 sinon

4.3.3.20 public GestionSauvegarde : :supprimerSauvegarde (std : :string *nomFichier*) [static]

Supprime la sauvegarde contenue dans le fichier portant le nom `nomFichier`. Cette fonction supprime donc le fichier `nomFichier` mais aussi son nom de la liste des sauvegardes située dans le fichier FICHIERLISTESAV.

Paramètres

nomFichier	le nom du fichier de sauvegarde que l'on souhaite supprimer.

4.3.4 Documentation des données membres

4.3.4.1 std : :string GestionSauvegarde : :nomFichier [private]

Le nom du fichier dans lequel on va sauvegarder ou à partir duquel on va lire la sauvegarde

La documentation de cette classe a été générée à partir des fichiers suivants :

— /home/lucas/Documents/GitHub/Bataille-Navale/src/GestionSauvegarde.hpp
— /home/lucas/Documents/GitHub/Bataille-Navale/src/GestionSauvegarde.cpp

4.4 Référence de la classe Grille

classe qui gère les fonctionnalités de la grille des joueurs de la Bataille Navale.

```
#include <Grille.hpp>
```

Fonctions membres publiques

- Grille ()
  Constructeur de la classe Grille qui initialise la grille.
- void setTaille (int largeur, int hauteur)
  fonction qui modifie l'attribut taille de la grille
- void setGrille (char **grille)
  fonction qui modifie l'attribut grille
- void reset ()
  fonction qui alloue la mémoire pour la grille et la remplie de 0, donc d'eau
- bool verifierPlace (int **tableauDeCoordonnees, int taille)
  fonction qui verifie les critères à respecter avant de placer le bateau sur la grille
- bool fini ()
  fonction qui scanne la grille jusqu'a ce qu'on trouve au moins 1 bateau qui n'a pas été touché
- char get (int x, int y)
  fonction qui renvoie le contenu de la grille (Eau, Bateau) sachant une certaine position
- int getLargeur ()
  fonction qui renvoie la largeur de la grille
- int getHauteur ()
  fonction qui renvoie la hauteur de la grille
- void set (int x, int y, char valeur)
  fonction qui permet de modifier le contenu d'une case du tableau sachant des coordonnées
- char ** getGrille ()
  fonction qui renvoie la grille

Attributs privés

- char ** grille

Attributs privés statiques

- static int largeur = 10
- static int hauteur = 10

Amis

- class **Affichage**

4.4.1 Description détaillée

classe qui gère les fonctionnalités de la grille des joueurs de la Bataille Navale.

4.4.2 Documentation des fonctions membres

4.4.2.1 public Grille : :fini ( )

fonction qui scanne la grille jusqu'a ce qu'on trouve au moins 1 bateau qui n'a pas été touché

Renvoie

true si tous les bateaux ont été touchés

4.4.2.2 public Grille : :get (int *x*, int *y*)

fonction qui renvoie le contenu de la grille (Eau, Bateau) sachant une certaine position

Paramètres

x,y	: les coordonnées de la position voulue

Renvoie

le contenu de la grille à cette position

4.4.2.3 public Grille : :getGrille ( )

fonction qui renvoie la grille

Renvoie

la grille

4.4.2.4 public Grille : :getHauteur ( )

fonction qui renvoie la hauteur de la grille

Renvoie

un entier représentant la hauteur de la grille

4.4.2.5 public Grille : :getLargeur ( )

fonction qui renvoie la largeur de la grille

Renvoie

un entier représentant la largeur de la grille

4.4.2.6 public Grille : :set (int *x,* int *y,* char *valeur*)

fonction qui permet de modifier le contenu d'une case du tableau sachant des coordonnées

Paramètres

x,y,valeur	les coordonnées de la position voulue et la modification à apporter

4.4.2.7 public Grille : :setGrille (char ∗∗ *grille*)

fonction qui modifie l'attribut grille

Paramètres

∗∗*grille*	le tableau à enregistrer pour grille

4.4.2.8 public Grille : :setTaille (int *largeur,* int *hauteur*)

fonction qui modifie l'attribut taille de la grille

Paramètres

largeur	: largeur voulue pour la grille
hauteur	: hauteur voulue pour la grille

4.4.2.9 public Grille : :verifierPlace (int ∗∗ *tableauDeCoordonnees,* int *taille*)

fonction qui verifie les critères à respecter avant de placer le bateau sur la grille

Paramètres

tableauDeCoordonnees	tableau de coordonnées du bateau
taille	la taille du bateau

Renvoie

true : si les caracteristiques sont respectées

4.4.3 Documentation des données membres

4.4.3.1 char∗∗ Grille : :grille [private]

Le tableau representant le cadrillage. Ici, la valeur d'une case est : EAU, BATEAU, BATEAUTOUCHE, TENTATIVEREUSSIE ou TENTATIVERATEE

4.4.3.2 int Grille : :hauteur = 10 [static], [private]

La longueur de la grille

4.4.3.3 int Grille : :largeur = 10 [static], [private]

La largeur de la grille

La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Grille.hpp
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Grille.cpp

4.5 Référence de la classe JeuBatailleNavale

classe qui gère le déroulement du jeu de Bataille Navale.

`#include <JeuBatailleNavale.hpp>`

Fonctions membres publiques

- JeuBatailleNavale ()
  Constructeur de la classe.
- JeuBatailleNavale (std : :string nomJoueur1, std : :string nomJoueur2, char typeJeu, bool IA1, bool IA2, char ∗∗grille1, char ∗∗grilleTentatives1, char ∗∗grille2, char ∗∗grilleTentatives2, int h, int l)
  Constructeur de la classe permettant de choisir l'initialisation des paramètres pour le chargement d'une sauvegarde.
- void nouveauJeu ()
  commence un nouveau jeu, demande au joueur si quels serons les jeurs : 2 IA, 2 humains ou un humain et une IA. Modifie l'attribut typeJeu.
- void jouer (bool tour=false)
  fonction qui verifie si la partie est finie, demande au joueur des coordonnées de bombes, et informe le joueur adverse si un de ses bateaux a été touché
- bool checkFinJeu ()
  Fonction pour déterminer si le jeu est fini.

Fonctions membres privées

- void instanciationDesJoueurs (char choix)
  Modifie joueur1 et joueur2 en isntanciant JoueurHumain ou JoueurIA.
- void demanderNomJoueurs ()
  initialise le nom du/des joueur/s.
- bool joueurPlaceBombe (bool joueur, int xBombe, int yBombe)
  fonction qui verifie qu'à une position donnée, la grille de l'adversaire ne contient pas d'eau
- void ecrireSauvegarde (std : :string nomSauvegarde, bool tour)
  fonction qui effectue une sauvegarde après un tour

Attributs privés

- Joueur ∗ joueur1
- Joueur ∗ joueur2
- char typeJeu

4.5.1 Description détaillée

classe qui gère le déroulement du jeu de Bataille Navale.

4.5.2 Documentation des fonctions membres

4.5.2.1 private JeuBatailleNavale : :checkFinJeu ( )

Fonction pour déterminer si le jeu est fini.

Renvoie

0 si ce n'est pas la fin du jeu et 1 si c'est la fin du jeu.

4.5.2.2 private JeuBatailleNavale : :ecrireSauvegarde (std : :string *nomSauvegarde,* bool *tour*) [private]

fonction qui effectue une sauvegarde après un tour

Paramètres

nomSauvegarde,tour	

4.5.2.3 private JeuBatailleNavale : :instanciationDesJoueurs (char *choix*) [private]

Modifie joueur1 et joueur2 en isntanciant JoueurHumain ou JoueurIA.

Paramètres

choix	1 : 2 joueurs humains, 2 : 1 joueur humain et 1 joueur IA, 3 : 2 joueurs IA

4.5.2.4 public JeuBatailleNavale : :jouer (bool *tour* = false)

fonction qui verifie si la partie est finie, demande au joueur des coordonnées de bombes, et informe le joueur adverse si un de ses bateaux a été touché

Paramètres

tour	

4.5.2.5 private JeuBatailleNavale : :joueurPlaceBombe (bool *joueur,* int *xBombe,* int *yBombe*) [private]

fonction qui verifie qu'à une position donnée, la grille de l'adversaire ne contient pas d'eau

Paramètres

joueur,xBombe,yBombe	

Renvoie

true si la case de la grille adverse ne contient pas d'eau

4.5.3 Documentation des données membres

4.5.3.1 Joueur* JeuBatailleNavale : :joueur1 [private]

le premier joueur de la partie

4.5.3.2 Joueur* JeuBatailleNavale : :joueur2 [private]

le deuxième joueur de la partie

4.5.3.3 char JeuBatailleNavale : :typeJeu [private]

le type de jeu : une bataille navalle normale ou améliorée

La documentation de cette classe a été générée à partir des fichiers suivants :

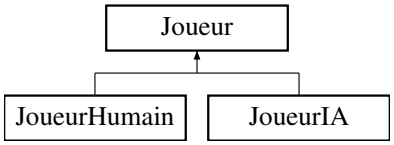
— /home/lucas/Documents/GitHub/Bataille-Navale/src/JeuBatailleNavale.hpp
— /home/lucas/Documents/GitHub/Bataille-Navale/src/JeuBatailleNavale.cpp

4.6 Référence de la classe Joueur

classe Joueur regroupant les fonctionnalités générales sur le joueur

```
#include <Joueur.hpp>
```

Graphe d'héritage de Joueur :



Fonctions membres publiques

— Joueur ()
Constructeur de la classe Joueur qui met ses caractéristiques en ses valeurs par défaut.
— virtual bool estUneIA ()
Permet de savoir si ce joueur est une IA. Est utile lorsque l'on manipule un Joueur et que l'on ne sait pas s'il est IA ou Humain.
— void resetGrilles ()
fonction qui alloue la mémoire pour la grille et la grille des tentatives et les remplie de 0, donc d'eau
— void definirBateauxType2 (int nbMaxDeBateaux, int tailleGrille)
function permettant de définir un tableau de bateaux de tailles aléatoires
— void setNom (std : :string nouveauNom)
fonction qui modifie l'attribut nom
— void setGrille (char ∗∗grille, int h, int l)
fonction qui modifie les caractéristiques de la grille
— void setGrilleTentatives (char ∗∗grilleTentatives)
fonction qui modifie l'attribut grilleTentatives
— virtual void demanderNom ()
initialise le nom du joueur en demandant le nom au joueur
— virtual void demanderTailleGrille ()
initialise la taille de la grille en la demandant au joueur
— virtual char demanderTypeJeu ()
fonction qui permet au joueur de choisir son type de jeu : bataille classique ou bataille améliorée
— virtual void placementDesBateaux (char typeJeu)
demande au joueur de placer ses différents bateaux. Utilise ajoutBateau pour chaqun des bateaux du joueur.
— virtual int ∗ tour ()
Permet de placer une bombe, affiche aussi la grille et la grille de tentatives.
— virtual void resultatBombe (bool touche, int x, int y)
Notifie le joueur si sa bombe a touché l'adversaire.
— virtual void resultatBombeAdverse (bool touche, int x, int y)
Notifie le joueur si la bombe de l'adversaire l'a touché.

Fonctions membres protégées

- virtual void placerBateau (Bateau *b)
  *Permet au joueur de positionner le bateau *b sur la grille. Cette fonction lui demande les coordonnées en x, y du bateau (en utilisant la fonction demanderCoordonneesBateau) mais aussi son orientation.*
- void marquerResultatBombeSurGrilleTentative (bool touche, int x, int y)
  function permettant de modifier une case de la grille des tentatives sachant les conséquences de la bombe
- void marquerResultatBombeSurGrille (bool touche, int x, int y)
  function permettant de modifier une case de la grille sachant les conséquences de la bombe

Attributs protégés

- std : :string nom
- Grille grille
- Grille grilleTentatives
- int nbCasesBateauxTouches
- int nbCasesBateaux
- int nbBateaux
- Bateau * bateaux
- Affichage * affichage

Amis

- class **JeuBatailleNavale**

4.6.1 Description détaillée

classe Joueur regroupant les fonctionnalités générales sur le joueur

4.6.2 Documentation des fonctions membres

4.6.2.1 public Joueur : :definirBateauxType2 (int *nbMaxDeBateaux,* int *tailleGrille*)

function permettant de définir un tableau de bateaux de tailles aléatoires

Paramètres

nbMaxDeBateaux,tailleGrille	le nombre de bateaux du tableau et la taille de la grille

4.6.2.2 public Joueur : :estUneIA ( ) [inline],[virtual]

Permet de savoir si ce joueur est une IA. Est utile lorsque l'on manipule un Joueur et que l'on ne sait pas s'il est IA ou Humain.

Renvoie

true si c'est une IA, false sinon.

Réimplémentée dans JoueurIA, et JoueurHumain.

4.6.2.3 protected Joueur : :marquerResultatBombeSurGrille (bool *touche,* int *x,* int *y*) [protected]

function permettant de modifier une case de la grille sachant les conséquences de la bombe

Paramètres

touche,x,y	les coordonnées de la position à modifier, et touche : l'indication si la bombe a atteint sa cible

4.6.2.4 protected Joueur : :marquerResultatBombeSurGrilleTentative (bool *touche,* int *x,* int *y*) [protected]

function permettant de modifier une case de la grille des tentatives sachant les conséquences de la bombe

Paramètres

touche,x,y	les coordonnées de la position à modifier, et touche : l'indication si la bombe a atteint sa cible

4.6.2.5 private Joueur : :placementDesBateaux (char *typeJeu*) [inline], [virtual]

demande au joueur de placer ses différents bateaux. Utilise ajoutBateau pour chaqun des bateaux du joueur.

Paramètres

typeJeu	le type de jeu choisis

Réimplémentée dans JoueurIA, et JoueurHumain.

4.6.2.6 protected Joueur : :placerBateau (Bateau * *b*) [inline], [protected], [virtual]

Permet au joueur de positionner le bateau *b sur la grille. Cette fonction lui demande les coordonnées en x, y du bateau (en utilisant la fonction demanderCoordonneesBateau) mais aussi son orientation.

Paramètres

*b	Un pointeur sur le bateau que le joueur va placer sur la grille.

Réimplémentée dans JoueurHumain, et JoueurIA.

4.6.2.7 public Joueur : :resultatBombe (bool *touche,* int *x,* int *y*) [inline], [virtual]

Notifie le joueur si sa bombe a touché l'adversaire.

Paramètres

touche	true si la bombe a touché un navire adverse, false sinon.
x	La coordonnée en x de la bombe que le joueur avait choisie.
y	La coordonnée en y de la bombe que le joueur avait choisie.

Réimplémentée dans JoueurIA, et JoueurHumain.

4.6.2.8 public Joueur : :resultatBombeAdverse (bool *touche,* int *x,* int *y*) [inline],[virtual]

Notifie le joueur si la bombe de l'adversaire l'a touché.

Paramètres

touche	true si la bombe a touché un navire, false sinon.
x	La coordonnée en x de la bombe que le joueur adverse avait choisie.
y	La coordonnée en y de la bombe que le joueur adverse avait choisie.

Réimplémentée dans JoueurIA, et JoueurHumain.

4.6.2.9 public Joueur : :setGrille (char ** *grille,* int *h,* int *l*)

fonction qui modifie les caractéristiques de la grille

Paramètres

grille,h,l	grille : object modifié, hauteur, largueur

4.6.2.10 public Joueur : :setGrilleTentatives (char ** *grilleTentatives*)

fonction qui modifie l'attribut grilleTentatives

Paramètres

***grilleTentatives*	

4.6.2.11 public Joueur : :setNom (std : :string *nouveauNom*)

fonction qui modifie l'attribut nom

Paramètres

nouveauNom	

4.6.2.12 public Joueur : :tour ( ) [inline],[virtual]

Permet de placer une bombe, affiche aussi la grille et la grille de tentatives.

Renvoie

Les coordonnées de la bombe cible. Sous la forme d'un tableau [x, y].

Réimplémentée dans JoueurIA, et JoueurHumain.

4.6.3 Documentation des données membres

4.6.3.1 Affichage* Joueur : :affichage [protected]

Un affichage pour afficher des informations générales sur le jeux.

4.6.3.2 Bateau* Joueur : :bateaux [protected]

Un tableau contenant les bateaux possédés par le joueur et qu'il a placé sur la grille.

4.6.3.3 Grille Joueur : :grille [protected]

La grille sur laquelle le joueur place ses bateaux

4.6.3.4 Grille Joueur : :grilleTentatives [protected]

La grille permettant de garder en mémoire les tentatives de bombes effectuées par le joueur

4.6.3.5 int Joueur : :nbBateaux [protected]

/ Le nombre de bateau initial du joueur. Permet de simplifier l'écriture de certaines boucles. (i $<$ nbBateaux etc)

4.6.3.6 int Joueur : :nbCasesBateaux [protected]

Le nombre de cases occupées par des bateaux sur la grille

4.6.3.7 int Joueur : :nbCasesBateauxTouches [protected]

Correspond au nombre de fois que l'adversaire a touché un bateau.Va être incrementé quand un des bateaux est touché pour faciliter la vérification de la fin de partie.

4.6.3.8 std : :string Joueur : :nom [protected]

Le nom du joueur

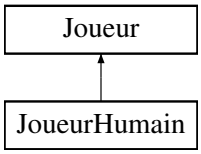
La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Joueur.hpp
- /home/lucas/Documents/GitHub/Bataille-Navale/src/Joueur.cpp

4.7 Référence de la classe JoueurHumain

classe representant le joueur qui est humain. Cette classe hérite de la classe Joueur

`#include <JoueurHumain.hpp>`

Graphe d'héritage de JoueurHumain :



Fonctions membres publiques

- JoueurHumain ()
  Constructeur de JoueurHumain.
- virtual bool estUneIA ()
  Permet de savoir si ce joueur est une IA. Est utile lorsque l'on manipule un Joueur et que l'on ne sait pas s'il est IA ou Humain.
- virtual void demanderNom ()
  Demande au joueur d'entrer son nom. Cette fonction modifie l'attribut nom.
- virtual void demanderTailleGrille ()
  Demande au joueur la largeur et la hauteur de la grille sur laquelle il souhaite jouer.
- virtual char demanderTypeJeu ()
  Demande au joueur selon quel type de jeu de Bataille Navale il souhaite jouer. Le type de jeu est soit 1 soit 2.
- virtual void placementDesBateaux (char typeJeu)
  Permet au joueur de placer tous ses bateaux sur la grille. Fais appel à la fonction placerBateau(Bateau ∗b).
- virtual int ∗ tour ()
  Permet au joueur de placer une bombe. Lui affiche aussi sa grille et sa grille de tentatives.
- virtual void resultatBombe (bool touche, int x, int y)
  Notifie le joueur si sa bombe a touché l'adversaire. On modifie alors sa grille en conséquence.
- virtual void resultatBombeAdverse (bool touche, int x, int y)
  Notifie le joueur si la bombe de l'adversaire l'a touché.

Fonctions membres privées

- virtual void placerBateau (Bateau ∗b)
  Permet au joueur de positionner le bateau ∗b sur la grille. Cette fonction lui demande les coordonnées en x, y du bateau (en utilisant la fonction demanderCoordonneesBateau) mais aussi son orientation.
- void demanderCoordonneesBateau (Bateau ∗b)
  Demande au joueur les coordonnées en x et y du bateau afin de le positionner sur la grille.
- int ∗ donnerCoordonneesBombes ()
  Demande au joueur les coordonnées en x et y de la bombe qu'il souhaite envoyer sur les bateaux adverses.

Membres hérités additionnels

4.7.1 Description détaillée

classe representant le joueur qui est humain. Cette classe hérite de la classe Joueur

La classe gere les actions du joueur.

4.7.2 Documentation des fonctions membres

4.7.2.1 private JoueurHumain : :demanderCoordonneesBateau (Bateau ∗ *b*) [private]

Demande au joueur les coordonnées en x et y du bateau afin de le positionner sur la grille.

Paramètres

$*b$	Un pointeur sur le bateau que le joueur va placer sur la grille.

4.7.2.2 public JoueurHumain : :demanderTypeJeu ( ) [virtual]

Demande au joueur selon quel type de jeu de Bataille Navale il souhaite jouer. Le type de jeu est soit 1 soit 2.

Renvoie

le type de jeu. Vaudra soit 1 soit 2. Attention il s'agit de la valeur 1 ou 2 et non du caractère '1' ou '2'.

Réimplémentée à partir de Joueur.

4.7.2.3 private JoueurHumain : :donnerCoordonneesBombes ( ) [private]

Demande au joueur les coordonnées en x et y de la bombe qu'il souhaite envoyer sur les bateaux adverses.

Renvoie

Un tableau représentant les coordonnées en x et en y de la forme [x, y].

4.7.2.4 public JoueurHumain : :estUneIA ( ) [virtual]

Permet de savoir si ce joueur est une IA. Est utile lorsque l'on manipule un Joueur et que l'on ne sait pas s'il est IA ou Humain.

Renvoie

true si c'est une IA, false sinon.

Réimplémentée à partir de Joueur.

4.7.2.5 public JoueurHumain : :placementDesBateaux (char *typeJeu*) [virtual]

Permet au joueur de placer tous ses bateaux sur la grille. Fais appel à la fonction placerBateau(Bateau $*$b).

Paramètres

typeJeu	Le type de jeu auquel le joueur joue. Cela influence le nombre de bateaux et leur taille.

Réimplémentée à partir de Joueur.

4.7.2.6 private JoueurHumain : :placerBateau (**Bateau** $*$ *b*) [private], [virtual]

Permet au joueur de positionner le bateau $*$b sur la grille. Cette fonction lui demande les coordonnées en x, y du bateau (en utilisant la fonction demanderCoordonneesBateau) mais aussi son orientation.

Paramètres

∗b	Un pointeur sur le bateau que le joueur va placer sur la grille.

Réimplémentée à partir de Joueur.

4.7.2.7 public JoueurHumain : :resultatBombe (bool *touche,* int *x,* int *y*) [virtual]

Notifie le joueur si sa bombe a touché l'adversaire. On modifie alors sa grille en conséquence.

Paramètres

touche	true si la bombe a touché un navire adverse, false sinon.
x	La coordonnée en x de la bombe que le joueur avait choisie.
y	La coordonnée en y de la bombe que le joueur avait choisie.

Réimplémentée à partir de Joueur.

4.7.2.8 public JoueurHumain : :resultatBombeAdverse (bool *touche,* int *x,* int *y*) [virtual]

Notifie le joueur si la bombe de l'adversaire l'a touché.

Paramètres

touche	true si la bombe a touché un navire, false sinon.
x	La coordonnée en x de la bombe que le joueur adverse avait choisie.
y	La coordonnée en y de la bombe que le joueur adverse avait choisie.

Réimplémentée à partir de Joueur.

4.7.2.9 public JoueurHumain : :tour ( ) [virtual]

Permet au joueur de placer une bombe. Lui affiche aussi sa grille et sa grille de tentatives.

Renvoie

Les coordonnées de la bombe que le joueur souhaite placer. Sous la forme d'un tableau [x, y].

Réimplémentée à partir de Joueur.

La documentation de cette classe a été générée à partir des fichiers suivants :

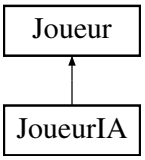
- /home/lucas/Documents/GitHub/Bataille-Navale/src/JoueurHumain.hpp
- /home/lucas/Documents/GitHub/Bataille-Navale/src/JoueurHumain.cpp

4.8 Référence de la classe JoueurIA

classe representant le joueur qui est IA. Cette classe hérite de la classe Joueur

`#include <JoueurIA.hpp>`

Graphe d'héritage de JoueurIA :



Fonctions membres publiques

- JoueurIA ()
  Constructeur de JoueurIA.
- virtual bool estUneIA ()
  Permet de savoir si ce joueur est une IA. Est utile lorsque l'on manipule un Joueur et que l'on ne sait pas s'il est IA ou Humain.
- virtual void demanderNom ()
  L'IA donnera "IA" comme nom.
- virtual void demanderTailleGrille ()
  L'IA donnera toujours 10x10 comme taille de grille.
- virtual char demanderTypeJeu ()
  L'IA donnera toujours un jeu de type 2.
- virtual void placementDesBateaux (char typeJeu)
  Permet à l'IA de placer tous ses bateaux sur la grille. L'IA choisit d'abord de façon aléatoire quels bateaux elle va placer. Puis fais appel à la fonction placerBateau(Bateau ∗b).
- virtual int ∗ tour ()
  Permet à l'IA de placer une bombe. Lui affiche aussi sa grille et sa grille de tentatives. Les coordonnées de la bombe sont déterminées à l'aide d'une fonction comme determinerCoordonneesBombesScan() ou determiner↩ CoordonneesBombesAleatoire().
- virtual void resultatBombe (bool touche, int x, int y)
  Notifie l'IA si sa bombe a touché l'adversaire. On modifie alors sa grille en conséquence.
- virtual void resultatBombeAdverse (bool touche, int x, int y)
  Notifie l'IA si la bombe de l'adversaire l'a touché.

Fonctions membres privées

- virtual void placerBateau (Bateau ∗bateau)
  Permet au joueur de positionner le bateau `∗b` sur la grille. Cette fonction lui génère les coordonnées en x, y du bateau et son orientation.
- int ∗ retirerDeLaListe (int ∗liste, int taille, int ind)
  Retire la valeur présente à l'indice `ind` de la liste `liste`. Il s'agit d'une liste d'indices correspondant aux bateaux que l'IA peut choisir. Cette liste est utilisée dans placementDesBateaux(char typeJeu)
- int ∗ determinerCoordonneesBombesAleatoire ()
  L'IA choisit d'envoyer une bombe à des coordonnées aléatoires sur la grille de l'adversaire.
- int ∗ determinerCoordonneesBombesScan ()
  L'IA choisit d'envoyer une bombe en partant de la coordonnée [0, 0] puis [1, 0] etc [0, 1] etc [n, n]. Elle balaye donc la grille de haut en bas et de gauche à droite.

Membres hérités additionnels

4.8.1 Description détaillée

classe representant le joueur qui est IA. Cette classe hérite de la classe Joueur

La classe gere les actions du joueur.

4.8.2 Documentation des fonctions membres

4.8.2.1 public JoueurIA : :demanderTypeJeu ( ) [virtual]

L'IA donnera toujours un jeu de type 2.

Renvoie

le type de jeu. Vaudra soit 1 soit 2. Attention il s'agit de la valeur 1 ou 2 et non du caractère '1' ou '2'.

Réimplémentée à partir de Joueur.

4.8.2.2 private JoueurIA : :determinerCoordonneesBombesAleatoire ( ) [private]

L'IA choisit d'envoyer une bombe à des coordonnées aléatoires sur la grille de l'adversaire.

Renvoie

Les coordonnées de la bombe sous la forme d'un tableau [x, y].

4.8.2.3 private JoueurIA : :determinerCoordonneesBombesScan ( ) [private]

L'IA choisit d'envoyer une bombe en partant de la coordonnée [0, 0] puis [1, 0] etc [0, 1] etc [n, n]. Elle balaye donc la grille de haut en bas et de gauche à droite.

Renvoie

Les coordonnées de la bombe sous la forme d'un tableau [x, y].

4.8.2.4 public JoueurIA : :estUneIA ( ) [virtual]

Permet de savoir si ce joueur est une IA. Est utile lorsque l'on manipule un Joueur et que l'on ne sait pas s'il est IA ou Humain.

Renvoie

true si c'est une IA, false sinon.

Réimplémentée à partir de Joueur.

4.8.2.5 public JoueurIA : :placementDesBateaux (char *typeJeu*) [virtual]

Permet à l'IA de placer tous ses bateaux sur la grille. L'IA choisit d'abord de façon aléatoire quels bateaux elle va placer. Puis fais appel à la fonction placerBateau(Bateau *b).

Paramètres

typeJeu	Le type de jeu auquel l'IA joue. Cela influence le nombre de bateaux et leur taille.

Réimplémentée à partir de Joueur.

4.8.2.6 private JoueurIA : :placerBateau (Bateau ∗ *b*) [private], [virtual]

Permet au joueur de positionner le bateau ∗b sur la grille. Cette fonction lui génère les coordonnées en x, y du bateau et son orientation.

Paramètres

∗*b*	Un pointeur sur le bateau que l'IA va placer sur la grille.

Réimplémentée à partir de Joueur.

4.8.2.7 public JoueurIA : :resultatBombe (bool *touche*, int *x*, int *y*) [virtual]

Notifie l'IA si sa bombe a touché l'adversaire. On modifie alors sa grille en conséquence.

Paramètres

touche	true si la bombe a touché un navire adverse, false sinon.
x	La coordonnée en x de la bombe que le joueur avait choisie.
y	La coordonnée en y de la bombe que le joueur avait choisie.

Réimplémentée à partir de Joueur.

4.8.2.8 public JoueurIA : :resultatBombeAdverse (bool *touche*, int *x*, int *y*) [virtual]

Notifie l'IA si la bombe de l'adversaire l'a touché.

Paramètres

touche	true si la bombe a touché un navire, false sinon.
x	La coordonnée en x de la bombe que le joueur adverse avait choisie.
y	La coordonnée en y de la bombe que le joueur adverse avait choisie.

Réimplémentée à partir de Joueur.

4.8.2.9 private JoueurIA : :retirerDeLaListe (int ∗ *liste*, int *taille*, int *ind*) [private]

Retire la valeur présente à l'indice `ind` de la liste `liste`. Il s'agit d'une liste d'indices correspondant aux bateaux que l'IA peut choisir. Cette liste est utilisée dans placementDesBateaux(char typeJeu)

Paramètres

∗*liste*	La liste que l'on va recopier en retirant un élément puis en renvoyer cet copie.
taille	La taille de la `liste`.
ind	L'indice de l'élément de la `liste` que l'on souhaite supprimer.

Renvoie

Une liste identique à `liste` mais avec l'élément à l'indice `ind` supprimé.

4.8.2.10 public JoueurIA : :tour ( ) [virtual]

Permet à l'IA de placer une bombe. Lui affiche aussi sa grille et sa grille de tentatives. Les coordonnées de la bombe sont déterminées à l'aide d'une fonction comme determinerCoordonneesBombesScan() ou determiner↩ CoordonneesBombesAleatoire().

Renvoie

Les coordonnées de la bombe que l'IA souhaite placer. Sous la forme d'un tableau [x, y].

Réimplémentée à partir de Joueur.

La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/lucas/Documents/GitHub/Bataille-Navale/src/JoueurIA.hpp
- /home/lucas/Documents/GitHub/Bataille-Navale/src/JoueurIA.cpp

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/Affichage.hpp

classe Affichage

```
#include <iostream>
#include <string>
#include "Grille.hpp"
```

Classes

— class Affichage
classe qui gère l'affichage d'une partie de Bataille Navale.

5.1.1 Description détaillée

classe Affichage

Auteur

groupe B7

Version

0.1

Date

22 mai 2018

Classe Affichage, classe qui gère l'affichage d'une partie de Bataille Navale.

5.2 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/Bateau.hpp

classe Bateau

```
#include "Grille.hpp"
```

Classes

— class Bateau
classe fixant des bateaux au début de la partie. L'extrémité d'un bateau est située en haut et à gauche. C'est à dire que l'éxtrémité d'un bateau situé en position verticale est en haut et l'éxtrémité d'un bateau situé à l'horizontale est à gauche.

5.2.1 Description détaillée

classe Bateau

Auteur

Groupe B7

Version

0.1

Date

22 mai 2018 Classe Bateau qui fixe des bateaux au début de la partie

5.3 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/GestionSauvegarde.hpp

classe GestionSauvegarde

```
#include <iostream>
#include <fstream>
#include <string>
#include <math.h>
#include <string.h>
```

Classes

— class GestionSauvegarde
classe qui gère les sauvegardes.

Macros

— #define ERREURATTRIBUT "$"
— #define FICHIERLISTESAV "listeDesSauvegardes.txt"

5.3.1 Description détaillée

classe GestionSauvegarde

Auteur

groupe B7

Version

0.1

Date

22 mai 2018

Classe GestionSauvegarde, classe qui gère l'écriture et la lecture de sauvegardes.

5.3.2 Documentation des macros

5.3.2.1 #define ERREURATTRIBUT "$"

Variable renvoyée par une fonction si la lecture d'un attribut a échoué

5.3.2.2 #define FICHIERLISTESAV "listeDesSauvegardes.txt"

Variable contenant le chemin vers le fichier contenant la liste des sauvegardes stockées

5.4 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/Grille.hpp

classe Grille

Classes

— class Grille
classe qui gère les fonctionnalités de la grille des joueurs de la Bataille Navale.

Macros

— #define null 0
— #define EAU 0
— #define BATEAU 1
— #define BATEAUTOUCHE 2
— #define TENTATIVEREUSSIE 4
— #define TENTATIVERATEE 5

5.4.1 Description détaillée

classe Grille

Auteur

groupe B7

Version

0.1

Date

22 mai 2018

Classe Grille, classe qui gère les fonctionnalités de la grille des joueurs de la Bataille Navale.

5.4.2 Documentation des macros

5.4.2.1 #define BATEAU 1

Variable correspondant à une case où se situe une partie de bateau

5.4.2.2 #define BATEAUTOUCHE 2

Variable correspondant à une case où se situe une partie de bateau qui a été touché par une bombe adverse

5.4.2.3 #define EAU 0

Variable correspondant à une case vide sur la grille

5.4.2.4 #define null 0

Variable correspondant au pointeur vide

5.4.2.5 #define TENTATIVERATEE 5

Variable correspondant à une case sur laquelle le joueur a déjà envoyé une bombe mais qui n'a pas touché de navire adverse

5.4.2.6 #define TENTATIVEREUSSIE 4

Variable correspondant à une case sur laquelle le joueur a déjà envoyé une bombe et qui a touché un navire adverse

5.5 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/JeuBatailleNavale.hpp

classe JeuBatailleNavale

```
#include "Affichage.hpp"
#include "Joueur.hpp"
#include "JoueurHumain.hpp"
#include "JoueurIA.hpp"
#include "GestionSauvegarde.hpp"
```

Classes

— class JeuBatailleNavale
classe qui gère le déroulement du jeu de Bataille Navale.

5.5.1 Description détaillée

classe JeuBatailleNavale

Auteur

groupe B7

Version

0.1

Date

22 mai 2018

Classe JeuBatailleNavale, classe principale qui contrôle la partie entre les deux joueurs.

5.6 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/Joueur.hpp

classe Joueur

```
#include <sstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Grille.hpp"
#include "Bateau.hpp"
#include "Affichage.hpp"
```

Classes

— class Joueur
classe Joueur regroupant les fonctionnalités générales sur le joueur

5.6.1 Description détaillée

classe Joueur

Auteur

Groupe B7

Version

0.1

Date

22 mai 2018 Classe joueur regroupant les fonctionnalités générales sur le joueur

5.7 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/JoueurHumain.hpp

classe JoueurHumain

```
#include "Joueur.hpp"
```

Classes

— class JoueurHumain
classe representant le joueur qui est humain. Cette classe hérite de la classe Joueur

5.7.1 Description détaillée

classe JoueurHumain

Auteur

groupe B7

Version

0.1

Date

13 Avril 2018

Classe JoueurHumain, classe qui contrôle ce que peuvent faire les joueurs humains durant la partie.

5.8 Référence du fichier /home/lucas/Documents/GitHub/Bataille-Navale/src/JoueurIA.hpp

classe JoueurIA

```
#include <math.h>
#include "Joueur.hpp"
```

Classes

— class JoueurIA
classe representant le joueur qui est IA. Cette classe hérite de la classe Joueur

5.8.1 Description détaillée

classe JoueurIA

Auteur

groupe B7

Version

0.1

Date

13 Avril 2018

Classe Joueur IA, classe qui contrôle ce que peuvent faire les joueurs IA durant la partie.

Index