Compte rendu : Bataille navale

PERNY MEZIERE Alfred - 4A IA2R SIR TD8

Le projet est disponible sur le répertoire github publique :

Ce document tend à décrire le fonctionnement général du programme de bataille navale. Le programme lui-même a de nombreux commentaires expliquant le fonctionnement de morceaux précis du code.

Conditions de démarrage :

- Lancer 1 MainServer
- Puis lancer 2 Client
- Ils sont réunis automatiquement par MainServer en une partie.
- Le serveur lance un ThreadPartie pour chaque duo de Client, il peut faire jouer plusieurs parties simultanément.

Ce que le programme fait :

- Initialise la partie (création des bateaux, grilles vides, ...)
- Positionne les bateaux selon les commandes des joueurs (réalisé en simultané par les deux joueurs à l'aide de deux ThreadInit, chacun sur sa grille ; Voir partie "Commandes")
    - Demande les coordonnées pour un bateau donnée (tiré de ships, le tableau de bateaux à placer du joueur) *(géré par ThreadInit)*
    - Vérifie la validité des coordonnées *(géré par ThreadInit)*
    - Vérifie la disponibilité des cases *(géré par Grille)*
    - Remplie la grille personnelle *(géré par Grille)*
    - Il faut que les deux ThreadInit soient terminés pour poursuivre ThreadPartie (utilisation de Thread.join)
- Lance la partie, le ThreadPartie gère la réception des commandes du joueur dont c'est le tour (voir partie "Commandes") et l'affichage des messages.
    - Un joueur ne peut donc interagir avec le programme qu'à son tour
    - Il peut tirer, envoyer un message, consulter ses grilles
    - Tirer met fin au tour du joueur
    - Les données sont accessibles et modifiées par le biais de Partie
- La partie se termine lorsqu'un joueur effectue autant de tirs réussis que la taille cumulée des navires. Un message annonce le gagnant.
- Une fois la partie terminée, il faut déconnecter et reconnecter les clients pour relancer une partie.

Commandes

Phase d'initialisation

Il n'y a qu'une commande disponible pendant la phase d'initialisation, elle permet d'ajouter un navire sur la grille personnelle.

[Ligne][Colonne][Orientation] (sans espace et sans /)

Ligne (0-9) : Ligne de la grille, correspond au point le plus en haut ou à gauche.
Colonne (A-J) : Colonne de la grille, correspond au point le plus en haut ou à gauche.
Orientation (H,V) : Orientation verticale ou horizontale.

Insérer une mauvaise commande ou un placement impossible affichera un message rappelant le format attendu et comment il est interprété.

Phase de jeu

Un joueur ne peut entrer des commandes qu'à son tour une fois la partie commencer. Essayer d'entrer des commandes hors de son tour effectuera toutes les commandes dans leur ordre d'insertion lors du passage du tour. (Ce comportement est non voulu, s'il était contrôlé il s'agirait d'une fonction, mais ce n'est pas le cas.)

Commandes essentielles (Elles commencent toutes par /)

/show target	Affiche la grille de tir.
/show self	Affiche la grille des navires alliés.
/[Colonne][Ligne]	(Sans espace) Tire aux coordonnées indiquées, met fin au tour.

Commande chat (Sans /)

[Message]	Envoie un message au joueur adverse.

Commandes test (Seraient retirées du jeu final)

/win	Met fin à la partie, fait gagner le joueur dont c'est le tour
/end	Met fin au tour

Eléments du jeu

Une partie réunit deux joueurs.
Chaque joueur a un lot de navires (ships) et deux grilles l'une affichant la position de ses navires et les tirs effectués par l'adversaire (selfGrid) et l'autre affichant ses tirs déjà effectués (targetGrid). Il peut consulter ces grilles à tout moment pendant son tour..

Chaque grille fait 11x11 mais n'a que 10x10 cases jouables. L'espace additionnel sert à indiquer les coordonnées.

Remarques

Propreté de la solution

Toute la gestion des bateaux et la modification des grilles a été conçue et écrite très rapidement par manque de temps. Elle mériterait d'être revue, on devrait :

- Mieux séparer les fonctions et les données entre ThreadInit, Partie, Joueur, Grille et Ship.
- Éviter les codes répétés comme la vérification et la conversion de commande en coordonnées.

Pour la même raison, les fonctionnalités sont très limitées, j'aurais aimé faire plus et mieux :

- Empêcher la juxtapositions de navire (vérification des cases adjacentes)
- Signaler quand un bateau entier est coulé (garder en mémoire les cases correspondant à chaque navire, mettre à jour quand touché), ce qui permettrait aussi d'éviter d'avoir à compter le nombre de tirs réussi pour déterminer la victoire
- Travailler avec une classe Coordonnées au lieu d'échanger un tableau d'entier, ce qui rendrait le tout plus lisible et potentiellement plus flexible (notamment pour garder en mémoire les cases d'un navire)

J'ai aussi probablement abusé de IllegalArgumentException, que j'utilise sur presque toutes les fonctions concernant les coordonnées pour indiquer un problème, qu'il s'agisse d'une de coordonnées impossibles (ThreadInit.askCoord<ThreadInit.placeShip), d'une sortie de grille ou d'une superposition de bateaux (Grille.verifySpace<Grille.addShip<ThreadInit.placeShip).