Projet de C Avancé, Bataille Navale

Charles Kempa & Thomas Dignoire

Novembre 2020 - Décembre 2020

Naval_battle / Bataille Navale

Jeu de bataille naval réalisé par Charles Kempa & Thomas Dignoire.

Ce jeu a été réalisé en tant que projet final de C Avancé.

Makefile et lancement

- `make` : pour compiler les fichiers et générer l'exécutable nommé "executable".
- `make clean` : supprimer tout les fichiers intermédiaire à la compilation (déjà fait dans le make).
- `./executable` : lancement du programme.
- `valgrind --tool=memcheck --leak-check=yes --track-origins=yes ./executable` : détails avec Valgrind.

Contenus

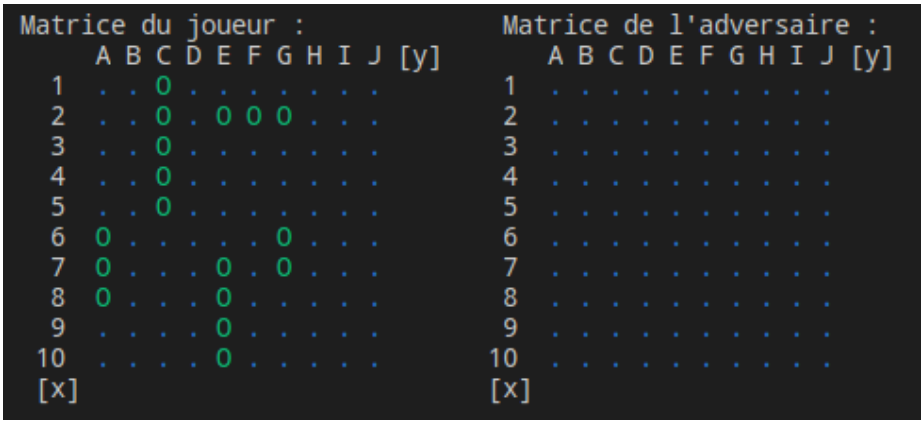

Figure 1: Exemple d'un plateau de jeu

- Une grille de bataille navale qui posséde 10 lignes (de 1 à 10) et 10 colonnes (de A à J), sa taille peut être changer par le joueur.
- Pour effectuer un tir sur une case, les adversaires donnent chacun leur tour les coordonnées de la case qu'ils souhaitent viser (par exemple "4E").
- L'IA répond, par exemple "Tir en 4A dans l'eau." si son coup ne touche aucun bateau, "Tir en 6I a touché." par exemple si un bateau se trouve sur la case, ou encore "Navire est déjà coulé .." si un bateau est coulé.
- Un bateau fait plusieurs cases et est considéré comme coulé si toutes ses cases ont été touchées par l'adversaire.
- Le joueur affronte l'ordinateur dans une partie qui se jouera dans le termenial.
- L'affichage dans la terminal comporte la grille du joueur ainsi que la grille de l'adversaire (clairement identifiées).
- Les cases vides sont marquées par un "." bleu.
- Les cases comportant des navires sont marquées par un "O" vert.
- Les cases comportant des coups "dans l'eau" sont marquées d'un "X" blanc.
- Les cases où un tir a touché sont marquées par un "#" rouge.
- Le joueur a le choix de créer sa flotte ainsi que de choisir où placer ses navires.
- Le joueur peut à tout moment sauvegarder sa partie.
- Le joueur peut en début de jeu choisir de charger une ancienne partie.

Règles du jeu

- Chaque joueur a une flotte (standard) composée de 5 navires : 1 porte-avion (5 cases), 1 croiseur (4 cases), 1 destroyer (3 cases), 1 sous-marin (3 cases), 1 torpilleur (2 cases).
- Les bateaux sont placés aléatoirement sur la grille au début de la partie, de telle sorte qu'ils ne se touchent pas.
- Chacun leur tour, les joueurs vont procéder à un tir qui concerne une seule case.
- Chaque joueurs disposent de 4 tirs spéciaux qui sont chacun lié à un navire. Si le navire concerné est encore en jeu, le joueur peut utiliser chacun de ces tirs spéciaux une et une seule fois dans la partie, à condition qu'au tour précédent il ait touché un navire et qu'il n'ait pas utilisé de tir spécial.

`Tir en ligne (sous-marin)` : permet de viser toute une ligne ou toute une colonne de la grille en une fois.

`Tir en "croix" (croiseur)` : permet de viser en une seule fois un "x" centré sur une case et de 3.

`Tir en "plus" (croiseur)` : permet de viser en une seule fois un "+" centré sur une case et de 3.

`Tir en carré (porte avion)` : permet de viser en une seule fois un carré de 3 cases par 3 centré.

- La partie se termine quand un joueur n'a plus de navires en jeu.

Intelligence Artificielle

- L'ordinateur (IA) joue une case sur deux.
- L'ordinateur (IA) posséde 3 états : R (recherche un navire), O (le navire est trouvé, on détermine l'orientation) et D (on coule le navire).
- L'ordinateur (IA) utilise les tirs spéciaux. Elle a les mêmes contraintes que le joueur : un seul tir de chaque par partie et l'IA doit avoir touché un navire au tour précédent sans y avoir utilisé un tir spécial. Elle utilise un tir spécial de manière aléatoire lors de son état de destruction (D).

Contraintes techniques

- Pour chacun des différents tirs, y compris le tir standard, une fonction renvoie un pointeur sur un tableau allouée dynamiquement des différentes cases cibles, en fonction des coordonnées centrales entrées en paramètres.
- La compilation est effectué par Makefile.
- Valgrind (pour les fuites mémoires) : `in use at exit: 0 bytes in 0 blocks` Il n'y a pas de fuite mémoire. Exemple d'utilisation de valgrind avec les détails : `valgrind --tool=memcheck --leak-check=yes --track-origins=yes ./executable`
- Le projet est à été fait en binôme.

Bonus (4/5)

- Possibilité au joueur de choisir où placer ses navires.
- Possibilité de sauvegarder une partie et d'y revenir.
- Possibilité de choisir 5 navires dans une listes de navires (tous entre 1 et 5 cases), en choisissant éventuellement plusieurs fois le même type de navire.
- La taille de la grille peut être modifié.

Information et liens utiles

- (gitHub)[https://github.com/iDrack/Naval_Battle] : https://github.com/iDrack/Naval_Battle
- (Trello)[https://trello.com/b/zV3UGXll/bataille-navale] : https://trello.com/b/zV3UGXll/bataille-navale
- Les fonctions possédent une en-tête avec une description de celle-ci avec le détail de ces paramètres. N'hésitez pas à jetter un oeil sur le code, propre et commenté.