Bataille navale

BONNAZ Aymeric, MASSÉ Robin

16 août 2025

Nom apprenant : _____

Date de réalisation : _____

Question:	1	2	3	4	5	6	7	8	Total
Points:	1	3	2	2	2	1	2	1	14
Score:									

1 Énoncé

La bataille navale est un jeu de société dans lequel deux joueurs doivent placer plusieurs navires sur une grille secrète et tenter de toucher les navires de l'adversaire. Le présent exercice présentera la visualisation d'un seul joueur et ne prendra pas en compte le point de vue du second joueur. Le jeu se joue sur une grille de taille 10 * 10 cases.

Avant le début de la partie, il est nécessaire de placer différents navires de tailles différentes, de manière horizontale ou verticale. Pour simuler ce placement, celui-ci se fera de manière aléatoire. Les navires sont cachés au joueur.

En début de partie, le contenu de toutes les cases est masqué. Le but du jeu est de choisir à chaque coup une case non jouée et de toucher chacun des navires. Lorsqu'une case est jouée, le statut de celle-ci est révélé. Lorsque toutes les cases constituant un navire sont révélées, le navire est coulé. La partie est finie lorsque l'ensemble des navires sont coulés.

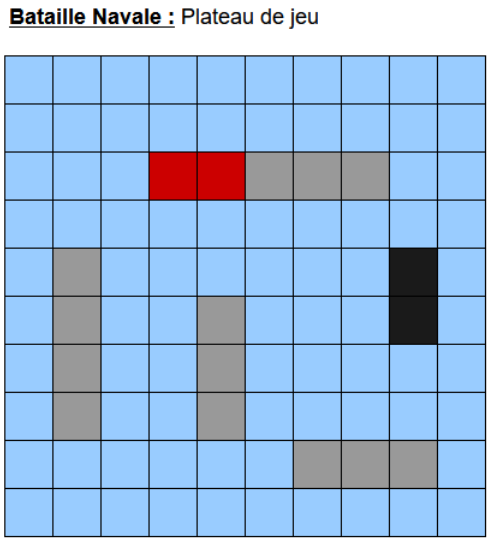

FIGURE 1 – Détermination des voisins d'une case de la grille

2 Exercice

L'exercice est divisé en deux grands blocs. La première partie aborde l'implémentation des différents éléments du jeu comme une position ,un bateau et leurs méthodes respectives. La seconde partie s'occupera du plateau de jeu et de son déroulement en tant que tel.

Position et Bateau : 6 pts

Position :
Pour commencer, il est nécessaire d'implémenter la structure **Position** définie ci-dessous. Une instance de celle-ci correspond à une case du plateau de bataille navale.
Elle est caractérisée par une position en deux dimensions (x, y) et un état. Les valeurs de x et y sont comprises entre 0 et 9. L'état peut avoir l'une des valeurs suivantes : Caché (état par défaut), Touché, Coulé et Plouf. Les méthodes **Touché**(), **Coulé**() et **Plouf**() permettent de mettre l'état correspondant sur l'instance de la **Position**.

Position :
- bool : **Touché**()
- bool : **Coulé**()
- bool : **Plouf**()

1. (1 point) Implémenter les méthodes d'état :
bool : **Touché**(), bool : **Coulé**(), bool : **Plouf**().

Bateau :

La structure **Bateau** est celle permettant de créer les différentes instances de bateaux à placer sur le plateau de jeu de la bataille navale.

Elle est caractérisée par un nom, une liste de **Positions** et une taille qui donne le nombre de cases couvertes par l'instance sur la grille. La cohérence des différentes **Positions** sera traitée lors de la seconde partie de cet exercice.

La méthode **Cible**(x,y) retourne la position (x,y) si celle-ci appartient au bateau concerné. Son implémentation est fournie.

Bateau :

- bool : **EstCoulé**()
- bool : **Touché**(int x, int y)
- **Position** : **Cible**(int x, int y)

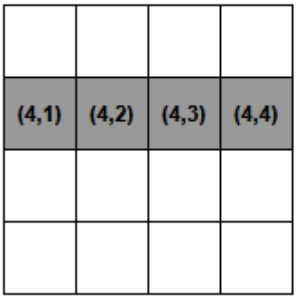

FIGURE 2 – Bateau en position horizontale

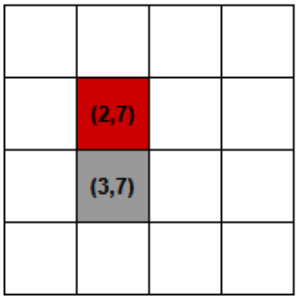

FIGURE 3 – Bateau en position verticale

La méthode **EstCoulé**() indique si le bateau a été coulé ou non.

2. Par le fond :
   (a) (1 point) Dans quelle circonstance un bateau est-il coulé ?

   (b) (2 points) Implémenter la méthode bool : **EstCoulé**().

La méthode **Touché**(x,y) modifie l'état de la position (x, y) à touché si celle-ci appartient au bateau.

3. (2 points) Implémenter la méthode : bool : **Touché**(int x, int y).

Plateau de jeu et partie : 8 pts

Cette seconde partie s'attaque à l'implémentation du plateau de jeu et le déroulement de la partie a proprement dite. Comme indiqué dans l'énoncé, deux grandes phases seront traitées à savoir le placement aléatoire des différents bateaux puis l'exécution des différents coups de la partie. La structure **Plateau** contient les différentes méthodes à implémenter de cette dernière partie ainsi qu'un tableau à deux dimensions de **Positions** et une liste de **Bateaux**.

Plateau :

- bool : **PlacerBateau**(int x, int y, int taille, bool estVertical)
- void : **CreationPlateau**()
- void : **AfficherPlateau**()
- void : **Viser**(int x, int y)
- bool : **FindePartie**()
- void : **LancementPartie**()

Positionnement :
Pour commencer, la première des choses est le positionnement des bateaux. Cinq bateaux sont à disposer sur la grille de jeu (un de taille 5, un de taille 4, deux de taille 3 et un de taille 2).
Avant d'implémenter la méthode **CreationPlateau**(), il est nécessaire de déterminer s'il est possible de placer un bateau dont la proue est à la position (x, y), de taille et de positionnement (verticalité, horizontalité) donnés. La principale contrainte est que deux bateaux ne peuvent pas avoir de contact **direct** (pas de partage de cases), de contact **latéral** (juste à côté, verticalement ou horizontalement) et de contact **diagonal**.

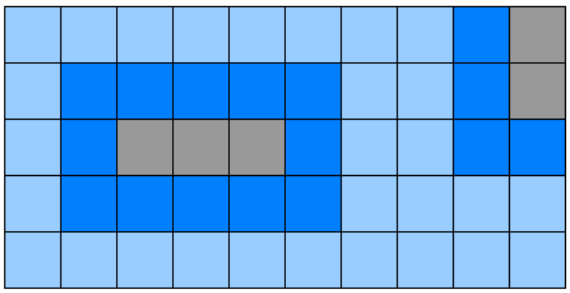

FIGURE 4 – Placement d'un bateau, les cases en bleu foncé sont interdites pour le placement d'autres bateaux.

4. (2 points) Implémenter la méthode suivante (le statut du placement est retourné) :
   bool : **PlacerBateau**(int x, int y, int taille, bool estVertical)

Pour le positionnement des bateaux, nous proposons l'algorithme suivant pour l'implémentation de la méthode **CreationPlateau**() :

- Initialiser le tableau de taille 10 * 10 avec l'ensemble des positions à l'état Caché.
- Pour chacun des bateaux dont la taille est fixée à l'avance, choisissez la position de la proue et le type de positionnement au hasard.
- Tester si le bateau généré peut être positionné. S'il l'est, ajoutez le à la liste de bateaux.
- Répéter pour un bateau donné l'opération jusqu'à ce qu'il soit possible de le placer.

5. (2 points) Implémenter la méthode suivante : `void : **CreationPlateau**()`

Suggestion : La classe **Random** permet de générer des entiers aléatoires.

Déroulement :
L'implémentation de la méthode **AfficherPlateau**() est fournie et consiste à afficher l'état des 5 bateaux positionnés et l'état de la grille.

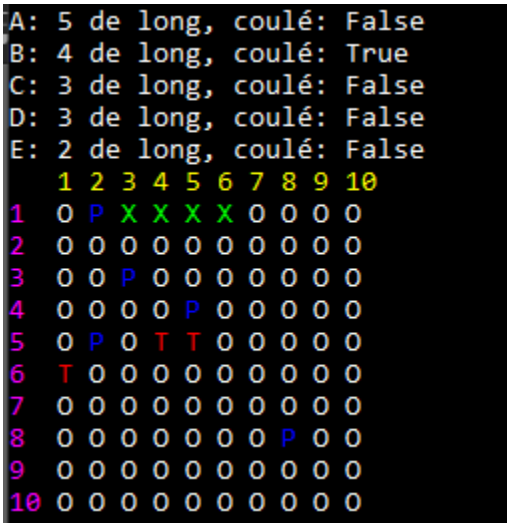

FIGURE 5 – Affichage d'un plateau au cours d'une partie

La méthode **Viser**(x, y) consiste à tester la position à l'emplacement (x, y) et modifier son état en fonction de la présence d'un bateau ou non (Touché ou Plouf).

6. (1 point) Implémenter la méthode : `void : **Viser**(int x, int y)`

Pour le bon déroulement du jeu, il est nécessaire de déterminer le moment où celle-ci se termine.

7. Fin de partie :
   (a) (1 point) Dans quelle situation une partie de bataille navale est terminée ?

(b) (1 point) Implémenter la méthode `bool : **FindePartie**()`.

Finissons par la méthode **LancementPartie** qui est déjà partiellement donnée. La seule partie à compléter concerne la lecture de l'entrée utilisateur. Le format attendu de celle-ci est ligne,colonne. Les valeurs de ligne et colonne sont comprises entre 1 et 10 (elles seront décrémentées de 1 par rapport aux positions (x, y).

8. (1 point) Implémenter la méthode : `void : **LancementPartie**()`