

## Le Mans Université

Licence Informatique *2ème année*

Module projet

**Bataille navale :**

**[https://github.com/KillianDEUX/bataille-  
navale](https://github.com/KillianDEUX/bataille-navale)**

DEUX Killian

GERARD Jules

DEROUET Corentin

20 avril 2019

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Organisation</b>	<b>4</b>
<b>3</b>	<b>Conception/Analyse</b>	<b>5</b>
3.1	Explication des règles . . . . .	5
3.2	Explication des fonctionnalités . . . . .	7
<b>4</b>	<b>Développement</b>	<b>8</b>
4.1	Présentation de la conception d'un bateau . . . . .	8
4.1.1	Présentation d'un bateau . . . . .	8
4.1.2	Présentation des listes de bateaux . . . . .	8
4.1.3	Fonctions de modification . . . . .	8
4.2	Présentation des matrices . . . . .	10
4.2.1	La matrice "joueur" . . . . .	10
4.2.2	La matrice "adverse" . . . . .	10
4.3	Présentation des pions . . . . .	11
4.4	Présentation de l'Intelligence Artificielle . . . . .	12
4.4.1	Placement des bateaux . . . . .	12
4.4.2	Choix des cases de tir . . . . .	12
4.5	Présentation du réseau . . . . .	13
4.5.1	Serveur . . . . .	13
4.5.2	Clients . . . . .	14
4.6	Programmation modulaire . . . . .	15
4.6.1	Fichiers utilisés . . . . .	15
4.6.2	Fichiers généraux . . . . .	15
<b>5</b>	<b>Résultat</b>	<b>16</b>
<b>6</b>	<b>Conclusion</b>	<b>17</b>
<b>7</b>	<b>Annexes</b>	<b>18</b>

# 1 Introduction

Dans le cadre de l'unité d'enseignement « Conduite de projets », nous avons du programmer un jeu en langage C à plusieurs.

L'objectif de ce projet est de mettre à contribution les enseignements appris depuis le début de notre cursus comme « Algorithmique ». Cela a permis de nous former au travail en groupe, une nouveauté pour nous.

Nous avons choisi comme sujet la « bataille navale ». C'est un jeu de société qui, dans sa forme originale, se joue à deux joueurs dans lequel chacun d'entre eux doivent placer des bateaux sur une grille, inconnue de l'adversaire. La partie se termine quand l'un des joueurs trouve l'ensemble des bateaux de son adversaire avant que tous les siens ne soient découverts.

Chaque joueur dispose alors de deux grilles. La première servira à placer les bateaux et la seconde à placer des pions. Cette seconde grille sera une vue de la grille adverse.

Lors d'un tir à des coordonnées précises dans la grille adverse, l'ennemi indiquera si le tir a atterri « dans l'eau », a « touché » ou a « coulé » un bateau. C'est en fonction de ces informations que l'on peut placer un pion, de différente couleur en fonction du tir. Afin d'enrichir le jeu, nous avons changé quelques règles définies en début de projet.

## 2 Organisation

La planification du projet a été articulée autour de l'outil "Projects" de GitHub.

Celui-ci permet, via un système de "cartes", d'assigner des tâches à chaque membre du groupe.

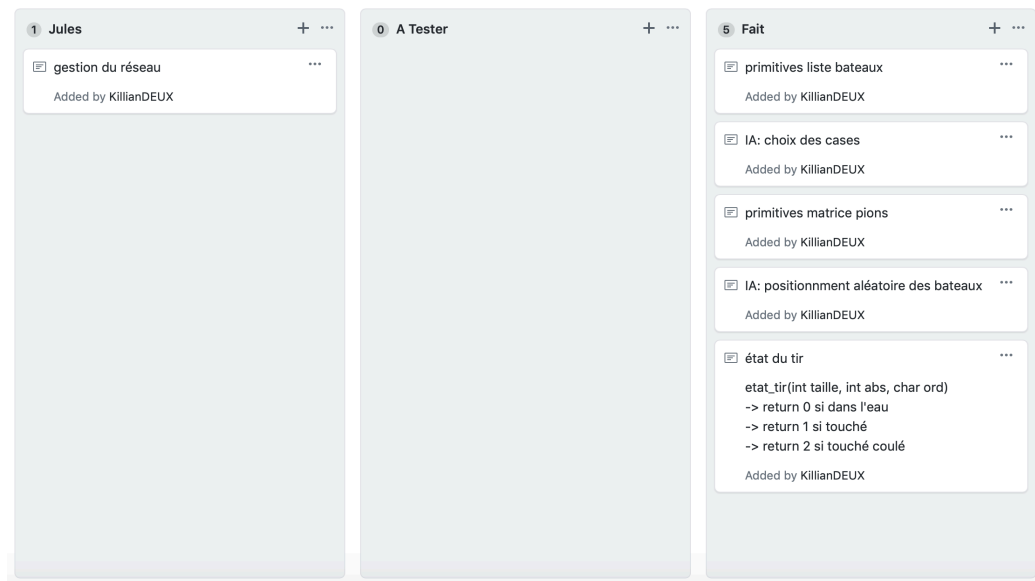


Figure 1 - Planificateur GitHub

De plus, l'outil "Code" permet de récupérer et de mettre à jour les changements effectués dans les différents fichiers.

La répartition globale est la suivante :

- Killian : gestion des bateaux et des opérations sur les matrices
- Corentin : gestion des matrices de pions et du mode jeu IA
- Jules : protocole de communication et gestion du réseau

## 3 Conception/Analyse

### 3.1 Explication des règles

Les règles de la bataille navale dans l'introduction sont fixes. Or nous voulions pouvoir adapter celles-ci aux exigences du ou des utilisateur(s).

Ainsi celui-ci est en mesure de changer les règles basiques pour chaque partie de jeu qu'il souhaite effectuer. Certaines règles restent inchangées : deux bateaux ne peuvent ni se toucher ni se croiser ni être placés en diagonale.

Le joueur ne peut pas tirer deux fois au même endroit car la case contient déjà un pion et donc ne peut pas bouger ses bateaux de place au cours de la partie.

Les modifications sont :

- changement de la taille du plateau de joueur ,
- choix du nombre de bateaux en fonction de la taille du plateau choisi précédemment :
- le nombre de joueur, de 1 à 5,
- choix de la taille de chaque bateau, mais toujours définie entre 1 et 5.

Chaque bateau de taille différente possède un type particulier, respectivement :

- Mine,
- Torpilleur,
- Sous-Marin,
- Croiseur,
- Porte-avion.

Pour calculer le nombre de navires pour une grille variable, une fonction a été utilisée.

Ci-dessous la courbe obtenue avec Geogebra en fonction de 3 points. Les X sont déterminés par la taille de la matrice et les Y par le nombre de bateaux maximum voulu. Les 3 points sont donc 2 bateaux pour une matrice de 5x5, 5 bateaux pour une matrice de 10x10 et finalement 20 bateaux pour une matrice de 20x20.

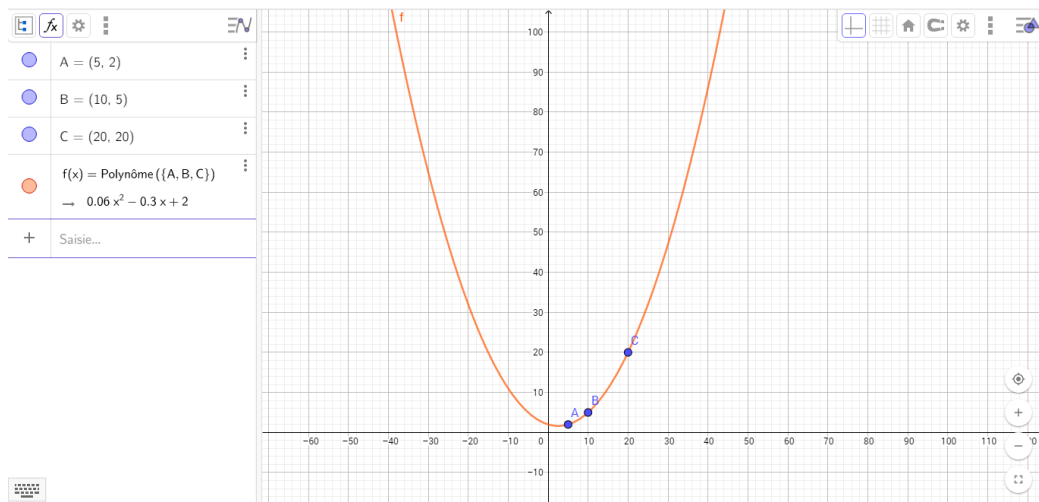


Figure 2 - Courbe Geogebra

## 3.2 Explication des fonctionnalités

En fonction des nouvelles règles ajoutées et des règles basiques de la bataille navale, nous avons dû adapter notre code.

Le choix des cases se fait par lettre et chiffre. Pour une question de praticité, les cases sont numérotées, en ligne et en colonne, de 1 à n et de 1 à m (n et m étant demandés au joueur).

Le joueur peut également choisir un plateau de jeu carré ou non.

Comme l'énoncent les règles de l'introduction, chaque joueur possède 2 plateaux. L'un comprenant ses bateaux et l'autre, ses cases attaquées. Chaque case du plateau bateaux possède une caractéristique, soit :

- une case est vide,
- une case touchée,
- un bateau normal,
- un bateau touché,
- un bateau coulé.

Et chaque case du plateau adverse possède d'autres caractéristiques, soit :

- pas de pion,
- un pion blanc,
- un pion rouge.

## 4 Développement

### 4.1 Présentation de la conception d'un bateau

#### 4.1.1 Présentation d'un bateau

Un bateau est constitué de :

- sa taille
- son type
- ses premières coordonnées (là où il débute)
- sa direction (verticale ou horizontale)
- son état (touché ou coulé)
- nombre de fois qu'il a été touché par un joueur adverse

Toutes ses données sont nécessaires pour les traitements et utilisées tout au long du code.

#### 4.1.2 Présentation des listes de bateaux

Le premier joueur définit une série de bateaux qui sera reprise par les autres participants. Nous avons donc besoin de stocker ces bateaux dans des listes (vues au premier semestre) pour traiter efficacement ces éléments. Une liste possède un élément courant permettant de la parcourir ainsi qu'un "drapeau" signifiant la fin de celle-ci. Evidemment, nous avons dû adapter le système de listes pour qu'il soit applicable à nos bateaux.

#### 4.1.3 Fonctions de modification

Toutes les fonctions de modification de listes ou de bateaux sont placées dans des fichiers séparés pour une réutilisation facile.

Lors de la mise en place des bateaux d'un joueur, une vérification est nécessaire concernant leur placement. La fonction `parcours_matrice` est utilisée lors de ce processus. Elle parcourt la liste de bateaux déjà placés et récupère toutes les cases où le joueur ne pourra pas poser le navire. Donc pour chaque bateau, la fonction vérifie s'il est placé à la verticale ou à l'horizontale et parcourt toutes les cases autour de lui dans le but de les rentrer dans un tableau de cases non disponibles.



Une deuxième fonction appelée `verif_placement_bateau`, permet de vérifier que les coordonnées transmises, et donc souhaitées, n'interfèrent pas avec les coordonnées d'un autre bateau déjà placé. Donc pour chaque coordonnée voulue du bateau, la fonction parcourt entièrement le tableau de cases non disponibles.

## 4.2 Présentation des matrices

Nous avons eu besoin de créer deux types de matrices. La première matrice servira à placer nos propres bateaux, c'est la matrice "joueur". La seconde matrice servira à placer des pions en fonction de l'état du tir qui comprend trois choix possibles : touché, coulé ou dans l'eau. C'est la matrice "adverse".

### 4.2.1 La matrice "joueur"

La matrice "joueur" se compose d'un nombre de lignes et de colonnes renseigné par l'utilisateur en début de partie. Le joueur a le choix de donner une taille de grille minimale (5 lignes par 5 colonnes pour avoir un plateau jouable) et une taille maximale de 26x26. Les lignes étant modélisées par les lettres de l'alphabet.

Grâce à ces informations, le programme créé alors un tableau en conséquence. Chaque case sera initialisée avec des cases vides. Le premier joueur décide du nombre de bateaux et de leur taille pour tous les autres participants. Il place alors, en toute discrétion, ses propres bateaux en choisissant la direction et les coordonnées de chacun d'eux. Les autres joueurs ou l'intelligence artificielle placent ensuite leur flotte. Pour une meilleure compréhension par l'utilisateur, une légende est affichée. Celle-ci comprend la définition de chaque symbole en fonction de l'état de la case. Lors d'un tir d'un adversaire sur la flotte du joueur, les informations sur la case se mettent à jour. Les changements se font en même temps sur la grille adverse.

### 4.2.2 La matrice "adverse"

De plus, une autre matrice est créée. Il s'agit de la matrice adverse, elle possède la même taille que la matrice joueur. Cette matrice représente la vision partielle de la matrice de l'adversaire.

Elle contient donc autant de lignes et de colonnes que la matrice "joueur". Pour bien différencier les deux types de matrices, l'affichage est complètement différent. Comme précédemment, la légende aide l'utilisateur à comprendre les tirs qu'il a fait sur la ou les grille(s) "adverse(s)". Une fois un tir effectué sur la grille de l'adversaire choisi (si un choix est possible), la case en question se transforme pour indiquer au joueur l'état de son tir. De plus avant chaque affichage de la matrice, le programme indique par une phrase si le tir est efficace ou nul.

### 4.3 Présentation des pions

La matrice "adverse" possède plusieurs caractéristiques. Les principales fonctionnalités de cette matrice sont d'indiquer l'état du tir et placer un pion, symbolisé par un caractère défini dans la légende en fonction de la case choisie.

Une règle nous interdit de placer des bateaux les uns à côtés des autres. Pour faciliter l'expérience utilisateur, lorsqu'un bateau est touché, une fonction transforme toutes les cases autour comme 'dans l'eau'.

Une fois le bateau coulé, la fonction repère ce bateau, puis va sélectionner une par une chaque case présente autour et va remplacer le contenu de cette case par un pion "dans l'eau". Elle vérifiera cependant que la case est bien dans la grille car si un bateau est le long d'un bord, certaines cases autour se trouvent hors de la zone de jeu.

## 4.4 Présentation de l'Intelligence Artificielle

En temps normal, on ne peut pas jouer en solitaire à la bataille navale. Or une des fonctionnalités de notre code est le mode solo contre une IA. Cette IA se met en place lorsque le joueur indique qu'il veut jouer seul.

### 4.4.1 Placement des bateaux

L'IA place ses bateaux aléatoirement sur le plateau tout en vérifiant qu'aucun ne sorte de la grille ou ne touche un autre (en utilisant la même fonction que pour les joueurs).

### 4.4.2 Choix des cases de tir

Lors de son premier tour, l'IA choisit une case du plateau aléatoirement. Puis, lors du tour suivant et pour tous les tirs, elle commence par vérifier si un bateau a déjà été touché et non coulé.

Si c'est le cas, elle effectue un tir sur toutes les cases qui entourent le bateau déjà touché. Elle commence par les cases situées au nord de cette embarcation puis par l'est ensuite par le sud et pour finir, l'ouest.

Si aucun bateau n'a été touché, l'IA choisit une case la plus éloignée possible de toutes les cases qui ont déjà été sélectionnées.

Si aucune case n'a été trouvée, elle essaye de trouver une case pseudo aléatoirement. C'est-à-dire qu'elle trouve une case aléatoirement mais recommence tant qu'elle reste collée à une autre case déjà touchée.

Si elle n'en trouve toujours pas, cette fois c'est une case totalement aléatoire qui est trouvée.

## 4.5 Présentation du réseau

Dans ce mode de jeu, on dissocie deux exécutable : le serveur et les différents clients. Le serveur, en plus de s'occuper des communications, présente une vue d'ensemble de la partie en cours. Le client propose l'interface de jeu.

### 4.5.1 Serveur

Pour permettre l'échange de données entre les différents utilisateurs, nous avons choisi d'implanter le réseau dans ce projet grâce au protocole TCP. Celui-ci présente un avantage important : le contrôle de l'information reçue.

Comme le jeu doit supporter un nombre variable de joueur, le serveur est en mesure d'ouvrir plusieurs sockets clients.

Pour cela, il a été nécessaire de faire un tableau pour établir les canaux de communication de chaque client. Ainsi, le canal d'indice 0, correspond au 1er client qui s'est connecté et ainsi de suite.

No Joueur	1	2	3	4	5
Socket	client_fd[0]	client_fd[1]	client_fd[2]	client_fd[3]	client_fd[4]

Figure 3 - Equivalence joueur et indice client

Ces canaux vont permettre d'envoyer des variables par le réseau via les fonctions send et recv de la bibliothèque « sys/socket.h ».

Ce mode de fonctionnement est adapté au jeu de la bataille navale tour par tour car la fonction recv est bloquante.

La main est rendu à l'utilisateur dès que l'information est reçue, dans le cas contraire, celui-ci reste en attente.

Du côté client, un unique socket est établi vers le serveur. Les clients ne peuvent donc pas communiquer directement entre eux. Cette contrainte implique de nombreux envois.

### 4.5.2 Clients

Il est nécessaire de fournir l'information au client pour trouver le serveur. Notre choix s'est porté sur l'adresse IP pour plus de simplicité au lieu du nom d'hôte. Celle-ci est demandée à l'utilisateur avant d'initier la connexion. Le port est quant à lui défini en constante.

Pour plus de cohérence et d'équité, seul le premier joueur (le premier client connecté au serveur) choisi la taille de la grille et la liste de bateaux. Ces choix sont mémorisés et redistribués aux autres clients. Il leur reste la liberté de placer leurs navires.

Le mode de jeu réseau présente une spécificité : la grille de pions des joueurs ne varie pas en fonction du joueur attaquant mais en fonction du joueur attaqué.

Ce choix, qui diffère de la règle originale, permet plus de stratégie lors des parties à plus de 2 joueurs.

D'autre part, toujours dans le cas d'une manche à plus de 2 joueurs, le jeu s'achève lorsqu'un joueur a perdu l'intégralité de sa flotte.

## **4.6 Programmation modulaire**

### **4.6.1 Fichiers utilisés**

Pour pouvoir utiliser toutes nos fonctions de traitement comme par exemple le traitement des listes ou des bateaux, les fichiers générant l'IA ou encore les fichiers gérant les matrices de pions, un makefile est nécessaire. Le makefile demande la création de fichiers .h contenant la déclaration des structures, par exemple de quoi est composé un bateau, ainsi que les primitives de chaque fonction.

Il permet de compiler, c'est-à-dire de transformer les fichiers, codés en langage C, en fichiers contenant le langage machine approprié. Ces fichiers transformés finissant par .o sont ensuite utilisés par le serveur et le client.

### **4.6.2 Fichiers généraux**

Le serveur et le client sont les deux fichiers utilisés par l'utilisateur lors de l'exécution. Pour jouer en solitaire et donc, contre l'IA l'exécution du serveur suffit. Sinon il faut exécuter le serveur ainsi que, pour chacun des joueurs, autant de client que de nombre de joueur. Chaque joueur ouvrant le sien. Le serveur et le client sont donc compilés en utilisant tous les fichiers .o nécessaires et générés précédemment.

## 5 Résultat

A l'heure actuelle, le programme est fonctionnel et propose de manière indépendante les deux modes de jeu que nous souhaitions : solo contre l'IA et multijoueur en réseau. Les règles personnalisées sont toutes présentes, ce qui apporte un peu de nouveautés au jeu classique. Néanmoins, certaines fonctionnalités n'ont pas été implémentées.

Le plus gros point noir est l'absence de l'interface graphique initialement prévue. Cette dernière aurait permis une bien meilleure expérience utilisateur. Par exemple, il aurait été plus naturel et ergonomique de devoir cliquer sur une case au lieu d'entrer des coordonnées au sein d'une console.

Nous aurions également voulu ajouter un "timer" afin d'imposer une limite de temps sur un tour. Ainsi, l'attaquant qui dépasserait le temps imparti pour choisir sa cible serait obligé de passer son tour. Cette limite éviterait de trop longues attentes pour les autre(s) participant(s).

Une autre option qui avait été envisagée était de pouvoir déplacer un bateau au lieu d'attaquer un joueur. Cette option aurait rajoutée un aspect stratégique intéressant. Chaque joueur pourrait alors déplacer un bateau à condition qu'il ne soit pas déjà touché.

De plus, il serait intéressant que le mode de jeu IA puisse prendre la relève en cas d'un joueur réseau inactif ou encore en cas de perte de connexion.

Malgré quelques difficultés à la réalisation finale de notre code, nous avons l'idée, qui n'a malheureusement pas pu aboutir, de faire une bataille navale mais en 3D. C'est à dire avec plusieurs types de sous-marins ainsi que des avions. Donc le but était de faire 3 plateaux de jeu pour chaque joueur.

Enfin, il aurait été souhaitable, de mettre en place des pseudos et des highscores.



## 6 Conclusion

Ce projet, assez conséquent, nous a permis de découvrir comment travailler à plusieurs. En effet, nous avons dû nous organiser et prévoir une répartition des tâches. Les contraintes de temps ont été les plus difficiles. À cause de celles-ci, nous n'avons pas réussi à ajouter toutes les fonctionnalités voulues, en particulier la SDL. Nous avons pourtant réussi à avoir un programme jouable sur terminal, que ce soit contre une IA ou en réseau, avec les règles prévues dans le cahier des charges. Même si, au niveau du réseau, l'optimisation n'est pas complète, il est quand même possible de jouer à la bataille navale jusqu'à 5 joueurs. Nous avons perçu les difficultés à travailler en groupe.

Chaque personne travaillait sur une partie du code et lors de la mise en commun, même si nous nous étions mis d'accord sur des fonctionnalités, certaines fonctions n'étaient pas compatibles entre elles. À force de débogage, nous avons tout de même pu finir par avoir un jeu auquel il est possible de jouer. S'il fallait continuer ce projet, nous pourrions alors améliorer certains points importants comme l'interface et l'optimisation du réseau. Une fonctionnalité intéressante aurait été de ne pas arrêter la boucle de jeu lorsqu'un joueur perd.

## 7 Annexes

```
-- BATAILLE NAVALE EN RESEAU - CLIENT --
Entrez l'@IP du serveur : 127.0.0.1
-- CONNECTE --
La partie est constituée de 2 joueurs
Tu es le joueur 1
Quelle largeur de la grille souhaitez-vous ? (entre 5 et 26)
10
Quelle hauteur de la grille souhaitez-vous ?(entre 5 et 26)
10
-----
Voici votre grille
  1  2  3  4  5  6  7  8  9 10
A [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
B [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
D [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
E [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
F [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
G [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
H [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
I [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
J [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Combien de bateau voulez-vous avoir ? (entre 1 et 5) : █
```

Capture d'écran de la connexion du joueur 1 au serveur

```

Vous allez placer le bateau numero 2 , il s'agit d'un(e) PORTE-AVION , de taille 5
Dans quelle direction voulez-vous placer le bateau ? (donnez un entier selon : vertical = 1, horizontal = 2)
2
Quelles sont les coordonnees a laquelle vous voulez placer le bateau ?(entre A et K)
Coordonnée lettre = I
Coordonnée chiffre = 3
Le bateau a ete place
  1  2  3  4  5  6  7  8  9 10
A [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
B [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
C [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
D [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
E [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
F [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
G [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
H [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
I [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
J [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

```

Capture d'écran du placement de 3 bateaux

---

A votre tour,  
 Quelle case voulez-vous attaquer ( de la forme "ligne colonne")?

Légende :

- : la case ne contient rien
- o : la case a été touché
- . : la case contient un bout de bateau non touché
- \* : la case contient un bout de bateau touché
- ! : la case contient un bout de bateau coulé

Pour la grille de tir :

- . : la case n'a pas encore été visé
- 0 : Le tir est dans l'eau
- X : Le tir vient de toucher un bateau

Affichage de votre tir :

RATÉ

	1	2	3	4	5	6	7	8	9	10
A	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.
E	.	.	.	.	.	.	.	0	.	.
F	.	.	.	.	.	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	.
I	.	.	.	.	.	.	.	.	.	.
J	.	.	.	.	.	.	.	.	.	.

Au tour de l'IA

RATÉ

Affichage de votre plateau :

	1	2	3	4	5	6	7	8	9	10
A	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[.]
B	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
C	[ ]	[ ]	[ ]	[.]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
D	[ ]	[ ]	[ ]	[.]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
E	[ ]	[ ]	[ ]	[.]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
F	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
G	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[o]	[ ]
H	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
I	[ ]	[ ]	[.]	[.]	[.]	[.]	[.]	[ ]	[ ]	[ ]
J	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]

Capture d'écran de l'attaque d'un bateau

```

-----
Tour de 1

Quelle case voulez-vous attaquer (de la forme "ligne colonne")? B 2

Vous avez COULÉ votre cible

-----

Tentatives d'attaque sur le joueur 2 :

  1  2  3  4  5
A 0  0  0  .  .
B 0  X  0  .  .
C 0  0  0  .  .
D .  .  .  .  .
E .  .  .  .  .

Affichage de votre flotte :

  1  2  3  4  5
A [.] [ ] [ ] [ ] [ ]
B [ ] [ ] [ ] [ ] [ ]
C [ ] [ ] [ ] [ ] [ ]
D [ ] [ ] [ ] [ ] [ ]
E [ ] [ ] [ ] [ ] [ ]

Le joueur perdant est le joueur 2
AU REVOIR !

```

Capture d'écran de la fin de partie