# Projet algorithmique :
# Bataille Naval

Agnés Blanchet Julien Richard WilliamKaczmarek

8 décembre 2019

# Table des matières

# I Introduction

Pour le premier semestre d'algorithmique nous avons du réaliser un premier projet en groupe de 2-3 personnes. Le but de ce projet est de développer en C un programme qui va permettre à 1 (contre une intelligence artificiel) ou 2 joueurs de jouer à la Bataille navale. Le cahier des charges nous a guidé vers la réalisation des différentes taches à réaliser.
Nous nous sommes alors organisé, afin de réaliser le maximun et produire un programme le plus fidèle possible à la Bataille naval, en respectant le cahier des charges à notre disposition.

Figure 1 – Jeu de bataille naval

# II Développement

La Bataille navale (ou « Battleship » en anglais) est un jeu de société dans lequel deux joueurs doivent placer des navires sur une grille tenue secrète et tenter de couler les navires adverses. Le gagnant est celui qui parvient à torpiller complètement les navires de l'adversaire avant que tous les siens ne le soient. Chaque joueur possède les mêmes navires, dont le nombre et le type dépendent des règles du jeu choisies. La taille de la grille peut varier selon le jeu aussi. Nous avons donc cherchés dans un premier temps à réaliser un jeu jouable pour 2 joueurs humains avec soit des paramètres par défaut : grille de 10*10 et chacun des joueurs à 1 portes avions de 6 cases, 2 croiseurs de 4 cases, 3 sous-marins de 3 cases et enfin 4 torpilleurs de 2 cases. Ou des paramètres personnalisable, dans ce cas-là tous les paramètres sont modifiables avant le début de la partie : taille de la grille N*N, Nombre et taille de chaque type de bateaux (porte avions, croiseurs sous-marins, torpilleurs).

Tout d'abord, nous avons commencé par bien lire le sujet plusieurs fois afin de bien comprendre ce qui nous était demandé, dans le but d'avoir une idée de base sur notre sujet et par la suite de poser nos idées sur papier, de répartir les tâches afin de réaliser nos diverses fonctions.
Nous nous sommes séparés le travails en plusieurs parties :
- La création des structures et de nos conastantes (*inc.h*)
- L'affichage des grilles (*affichage.c*)
- La verifications du placement des bateaux selon les règles du jeu (*voisin.c*)
- La gestion des grilles, la création, l'initialisation, le placement des bateaux (grille.c)
- La gestion d'une partie et de chaque tour, une intelligence artificiel (*partie.c*)
- Le menu, qui permet de lancer et initialiser les paramètres de la partie personnalisable ou par défaut (*menuParametres.c*)

Afin de pouvoir bien attaquer le projet, on a de suite créé nos constantes, et la façon dont on allait gérer les différentes données relatives aux joueurs, aux bateaux. C'est donc dans *inc.h* que nous avons tout réunis. Les structures, qui gèrent les joueurs : *struct joueur*. Où sont stockées toutes les infos relatives aux joueurs de la partie. En effet, cette structure contient : sa grille avec ses bateaux, la taille de cette grille et enfin son statut (s'il est humain ou si c'est un IA).

```
struct coordonnees {
    int  int_ligne;
    int  int_colonne;
    int int_precedent;
    int int_card;
};


struct joueur{
    int int_hum_ia;
    int int_taille;
    int** ttint_grille;
};
```

Figure 2 – Exemple de nos structures

# III Déroulement d'une partie type

## III.1 Choix des joueurs et des paramètres

Au commencement d'une partie, l'utilisateur est invité à choisir si sa partie utilisera des paramètres par défaut (ceux d'une partie classique de bataille navale) ou personnaliser ses paramètres (les tailles des bateaux et le nombre de bateaux).
Une fois ce choix effectué, l'utilisateur doit décider s'il affronte un autre joueur humain ou s'il joue contre notre Intelligence artificiel.
Toutes les fonctions permettant la réalisation de ces étapes se trouvent dans*MenuParamètre.h*

## III.2 Tour par tour

La partie débute, le premier joueur doit dans un premier temps décider si son bateau sera posé verticalement ou horizontalement. Ensuite, il faut qu'il entre les coordonnées de la première case de son bateau, ça jusqu'à que tous les bateaux soit posés.
Maintenant le jeu commence, le joueur 1 doit tenter un tir, pour cela il entre les coordonnées de la case sur laquelle il souhaite tirer. Si le tir a touché, on le lui indique, mais on ne le fait pas rejouer, contrairement aux règles de base. On sait comment il faut faire (en créant un type entier dans notre struct joueur qui stock une donnée qui indique si oui ou non, il a touché au tour précédent.). Malheureusement, nous n'avons pas eu le temps d'ajouter ça à notre code, par manque de temps et trop de bug sur l'intelligence artificielle. La partie se déroule, chacun des joueurs tir, un bateau s'affiche avec un B, si il a été touché par l'adversaire une croix rouge s'affiche, et si l'entiereté du bateau a été touché les croix deviennent vertes.

## III.3 Victoire/Gagnant

À chaque tour, nous comparons la grille du joueur avec celle où joue son adversaire. Si les deux son similaire dans leur entièreté, cela signifie que tous les bateaux sont coulés et donc qu'il y a un gagnant. Dans ce cas, la partie s'arrête et le gagnant est annoncé.

Figure 3 – Exemple de nos structures

# IV Difficultés rencontrées

La principale difficulté rencontrée fut lors de la compilation du jeu complet, quand nous avons mis en commun le travail de chacun pour tester notre jeu. À ce moment-là, de nombreuse erreurs de compilation et de variable.
L'un des problèmes majeurs rencontrés fut la modification des données à l'intérieur de nos structures créé, en effet à certains moments le changement d'une valeur d'une structure, changeait la totalité des valeurs de cette structure en un autre nombre sans raison apparente...
Par exemple dans notre structure joueur, on stocke une donnée (1 ou 2) qui nous permet de savoir si le joueur est un humain ou un ordinateur. Et lors de l'appel de cette variable dans notre menu pour faire jouer chacun son tour les joueurs ne marchait pas, pour une raison qui nous échappe la valeur changeait de 2(ordinateur) à autre nombre aléatoire... Ce qu'évidemment ne pouvait plus être reconnu dans notre menu.

Un autre point qui n'a pas été facile à gérer, fut le debug de tous nos fichier, en effet nous avons un nombre conséquent de fichier et certaines erreurs à une étape venait d'un conflit avec une variable d'un autre fichier ou encore d'une fonction elle-même remplis de bug. Une grande partie de notre temps a été prise dans la compréhension de ces bugs et leur résolution, qui n'était pas de la même ampleur que ceux rencontrés durant les différents tp de ce module.

C'est différents problèmes nous on prit énormément de temps de réflexion pour les trouvers et les résoudre, temps que nous n'avions pas en seulement 2 semaines bien remplies (Khôlles de maths et 2 DST de math et l'école fermé un des deux week-end, et pour nous achever les grêves SNCF).

# V Avantages et limites de la solution

En ce qui concerne, les avantages et les limites de notre programme, nous pouvons dire que l'avantage de notre programme est la possibilité de personnaliser si on le souhaite la taille et le nombre des différents bateaux.
Et du point de vue, de la limite de notre solution, il s'agit du fait que nous n'avons pas pu coder d'IA avec différents niveaux de difficulté car nous avons réussi à la faire mais impossible de la rajouter dans nôtre partie, des bugs de segmentation apparaisse certaine fois... Ainsi que d'améliorer notre interface graphique pour rendre notre jeu un peu plus esthétique, notamment dû au manque de temps. Nous avons donc décidé de ne pas entreprendre l'amélioration de l'interface graphique, mais malheureusement afin de ne pas rendre un projet qui ne fonctionne pas.

Mais maintenant c'est à vous de tester et de jouer !!!

# VI Excuter notre programme

Il est à noter qu'il faut compiler et éxécuter afin de pouvoir jouer.
Pour cela, il suffit de lancer la console dans le dossier où nos fichiers sont situés, écrire *make* dans la console pour compiler notre programme, en effet le fichier makefile s'occupe de compiler tous les fichiers nécessaire à l'excution de notre programme.

Ensuite il ne reste plus qu'à écrire *./jeu* dans la console afin de lancer notre programme.
Du point de vue de la jouabilité, notre programme s'excute et se joue dans la console en utilisant seulement le clavier.

# VII Conclusion/Bilan personnel

Pour conclure ce projet d'algorithmique, nous nous sommes organisés correctement afin d'arriver à un résultat jouable et remplir le plus possible le cahier des charges. Qui étaient : la création d'une version "humaine" d'une partie classique de bataille navale et la création d'une version "ordinateur" pour implémenter des IA (intelligence artificielle) dans le jeu.
Malheureusement le temps pour la réalisation de ce projet était trop court à notre goût(nous n'avions que seulement 2 semaines bien remplies (Khôlles de maths et 2 DST de math et l'école fermé un des deux week-end, et pour nous achever les grêves SNCF pour réaliser ce projet...), ne nous laissant pas la possibilité d'aller plus loin, dans l'esthétique ou l'interface de notre jeu. De même pour ce rapport, nous aurions aimé mettre des images d'illustration de notre jeu, mais nous n'avons pas eu le temps de prendre les captures d'écrans. En effet, 20 min avant la date de rendu, nous essayons toujours d'incrémenter notre intelligence artificielle dans notre programme, étant donné que nous ne pouvions pas mettre des captures d'écrans différentes de la réalité nous n'en avons pas mis.

Nous pouvons préciser que nous nous sommes très bien entendus au sein de notre groupe avec un bon esprit d'équipe et beaucoup d'échanges et d'entre-aide. Ce projet nous a malgré tout apporté beaucoup en effet, en premier lieu nous avons pu davantage nous découvrir, nous perfectionner en langage C#. Mais aussi nous avons pu approfondir l'expérience du travail en équipe, ce qui pourra être attendu de nous en entreprise, à travers l'élaboration, les recherches et la conception d'un projet.
Cette expérience nous a donc été très bénéfique à tous et nous sommes prêt pour les prochains projets le semestre prochain en espérant pouvoir faire mieux.