# Naval Fight

## ***Objectives:***

1/Commenter la version original

2/Extendre le jeu à trois joueurs

3/Gènese d'obstacle où un navire n'a pas le droit d'y être posé

4/Extension de la grille

**0/Preparation :**

Avant de commencer tout travail, nous avons d'abord préparer le terrain en créant un répertoire de travail dédié dans notre entrepôt GitHub ( https://github.com/bk211/realisationProgramme ).

Ayant un peu acquis plus d'expérience grâce à notre projet Yahtzee, une des premiers fichiers que nous avons crée et configurer était le Doxygen ainsi que le .gitignore.

Après cela, nous nous somme donnés un moment pour lire en intégralité le code du programme reçu, ce fut très pénible car il n'y avait qu'un seul .h (autrement dit, aucun). Cela fut très difficile de garder une lecture continue du code car il faut passer énormément de temps à trouver le .c qui contient la fonction appelée.

Par la suite, nous nous somme réunis ensemble pour faire un petit débrief du code, de ce qu'on en pense en général, et d'apporter une clarification sur quelque fonction complexe. Notamment usage d'une fonction récursif parcourant le tableau de jeu pour la genèse des bateaux.

De manière général, l'ensemble est confus et mal organisé, il y a des .c qui n'avait qu'un ou deux fonctions, d'autre qui en contenait 7-8 mais extrêmement répétitif.

Enfin nous nous somme partager le travail en plusieurs partie distinct, les parties étant suffisamment indépendant pour être traité individuellement.

J'ai d'abord commencer à créer des .h associé à chaque fichier .c , et à m'occuper des dépendance inter fichier, cela fut très pénible car il fallait partir d'un seul .h et refaite toute la dépendance, voici d'ailleurs la dépendance généré par le Doxygen avant et après

2/Extendre le jeu à trois joueurs :

Pour cette partie, il n'y avait pas énormément de chose à modifier, il suffisait d'initialiser un troisième joueur et de modifier le prototype de la fonction boucle() du fichier boucle.c de manière à prendre en paramètre un troisième joueur. Enfin il suffisait de modifier la cible d'attaque du joueur 2 afin de crée une boucle d'attaque de type A>>B >>C >>A

Après cela, j'ai confié à Ramu la tâche de mettre du couleur dans la grille via une bibliothèque que j'avais écrite pour Yahtzee, il permettait de faire abstraction du code couleur de la printf via des fonction type red(), green() etc.  Enfin j'ai débugué l'affichage car il y avait un problème avec la couleur. Cela m'a causé pas mal de temps car l'erreur n'avait pas vraiment de sens logique à mes

yeux, il s'agit de la ligne 70 où le fait d'enlever cette ligne causait un segFault mystérieux…
Cette ligne permettait de réinitialiser la couleur du printf par défaut (l'équivalence dans ma bibliothèque est la fonction resetColor().

```
15  void afficher(char** grille){
16    int ligne, colonne;
17    char lettre = 'A';
18
19    for(colonne = 0 ; colonne <= 19 ; ++colonne){
20
21      if(colonne < 10)
22        printf("  %d  ",colonne);
23      else
24        printf(" %d  ",colonne);
25    }
26
27    printf("\n");
28
29    for(ligne = 0 ; ligne < Dim ; ++ligne){
30      printf("-----------------------------------------------------------------------------------------------------\n");
31
32      for(colonne = 0 ; colonne < Dim; ++colonne){ ...
57      printf("|  ");
58      magenta() ;
59      printf("%c ", grille[ligne][colonne]) ;
60      resetColor() ;
61        }
62        else{
63          printf("| ");
64          resetColor();
65          printf("%c  ",grille [ligne][colonne]);
66
67        }
68      }
69
70      printf("\033[0m");
```