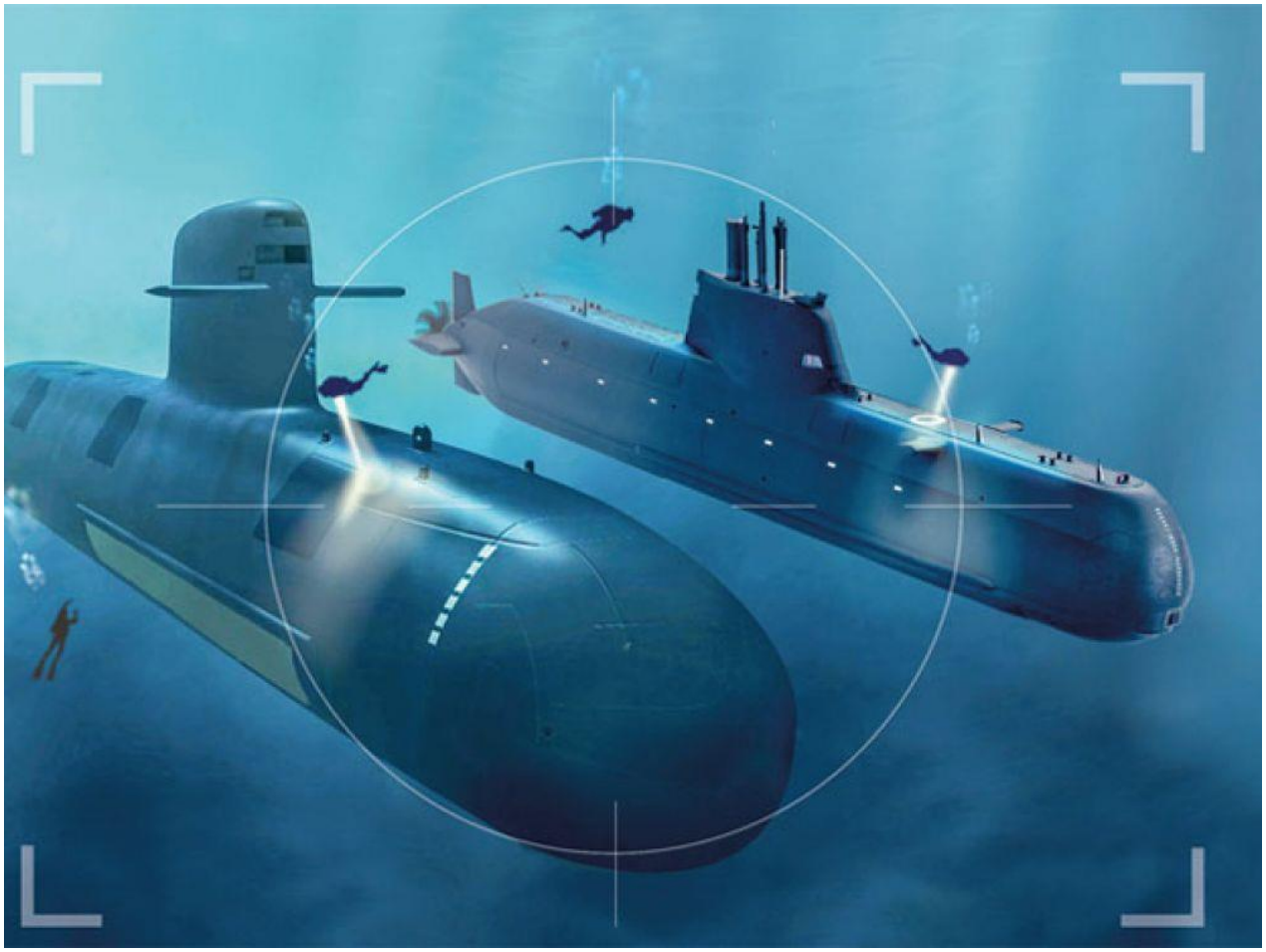
Bataille navale de sous-marin



Alexis FREDOUILLARD

Anthony CAROFF

SOMMAIRE :

- # Présentation du projet :

Présentation globale :

Le projet consiste à faire une bataille navale de sous-marin en langage c

Il s'agit d'une bataille navale classique où deux joueurs doivent placer des bateaux sur une grille x et y le joueur adverse doit trouver les diffèrent bateau et les couler. Ici, dans cette bataille navale de sous-marin il existe une troisième dimension, z, celle des différents niveaux de profondeur où les joueurs pourrons placer ses sous-marins sur trois profondeurs représentées par trois tableaux différents,

Définition des règles

Pour commencer à jouer :

- Les deux joueurs ont trois sous-marins chacun de différentes tailles à placer sur la grille sur les trois dimensions en ne pouvant pas les superposer et également ne pouvant pas les mètres sur deux profondeurs en même temps,
    - Le premier fait une taille de 1*1
    - Le deuxième fait une taille de 1*2
    - Le troisième fait une taille de 1*3

- Une fois que les deux joueurs ont placé leurs bateaux, ils doivent trouver les sous-marins du joueur adverse en effectuent de "tirs" sur une case de la grille du joueur adverse, ensuite suivant ce que le tir à toucher le jeu affichera une lettre sur la grille.
    - Si dans une zone en croix au tour du tir il y a un sous-marin le jeu affichera des "v" en croix qui indiquera qu'un sous-marin est vu sur ces cases.
    - Si le tir ne touche rien ou rien n'est en vue le jeu n'affichera "R" en croix au tour du tir.
    - Si un tir touche un sous-marin, le jeu affichera "t" sur la case du tir.
    - Si le sous-marin est coulé, donc que tous ses cases sont toucher le jeu marquera "C" soit coulé, sur toutes les cases du sous-marin.
- Enfin, pour finir le jeu, il faut que l'un des deux joueurs ai coulé tous les sous-marins du joueur adverse.

- # Les tâches faire

Les fonctions et structures utilisées

Structures

- Dans un premier temps, nous avons créé plusieurs structures :
  - "Joueur" qui permet de regrouper l'id du joueur, le nom, les sous-marins que le joueur va placer et le tableau "mer".
  - "Position" qui permet de regrouper les variables x, y, z qui sont les variables de position du sous-marin.
  - "bool" qui permet de créer un type de variable qui retournera vrai ou faux.
  - "sousMarin" qui regroupe toutes les données qui concerne les sous-marins comme sa taille, son initialisation de position, les différents états des sous-marins s'ils sont touchés ou coulés.

- Il y a la gestion des joueurs que nous avons réalisée à l'aide d'une fonction, qui permet d'assigner un id et un nom au joueur pour pouvoir différencier les deux joueurs, leur assigner les sous-marins qu'ils doivent placer et une grille chacun, en utilisant la structure "Joueur" pour définir les joueurs. Cette fonction est **:**
  - **void InitJoueur(Joueur * joueur, int numeroJoueur);**

- Ensuite pour le placement des diffèrent sous-marin nous avons réalisés une fonction qui demande au joueur les coordonnées dans la grille pour placer un sous-marin case par case qui utilise les structure "sousMarin" et "Position", avec une vérification du bon placement des différents sous-marin sur la grille. Cette fonction est :
  - **void placementSousmarin(SousMarin *sousmarin, int taille);**

- Pour pouvoir placer les trois sous-marins nous avons fait une fonction qui appelle 3 fois la fonction précédente qui est :
  - **void PlacementParJoueur (Joueur *joueur);**

- Pour la gestion du tir sur les sous-marins la fonction réalisée permet de sélectionner des coordonnées pour cibler une case de la grille. Cette fonction est :
  - **bool SousMarinPositionCible(Joueur *joueur, Position posTire);**

- Ensuite, nous avons réalisé une fonction pour l'impact du tir qui permet d'afficher si un sous-marin a été vue en affichent "V" ou s'il n'y a rien avec "R" mais aussi si le tir a touché avec un "T". Cette fonction est :
    - **void impactTire(Position *posTire, Joueur *joueur, char mer);**

- Pour savoir si le sous-marin est coulé nous avons fait une fonction, qui suivant la taille du sous-marin et suivant le nombre de fois qu'il a été touché, on remplace par "c" les cases du sous-marin. Cette fonction est :
    - **bool SousMarinCoule(SousMarin *sousmarin);**

- Enfin pour réalisation du jeu nous avons fait deux fonctions, une première pour initialiser le jeu en définissant les deux joueurs le placement des sous-marins pour les deux joueurs et une assignation de leurs grilles de jeu chacun et pour la deuxième elle permet de réaliser les tours de jeu et la fin du jeu. Ces fonctions sont :
    - **void InitGestionJeu()**
    - **void JeuEnCours()**
- Pour finir nous avons fait une fonction pour l'affichage de la grille du tableau de la mer. Cette fonction est :
    - **void MerInitialisation(char mer[PROFONDEUR_MAX_Z][MER_MAX_X][MER_MAX_Y])**

Organisation des fichiers

Pour l'organisation des fichiers nous avons reparti et regroupé les différentes fonctions dans diffèrent fichiers de façon cohérente :

- **Joueur.c / Joueur.h** -> Qui regroupe les fonctions et les structures qui touchent aux joueurs.
    - void InitJoueur(Joueur * joueur, int numeroJoueur)

- **Sousmarin.c / sousmarin.h** -> Qui regroupe les fonctions et les structures qui touche au placement du sous-marin et à son état.
    - void placementSousmarin(SousMarin *sousmarin, int taille);
    - void PlacementParJoueur (Joueur *joueur);
    - bool SousMarinCoule(SousMarin *sousmarin);

- **Tire.c / Tire.h** qui regroupe les fonctions et les structures permettent d'effectuer les tirs sur le sous-marin et de savoir s'ils sont vus touché ou s'il n'y a rien.
  - bool SousMarinPositionCible(Joueur *joueur, Position posTire);
  - void impactTire(Position *posTire, Joueur *joueur, char mer);

- **Gestion_du_jeu.c / gestion_du_jeu.h** qui sont les ficher qui regroupe les fonctions pour la boucle du jeu permettent de joueur.
  - void InitGestionJeu()
  - void JeuEnCours()

- **Mer.c / mer.h** sont pour l'affichage de la grille de la mer
  - void MerInitialisation(char mer[PROFONDEUR_MAX_Z][MER_MAX_X][MER_MAX_Y])

Répartition des tâches

Alexis :

Devait réaliser la partie de la gestion des joueurs ainsi que l'initialisation de la mer et de son affichage.

Les fonctions réalisées :

- **void InitJoueur(Joueur * joueur, int numeroJoueur);**
- **void MerInitialisation(char mer[PROFONDEUR_MAX_Z][MER_MAX_X][MER_MAX_Y])**
- **void placementSousmarin(SousMarin *sousmarin, int taille);**
- **Réalisation du compte rendu**

Anthony :

Devait réaliser la partie qui traite les sous-marins et la partie sur les différents tirs.

Les fonctions réalisées :

- **void PlacementParJoueur (Joueur *joueur);**
- **bool SousMarinPositionCible(Joueur *joueur, Position posTire);**
- **void impactTire(Position *posTire, Joueur *joueur, char mer);**
- **bool SousMarinCoule(SousMarin *sousmarin);**

Réalisation a deux :

Une fois les deux parties faites nous avons mis en commun notre travail pour réaliser le programme principal qui fera appel aux fonctions présente au-dessus.

Les fonctions réaliser :

- **void InitGestionJeu()**
- **void JeuEnCours()**

Conclusion

Problèmes rencontrés

Lors de ce projet nous avons rencontré plusieurs problèmes, tout d'abord nous avions divisé le travail à faire un deux parties sauf que lors de la création des fonctions nous ne pouvions pas les tester car certaines fonctions utilisaient d'autres fonctions de la partie de l'autre partie, ce qui a posé énormément de problèmes lors de la mise en commun. La prise en main d'Eclipse nous a aussi posé problème, ce qui nous a fait perdre du temps

Ce qui a été fait

- Toutes les fonctions individuelles qui permettent de faire fonction le jeu
- L'affichage

Ce qui reste à faire

- La boucle principale qui régit le jeu
- La sauvegarde du jeu la sauvegarde du jeu