Projet du cinquième semestre dans le cadre du cours de C Avance
2020 - 2021

PROJET C AVANCE : BATAILLE NAVALE

Réalisé par : Quentin Carpentier & Paul-Joseph Krogulec

❖ Le projet

Le but de ce projet est de proposer un jeu de type bataille navale où un joueur peut affronter l'ordinateur dans une partie qui se jouera en console. Nous avons donc défini les structures ainsi que le code qui permettra à l'utilisateur d'initialiser et de jouer une partie, et à l'ordinateur de développer une stratégie de jeu efficace contre l'utilisateur.

Pour réaliser ce projet, nous avons tout d'abord établie 3 problématiques principales :

1. Comment gérer la mise en place des bateaux ?

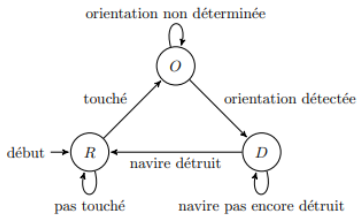
Durant les premiers jours, nous avons donc démarrer le projet en travaillant sur la gestion du positionnement des bateaux. Nous avons créé des structures essentielles Ship et Case pour nous permettre de garder en mémoire le positionnement des bateaux.
Puis nous avons créé des fonctions qui permette de mettre en place les bateaux et de vérifier s'il ne dépasse pas la grille de jeu ainsi que l'affiche d'une grille standard de taille 10x10.

2. Comment gérer le tir simple et les tires spéciaux ?

Une fois que les bateaux ont été mise en place, nous avons crée une fonction qui permet de tirer sur une case et de le mettre à jour sur le plateau de jeu en prenant en paramètre un pointeur de fonction de tirs. Puis nous avons réalisés les tirs spéciaux et avons dû modifier la structure Player pour que nous pouvons fixer un tir de chaque tir spécial par Joueur.

3. Comment créer l'IA, quelle stratégie ?

Pour pouvoir crée l'IA nous avons dû crée une nouvelles structures Ordi qui se trouve être une extension d'un joueur comportant en plus une matrice d'entiers contenant l'historique de ses tirs, l'état dans lequel elle se trouve ainsi que l'orientation de tir.



Lors de son premier tir, on récupère une case aléatoirement au sein de la grille. Par la suite, si elle se trouve dans un état R de recherche de navire, elle va effectue un tir toutes les deux cases au sein de la grille adverse. Une fois un navire détecté, elle passe en état de recherche d'orientation en effectuant un tir aux positions adjacentes. Enfin, lorsque l'orientation du navire est trouvée, on passe en état de destruction du navire qui va tirer sur chaque case jusqu'à la destruction de celui-ci. Une fois le navire coulé, on retourne en état de recherche d'un nouveau navire en démarrant d'une case aléatoire à nouveau.

❖ Les fonctionnalités réalisées & la répartition des tâches.

Ci-dessous, vous trouverez l'ensemble des fonctionnalités détaillés du projet :

- ☑ **Construction des structures globales Case, Ship et Player** modélisant respectivement : une case d'une grille, la grille d'un joueur, et le joueur en question. Pour cette fonctionnalité, nous avons travaillé ensemble pour réfléchir au meilleur moyen de structurer les bases du projet.
- ☑ **Ajout d'un navire à une grille** en vérifiant si le bateau ne touchera pas d'autre bateau *(réalisé par Quentin)*.
  Ajout d'un tableau de tableau de navires répartis aléatoirement dans une grille *(réalise par Paul-Joseph)*.
- ☑ **Ajout des tirs spéciaux et d'une fonction de tir**, qui permettent de retourner un tableau de case cibles pour tirer sur toutes ces cases dans une fonction prenant en paramètre les mêmes paramètres ainsi qu'un pointeur de fonction d'une procédure de tirs *(réaliser par Quentin)*.
- ☑ **Ajout d'une fonction réalisant une partie**. **Demande des coordonnées** de tirs et **un type de tirs** + **vérification des tirs possibles** *(réalisé par Paul-Joseph avec l'aide de Quentin)*.
- ☑ **Ajout d'une IA**, structure étendue de celle d'un joueur + fonction qui effectue elle-même un tour selon son dernier tir *(réalisé Quentin et Paul-Joseph)*.
- ☑ Ajout des bonus : **modification de la taille + choix du placement des bateaux**, deux fonctionnalités bonus *(réalisé par Quentin)*.
- ☑ **Gestion des fuites mémoires**, qui permettent de libérer l'espace mémoire utilisé par le programme *(réalisé par Paul-Joseph)*.
- ☑ **Ajout d'un fichier Makefile**, qui effectue la compilation du programme *(ajouté par Quentin)*.

❖ Infos Pratiques

Pour compiler en supprimant les « .o » : **make clean;make;./battleShip**

Pour compiler et lancer le programme : **make;./battleShip**

Pour un complément d'informations, voici le lien GitHub que nous avons utilisés pour se travail de groupe : https://github.com/pauljosephkrogulec/L3-CAV-PROJECT/