Rapport de projet sur la Bataille Navale ++

Par Bouvet Rémi, Zerbane Mehdi et Pradere-Niquet Alexandre

Université du Maine
L2 SPI Informatique

Table des matières

Introduction :

Ce document a pour but de décrire le déroulement de notre projet informatique sur le jeu de la bataille navale ++.

Ce projet a été réalisé durant les heures de travaux pratiques à l'Université du Maine, mais aussi durant notre temps personnel, il a pour but de nous apprendre à mieux maîtriser le langage C et travailler sur un projet en condition réel (temps imposé

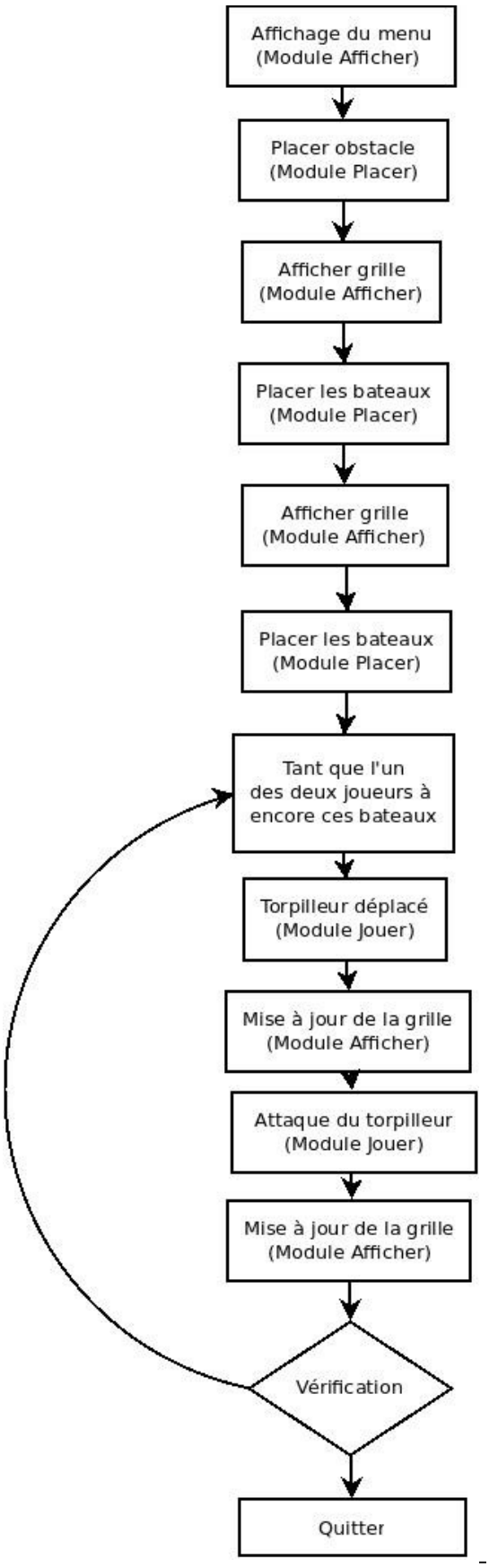
La bataille navale ++ est un jeu de société qui oppose deux joueurs. Le but est de couler tous les bateaux de son adversaire. Les deux joueurs doivent placer des navires de façon stratégique, bien entendu l'adversaire ne voit pas la grille du joueur, chaque joueur dispose d'une flotte composée de maximum quatre sous-marins (trois cases), de quatre destroyers (4 cases) ainsi que de quatre porte-avions (5 cases). Les joueurs ont le choix de décider du nombre de bateaux qu'ils possèdent au début du jeu. Ce jeu est différent de la bataille navale classique, en effet dans le jeu de la bataille navale classique on dispose de plusieurs bateaux mais ils sont immobiles, dans la bataille navale ++ nous avons décidé d'ajouter un élément qui pourra se déplacer dans la grille de l'adversaire, on l'appelle "le torpilleur", c'est uniquement le torpilleur qui tire sur les bateaux ennemi. Cet élément se trouve sur la grille de l'adversaire et il est considéré comme intouchable, il ne peut être détruit par le torpilleur adverse, il dispose d'une portée choisit par les joueurs au début du jeu, la portée est soit d'une case ou de deux cases selon la volonté du joueur, il peut tirer verticalement horizontalement ou en diagonale et il peut avoir un déplacement de huit cases maximum, le nombre de déplacement pour chaque joueurs sera demandé au début de chaque tour. La grille est numérotée de A à J verticalement et de 1 à 10 horizontalement.

Pour réaliser ce projet on a séparé les tâches en différents modules, le premier module est le module grille qui contient plusieurs fonctions principales comme lire sur une case s'il y a un bateau ou un obstacle ou bien écrire à un emplacement qu'un bateau est positionné, ce module a été fait en groupe et est utilisé par tous les membres de ce projet, le deuxième module est le module placer qui permet de placer les obstacles et les bateaux sur les grilles, ce module a été réalisé par Zerbane Mehdi, le module jouer qui permet aux joueurs de déplacer leurs torpilleurs et d'attaquer les différents bateaux présent sur la grille de l'adversaire, ce module a été réalisé par Bouvet Rémi, ainsi que le module afficher qui permet d'afficher les différentes grilles ainsi que les légendes leurs appartenant, ce module a été réalisé par Pradere-Niquet Alexandre.

Récapitulatif de la séparation du travail :

	Grille	Jouer	Placer	Afficher
Zerbane Mehdi	X		X	
Bouvet Rémi	X	X		
Pradere-Niquet Alexandre	X			X

Organisation des modules :



Analyse du problème :

Structures de données nécessaires :

Nous avons besoin d'une structure t_case qui contient l'ensemble des informations de ce qui peut y avoir dans une case d'une grille. Dans celle-ci nous y avons inséré quatre énumérations : une qui nous permet de vérifier la présence d'un bateau, une pour déterminé si un tir a eu lieu ou non sur la case, une qui détermine s'il y a un obstacle et enfin une qui détermine la présence ou non d'un torpilleur.

Une structure t_plateau regroupe deux matrices de type t_case, elle rassemble toute les données relatives aux grilles.

Une structure t_coordonnee qui permet de rassembler deux entier x et y pour y stocker comme son nom l'indique des coordonées, elle est principalement utilisée pour stocker les coordonnées des curseurs.

Nous utilisons également d'autres énumérations en dehors de nos structures: une énumération t_portee qui permet de gérer et vérifier si la portée du torpilleur et une énumération t_direction qui permet d'indiquer la direction dans laquelle on va déplacer le torpilleur.

Fonctionnalités du programme :

Au début du jeu on propose aux deux joueurs s'ils souhaitent jouer ou quitter la partie, si les deux joueurs choisissent de jouer on leurs demandera combien de torpilleurs ils souhaitent avoir, cependant on leurs donne un nombre maximum de cinq torpilleurs et un minimum d'un torpilleur. Si les deux joueurs ne disposent d'aucun torpilleurs ils ne pourront pas jouer car c'est uniquement les torpilleurs qui tirent sur les bateaux. On leurs impose cinq torpilleurs maximum pour éviter que la partie se termine beaucoup trop rapidement car il y aura au moins un torpilleur qui sera positionner à côté d'un bateau de l'adversaire, on leur demande également de choisir le nombre de déplacement maximum pour le(s) torpilleur(s) et la taille de la portée du torpilleur.

Ensuite, on leurs demandent s'ils souhaitent choisir une configuration personnalisée de leurs nombres de bateaux (exemple : trois sous-marins, deux destroyer, deux porte-avions) ou une configuration déjà crée où il n'y qu'un bateau de chaque type (un sous-marins, un destroyer et un porte-avions). Ensuite, les deux joueurs pourront choisir s'ils placent leurs bateaux manuellement ou aléatoirement sur la grille. Durant le placement de chaque bateau on demande au joueur s'il est satisfait de son placement, s''il ne l'ai pas, on le laisse replacer son bateau. Pour mieux visualiser où l'on à placé notre bateau on affichera la grille avec l'emplacement du bateau.

A la fin de chaque tour le joueur aura le choix de sauvegarder, de continuer à jouer ou de quitter la partie. Lors de chaque tour les deux joueurs pourront déplacer un torpilleur du nombre de case choisis et tirer une seule fois selon une portée choisie.

Codage, méthode de programmation :

Partie méthode :

Notation :

Lors de notre projet nous avons décidé d'utiliser une convention de nommage des variables : la notation hongroise.
Nous avons donc préfixé le nom de nos variables en fonction de leurs types ou de leurs utilisations, le préfixe est écrit en minuscule et la première lettre de la variable en majuscule ce qui permet lorsque que l'on lit notre code de tout de suite connaître le type de notre variable sans revenir à sa déclaration.
Un entier dans notre code est précédé d'un "e", une variable booléenne est précédée d'un "b" et un pointeur sur un entier est précédé de "pe".
Voici un exemple à la déclaration :

```
int eBateau;
int bGagnant;
```

Nous avons également nommé nos fonctions en écrivant l'action qu'elles traitent et l'objet sur lequel le traitement est effectué.

Outils de gestion de version :

Pour mutualiser le développement de notre projet, nous avons utilisé un logiciel de gestion de version : git. Associé à Github un service d'hébergement et de gestion de développement de logiciels, l'utilisation de git nous permet de mettre à jour rapidement notre code et de le fusionner rapidement.

Utilisation de "boites à outils" :

Pour faciliter notre travail et nous faire gagner du temps, nous avons utilisé ce que l'on appelle une boite à outils, elle est utilisée notamment dans le module grille pour vérifier que par exemple une variable i est comprise entre 1 et 2 inclus ainsi que dans le module placer pour prendre un nombre au hasard et l'affecter à une variable.

Différents modules :

Module Grille

void Grille_Sauvegarder(...) :

Cette fonction permet de sauvegarder la structure t_plateau dans un fichier appelé fic1.

void Grille_Charger(...) :

Cette fonction permet de charger le fichier fic1 et de transférer les données présentes dans ce fichier dans la structure t_plateau.

void Grille_lire_bateau(...) :

Cette fonction lit la case indiquée pour déterminer quel type de bateau s'y trouve ou s'il n'y en a pas.

void Grille_ecrire_bateau(...) :

Cette fonction permet d'écrire aux coordonnées indiquées le type de bateau présent s'il y en a un.

void Grille_lire_toucher(...) :

Cette fonction permet de connaître les informations permettant de déterminer s'il y a eu un tir effectué dans la case.

void Grille_ecrire_toucher(...) :

Cette fonction fournit les informations pour déterminer s'il y a eu un tir effectué dans la case aux coordonnées indiquées.

void Grille_lire_obstacle(...) :

Cette fonction permet de vérifier la présence d'un obstacle dans la case indiquée.

void Grille_ecrire_obstacle(...) :

Cette fonction permet de placer un obstacle ou non dans selon les coordonnées indiquées en paramètres dans la grille donnée.

void Grille_lire_torpilleur(...) :

Cette fonction permet de savoir si un torpilleur est placé dans la case indiquée.

void Grille_ecrire_torpilleur(...) :

Cette fonction place un torpilleur ou non selon le choix donné dans la grille indiquée à l'emplacement fourni.

void Grille_lire_case(...) :

Cette fonction reprend toute les fonctions "lire" pour vérifier d'un coup le contenu de la case.

void Grille_ecrire_case(...) :

Cette fonction reprend toute les fonctions "lire" pour écrire d'un coup les information aux coordonnées indiquées en paramètres.

Module Placer

Pour placer les obstacles :

Tout d'abord pour placer les obstacles nous avons besoin de connaître le nombre d'obstacles à placer, mais nous avons définit un nombre par convention qui est de 5 obstacles.

void placer_obstacle():

Pour placer les obstacles aléatoirement sur la grille on utilise une fonction qui se nomme "uHasard(N)", elle permet d'affecter à une variable la valeur que va retourner la fonction "uHasard(N)" (N étant la borne maximal).
Nous affectons donc à une variable i représentant les lignes de la matrice un nombre au hasard et à une variable j représentant les colonnes de la matrice un nombre au hasard aussi. Pour éviter de retrouver les même indices i et j, avant d'inscrire l'obstacle dans la grille on va vérifier aux coordonnées i et j à l'aide d'une fonction "Grille_lire_obstacle(...)" si un obstacle est déjà présent. Cette fonction retourne 1 si un obstacle est présent aux coordonnées rentrées et 0 si il n'y a pas d'obstacle.
On incrémente le compteur d'obstacle lorsque l'on rentre un obstacle dans la grille.

Pour placer les bateaux :

Le placement des bateaux est l'une des parties importantes pour le bon déroulement du jeu, en effet si un bateau est positionné par exemple en partie en dehors de la grille de jeu et l'autre partie dans la grille de jeu, lorsque l'un des joueurs aura touché toutes les parties des bateaux sauf celles qui sont en dehors de la grille, le joueur ne gagnera jamais car dans le module jouer on oblige le joueur à tirer dans une zone présente dans la grille de jeu. Il est donc nécessaire de faire plusieurs vérifications dont celle ou le bateau ne sort pas de la grille, et celle ou le bateau ne se positionne pas sur un autre bateau.

int bStringtonum(…) :

Cette fonction convertie une chaîne de caractères en entier si dans celle-ci il n'y a que des entiers. Elle retourne 1 si il n'y a que des entiers, et 0 dans le cas contraire.
Exemple : Chaine de caractère : 123a Retourne 0 car présence du caractère "a"
Chaine de caractère: 1234 Retourne 1 car il n'y a que des entiers

int verif_presence(…) :

Cette fonction permet de vérifier avant de placer un bateau si aux emplacements données il y a déjà un bateau ou un obstacle de présent. Cette fonction est importante car c'est une vérification avant de positionner un bateau. Elle retourne 1 si il n'y a pas de bateau ou d'obstacle présent et 0 si il y a déjà un bateau ou un obstacle de positionné.
Cette fonction s'adapte en fonction de la taille du bateau que l'on souhaite placer.

int Assez_de_place(…) :

Cette fonction vérifie selon la taille du bateau s'il y a la place pour le positionner, pour évité de placer une partie du bateau en dehors de la grille de jeu. Elle est aussi importante que la fonction précédente pour positionner un bateau sur la grille

void Placer_grillebateau(...) :

La fonction positionne dans la grille du joueur le bateau rentré en paramètre et dans le sens souhaité.

void Enlever_grillebateau(…) :

Cette fonction permet de supprimer de la grille un bateau, le nom du bateau sera rentré en paramètre. Elle est utilisé lorsque l'utilisateur n'est pas satisfait de son placement et souhaite placer à un endroit différent son bateau.

void Enlever_grilletousbateau(…) :

La fonction permet de **supprimer tous les bateaux** de la grille, elle est utilisée notamment lorsque l'on choisit de placer les bateaux aléatoirement et que l'on est pas satisfait du résultat.

void Placer_bateau_auto(...):

Cette fonction place **automatiquement** tous les bateaux et dans un sens choisit aléatoirement, cette fonction s'adapte en fonction du nombre de bateau souhaité (exemple: deux sous-marins, deux destroyers, trois Porte-Avions).

int Changement_colonne(...):

Cette fonction est utilisée lorsque le joueur choisit de placer **manuellement** les bateaux, lorsque le joueur entre une lettre on affecte son équivalent en entier selon son positionnement en colonne sur la grille de jeu dans une variable en pointeur passé en paramètre.

void Placer_bateau_manuelle(...):

La fonction permet de placer manuellement tous les bateaux, cette fonction prend en paramètre les différents nombre de bateaux (exemple : 2 sous-marins, 2 destroyer, 3 porte-avions, 4 torpilleurs) ainsi que le numéro de grille ou seront placés les bateaux.

void Commencer_jeu_placement_bateau(...);

Cette fonction initialise la grille avec aucun bateau et aucun obstacle à chaque case, après l'initialisation on place aléatoirement les obstacles sur la grille à l'aide de la fonction placement obstacle.

On demande ensuite si l'utilisateur souhaite avoir un bateau de chaque type (Sous-Marins,Destroyer,Porte-Avions) ou de choisir son nombre de bateaux. Après sa saisie, on lui demande s'il souhaite faire un placé automatique ou bien un placement manuelle, et selon son choix on appellera la fonction placement bateau automatique ou placement bateau manuelle.

Module Jouer

Le module jouer (jouer.c) est le module qui gère le déroulement du jeu c'est à dire les tours des joueurs, le choix, le déplacement et l'attaque des torpilleurs.Il est divisé en plusieurs sous module :

- Choisir (TorpilleurChoisir.c)
    - Contient les fonctions relative à la selection du torpilleur
- Deplacer (TorpilleurDeplacer.c)
    - Contient les fonctions relative au déplacement du torpilleur
- Attaquer (TorpilleurAttaquer.c)
    - Contient les fonctions relative à l'attaque du torpilleur

void Jouer_Init_Torpilleur(…) :

Cette fonction permet à l'utilisateur de définir le nombre de torpilleur, le nombre de point de déplacement et la portée du torpilleur durant le reste de la partie.

void Jouer_Changer_Joueur(…) :

Cette fonction permet de changer le numéro de joueur actuel en numéro de joueur adverse.

int Jouer_Gagnant(…) :

Cette fonction permet de tester si le joueur rentré en paramètre a gagné la partie c'est a dire que le joueur adverse n'a plus de case dans la grille où il y a un bateau qui n'est pas touché.

void Jouer_Quitter_Continuer_Sauvegarder(…) :

Cette fonction est appelée en fin de tour d'un joueur, elle permet au joueur sauvegarder, de quitter ou de continuer la partie en cours.

void Jouer_Choisir(…) :

Cette fonction appelle les fonctions du sous module **Choisir**, elle permet donc de trouver les torpilleurs sur la grille et permet à l'utilisateur de pouvoir en sélectionner un pour la suite de son tour.

void Jouer_Deplacer(…) :

Cette fonction appelle les fonctions du sous module **Deplacer**, elle permet de déplacer le torpilleur dans la grille. Pour ce faire on demande à l'utilisateur de choisir une direction, si celle-ci est valide on déplace le torpilleur. On répète cette routine autant de fois que le nombre de point de déplacement configuré en début de partie.

void Jouer_Attaquer(…) :

Cette fonction appelle les fonctions du sous module **Attaquer**, elle permet d'attaquer à partir du torpilleur. La fonction commence par calculer la portée du torpilleur, si celui-ci peut attaquer alors l'utilisateur va choisir l'endroit où il souhaite tirer en déplacent le curseur dans les cases correspondantes puis valider avec entrée. Si le torpilleur ne peut pas attaquer, un message sera adressé à l'utilisateur.

Sous module Choisir :

void Jouer_Trouver_Torpilleur(...) :

Cette fonction permet de calculer les positions des torpilleurs du joueur en train de jouer et de les stocker dans un tableau de type t_coordonnee.

void Jouer_Selectionner_Torpilleur(...) :

Cette fonction permet à l'utilisateur de sélectionner un torpilleur à l'aide des flèches du clavier pour la suite de son tour.

Sous module Deplacer :

void Jouer_Choisir_Direction(...) :

Cette fonction permet à l'utilisateur de choisir une direction dans laquelle déplacer le torpilleur à l'aide des flèches du clavier.

int Jouer_Deplacement_Valide(...) :

Cette fonction permet de tester si la direction choisie est bien valide, c'est à dire qu'il n'y a pas d'obstacle, de torpilleur et que les coordonnées sont bien dans les limites de la grille.

Jouer_Deplacer_Torpilleur(...) :

Cette fonction permet de mettre a jour les coordonnées du torpilleur en fonction d'une direction demandée.

Sous module Attaquer :

void Jouer_Calculer_Portee(...) :

Fonction qui permet de calculer et de stocker dans une grille les cases où le torpilleur sélectionné a la possibilité d'attaquer c'est à dire les cases où qui n'ont pas déjà été touchées et où il n'y a pas d'obstacle.

int Jouer_Attaque_Possible(...) :

Fonction qui permet de déterminer si le torpilleur a bien une case où il peut attaquer, dans le cadre où il ne peut pas attaquer on passe son tour.

void Jouer_Init_Curseur(...) :

Fonction qui permet d'initialiser le curseur sur une case où le torpilleur a la portée nécessaire.

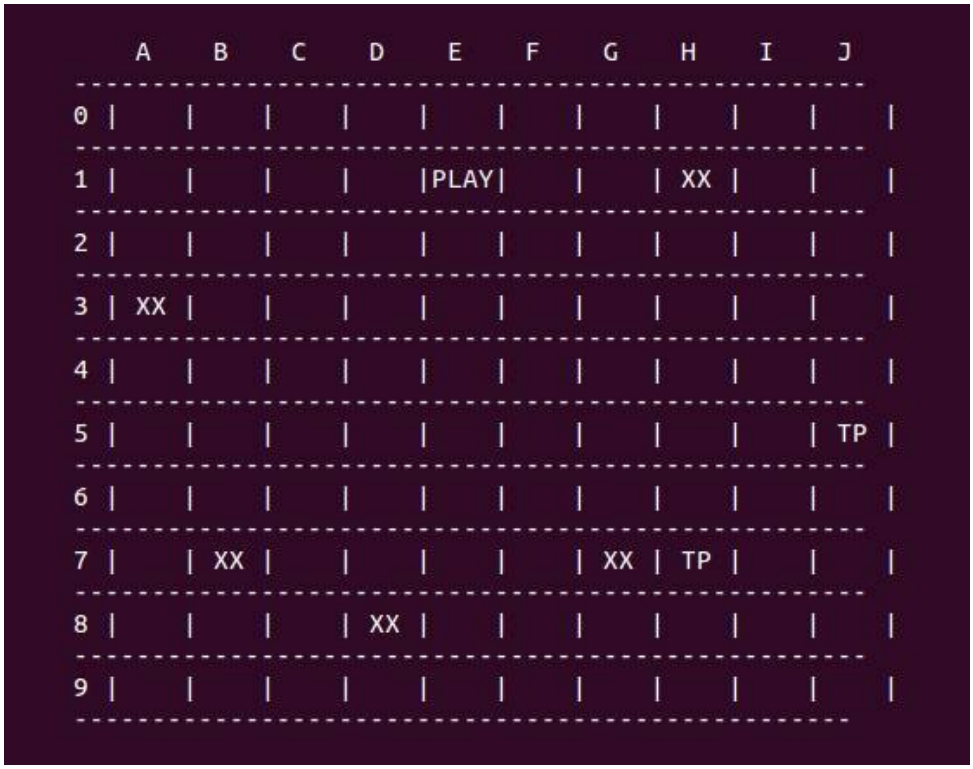
void Jouer_Choisir_Attaque(...) :

Fonction qui permet a l'utilisateur de choisir la case où il souhaite tirer, pour ce faire l'utilisateur déplace à l'aide des flèches du clavier le curseur dans la zone où il a la portée de tirer, ensuite il valide son choix avec entrée.

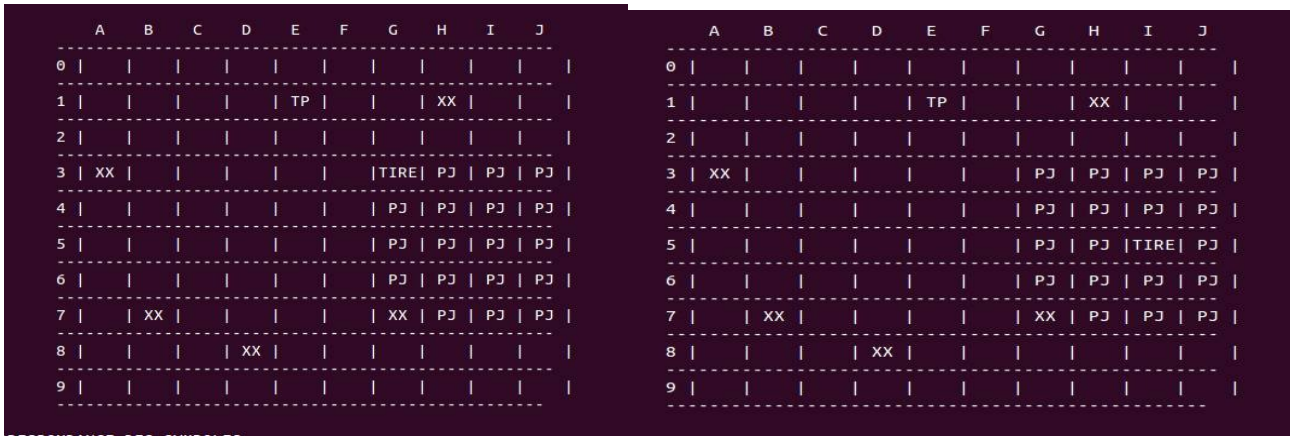
Module Afficher

Au niveau de l’affichage, il est nécessaire d’utiliser une fonction Grille_perso_afficher où l’on entrera en paramètre le numéro de la grille qui permet d’exposer son contenu selon chaque case. On y retrouve les obstacles aléatoirement placés au début de la partie qui sont au même endroit sur la grille de chaque joueur. On peut observer l’emplacement des bateaux ainsi que les informations qui indiquent s’ils ont été touchés par les torpilleurs de l’adversaire.

Une fonction torpilleur_selection_afficher qui possède en paramètre le numéro de la grille de celui qui doit jouer ainsi que des entiers x et y représentant les coordonnées du curseur qui permettront de modifier l’état du torpilleur positionné au niveau du curseur. Le joueur peut alors choisir le torpilleur avec lequel il va attaquer les navires ennemis. Tout comme la fonction qui permet d’afficher le contenu du plateau, on y retrouvera l’emplacement des obstacles, les torpilleurs qu’il possède, les bateaux qu’il a réussi à touché mais également les tirs où il a visé dans le vide.



Et enfin une fonction Portee_torpilleur_afficher où l’on y indique le numéro de la grille à afficher selon l’utilisateur qui joue, une grille contenant les coordonnées de la portée des torpilleurs et des entiers x et y qui sont utilisées pour indiquer l’emplacement de là où ce situe le curseur du joueur qui permettront de changer l’état de la portée selon où ce trouve le curseur. Cette fonction s’utilise après le déplacement de l’un des torpilleurs pour ensuite permettre au joueur d’attaquer son adversaire. Tout comme la fonction qui permet d’afficher la sélection du torpilleur avec lequel on va jouer, on y retrouvera l’emplacement des obstacles, les torpilleurs en sa possession, les bateaux qu’il a réussi à touché ainsi que les tirs qu’il a raté.



Résultats :

Les principales fonctionnalités du programme ont été réalisées, notamment la possibilité de sauvegarder une partie en milieu de jeu, et de la charger lorsque l'on démarre le jeu. Nous avons définit au début du projet que dans le module placer, l'utilisateur devait avoir le choix entre un placement automatique et un placement manuelle, et que lorsqu'il positionne un bateau, du moment qu'il n'est pas satisfait de son placement on lui repropose de positionner le bateau. Nous pouvons voir que lorsque l'on joue au jeu, cette fonctionnalité a été réalisée.
Une fonctionnalité demandée au début a elle aussi été respecté, cette fonctionnalité était que l'utilisateur peut choisir le nombre de bateaux qu'il souhaite positionner, par exemple il peut avoir plusieurs bateaux du même type ou bien posséder un seul bateau du même type.
Dans le module jouer, une fonctionnalité importante était que l'utilisateur puisse réaliser plusieurs déplacement avant de pouvoir attaquer, on avait définit aussi que l'utilisateur pourrait choisir le torpilleur avec lequel attaquer, ces deux fonctionnalités ont été réalisées. De plus nous avons ajouté une fonctionnalité en plus pour enrichir les possibilités du joueur, celle de pouvoir choisir la portée du tir avant de tirer, il a la possibilité de tirer soit à une portée d'une case ou bien de deux cases. Dans le module afficher, les principales fonctionnalités définit lors du début du projet ont été respectées, en effet l'utilisateur peut voir ces différents bateaux répartis sur sa grille, il peut visionner les obstacles ainsi que le(s) torpilleur(s) présent sur la grille de son adversaire, il peut voir les tirs qu'il a réalisé, voir si un des bateaux de son adversaire a été touché, il dispose d'une légende mise à disposition lorsque l'utilisateur à un doute sur la compréhension d'un élément présent sur la grille de jeu.
Malgré les différentes fonctionnalités présentes dans le jeu, il y a plusieurs améliorations à faire dessus, comme par exemple dans le module afficher, il nous manque la possibilité de voir si nous avons coulé un bateau de l'adversaire, de voir les bateaux non coulée qu'il nous reste en notre possession, de rajouter des couleurs sur la grille à l'aide de la librairie ncurses, ainsi que de visualisé notre grille à coté de celle de l'adversaire qui est caché lorsque l'on s'apprête à tirer avec le(s) torpilleur(s). Dans le module placer, nous aurions pu rajouter la possibilité de placer ses bateaux à l'aide du clavier et de la souris selon les préférences des joueurs.

Dans le module jouer il aurait été intéressant d'ajouter la possibilité que lorsque le joueur touche un bateau ennemi il ait la possibilité de continuer à jouer tant qu'il n'a pas tiré dans le vidé (bateau non touché).

Le planning prévisionnel n'a pas pu être respecté, en effet avoir plus de temps consacré à la programmation du jeu nous aurait permis d'avoir la possibilité d'ajouter plusieurs fonctionnalités comme celles énoncées dans la partie ci-dessus.

Conclusion :

Ce projet s'est révélé très enrichissant pour l'équipe, en effet c'est le premier projet en groupe de cette importance que nous ayons réalisé et il a permis de nous confronter a des contraintes du monde du travail. En effet, le respect des délais et le travail en équipe seront probablement des aspects importants de nos prochains futurs métiers.

Malgré le temps consacré en début de celui-ci pour faire la conception du programme, planifier notre travail et répartir les tâches à accomplir nous sommes revenus sur toutes ses composantes en cours de projet ce qui nous a parfois fait perdre beaucoup de temps. Cela montre l'importance de cette phase au début du projet et qu'il faut détailler au maximum toutes les fonctionnalités pour avoir le moins d'imprévu possible. Un autre facteur s'est révélé être très important : la communication. En effet même si nous avions fait une conception écrite de notre projet nous nous sommes rapidement rendu compte que malgré tout chacun avait une idée différente de la conception de celui-ci et c'est pour cela que nous avons fais régulièrement des pauses lors de séance de travail pour ne plus avoir la tête plongée dans son code et de pouvoir parler du projet en général et des interactions entre nos différentes parties.

Nous pouvons également ajouter que le projet nous a permis d'appliquer nos connaissances en programmation et que chaque membre du groupe a amélioré ses compétences générales.