Projet Informatique C - Bataille Navale

Le Guériflit Aquatimer

T. Lefloch, H. Pointereau

25 janvier 2020

Table des matières

1 Résumé du sujet

1.1 Jeu de la Bataille Navale

Les règles :

Dans la bataille navale, chaque joueur pose des bateaux de taille différente horizontalement ou verticalement. Ensuite durant les tirs, les joueurs donnent des coordonnées sous la forme "A10" ou "E6", la partie s'arrête quand tous les bateaux d'un joueur sont coulés.

Déroulement :

La partie commence par la pose réglementaire des bateaux, c'est-à-dire que les bateaux doivent être sur le plateau ou la grille et ils ne doivent pas et ne peuvent pas se chevaucher.
Ensuite c'est l'heure "du-du-du-duel", les 2 joueurs s'affrontent via des tirs, si un des deux joueurs touche son adversaire, il peut immédiatement tirer jusqu'à ce que l'un de ses tirs ne touchent pas. La partie s'arrête quand un des deux joueurs n'a plus de bateaux sur le plateau.

Objectif :

Le but de la bataille navale est de couler tous les bateaux de l'adversaire avant que celui-ci coule les nôtres.

2 Description du programme, fonctionnalités et utilisation :

Le programme se divise en différentes parties :
— Le menu et l'interface utilisateur
— Le placement des bateaux et la flotte
— La gestion du tir et du déroulement de la partie
— Les intelligences artificielles
— La sauvegarde et le chargement de parties

2.1 Le menu et l'interface utilisateur :

Parlons tout d'abord du nom du projet, "Le Guériflit Aquatimer", ce dernier à été généré à l'aide du programme créé au S1 de PREPA1, permettant de générer, à partir d'un dictionnaire de mots, de nouveaux mots ressemblants à de vrais mots, mais n'ayant aucun sens. Ici, nous avons utilisé le dictionnaire des synonymes de "Bataille" et de "Navale, afin de générer "Guériflit" et "Aquatimer" [1]. Afin de rendre l'expérience de jeu la plus fluide et intéressante possible, nous avons choisi de faire un menu interne à l'application.
Affiché directement dans le terminal à l'exécution, ce dernier permet de choisir entre les différents modes de jeu (Joueur contre Joueur, contre IA aléatoire, contre IA intelligente). Il permet également de charger ou non une sauvegarde **??** Nous avons utilisé un générateur de texte ASCII Art [2] ainsi qu'une batterie de couleurs afin de le rendre plus agréable !
Le design de la grille à également été soigné, cette dernière est décomposée en 4 couleurs :

— ■ Eau non touchée.
— ■ Eau touchée.
— □ Bateau intact.
— ■ Zone de bateau touchée.

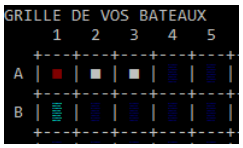

FIGURE 1 – Vue flotte locale

Nous avons également choisi d'apporter une touche d'art au niveau du retour d'information lorsque l'on rate un tir, touche ou coule un bateau. En continuant dans la même lancée que le menu, nous avons réutilisé l'ASCII Art afin de réaliser : une goutte d'eau si le tir est manqué, avec une inscription "*Splash !*", un bateau enflammé pour le tir réussi avec une inscription "*Hit !*" ainsi qu'un champignon nucléaire et une inscription "*Boom !*" si l'on coule un bateau. - voir Figure 2

FIGURE 2 – ASCII Art

1. Se prononce *[Akoua-ti-mère]*
2. Générateur texte ASCII [#]

2.2 Placement des bateaux et gestion de la flotte :

Gestion de la flotte :

Le joueur doit tout d'abord générer sa flotte. Il a alors deux options, soit il joue avec une flotte personnalisable à 100% (nom, taille, nombre de bateaux), soit il choisit de jouer avec la flotte par défaut. À noter également que pour choisir le nombre maximum de bateaux sur la grille, pour ne pas que l'utilisateur se retrouve avec trop de bateaux par rapport à la taille de sa grille, on calcule :

$$nombreBateaux = round((tailleGrille)^2/10) \quad (1)$$

Ce qui permet d'avoir toujours un nombre de bateaux plaçables dans la grille. On initialise la flotte via un tableau dynamique, de taille variable selon la taille de cette dernière, et on demande à l'utilisateur de rentrer le nom et la taille de chacun des bateaux. Ceci est alors stocké dans une structure contenant 3 champs, son nom, sa taille et son statut (coulé pour le J1, le J2, les deux ou non).

Gestion des coordonnées :

Toujours dans l'optique d'améliorer l'expérience utilisateur, nous avons choisi de permettre au joueur de rentrer ses coordonnées sous la forme "*A1*". On découpe alors la lecture des coordonnées en 4 parties, récupération, séparation et conversion et vérification, si la vérification échoue, on recommence. - voir Figure 3

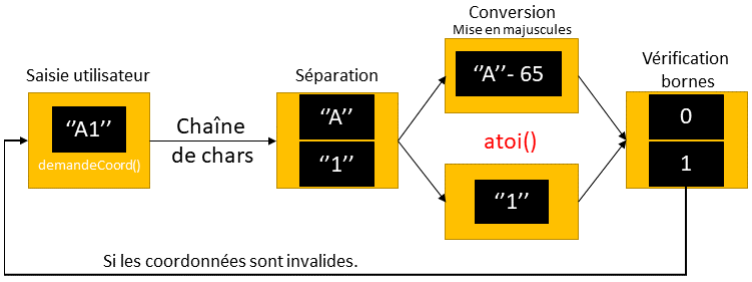

FIGURE 3 – Méthode conversion des coordonnées

Gestion de la grille et placement :

Nous avons choisi de ne pas utiliser de tableau de structure pour notre grille et ainsi placer les bateaux dans la grille à l'aide de valeurs particulières, nous avons donc déclaré les constantes suivantes :

- BATEAU_VIVANT 1
- BATEAU_TOUCHE 2
- EAU_VIVANTE 0
- EAU_MORTE -1

Ces constantes nous permettent alors de manipuler la grille simplement, avec des entiers.
Pour placer les bateaux, l'utilisateur choisi l'orientation de son bateau (horizontale ou verticale) ainsi que la coordonnée ou placer le début de son bateau.

Ensuite, une fonction permet de vérifier que l'intégralité des cases autour du bateau à placer sont bien vides en utilisant la méthode des voisins (reprise du Jeu de la vie), afin de respecter la consigne du couloir d'eau. Si la condition est remplie, la grille est mise à jour, et le joueur place son bateau suivant, sinon il recommence.

2.3 La gestion du tir et du déroulement de la partie :

La gestion du tir :

Pour notre gestion du tir, nous l'avons décomposée en plusieurs fonctions distinctes :

— La validité des coordonnées du tir : savoir si les coordonnées rentrées sont bien dans notre tableau et sinon demander à l'utilisateur dans donner d'autres jusqu'à se qu'elles soient valable.

— Le tir en lui même : l'effet qu'il affecte sur notre grille c'est-à-dire le changement d'état, il peut faire passer de BATEAU_VIVANT 1 à BATEAU_TOUCHE 2 ou alors de EAU_VIVANTE 0 à EAU_MORTE -1.

— Le compte rendu du tir, s'il est manqué, s'il a touché ou bien s'il a coulé. Si le tir est manqué, cette fonction nous retournera "-taille de la grille - 3", le moins 3 vient du faite que la fonction s'incrémente sur chaque côté du tir, il faut donc au final enlevé les incrémentassions de trop. Si le tir touche alors il sera différent de "-taille de la grille - 3" et pour finir, si le tir coule un bateau, il retourna la taille entière du bateaux.

— La demande des coordonnées qui est la même que celle vue ci-dessus. - voir Figure 3

— La fonction qui regroupe toutes celle citées ci-dessus pour effectuer un tir valide.

La gestion du déroulement de la partie :

Dans notre version de la bataille navale, la partie ne se finit que lorsqu'un des deux joueurs n'a plus de bateaux, notre fonction de fin regarde donc dans toute la grille s'il reste encore des BATEAU_VIVANT, s'il n'y en à pas, l'adversaire gagne la partie.

Pour le reste de la partie, elle marche via une alternance de tir des joueurs, si un joueur manque son tir, c'est au tour de son adversaire de jouer mais s'il manque alors il retire jusqu'à ce qu'il manque son tir ou qu'il termine la partie s'il est chanceux.

Chaque joueur est spécifié quand c'est à lui de tirer mais quand c'est contre une IA, le tir de l'IA est fait tout seul en arrière plan.

2.4 Les intelligences artificielles :

L'intelligence artificielle aléatoire :

Pour cette première intelligence artificielle au comportement "simpliste", elle ne fait que tirer de façon aléatoire à chaque tir, elle ne fait rien d'autre, c'est déjà pas mal.

L'intelligence artificielle intelligente :

Pour notre intelligence artificielle plus intelligente, nous sommes partie en utilisant la base de l'IA aléatoire, c'est-à-dire que notre IA a un comportement de tir aléatoire jusqu'à se qu'elle touche un bateau, et des qu'elle touche, elle regarde autour pour savoir si il y a des morceaux de bateaux, si oui, elle continue à tirer dans la bonne directions jusqu'à se qu'elle tombe sur de l'eau, puis elle repasse en mode aléatoire jusqu'à rencontrer un autre bateau pour recommencer cette procédure.

2.5 La sauvegarde et le chargement de parties :

N'avez-vous jamais rêvé de pouvoir arrêter ce que vous êtes en train de faire, et de le reprendre plus tard sans aucun problème ? C'est le but d'une sauvegarde et nous l'avons fait !

Sauvegarde :

La sauvegarde est proposée automatiquement à la fin de chaque séquence de tir, le joueur peut alors sauvegarder ou continuer de jouer. Si il choisit de sauvegarder, le fichier de sauvegarde est généré d'une seule traite et comporte l'intégralité des données de la partie - voir Figure 4.
Son nom n'est pas choisi aléatoirement, ni par l'utilisateur, afin de se rapprocher le plus des "formats" de saves utilisés dans les jeux-vidéos, la sauvegarde se nomme de la manière suivante : "*.\saves\GAMESAVE-$DATE-$HEURE.sav*", ou $DATE et $HEURE sont la date et l'heure du système. Ainsi il est plutôt facile de retrouver une partie précédemment sauvegardée !

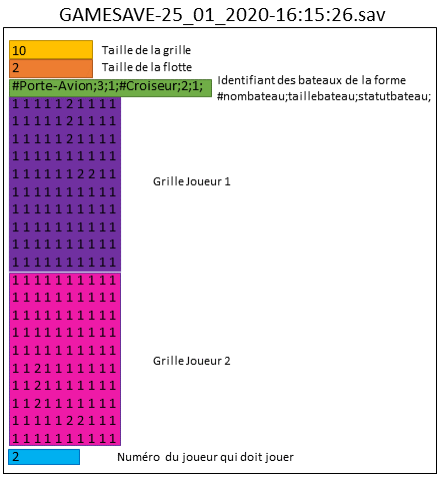

FIGURE 4 – Format de données pour la sauvegarde.

Chargement de parties :

Le chargement d'une partie précédemment sauvegardée est proposée au démarrage du jeu, après sélection d'un des modes de jeu. Le programme affiche la liste des fichiers présents dans le dossier "*./saves*", et laisse l'utilisateur entrer le nom du fichier à charger. Le programme effectue alors une première vérification, qui est celle de l'extension, qui doit être ".sav".
Ensuite, le programme ouvre le fichier et cherche la première ligne, il va ensuite lire au fur et à mesure l'ensemble des informations selon le formatage défini par le fichier de sauvegarde. Une fois le chargement terminé, on peut reprendre la partie normalement.

NB :

Au cas où l'on souhaiterait sauvegarder une partie qui à été chargé précédemment, le fichier de sauvegarde précédent sera automatiquement écrasé ! Plutôt pratique pour ne pas se perdre dans ses fichiers de sauvegarde ! À noter que si l'on fini la partie, le fichier est supprimé également.

Remarquez également que même si cela paraît bizarre au premier abord, le programme propose de sauvegarder même en JcIA, lorsqu'elle tire, le statut du tir apparaît et on nous propose tout de suite de sauvegarder, c'est normal, l'IA à joué et vous pouvez sauvegarder en suivant !

3 Difficultés, points faibles et points forts :

3.1 Difficultés :

Les difficultés rencontrées lors de notre projet sont l'élaboration d'un makefile sans aucunes connaissances, l'implémentation des sauvegardes avec des noms de fichiers intelligents et une syntaxe de sauvegarde prenant en charge les flottes personnalisées, mais également le fait de pousser l'ergonomie au maximum, en effet nous avons souhaité rendre le jeu agréable à jouer et à regarder, ce fut un véritable défi.

3.2 Points faibles :

Pour nos point faibles, le fait qu'on ne puisse pas initialiser une partie depuis un fichier texte mais seulement par l'action des/du joueur(s) et aussi le fait que notre IA intelligente ait quelques bugs et qu'on aurait pu peut-être faire une IA encore plus intelligente et performante.

3.3 Points forts :

Pour nos points forts, nous avons notre système de sauvegarde qui est performant et qui est une fonctionnalité agréable pour l'utilisateur. L'ergonomie que nous avons essayé de rendre user friendly au maximum : tout est clair, l'utilisateur à toujours des indications sur les actions qu'il peut ou doit faire et en ajoutant des ASCII Arts ainsi que des couleurs pour le rendre plus vivant. Et pour finir, le fait que notre bataille navale peut-être totalement personnalisé, on peut choisir la taille de la grille, la taille de chaque bateau ainsi que son nom, ce qui peut rendre chaque partie unique !

4 Contributions de chaque élèves :

Afin de se répartir au mieux les tâches et dêtre le plus efficace possible dans notre travail, nous avons donc utilisé GitLab afin de mettre le code à jour à chaque nouvelle fonctionnalité.
Voici le lien du Git : https://gitlab.etude.eisti.fr/projetdec1/bataillenavale.git

T.LEFLOCH

Thomas s'est chargé du placement des bateaux ainsi que de l'élaboration de la sauvegarde et de son chargement. Il a également aidé Hugo au niveau de l'affichage et du menu et élaboré le makefile.

H.POINTEREAU

Hugo s'est chargé de faire les fonctionnalités de combat, de l'intelligence artificielle ainsi que le menu, les différents mode de jeu et l'affichage avec en coopération avec Thomas.

5 Bilan personnel :

T.LEFLOCH

Ce projet m'a permis d'améliorer mes compétences en lecture de fichiers et de la gestion de mémoire, en effet, la sauvegarde s'est avérée être plutôt complexe si on se concentrait sur l'optimisation mémorielle de la chose (tout faire en malloc, sans allouer de la mémoire inutilement), j'ai également progressé au niveau de ma réflexion sur l'ergonomie et l'expérience utilisateur et j'ai hâte d'en apprendre plus dans ce domaine.

H.POINTEREAU

Pour moi, ce projet m'a permis de fixer mes connaissances en C ainsi que de les améliorer sur certains points importants, j'ai pu m'améliorer sur les tableaux dynamiques, l'utilisation de pointeurs. Et m'a permis d'apprendre de nouvelle fonctionnalité et implémentassions possible en C via mon camarade Thomas.