

Rapport du projet Bataille Navale (IA)

Étudiants : Quentin Coloos - Alexandre Tholliez - Thomas Prunier - Benoît Verhaghe

Encadrant : Ahmad Mazyad

Date de début : 29 mai 2017

Date de fin : 16 juin 2017

Objet du projet : Création d'un jeux vidéo avec une Intelligence Artificielle

Table des matières

1 Présentation du projet	2
1.1 Analyse de la demande	2
1.2 Contexte	2
1.3 Spécifications	2
2 Réalisation	3
2.1 Présentation	4
3 Bilan	6
3.1 Déroulement du projet	6
3.2 Réalisation des objectifs	6
3.3 Conclusion pour les projets futurs	7

1 Présentation du projet

1.1 Analyse de la demande

1.2 Contexte

Dans le cadre du projet de fin de Licence, il nous était demandé de choisir entre :

- Le développement d'un jeu vidéo en réseau en C++
- Le développement d'un anti-spammer.
- Le développement d'un jeu vidéo avec une Intelligence Artificielle avec langage de programmation au choix.

Dans ces 3 projets, seul l'anti-spammer ne nous intéressait pas du tout. Nous devions donc choisir entre l'IA ou le réseau, n'ayant aucune connaissance en réseau nous avons décidé de nous orienter vers le projet avec une IA.

Le langage de programmation étant au choix, nous avons hésité entre le Java et le C++. Cependant comme nous venions de passer tout le semestre à apprendre le C++ et qu'il était possible d'utiliser les librairies GTKMM et SFML, nous avons choisi celui-ci.

Besoins et priorités

Pour le projet, il nous était demandé de réaliser un jeu avec une Intelligence Artificielle et relié à une interface graphique. Le langage C++ étant déjà choisi et la librairie GTKMM utilisable (nous avons par la suite changé pour SFML), nous avions déjà les bases définies afin de créer l'interface graphique. Nous devons également créer une Intelligence Artificielle avec plusieurs niveau de difficultés (facile / moyen) .

1.3 Spécifications

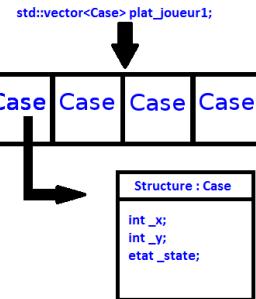
1. le jeu doit fonctionner sur toute machine avec une distribution linux
2. fonctionnalités du jeu de bataille navale :
 - Choisir le mode de jeu
 - Joueur contre IA
 - IA contre IA
 - Jouer une partie
 - Initialisation d'un plateau de jeu 9x9
 - Placement des bateaux avec choix de la direction (NORD / EST / SUD / OUEST)
 - Un Porte-avion(longueur 5)
 - Un Croiseur (longueur 4)
 - Deux Sous-marins (longueur 3)
 - Un Torpilleur (longueur 2)
 - Jouer un coup
 - Vérification du coup (touché ou raté ou coulé ou victoire)
 - Jouer coup suivant tant qu'aucun joueur n'a gagné la partie
3. Interface utilisateur :
 - Interface graphique
 - Affichage du plateau avec les tirs et bateaux du joueur
 - Choix de la difficulté
4. Performances demandées :
 - vérification de la légalité des coups
 - IA avec plusieurs niveaux de difficulté

2 Réalisation

Nous avons donc choisi de faire une bataille navale accompagné d'une IA.

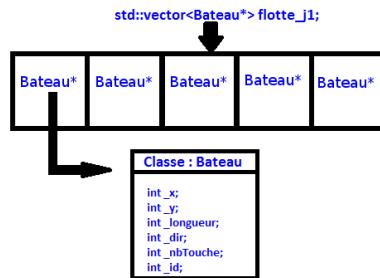
Pour jouer nous avons donc besoin de plusieurs choses. Premièrement deux plateaux de jeux, le premier qui contient nos bateaux et les tirs de l'adversaire. Deuxièmement une fonction permettant de placer nos bateaux. Troisièmement une fonction permettant d'effectuer un tir et de vérifier si nous avons raté/touché/coulé un bateau ou gagner. Et quatrièmement une IA fonctionnelle.

Nous avons choisi d'utiliser des vectors afin de créer nos différents plateaux de jeu et de créer plusieurs flottes de bateaux. Nous les utilisons car ceux-ci sont des tableaux dynamiques où il est particulièrement aisés d'accéder directement aux divers éléments par un index, et d'en ajouter ou en retirer à la fin. A la manière des tableaux de type C, l'espace mémoire alloué pour un objet de type vector est toujours continu, ce qui permet des algorithmes rapides d'accès aux divers éléments.



Nous avons donc une case qui prend des coordonnées x,y afin de nous repérer dans le vecteur, ensuite nous avons créé une énumération "etat" qui prends les valeurs "VIDE, RATE, TOUCHE, BATEAU, COULE". Donc grâce à notre vecteur nous avons accès à la position de la case puis son état.

Pareillement pour le bateau, nous avons ses coordonnées (x,y), sa longueur (2,3,4 ou 5), une direction (NORD, EST, OUEST, SUD) qui ont été définie dans un fichier qui contient les variables globales. Le nombre de fois que le bateau a été touché et un id. Quand un bateau reçoit un tire on l'identifie grâce à son id, puis celui-ci incrémentera la valeur du nombre de fois qu'il a été touché. Si la longueur du bateau est égale au nombre de fois qu'il a été touché, alors on supprime le bateau vecteur et change les cases correspondantes en "COULE".



Nous avons également une classe Joueur qui nous permet de savoir si c'est le joueur 1 ou le joueur 2 qui joue le tour. Puis un type qui est égale soit à 0 (HUMAIN) soit à 1 (IA).

Grâce à ces éléments nous avons de quoi créer le plateau de jeu des joueurs, nous pouvons également créer et placer nos bateaux, ainsi que choisir le mode de jeu (IA vs IA ou Humain vs IA).

2.1 Présentation

Après compilation grâce au Makefile et que nous exécutons notre programme nous obtenons notre menu de jeu.



Nous pouvons alors soit jouer une partie, soit quitter le jeu.

Si l'utilisateur choisit de jouer une partie alors il est redirigé vers d'autres menus :

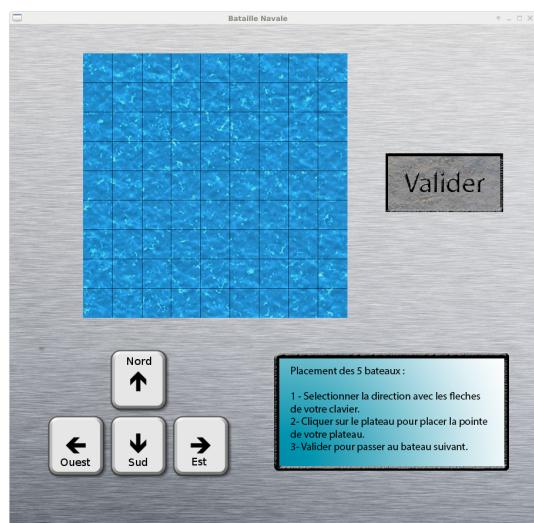
- Il devra alors choisir la difficulté de l'IA (Facile / Normal)
- Ensuite il choisit si le jeu est : Joueur vs IA ou IA vs IA



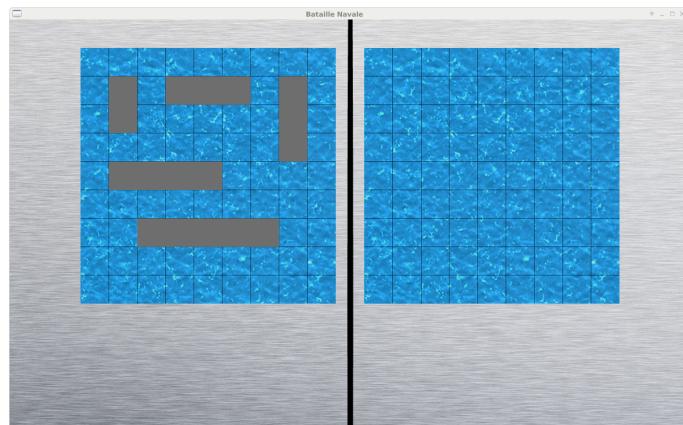


Une fois les modes de jeux choisis on arrive à l'écran de placement des bateaux :

- Il peut choisir grâce aux flèches l'orientation du bateau qu'il veut placer
- Il peut cliquer sur la case souhaitée pour placer son bateau puis valider le placement de celui-ci



Une fois que le placement est terminé, la partie peut se lancer.



Enfin, comme convenu dans le cahier des charges, le logiciel fonctionne sur l'environnement Linux avec la bibliothèque SFML

3 Bilan

3.1 Déroulement du projet

Problèmes rencontrés et solutions adoptées :

- Pendant les 10 premiers jours du temps imparti pour réaliser notre projet, nous avons utilisé la librairie GTKMM pour réaliser notre interface graphique. Nous avons choisi cette librairie car c'est celle qui nous avions étudiée en cours, nous connaissions donc déjà les bases. Cependant, nous nous sommes aperçus que cette librairie était mal adaptée pour notre projet, notamment pour relier notre interface graphique à notre code. Nous avons donc opté pour une solution radicale : tout recommencer avec une nouvelle librairie qui nous semble mieux écrite en langage C++, la librairie SFML.
- Avec SFML, nous avons eu du mal à récupérer la grille où nous avions placé les bateaux pour la remettre sur la fenêtre suivante : celle du jeu. Pour résoudre ce problème, nous avons rassemblé menuPlateau.cpp et menuJouer.cpp. De cette manière, les deux fenêtres sont codées dans le même cpp et la grille de placement des bateaux est facilement récupérable.
- Nous avions un problème de placement des bateaux étant donné qu'ils étaient placés dans un vecteur et pas un tableau. Nous avons dû trouver la formule pour placer la suite de notre bateau dans les bonnes cases
- Pour l'IA, nous avons eu beaucoup de mal avec Min-Max, puisque la bataille navale est un jeu basé sur l'aléatoire

Différences par rapport aux prévisions (conception, planification...) :

- On utilise des rectangles pour symboliser les bateaux, et non pas des images de bateaux.
- Aucune implémentation de bruits/bruitages lors des tirs.
- Pas de gestion d'erreur lors du placement des bateaux.
- L'intelligence artificielle a demandé plus de temps que prévu, en particulier à cause de Min-Max

3.2 Réalisation des objectifs

fonctionnalité	réalisation
Projet de base	complète
Interface graphique	partielle
IA	partielle

Pour l'interface graphique, il manque la vérification lors du placement des bateaux : si on place un bateau en dehors de la grille ou sur un autre bateau, on obtient une erreur de segmentation. De plus, lorsqu'on place les bateaux, il faut cliquer le plus légèrement possible sur valider, car si on reste appuyé trop longtemps, l'interface va détecter plusieurs cliques et certains bateaux pourraient ne pas être placés.

A l'heure actuelle, il est impossible de jouer à notre jeu via l'interface graphique, car par manque de temps, nous n'avons pas entièrement relié le jeu console avec l'interface graphique.

Ce qui est fonctionnel dans l'interface graphique :

- Le placement des bateaux pour le joueur
- Le choix de la difficulté
- Le choix du mode de jeu

Ce qui n'est pas fonctionnel dans l'interface graphique :

- Le placement des bateaux pour l'IA
- Jouer une partie

Pour l'IA, nous rencontrons un problème : l'IA ne va pas jusqu'au bout de la victoire

3.3 Conclusion pour les projets futurs

Ce qui a bien marché :

- La cohésion de groupe
- La répartition des tâches

Les erreurs à ne plus commettre :

- Ne plus se précipiter sur une bibliothèque en début de projet sans réfléchir.
- Github : Ne pas faire le projet sur la branch master (comme nous avions fait les premiers jours), surtout si nous travaillons en plusieurs groupe