PICARD Lucas
GERVY Tiago

Seconde Pro B (CIEL)
Documentation

BATAILLE NAVALE

Sommaire :

I. Introduction

Du 10 mars 2025 au 25 mai 2025, un projet nous a été donné, celui de créer un jeu Bataille Navale codé en Python à l'aide des logiciels PYcharm et Visual Studio Code. L'objectif était de reproduire le jeu Bataille Navale avec les vraies règles. Le jeu doit contenir une interface graphique et deux IA afin d'avoir un niveau facile et un niveau difficile. Les IA doivent posséder une logique qui fait que lorsqu'un bateau est touché, elle doit tirer autour pour couler le bateau.

II. Comparaison

A. Diagramme de GANTT

Avant de réaliser le code nous avions fait un diagramme de GANTT afin de posséder des dates avec des objectifs de production (de tels à tels dates l'interface etc …).

Nous avions fait un diagramme avec les objectifs de réaliser :

→ Interface graphique	Du 17/03 au 02/04
→ Menu	Du 03/04 au 19/04
→ Amélioration de l'interface graphique	Du 20/04 au 06/05
→ Le jeu	Du 17/03 au 02/04
→ Les IA	Du 03/04 au 06/05
→ Facile	Du 03/04 au 19/04
→ Difficile	Du 20/04 au 06/05
→ Mise en commun	Le 07/05
→ Rassemblement des codes	Du 07/05 au 12/05
→ Livraison du projet	Le 12/05
→ La documentation	Du 13/05 au 18/05
→ La préparation et passage à l'oral	Du 19/05 au 25/05

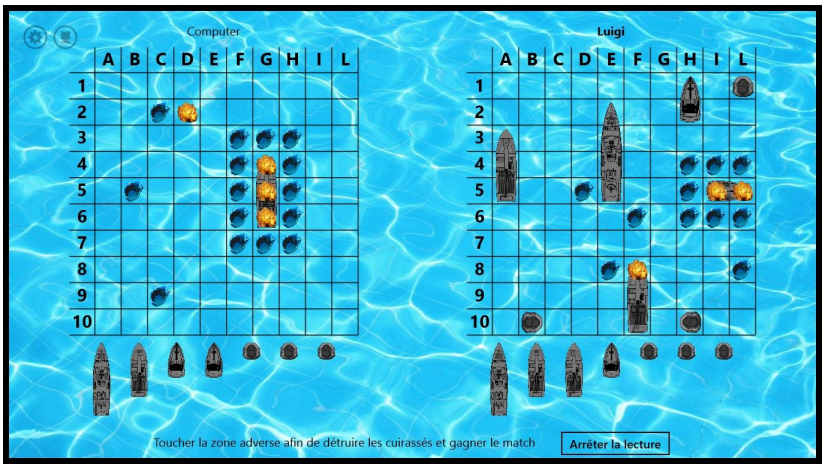
Lors de la réalisation du projet nous nous sommes rendu compte qu'énormément d'étapes étaient nécessaires. Nous n'avons pas imaginé que cela puisse être d'une telle complexité.

Nous avons donc commencé par l'interface graphique,puis le jeu et enfin IA qui nous ont bien embêtés, à cause de leur mode de chasse assez complexe.

B. Interface Graphique

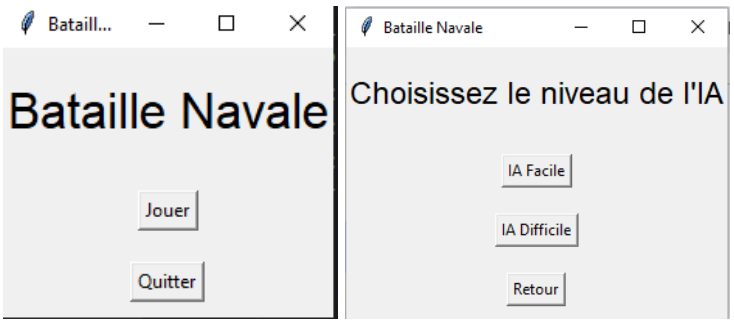
Interface prévu:

(image exagérée)

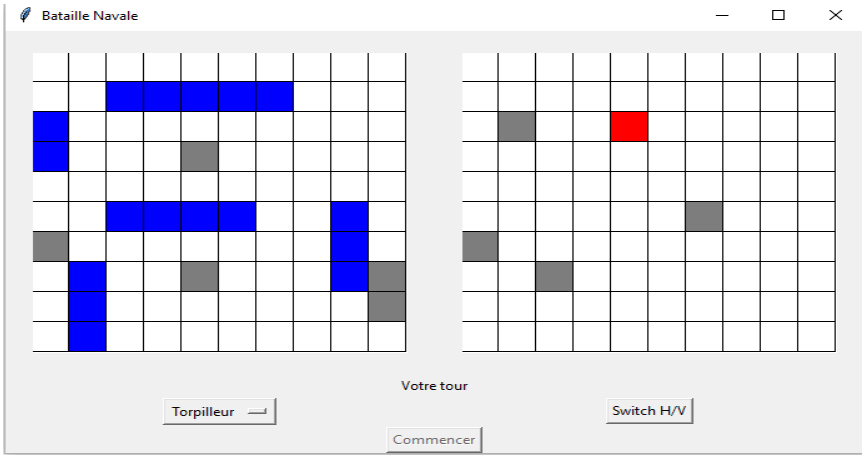


Interface réalisé:

Menu:



Champ de bataille:



Nous avions prévu une interface graphique avec plus de couleur, plus dynamique, plus d'option mais surtout avec une interface plus developer.

Options prévues:

→ Menu complet
- → Choix du niveau
- → Statistique
- → Quitter

→ Déplacer les bateaux pour les placer
→ Revenir au menu
→ Bouton rejouer
→ Belle affichage lorsque c'est toucher ou couler

Option réalisées:

→ Menu
- → Choix du niveau
- → Quitter

→ Cliquer pour placer les bateaux
→ Affichage lorsqu'un bateau est coulé

III. Explications du jeu

A. Règles du jeu

La bataille navale est un jeu pour deux joueurs. Chacun a deux grilles : une pour placer ses bateaux, et une pour marquer ses tirs. Chaque joueur place ses bateaux en secret, horizontalement ou verticalement, sans les coller. Les bateaux ont différentes tailles. Par exemple, le plus grand prend 5 cases, et le plus petit en prend 2.

Quand la partie commence, les joueurs jouent à tour de rôle. À chaque tour, un joueur dit une case, comme "C3". Si un bateau ennemi est sur cette case, c'est "touché". Si toutes les cases d'un bateau sont touchées, il est "coulé". Si aucun bateau n'est là, c'est "à l'eau".

Le but du jeu est de trouver et couler tous les bateaux de l'adversaire. Le premier qui y arrive a gagné.

B. Fonctionnement du jeu

Nous avons créé notre code en repositionnant chaque élément par paragraphe afin de pouvoir nous y repérer (où commence telle et telle fonction etc…).

Nous avons tout d'abord importé tkinter qui nous a permis de réaliser l'interface graphique mais aussi la fonction messagebox de tkinter qui permet d'afficher des messages, les bibliothèques random et copy.

Nous avons ensuite créé un dictionnaire qui nous a permis de stocker les informations des bateaux, c'est-à-dire leur nom et leur taille.

Nous avons une classe Cell qui représente chaque cellule des grilles. Cette classe permet de contenir les informations de si il y a un bateau sur cette case, si il y a eu un tir et si il y a un bateau elle retient son nom.

La classe Board représente chaque grille, la notre et celle de l'adversaire. Elle nous permet d'accéder à plusieurs informations. Cette grille nous permet de placer les bateaux, gérer les attaques et vérifier si un ou plusieurs bateaux est coulé.

Arrive ensuite la classe BattleshipGame, cette classe est en faite la classe principale du jeu, cette classe gère toute l'interface graphique et la logique du jeu. Elle contient l'affichage du menu principal, la sélection de l'IA, l'initialisation du plateau et de l'interface, la gestion du placement des bateaux par joueur, le placement des bateaux et du lancement de la bataille, l'affichage des grilles et des couleurs personnalisés en fonction de la présence d'un bateau ou non si il y a eu un tir raté, touché ou si le bateau est coulé et enfin la gestion de l'ordre des joueurs (chacun son tour).

Il y a ensuite les IA

→ L'IA facile tire aléatoirement et si elle touche un bateau alors elle passe en mode chasse en testant les cases autour afin d'essayer de couler le bateau malgré quelques soucis (elle n'arrive pas à rester dans la même direction une fois qu'elle est trouvée).

→ Pour l'IA difficile, nous avons essayé de trouver une méthode qui rendrait les tirs aléatoires plus efficaces. Nous avons donc pensé que comme il n'y existe pas de bateau de la taille d'une cellule il suffisait de tirer une fois sur deux afin d'optimiser les chances de toucher un bateau plus rapidement. Et si elle touche un bateau elle active le même mode chasse que l'IA facile afin d'essayer de couler le bateau trouvé.

Et enfin les lignes de code qui nous permettent de lancer le jeu. Ces dernières ligne de code nous permettent de créer la fenêtre que nous avons pu personnaliser avec tkinter.

C. Fonctions intéressantes et compliquées

La fonction "place_boat" de la classe "Board" :

Cette fonction gère les placements de bateaux sur la grille. Elle vérifie que le bateau ne déplace pas la grille et elle s'assure qu'il n'y a pas déjà un bateau là où l'on veut (ou l'IA) poser le bateau. Si les conditions sont remplies alors le bateau est placé (la case enregistre duquel il s'agit) et renvoi "True" pour informer qu'il y a un bateau. Si les conditions ne sont pas remplies alors le bateau ne se pose pas.

La fonction "easy_ai_turn" :

Cette fonction gère l'IA facile en utilisant deux modes qui sont le mode normal et le mode chasse. Le mode chasse est le mode par défaut de l'IA facile qui fait que lorsque c'est son tour, elle tire sur n'importe quel case, et si elle touche un bateau alors le mode chasse s'active. Le mode chasse est utilisé pour essayer de couler le bateau adverse, l'IA va faire en sorte de tirer autour de la ou elle a touché pour faire couler le bateau complètement.

La fonction "hard_ai_turn" :

Cette fonction gère l'IA difficile en utilisant deux modes également, le mode de tir normal qui est le tir par défaut et de chasse. Le mode chasse reste le même tandis que le mode de tir par défaut est plus complexe car l'IA ne tire que sur des cases qui ne sont jamais à côté du style damier (sauf pendant le mode chasse) car il n'y a pas de bateau d'une cellule et donc elle essaie d'avoir un tir plus optimisé afin de couler les bateaux du joueur plus rapidement ce qui force le joueur a trouver une meilleure logique lors de son placement de bateaux et de tirs afin d'avoir un peu plus de difficulté.

La fonction "draw_grid" :

Cette fonction gère tout l'affichage graphique du plateau de jeu grâce à tkinter. Elle colorie chaque cellule en fonction de son état c'est-à-dire blanc pour une cellule vide, gris pour un tir raté, bleu pour un bateau non touché et rouge
pour un bateau touché. Cette interface nous permet de savoir où en ai le jeu sans difficultés malgré une absence de légende.

IV. Notre ressenti

"Je pense que l'on a plutôt bien réussi notre projet de bataille navale, même si pour moi on aurait pu faire beaucoup mieux en tenant compte du temps qui nous était imparti. Je pense que l'on a plutôt bien réussi notre interface graphique, mais que l'IA n'était pas assez bien à mon goût. Et surtout, le plus gros problème de ce projet est que l'on a trop eu recours à l'IA pour réaliser notre projet, et donc je ne suis pas très fier du rendu et de l'impression d'être mauvais."

Lucas PICARD

"Personnellement je trouve que cette idée de projet était très intéressante seulement je trouve que je n'avais pas les compétences nécessaire pour ce projet. J'ai donc dû faire appel à l'intelligence artificielle et je n'en suis pas très fier. Je sais que ce n'est pas de moi que vient ce projet mais d'une IA. J'ai le même ressenti que Lucas dans l'impression d'être mauvais et j'ai aussi l'impression de ne pas avoir progressé depuis les anciens projets du style des petits exercices de début d'année qui nous permettait d'apprendre un petit peu de tout ou depuis le projet Pokémon"

Tiago GERVY

V. Conclusion

A. Difficultés rencontrées

Nous avons eu énormément de soucis lors du codage car manque de connaissance et c'était également notre premier gros projet donc nous avons eu du mal à imaginer les différentes tâches nécessaires au bon maintien du projet.

B. Compétences apportées

Nous sommes du même avis pour dire que ce projet était particulièrement complexe dans la programmation car nous avons un grand manque de connaissance et d'expérience. Nous souhaitons avoir l'occasion une prochaine fois de recommencer un projet du même style mais moins compliqué afin de progresser et de ne pas rester sur une mauvaise expérience.

PICARD Lucas

GERVY Tiago