

Projet IHM : Jeu de navigation

Réalisé par BRACQ Paolo et REVERET Pablo

Introduction :

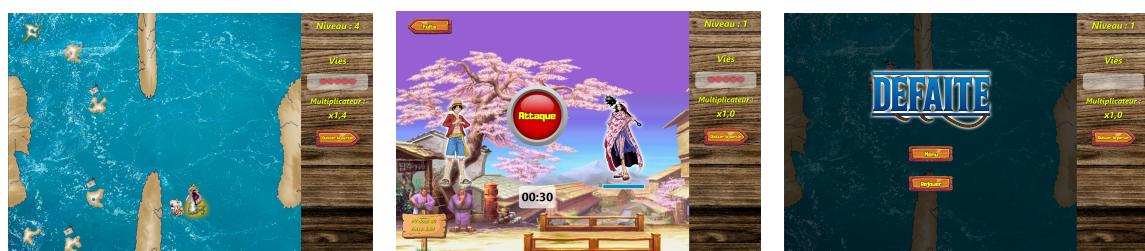
Notre jeu intitulé "Grand Line" s'inspire de l'univers de One Piece créé par Eiichirō Oda. Le joueur incarne un bateau pirate qui explore un monde semi-ouvert pour affronter des boss répartis sur différentes îles. Chaque combat permet de gagner des niveaux, avec un multiplicateur de puissance croissant. Le but final est d'atteindre l'île du One Piece, qui n'apparaît qu'une fois tous les boss battus.

Répartition du travail :

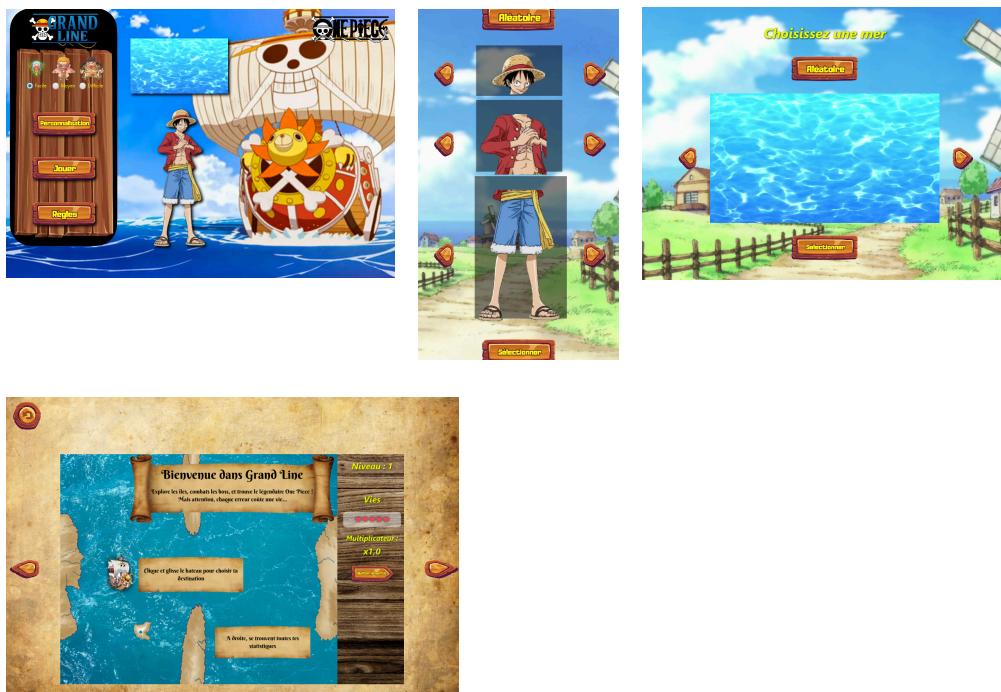
#	Tâche	Responsable	Date début	Date fin	Nbr jours	Status	% réalisé
1	Fenêtre principale menu	Pablo	26/05/2025	01/06/2025	6	Validé	100%
2	Créer la carte principale	Pablo	26/05/2025	01/06/2025	6	Validé	100%
3	Fenetres personnalisations	Pablo	26/05/2025	03/06/2025	8	Validé	100%
4	Créer personnage bateau, île,...	Pablo	26/05/2025	03/06/2025	8	Validé	100%
5	Gérer la personnalisation	Pablo	04/06/2025	08/06/2025	7	Validé	100%
6	Création Drag and Drop + Combat+final	Pablo	04/06/2025	08/06/2025	4	Validé	100%

Vues et Interfaces:

Voici trois vues réalisées par Paolo, l'une correspond à la carte principale en milieu de partie, et l'autre correspond à l'interface d'un combat et la dernière à une possible vue de fin de partie:



Ensuite ces différentes vues réalisé par Pablo correspondent au menu principal, aux règles, à la personnalisation du personnage et de la carte :



Cahier des charges & ajustements:

I. Composants visuels

- Interaction drag & drop : le joueur déplace son bateau d'une île à une autre via glisser-déposer.
- Utilisation d'images personnalisées : images pour les personnages, le fond de la carte, les îles, les boss, le bateau, les boutons, etc.
- Création aléatoire de personnages : réalisée à l'aide d'un bouton aléatoire dans la fenêtre de personnalisation.
- Création graphique de personnages: personnalisables via une fenêtre dédiée avec choix tête/corps/bas.
- Création aléatoire de plateau: même principe que pour le personnage
- Création graphique de la carte : via une interface de sélection du fond et de la difficulté (qui détermine le nombre d'îles).
- Fenêtre de jeu de navigation : avec carte interactive, boss visibles, déplacements.

- Contrôle du personnage via interaction avancée : déplacement libre via drag & drop, avec animation fluide et détection des collisions avec les îles.
- Retours visuels clairs : PV boss, timer, niveau, vies, halo autour des îles atteignables, effets visuels en combat.

II. Composants applicatifs

- Représentation interne du plateau : structure GameMap avec îles, boss, états de visibilité/exploration.
- Génération procédurale du plateau : positionnement aléatoire des îles tout en évitant les collisions.
- Déplacement géré par interactions : le modèle est mis à jour à chaque mouvement du bateau.
- Conditions de victoire/défaite intégrées :Victoire : boss final vaincu (île One Piece)/Défaite : 0 cœur restant.
- Sauvegarde de la carte (layout plateau) : partiellement implémentée dans la structure interne.
- Sauvegarde complète du jeu (reprise plus tard) : non réalisée (fonctionnalité prévue mais non finalisée car certaines de nos classes ne le permet pas).
- Comptage du temps / nombre d'actions : non intégré car non essentiel au gameplay.
- Sauvegarde de personnages pour réutilisation : non implémenté.

Reflection sur le déroulement du projet

(Paolo) J'ai apprécié ce projet car on a pu mettre en pratique tout ce que l'on a pu voir cette année dans les différentes matières. De plus, je trouve que l'accompagnement dans les séances et avec le cahier de suivi est une super idée, ça permet de se situer dans le projet, d'avoir une meilleure vision des attentes et surtout de se sentir libre tout en étant encadré. Concernant notre travail je suis content car on a réussi à avoir:

- Collaboration fluide, MVC bien structuré
- Visuels travaillés, très bon rendu final
- Comportement dynamique réaliste (boss, halo, combat,...)

J'aimerais également évoquer quelques tâches que j'aurais voulu pouvoir accomplir, si j'en avais eu le temps ou les compétences:

- mouvement du bateau plus réaliste avec des contournement des îles, j'ai essayé mais le rendu était catastrophique
- ajouter des animations pour la phase de combat, cependant créer des animations prend beaucoup de temps
- créer un système de sauvegarde mais à cause de ma classe GameState qui contient des variables IntegerProperty, je n'ai pas réussi à sauvegarder ces informations, donc j'ai abandonné l'idée
- réussir à pouvoir afficher le jeu en plein écran en gardant toutes les proportions (positions des îles,...)

(Pablo) Pour ma part, j'ai trouvé ce projet super intéressant et très plaisant à faire d'abord car c'est vraiment finalité et une suite logique par rapport à nos cours et TP précédents, il nous permet de mettre en pratique de façon concrète tout ce que nous avons appris ce qui est vraiment plaisant.

Je me suis donc occupé du menu du jeu avec la création des différents design des personnages que nous avons nous même créer à partir d'image sur internet. On a choisi de créer nos fenêtres avec des fichiers fxml pour pouvoir utiliser SceneBuilder et séparer cette partie de notre code pour plus de clarté. Nous avons utilisé un fichier .css pour augmenter la taille des boutons quand la souris passe dessus.

La plus grande difficulté que j'ai rencontré était la sauvegarde du personnage choisi dans la personnalisation pour qu'il puisse être affiché et réutilisé ensuite pour le jeu, en effet, un bouton sur une fenêtre devait actionner un événement sur une autre fenêtre ce qui m'a obligé à lier des contrôleurs entre eux ce qui n'était pas simple. Une autre partie compliquée était le fait de lier les radio boutons (difficulté) à l'image au dessus et à la difficulté choisie.

Les images qui représentent les îles viennent d'un générateur en ligne (<https://www.redblobgames.com/maps/mapgen2/>), et pour ce qui est des boutons, ils ont été conçus "à la main" sur Canva (<https://www.canva.com/>).

En terme de librairie pure et dure on a utilisé:

- 1) JavaFX (javafx-controls, javafx-fxml, javafx-graphics, etc.)
 - afficher les vues (carte, combats, menu...),
 - gérer les images (bateau, personnages, boss, etc.),
 - créer les animations (déplacement du bateau, halo des îles, effets de clic...),
 - gérer le layout responsive et les interactions (drag & drop du bateau, clics sur les boss...).
- 2) JavaFX Timeline & Animation API: tangage du bateau, timer du combat, translation animée, etc...
- 3) FXML (via FXMLLoader): pour organiser proprement toutes les vues de notre application (GameView, CombatView, EndGameView...).

Pour ce qui est de l'utilisation de l'IA, elle a clairement joué un rôle important dans ce projet, mais surtout en tant que support. On l'a souvent utilisée pour nous aider à comprendre certaines erreurs, ou pour débugger quand on était vraiment bloqués.

On a aussi essayé de l'utiliser pour gagner un peu de temps en lui demandant de générer quelques bouts de code, mais au final, comme notre projet était assez complexe, original, et surtout très spécifique, les réponses étaient la plupart du temps à côté de la plaque.