Projet SMA : Système Multi-Agents

Antoine Carlier - Loïc Decoster - Thibault Desprez - Pierre Marez - Joffrey Paprocki

Encadrant

Samuel DELEPLANQUE

Novembre 2020

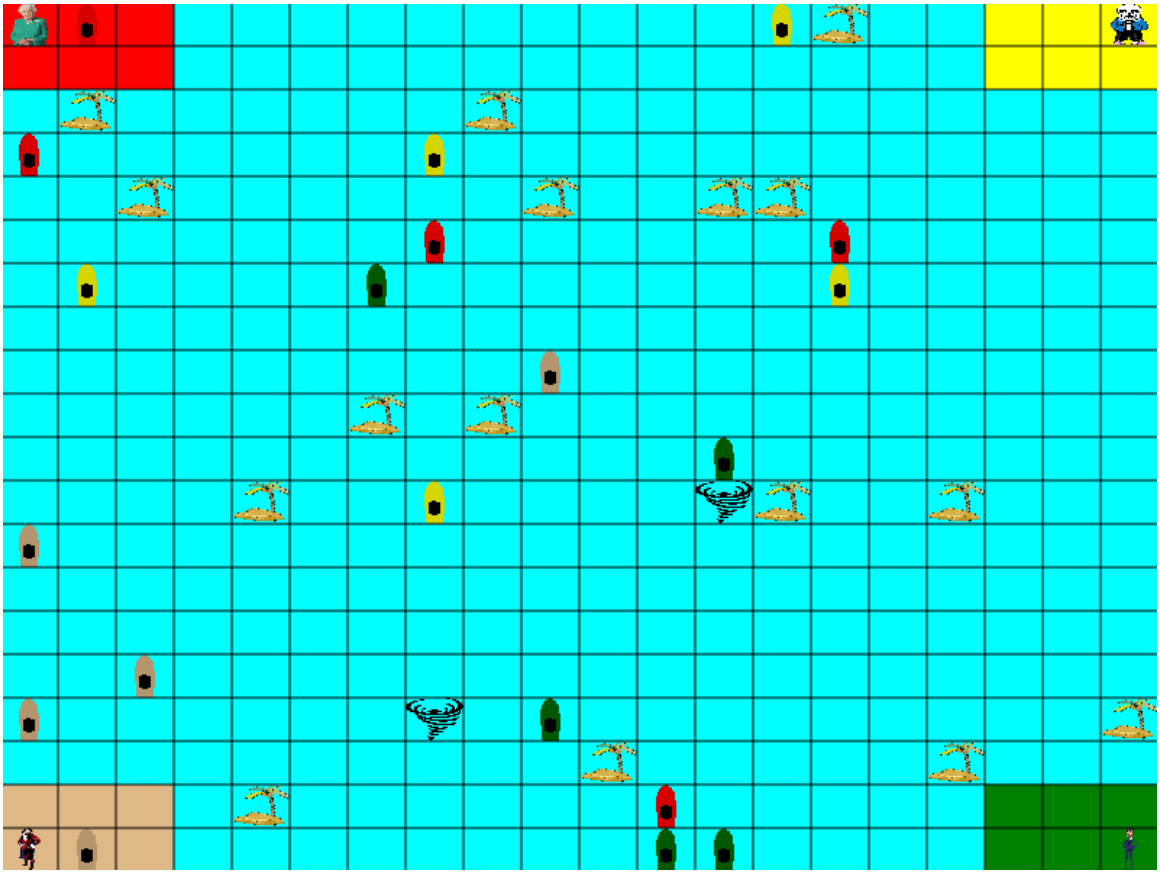

FIGURE 1 – Présentation de la simulation

1 Présentation du projet

Pour lancer l'application nous utilisons **Gradle** il faut exécuter la commande :
`./gradlew run` sous linux et `gradlew.bat run` sous windows.
Il faut avoir installer au préalable un JDK 8 avec javafx.

L'interface est séparée en 3 zones : L'interface utilisateur à gauche qui permet de contrôler le programme. La zone centrale qui représente la Carte et les Pions et la zone à droite qui affiche les Pions actuels et les ressources qu'ils possèdent.

Lorsque l'on clique sur le bouton *Start*, la simulation se lance selon le timer et la carte actuelle. Le timer décide du temps entre chaque étape de la simulation, au plus petit, au plus la simulation ira vite.

Le bouton *Pause* permet d'arrêter la simulation.

Le bouton *Step* permet de faire avancer la simulation d'une étape manuellement.

La seed permet d'initialiser le générateur de nombres pseudo-aléatoire, générer une simulation avec la même seed donnera le même résultat.

Pour appliquer la seed, il faut utiliser le bouton *Reset* qui initialise la carte actuelle selon la seed entrée dans le champ prévu à cet effet.

Appuyer sur *Random* remplira le champ de la seed avec un nombre aléatoire sans pour autant initialiser la carte.

2 Rôles des membres

Comme nous étions cinq sur le projet, nous avons dû nous répartir les tâches afin que chacun puisse contribuer au projet.

Pour réaliser ce projet nous avons réparti les rôles de les pôles suivants :
— Un pôle *vue* qui gérait les vues du programme à l'aide de JavaFX.
— Un pôle *modèle* qui a développé la carte de la simulation et ses composants.
— Un pôle *contrôleur* qui s'est occupé d'interfacer la vue avec le modèle.

En fin de projet, quand notre simulation était fonctionnelle, nous avons changé nos pôles de travail en deux autres distincts :
— Un pôle s'occupant de régler les bugs et autres anomalies du programme.
— Un pôle s'occupant de mettre en forme le rapport et les diagrammes.

3 Diagramme de classes UML complet (attributs + méthodes et leur visibilité)

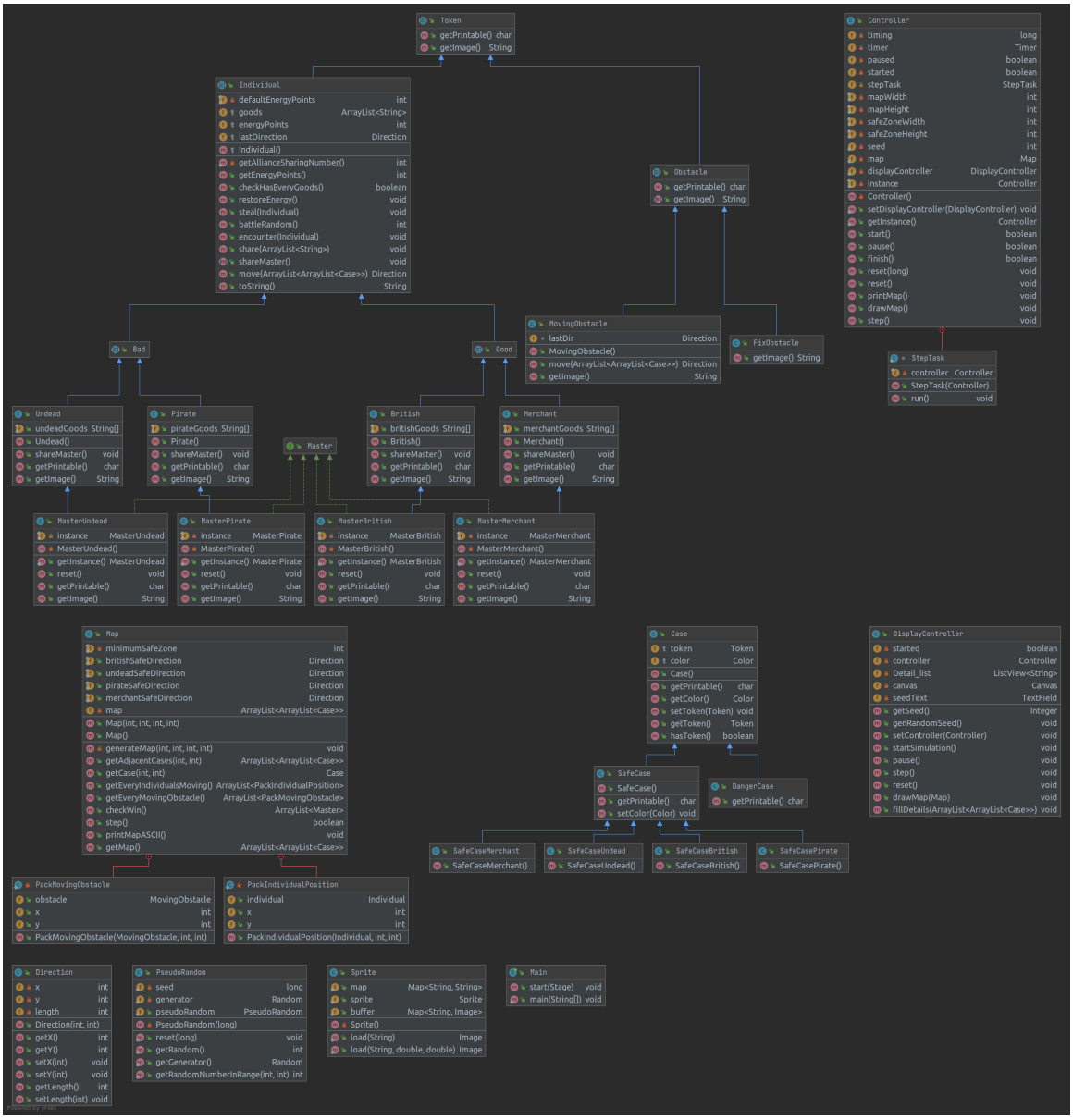

FIGURE 2 – Diagramme de classes (générée par IntelliJ)

4 Diagramme d'états-transitions

Pour une meilleure visibilité et compréhension du projet, nous avons réalisé quatre diagrammes d'états-transitions. Ceux-ci commencent lorsque la fenêtre de jeu est affichée. Les entités sont déjà créées et leurs déplacements à l'écran ne seront pas mis dans les diagrammes afin de les alléger.

Le premier représente la manipulation de la fenêtre de jeu par l'utilisateur. Il comporte un état composite nommé "Simulation en cours" et est lié à un état historique profond permettant ainsi de stopper et reprendre la simulation dans n'importe lequel de ses sous-états. Une sortie globale est présente sous l'état "Simulation" pour indiquer la fin du programme.

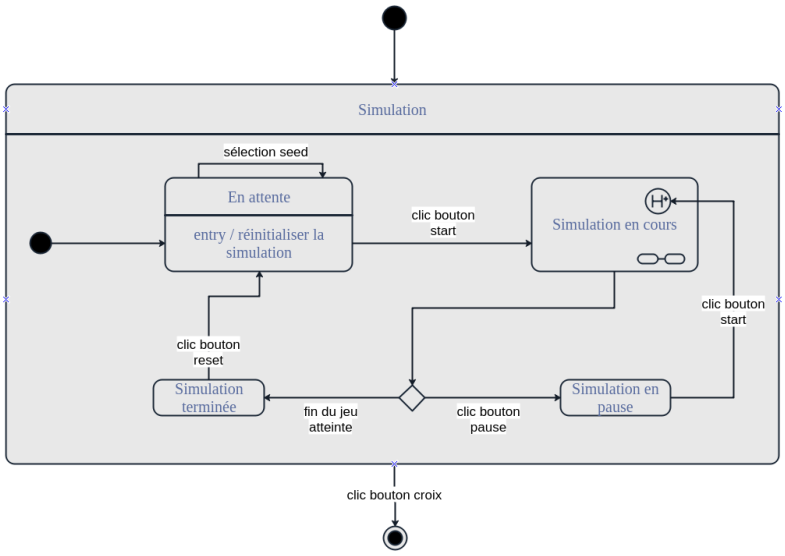

FIGURE 3 – Diagramme d'états-transitions

Le deuxième diagramme comporte lui aussi un état composite et fait la distinction entre une étape ("step") réalisée automatiquement au bout de X temps et le passage manuel à l'étape suivante de la simulation en cliquant sur le bouton dédié.

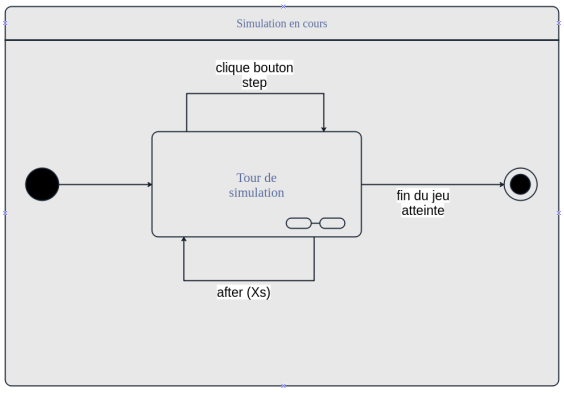

FIGURE 4 – Diagramme d'états-transitions

Le troisième diagramme représente quant à lui la mécanique de la simulation avec le déplacement et les interactions des tornades et des bateaux.

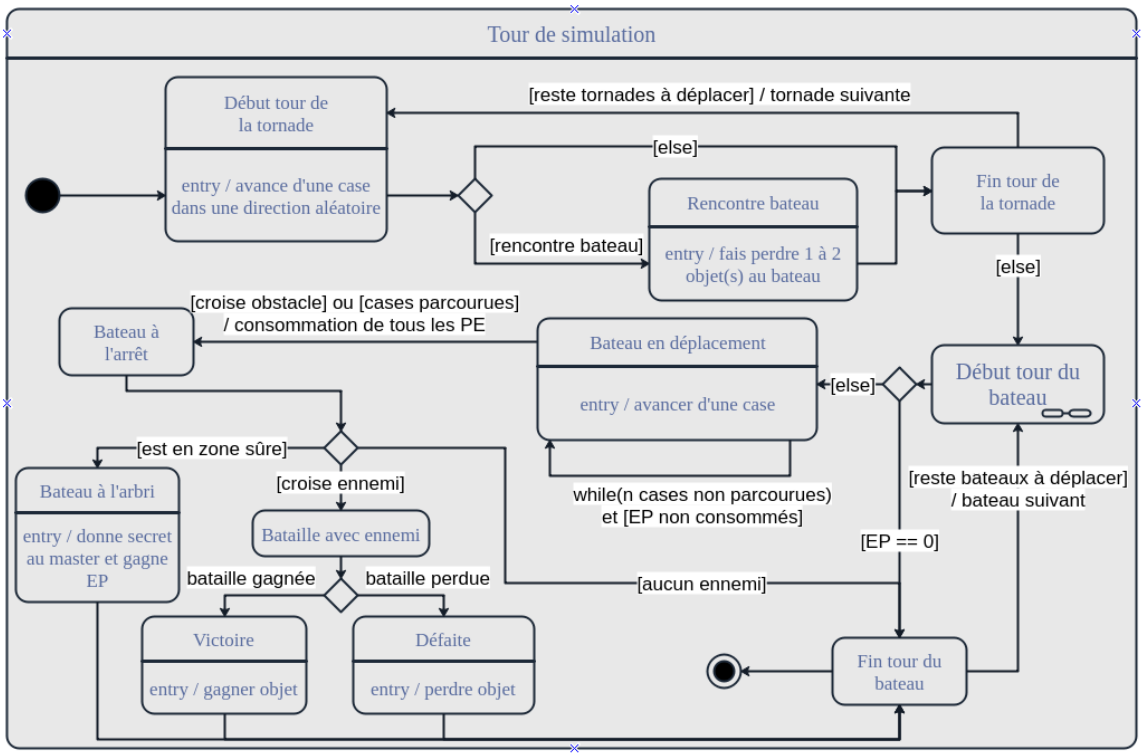

FIGURE 5 – Diagramme d'états-transitions

Le dernier diagramme indique le choix de la direction prise par les bateaux en fonction de leur nombre d'EP.

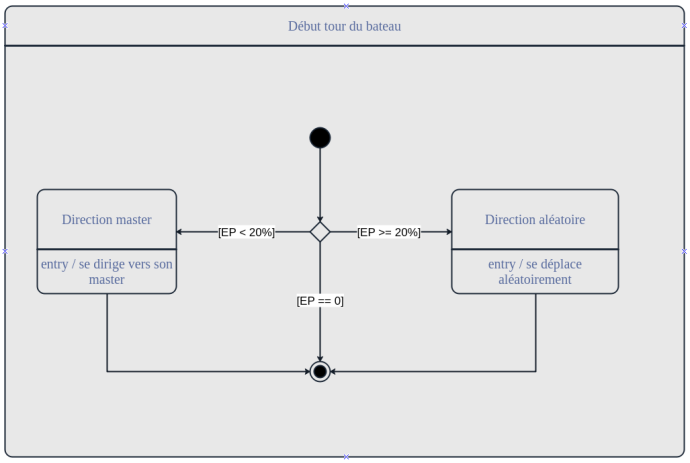

FIGURE 6 – Diagramme d'états-transitions

5 Diagramme d'activités

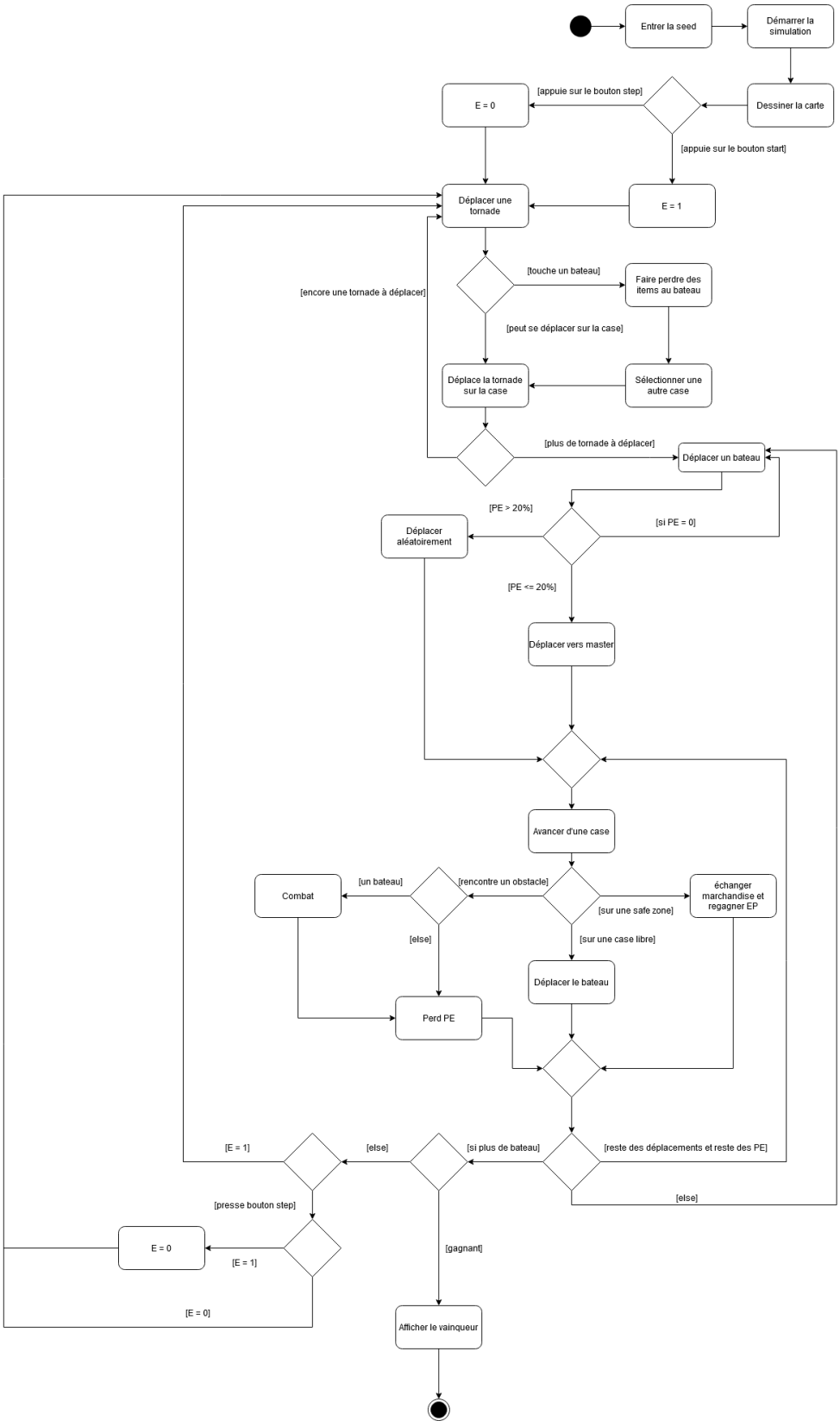

FIGURE 7 – Diagramme d'activité