# Conception d'architectures distribuées

Conception - Bataille navale

**Antoine Courtil** 

Simon Hajek

Guillaume Petit





# **SOMMAIRE**

Introduction	3
Choix de représentation du modèle	3
Comportement général et changement d'états de l'application	3
Diagrammes de classes	3
Déroulement d'un tour de jeux	6
Sauvegarde de la partie	8
Diagramme d'état de l'application	8
Diagramme de séquence - Comportement général	9
Vues proposées pour l'application	10

#### Introduction

Ce rapport présente la partie conception du projet bataille navale. Il contient majoritairement des diagrammes UML représentants la structure du programme et quelques exemple de comportement ainsi que des explications sur les choix de conception.

#### Choix de représentation du modèle

Pour représenter les plateaux, aucun tableau en deux dimensions n'est utilisé. Nous avons choisi d'intégrer les positions dans les différentes classes :

- La position d'un bateau dans la classe Bateau
- La position des tirs réussis dans la classe Bateau
- La position des tirs échoués dans la classe Player

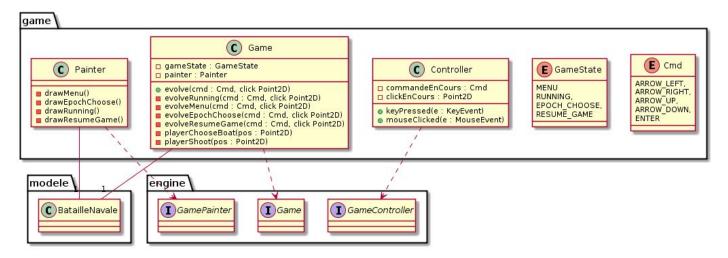
Nous avons choisi cette implémentation afin de ne pas à avoir dupliquer les différentes positions. L'affichage qui découle de cette implémentation ne sera simplement qu'un parcours des différents éléments afin de les ajouter visuellement.

#### Comportement général et changement d'états de l'application

La boucle principale sera la boucle Evolve de la classe Game qui traitera selon si c'est au joueur ou à l'IA de jouer les cliques et les commandes envoyées. Cette même fonction appellera la sous fonction evolve correspondante à l'état du jeu pour réaliser les traitements nécessaires.

#### Diagrammes de classes

Diagramme de classe du package game



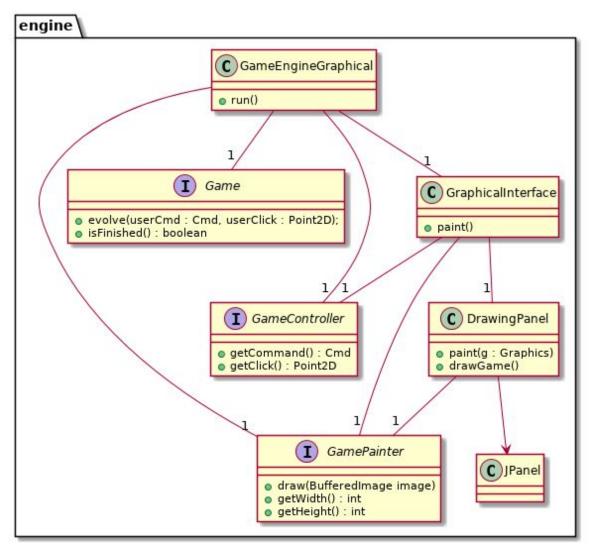
boatFactory C Bateau shoot(pos: Point2D)
 receiveShoot(pos: Point2D, damage: int): boolean
 chooseBoat(pos: Point2D): boolean
 getBoatIndexFromPos(pos: Point2D): int
 getCurrentBoat(): Bateau
 addFailedShoot(pos: Point2D) Tirs échoués playerShoot(pos: Point2D)
 loadFromFile(filename: String): BatailleNavale
 saveToFile(filename: String) getStrategie() : StrategiesetStrategie(strategie : Strategie) C Point2D C Player □ currentBoatIndex : int □ hasChoosedBoat : boolean □ strategie : Strategie BatailleNavale Humain uvidth: int height: int turnPlayer: boolean epoch: int C Serializable 0 □ lastShootHasTouched: boolean o generateShoot(): Point2D o generateShoot(): Point2D C strategieAlea 0..1 Strategie □ lastShootHasTouched: boolean generateShoot(): Point2D C strategieCroix modele

Diagramme de classe du package modele

C XIXBateau2Cases int getHP();int getPrecision();int getDegat();int getPortee(); C XIXBateau3Cases int getHP();int getPrecision();int getDegat();int getPortee(); Bateau getBateau5Cases();
 Bateau getBateau4Cases();
 Bateau getBateau3Cases(); (C) XIXBateauFactory C XIXBateau4Cases int getHP();int getPrecision();int getDegat();int getPortee(); int getSize()Point2D getPosition(); (A) Bateau2Cases AbstractBateauFactory getFactoryByEpoque(int epoque); C XIXBateau5Cases int getHP();int getPrecision();int getDegat();int getPortee(); int getSize()Point2D getPosition(); (A) Bateau3Cases int getHP();
int getSize();
int getPecpat();
int getPortee();
int getPortee();
int getPortee();
int getPortee();
int getPortee();
int getPorte();
int getPoint2D> getPointsTouches(); boatXIX (A) AbstractBateauFactory int HP in the float floa Bateau 3 C XVIIIBateau2Cases int getSize()Point2D getPosition(); int getHP();int getPrecision();int getDegat();int getPortee(); (A) Bateau4Cases Bateau getBateau5Cases();
 Bateau getBateau4Cases();
 Bateau getBateau3Cases();
 Bateau getBateau2Cases(); C XVIIIBateau3Cases C XVIIIBateauFactory int getHP();int getPrecision();int getDegat();int getPortee(); int getSize()Point2D getPosition(); A Bateau5Cases abstractBoat | C XVIIIBateau4Cases int getHP();int getPrecision();int getDegat();int getPortee(); C XVIIIBateau5Cases int getHP();int getPrecision();int getDegat();int getPortee(); boatXVIII boatFactory

Diagramme de Classe du package Factory

# Diagramme de classe du package engine



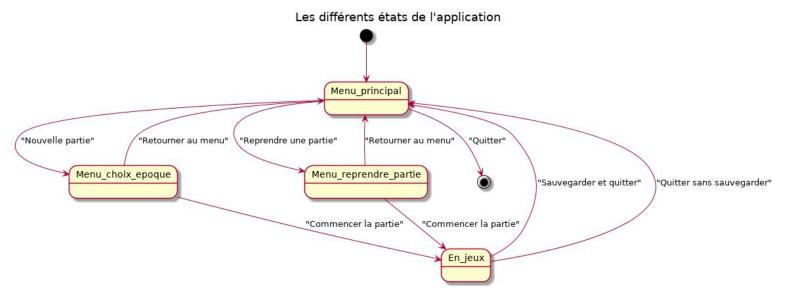
## Déroulement d'un tour de jeu

Pour récupérer la position cliquée, le Contrôleur hérite de *MouseEventListener* afin de récupérer les positions cliquées sur la fenêtre pour par la suite vérifier la cohérence des cliques.

#### Sauvegarde de la partie

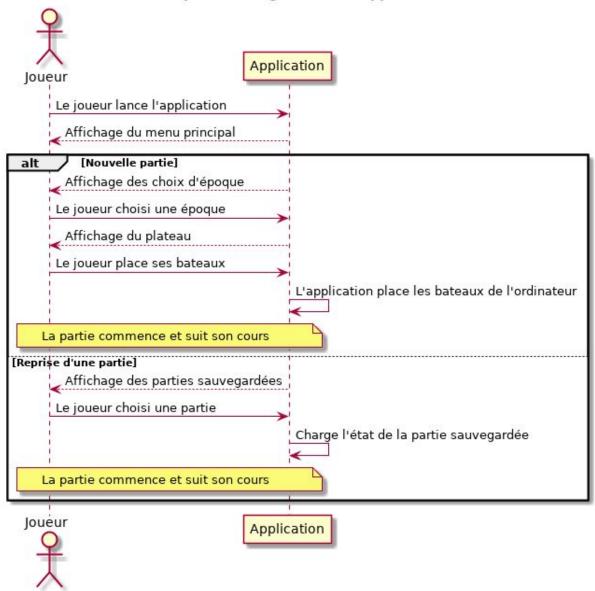
Pour sauvegarder la partie, nous avons décidé de simplement 'serializer' la classe BatailleNavale. Les classes BatailleNavale, Player et Bateau implémente donc l'interface Serializable et les méthodes loadFromFile et saveToFile sont utilisées respectivement pour charger et sauvegarder la partie depuis le menu principal.

#### Diagramme d'état de l'application



## Diagramme de séquence - Comportement général

#### Comportement général de l'application



# Vues proposées pour l'application

