Guillaume ROTH
Paul DECANTER
Vincent FAGNON

Conception Architectures Distribuées

Master 1

Bataille navale

Dossier d'Analyse-Conception

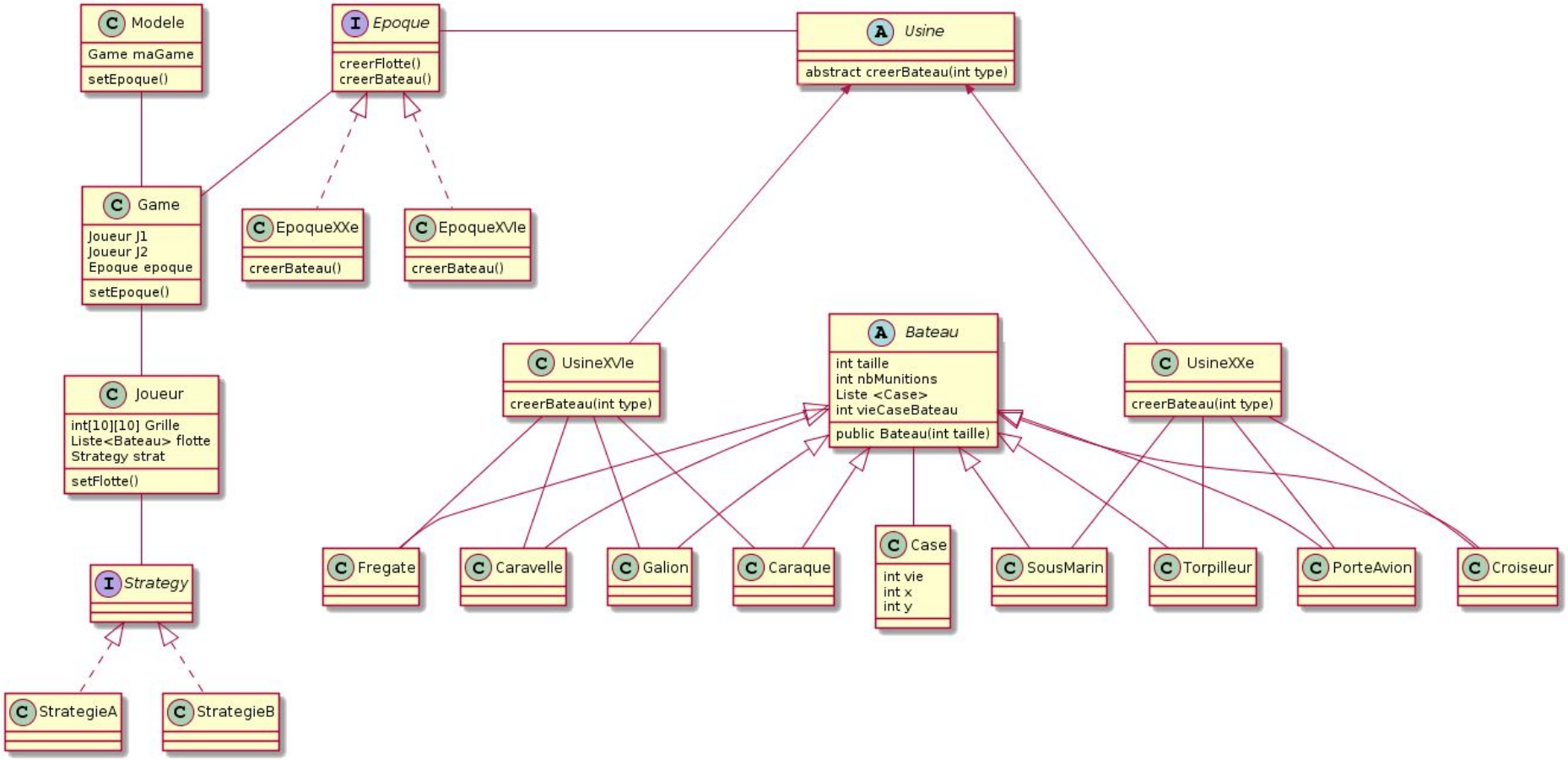
Mars 2018

Table des Matières

Diagramme de Classes **2**
Initialisation du moteur de jeu 2
Diagramme Sprite Factory 3
Diagramme de Séquence **4**
Initialisation de la Flotte des Joueurs après choix de l'époque 4
Choix de l'emplacement d'un bateau 5
Diagramme d'Activité **6**
Séquence de Tir 6
Annexes **7**
Exemple d'implémentation expliquant l'initialisation de la flotte en fonction de l'époque choisie par l'utilisateur 7

Diagramme de Classes

Initialisation du moteur de jeu



- Nous proposons un Modèle qui possède une instance de Game et une fonction setEpoque() qui définit l'époque de Game au lancement du programme.
- Game instancie deux Joueur et une Époque (par défaut dans le constructeur)
- Chaque joueur possède une grille et une liste de Bateau.
- Strategy correspond aux différentes stratégies de jeu pour l'IA)
- Le trio Epoque, EpoqueXXe et EpoqueXVIe constitue un desgin pattern de type Strategie.
- Epoque est ensuite reliée à un design pattern de type Factory permettant de générer une liste de bateaux en fonction de l'Epoque sélectionnée par l'utilisateur.

Diagramme Sprite Factory

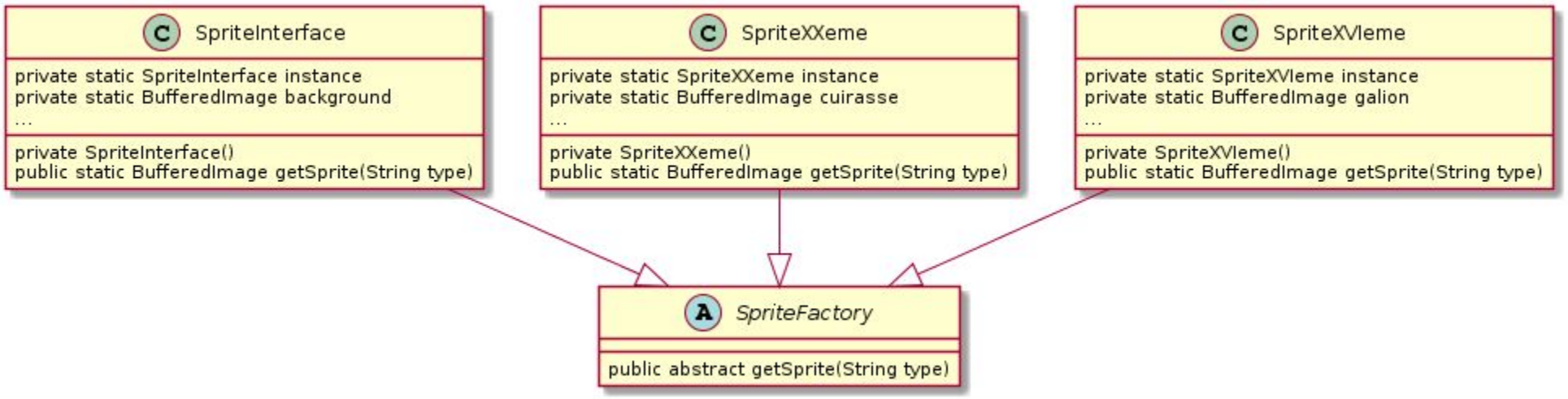
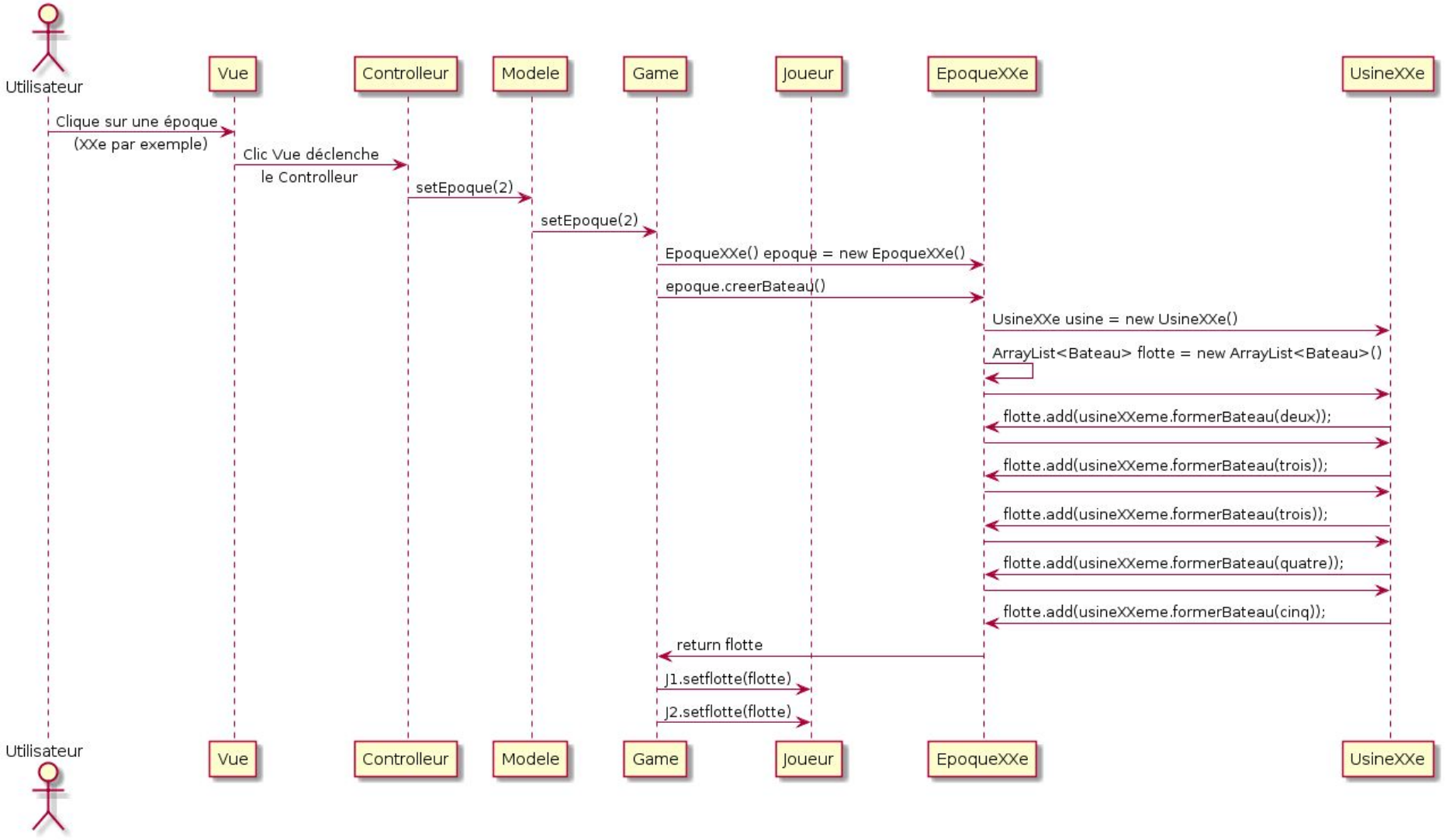


Diagramme de Séquence

Initialisation de la Flotte des Joueurs après choix de l'époque



Choix de l’emplacement d’un bateau

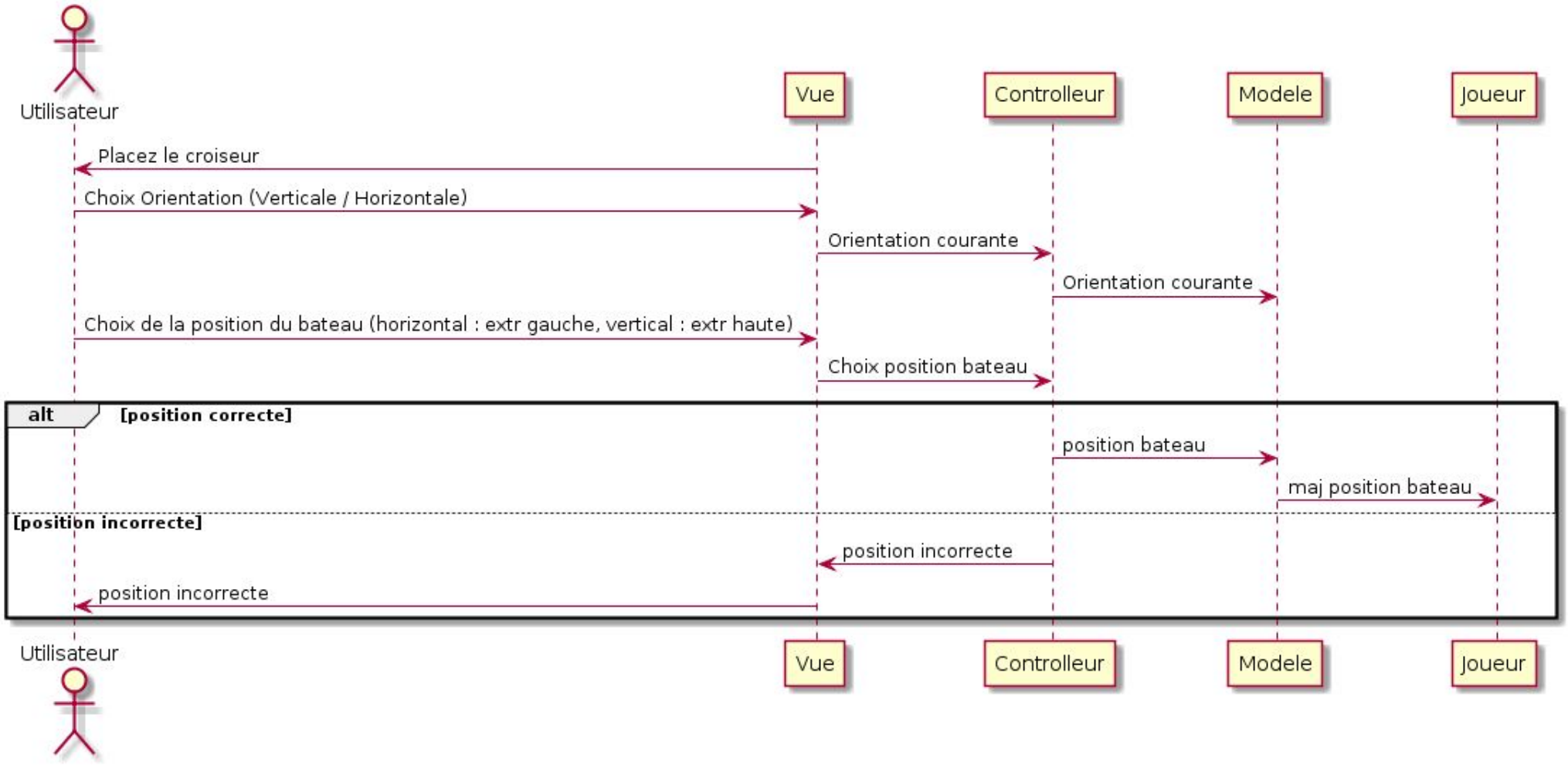
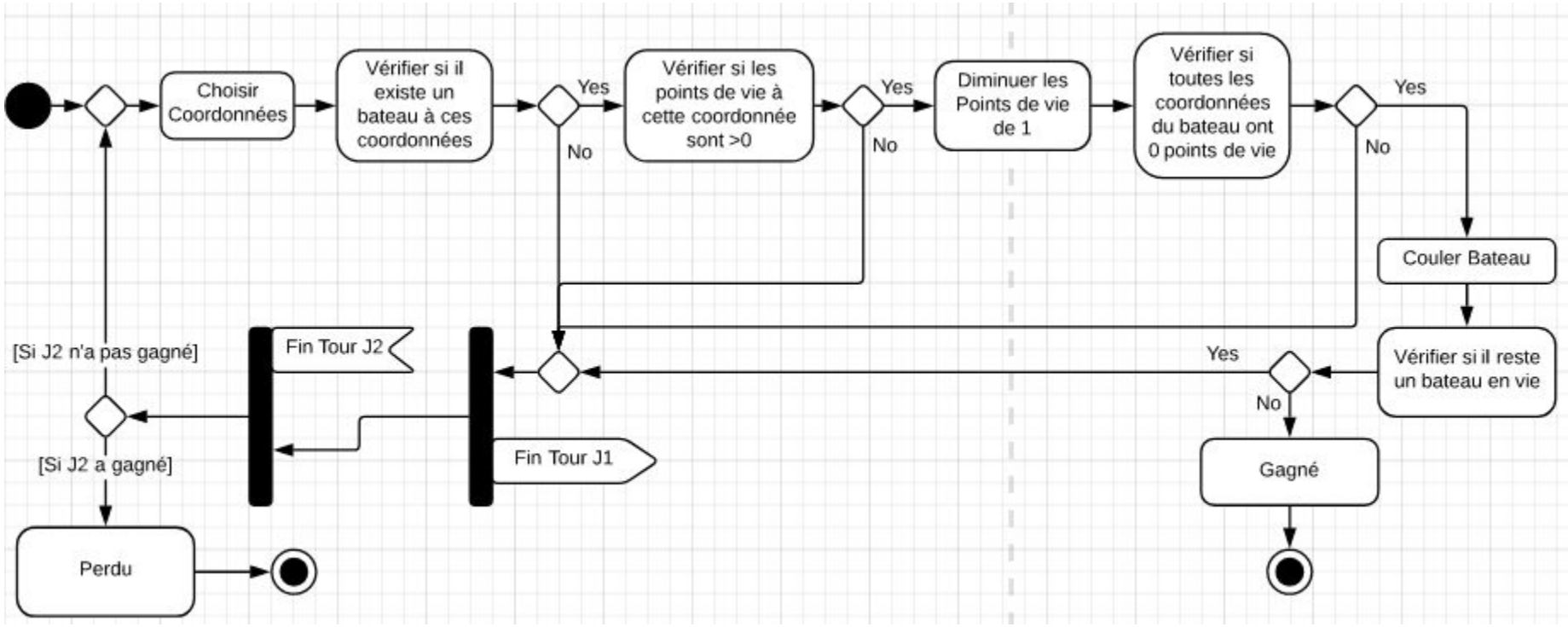


Diagramme d’Activité

Séquence de Tir



Annexes

Exemple d'implémentation expliquant l'initialisation de la flotte en fonction de l'époque choisie par l'utilisateur

```
public class Game {
        ...
        public void setEpoque(int choix){
                Epoque epoque;
                switch(choix){
                        case 20 :  epoque = new XXeme(); break;
                        case 16 : epoque = new XVIeme(); break;
                        }
                ArrayList<Bateau> flotte =  epoque.creerFlotte();
                joueur.setFlotte(flotte);
                …
                }
        ….
        }
```

```
public abstract class Usine{
        ...
        public Bateau formerBateau(String type){
                return this.creerBateau(type);
                }
        public abstract creerBateau(String type);

        }
```

```
public class EpoqueXXeme implements Epoque {
        ...
        ArrayList<Bateau> creerFlotte(){
                ArrayList<Bateau> flotte = new ArrayList<Bateau>();
                Usine usineXXeme = new usineXXeme();
                flotte.add(usineXXeme.formerBateau(deux));
                flotte.add(usineXXeme.formerBateau(trois));
                flotte.add(usineXXeme.formerBateau(trois));
                flotte.add(usineXXeme.formerBateau(quatre));
                flotte.add(usineXXeme.formerBateau(cinq));
                return flotte;
                }
        …
        }
```

```
public class UsineXXeme extends Usine{
        ...
        public Bateau creerBateau(String type){
                Bateau bateau = null;
                switch(type){
                        case deux : bateau = new torpilleur();break;
                        case trois : bateau = new ssmarin();break;
                        case quatre : bateau = new croiseur();break;
                        case cinq : bateau = new porteavion();break;
                        }
                return bateau;
                }
        }
```