AZUL

UN JEU PLATEAU

L3 MIAGE

PROJET TUTORÉ

Canavaggio - Choubrac -
GandoDiallo - Mohdad -
Quatela - Vasseur

SOMMAIRE

I- Fonctionnalités

LISTE DES PRINCIPALES FONCTIONNALITÉS

Support de toutes les règles du jeu de base

Support de l'extension mur en couleur

Statistiques sur 500 parties

Support de l'extension mur gris

Support de l'extension mur variante

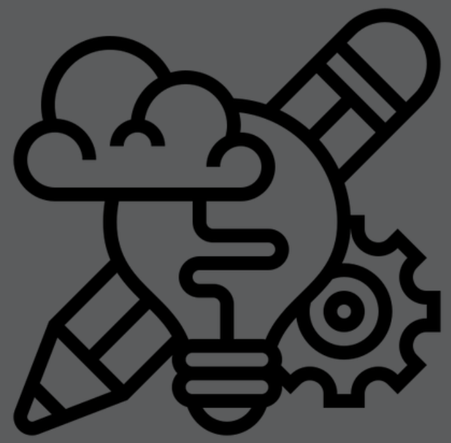
LES AXES

Configuration du jeu

Représentation du plateau

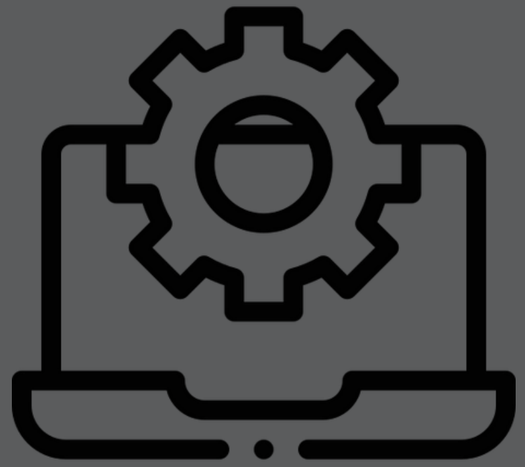
Conception de l'Intelligence Artificiel

CHOIX DE CONCEPTION



- MoteurDeJeu
  Configuration de la partie
- Manager
  Gestion de la partie
- Joueur
  Attribut du joueur
- Plateau
  Eléments du jeu
- IA
  Types IA

Configuration du jeu



Logique de jeu
Commandes

- mvn compiler:compile
- mvn exec:java@1 -Darg1=value -Darg2=value
- mvn exec:java@1 -DnombreDeJoueur=[1,...,n]]

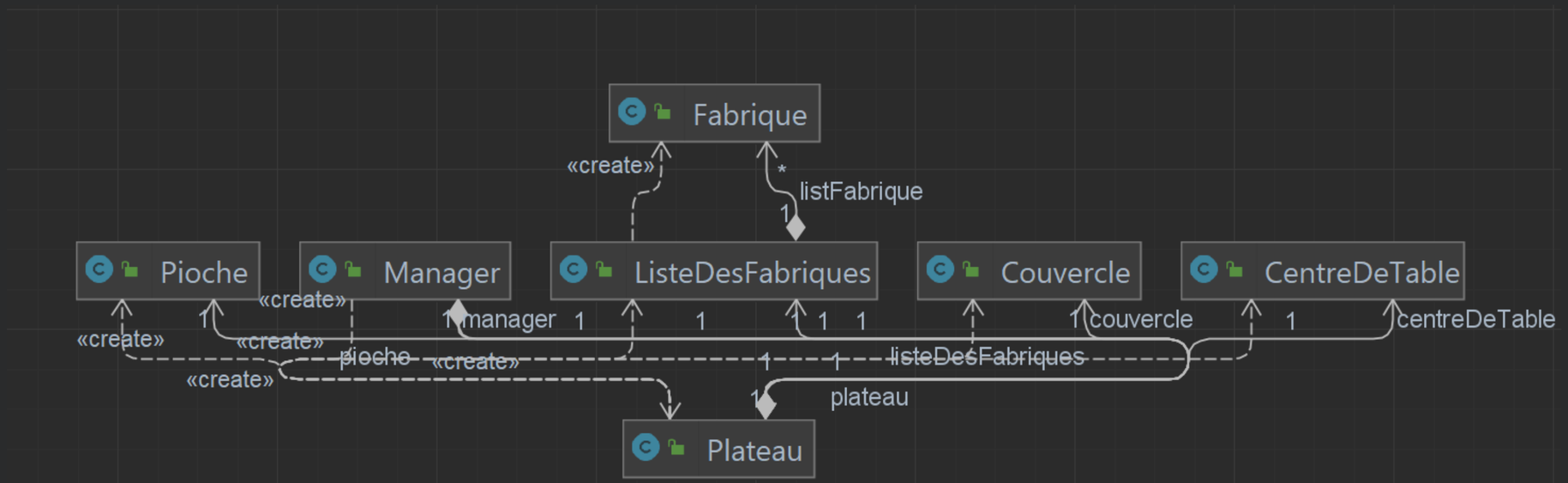
  -Dthread=[1...n]

  -DnombreDePartie=[1...n]

  -Dmur=gris ou variant

Logique de Jeu

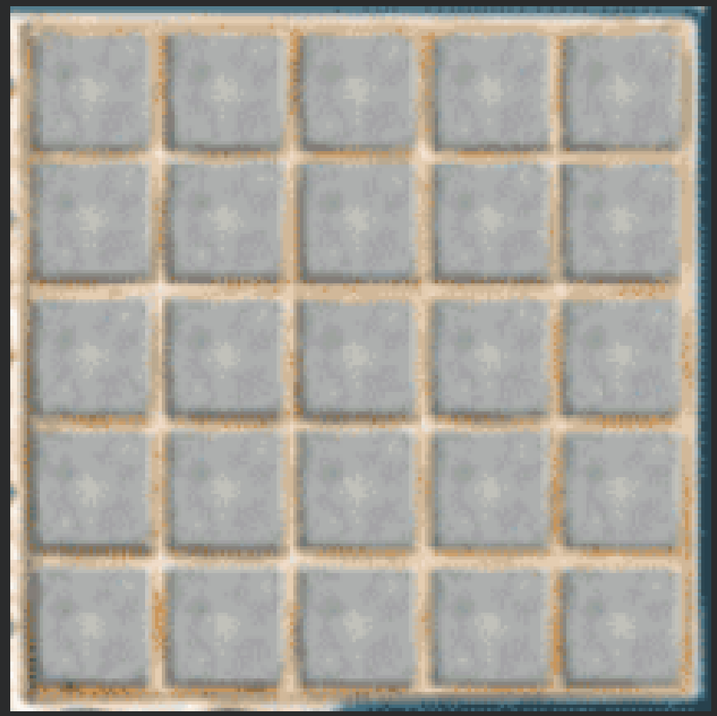
Logique de jeu
Plateau

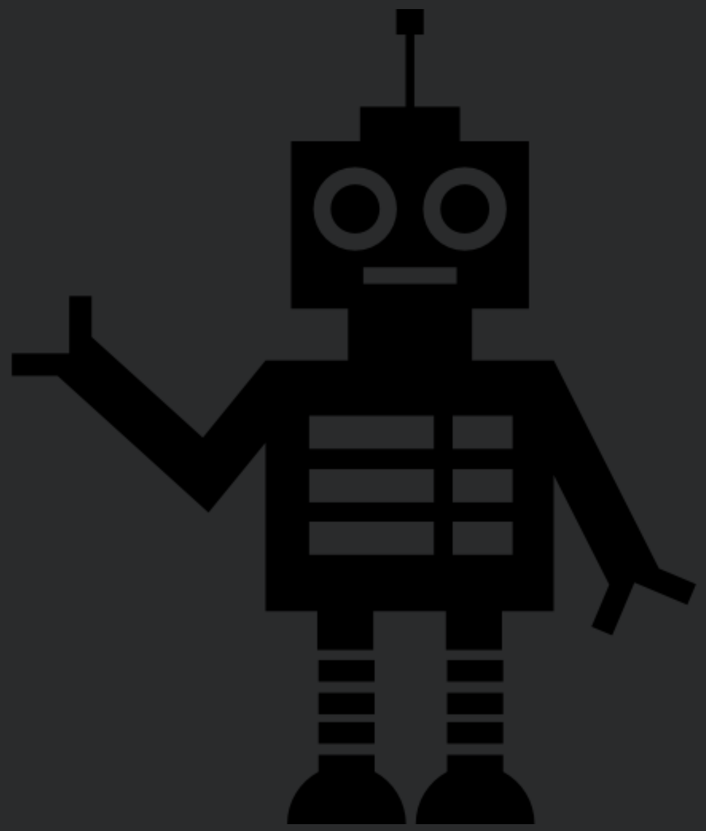


Choix de la structure des murs

```
public abstract class Mur {
```

```
public enum TypeMur {MUR_COULEUR,MUR_GRIS,MUR_VARIANTE}
```





L'Intelligence Artificielle

L'Intelligence artificielle
Robot aléatoire

```
Tuiles dans les fabriques :
Fabrique n°1 : [BLANC, BLANC, JAUNE, BLEU]
Fabrique n°2 : [NOIR, JAUNE, ROUGE, BLANC]
Fabrique n°3 : [NOIR, NOIR, BLANC, BLEU]
Fabrique n°4 : [BLANC, ROUGE, JAUNE, BLEU]
Fabrique n°5 : [ROUGE, BLANC, BLANC, JAUNE]
Fabrique n°6 : [BLANC, BLANC, JAUNE, BLANC]
Fabrique n°7 : [ROUGE, JAUNE, JAUNE, BLEU]
Tuiles au centre :
[JETON ]
```

```
- $> Joueur1 a pioché 1 tuile de couleur JAUNE depuis la fabrique numero 2 et pose
 la tuile sur la ligne de motif numero 2
                        [VIDE]
                    [JAUNE, VIDE]
                [VIDE, VIDE, VIDE]
            [VIDE, VIDE, VIDE, VIDE]
        [VIDE, VIDE, VIDE, VIDE, VIDE]
```

Intelligence artificielle
Robot plus de couleur
(niveau 2)

```
Tuiles dans les fabriques :
Fabrique n°1 : [BLANC, BLANC, JAUNE, BLEU]
Fabrique n°2 : [NOIR, JAUNE, ROUGE, BLANC]
Fabrique n°3 : [NOIR, NOIR, BLANC, BLEU]
Fabrique n°4 : [BLANC, ROUGE, JAUNE, BLEU]
Fabrique n°5 : [ROUGE, BLANC, BLANC, JAUNE]
Fabrique n°6 : [BLANC, BLANC, JAUNE, BLANC]
Fabrique n°7 : [ROUGE, JAUNE, JAUNE, BLEU]
Tuiles au centre :
[JETON ]
```

```
- $> Joueur2 a pioché 3 tuiles de couleur BLANC depuis la fabrique numero 6 et pose
 les tuiles sur la ligne de motif numero 3
                    [VIDE]
                [VIDE, VIDE]
            [BLANC, BLANC, BLANC]
        [VIDE, VIDE, VIDE, VIDE]
    [VIDE, VIDE, VIDE, VIDE, VIDE]
```

Robot "remplis la 1ère ligne" (niveau 3)

```
Tuiles dans les fabriques :
Fabrique n°1 : [BLANC, BLANC, JAUNE, BLEU]
Fabrique n°2 : [NOIR, JAUNE, ROUGE, BLANC]
Fabrique n°3 : [NOIR, NOIR, BLANC, BLEU]
Fabrique n°4 : [BLANC, ROUGE, JAUNE, BLEU]
Fabrique n°5 : [ROUGE, BLANC, BLANC, JAUNE]
Fabrique n°6 : [BLANC, BLANC, JAUNE, BLANC]
Fabrique n°7 : [ROUGE, JAUNE, JAUNE, BLEU]
Tuiles au centre :
[JETON ]
- $> Joueur3 a pioché 1 tuiles de couleur ROUGE depuis le centre et pose la/les tuile
(s)sur la ligne de motif numero 1
                        [ROUGE]
                    [VIDE, VIDE]
                [VIDE, VIDE, VIDE]
            [VIDE, VIDE, VIDE, VIDE]
        [VIDE, VIDE, VIDE, VIDE, VIDE]
```

Organisation des tests

COMMENT ON A FAIT NOS TESTS

Test unitaire avec assertEquals et assertNotEquals

```java
@Test
public void TuileVide () {
    /* Créer une tuile vide et vérifie si elle est bien vide */
        Tuile tuile = new Tuile();
        assertEquals(Couleurs.VIDE,tuile.getCouleur());
}
```

```java
@Test
public void CouleurTuileNonVide() {
    /* Genere une tuile de couleur aleatoire 20 fois et verifie qu'elle n'a pas ete creer en VIDE */
    for (int i = 0; i < 20; i++) {
        Tuile tuile = Utilitaire.getRandom();
        assertNotEquals(Couleurs.VIDE,tuile.getCouleur());
    }
}
```

Test du bon fonctionnement du jeux

```java
public void remplirPlancherUneCouleur () {
    Couvercle couvercle = new Couvercle();
    Tuile[] tuiles = joueur.getPlancherJoueur().remplirPlancher(Couleurs.BLEU, nombreTuile: 3);
    couvercle.remplirCouvercle(tuiles);

    Tuile[] plancher = joueur.getPlancherJoueur().getPlancher();
    for (int i = 0; i < 3; i++) {
        assertEquals(Couleurs.BLEU, plancher[i].getCouleur());
    }
}
```

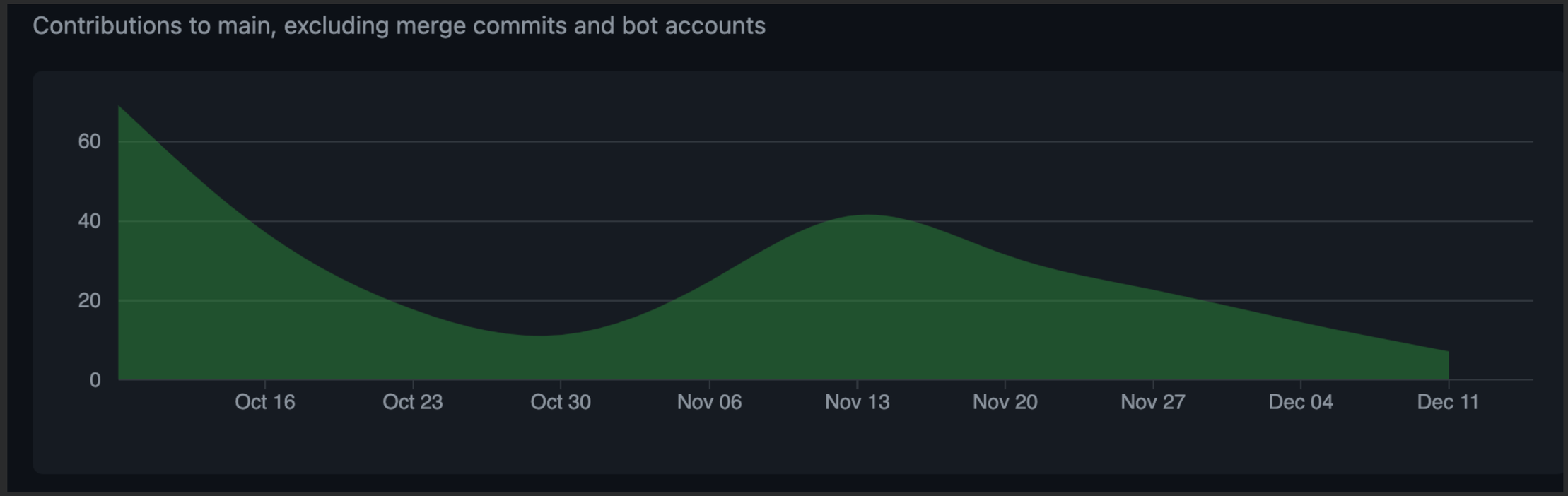
Test du bon fonctionnement des IAs

```java
public void setUp () {
    /* Création des objets permettant le test */
    fabriques = new ListeDesFabriques( nombreJoueur: 1, new Pioche());
    fabrique = new Fabrique(Couleurs.JAUNE, Couleurs.JAUNE, Couleurs.ROUGE, Couleurs.BLEU);
}
```

```java
@Test
/*
  Vérifie que l'IA random pioche bien aléatoirement dans la fabrique
 */
public void testChoixPiocheAleatoireDansUneFabrique () {
    setUp();
    fabriques.addFabrique(fabrique);
    //sélectionne une couleur aléatoire de la fabrique (jaune, rouge ou bleu)
    joueur.getIa().choixCouleurDansFabrique(fabriques, indexDansFabrique: 3, joueur.getMurJoueur());
    Couleurs couleurSelectione = joueur.getIa().getChoixCouleur();
    //vérifie que la couleur fait bien partit de la fabrique
    assertNotEquals(couleurSelectione, Couleurs.BLANC);
    assertNotEquals(couleurSelectione, Couleurs.VIDE);
    assertNotEquals(couleurSelectione, Couleurs.NOIR);
}
```

Gestion du projet

Nombre de commit en fonction du temps



Gestion de projet

Fréquence du code

Contribution au main

Fonctionalité manquante

- Variante 1
- Mise en réseau

DÉMONSTRATION