Hippolyte CAMUS-JUGIEAU
Loric GRENIER
Paul SAGUEZ

Groupe BUT3 - Alternance

Documentation Technique – Puissance 4

I. Introduction

A. Présentation du jeu

Puissance 4 est un jeu de stratégie à deux joueurs. Chaque joueur insère à tour de rôle un jeton dans une grille verticale de 7 colonnes × 6 lignes. Le premier joueur à aligner 4 jetons horizontalement, verticalement ou en diagonale remporte la partie.

B. Objectifs de la documentation technique

- Décrire les choix techniques du projet.
- Présenter l'architecture logicielle.
- Guider les développeurs et mainteneurs pour la compréhension, l'évolution et la correction du code.

II. Architecture du jeu

A. Moteur de jeu

- Gestion de la grille (tableau 2D `int[6][7]`).
- Détection des conditions de victoire et du match nul.

B. Système d'affichage

- Interface graphique en Java et JavaFX.
- Rafraîchissement du plateau après chaque coup.

C. Gestion des ressources

- Pas de ressources multimédia lourdes (images simples ou formes géométriques).
- Couleurs utilisées : Rouge (#FF0000) et Jaune (#FFD700).

Hippolyte CAMUS-JUGIEAU
Loric GRENIER
Paul SAGUEZ

III. Conception du gameplay

A. Mécaniques de jeu

- Deux joueurs humains.
- Alternance automatique des tours.
- Validation des coups (colonne valide, non pleine).

B. Systèmes de contrôle

- Sélection de la colonne via clic sur l'interface (ou saisie clavier si mode console).

C. Interactions joueur-système

- Message en cas d'erreur (colonne pleine, saisie invalide).
- Indication du joueur courant.
- Affichage d'un message en cas de victoire ou de match nul.

IV. Graphismes et Interface

A. Représentation

- Plateau affiché sous forme de grille (7 colonnes × 6 lignes).
- Jetons représentés par des cercles colorés.

B. Animations

- Chute du jeton simulée visuellement (optionnel).

V. Son et Musique

- Son lors du placement d'un jeton

Hippolyte CAMUS-JUGIEAU
Loric GRENIER
Paul SAGUEZ

VI. Optimisation et Performance

A. Gestion des ressources système

- Utilisation mémoire faible (tableau 2D simple).
- Temps de réponse quasi instantané.

B. Optimisation du code

- Boucles efficaces pour la détection de victoire.
- Utilisation de méthodes réutilisables pour limiter la redondance.

C. Tests de performance

- Vérification du temps de détection de victoire (<100ms).

VII. Documentation du Code

A. Structure du code source

- `Main.java` : point d'entrée de l'application.
- `Game.java` : logique du jeu (plateau, coups, victoire).
- `Player.java` : gestion des joueurs.
- `Board.java` : affichage graphique.
- `Controller.java` : gestion des interactions joueur/jeu.

B. Commentaires et documentation interne

- Chaque méthode commentée en français.
- Utilisation de Javadoc pour générer la documentation.

C. Bonnes pratiques

- Respect des conventions de nommage Java.

Hippolyte CAMUS-JUGIEAU
Loric GRENIER
Paul SAGUEZ

X. Déploiement et Maintenance

A. Configuration requise

- Java 17 ou supérieur.
- Système d'exploitation : Windows, Linux ou macOS.

B. Procédure d'installation

- Cloner le dépôt GitHub.
- Compiler avec `javac *.java`.
- Lancer l'application avec `java Main`.

C. Mises à jour et correctifs

- Utilisation de GitHub pour le suivi des issues et des branches.
- Versionnage du code (GitFlow).

XI. Annexes

A. Diagrammes UML

- Diagramme de classes (Game, Player, BoardUI, Controller).
- Diagramme de séquence (tour de jeu).

B. Schéma du plateau

- Représentation 7 × 6 du tableau.

C. Captures d'écran

- Exemple d'écran de menu.
- Exemple d'une partie en cours.
- Exemple de message de victoire.