Contents

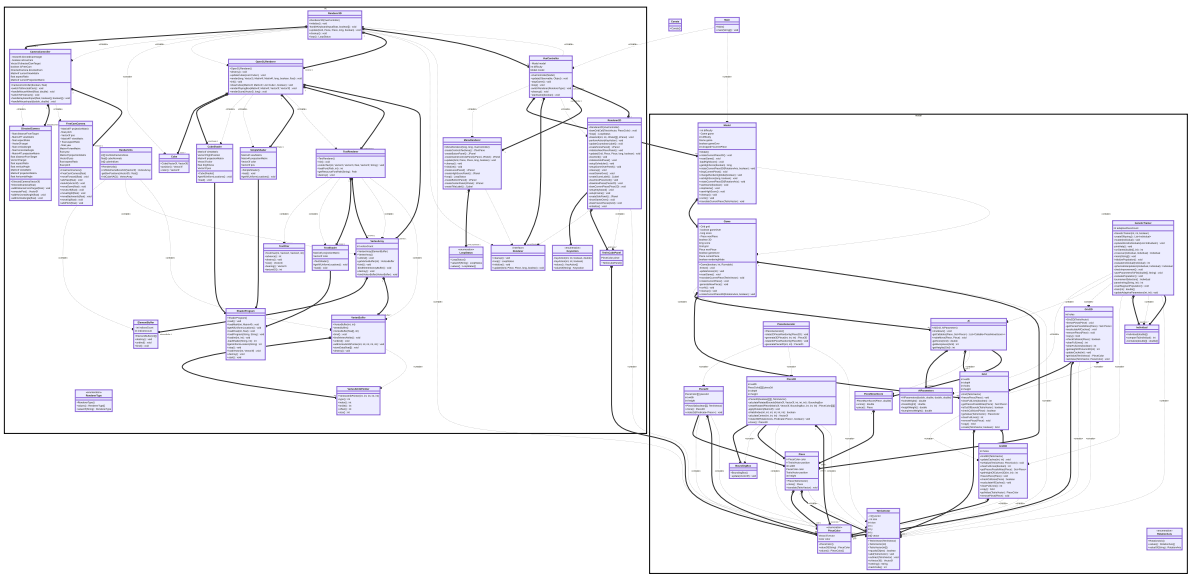
Tetris en 2D et 3D

Ce projet est une implémentation du jeu Tetris en utilisant le modèle MVC (Modèle-Vue-Contrôleur). Il inclut des
fonctionnalités avancées telles que l'intelligence artificielle, le rendu 3D, et bien plus.

Fonctionnalités

- **Jeu Tetris** : Une implémentation complète du jeu Tetris avec les rotations wallkick et T-spin.
- **Architecture MVC** : Utilisation du modèle Modèle-Vue-Contrôleur pour une séparation claire des responsabilités.
- **Abstraction entre 2D et 3D** : Une abstraction maximale entre les rendus 2D et 3D.
- **Rendu 3D bas niveau** : Utilisation d'OpenGL pour un rendu 3D très bas niveau (deux caméras).
- **Intelligence Artificielle** : Une IA qui calcule le meilleur coup à chaque fois avec une récursion de profondeur 2 (
  la prochaine pièce est prise en compte).
- **IA en 3D** : L'IA fonctionne également en mode 3D.
- **Entraînement de l'IA** : Entraînement de l'IA fait maison avec un algorithme génétique.
- **Parallélisation des calculs de l'IA** : Utilisation du nombre de threads le plus efficace pour paralléliser les
  calculs de l'IA (WorkStealingPool).
- **Musique de Tetris** : Intégration de la musique de Tetris.
- **Stockage du meilleur score** : Sauvegarde et affichage du meilleur score.
- **Prévisualisation de la chute de la pièce** : Visualisation de la chute de la pièce avant de la placer.
- **Prévisualisation de la prochaine pièce** : Affichage de la prochaine pièce à venir.

Diagramme de Classes



Namespace `model`

- **Classes Principales :**
  - **`Model`** : Gère la logique du jeu, y compris la rotation des pièces, le démarrage et l'arrêt du jeu, et l'exécution de l'IA.
  - **`Game`** : Représente une instance du jeu, gérant le score, la grille, et les pièces actuelles et suivantes.
  - **`Grid`**, **`Grid2D`**, **`Grid3D`** : Gèrent la grille de jeu, avec des fonctionnalités pour vérifier les collisions, geler les pièces, et effacer les lignes complètes.
  - **`Piece`**, **`Piece2D`**, **`Piece3D`** : Représentent les pièces du jeu, avec des fonctionnalités pour la rotation et le déplacement.
  - **`Ai`** : Implémente l'intelligence artificielle pour jouer au jeu, calculant les meilleurs mouvements possibles.
  - **`GeneticTrainer`** : Utilisé pour entraîner l'IA en utilisant un algorithme génétique.

Namespace `vc` (Vue-Contrôleur)

- **Classes Principales :**
  - **`VueController`** : Gère la vue et le contrôleur, coordonnant les interactions entre le modèle et la vue.

- **`Renderer`**, **`Renderer2D`**, **`Renderer3D`** : Gèrent le rendu du jeu en 2D et 3D.
- **`OpenGLRenderer`** : Utilise OpenGL pour le rendu graphique.
- **`CameraController`** : Gère les caméras pour le rendu 3D, permettant de basculer entre une caméra libre et une caméra dirigée.
- **`TextRenderer`** : Gère le rendu du texte à l'écran.

Instructions de Build

Utilisation de Nix (recommandé)

Pour garantir que vous avez les bonnes versions des logiciels et des dépendances, il est recommandé d'utiliser Nix.

```
nix build
./result/bin/PIE_Swin
```

Ou simplement :

```
nix run
```

Utilisation de Gradle

Vous pouvez également utiliser Gradle seule pour construire et exécuter le projet.

```
gradle fatJar
java -jar build/libs/PIE_Swing-1.0-SNAPSHOT-all.jar
```

Ou simplement :

```
gradle run
```