Contents

1 Tetris en 2D et 3D

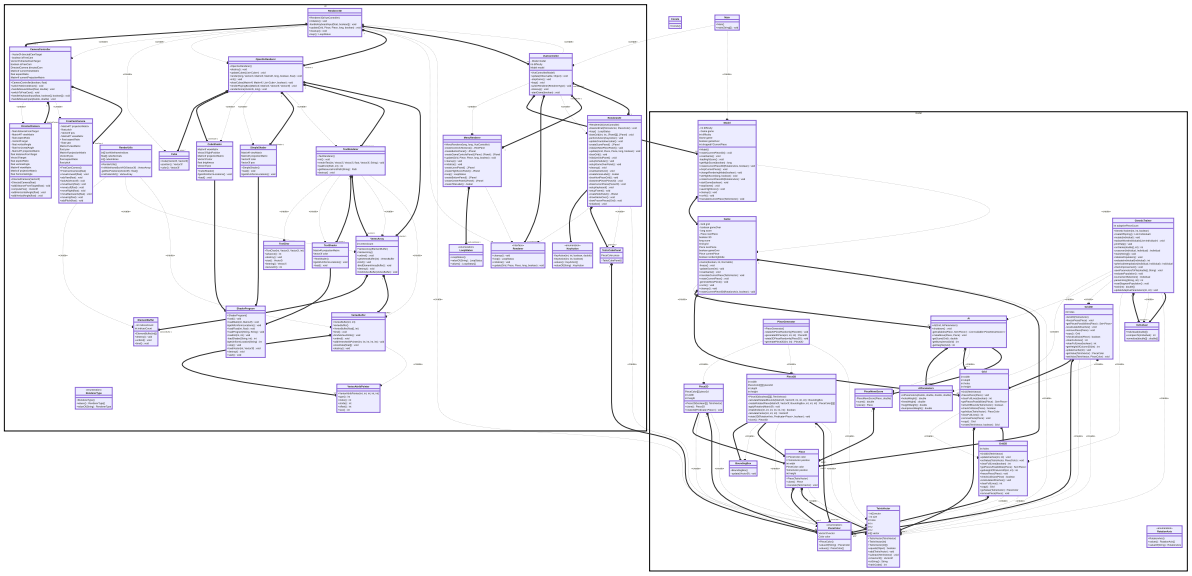
Ce projet est une implémentation du jeu Tetris en utilisant le modèle MVC (Modèle-Vue-Contrôleur). Il inclut des fonctionnalités avancées telles que l'intelligence artificielle, le rendu 3D, et bien plus.

1.1 Fonctionnalités

- **Jeu Tetris** : Une implémentation complète du jeu Tetris avec des rotations prenant en compte les wallkicks et les 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 2D** : Utilisation de Swing pour le rendu 2D, avec des blocs stylisés pour un effet Tetris classique.
- **Rendu 3D** : Utilisation d'OpenGL pour un rendu 3D (bas niveau), et implémentation de deux caméras (caméra libre et caméra dirigée).
- **Passage de 2D à 3D** : Possibilité de passer du mode 2D au mode 3D à tout moment, à travers un menu, sans redémarrer le jeu.
- **Rendu de texte en 3D** : Utilisation de FreeType (lib C avec des bindings Java) pour le chargement de polices et la transformation de chaque caractère en une texture et une list de vertices, permettant d'afficher des messages à l'écran.
- **Rotations en 3D** : Rotations en 3D, en prenant en compte l'orientation et la position de la caméra pour une utilisation plus intuitive.
- **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.

1.2 Diagramme de Classes



1.2.1 Package `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.

1.2.2 Package 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.
  - **`ElementBuffer`, `VertexArray`, `VertexBuffer`, `VertexAttribPointer`** : Classes d'abstraction pour gérer les différents types de buffers OpenGL.
  - **`CubeShader`, `SimpleShader`, `TextShader`, `ShaderProgram`** : Gèrent les shaders pour le rendu graphique. Le rendu des cubes prend en compte la réflexion de la lumière pour un meilleur effet de perspective.

1.3 Instructions de Build

1.3.1 Utilisation de Nix (recommandé)

Pour garantir que vous avez le même environnement de développement que nous, il est recommandé d'utiliser Nix pour un build reproductible.

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

Ou simplement :

```
nix run
```

1.3.2 Utilisation de Gradle

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

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

Ou simplement :

```
gradle run
```

1.4 Auteurs

- **Guillaume CALDERON** : p2205143
- **Eymeric DÉCHELETTE** : p2202851