Modélisation

Le jeu

La grille est modélisée par une matrice de booléens.
Les couleurs des cases sont contenues dans une deuxième matrice de type couleur.
Chaque pièce possède 4 cases, chaque case a une abscisse et une ordonnée, contenues respectivement entre 0 et 10 et 0 et 20. Chaque pièce possède sa propre couleur.

Les vérifications lors de déplacements/rotations se font avec les coordonnées de la pièce et la matrice de booléens. Une fois un déplacement validé, on insère les nouvelles coordonnées dans la matrice, ainsi que les couleurs (la pièce est retirée de la matrice avant chaque déplacement potentiel).

Des contrôles sont effectués pour éviter que la pièce ne sorte du cadre, et pour éviter les collisions entre pièces. Une rotation sur un bord de la grille va déplacer la pièce en plus de son mouvement de rotation pour qu'elle ne sorte pas du cadre, alors qu'une rotation près d'une autre pièce sera annulée si une collision en résulte. En effet, ajuster la pièce sur la grille selon les collisions entraînée par une rotation demanderait des vérifications en cascade, puisque l'ajustement lui-même pourrait provoquer d'autres collision. Il est donc nécessaire de se déplacer manuellement avant d'effectuer une rotation si cette rotation n'est pas validée de base.

Les rotations ne sont pas identiques à celles d'un "vrai" jeu Tetris, car elles sont effectuées selon un centre (une des cases de la pièce en question).
Par exemple, les Tetromino I, S et Z n'ont en principe que deux rotations possibles, ici ils en auront quatre. La rotation du tetromino O est désactivée.

Intelligence Artificielle

L'intelligence artificielle imite les méthodes de déplacement du jeu, en se ramenant à des coordonnées et une matrice de booléens (et non pas à des données de type Piece et Matrice). Pour une pièce donnée (des coordonnées), elle va simuler les 4 rotations possibles (sauf dans le cas du carré), puis toutes les positions possibles sur la grille.

A la fin de la simulation, le programme va sélectionner le meilleur score, et donc les mouvements effectués correspondants, et envoyer les touches correspondantes au jeu.

Module de Tests

Un module a été codé pour tester différentes situations et calculs. Il est possible de lancer le jeu avec des grilles pré-remplies, et une suite de pièces prédéfinie.

Répartition du travail

Valéry
- ✔ interface graphique
- ✔ débogage
- ✔ module de tests

Thomas
- ✔ timers
- ✔ génération aléatoire
- ✔ log
- ✔ IA

Timothée
- ✔ pièces
- ✔ méthodes de contrôle/vérification
- ✔ IA
- ✔ script algo génétique

Cependant nous ne nous sommes pas toujours restreints à cette répartition, chacun d'entre nous ayant parfois travaillé sur telle ou telle partie.

Détails des algorithmes

Vérification des lignes complétées

Le programme cherche des lignes complétées à chaque fois qu'une pièce est posée. S'il trouve une ligne, il vérifie les 3 lignes suivantes et s'arrête (on ne peut par définition compléter que 4 lignes en un coup). Puis chaque ligne complétée est supprimée de la matrice en partant du haut. Les pièces sont alors actualisées, les coordonnées des cases étant positionnées sur les lignes supprimées sont redéfinies à [10,20] (à l'extérieur de la grille), et les ordonnées des cases se trouvant au-dessus des lignes supprimées sont incrémentées (ici le point [0,0] se trouve tout en haut à gauche).

Après avoir vérifié les lignes complétées, actualisé la matrice et les pièces, le programme va retirer de la liste des pièces toutes celles qui ont été totalement détruites, c'est-à-dire les pièces dont les 4 cases ont pour coordonnées [10,20], afin de ne pas ralentir les prochaines vérifications.

Calcul du score

Pour une pièce donnée :

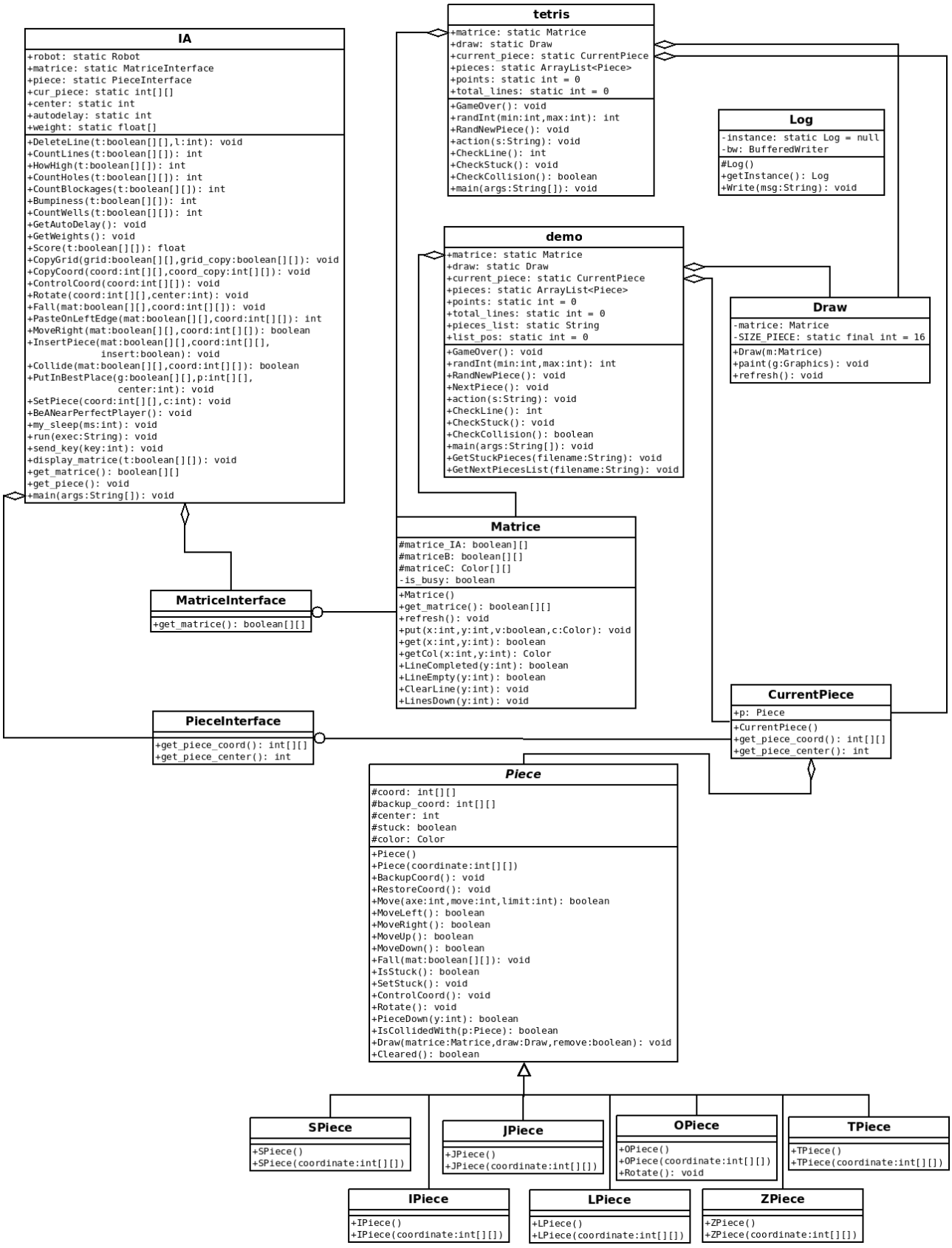
- ✗ on effectue de 0 à 3 rotations
- ✗ on se colle le plus à gauche possible
- ✗ on bouge de 0 à 10 fois à droite (tant que c'est possible)
- ✗ on chute (la pièce tombe le plus bas possible directement)
- ✗ on calcule un score

Pour calculer le score, la programme va d'abord compter les lignes complétées dans la grille une fois la pièce posée, puis les supprimer s'il y en a, et ensuite compter 5 autres données (ou critères) : les trous formés, la hauteur, les blocs au-dessus de trous, les puits (suite verticale d'au moins 3 trous), et le dénivelé (la variation de hauteur d'une colonne à une autre). Chacun de ces 6 critères est associé à un poids, récupéré dans un fichier de configuration. Le score est la somme des 6 critères multipliés par leurs poids respectifs.

Génétique

Ici, les poids utilisés sont déterminants dans l'efficacité de l'IA. On a donc décidé d'utiliser un algorithme basé sur la génétique pour améliorer ces poids. Cet algorithme est écrit en bash. Le script va générer N sets aléatoires de paramètres (nos poids), puis lancer le programme avec chacun de ces sets, et ne gardera que les plus performants (selon le nombre de ligne ou le score atteint). Ces "survivants" donneront naissance à d'autres sets en formant tous les couples possibles, ainsi pour X survivants, on aura (2 parmi X) "enfants". Les paramètres d'un enfant sont les moyennes des paramètres de ses "parents". Chacun des paramètres de ces nouveaux sets sera modifié aléatoirement, cela correspond à une mutation. Le script réitère alors en lançant le programme avec les survivants et les enfants, l'ensemble de ceux-ci définissant une nouvelle génération de sets.

Diagramme d'héritage



Utilisation

3 scripts de lancement

- **go_tetris** : lance le jeu normalement
- **go_demo** : lance la démo avec une liste de pièces [et une grille pré-remplie]
- **go_ia** : lance l'intelligence artificielle sur le jeu normal ou sur une démo

go_demo et *go_ia* requièrent des arguments sur la ligne de commande, et de ce fait procèdent à une vérification des arguments passés. L'usage est affiché en cas d'erreur de syntaxe.

Pendant une partie normale

Touche flèche gauche, droite et bas pour se déplacer.
Touche flèche haut pour effectuer une rotation.
Touche Espace pour faire tomber la pièce tout en bas.
Le score et le nombre de lignes complétées sont affichés à droite de la grille.
Le jeu s'arrête dès qu'il y a collision à l'apparition d'une nouvelle pièce.
Les données sont sauvegardées dans le fichier *tetris.log*.

Pendant une partie démo

Utilisation identique

Pendant une partie IA

Utilisation identique
On peut "saborder" l'IA en jouant en même temps qu'elle :)

Script genetic_algorithm.sh

Sans argument, le script affiche sa syntaxe d'utilisation.
Avec -h ou --help, affiche une aide plus détaillée.
custom : script personnalisé d'utilisation de genetic_algorithm.sh