COMPTE RENDU

JEU KHET

sous la supervision de Mr Padovani Vincent

MEMBRES DU GROUPE:

Paul Nabti
Sarah Salhi
Maroun Gebrayel
Abderrahman Ajinou
Omar Konate

PARTIE 1 : UNE BRÈVE EXPLICATION DU JEU

Khet est un jeu de plateau stratégique pour deux joueurs qui combine des éléments de stratégie et de réflexion. Le jeu est souvent comparé aux échecs, mais avec une touche unique : l'utilisation de lasers. Chaque joueur dispose de pièces de jeu dotées de miroirs et d'autres propriétés uniques. Le but du jeu est de diriger un laser pour toucher le pharaon adverse tout en protégeant son propre pharaon.

Composition du Plateau de Jeu:

Le plateau de jeu Khet est composé d'une grille de cases sur laquelle les pièces des deux joueurs sont disposées. Les pièces sont réparties en deux couleurs : orange et bleu, représentant les deux joueurs.

Les Pièces:

- *Pharaon : La pièce la plus importante. Le but est de protéger votre propre Pharaon et de toucher celui de votre adversaire avec le laser.*
- *Pyramides : Pièces triangulaires avec une face réfléchissante. Elles dévient le laser à +/-90 degrés.*
- *Djed : Pièces rectangulaires avec deux faces réfléchissantes. Elles peuvent être utilisées pour dévier le laser dans différentes directions.*
- *Horus : Pièces qui divisent le laser en deux faisceaux.*
- *Obélisques : Pièces qui bloquent le laser sans le dévier. Elles peuvent être empilées pour former des doubles Obélisques.*

Règles de Base:

Déplacement des pièces : Chaque joueur peut déplacer une de ses pièces d'une case adjacente dans n'importe quelle direction ou faire pivoter une pièce de +/-90 degrés par tour.
Utilisation du Laser : À la fin de chaque action menée par le joueur, le laser en temps réel se met à jour et se propage à travers le plateau en étant dévié par les miroirs des pièces. Le laser suit une trajectoire définie par les interactions avec les pièces.
Condition de Victoire : Le jeu se termine lorsqu'un laser touche un Pharaon. Le joueur dont le Pharaon est touché perd la partie.

Objectif du Jeu:

Le but ultime est d'utiliser judicieusement les mouvements et les rotations des pièces pour diriger le laser vers le Pharaon de l'adversaire tout en évitant que le laser adverse ne touche le vôtre.

PARTIE 2 : L'ARCHITECTURE DU JEU

Architecture du Jeu Khet:

Le jeu Khet est structuré autour de plusieurs composants clés qui interagissent pour créer une expérience de jeu dynamique et réflexive. Chaque composant a un rôle spécifique, orchestrant ensemble le déroulement du jeu.

Classes Principales:

- GameView : Cette classe sert de vue principale pour l'interface utilisateur du jeu. Elle gère l'affichage de l'état actuel du plateau de jeu et interagit directement avec l'utilisateur. Elle inclut également des éléments de l'interface tels que les boutons pour pauser le jeu, accéder aux paramètres ou quitter le jeu.
- BoardPanel : Il s'agit de la représentation graphique du plateau de jeu. Cette classe gère le rendu des pièces sur le plateau, les interactions de la souris pour le déplacement et la rotation des pièces, et l'affichage des trajectoires du laser.
- Game : Cette classe logique contrôle la progression du jeu, les règles, et la gestion des tours. Elle fait le lien entre les actions de l'utilisateur et l'état du jeu, décidant par exemple du joueur suivant ou de la fin de la partie.

Modélisation des Pièces et du Plateau:

- Plateau : Gère la disposition des pièces sur le plateau et contient la logique pour mettre à jour le plateau après chaque action. Cela inclut le déplacement des pièces, les interactions entre elles, et la propagation des lasers.
- Pion : Représente les diverses pièces du jeu, comme les Pharaons, les Pyramides, les Djed, etc. Chaque pion est conscient de son type, sa position, et peut interagir selon les règles spécifiques à son type (rotation, empilement, dépilement).

Gestion des Laser:

- Laser : Représente le laser pour chaque joueur, capable de calculer et d'illustrer le chemin du laser à travers le plateau en fonction de l'orientation et de la position des pièces réfléchissantes.

Interaction Utilisateur et Temps de Jeu

- Menu : Fournit l'écran d'accueil et le menu principal du jeu où les utilisateurs peuvent démarrer le jeu, ajuster les paramètres, ou quitter l'application. Il joue également des sons lors des interactions pour renforcer la réponse visuelle.
- GameTimer : Une classe dédiée à la gestion du temps pour chaque tour, assurant que les joueurs doivent faire un mouvement dans un temps imparti, augmentant ainsi la difficulté et le dynamisme du jeu.

Observation et Notifications:

Observer et Observable : Un modèle observer est utilisé pour permettre à différentes parties du jeu de réagir aux changements d'état. Par exemple, lorsque le laser touche le Pharaon, un événement est déclenché pour informer le système du jeu que la partie est terminée.
Interaction Entre les Classes
Les interactions entre ces composants sont principalement centrées autour du flux de jeu. L'utilisateur interagit avec le GameView, qui capture les commandes utilisateur et les transmet au BoardPanel pour l'exécution. Les actions sur le plateau modifient l'état du jeu dans la classe Game, qui met à jour le plateau à travers la classe Plateau. Les résultats de ces actions (par exemple, le déplacement d'une pièce ou le tir d'un laser) sont ensuite reflétés visuellement sur le BoardPanel.

La classe Menu sert de point d'entrée, lançant le GameView et permettant aux utilisateurs de configurer leur jeu ou de quitter l'application. La gestion du temps par GameTimer garantit que le jeu progresse à un rythme soutenu et que les joueurs sont limités dans le temps pour effectuer leurs actions.

En résumé, l'architecture du jeu Khet est conçue pour séparer clairement la logique du jeu de l'interface utilisateur, tout en assurant une interaction fluide et réactive entre les différentes parties du système de jeu.

PARTIE 3 : RÉSUMÉ DES SPRINTS

Le développement du projet Java Khet (Laser Game) a suivi une progression méthodique à travers plusieurs sprints, chaque étape apportant des améliorations et des fonctionnalités clés.

Voici un résumé de ces sprints :

- *Classes de base et architecture MVC : Implémentation des classes de base du jeu, discussions sur l'architecture MVC, conception de la classe Grille et définition des types de pions avec leurs énumérations correspondantes.*

- *Mouvements et comportements des pions : Élaboration des mouvements spécifiques des pions et leurs comportements uniques, incluant la finalisation des énumérations pour les directions et l'implémentation des capacités de réflexion ou de blocage du laser.*

- *Tests et intégration : Intégration et tests des fonctionnalités développées, avec un accent sur le développement de l'interface utilisateur et la logique du laser. Implémentation des mécanismes de rotation et d'échange de pièces, respectant les règles du jeu.*

- *Développement de l'interface utilisateur : Renforcement de l'application de la logique MVC, développement de l'interface utilisateur, implémentation de la logique du laser, et finalisation des directions cardinales et intermédiaires. Restructuration des fichiers et finalisation des types de pions.*

- *Amélioration des lasers et notifications : Tests et améliorations des chemins des lasers, détection des pièces rencontrées et réflexion du laser. Mise en place des classes Observer et Observable pour intégrer la logique de notification des changements d'état du plateau.*

- *Résolution des bugs et ajout de fonctionnalités : Résolution des bugs concernant l'affichage du laser et les mouvements par rapport à la taille du panel. Ajout de fonctionnalités comme l'apparition des cases colorées et l'implémentation du tour par tour, ainsi qu'un écran de GameOver.*

En plus de ces fonctionnalités, nous avons développé différents niveaux de jeu, ajouté un timer avec un nombre de secondes différent pour chaque mode de jeu, et intégré des sons pour les boutons, rendant l'expérience de jeu plus dynamique et immersive. Le projet s'est ainsi achevé sur un jeu fonctionnel avec des interactions laser-pions intégrées et une interface utilisateur développée selon la logique MVC.

PARTIE 4 : LES PROBLÈMES RENCONTRÉS

Dans le développement du jeu Khet, plusieurs problèmes techniques et conceptuels ont été rencontrés et résolus.

Voici une description détaillée de ces défis :

Structure et arborescence du code:

- *Problème :*

Déterminer une structure optimale pour organiser les classes, méthodes et interactions.

- *Solution :*

MVC (Model-View-Controller) : L'adoption du modèle MVC a aidé à séparer clairement les responsabilités. Les classes du modèle (Plateau, Pion, Laser) gèrent la logique du jeu, les classes de la vue (GameView, BoardPanel) s'occupent de l'affichage, et les classes du contrôleur orchestrent les interactions utilisateur et les mises à jour du modèle.
Organisation des Packages : Le code a été organisé en plusieurs packages, chacun ayant une responsabilité distincte (model, view, controller, etc.), facilitant ainsi la navigation et la gestion des dépendances entre classes.

Approche de codage:

- *Problème :*

Décider de la manière d'écrire le code avant de s'occuper de l'affichage et gérer l'absence de feedback visuel immédiat pour évaluer l'avancement du projet.

- *Solution :*

Test-Driven Development (TDD) : En écrivant des tests unitaires pour la logique du jeu, il était possible de vérifier que chaque composant fonctionnait correctement avant de l'intégrer dans l'interface utilisateur.
Prototypage : La création de prototypes pour les principales fonctionnalités a permis de valider les choix de conception avant de les intégrer dans l'application principale.
Log Statements : Utilisation de nombreuses instructions de log (System.out.println) pour suivre l'exécution du code et comprendre le comportement du jeu.
Éléments UI temporaires : Ajout d'éléments UI temporaires pour visualiser les changements d'état pendant le développement.
Défis techniques
Déplacement des pions

- *Problème :*

Implémenter la logique pour déplacer les pions selon les règles spécifiques du jeu Khet tout en gérant les collisions et les mouvements invalides.

- *Solution :*

Développement de méthodes spécifiques dans la classe Plateau pour gérer les déplacements valides et invalides des pions. Empilement et dés-empilement des pions Obélisque

- *Problème :*

Gérer l'empilement et le dés-empilement des pions Obélisque pour former des doubles Obélisques.

- *Solution :*

Utilisation de méthodes polymorphiques et de logique conditionnelle pour gérer ces interactions complexes. Mise en place des règles pour chaque pion

- *Problème :*

Chaque type de pion (Pharaon, Pyramide, Djed, Horus) a des règles spécifiques pour le mouvement et l'interaction avec les lasers.

- *Solution :*

Utilisation d'une classe abstraite Pion avec des sous-classes pour chaque type de pion, encapsulant les règles spécifiques. Méthodes polymorphiques pour gérer les comportements spécifiques à chaque type de pion.

Gestion des boutons de la souris:

- *Problème :*

Gérer les interactions complexes de la souris, incluant les clics gauche et droit, et les glissements pour déplacer les pions.

.

- *Solution :*

Implémentation de MouseListener et MouseMotionListener pour capturer et gérer les événements de la souris. Différenciation des actions en fonction du bouton de la souris et de l'état du jeu. Ajustement de la taille du plateau en fonction de la taille de la fenêtre

- *Problème :*

Assurer que le plateau de jeu s'ajuste correctement à la taille de la fenêtre pour offrir une expérience utilisateur optimale.

- *Solution :*

Utilisation de LayoutManager comme BorderLayout pour gérer dynamiquement la taille et la position des composants. Redimensionnement automatique des éléments graphiques en fonction des dimensions de la fenêtre.

Laser

- *Problème :*

Implémenter la propagation du laser, ses interactions avec les pions, et l'affichage de sa trajectoire.

- *Solution :*

Développement d'un algorithme pour calculer la trajectoire du laser en fonction des orientations des pions. Utilisation de Graphics2D pour dessiner la trajectoire du laser sur le plateau.

Affichage plateau

- *Problème :*

Rendre un plateau de jeu graphique interactif, reflétant fidèlement l'état du jeu.

- *Solution :*

Utilisation du double buffering pour réduire le scintillement et améliorer les performances de rendu. Appels fréquents à repaint pour mettre à jour l'affichage après chaque changement d'état.

Moteur pour extraire les pièces

- *Problème :*

Développer un moteur pour gérer l'extraction et la manipulation des pièces sur le plateau.

- *Solution :*

Utilisation de méthodes abstraites pour gérer les interactions avec les pièces, facilitant l'ajout de nouveaux types de pions ou de nouvelles règles. Conception modulaire pour permettre l'extension et la modification facile du moteur de jeu.

PARTIE 5 : ET LA SUITE

plusieurs aspects que nous souhaitons améliorer et de nouvelles fonctionnalités que nous envisageons d'ajouter.
Voici un aperçu des prochaines étapes :

1. Améliorations de l'interface utilisateur:

Affichage et animations des lasers : Actuellement, la propagation du laser est fonctionnelle, mais nous pouvons ajouter des animations plus fluides et visuellement attractives pour améliorer l'expérience utilisateur.
Indications visuelles : Ajouter des indications visuelles pour les mouvements possibles, les cases sur lesquelles un pion peut se déplacer ou pivoter.
Thèmes et personnalisation : Permettre aux joueurs de choisir des thèmes de plateau et des apparences de pions pour personnaliser leur expérience de jeu.

2. Fonctionnalités multijoueurs:

Multijoueur en ligne : Implémenter un mode multijoueur en ligne permettant aux joueurs de se connecter et de jouer ensemble via Internet. Cela nécessitera la mise en place d'un serveur et d'une infrastructure réseau pour gérer les parties.

3. Intelligence artificielle:

- *Adversaire IA :*

Développer une intelligence artificielle capable de jouer contre un joueur humain. Cela inclura différents niveaux de difficulté pour s'adapter aux compétences des joueurs.

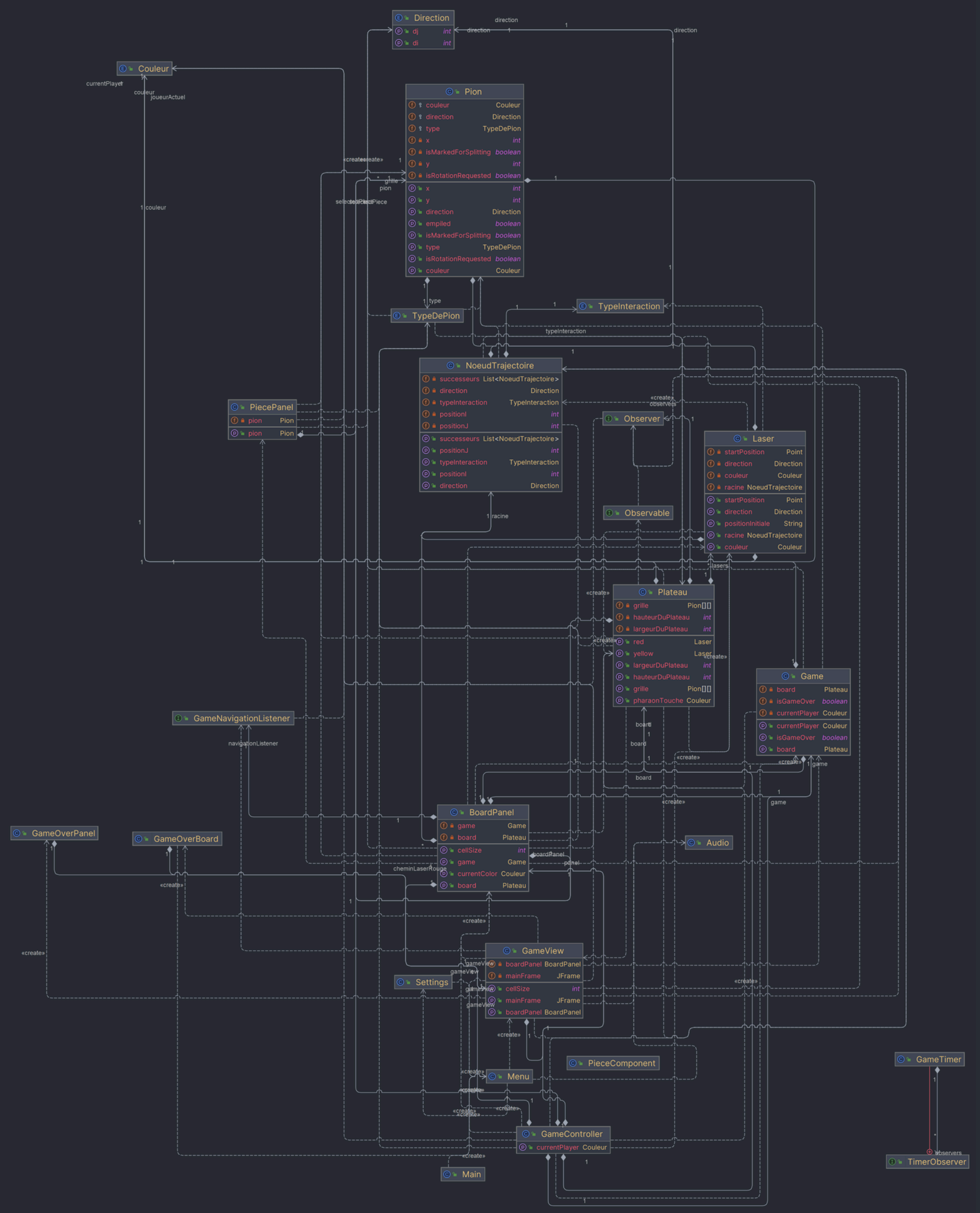
- *Algorithmes de prise de décision :*

Implémenter des algorithmes de prise de décision basés sur des techniques de machine learning et d'heuristiques pour rendre l'IA compétitive et engageante.

4. Optimisation et performance:

Amélioration des performances : Optimiser le code pour s'assurer que le jeu fonctionne de manière fluide même sur des machines moins puissantes.
Tests et débogage : Continuer à effectuer des tests rigoureux pour identifier et corriger les bugs, ainsi que pour améliorer la stabilité générale du jeu.

ARBORESCENCE DES CLASSES



CONCLUSION

Le développement de Khet ne s'arrête pas à sa version actuelle. Nous avons l'ambition de continuer à améliorer et à enrichir le jeu pour offrir une expérience de plus en plus immersive et captivante. Grâce aux retours des joueurs et à l'innovation continue, nous espérons faire de Khet un jeu de référence pour les amateurs de jeux de stratégie et de réflexion.