Compte-rendu du projet : Les Dames Chinoises

Contexte

Le projet consiste à implémenter une variante du jeu des Dames Chinoises en utilisant le langage de programmation fonctionnelle OCaml. Le jeu se joue sur un plateau en forme d'étoile à six branches, avec des pions que les joueurs doivent déplacer de leur camp vers le camp opposé. Le projet est divisé en deux phases, avec des rendus intermédiaires et une soutenance finale.

Objectifs

1. **Modélisation du jeu** : Définir les types et structures de données pour représenter le plateau, les pions, les coups, et les configurations du jeu.

2. **Fonctionnalités de base** :

   - Vérifier la validité des cases et des coups.

   - Gérer les déplacements unitaires et les sauts multiples.

   - Mettre à jour la configuration du jeu après chaque coup.

3. **Fonctionnalités avancées** :

   - Calculer les coups possibles.

   - Déterminer le gagnant.

   - Implémenter une interface graphique (optionnelle).

Structure du projet

Le fichier `rendu_etd.ml` fournit une base avec :

- **Types** : `dimension`, `case`, `vecteur`, `couleur`, `case_coloree`, `configuration`, et `coup`.

- **Fonctions à compléter** :

- `est_dans_etoile` et `est_dans_losange` pour vérifier la position des cases.

- Fonctions pour les déplacements (`translate`, `diff_case`), les sauts (`calcul_pivot`), et la gestion des configurations (`supprime_dans_config`).

- Fonctions pour l'affichage (`affiche`).

Travail réalisé

1. **Partie 1** :

- **Q1-Q3** : Définition des conditions pour vérifier si une case est dans l'étoile ou le losange Nord-Sud.

- **Q4-Q6** : Implémentation des fonctions pour la rotation des cases et les translations.

- **Q7-Q9** : Vérification des cases voisines, calcul des pivots pour les sauts, et gestion des vecteurs de déplacement.

2. **Partie 2** :

- **Q10-Q16** : Fonctions pour manipuler les listes (rotation, remplissage de segments et triangles), et création de la configuration initiale.

- **Q17-Q21** : Recherche et suppression de cases, validation et application des coups unitaires.

- **Q22-Q25** : Gestion des sauts multiples et intégration dans les fonctions de coup.

3. **Partie 3** :

- **Q26-Q28** : Calcul des scores, détermination du gagnant, et vérification d'une partie valide.

- **Bonus (Q29-Q30)** : Calcul des coups possibles et stratégie pour maximiser le score.

Difficultés rencontrées

- **Coordonnées 3D** : La gestion des cases avec des coordonnées (i, j, k) telles que i + j + k = 0 a nécessité une attention particulière.

- **Sauts multiples** : L'implémentation des sauts avec pivots et la vérification des cases libres entre les sauts ont été complexes.

- **Validation des coups** : Assurer que les coups respectent les règles tout en gérant les cas particuliers (comme les camps de côté).

Solutions apportées

- **Approche modulaire** : Découpage des problèmes en petites fonctions réutilisables.

- **Tests rigoureux** : Utilisation d'exemples concrets pour vérifier chaque fonction.

- **Documentation** : Commentaires détaillés pour expliquer les choix d'implémentation.

Résultats

- Le projet permet de jouer une partie complète avec déplacements unitaires et sauts multiples.

- L'affichage du plateau fonctionne correctement, comme le montrent les exemples fournis (`conf_1`, `conf_reggae`, `conf_init`).

- Les fonctions de validation et de score offrent une base solide pour des extensions futures (IA, interface graphique).

Perspectives

- **Améliorations possibles** :

  - Optimisation des performances pour les grands plateaux.

  - Implémentation d'une IA pour jouer contre l'ordinateur.

  - Interface graphique plus interactive.

- **Extensions** :

  - Variantes des règles (comme les sauts asymétriques mentionnés dans le bonus).

Conclusion

Ce projet a permis de maîtriser les concepts de la programmation fonctionnelle avec OCaml, tout en développant un jeu stratégique complet. Les défis techniques ont été surmontés grâce à une approche méthodique et des tests approfondis. Le code est prêt pour des extensions futures, offrant une base solide pour des fonctionnalités supplémentaires.