5ème (et dernière) étape - UE S2 Projet

1 Plusieurs axes de développement : améliorations et/ou IA avec heuristique

Pour cette dernière étape vous êtes invités à finaliser votre programme. Cette étape sera corrigée par les enseignants. Voici deux axes à explorer. Le premier est d'améliorer ce que vous avez fait jusque là et de rattraper votre éventuel retard. Le second est de developper une IA guidée par une heuristique.

2 Les améliorations

Voici une liste non exhaustive des améliorations possibles :

- améliorer les structures de données : pas de variables globales (sauf justification), différencier la représentation graphique (afficher des caractères, des lignes/colonnes alphanumériques) et la représentation des données (matrice d'entiers par exemple), permettre un changement de taille de plateau,
- améliorer l'interaction avec le joueur : pas d'interruption du programme à la saisie, messages explicites, fluidité du jeu, permettre l'erreur de de saisie...
- améliorer la lisibilité du code : conserver des fonctions de moins de 15 lignes, faire des sous-fonctions, paramétrer les codes "qui se répètent"...
- améliorer la fonction générale de tests : s'assurer que pour chaque fonction il y a une fonction de tests unitaires
- améliorer les algorithmes : s'assurer que les enchaînements (quand ils sont dans la règle) respectent la règle, s'assurer que l'IA naïve est facile à suivre pour le joueur...

3 IA guidée

Pour chaque règle de jeu, nous vous proposons des *heuristiques*[1] qui orientent les choix. Une heuristique va permettre de "classer" les propositions et donc de choisir la meilleure selon cette heuristique.

A minima, vous respecterez ces heuristiques. Si vous souhaitez appliquer d'autres heuristiques (ou d'autre type d'algorithme), vous justifierez vos propositions dans un fichier .txt

4 L'Intelligence Articifielle

1. Ces heuristiques ne sont pas forcément gagnantes, il s'agit d'une stratégie proposée.

4.1 Rappel de l'algo proposé

```
Créer autant de liste que de types de déplacements possibles
Pour chaque case de la grille
    Si elle contient un pion du joueur
        Pour chaque type de déplacement possible
            Ajouter les couples départ/destination dans sa liste de déplacements
                possibles (et qui respectent la règle)
Choisir au hasard un type de déplacement
Choisir au hasard un couple dans la liste correspondant à ce déplacement et l'appliquer
Tant que le type de déplacement choisi autorise un enchaînement
   Choisir au hasard si on souhaite essayer de l'appliquer
   Si oui
    Calculer tous les couples départ/arrivée possibles (en excluant le coup précédent)
    Choisir au hasard parmi ce couple et l'appliquer
    Recommencer
  Si non
    sortir.
```

- Pour Açores : L'heuristique proposée est de choisir le déplacement "saut avec prise" plutôt que le déplacement "simple" et d'enchaîner quand cela est possible.
- Pour Bermudes : L'heuristique proposée est de choisir le déplacement "prise par élimination" plutôt que "prise par retournement".
- Pour Canaries : L'heuristique proposée est de choisir un déplacement "capture" plutôt que le déplacement "simple".

4.2 Classement des coups

En général, pour les étudiants les plus avancés, vous pouvez lister l'ensemble des coups possibles et leur donner un score en fonction :

- du nombre de pions capturés,
- de la possibilité d'enchaîner,
- de la position d'arrivée dans le plateau : contre le bord ou isolé pour minimiser les risques de capture ou dans le camp adverse (pour certains jeux)...

Par exemple, simple = 0 points, capture = 1 point, capture + enchainement = 2 points. La note en fonction de la position d'arrivée du pion : s'il n'est pas "prenable" = 2 points, si la position est sur un bord = 1 point, sinon 0. Il suffit ensuite de choisir le coup qui donne le plus de points. Pour les points d'enchaînement, attention à ne pas "revenir en arrière".

Il faudra ensuite choisir le coup qui donne le plus de points. Attention, regarder plus d'un coup à l'avance demande des structures de données efficaces.

5 Fonctionnalités attendues (et notées)

5.1 Du point de vue "joueur"

- un ou plusieurs menus pour choisir : la configuration, le type de partie (joueur contre joueur, joueur contre IA naïve, IA avancée...)

— le programme ne s'interrompt pas si on se trompe lors d'une saisie (robustesse)
— les règles du jeu sont vérifiées
— c'est "jouable" : pas trop de messages, on sait qui joue (et quels pions), le joueur sait ce qui est attendu comme réponse (lettre, chiffre...), où il en est dans la partie (affichage de la grille régulièrement)...

5.2 Du point de vue "évaluateur" et "programmeur"

— le programme principal et les fonctions font moins de 15 lignes (exception tolérée pour les fonctions d'affichage).
— les algorithmes ne sont pas trop compliqués
— le code est structuré à l'aide de commentaires pour séparer les différentes fonctionnalités (par exemple : partie affichage, partie tests, partie "petites fonctions de vérification", partie déplacement, partie "jeu", partie IA...)
— le code est commenté ni trop, ni trop peu
— il y a une fonction générale de tests qui appelle les fonctions de tests unitaires (il y a donc des fonctions de tests unitaires pour les "petites" fonctions)

6 Dépôt

Vous rendrez donc un fichier .py et éventuellement un fichier .txt (uniquement si vous avez à proposer une heuristique différente).

ATTENTION aux noms de vos fichiers (si les noms ne sont pas respectés, vous aurez 1 points en moins sur cette évaluation) :

— le fichier des fonctionnalités doit porter le nom du jeu : `acores.py`, `bermudes.py` ou `canaries.py`
— le fichier (facultatif) de description des heuristiques doit porter le nom `heuristiques.txt`
— PAS DE FICHIER .ZIP

Le programme doit bien entendu poursuivre le travail des ateliers précédents : les fonctionnalités demandées en atelier 2, 3 et 4 doivent se retrouver, même programmées différemment (pour les améliorer).

Il est possible de "s'inspirer" des ateliers 2, 3 et 4 corrigés lors de l'évaluation par les pairs, sans copier. Les codes des ateliers 2, 3, 4 et 5 seront comparés par nos soins à l'aide de l'outil moss.

7 Evaluation

Cette partie sera une partie évaluée par les enseignants. D'une manière générale, nous évaluerons plus en profondeur la qualité du code (et pas uniquement les aspects Clean Code ni la présence des fonctionnalités). Nous évaluerons également votre approche IA avancée.

La grille du barème est donnée dans la section moodle.

8 Conseils

8.1 Fraudes

Les programmes seront tous testés à l'aide de l'outil moss. S'il s'avère que des programmes sont identiques (ou similaires), vous serez sanctionnés au delà du simple 0 à l'atelier.