

AI Pêche aux Moules

Manuel Utilisateur

KARAMI Anir • BUT3APP TPC

Objectif

Comprendre les règles et lancer une partie locale rapidement. Ce manuel se concentre sur l'usage joueur et la lecture des actions.

Table des matières

1 Démarrage rapide	3
1.1 Serveur	3
1.2 Client IA	3
1.3 Paramètres de partie (serveur)	3
2 Règles du jeu	3
2.1 Ce qui se passe quand on entre sur une case	3
2.2 Terrain et symboles	4
2.3 Murs, bords et collisions	4
2.4 Bonus (règle générale)	4
2.5 Coordonnées	4
3 Actions disponibles	4
3.1 Cas limites importants	5
3.2 Exemples	5
4 Bonus et score	5
4.1 Inventaire	6
5 Exemple de tour (log)	6
5.1 Format du message serveur	6
6 Fin de partie	6
7 Mode multi-joueur	6
8 Utiliser l'IA	6
8.1 Connexion	7
8.2 Paramètres utiles	7
9 Comment l'IA décide (résumé)	7
9.1 Pseudo-code minimal	7
10 Lecture des logs	7
11 Conseils	8
12 Stratégie avancée	8

13 FAQ	8
14 Glossaire et conventions	8
15 Crédits	8
16 Ressources	9

1 Démarrage rapide

En 5 minutes

- Compiler le serveur.
- Lancer le serveur.
- Compiler le client IA.
- Lancer le client IA.

1.1 Serveur

```
cd Serveur  
javac PecheAuxMoulesBoucle.java  
java PecheAuxMoulesBoucle -nbJoueur 1 -delay 0 -timeout 3000
```

1.2 Client IA

```
javac -d IA IA/superAI/*.java  
java -cp IA superAI.ClientSuperAI 127.0.0.1 1337 MonEquipe
```

1.3 Paramètres de partie (serveur)

Option	Effet
-nbJoueur	Nombre de joueurs attendus
-delay	Délai entre tours (ms)
-timeout	Durée max de la partie (ms)
-numLaby	Seed du labyrinthe
-numPlacementBonus	Seed des bonus
-tauxDeMur	Pourcentage de murs (0 à 50)

2 Règles du jeu

But

Ramasser le maximum de moules (points) avant la fin de partie. Les bonus permettent d'accélérer les déplacements.

2.1 Ce qui se passe quand on entre sur une case

- Une moule ajoute immédiatement sa valeur au score.
- Un bonus est ajouté à l'inventaire du joueur.
- La case devient ensuite du sol (So).

- Le ramassage se fait après l'action, même si le joueur n'a pas bougé.

2.2 Terrain et symboles

Symbole	Signification
Mu	Mur (infranchissable)
So	Sol vide
Bs	Bonus saut (2 cases)
Bp	Bonus trois pas
Nombre	Moule (valeur en points)

Variété des labyrinthes

Les cartes peuvent être très fermées (labyrinthe « parfait ») ou plus ouvertes avec des zones larges et moins de murs.

2.3 Murs, bords et collisions

- Le labyrinthe est entouré de murs, il n'y a pas de sortie.
- Un déplacement vers un mur est refusé (le joueur reste sur place).
- Plusieurs joueurs peuvent se trouver sur la même case.

2.4 Bonus (règle générale)

- Un bonus est gagné quand on ramasse la case correspondante.
- Les bonus sont stockés dans l'inventaire du joueur.
- Bonus saut (Bs) : permet Bs-D (saut de 2 cases), même si un mur est entre les deux.
- Bonus trois pas (Bp) : permet Bp-D1-D2-D3 (3 pas).

2.5 Coordonnées

Origine en haut-gauche. x augmente vers la droite, y vers le bas. Les indices commencent à 0.

3 Actions disponibles

Actions simples

N, S, E, O : déplacement d'une case (Nord, Sud, Est, Ouest).
C : passe son tour.

Bonus saut

Bs-D : saut de deux cases dans la direction D. Si la case à +2 est un mur, un pas simple est tenté.

Bonus trois pas

Bp-D1-D2-D3 : trois déplacements simples successifs. Les pas invalides n'annulent pas les suivants.

3.1 Cas limites importants

- Si la case cible est un mur, le déplacement est annulé.
- Un saut n'est consommé que si la case à +2 est valide (sol, bonus ou moule).
- Si la case à +2 est un mur, un pas simple est tenté (bonus conservé).
- Si la case à +2 est hors plateau, le saut est ignoré.
- Bs ne ramasse que la case d'arrivée.
- Si le joueur n'a pas de bonus, l'action Bs/Bp est ignorée.
- Si la direction de Bs est invalide, l'action est ignorée.
- Pour Bp, chaque pas est tenté séparément (les murs bloquent seulement ce pas).
- Pour Bp, une direction invalide est traitée comme C.
- Bp consomme toujours le bonus si le joueur en possède un.
- Bp ramasse sur chaque case atteinte.

3.2 Exemples

N

Bs-E

Bp-N-N-0

Aide-mémoire rapide

Action	Effet
N/S/E/0	Avancer d'une case
C	Passer le tour
Bs-D	Saut de deux cases
Bp-D1-D2-D3	Trois pas successifs

4 Bonus et score

- Une moule ajoute sa valeur au score.

- Un bonus est ajouté à l'inventaire quand la case est ramassée.
- Les bonus sont consommés lors de l'action correspondante.
- Les valeurs des moules sont fixées au début de la partie (multiples de 10, 10 à 90).

4.1 Inventaire

Chaque joueur possède un compteur de bonus saut et bonus trois pas. L'IA affiche leur évolution dans les logs (gain et utilisation).

5 Exemple de tour (log)

```
reçu: 25x17/So-So- ... /1-11,3  
action envoyée: Bs-E
```

Ce type de log permet de vérifier que l'action correspond bien aux règles.

5.1 Format du message serveur

LxH/structure/infosJoueurs

La structure est une liste de L*xH tokens (de gauche à droite, de haut en bas).

6 Fin de partie

- Toutes les moules ont été ramassées.
- Limite de tours atteinte.
- Tous les joueurs passent leur tour.

Le serveur envoie alors le message FIN.

7 Mode multi-joueur

Le serveur attend -nbJoueur connexions. Chaque client envoie ses actions indépendamment. En multi, il est souvent utile d'abandonner une cible si un adversaire est clairement plus proche.

8 Utiliser l'IA

Aide rapide

```
java -cp IA superAI.ClientSuperAI -h
```

8.1 Connexion

Au démarrage, le client envoie le nom d'équipe, puis reçoit un numéro de joueur attribué par le serveur.

Port par défaut

Le serveur écoute par défaut sur le port 1337.

8.2 Paramètres utiles

- log : active les logs.
- logFichier=CHEMIN : écrit le log dans un fichier.
- modePlan=1 : planification locale des cibles.
- modeBeam=0/1/2 : beam off / on / auto.

Preset par défaut

Le preset conseillé est **G1B0** : rapide et stable, avec les modules optionnels désactivés (meilleure robustesse sur des seeds variées).

9 Comment l'IA décide (résumé)

Pipeline

- Calcul des distances (BFS) depuis la position du joueur.
- Évaluation des cibles (moules, bonus) selon valeur et distance.
- Sélection d'une action via scoring + mémoire anti-boucle.
- Optionnel : beam search pour simuler plusieurs coups.

9.1 Pseudo-code minimal

```
etat = lire_serveur()
dist = bfs(etat)
cible = choisir_cible(etat, dist)
action = argmax(score(action, cible))
envoyer(action)
```

10 Lecture des logs

- **Serveur** : valide les mouvements, affiche le score final.
- **Client** : affiche l'action choisie et les bonus utilisés.

Où les trouver ?

Les logs client sont écrits dans IA/superAI/logs/ si l'option log est activée.

11 Conseils

- Préserver les bonus pour les trajets longs.
- En multi, éviter une cible déjà perdue face à un adversaire plus proche.
- Sur carte dense, le beam auto peut aider à éviter les boucles.

12 Stratégie avancée

- **Valeur vs distance** : une grosse moule loin n'est pas toujours rentable.
- **Sauts utiles** : garder les bonus pour franchir des zones de murs.
- **Anti-aller-retour** : si l'IA oscille, augmenter les pénalités.
- **Multi** : si l'adversaire est plus proche, changer de cible vite.

13 FAQ

Connexion refusée

Le serveur n'est pas lancé ou le port est incorrect.

ClassNotFound

Recompiler le client : javac -d IA IA/superAI/*.java.

L'IA tourne en rond

Essayez modeBeam=2 (auto) ou augmentez légèrement penaliteAllerRetour.

14 Glossaire et conventions

- **Seed** : numéro fixant le labyrinthe ou les bonus.
- **BFS** : parcours en largeur pour calculer les distances.
- **Cible** : case choisie par l'IA pour guider le déplacement.

15 Crédits

Projet réalisé par **KARAMI Anir** (BUT3APP TPC). Serveur et sujet fournis dans le cadre de la SAE.

16 Ressources

- Repo : https://github.com/anirkm/AI_peche_aux_moules
- Doc : https://anirkm.github.io/AI_peche_aux_moules/