Pauline Cerello S1A
William Tchang

Rapport - Projet Lowatem IA

Présentation de IA :

I. IA à calcul de différences de PVs

Nous nous sommes basés sur le code du niveau 6 de la phase 1 pour faire notre IA.

L'idée ici est de maintenir le plus gros écart supérieur de points de vie avec l'ennemi, afin de tenir le plus longtemps possible.

Pour cela notre IA parcourt toutes les actions possibles retournées par le code du niveau 6. Pour chacune de ces actions, elle va vérifier si c'est une action avec attaque. Si c'est le cas, elle va récupérer les points de vie du soldat ennemi qu'elle compte attaquer et celle de notre soldat qui va attaquer. Elle va ensuite faire la différence de nos points de vie avec ceux de l'ennemi. Cette différence est alors stockée. Pour le cas d'une action sans attaque, on fait la différence de points de vie totaux de nos soldats avec ceux de l'ennemi. On cherche ensuite la première action qui donne le plus gros score stocké et on joue celle-là.

Pour se faire, nous avons réutilisé quelques fonctions de la phase 1, telle que celle pour récupérer les points de vie d'un soldat et celle qui convertit les coordonnées de type de caractère à celles en numéro.

II. IA avec la différence de PV et amélioration, esquive

Cette seconde IA est une amélioration de la première, toujours basée sur le même code.

L'idée ici est de privilégier les actions avec attaques et d'éviter de se retrouver aligné avec un ennemi dans le cas d'une action de simple déplacement.

Pour cela nous faisons comme l'autre IA, c'est-à-dire comparer les pv des soldats attaquant et attaqué. Donc dans le cas où il y a des actions avec attaques, on fait ce calcul pour les actions d'attaque et pour les actions de simple déplacement, on stock un nombre très petit. De même que dans l'autre IA, on calcule le maximum de ces nombres stockés. Cependant cette fois-ci, on va pas récupérer non pas la première action avec cette différence mais toute celle avec ce maximum, ce qui nous fait un tableau d'actions uniquement d'attaques possibles. On vient ensuite prendre aléatoirement une action dedans.

Dans le cas où nous n'avons que des actions déplacements simples possibles, nous testons pour chaque action envisagée si la position d'arrivée de notre soldat serait sur une ligne ou une colonne comprenant un ennemi. Si c'est le cas, nous retirons cette action de notre tableau d'actions possibles. Enfin nous prenons aléatoirement une action parmi celles qui restent.

Pour se faire, nous avons réutilisé quelques fonctions de la phase 1, telle que celle pour récupérer les points de vie d'un soldat et celle qui convertit les coordonnées de type de caractère à celles en numéro. Ainsi que celle de parcourir toutes les directions et toutes les cases dans ces directions, celle qui vérifie si on est sur le plateau. Enfin, nous avons eu besoin aussi de la fonction qui vérifie si un soldat se trouve sur la case et celle-ci passe à la case suivante.

Choix de l'IA

Nous avons choisi de prendre la première IA, qui fait seulement un comparatif de points de vie. Malgré le fait que l'idée de la seconde soit plus intéressante, plus stratégique et plus approfondie mais il y a une erreur dans le code qui renvoie une erreur et donc, nous disqualifie. Malgré que la première IA n'ai pas beaucoup de tactique, elle arrive à gagner de temps en temps contre le guerrier, en choisissant l'action où on a le plus grand écart supérieur de points de vie avec l'ennemi. En comparant les simulations de nuits auxquelles elles ont participé chacune, la première IA est nettement mieux classée que la seconde, qui ne parvient pas à dépasser le tâcheron. Avec les simulations de jours contre tâcheron et guerrier nous avons pu, pour les rares fois où la seconde IA ne tombait pas sur une action erronée, constaté qu'elle perdait beaucoup plus de fois que l'autre IA.

Pauline Cerello
William Tchang S1A

Portfolio

Pauline :

Lors de la première phase, j'ai appris à retranscrire des règles d'un jeu en code. A utiliser le langage Java, faire des classes, des fonctions statiques ou non, des fonctions qui se servent de fonctions venant de différentes classes. Lors de la seconde phase, j'ai pu revoir ces notions et les consolidées. J'ai pu découvrir d'autres méthodes telles que Arrays.copyOf, charAt, split, Integer.parseInt. Ce qui m'a fait découvrir comment coder une IA.

William :

Dans la phase une, j'ai appris à mettre en application le tri d'un tableau à deux dimensions vue en cours.Et aussi cela m'a permis d'approfondir mes connaissances en java.Dans la 2nd phase du projet j'ai contribué à la création des IAs avec Pauline. J'ai appris à utiliser des méthodes comme split(), Integer.parseInt(). Ma contribution a été dans une IA, qui n'a pas été présentée dans le rapport et aussi dans la correction du code des autres IA.

```
Les 2 joueurs :
  Joueur n° 1 :
    rouge
    « Intelligence artificielle » Guerrier
    Perdant
    Points de vie sur le plateau final : 0
    Nombre de points : 75
  Joueur n° 2 :
    noir
    pcerello
    Gagnant
    Points de vie sur le plateau final : 6
    Nombre de points : 87
Historique : 1[1/2]gCDjCAkC(85) 2[2/2]hADcAAcB(16) 3[1/2]fNDfLAeL(18) 4[2/2]hKDfKAfL(96) 5[1/2]lJDlCAkC(16) 6
[2/2]jHDmHAmI(112) 7[1/2]lCDlHAmH(14) 8[2/2]kHDkHAlH(105) 9[1/2]mIDkIAkH(11) 10[2/2]fCDiCAjC(100) 11
[1/2]iFDiDAiC(5) 12[2/2]cADcCAcB(98) 13[1/2]bGDmGAmH(11) 14[2/2]cCDiCAiD(93) 15[1/2]mGDmIAlI(12) 16
[2/2]iCDiEAiD(93) 17[1/2]mIDkIAlI(6) 18[2/2]kHDkJAkI(91) 19[1/2]kIDkKAkJ(7) 20[2/2]eLDkLAkK(90) 21[1/2]aGDaE(4)
22[2/2]iEDbEAaE(95) 23[1/2]aEDjE(5) 24[2/2]kLDkEAjE(91) 25[1/2]jEDgE(4) 26[2/2]fKDfEAgE(92)
  Format d'un tour de jeu : <tour> [ <ordre joueur> ] <action> ( <durée en ms> ).
```