Compétition d'IA – BomberMan en simulation à événements discrets

laurent.jospin.59@free.fr, `http://jospin.lstl.fr`

Lycée Saint-Louis, Paris

L'objectif de ce travail est de construire une stratégie automatisée (souvent appelée intelligence artificielle dans le domaine des jeux) pour un jeu compétitif de type BomberMan : les joueurs se déplacent dans un labyrinthe organisé dans une grille et peuvent déposer des bombes qui explosent quelques instants plus tard.

1 Présentation du problème

Le jeu se joue usuellement par des déplacements quasi-continus dans le labyrinthe en actionnant les touches directionnelles du clavier puis en déposant une bombe sur une case de la grille produisant des déflagrations en croix détruisant ainsi les cases en bois et les concurrents.

Le personnage peut également ramasser des PowerUps – des bonus – lui permettant d'augmenter la longueur des déflagrations, le nombre de bombes qu'il peut disposer simultanément ou la vitesse de son personnage. Il peut aussi ramasser deux consommables : des dashs et des pièges. L'utilisation d'un Dash confère 3 actions sans attente. Le piège, une fois posé, bloque le prochain adversaire marchant sur la case.

Nous jouerons la variante suivant du jeu, appelée Conquête de territoire : chaque fois qu'une déflagration atteint une case vide du plateau, celle-ci est attribuée au joueur qui a posé la bombe. A la fin du temps, le joueur qui a revendiqué le plus de cases a gagné.

Les déplacements, normalement quasi-continus, ont été discrétisés par une méthode dite de simulation à événements discrets. Ainsi le jeu progresse de façon discrète au travers d'événements de différentes natures : explosion de bombe, propagation de la déflagration, ou tour d'un joueur. Plus la vitesse d'un joueur est importante plus l'événement correspondant à son tour de jeu se produit à intervalle rapproché.

2 Cahier des charges

Pour répondre au problème et entrer en concurrence dans le jeu, vous devrez programmer un programme autonome, en C ou en Caml, qui devra lire en entrée et dans cet ordre :

- l'instant actuel de jeu (sous la forme d'un nombre à virgule flottante)
- votre numéro de joueur (sur une ligne)
- les dimensions du labyrinthe (`ligne colonne` inscrit sur une ligne)
- la grille du labyrinthe constituée de `ligne` lignes, et `colonnes` nombre par ligne. Les nombres possibles sont 0 pour les cases vides non revendiquées, 1 pour les murs en pierre (indestructibles) et 2 pour les caisses en bois (destructibles), puis $3 + j$ pour les cases vides revendiquées par le joueur de numéro j
- le nombre de bombes actuellement en jeu (sur une ligne)
- pour chaque bombe actuellement en jeu, une ligne indiquant ses coordonnées, la taille des déflagrations et son instant d'explosion (`ligne colonne taille instant`) inscrit sur une ligne)
- le nombre de joueurs actuellement en jeu (sur une ligne)
- pour chaque joueur actuellement en jeu, une ligne indiquant ses coordonnées, son numéro et le nombre de powerups qu'il a actuellement (`ligne colonne joueur vitesse nbbombes deflagration nbdashs nbpieges`)
- le nombre de powerups actuellement en jeu (sur une ligne)
- pour chaque powerup actuellement en jeu, une ligne indiquant ses coordonnées et son type (`ligne colonne` **`type`**). Avec type qui vaut 0 pour Vitesse augmentée, 1 pour Nombre de bombes augmenté, et 2 pour Taille des déflagration augmentée.

Votre programmer devra afficher sur la sortie standard un couple de valeurs composé d'un entier donnant la direction dans laquelle vous souhaitez vous déplacer et d'un entier 0/1/2/3 indiquant si, avant le déplacement, vous souhaitez déposer ne rien faire de spécial, poser une bombe, utiliser un dash ou poser un piège (une seule des trois actions est possible par tour de jeu). Vous pouvez utiliser un fichier de sauvegarde ayant le même nom que votre programme (`argv[0]`) et l'extension `.sav` si c'est utile.

3 Evaluation des stratégies

Les stratégies seront évaluées au cours de compétitions, chaque joueur recevant ou perdant des points en fonction de l'issu du combat et de son écart de points par rapport aux concurrents, sur la base du classement ELO en particulier utilisé aux échecs.

Remarque. La simulation à événements discrets peut évidemment être mise en oeuvre pour programmer d'autres jeux tels que PacMan, elle peut aussi être utilisée pour simuler un billard par exemple.