BOUM ! BOUM ! BOUM !!!!

Sujet du concours ProgCnet du samedi 10 mars 2007 (v0.2)

Voilà, le grand jour du concours est enfin arrivé !

L'IA que vous allez devoir réaliser est destinée à jouer à un jeu très connu … LE BOMBERMAN !
Quoi ? Vous ne connaissez pas bomberman ? Je suis sympa je vais vous faire un petit topo.
Chaque joueur est un bomberman qui doit tuer les autres en posant des bombes.
Chaque bombe provoque une explosion qui s'étend sur ligne et sur la colonne sur laquelle elle est placée.
La map est constituée de murs indestructibles, de briques qui sont détruites par les explosions des bombes et qui donnent des bonus (parfois), de bonus, et bien sur de d'autres joueurs qui essaieront de cramer votre IA.

Comment allez vous coder ?
Chaque IA est un module (appelé aussi greffon, l'équivalent des dll sous windows).
C'est ceci que vous allez devoir programmer. Ensuite le programme principal va devoir les charger pour permettre les matchs.
Vous disposez :
- du programme de match (une copie locale sur chaque machine, vous permettant de tester au fur et à mesure),
- d'une GUI en SDL sous forme de module aussi,
- d'une interface web afin d'uploader vos IAs pour les tester contre celles de d'autres participants,
- de l'API complète permettant d'avoir toutes les informations nécessaires et en cadeau : le pathfinding est déjà implémenté,
- de très peu de temps ! En effet vers 19h30 votre code doit être uploadé sur le serveur web en cochant la case « IA finale » (sinon elle ne sera pas prise en compte pendant la phase de classement final).

N'oubliez pas : l'important n'est pas de gagner ! Mais d'écraser ses adversaires le plus brutalement possible =)

Les règles

Règles générales :

Lorsque vous posez une bombe elle est sur la case de votre IA.
Deux IAs ne peuvent être sur la même case au même moment.
Vous ne pouvez pas passer sur une bombe, elle constitue un obstacle.
Deux bombes ne peuvent être sur la même case.
Il ne peut pas non plus y avoir deux bonus sur la même case.
Vous ne pouvez au départ faire que 3 mouvements, poser 1 bombe dont l'explosion s'étend sur 2 cases vers gauche, 2 cases vers droite, 2 cases vers le haut et 2 cases vers le bas depuis la position de la bombe.

Une bombe explose au bout de 3 tours, l'explosion reste quant à elle pendant 1 tour.
Si la flamme d'une explosion touche une autre bombe elle explosera pendant le même tour.
Les explosions se font à la fin du tour, donc après les mouvements de chaque IA.
Si une IA se met sur la case de la flamme d'une explosion l'IA est morte, l'IA disparaît du jeu mais la flamme reste.
Si une flamme touche une brique, la case de la brique est changée soit en bonus soit en flamme.

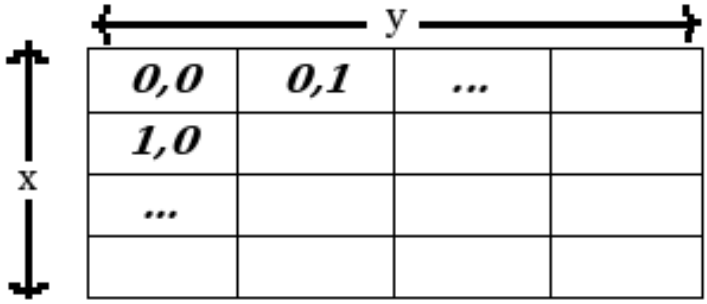
Les restrictions :

Lors de la phase finale le serveur va effectuer des centaines de matchs tous limités à 1500 tours.
Vous ne disposez que de 2ms par tour.

La map

Représentation :

La map est un tableau à 2 dimensions organisés ainsi :
x : la ligne,
y : la colonne.



Les constantes de la map :

JOUEUR
Une case contenant une IA a la valeur : JOUEUR + id de l'IA.

RIEN
Indique que le chemin est libre, il n'y a rien sur la case.

MUR
C'est un mur indestructible.

BRIQUE
La brique est destructible (par l'explosion d'une bombe uniquement) et peut donner un bonus lors de sa destruction. En tant que telle vous ne pouvez pas passer dessus, c'est un obstacle.

MOTION_BONUS
Bonus qui permet d'augmenter le nombre de déplacements par tour.

BOMBE_BONUS
Bonus qui permet d'augmenter le nombre de bombes posables par tour.

FLAMME_BONUS
Bonus qui permet d'augmenter la propagation des flammes d'une explosion.

FLAMME
Il y a une flamme d'une explosion sur la case.

BOMBE
Il y a une bombe posée par un joueur sur la case.

BOMBE_JOUEUR
Il y a une bombe et le joueur qui l'a posé sur la même case. La valeur de la case est BOMBE_JOUEUR + id de l'IA.

(note : id = identifiant, c'est un nombre entier non signé entre 0 et le nombre de joueurs lors du match).

La copie locale du serveur

L'utilisation est très simple :
Placez vous dans le répertoire du programme : cd ~/boum
Pour faire un match : ./boum (--gui) map nombre_tours ./votre_ia1.so ./votre_ia2.so ...
Si vous mettez --gui la GUI en SDL sera alors chargée (appuyez sur espace pour passer au tour du joueur suivant).
Map est le nom du fichier de la map.
nombre_tours est le nombre de tours que vous voulez pour le match.
Chaque fichier d'IA doit être indiqué avec le path, si le fichier est dans le répertoire du serveur indiquez bien ./ devant le nom.

ATTENTION : vous ne devez pas charger plusieurs fois le même fichier car sinon ils partageront le même espace mémoire (et donc les mêmes variables globales etc), vous devez copier votre .so pour lancer un match avec plusieurs fois la même IA !

Sortie du server :

Erreur : bla bla
Lisez l'erreur, normalement c'est explicite.

Victoire : id
id de l'IA qui a gagné, le premier fichier .so en entrée à l'id 0, le deuxième l'id 1 etc...

Draw : tour
Personne n'a gagné, tour correspond au tour où le match s'est arrêté.

Segfault : id,tour
id de l'IA qui a provoqué un « segmentation fault » et à quel tour. Le match n'est pas arrêté mais le joueur est considéré mort.

Timeout : id,tour
id de l'IA qui a dépassé la limite des 2ms et à quel tour. Le match n'est pas arrêté mais le joueur est considéré mort.

Frozen : id,tour
id de l'IA qui a dépassé 1s pendant le tour, et à quel tour. Le match n'est pas arrêté mais le joueur est considéré mort.

Les APIs

Map et informations

int largeur_map (void);
Description : permet de connaître la largeur de la map.
Retour : largeur de la map.

int hauteur_map (void);
Description : permet de connaître la hauteur de la map.
Retour : hauteur de la map.

int ya_quoi (int x, int y);
Description : renvoie la valeur d'une case de coordonnées x,y de la map selon les constantes définies au dessus.
Paramètres : x : coordonnée en x de la case demandée.
y : coordonnée en y de la case demandée.
Retour : la valeur de la case en coordonnées x,y de la map.

int nb_bombe_bonus (void);
Description : permet de connaître combien il y a de bonus de bombe placés sur la map au moment de l'appel.
Retour : le nombre de bonus de bombe.

int nb_flamme_bonus (void);
Description : permet de connaître combien il y a de bonus de longueur de flamme placés sur la map au moment de l'appel.
Retour : le nombre de bonus de longueur de flamme.

int nb_motion_bonus (void);
Description : permet de connaître combien il y a de bonus de déplacement placés sur la map au moment de l'appel.
Retour : le nombre de bonus de déplacement.

Déplacements

int aller_gauche (void);
Description : effectue un déplacement vers la gauche à votre IA.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
PLUS_DE_MOUVEMENT vous avez déjà bougé au maximum,
JOUEUR_MORT votre joueur était déjà mort ou vient de mourir,
ERREUR_TERRAIN la case n'est pas accessible (mur, bombe, joueur),
NP le déplacement a bien été fait.

int aller_droite (void);
Description : effectue un déplacement vers la droite à votre IA.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
PLUS_DE_MOUVEMENT vous avez déjà bougé au maximum,
JOUEUR_MORT votre joueur était déjà mort ou vient de mourir,
ERREUR_TERRAIN la case n'est pas accessible (mur, bombe, joueur),
NP le déplacement a bien été fait.

int aller_haut (void);
Description : effectue un déplacement vers le haut à votre IA.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
PLUS_DE_MOUVEMENT vous avez déjà bougé au maximum,
JOUEUR_MORT votre joueur était déjà mort ou vient de mourir,
ERREUR_TERRAIN la case n'est pas accessible (mur, bombe, joueur),
NP le déplacement a bien été fait.

int aller_bas (void);
Description : effectue un déplacement vers le bas à votre IA.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
PLUS_DE_MOUVEMENT vous avez déjà bougé au maximum,
JOUEUR_MORT votre joueur était déjà mort ou vient de mourir,
ERREUR_TERRAIN la case n'est pas accessible (mur, bombe, joueur),
NP le déplacement a bien été fait.

int aller_xy (int x, int y);
Description : effectue un déplacement vers la case de coordonnées x,y en passant par le chemin le plus court et sans passer sur les flammes des explosions.
Paramètres : coordonnée en x de la case de destination,
coordonnée en y de la case de destination.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
PLUS_DE_MOUVEMENT vous avez déjà bougé au maximum,
ERREUR_COORD les coordonnées sont en dehors de la map,
JOUEUR_MORT votre joueur est mort,
MOUVEMENT_IMPOSSIBLE vous êtes déjà sur la case demandée,
ERREUR_TERRAIN la case n'est pas accessible (mur, bombe, joueur),
PAS_DE_CHEMIN il n'y a aucun chemin pour accéder à la case demandée,
sinon le nombre total de cases pour y arriver est renvoyé.

int distance (int x, int y);
Description : permet de connaître la distance entre votre joueur et une case en passant par le chemin le plus court.
Paramètres : coordonnée en x de la case de destination,
coordonnée en y de la case de destination.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
ERREUR_COORD les coordonnées sont en dehors de la map,
JOUEUR_MORT votre joueur est mort,
ERREUR_TERRAIN la case n'est pas accessible (mur, bombe, joueur),
PAS_DE_CHEMIN il n'y a aucun chemin pour accéder à la case demandée,
sinon le nombre total de cases pour y arriver est renvoyé.

Informations sur le match

int nb_total_joueurs (void);
Description : permet de connaître le nombre total de joueur au début du match.
Retour : le nombre total de joueurs qu'il y avait au début du match.

int nb_joueurs_lice (void);
Description : permet de connaître le nombre de joueurs encore en vie.
Retour : le nombre de joueurs encore en vie.

int nb_total_tours (void);
Description : permet de connaître le nombre total de tours du match.
Retour : le nombre de tours du match.
Note : les matchs lancés sur le serveur sont de 1500 tours, lors du classement final les matchs seront aussi de 1500 tours.

int tour_en_cours (void);
Description : permet de connaître combien de tours se sont écoulés depuis le début du match.
Retour : le tour actuel.

Informations sur les joueurs

int nb_bombes_total (int id);
Description : permet de connaître le nombre de bombes au total dont dispose un joueur.
Paramètre : l'id du joueur.
Retour : nombre de bombes total du joueur.

int nb_bombes_dispo (int id);
Description : permet de connaître le nombre de bombes que peut encore poser le joueur.
Paramètre : l'id du joueur.
Retour : nombre de bombes que peut poser le joueur.

int mon_nb_bombes_total (void);
Description : permet de connaître le nombre de bombes total dont vous disposez.
Retour : nombre de bombes que peut poser le joueur.

int mon_nb_bombes_dispo (void);
Description : permet de connaître le nombre de bombes que vous pouvez encore poser
Retour : nombre de bombes que vous pouvez encore poser.

int position_joueur (int id, int *x, int *y);
Description : permet de connaître la position en x,y d'un joueur selon son id.
Paramètres : l'id du joueur,
un pointeur pour la coordonnée en x,
un pointeur pour la coordonnée en y.
Retour :
PAS_DE_JOUEUR il n'y a pas de joueur avec l'id demandé,
JOUEUR_MORT le joueur est mort et donc n'est plus sur la map,
NP les coordonnées sont mises dans x et y.

int position_x_joueur (int id);
Description : permet de connaître la position en x d'un joueur selon son id.
Paramètre : l'id du joueur.
Retour :
PAS_DE_JOUEUR il n'y a pas de joueur avec l'id demandé,
JOUEUR_MORT le joueur est mort et donc n'est plus sur la map,
sinon la position en x du joueur est renvoyé.

int position_y_joueur (int id);
Description : permet de connaître la position en y d'un joueur selon son id.
Paramètre : l'id du joueur.
Retour :
PAS_DE_JOUEUR il n'y a pas de joueur avec l'id demandé,
JOUEUR_MORT le joueur est mort et donc n'est plus sur la map,
sinon la position en y du joueur est renvoyé.

int ma_position (int *x, int *y);
Description : permet de connaître votre position en x,y.
Paramètres : un pointeur pour la coordonnée en x,
un pointeur pour la coordonnée en y.
Retour :
JOUEUR_MORT vous êtes mort et donc vous n'êtes plus sur la map,
NP les coordonnées sont mises dans x et y.

int ma_position_x (void);
Description : permet de connaître votre position en x.
Retour :
JOUEUR_MORT vous êtes mort et donc vous n'êtes plus sur la map,
sinon votre position en x est renvoyée.

int ma_position_y (void);
Description : permet de connaître votre position en y.
Retour :
JOUEUR_MORT vous êtes mort et donc vous n'êtes plus sur la map,
sinon votre position en y est renvoyée.

int mon_id (void);
Description : permet de connaître votre id.
Retour :
JOUEUR_MORT vous êtes mort,
sinon votre id est renvoyé.

int mvt_total (void);
Description : permet de connaître le nombre de déplacements maximum disponibles au début d'un tour.
Retour :
JOUEUR_MORT vous êtes mort,
sinon le nombre de déplacements que vous pouvez faire au début du tour est renvoyé.

int mvt_dispo (void);
Description : permet de connaître le nombre de déplacements que vous pouvez encore faire.
Retour :
JOUEUR_MORT vous êtes mort,
sinon le nombre de déplacements que vous pouvez encore faire est renvoyé.

int status_joueur (int id);
Description : permet de savoir si un joueur est mort ou vivant selon son id.
Paramètre : l'id du joueur.
Retour :
PAS_DE_JOUEUR il n'y a pas de joueur avec cet id,
JOUEUR_MORT le joueur est mort,
JOUEUR_VIVANT le joueur est vivant.

int ma_puissance (void);
Description : permet de connaître la longueur des flammes de l'explosion des bombes que vous poserez.
Retour :
JOUEUR_MORT vous êtes mort,
sinon la longueur de la flamme qui s'étend de puissance/2 vers le haut, bas, gauche, droite depuis la case de la bombe est renvoyée.

int puissance_joueur (int id);
Description : permet de connaître la longueur des flammes de l'explosion des bombes d'un joueur.
Paramètre : l'id du joueur.
Retour :
JOUEUR_MORT le joueur est mort,
PAS_DE_JOUEUR il n'y a pas de joueur avec l'id demandé,
sinon la longueur de la flamme qui s'étend de puissance/2 vers le haut, bas, gauche, droite depuis la case de la bombe est renvoyée.

Les bombes

int puissance_bombe (int x, int y);
Description : permet de connaître la longueur des flammes de l'explosion d'une bombe en coordonnées x,y.
Paramètres : coordonnée en x de la bombe,
coordonnée en y de la bombe.
Retour :
PAS_DE_BOMBE il n'y a pas de bombe aux coordonnées x,y demandées,
sinon la longueur de la flamme qui s'étend de puissance/2 vers le haut, bas, gauche, droite depuis la case de la bombe est renvoyée.

int temps_restant_bombe (int x, int y);
Description : permet de connaître le nombre de tours qu'il reste avant l'explosion d'une bombe en coordonnées x,y.
Paramètres : coordonnée en x de la bombe,
coordonnée en y de la bombe.
Retour :
PAS_DE_BOMBE il n'y a pas de bombe aux coordonnées x,y demandées,

sinon le nombre de tours est renvoyé.

int joueur_bombe (int x, int y);
Description : permet de connaître l'id du poseur de la bombe en coordonnées x,y.
Paramètres : coordonnée en x de la bombe,
coordonnée en y de la bombe.
Retour :
PAS_DE_BOMBE il n'y a pas de bombe aux coordonnées x,y demandées,
sinon l'id du joueur est renvoyé.

int poser_bombe (void);
Description : vous permet de poser une bombe sur la case de votre joueur.
Retour :
PHASE_INIT si vous appelez la fonction dans votre fonction d'initialisation,
JOUEUR_MORT votre joueur est mort,
PAS_DE_BOMBE vous n'avez plus de bombe disponible,
DEJA_UNE_BOMBE vous avez déjà posé une bombe sur cette case,
NP la bombe est posée.