GCB 202

Projet : BING is not Galcon

## Objectif

Le but du jeu est de descendre le score de son adversaire à 0. Le score correspond à la force totale du joueur. La force d’un joueur peut se retrouver soit sur un ensemble de planètes et de vaisseaux. Le jeu se joue avec un script (bot) que le joueur code et soumet au jeu. Le jeu se joue un contre un.

## Méthode

Pour accomplir son objectif, un joueur peut (doit) conquérir les planètes de son adversaire. Pour conquérir une planète, il suffit d’envoyer un vaisseau possédant une force plus grande que la force de la planète attaquée.

### Factions

Il y a trois factions dans le jeu, soit le joueur 1, le joueur 2 et le joueur neutre. Les différentes factions sont discernables par leur couleur respective. Les actions des joueurs 1 et 2 proviennent de deux scripts-joueurs différents. Le joueur neutre lui n’a aucune action possible.

### Lancement d’attaque

Pour attaquer une planète ennemie, un joueur sélectionne un certain nombre de forces d’une planète et l’envoie vers une planète ennemie. Cela crée un vaisseau qui se déplace vers la planète ennemie jusqu’à ce qu’il y arrive et complète l’attaque.

### Résolution d’attaque

Lorsqu’il y a une attaque, soit un vaisseau arrivant à une planète ennemie, le combat se fait simplement 1 pour 1. Par exemple, si une planète de force 10 se fait attaquer par un vaisseau de force 7, il restera donc 3 (10 - 7) de force sur la planète.

### Capture d’une planète

Si, après une attaque, une planète possède un nombre négatif de force, elle change de leadership (propriétaire). Par exemple si une planète de force 7 se fait attaquer par un vaisseau de force 10, la population devient -3 (7 - 10). Ayant un nombre négatif, la planète change de propriétaire (deviens donc la propriété du joueur possédant le vaisseau qui a attaqué) et on prend la valeur absolue de la force, donc 3.

### Attaque simultanée

Si deux vaisseaux ennemis arrivent à une planète en même temps, les deux vaisseaux entrent en combat en orbite avant d’interagir avec la planète. Ainsi la valeur du vaisseau avec la force la plus faible est soustraite de la valeur du vaisseau avec la force la plus forte. Si les deux vaisseaux sont de force identique, les deux vaisseaux disparaissent et la planète n’est pas affectée.

### Défendre une planète

Lorsqu’une attaque est imminente, il est possible de déployer un bouclier afin d’aider la défense d’une planète. Un bouclier coûte de la force à générer et ne dure qu’un seul tour, mais il peut bloquer deux fois le nombre de force utilisé pour le créer. Ainsi, si un vaisseau de force 10 attaque une planète de force 8, celle-ci peut générer un bouclier en utilisant 5 forces qui bloqueront le vaisseau (2 x 5 est supérieur à 8). Après l’utilisation de ce bouclier pendant un tour de jeu, les 5 forces utilisées disparaîtront et il restera (8-5) 3 forces sur la planète.

### Au renfort!

Si un joueur voit qu’une de ses planètes est en danger, il peut décider d’envoyer des vaisseaux en renfort, procédant comme pour déclarer une attaque, mais en visant une de ses propres planètes. Lorsque le vaisseau arrivera à l’autre planète, la force du vaisseau sera simplement additionnée au reste de la force de la planète.

### Se retrouver dans la galaxie

Le positionnement dans la galaxie (la carte) se fait avec des nombres entiers. La position (0, 0) se trouve en haut à gauche de la carte. Si un positionnement est un nombre réel, la position réelle sera arrondie à l’entier le plus proche. Par exemple, si vos calculs de déplacement du vaisseau disent qu’il arrivera à la position (154.12, 274.75), il sera réellement à la position (154, 275). Le premier index de la position correspond à l’axe des X, qui est situé sur l’horizontal. Le second index correspond aux Y, qui est sur l’axe vertical.

### Repeuplement des planètes

Puisque les confrontations spatiales sont des événements de longue haleine, les planètes ont le temps de se générer des forces. La fréquence de génération dépend de la taille de la planète.

## Déroulement d’un tour de jeu

Les différentes actions sont séparées en tour de jeu. Les joueurs sont toujours informés du numéro du tour en cours, comme indiqué plus bas. Voici la séquence des actions dans un tour :

1. Les joueurs sont appelés pour leur tour de jeu. Cela veut dire que les fonctions makeOrder de chaque joueur sont appelées à ce moment.
2. Les planètes génèrent de la force s’il y a lieu.
3. Boucle de calcul. Cette boucle est effectuée cinq fois par tour.
   1. Les attaques possibles sont résolues
   2. Le score des joueurs est comptabilisé
   3. Les vaisseaux se déplacent vers leur cible respective
   4. Les conditions de victoire sont vérifiées.

## Les constantes du jeu

Différentes constantes régissent la fréquence et rapidité des actions.

### Temps de génération

Les planètes des joueurs 1 et 2 génèrent une force tous les (12 / planetSize) tours. Les planètes du joueur neutre génèrent ces forces trois fois plus lentement.

### Vitesse des vaisseaux

À chaque tour les vaisseaux se déplacent en calculant la norme du vecteur de déplacement, où les deux dimensions sont la distance en X et la distance en Y entre le vaisseau et sa cible. Les deux composantes (X et Y) sont ensuite divisées par la norme, puis multipliées par 8.

### Distance d’attaque

Les attaques sont résolues si la distance entre le centre de la planète visée et le vaisseau est inférieure à 9. La distance étant la norme des vecteurs de déplacement en X et en Y.

## Interaction entre les joueurs et le jeu

Les joueurs vont être appelés par le jeu par l’appel d’une fonction. Chaque joueur doit avoir une fonction nommée makeOrder comme la suivante dans son script-joueur :

def makeOrder(planets, ships, turn, playerId):  
 ...  
 return orders

Les quatre arguments de la fonction sur les informations qui seront passées au joueur afin qu’il puisse prendre des décisions à chaque tour de jeu. orders est une liste des ordres que le joueur passe au jeu afin de les exécuter.

### Ordres d’attaque

Chaque ordre est une liste composée de trois éléments, soit planet.planetName de la planète de laquelle les vaisseaux partent, planet.planetName de la planète cible et le nombre de forces (nombre entier).

Ainsi un ordre pourrait être : ['P1', 'P4', 12], et la réponse orders d’un script-joueur pourrait être :

[['P1', 'P4', 12]]

Dans le cas où il n’y a qu’une seule attaque, ou :

[['P1', 'P4', 12], ['P1', 'P3', 12]]

Dans le cas où il y a deux attaques lancées dans le même tour.

### Ordre de bouclier

Un ordre de bouclier est aussi une liste de trois éléments qui doit être rajouté à la liste orders. Le premier élément de la liste doit être un string 'Shield', le second élément est le nom de la planète et le troisième élément est un nombre entier de la force utilisée pour générer le bouclier. Ainsi un ordre ressemblerait à :

['Shield', 'P4', 12]

### Amélioration de planète

Un ordre d’amélioration de planète permet de faire grossir sa planète d’une taille (planet.planetSize). La taille d’une planète est corrélée avec son taux de production comme vue dans la section des constantes. Une planète ne peut pas être de taille supérieure à 4. Pour coût d’amélioration est de 6 fois la taille de planète actuelle. Donc pour une planète de taille 2 (planet.planetSize == 2), il en coûte 12 forces. L’ordre d’amélioration prend trois termes. Le premier est un string 'Upgrade', le second est le nom de la planète et le troisième est le nombre nécessaire de force investi. Par exemple, si je voulais améliorer ma planète 'P4', qui est présentement de niveau 3 je passerais l’ordre suivant :

['Upgrade', 'P4', 18]

### return orders

Ainsi, à chaque fois que la fonction makeOrder d’un joueur est appelée, il doit retourner une liste de listes, orders, qui doit avoir de zéro à plusieurs éléments d’ordres d’attaques et de boucliers, comme dans l’exemple suivant :

[['P4', 'P7', 12], ['P1', 'P3', 12], ['Shield', 'P4', 12]]

## Informations disponibles pour les joueurs

Les informations disponibles sont passées à travers les quatre variables vues à la section précédente, soit planets, ships, turn et playerId.

planets Contient une liste des planètes et de certaines informations relatives. Les planètes sont des objets.

ships Contient une liste des vaisseaux actifs. Les vaisseaux sont des objets.

turn Contient uniquement un entier qui correspond au numéro du tour en cours.

playerId Contient le numéro unique du joueur (1 ou 2).

### planets

Les planets sont des objets. Cela veut dire que l’on peut accéder à leur caractéristique avec une notation particulière comme suivant :

planet.planetName Contient le nom de la planète dans un string. Les planètes sont numérotées de 0 à N, précédé de la lettre P. Par exemple la planète numéro 4 contient le string suivant : 'P4'.

planet.planetLocation Contient la position de la planète dans la galaxie. L’information est contenue sous la forme d’une liste. Le premier index possède la position en X et la seconde la position en Y.

planet.planetSize Contient la taille de la planète sous la forme d’un nombre entier. Il y a trois possibilités de grosseur de planète, soit de 1 à 4 inclusivement.

planet.planetOwner Contient le numéro du joueur sous la forme d’un nombre entier. Le joueur neutre possède toujours le 0 et les deux autres joueurs sont 1 et 2.

planet.planetNum Contient la force de la planète sous la forme d’un entier.

### ships

Les ships sont aussi des objets. Cela veut dire que l’on peut accéder à leur caractéristique avec une notation particulière comme suivant :

ship.shipLocation Contient la position actuelle du vaisseau dans la galaxie. L’information est contenue sous la forme d’une liste. Comme pour les planètes, le premier index indique la position en X et le second, en Y.

ship.shipDestination Contient la position de la destination du vaisseau, sous la forme d’une liste avec position en X et en Y comme les autres positionnements.

ship.shipDestinationID Contient le nom de la planète de destination du vaisseau, sous la même forme que planet.planetName.

ship.shipOwner Contient le numéro de joueur du propriétaire du vaisseau. Cette valeur peut être 1 ou 2.

ship.shipNum Contient la force du vaisseau sous la forme d’un nombre entier.

## Pour jouer

Pour jouer au jeu, il faut s’assurer d’avoir complété les étapes suivantes :

### Configuration

Afin de pouvoir faire rouler le jeu, il faut avoir Python 3 avec Pygame installé sur son ordinateur. Pour les personnes travaillant sur Windows avec Anaconda3 et sur les ordinateurs disponibles à l’université, vous pouvez utiliser le fichier installPygame.bat pour faire l’installation de Pygame à l’aide de conda.

### Lancer le jeu

Pour lancer le jeu, on peut lancer la commande suivante dans un terminal (cmd) :

python mainServer.py nomBot1 nomBot2

Pour cela il faut mettre deux scripts .py de joueur dans le dossier bots du jeu et utiliser le terminal dans le dossier principal. On peut aussi utiliser le fichier startGamePrompt.bat pour commencer le jeu en demande au joueur d’entrer le nom des deux bots, soit le nom des deux fichiers.

On peut aussi utiliser le fichier startGame.bat en changeant le nom des bots dans le fichier.