Défi d'intelligence artificielle - Saint Louis' Arena

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

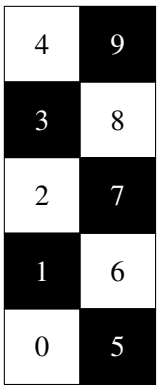
Lycée Saint-Louis, Paris

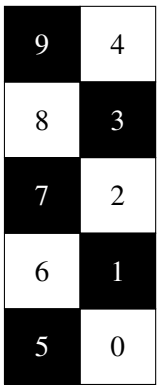
Ce jeu est librement et fortement inspiré du jeu Disney Sorcerer's Arena.

1 Présentation du jeu

1.1 Présentation générale

L'objectif de ce sujet est de construire une intelligence artificielle (au sens des jeux vidéos, c'est-à-dire une stratégie automatisée) pour un jeu de type RPG où deux joueurs s'affrontent en utilisant une équipes de personnages. Chaque joueur effectue une sélection de 5 personnages qu'il place sur les dalles noires d'un damier de 5 lignes et 2 colonnes. Les programmes écrits doivent répondre à deux tâches, la première dite de *draft* va consister à constituer une équipe parmi les personnages proposés aux joueurs, la seconde dite de *combat* va consister à choisir la capacité à utiliser pour le personnage dont c'est le tour.





1.2 Base de données

Les personnages disposent de différentes statistiques (vie, défense, taux d'esquive, vitesse etc.) ainsi que de trois capacités au maximum. Lorsque c'est le tour de jeu d'un personnage d'un joueur, le joueur doit activer l'une des capacités disponibles du personnage. La capacité produit alors des effets sur tout ou partie des personnages amis ou ennemis. Une fois utilisée, la capacité met un certain nombre de tours à se recharger. La mécanique complète du jeu sera détaillée dans la partie suivante.

Pour le fonctionnement du jeu, on a construit une base de données organisée de la façon suivante :

La table *personnages* répertorie la liste des personnages dans laquelle pid est un numéro unique identifiant les personnages.

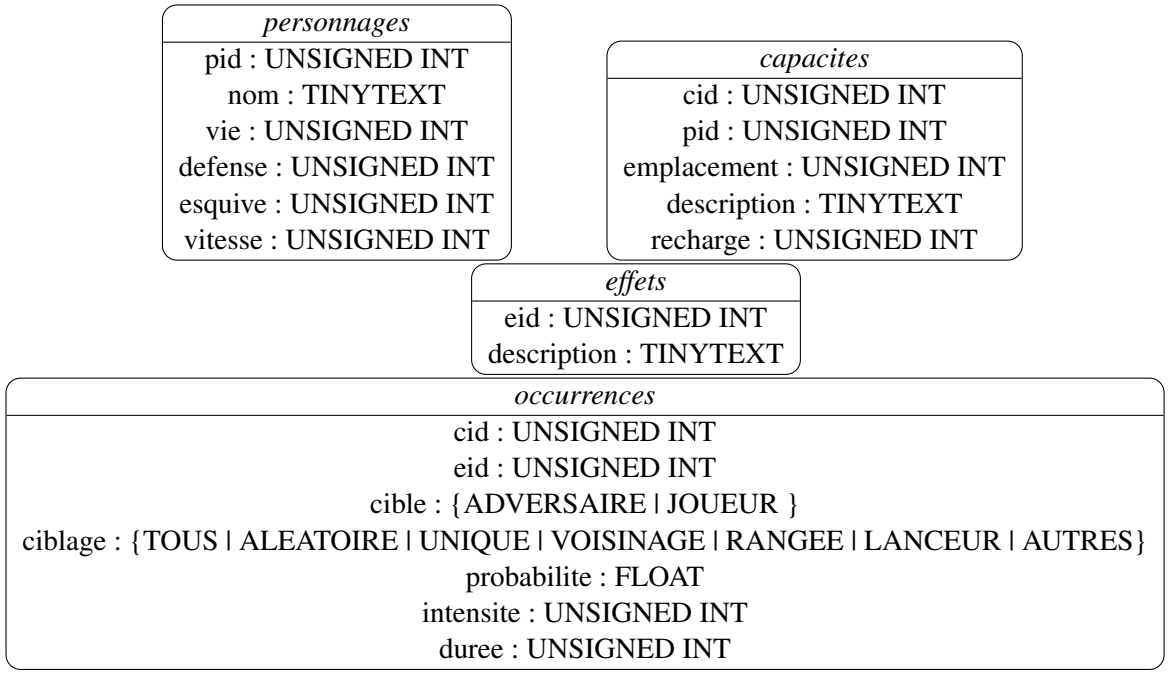
Les capacités sont répertoriées dans la table *capacites*, elles disposent d'un numéro unique cid mais contiennent également le numéro pid du personnage qui en dispose et l'emplacement de la capacité dans la liste des capacités du personnage : un numéro entre 0 et 2. *recharge* indique le nombre de tour qu'il faut attendre avant de pouvoir réutiliser une capacité.

La table *effets* répertorie l'ensemble des effets positifs et négatifs que peuvent déclencher les capacités (soins, attaques, affaiblissements divers, améliorations diverses...). Dans la base de données on a simplement incorporé une description de l'effet, la mécanique étant programmée dans le jeu.

La table *occurrences* indique quel(s) effet(s) peuvent se déclencher lorsqu'on utilise une capacité. Un même effet ne peut apparaître qu'une fois pour une capacité donnée. Les champs cible et ciblage ont un type spécifique, dit énumération, qui peuvent prendre comme valeur uniquement l'une des valeurs énumérées.

Supposons par exemple définit la capacité "Ton monotone" (cid = 1) qui cible un adversaire et qui déclenche l'effet "Endormi" (eid = 1) et peut déclencher avec une probabilité de 0.3 l'effet "Défense affaiblie" (eid = 2) sur la rangée de l'adversaire. Alors la table *occurrences* contient entre autres les entrées :

cid	eid	cible	ciblage	probabilite	intensite
1	1	ADVERSAIRE	UNIQUE	1.	100
1	2	ADVERSAIRE	RANGEE	0.3	100



1.3 Règles de combat

Le moteur de jeu constitue la liste des personnages du jeu et remplit leur jauge de combat en fonction de leur statistique de vitesse. Le personnage dont la jauge est pleine en premier commence le combat. Une fois son tour de jeu effectué, les jauges se remplissent de nouveau en fonction des statistiques de vitesse jusqu'à ce qu'une jauge soit pleine et ainsi de suite.

Les capacités des personnages peuvent produire différents effets positifs sur les personnages alliés :

- Soin direct : rétablit une partie de la vie des personnage touchés dans la limite de leur vie initiale
- Soin au fil du temps : rétablit une partie de la vie des personnages touchés au début de leur tour de jeu
- Vitesse augmentée : augmente la vitesse (de remplissage de la jauge) des personnages touchés (multipliée par 2)
- Attaque/défense augmentée : augmente la puissance des attaques/la défense des personnages touchés (multipliée par 2)
- Provocation : les ennemis ne peuvent cibler qu'un personnage ayant provocation lorsqu'il y en a.
- Tactique : quand le personnage touché utilise une capacité, toutes les occurences possibles deviennent certaines
- Esquive garantie : le personnage touché obtient une esquive certaine pour la prochaine attaque qu'il aurait dû subir
- Assistance : les personnages touchés effectuent immédiatement leur capacité 0 (sans que cela n'altère leur jauge de combat)

et effets négatifs sur les personnages ennemis :

- Dégâts directs : inflige des dégâts qui font baisser la vie des personnages touchés, ces dégâts peuvent être esquivés
- Dégâts au fil du temps : inflige des dégâts au début du tour des personnages touchés
- Endormi : les personnages touchés passent leur prochain tour. Si un personnage endormi reçoit des dégats directs, il est automatiquement réveillé et ne passera donc pas son prochain tour
- Sonné : les personnages touchés passent leur prochain tour.
- Attaque/vitesse/défense réduite : diminue la statistiques correspondante des personnages touchés. (multipliée par 0.5)

En dehors des effets directs (et de l'esquive garantie qui disparaît quand elle est utilisée), les effets persiste un ou deux tours suivant leur durée.

Le calcul des dégâts réellement subis par les dégâts directs tient compte de la défense du personnage touché : on ôte la défense du montant des dégâts à infliger, le résultat est ramené à 1 s'il lui est inférieur.

Chaque effet d'une capacité peut cibler tous les personnages (de l'équipe du joueur ou de l'équipe adverse), des cibles aléatoires, la cible uniquement, la cible et son voisinnage (y compris en diagonales), la cible et sa rangée (rangée de devant ou de derrière), le lanceur, ou tous sauf le lanceur.

1.4 Test du jeu

Une version permettant de tester le jeu est proposé dans le fichier `stlademo.py`, elle nécessite cependant l'installation du module `pygame` (en tapant `pip install pygame` dans une console système). Elle vise uniquement à tester le jeu pour mieux en comprendre la mécanique.

2 Cahier des charges

2.1 Conseils

Pour mener à bien ce projet, vous devez commencer par vous fixer des objectifs raisonnables, à commencer par produire quelque chose qui fonctionne même s'il n'y a aucune stratégie sous-jacente. Vous pouvez tester vos programmes en utilisant le fichier `stlagraphique.py` qui utilise le module `pygame` ou `stlaconsole.py` qui affiche uniquement l'historique du combat dans la console. Dans les deux cas, il faut importer deux intelligences artificiellles puis les passer en argument à la fonction `lance_jeu`. Vous pouvez utiliser deux fois la même intelligence artificelle ou en comparer deux différentes.

2.2 Phase de draft

Pour chacun des 5 personnages à choisir, le moteur de jeu propose deux choix (les mêmes) aux deux joueurs qui doivent en retenir un. Les choix sont proposés sous la forme d'une liste de 5 listes de 2 numéros de personnage qui est passée en argument à votre fonction `draft` et qui doit renvoyer une liste des 5 numéros de personnage choisis. L'ordre des numéros de personnage ne doit pas nécessairement être le même que celui des 5 choix et déterminera l'emplacement des différents personnages (dans l'ordre, les personnages seront placés aux emplacements 1, 3, 5, 7 et 9.

Exemple. Votre fonction `draft` reçoit en arguments la liste : `[[1, 3], [2, 4], [5, 9], [6, 7], [8, 10]]`, vous pouvez décider de prendre les personnages 3, 4, 9, 6, 8 et les placer dans un ordre librement choisi en renvoyant par exemple `[8, 3, 6, 9, 4]`.

2.3 Phase de combat

Lorsque c'est le tour de jeu d'un de vos personnages, la fonction `tour_de_jeu` de votre programme est appelée par le moteur de jeu avec pour argument l'état du jeu et le memo du tour précédent (None initialement). Votre fonction doit renvoyer la cible dans l'équipe adverse (un numéro d'emplacement), la cible dans l'équipe alliée, et le numéro de la capacité (0, 1 ou 2) à utiliser ainsi que le nouveau memo.

Le memo peut prendre n'importe quelle forme, il vous permet de retenir des informations d'un tour de jeu à l'autre (cibles, objectifs, rôles etc.). Il remplace des variables globales dont le bon fonctionnement n'est pas garanti dans le cadre de la compétition. Si vous ne souhaitez pas utiliser le memo, vous pouvez par exemple le garder égal à None.

Exemple. Votre fonction doit par exemple renvoyée : 3, 5, 2, None (placer la cible adverse sur l'emplacement adverse 3, la cible alliée sur l'emplacement allié 5, et utiliser la capacité 2 du personnage.

Si votre fonction renvoie un emplacement adverse incorrect (lié à la non prise en compte d'une provocation), le moteur de jeu le changera en un emplacement correct.

Si votre fonction renvoie une capacité incorrecte (liée à la non prise en compte du temps de recharge), la capacité 0 sera utilisée à la place.

2.4 Disqualification

Sont disqualifiés de la compétition les programmes qui déclenchent régulièrement des erreurs ou ne répondent pas au cahier des charges, ainsi que les programmes dont le temps d'éxécution ne permet pas raisonnablement la compétition. La signature (nom de la fonction, arguments, valeur de retour) des deux fonctions principales `draft` et `tour_de_jeu` doit être rigoureusement respecté pour que le programme fonctionne.

3 Description des constantes et des fonctions utiles du moteur de jeu

Le module `stlacore` (fichier `stlacore.py`) contient le moteur de jeu. Il n'est pas utile de le parcourir pour programmer votre stratégie, mais certaines constantes et fonctions utiles peuvent être utilisées :

```
# Constantes liées aux effets
EF_DEGATS = 0
EF_SOIN = 1
EF_ENDORMI = 2
EF_SONNE = 3
EF_DEFENSE_PLUS = 4
EF_DEFENSE_MOINS = 5
EF_DEGATS_DOT = 6
EF_SOINS_DOT = 7
EF_VITESSE_PLUS = 8
EF_VITESSE_MOINS = 9
EF_JAUGE_MOINS = 10
EF_JAUGE_PLUS = 11
EF_PROVOCATION = 12
EF_ESQUIVE_GARANTIE = 13
EF_TACTIQUE = 14
EF_ASSISTANCE = 15

# Constantes de ciblage
CI_TOUS = 0
CI_ALEATOIRE = 1
CI_UNIQUE = 2
CI_VOISINNAGE = 3
CI_RANGEE = 4
CI_LANCEUR = 5
CI_AUTRES = 6

CI_JOUEUR = 0
CI_ADVERSAIRE = 1
```

La fonction `charge_personnage(pid, equipe, emplacement)` peut être utilisée pour charger un personnage de la base de données à partir de son numéro dans la phase de draft afin d'estimer sa puissance.

Les personnages, que l'on retrouve également dans la propriété equipes de etatJeu (`etatJeu.equipes[0]` ou `etatJeu.equipes[1]`), sont des objets qui contiennent les propriétés vie, defense, vitesse, esquive, jauge, equipe et emplacement ainsi que la propriété `perso.capacites` qui est la liste des 3 capacités du personnage. Les capacités sont elles-mêmes des objest qui contiennent la propriété `capacite.occurences` qui est la liste des effets qu'elles peuvent déclencher. Les effets ont pour propriété eid (un numéro d'effet conforme aux constantes ci-dessus), cible (`CI_JOUEUR` ou `CI_ADVERSAIRE` suivant que la compétence vise l'équipe adverse ou l'équipe du joueur), ciblage (une constante de ciblage ci-dessus), probabilite, intensite (la puissance si l'effet est une attaque ou un soin), et duree (si l'effet s'inscrit dans la durée).

Les personnages disposent également d'une méthode `perso.subit_effet(effet)` qui renvoie un booléen indiquant si le personnage subit l'effet demandé (un numéro d'effet conforme aux constantes ci-dessus).

L'objet `etatJeu` contient les propriétés equipes (décrites ci-dessus), cibles (cibles actuelles du joueur1 dans son équipe `cibles[0][0]` et chez l'adversaire `cibles[0][1]`, cibles actuelles du joueur 2 dans son équipe et chez l'adversaire) ainsi que `doitJouer` qui contient le personnage qui doit jouer. Ce personnage vous appartient forcément (dont `etatJeu.doitJouer.equipe` est votre numéro d'équipe. `etatJeu` dispose également de la méthode `donne_provocateurs` qui renvoie la liste des emplacements des personnages ayant provocation dans une équipe donnée, ainsi que de la méthode `applique_capacite(capacite, lanceur)` qui prend en argument le personnage qui déclenche la capacité, et la capacité déclenchée (`lanceur.capacites[0]` ou 1 ou 2) qui pourrait servir à simuler l'effet d'une capacité dans l'état du jeu.

4 Pistes de travail

4.1 Phase de draft

On pourra chercher à évaluer la puissance d'un personnage à l'aide des données statistiques afin de choisir entre 2 personnages le plus puissant. On pourra chercher à évaluer la nature des personnages (offensif, défensif, soutien...) afin de créer une équipe équilibrée.

4.2 Phase de combat

Une méthode générale consiste à attribuer un score à l'état du jeu et chercher la capacité et les cibles qui maximiseraient ce score. La difficulté consiste alors simplement à établir une fonction qui donne un score à l'état du jeu.

Sinon, on pourra chercher à éliminer le personnage le plus faible en premier, ou le personnage susceptible de délencher provocation. On pourra chercher à contrôler ou affaiblir l'équipe adverse à l'aide des effets négatifs. On pourra chercher le meilleur choix de personnage à soutenir.