War of blobs

1 Principe

1.1 Situation initiale

« War of blobs » est une simulation mettant en scène des blobs sur une grille rectangulaire.

Au démarrage de la simulation, on attribue à chaque blob :

- un nom sous forme d'une chaîne,
- un poids entier,
- une couleur sous forme d'un triplet (r, v, b) où les nombres `r`, `v` et `b` appartiennent à l'intervalle $[0, 1]$.

1.2 Fusion

Les blobs sont des animaux curieux : ils peuvent fusionner !

Considérons un blob nommé « Ga » de poids 3, et un blob « Mu » de poids 5 Lorsque les deux blobs fusionnent, ils donnent un blob « MuGa » créé à l'emplacement du plus petit (« Ga ») de poids 8 (=3+5).

Plus tard, si le blob `Muga, 8` fusionne à son tour avec le blob `Bu, 6`, cela donnera le blob `MuGaBu` de poids 14.

La fusion des couleurs est réalisé en calculant, pour chaque composante, la moyenne pondérée par le poids de chaque blob des composantes des 2 blobs.

Par exemple, si `r1` désigne la composante `r` du blob « Ga » et `r2` la composante `r` du blob « Mu », alors la composante `r` du blob « MuGa » sera égale à $\frac{3\times r_1+5\times r_2}{3+5}$. Les autres composantes se calculant de manière identique.

L'étape de fusion consiste à fusionner les blobs sur la grille. On commence par identifier l'ensemble des paires de blobs côte à côte. Un blob ne peut être présent dans plus d'une paire.

En cas de conflit :

- Sur une même ligne, on choisit la paire la plus à gauche. `Ga|Bu|Mo|` donnera la paire `(Ga, Bu)` les blobs `Ga|Bu|Mo|Ze` donneront les paires `(Ga, Bu)` et `(Mo,Ze)`

- Sur une même colonne, l'ordre de haut en bas est privilégié.
- Sur une diagonale, l'ordre de parcours de la grille prévaut (de haut en bas puis de gauche à droite.
- En cas conflit ligne/colonne/digonale, on privilégie d'abord les lignes, puis les colonnes et enfin les diagonales :

  Dans la configuration

Ga	**Bu**
Mo	

  On sélectionnera la paire `(Ga, Bu)`

  Dans la configuration

Ga	
Mo	Bu

  On sélectionnera la paire `(Mo, Bu)`

Cette opération est délicate, car il ne faut pas fusionner plusieurs fois le même blob. On pourra maintenir durant les itérations la liste des coordonnées des blobs destinés à être fusionnés : une coordonnée d'indice pair dans la liste indique que le blob correspondant fusionnera avec celui dont les coordonnées sont à l'indice suivant (donc impair).

Les paires étant identifiées, on effectue les fusions comme expliqué dans la section précédente.

1.3 Déplacement

À chaque étape de la simulation :

- On fusionne les blobs côte à côte.
- Puis les blobs **n'ayant pas fusionnés** se déplacent d'une case choisie parmi les huit cases voisines. On déplace les blobs dans l'ordre que l'on veut et immédiatement (attention donc à ne pas faire bouger les blobs plus d'une fois). En conséquence, la case choisie peut contenir un blob (qui a déjà bougé). Dans ce cas, on ne bouge pas et les blobs fusionneront à l'étape suivante. Là encore on pourra maintenir un ensemble contenant les coordonnées des blobs à ne plus modifier (fusionnés et déplacés).

Le choix du déplacement d'un blob dépend de plusieurs paramètres :

- sa position `pos` et son poids `p`,
- Les autres blobs de la grille.

plusieurs fonctions de déplacement peuvent être proposées. Ces fonctions prennent en paramètre les éléments précédents et renvoient le déplacement calculé. La position du blob est ensuite calculée en fonction des limites de la grille et de la présence ou non d'un blob dans

la case choisie.

1. Un première possibilité est de choisir le blob le plus proche et à distance égale celui le plus gros et ensuite de se diriger vers ce blob. Il faut pour cela disposer d'au moins deux blobs. On pourra créer des listes de tuples `(distance, -poids, coordonnées)` puis les trier.

   La fonction suivante permet de calculer un déplacement vers une case voisine d'une position `p` en allant dans la direction donnée par le vecteur `v` :

```
 1 def vector_to_dep(v):
 2     """
 3     :param v: (tuple of int) a vector
 4     :return: (tuple of int) a vector corresponding to neighborhood
 5              deplacement
 6     """
 7     n = math.sqrt(v[0]**2+v[1]**2)
 8     cos_theta = v[0]/n
 9     sin_theta = v[1]/n
10     theta = math.acos(cos_theta)
11     if sin_theta < 0:
12         theta = 2*math.pi-theta
13     conv = { 1 : (1,0), 2 : (1,1), 4 : (0,1), 5 : (-1,1), 7 : (-1, 0), \
14              8 : (-1,-1), 10 : (0,-1), 11 : (1, 1), 12 : (1, 0) }
15     it = iter(conv)
16     k = next(it)
17     while theta > k*math.pi/6:
18             k = next(it)
19     return conv[k]
```

2. Une autre possibilité est de choisir le blob le plus gros et, à poids égal, celui le plus près et de se diriger vers ce blob.

   On pourra créer des listes de tuples `(-poids,distance,coordonnées)` puis les trier.

3. Vous pouvez également considérer non pas l'ensemble de la grille, mais seulement les blobs situé à une certaine distance. Après tout, on ne sait pas les blobs ont une bonne vue !

4. Une autre possibilité est de simuler la force de gravité : le déplacement d'un blob est obtenu en calculant la somme des forces de gravité qu'exercent les autres blobs sur lui.

5. Enfin, une dernière possibilité est de simuler la gravité en fuyant les blobs plus gros : le déplacement d'un blob est obtenu en calculant la somme des forces de gravité qu'exercent les autres blobs sur lui.

2 Travail demandé

Toutes les fonctions que vous produirez devront être documentées avec des exemples.

2.1 Déterminer une structure de donnée adaptée pour représenter en mémoire la simulation.

Utiliser à profit ce que vous avez appris sur les tuples, les listes et les dictionnaires.

2.2 Écrire une ou plusieurs fonctions permettant d'afficher sous forme textuelle la simulation

Attention : l'affichage de la grille doit s'adapter à la longueur des noms des blobs.

2.3 Écrire une fonction permettant de sauvegarder une simulation dans un fichier.

On imaginera un format de fichier texte adapté à la sauvegarde.

2.4 Écrire une fonction permettant de charger une simulation depuis un fichier.

C'est l'opération réciproque de la précédente.

2.5 Écrire une fonction permettant de générer aléatoirement une simulation

Il s'agit ici de fournir les dimensions de la grille, le nombre de blobs désirés et une manière de contrôler leur poids et leur couleur.

2.6 Écrire des fonctions permettant d'effectuer un pas de la simulation

On choisira pour commencer une fonction de déplacement simple. La fonction permet de passer de l'état courant de la grille à l'état suivant.

Voici un exemple de simulation que l'on pourrait obtenir :

```
1 # crée un plateau de 10 colonnes, 6 lignes avec 5 blobs de poids maximum 7,
2 g = wob_random(5,10,6,7)
3 while wob_number_of_blobs(g) > 1:
4     print(wob_to_str(g))
5     wob_next(g)
```

```
6
7 print(wob_to_str(g))
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |b5(7)|     |     |     |     |b2(6)|     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |b4(1)|     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |b1(3)|     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |b3(2)|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |b5(7)|     |     |     |b2(6)|     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |b4(1)|     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |b1(3)|     |     |     |     |b3(2)|     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |b5(8)|     |     |     |b2(6)|     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |b1(3)|     |     |     |     |b3(2)|     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |b5(11)|      |      |      |b2(8) |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+

+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
```

```
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |b5(11)|      |      |b2(8) |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+

+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |b5(11)|b2(8) |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+

+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |b5(19)|      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
|      |      |      |      |      |      |      |      |      |      |
+------+------+------+------+------+------+------+------+------+------+
```

3 Pour aller plus loin

3.1 Implémenter plusieurs fonctions de déplacement.

Comparer les résultats obtenus avec différentes stratégies. Peut être même placer deux types de blobs sur la grille en nombre égaux et vérifier qui « gagne ».

3.2 Détecter les fins de simulation ou les simulations infinies.

Un cas de simulation infinie peut être abordé, celui où la grille revient dans un état précédemment observé.

Il faut donc stocker les différents états de la grille et tester.

3.3 Écrire une interface graphique

Écrire une interface graphique de la simulation : les blobs sont représentés par des disques dont le rayon dépend du poids du blob, et coloriés avec la couleur de chaque blob.

Vous trouverez ici le squelette d'une interface graphique réalisée avec pygame, dans lequel le « plateau » et les blobs sont générés aléaoirement.