Rapport Prey and Predators

I - Approche méthodologique

Pour réaliser la simulation, nous avons défini les différentes méthodes pour nos agents.

I.1 - Agent GrassPatch

Pour l'herbe, nous avons défini sa repousse comme suit : si l'herbe est *fully_grown*, alors le coundown est remis à sa valeur initiale. Dans le cas où celle-ci a été mangée par un mouton, alors à chaque étape le countdrown est décrémenté jusqu'à ce que celui-ci atteigne 0 (on se retrouve alors dans le cas n°1).

I.2 - Agent Sheep

En ce qui concerne les moutons, nous avons défini premièrement la méthode *eat()* afin de simuler le fait que le mouton se nourrit d'herbe lorsqu'il se retrouve sur une case où l'herbe a poussé. Nous avons ensuite défini une méthode *reproduce()* pour simuler la reproduction asexuelle du mouton avec une certaine probabilité, puis la méthode *die()* pour simuler sa mort lorsqu'il se fait manger ou qu'il arrive à court d'énergie. Sur cette dernière méthode, la mort impliquait non seulement de retirer le mouton de la carte, mais également de le retirer du *schedule* qui répertorie l'ensemble des agents vivants.

Nous avons ensuite implémenté l'ensemble des actions se produisant lors d'une étape: tout d'abord le mouton bouge et perd donc de 1 point d'énergie. S'il se situe sur une case avec de l'herbe foisonnante, alors il mange et gagne 4 points d'énergie. Puis s'il a suffisamment d'énergie, il se reproduit avec une certaine probabilité. Enfin, si son énergie est tombée à 0 durant cette étape, il meurt.

I.3 - Agent Wolf

Concernant les loups, on retrouve des comportements similaires aux moutons: la reproduction s'effectue de la même manière avec une probabilité q et la mort survient quand le loup n'a plus d'énergie. Néanmoins, au lieu de manger de l'herbe, celui-ci se nourrit de moutons : s'il se retrouve sur une case où est présent un mouton, alors celui-ci le mange et gagne 20 points d'énergie. Chaque step se déroule alors comme suit: le loup bouge et perd 1 point d'énergie. Puis il mange un mouton dans le cas où un d'entre eux se trouve sur la même case que lui. Il se reproduit ensuite avec une probabilité q et meurt si son énergie est tombée à 0 durant le déroulement de l'étape.

I.4 - Simulation

Enfin, nous avons mis en place la simulation à proprement parler grâce au fichier *server.py* et *model.py*. Pour cela, nous avons affiché une carte et placé nos agents dessus : l'herbe est représentée en verte sur la couche 0, les moutons en rouge sur la couche 1 et les loups en noir sur la couche 2. Nous avons prédisposé sur le côté gauche des boutons permettant de faire varier les paramètres afin de rendre la simulation plus interactive.

Clément BLANCHOZ--RHÔNE et Juliette COCHEZ

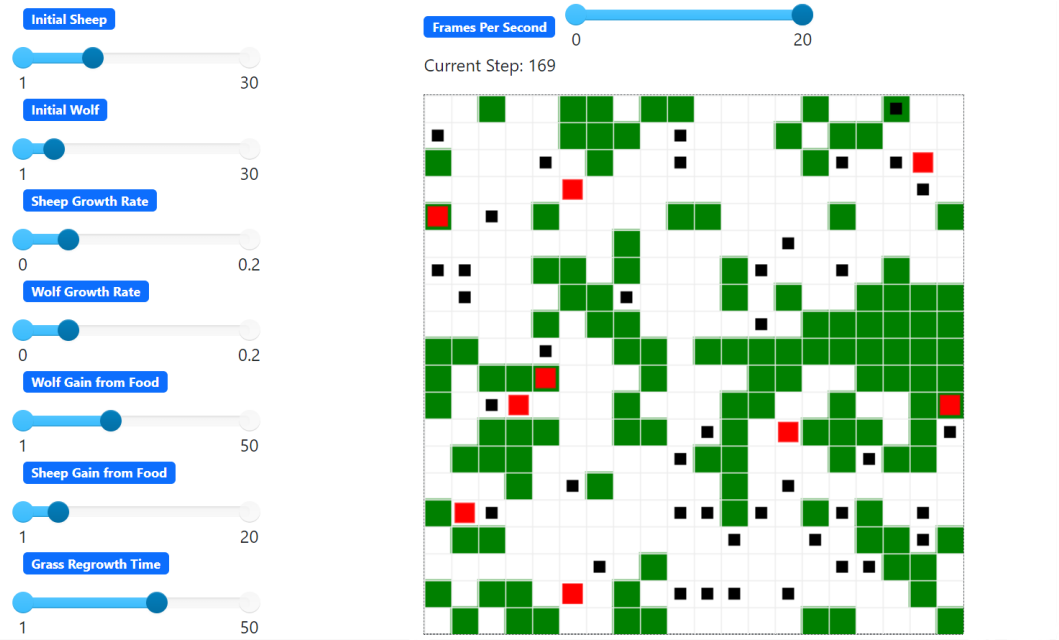
II - Choix des paramètres

L'équilibre loup-mouton est assez délicat à obtenir. En particulier, il s'agit de trouver la bonne proportion entre le taux de reproduction et l'énergie apportée par la nourriture. Finalement, pour un *growth rate* identique, nous sommes partis sur les paramètres suivants:

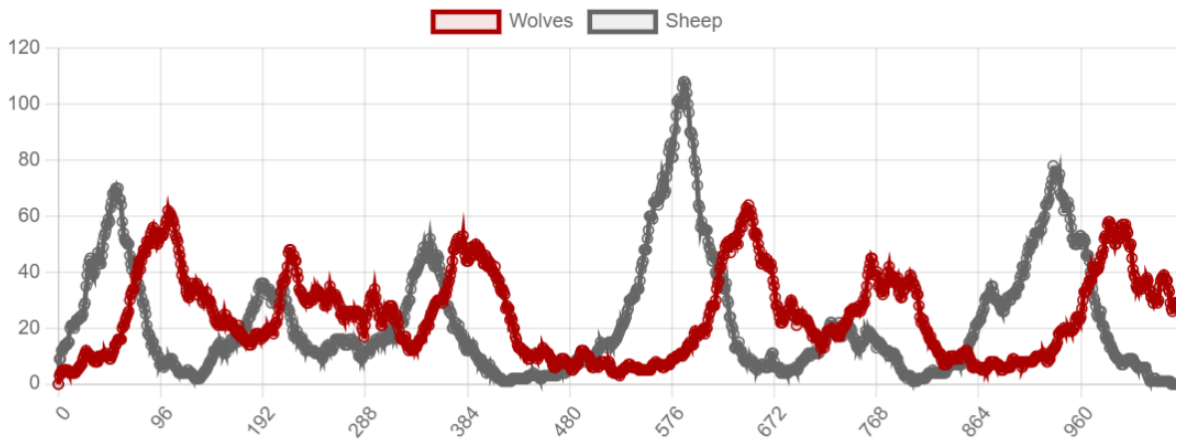
- 2 à 3 fois plus de moutons que de loups à l'état initial (par exemple 10 pour 5 ou 9 pour 3 nous permettent d'atteindre les 1000 itérations)
- *wolf gain from food* = 20
- *sheep gain from food* = 4

III - Captures d'écran de la simulation

Voici la carte que nous obtenons après une centaine d'itérations : les loups apparaissent en rouge, les moutons en noir, l'herbe ayant poussé en vert et l'herbe en cours de pousse appraît en blanc.



Voici un résultat que nous obtenons après plus de 900 itérations. Il est intéressant de voir que le modèle ne semble pas diverger lorsque l'une des populations commencent à grossir brusquement (ex: vers les 576 itérations)



Clément BLANCHOZ--RHÔNE et Juliette COCHEZ