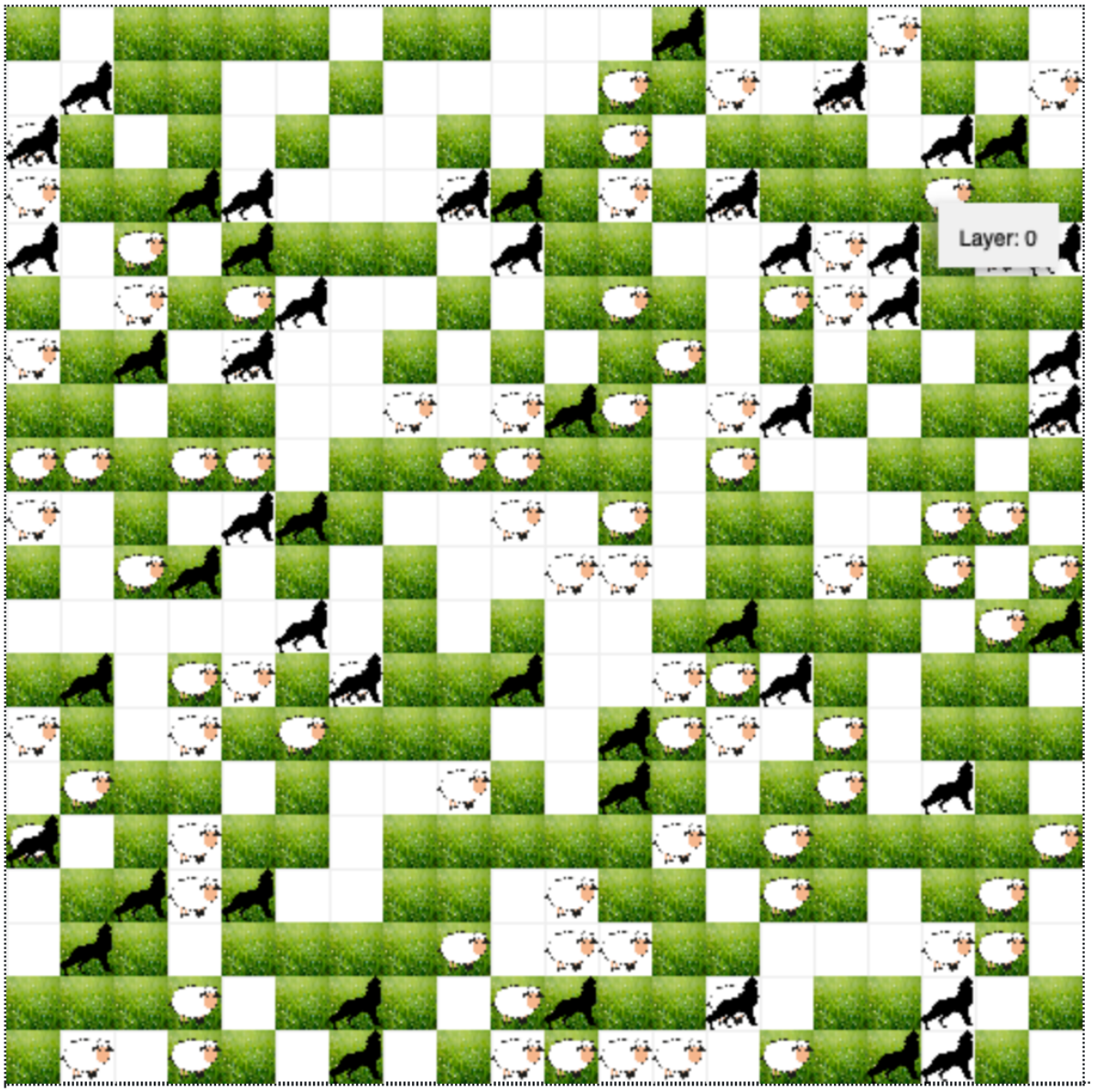
# Prey Predator

Mohammed-Karim Khaldi, Leila Berrada



# 1 Description de la méthode

## 1.1 Architecture générale

Le code peut se retrouver sur notre repository git : Code

- Initialisation des paramètres : Nous avons placé des sliders pour choisir différentes valeurs pour le modèle comme le nombre de moutons, de loups et le nombre, l'énergie initiale....
- Agencement du code : pour plus de lisibilité du code et notamment du fichier agent ce dernier à été séparé en 3 fichiers et placé dans un dossier agent.
- Visualisation sur la grille : par mesure d'esthétique nos agents sont représentés par des images décrivant leur nature.

## 1.2 Loups

Les agents représentant les loups ont été modélisé via un ensemble de variables, plus précisément leurs énergies, le gain obtenu lorsqu'ils mangent des moutons, la nature de leurs mouvements, leurs taux de reproduction ainsi que l'impact des 2 derniers paramètres sur l'évolution de leurs énergies. Plus concrètement,

- Lorsqu'un loup mange un mouton, celui-ci gagne de l'énergie, qui est proportionnelle à l'énergie que possédait le mouton : ceci sert à simuler le fait que les moutons puissent être de taille différente.
- Un loup qui a atteint un seuil de satiété ne se nourrit pas.
- Tous les loups ne sont pas aptes à se reproduire simultanément. A chaque instant, un pourcentage défini des loups est considéré comme apte à se reproduire, et se reproduit si celui-ci se dirige vers une cellule où un autre loup est présent et que les deux disposent de suffisamment d'énergie.
- Les loups qui souhaitent se reproduire se déplacent en cherchant en priorité à rejoindre d'autres loups disposant de suffisamment d'énergie pour cela.
- Lorsqu'un loup ne peut se reproduire, celui-ci cherche à se diriger vers les moutons adjacents s'il y en a afin de les manger. Sinon, il bouge aléatoirement.

## 1.3 Moutons

Les agents représentant les moutons ont été modélisé via un ensemble de variables, plus précisément leurs énergies, le gain obtenu lorsqu'ils mangent de l'herbe, la nature de leurs mouvements, leurs taux de reproduction ainsi que l'impact des 2 derniers paramètres sur l'évolution de leurs énergies :

- Lorsqu'un mouton mange de l'herbe, celui-ci gagne de l'énergie, qui est un paramètre fixe prédéfini avant chaque simulation.
- Lorsqu'un mouton se reproduit, son énergie est divisée par deux, et une nouvelle instance de mouton est créée sur la même case.
- Pour les déplacements du mouton
  - Avec une certaine probabilité, les moutons iront chercher de l'herbe :
    - Si de l'herbe se trouve sur l'une des cases de son voisinage de Moore :
      - Si elle est unique : Il ira sur cette case.
      - Si plusieurs cases dans son voisinage contiennent de l'herbe, il ira aléatoirement sur l'une d'entre elle.
    - Sinon : Il aura un mouvement aléatoire.
  - Avec une certaine probabilité, ils chercherons principalement à éviter les loups.

### 1.4 Herbe

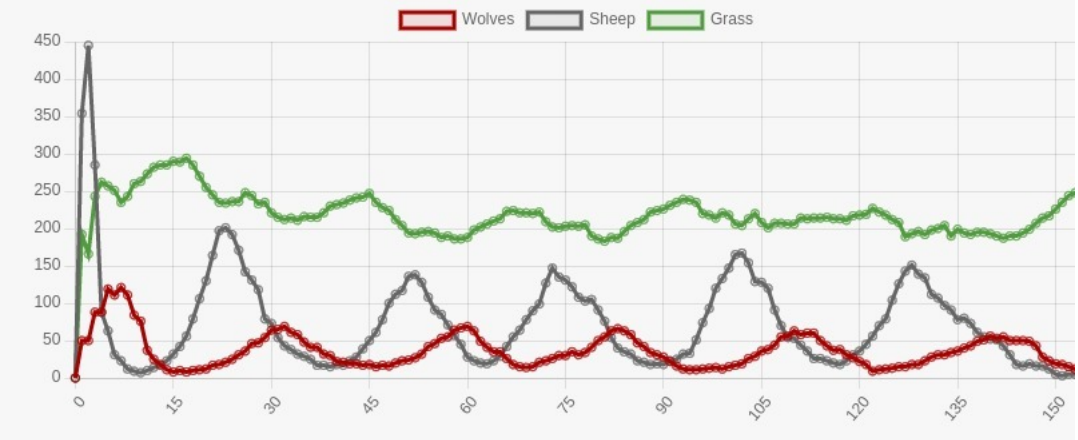
Pour l'herbe, différentes éléments ont été ajoutés :

- À l'initialisation, une quantité d'herbe verte aléatoire est ajoutée à la grille.
- À chaque step :
  - Une quantité aléatoire entre 0 et 3 d'herbe est ajoutée à la grille.
  - Pour les patchs d'herbe morte, il y a 15% de chance que l'herbe disparaisse.
- Lorsqu'un mouton ou un loup meurt sur un case il devient du composte pour l'herbe ! :)
  - Si la case sur laquelle il se trouvait est une case d'herbe verte : rien ne se passe.
  - Si la case sur laquelle il se trouvait quand il est mort est une case d'herbe morte, son countdown pour redevenir verte diminue, ainsi accélérant sa repousse.
  - Si la case sur laquelle il meurt n'était pas une case d'herbe, elle devient une case d'herbe morte, qui poussera donc au terme du countdown.

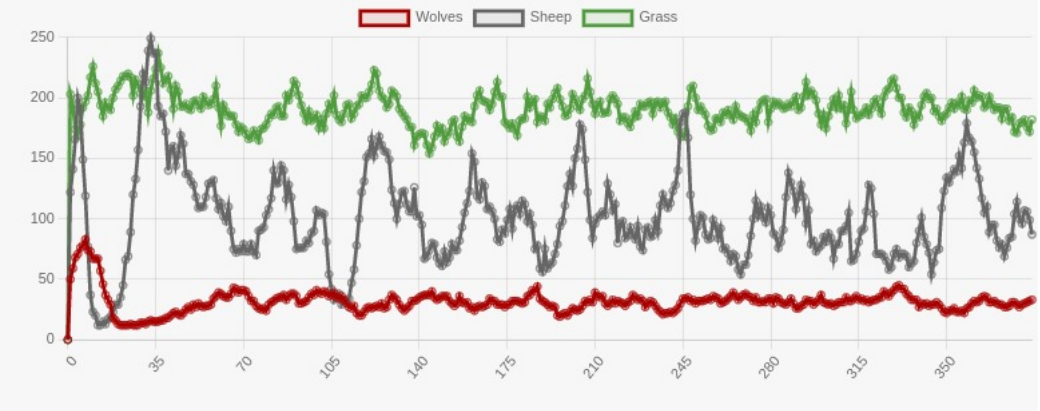
## 2 Analyse et Résultats

Nous avons lancé différents tests et nous avons obtenus des résultats intéressants :

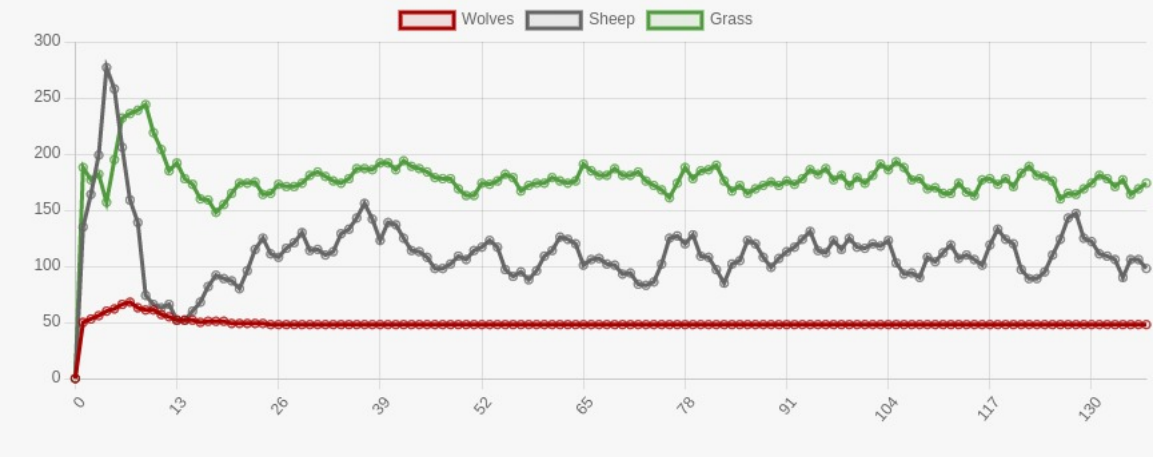
- Cas 1 : Nous obtenons des résultats périodiques :
  - Ce cas est intéressant, car on observe une périodicité dans la quantité des proies et des prédateurs. Dans cet exemple, les moutons se déplacent systématiquement aléatoirement, et les loups récupèrent assez peu d'énergie en mangeant des moutons. Un effet de régularisation des populations a lieu naturellement.




  - Dans cet autre exemple où les moutons cherchent à éviter les loups, on constate aussi une périodicité des résultats où les trois populations ( loups, moutons, herbes) coexistent. Notons que le nombre minimal de mouton est ici plus élevé que précédemment, mais que le nombre maximal n'excède que rarement 150 moutons. En effet, nous travaillons sur une grille de 400 cases, ainsi lorsqu'un seuil critique de loups + moutons est atteint, les loups dévorent les moutons et se reproduisent.




- Cas 2 : Dans ce cas, nous avons choisi d'avoir une part de loups définis à leurs créations qui peuvent se reproduire. La reproduction consommant de l'énergie, les loups pouvant se reproduire finissent par consommer cette énergie et disparaître, et la population des loups restante parvient à se nourrir en étant incapable de se reproduire, d'où une population constante de celle-ci. Via les résultats des moutons et de l'herbe, On conclut donc que réguler le nombre de prédateurs dans notre modèle semble réguler le nombre de proies, comme visible ci-dessous :




- Cas 3 : Dans ce cas, nous observons que le nombre de moutons et de loups augmente, jusqu'à un seuil critique où les loups sont plus nombreux que les moutons et les dévorent, amenant ainsi l'extinction des deux espèces, tandis que l'herbe reste en vie, c'est "superherbe" !