Notre objectif était de coder un programme python afin de simuler les interactions entre lapins et renards sur un plateau.

Pour commencer, nous avons dû importer les librairies suivantes : tkiteasy (pour le graphique), random (car nous avons eu besoin de mettre en place de l’aléatoire dans plusieurs parties du programmes), time (pour mieux voir les tours s’enchainer) et matpotlib (afin de pourvoir réaliser le graphique des populations des animaux).

En commençant le projet, nous avons décidé d’utiliser les classes car on venait de le voir en cours et on trouvait le principe intéressant (on se rendra compte plus tard qu’avec la manière dont on a codé notre programme, notre unique classe qui n’est pas forcément très utile mais qui rend notre projet plus simple). Nous avons donc créé la classe forêt() en l’initialisant avec le plateau.

Effectivement, on utilise pour faire fonctionner la simulation un plateau qui est un tableau à 2 dimensions et qui permet de pouvoir trouver facilement l’élément qui se trouve sur une case précise. On initialise donc ce tableau avec les premières positions des animaux qui sont stockées précédemment.

On va ici détailler toutes nos structures de données essentielles :

-Stockage des coordonnées :

poslapins et posrenards qui sont des listes stockant respectivement les coordonnés des lapins et des renards sous formes de tuples (x,y) qui correspondent à la position dans le plateau

-Stockage de la vie et de l’énergie :

agelapins et agerenards qui sont des listes stockant le nombre de vie de chaque lapin ou renard sous forme d’entier.

Energierenards qui stocke sous forme d’entier l’énergie des renards (les lapins n’ont pas cette contrainte d’énergie) dans une liste à une dimensions.

-Stockage des objets graphiques :

Objetlapins qui est une liste à une dimension qui stocke les disques de couleur verts qui représentent les lapins sur la fenêtre graphique.

Objetrenards qui est une liste à une dimension qui stocke les disques de couleurs rouges qui représentent les renards sur la fenêtre graphique.

-Le plateau qui est un tableau à deux dimensions contenant uniquement des entiers (0 pour une case vide, 1 pour un lapin et 2 pour un renard

De plus, il faut savoir que pour trouver les caractéristiques d’un lapin, il suffit de regarder sa position sur le plateau et de trouver son indice dans la liste poslapins car les indices correspondent entre toutes les listes car ils sont synchronisés grâce à la fonction actulapins() qui vient supprimer de toutes les listes les éléments correspondant à l’indice du lapin dont le nombre de vie et nul. Donc dans la liste poslapins, par exemple, les coordonnées du huitième lapin correspondent donc au huitième élément de la liste objetlapins

Le fonctionnement est le même pour les renards, la seule différence est qu’il y a aussi la liste energierenards qui se rajoute à agerenards dans la condition de suppression des renards.

Enfin, il faut aussi noter que tous les problèmes rencontrés au cours du projet sont parvenus à trouver leur solution, nous avons essayé au maximum de rendre notre programme le plus robuste possible pour l’empêcher de planter quand par exemple, le nombre de lapins et de renards initiaux est trop important par rapport à la taille du plateau.

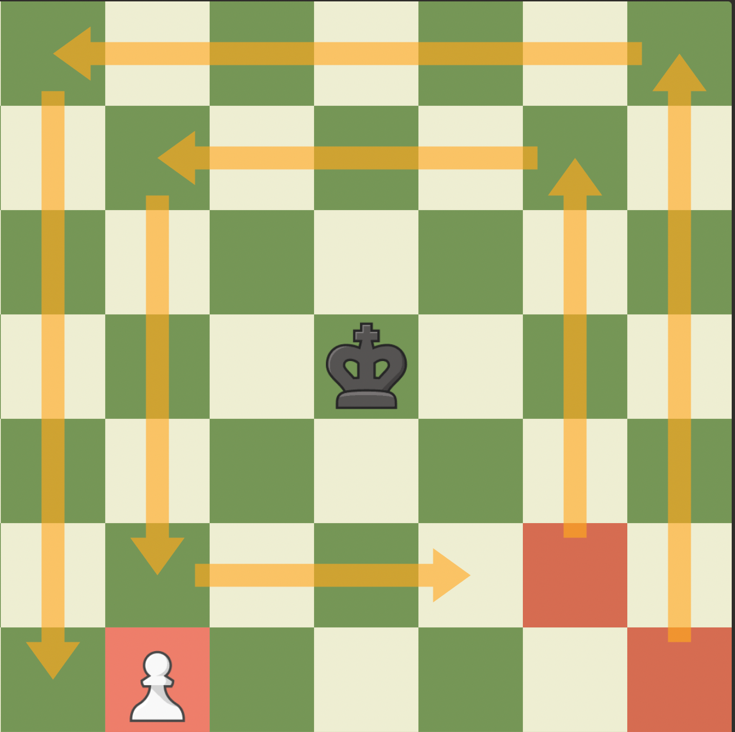
À l’heure d’aujourd’hui, après de multiple test, nous ne subissons plus aucun bug majeurs.

Pour le fonctionnement de la simulation, nous avons décidé l’ordre des actions suivants qui est pour nous le plus réaliste possible :

Le déplacement (d’abord celui des lapins puis celui des renards), la reproduction (des renards d’abord puis des lapins), l’actualisation des listes (l’ordre n’a pas d’influence sur la simulation) et enfin l’apparition des nouveaux animaux (d’abord les renards car sinon, une fois que le plateau est plein, les nouveaux renards ne peuvent plus apparaître.

Il faut aussi savoir que nous avons changé une règle sur la reproduction des renards, l’âge nécessaire n’est pas l’unique condition car nous avons aussi rajouter la condition que les renards voulant se reproduire doivent être à proximité d’un autre renard pouvant se reproduire.

Avant de finir, on va parler de la partie technique du projet et de l’algorithme de chasse que nous avons mis en place après réflexion. Nous avons réussi à trouver un algorithme de chasse comme on peut le voir sur le visuel ci-dessous que nous avons réalisé afin d’expliquer facilement la façon dont le renard recherche le lapin le plus proche de lui. Dans ce visuel réalisé à partir d’un jeu d’échec, le roi représente le renard et le pion le lapin, les points rouges sont les cases où l’on place le détecteur et les flèches représentent le déplacement du détecteur, celui-ci s’arrête une fois un lapin trouvé.



Et maintenant, pour conclure notre travail, nous avons voulu apporter notre petite touche personnelle au projet en ajoutant quelque chose, nous avons donc décider de rajouter sur la fenêtre graphique un compteur en direct du nombre de lapins et de renards ainsi qu’une petite jauge pour se visualiser la répartition entre les lapins, les renards et le vide.