# Rapport de projet en Conception Logiciel

mise en page par
Mechtouf Sami,
Elhadj Alseiny Diallo,
L1 info,
Groupe 1b,

20 avril 2021

Ce présent document à pour but de vous décrire les étapes de la création du projet que nous avons réalisé tout au long de l'année, reposant sur l'élaboration d'un simulateur d'écosystème.

# Sommaire

# 1 Introduction

## 1.1 Présentation du projet

L'idée de la création d'un écosystème nous a fasciné immédiatement. En effet, ce qui nous a poussé à choisir ce thème c'est l'imagination d'un système capable d'évoluer en autonomie grâce à nos lignes de code.

C'est pourquoi nous sommes partis sur une première base où l'évolution ce ferait dans une fenêtre qui afficherait une grille, et où les espèces seraient représentées par des points de couleurs différentes. Un programme qui s'attardait plus sur le résultat à l'aide d'un graphique, que sur le développement de fonctionnalités liés à l'évolution de l'écosystème.

Néanmoins, comme dit précédemment, ce programme était assez "simple" car il nécessitait seulement la création de classes (bien que difficile puisque nous n'y connaissions pas grand chose), une grille pour une modélisation basique et enfin des déplacements aléatoire. Nous avions donc décidé d'aller plus loin dans notre travail et surtout dans ce qui pourrait ressembler au maximum à de l'intelligence artificielle.

## 1.2 Les objectifs fixés

Nous nous sommes donc fixés plusieurs objectifs dont le premier était de basculer d'un jeu où le résultat était le plus important à un jeu où le développement du jeu lui même est le plus important (notamment grâce à des déplacements plus 'logique' et moins aléatoire, à l'ajout de modes interactifs, à des conditions précises de croisement entre espèces etc...). Le second objectif était de réaliser un support graphique adéquat à toutes les fonctionnalités que nous voulions ajouter, bien plus lisible comme ça plutôt qu'une grille de couleur qui empêcherait toute vue sur, par exemple, les séances de chasse, de reproduction, de nutrition etc... Pour finir, pour éviter de tomber dans un jeu redondant dont le résultat serait toujours le même nous voulions ajouter un menu où nous pourrions décider du nombre de proies, de prédateurs et de végétaux avant chaque début de partie. De plus, cela nous permet d'ajouter un maximum possible entre le joueur et le jeu, qui sont déjà censé être faibles dans un simulateur d'écosystème. Pour résumer, nous voulions :

1. Un jeu intelligent et non aléatoire (ou trop différent de la réalité)
2. Une interface graphique propre et lisible pour que l'on puisse observer ce que l'on veut faire
3. Un jeu différent à chaque lancement

# 2 Fonctionnement du jeu

## 2.1 Les bases du jeu

Ce jeu est une simulation d'écosystème. Il mets en jeu deux types d'animaux (les lions et les gazelles) qui apparaissent et évoluent dans un espace extérieur clos. Au lancement du jeu, nous devrons choisir le nombre d'animaux et de végétaux que nous voudrons au démarrage du jeu, puis nous aurons les lions et les gazelles, de sexe complètement aléatoire, éparpillés sur tout le terrain (ainsi que de l'herbe pour nourrir les gazelles).

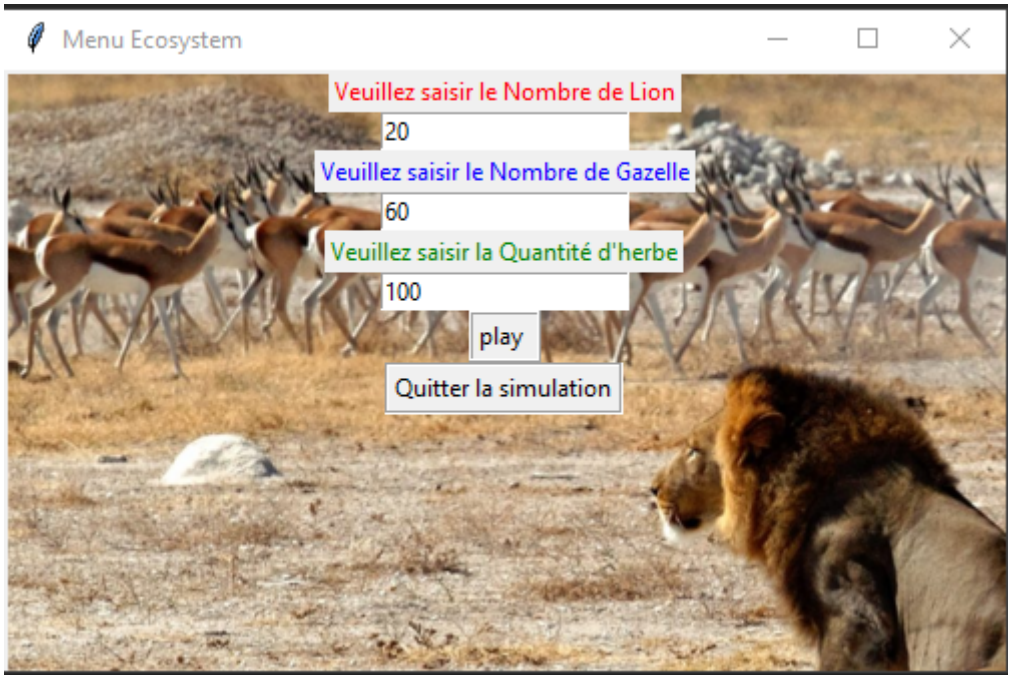
Au fur et à mesure du temps indiqué par le jour et la nuit, on assistera au développement de l'écosystème ainsi créé. Nous pourrons voir différentes actions de la part des animaux, tels que la chasse des gazelles lorsque les lions ont faim, la fuite des gazelles lorsqu'un lion est trop proche, l'apparition de nouveaux nées après reproduction d'une espèce, les déplacements dans un espace réduit lorsque les animaux n'ont plus faim etc...

Ce jeu à été conçu de telle sorte à ce que la simulation se rapproche au mieux de la réalité, mais nous sommes bien évidemment conscient qu'il manque d'innombrables paramètres pour que ce jeu soit digne d'une représentation de la réalité. Puis, à partir du premier jour d'évolution, un graphique apparaîtra afin de pouvoir suivre l'évolution du nombre d'animaux prélevé chaque demi journée. Enfin, il sera possible tout au long du jeu d'ajouter des lions ou des gazelles, et même de supprimer les animaux que nous ne voulons plus afin de guider l'écosystème où l'on veut.

## 2.2 Les interactions jeu/joueur disponibles

### 2.2.1 Le menu

Au lancement, un menu apparaît afin de de pouvoir sélectionner les nombres d'animaux et de végétaux que l'on veut pour chaque espèce. En effet, cet interface graphique a été codé avec une fenêtre Tkinter simple dont on aura programmé le nom, la taille, l'image de fonds d'interface, les boutons ainsi que les boîtes de saisies et les valeurs par défaut que nous conseillons pour avoir un écosystème stable (7 lions, 30 gazelles et 100 herbes).
Ce menu et ces saisies constituent les premières interactions avec le jeu.



#### 2.2.2 Le jeu

Les interactions suivantes sont celles qui vont apporter une modifications extérieur à travers les touches de la souris. En effet, nous avons fait en sorte que le clic gauche fasse apparaître une gazelle et le clic droit un lion. De plus, effectuer un clic de molette de souris sur n'importe quel élément du jeu (gazelle, lion ou herbe) permet de le supprimer totalement du jeu.

# 3 La création du code

## 3.1 Création des êtres vivants

Les êtres vivants ont été créé grâce à un diagramme de classe pré-établit. En effet, une classe 'Espece' à été créé et toutes les classes suivantes (concernant la créations des animaux et des végétaux) hériterons de cette classe car elle regroupe tout se dont une espèce quelle qu'elle soit, c'est à dire :

— Un nom
— Des coordonnées X et Y

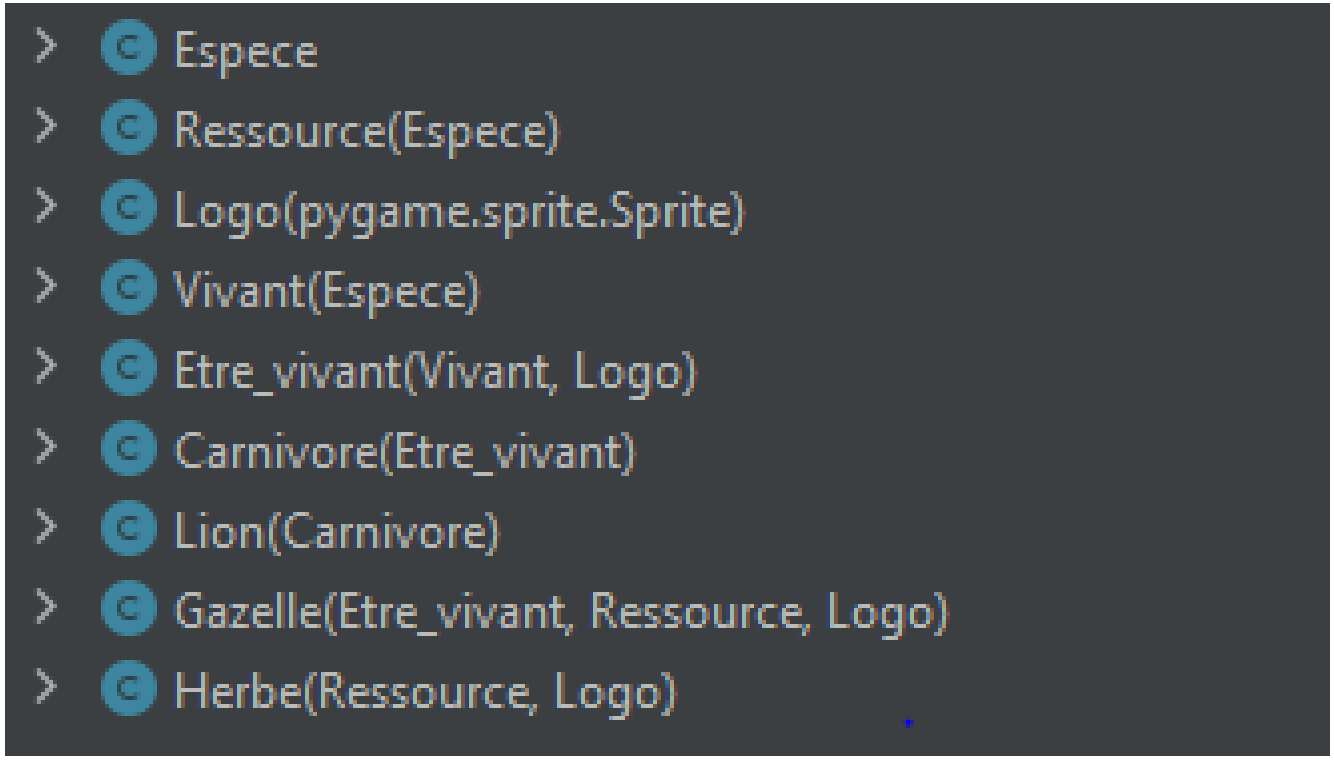
Puis on voit apparaître 2 classes différentes héritant d'Espece, c'est à dire d'un côté la classe 'Ressource', qui regroupe toutes les espèces pouvant être mangés (l'herbe et les gazelles) et de l'autre la classe 'Vivant', regroupant les espece capable de bouger et manger (les lions et les gazelles), de plus elle contient la fonction 'mourir' permettant de supprimer de mettre à 0 l'energie et de mettre le paramètre 'en vie' à False. Dans la classe Vivant nous attribuons donc :

— Le genre
— S'il l'animal et vivant ou non grâce à un booléen
— L'energie
— Son rayon d'observation
— Ses parents (pour savoir si la mère enceinte est en vie ou non)

La classe Ressource de dispose quant à elle que des paramètre determinant combien d'énergie l'espèce rapporte si elle est mangée ainsi que son nom.

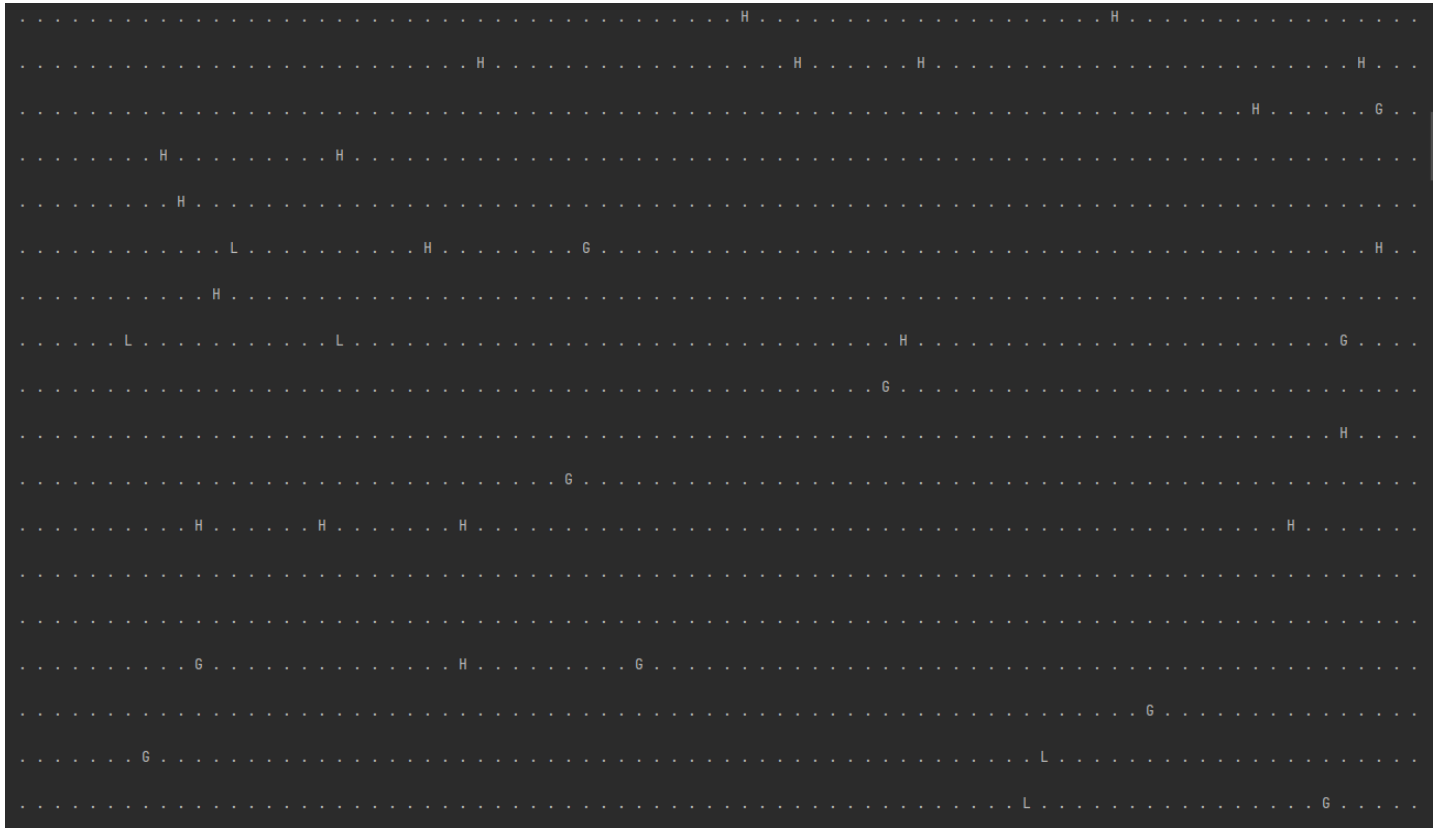
Nous avons aussi une classe 'Etre Vivant', héritant de la classe 'Vivant', contenant une fonction 'mange' qui apporte à l'animal qui mange, l'énergie que la victime lui aura apporté.

Pour finir, nous avons les classes créatrices d'espèce (c'est à dire la classe Lion, Gazelle et Herbe), comprenant tout les paramètres de l'animal (le nom, la vitesse de déplacement, l'énergie, la liste de sa nourriture etc...) ou l'énergie récupérer par les animaux pour l'herbe. Les lions sont casés dans une classe 'Carnivore', car leur mode d'alimentation diffère des Herbivores (gazelle), ils doivent donc contenir une fonction 'mange' permettant de 'tuer' l'animal dans le jeu alors que la fonction mange de la classe 'Etre Vivant' permet seulement d'augmenter l'énergie de l'animal qui mange. Les animaux contiennent aussi une fonction 'reproduction', créant une un nouveau née à chaque reproduction.




## 3.2 Mise en place du terrain de jeu : la grille

Il nous à fallut créer un nouveau fichier nommé 'grille' qui s'occuperait non seulement de la création du terrain, mais aussi des fonctionnalités impactant l'affichage des espèces et du terrain en lui même. La mise en place du terrain n'était pas forcément si compliqué, néanmoins, implémenter les espèces et les fonctionnalités au terrain l'était un peu plus. Nous avons donc décidé, pour la formation du terrain, de créer une grille à l'aide d'une liste python où toutes les espèces viendraient s'y loger, chaque déplacement d'animal serait alors représenté par un changement de case, quelque soit la forme graphique du jeu. Voici un exemple de ce que donnait la visualisation de la grille au prémices du jeu :

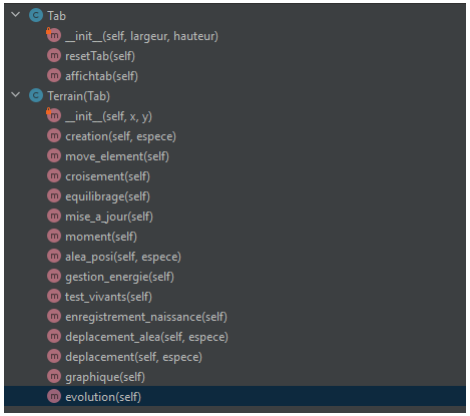


Les espèces étaient représentés par des logos, le 'L' pour les Lions, le 'G' pour les Gazelles et le 'H' pour l'Herbe.

Dans ce fichier 'grille', nous trouverons aussi des fonctionnalités liées à la création des espèces, à leurs déplacements, à la gestion de leur énergie, à l'implémentation des nouveaux nées etc... Il nous paraissait logique que ce soit ce fichier qui gère les actions plutôt que le fichier 'espece' qui n'est censé regrouper que les différentes espèces.

Le fonctionnement du jeu est finalement assez simple, le fichier 'grille' créé le terrain de jeu et au fur et à mesure que le jeu et en cours, il récupère des informations du fichier 'espece' pour les utiliser des le développement du terrain.

Aujourd'hui, concernant l'aspect graphique de la grille, le jeu a bien évolué mais le principe de déplacement et des autres fonctionnalités reste les mêmes.




## 3.3 Utilisation de Pygame

*Casiment toute la conversion en Pygame à été réalisé sur le fichier 'grille'.*

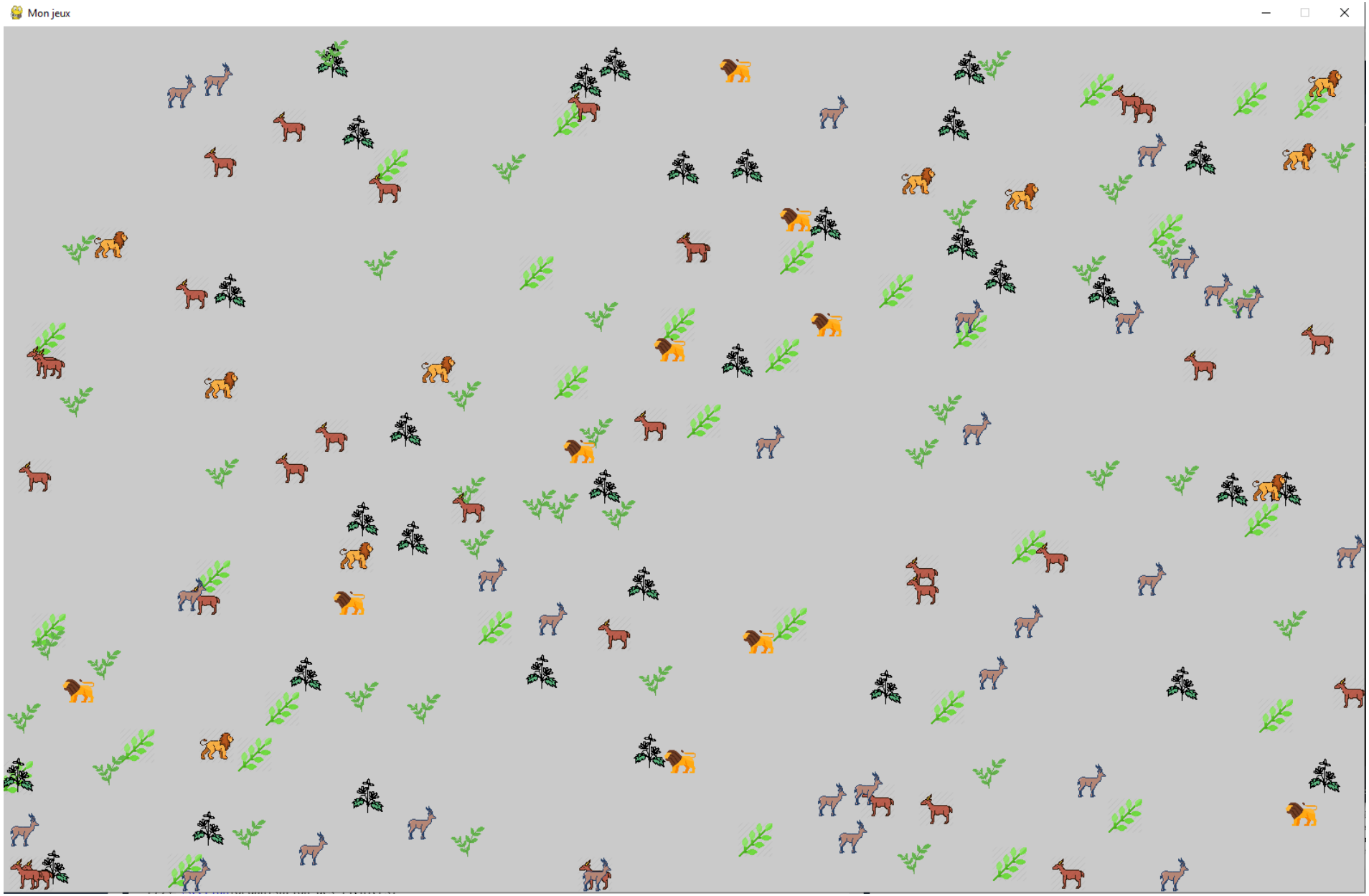
La conversion du premier prototype d'interface graphique à une interface plus lisible et agréable à l'oeil nous a parut être une évidence. En effet, bien que la première version ci-dessus était assez clair, le concept de 'jeu' perdrait son sens si son but n'était que d'avoir un résultat de simulateur final et non un jeu aux graphismes agréable.

C'est pourquoi, nous avons utilisé la bibliothèque Pygame afin d'obtenir le résultat que l'on souhaitait. A l'aide d'une fenêtre Tkinter préalablement créé, nous avons remplacer les dimensions de la grille du terrain par le nombre de pixel de la fenêtre, ainsi, nous avons modifié les représentations de chaque espèce par des images afin qu'ils soient visibles. Les 'croisement' entre espèce ne se font plus lorsque les deux espèces sont sur un même pixel mais lorsque leur image entre en colision notamment gràace à la commande **'pygame.sprite.collide-rect(animal1,animal2)'**

Une fonction 'moment' a aussi été ajouté afin de pouvoir régler la couleur de fond de la fenetre avec des basculement de noir au blanc afin de simuler le jour et la nuit. Ceci à pu être réalisé par décrémentation et incrémentation de valeurs RGB en fonction d'un temps donnés. Cette fonction va aussi nous permettre un réglage du volume sonore de l'environnement en fonction du jour et de la nuit.

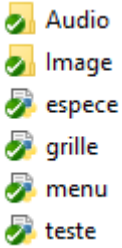
Enfin, pour une demi journée, nous avons intégrer une boucle qui tournera 500 fois en appellant les fonctions de déplacements pour donner l'impression de demis journées plus longues pour éviter un affichage instantané et unique.

Voici donc ce que donne le rendu graphique final de notre jeu.


Mon jeux

# 4 Organisation des fichiers

L'organisation des fichiers fut assez simple puisque nous avons seulement 4 fichiers python comme vu précédemment. En effet, nous avons regroupé les fichiers 'espece', 'grille', 'menu' et 'teste' (qui permet le lancement du jeu) dans le répertoire principal, ainsi que les dossiers 'image' et 'son' qui regroupent les images et les sons utilisés pour le jeu.




# 5 L'intelligence artificiel

Nous voulions un programme qui soit le plus intelligent possible et ce rapprochant un maximum de la réalité. C'est pourquoi, nous avons décidé d'incorporer un maximum à ce qui pourrait s'apparenter à de l'intelligence artificielle.

## 5.1 Le croisement

La fonctionnalité de croisement est la toute première étape de l'intelligence que nous avons décidé de créer. En effet, c'est elle qui sélectionne quelle fonction doit être appelée en cas de croisement : la reproduction ou la nutrition. Comme vous l'aurez compris, rien ne ce passe si une des deux actions ne se produit pas, et le jeu continue son développement.

Lorsque deux espèces se croisent (donc si ils ont les mêmes coordonnées x et y pour un lion, une gazelle ou de l'herbe), la fonction croisement vérifie plusieurs choses et appelle des fonctions précise en conséquence. En effet, cette fonction lance une boucle en parcourant toutes les espèces existante dans le jeu pour vérifier si au moins deux ou plusieurs espèces possèdent les mêmes coordonnées (en prenant soin de vérifier si les deux animaux testé ne sont pas les mêmes d'un point de vue informatique). Si oui, alors plusieurs solutions s'offrent à nous :

1. Si l'un des deux fait partie de la liste de nourriture de l'autre, alors on fait appelle à la fonction 'mange'.
2. Si les deux espèces testés font partie de la même espèce, qu'ils sont vivant (cette conditions de marcherait donc pas pour plusieurs herbes au même endroit) et que leur sexe sont différents, alors on appelle la fonction 'reproduction'.

   Mais que font réellement ces deux fonctions ?

### 5.1.1 La reproduction

Nous avons décidé lors de l'élaboration de ce projet que la fonction 'reproduction' appelée par la fonction 'croisement', ferait appelle à plusieurs conditions qui rendraient le programme un peu plus intelligent au lieu de seulement se faire reproduire deux animaux de même espèce qui se croisent. Voici donc les règles imposées :

— Une femelle ne peut se reproduire qu'avec un mâle de la même espèce
— Après reproduction, la femelle serait "enceinte" et ne pourrait "mettre au monde" le nouveau née seulement le lendemain
— La mère ne peut plus se reproduire tant qu'elle est enceinte
— Si jamais la mère meurt de la chasse d'un autre animal ou si elle meurt de faim pendant qu'elle est enceinte, le nouveau née de viendrait pas au monde
— Le mâle ne peut se reproduire qu'avec une seule femelle par pas effectué. En effet, si le mâle tombe sur une deux ou plusieurs femelles en même temps, il ne pourra se reproduire qu'avec une seule d'entre elle avant le prochain pas qu'il effectuera.

Si les conditions de reproduction sont remplit, les nouveaux nées seront ajoutés à une liste spéciale nouveau née pendant que la mère sera enceinte et, le lendemain, ils seront ajoutés à la listes qui regroupent toutes les espèces du jeu si et seulement si la mère est toujours vivante(ils intégreront donc le jeu à ce moment là).

### 5.1.2 La nutrition

La fonction 'nutrition', comme la fonction 'reproduction', à était régit par des règles qui sont les suivantes :

— Les lions mangent les gazelles et les gazelles mangent de l'herbe
— Même si les lions n'ont plus faim, ils continuent de tuer les gazelles qui sont trop proches d'eux.

Lorsque la fonction mange est appelé, l'énergie du prédateur est restauré en fonction du nombre de point que rapporte la proie mangé. Si l'animal et déjà au maximum de son énergie, alors le prédateur tuera sa proie seleument pour son divertissement. Enfin, vient la suppression du jeu de l'animal mangé ou tué.

## 5.2 Le déplacement

### 5.2.1 Le déplacement aléatoire

La fonction de déplacement aléatoire permet aux espèces de se déplacer aléatoirement lorsqu'ils n'ont pas faim. En effet, dans notre jeu, la faim dépends de l'énergie de l'animal. Mais comment est représentée la faim ?

> La faim est déterminée par un niveau d'énergie qui est elle même représentée par un nombre de point. Par exemple, un lion commencera sa vie avec un niveau d'energie maximum de 300 points tandis qu'une gazelle en aura seulement 150. A chaque fin de journée, le niveau d'energie se décrémente de 20 points pour chaque animal dans le jeu. Deux cas s'offre alors à nous en fin de journée, soit l'animal arrive à un moment donnés en dessous de la moitié de son énergie maximum et entre donc en mode 'prédateur' et part à la chasse d'une proie, soit l'animal tombe à une énergie de zéro, dans quel cas l'animal meurt et se supprime du jeu. Lorsqu'un animal mange, il acquérit le nombre de point 'nourrisant', qu'attribut l'espèce mangé, et l'ajoute à son niveau d'énergie.

### 5.2.2 La chasse/La fuite

La fonction de deplacement logique est sans doute la fonction la plus complexe de tout le projet. En effet, elle a demandée une certaine logique mais aussi des calculs mathématiques qui ont été loin d'être facile à trouver. Malgré tout nous y sommes arrivé. Pour la chasse des prédateurs, voici comme cette fonction marche par étape :

1. Une formule permettant de calculer la distance entre deux espèce x et y a été calculé :

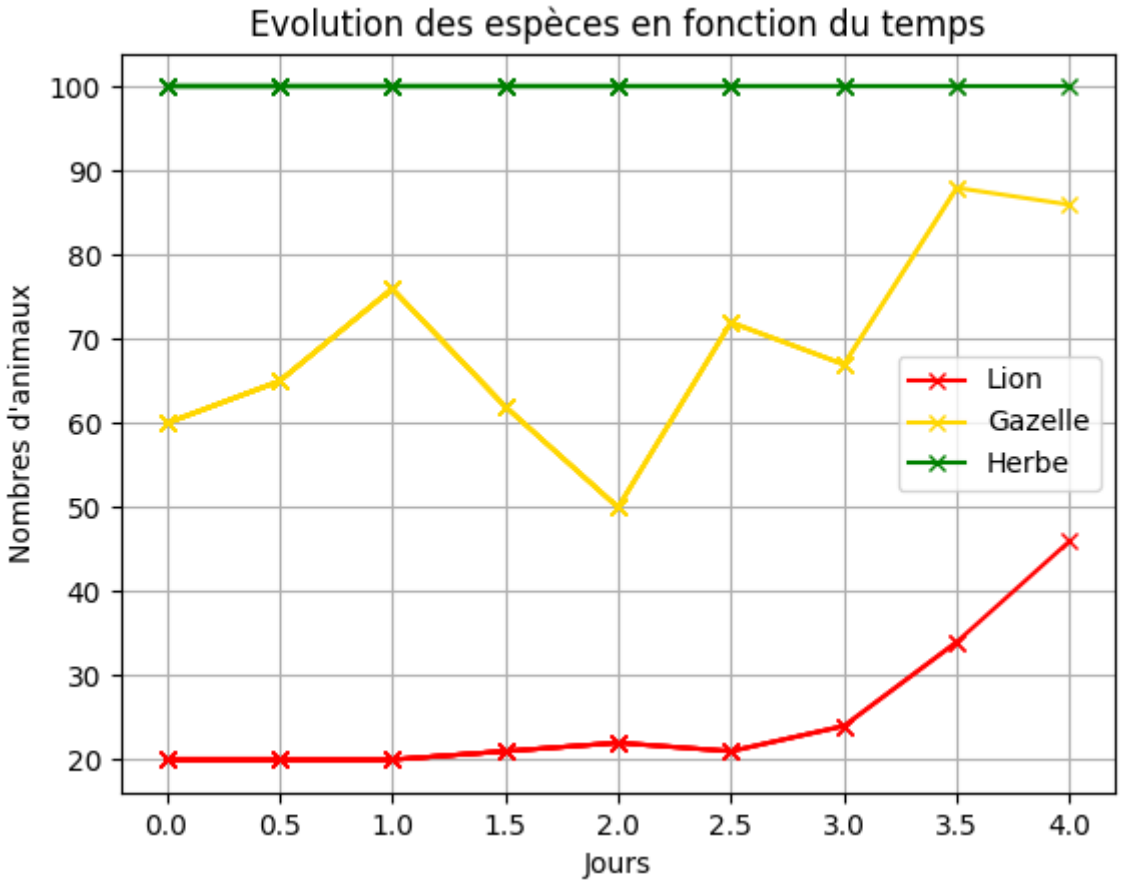
   **math.sqrt((especе.x - i.x) ** 2 + (especе.y - i.y) ** 2)**

2. La fonction parcours ensuite la liste de chaque animal sur le terrain puis, parcours une nouvelle fois chaque animal afin de déterminer la distance entre tout les animaux grâce à la formule précédemment calculée.
3. Chaque especе possède 'un champs de vision' qui à été défini dans sa classe. La fonction vérifie donc si l'espèce y est dans le champs de vision de l'espèce x grâce à la formule précedente.
4. Si l'animal y est dans la liste de nourriture de l'espèce x et que l'especе x à 'faim', alors on modifie les coordonnées de déplacement de l'espèce x afin qu'il se dirige vers l'espèce y. Sinon on appelle la fonction de déplacement aléatoire.

La fuite de l'animal lorsque celui ci est pourchassé repose sur le même principe, seule l'étape 4 change :
Si l'animal prédateur est dans le rayon de l'animal proie, alors on modifie les coordonnées de déplacement de l'aniaml proie afin qu'il s'éloigne le plus possible de l'animal prédateur, que l'animal prédateur soit en chasse ou non. Sinon on appelle la fonction de déplacement aléatoire.

# 6 Le graphique

Dans le fichier 'espece', nous avons intégrer une fonction 'graphique' qui nous permettra de suivre l'évolution du nombre d'animaux et de végétaux pour chaque espèce. Nous avons fait appelle à la bibliothèque matplotlib ainsi qu'à nos cours de calcul scientifique afin de matérialiser ce graphique. Les valeurs de celui ci sont récoltées toutes les demi journées afin d'avoir des récoltes de donnée fréquente, avec une actualisation du graphique qui se fait juste après cette nouvelle récolte. Ce graphique peut être enregistrer à tout moment grâce à la visualisation de celui-ci par un 'plt.show()'.





14

## Les difficultés rencontrées

Tout au long de la réalisation du projet, nous avons connue d'innombrable problèmes sur lesquels nous avons bloqué pendant de longues heures.

Le premier qui n'est pas des moindres, est le fait que nous connaissions pas les subtilités de l'héritage double entre classe. En effet, l'ordre d'initialisation des paramètres est très important si l'on veut éviter toutes pertes de données, chose que nous ne savions pas et qui nous à causé beaucoup de soucis au début...

Un de nos soucis majeurs était aussi le fait que nous devions différencier les êtres vivant des êtres non vivant (tel que l'herbe) pour pouvoir les sélectionner dans certaines fonctions. Encore une fois, nous sommes passé par de multiples petits algorithmes que nous avions créé, avant de connaître la fonction python 'hasattr()' qui pouvait différencier les espèces contenant un attribut 'inlive' de celles qui n'en n'ont pas. Pratique pour remplacer un algorithme de 15 lignes...

Maintenant, si on sort de l'aspect technique, nous avions eu aussi beaucoup de soucis dans la réalisation de l'image qu'on avait de notre programme. En effet, lors du début de la mise en place du jeu, nous avions un problème de reproduction massive car à chaque fois qu'un mâle et une femelle d'une même espèce se croisait, il y avait accouplement. Nous pouvions très bien nous retoruver du jour au lendemain avec une population multiplié par deux. Pour pallier à ce problème, nous avons eu l'idée d'intégrer un système de femelle 'enceinte' qui ne peuvent se reproduire qu'une seule fois par jour du temps qu'elle son enceinte, et les mâles une seule fois par pas. De plus, le nouveau née viennent au monde seulement si la mère est toujours en vie. Nous nous retrouvions donc avec une très bonne limitation d'accouplement.

Enfin, même si ce problème est loin d'être le dernier, nous étions bloqué sur le calcul d'un périmetre de 'vision' pour un animal lorsqu'il chasse ou lorsqu'il fuit. En effet, nous ne savions pas comment nous y prendre afin de calculer la distance entre deux animaux dans ce qui pouvait sembler n'être "qu'une" grille. Après de très long recherche nous avons réaliser que la grille pouvait être considérée comme un plan cartésien (donc un plan), les principes géométrque ont tout de suite suivit...

# 7 Conclusion

Si l'on se base sur nos attentes concernant ce projet, comme énoncés en début de rapport, nous pouvant fièrement dire que nous les avons atteint. En effet, c'est un simulateur qui marche, avec un bel aspect visuel et surtout avec des developpements intelligent.

Néanmoins, nous savons qu'il y a encore beaucoup de chose à ajouter pour que le programme se rapporte un maximum à la réalité, et c'est d'ailleurs une chose que nous voulions faire mais qui nécessitait beaucoup trop de recherche avec pas assez de temps. De plus, nous voulions développer un programme capable de déterminer le nombre idéal pour chaque espèce afin que le simulateur perdure le plus longtemps possible avec des espèces en vie. Nous avions même commencé à le faire mais nous avions fait des estimations sur le temps d'action du programme, comme il se comptait en jour, nous avions décidé de reporter ce projet à plus tard. Eviter l'aspect vibratoire de déplacement aléatoire été aussi un de nos objectifs. Enfin, notre but ultime été de pouvoir faire tourner une simulation d'écosystème indéfiniment en se rapproche le plus possible de la réalité en ajoutant par exemple des catastrophes naturelles, ou bien donner vie à l'herbe qui dépendrait donc des saisons et de la température, et même faire, pour un aspect graphique, défiler en fond un soleil et une lune tout au long de la journée et de la nuit.

Nous pouvons faire usage de cette application dans beaucoup d'endroits différents en entrant des paramètres plus précis. Par exemple, il pourrait très bien servir à simuler l'évolution d'animaux dans un enclos de zoo pour réguler le nombre d'animaux afin d'éviter des accouplements massif, ou dans un parc naturel afin de réguler dès le début le nombre d'animaux à intégrer dans un espace donnée avec un nombre précis de proies et de prédateurs.

Globalement nous somme très fière et satisfait de notre développement et de notre apprentissage tout au long de l'année. Nous avons pu découvrir de nouvelles techniques, de nouvelles méthodes afin de pouvoir créer un jeu intelligent. Ce jeu connaîtra sûrement des améliorations futur car même si le rendu est daté, notre enthousiasme ne l'est pas.

Amélioration : niveau moteur du programme on pourrais se pencher un peu plus sur le déplacement pour éviter l'effet vibratoire du déplacement aléatoire et trouver les bonne proportion pour faire tourner écosystème a l'infini tout en respectant les lois de la nature, on pourrais ajouter plus de facteur pour la perturbation comme le climat et un cours d'eau , donner une vie a l'herbe qui dépendra des saison,ainsi pour le graphique on pourrais histoire d'appuyer la théorie jour et nuit déjà élaborer faire défiler le soleil.

# Annexe 1 :Le manuel d'utilisation

Etape 1 : Installation des modules nécessaires sur sa machine :
— Python : voir `https://www.python.org/downloads/` pour l'installation.
— Pygame : voir `https://www.pygame.org/wiki/GettingStarted` pour l'installation.
— Matplotlib : voir `https://matplotlib.org/stable/users/installing.html` pour l'installation.
Etape 2 : Ouvirir un terminal de commande dans le dossier principal du jeu et taper la commande 'python3 test.py'.
Etape 3 : saisissez le nombre de Lion, Gazelle et d'Herbe que vous désirez au lancement du jeu, puis appuyez sur 'Play' ou 'Quittez la simulation' si vous souhaitez fermer le jeu.
Etape 4, le jeu est lancé, vous pouvez ajouter des gazelles par le clic gauche, des lions par le clic droit, ou bien supprimer tout élément du jeu par le clic de molette de souris.
Vous pouvez enregistrer le graphique si vous le souhaitez grâce au bouton 'enregistrer' de celui ci.

# Annexe 2 : Répartition des rôles

Tout le fichier 'espece' a été réalisé par Sami Mechtouf, exepté la classe 'Logo' qui à été réalisé par Elhadj Alseiny Diallo.

Tout le fichier 'grille' à été réalisé par Elhadj Alseiny Diallo exepté la fonction 'Croisement' et la fonction 'Graphique' qui ont été réalisé par Sami Mechtouf. Elhadj Alseiny Diallo a aussi réalisé le fichier 'menu'.

En effet, Elhadj Alseiny Diallo à été plus à l'aise avec la création et la gestion du terrain et le déplacement des espèces tandis que Sami Mechtouf l'étais plus avec la créations des espèces et les événements qui existent entre les espèces.

Il va sans dire que sur les parties où un élève été plus présent que l'autre, l'autre élève a toujours été là pour soutenir et travailler avec lui.

# Annexe 3 : Sources

— Toutes les images servant à représenter les espèces ont été récupérés sur le site suivant : `https://www.flaticon.com/`
— Source du son de menu : `https://youtu.be/NLYiUdWC_CA`
— Soure de son du jeu : `https://youtu.be/QpEo0M-Z22A`