Collective Movement Simulation and Analysis: the wildebeest case

groupe 10-1

Notre sujet porte sur l'étude de mouvement de foule, et plus particulièrement dans le cas des gnous. Nous avons choisi de rendre la simulation plus intéressante en ajoutant la présence de prédateurs ainsi que d'autres proies : les zèbres. Nous avons codé notre simulation dans l'objectif de répondre à la problématique suivante :

QUEL EST L'IMPACT DE LA STRATÉGIE DE MÉLANGE DES ZÈBRES AVEC LES GNOUS SUR LEUR SURVIE DANS UN MILIEU SAUVAGE ?

I. Présentation du modèle

A. Aspects généraux

Afin d'étudier le comportement de différents agents qui communiquent entre eux, on a donc implémenté notre code en Java sur Processing. Comme nous avions décidé d'étudier le comportement de gnous lors de l'apparition de prédateurs et d'autres proies, nous avons choisi de développer un modèle déjà existant sur l'environnement Processing, qui reposait sur l'algorithme de Craig Reynold. Ce modèle était composé de trois classes : Flocking, Flock et Boid. Flocking permet d'instancier toutes les espèces mises en jeu, ainsi que de les dessiner à tout instant, Flock permet de regrouper les animaux, et Boid permet de déterminer le comportement individuel des agents. Nous avons donc construit un modèle à partir de la classe Boid (que l'on a rendu abstraite) :

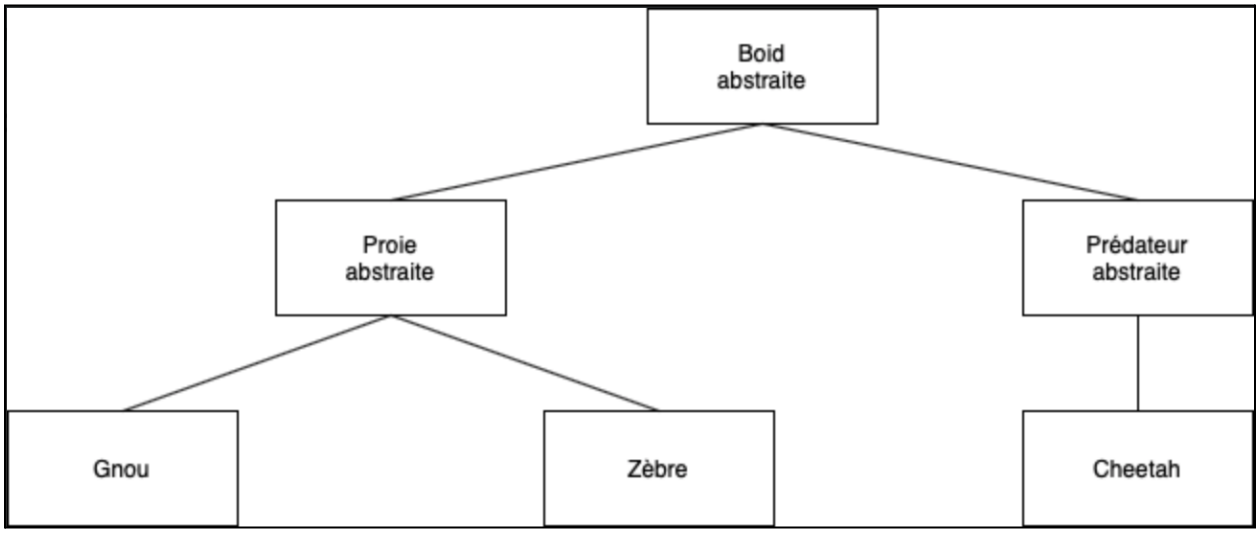

Figure 1 : Diagramme d'héritage des classes implémentées

Les classes abstraites nous ont permis de ne définir certaines fonctions abstraites sur le comportement des espèces, afin de créer des comportements différents pour proies et prédateurs en redéfinissant ces fonctions dans chaque classe correspondante. Le comportement des espèces est régi par trois règles : l'alignement, la cohésion et la séparation. L'alignement permet de calculer une vitesse moyenne sur un ensemble d'espèces afin d'avoir un mouvement uniforme au sein d'un troupeau, la cohésion permet de calculer une direction pour l'ensemble du troupeau et la séparation permet d'éloigner deux agents qui seraient trop proches.

B. Comportement des proies

Pour le comportement des proies, nous avons travaillé en deux temps. Dans un premier temps, nous nous sommes concentrés sur la fuite des proies lors de la présence d'un prédateur. Pour cela nous nous sommes appuyés sur la méthode de séparation lorsque deux agents sont trop proches. Nous avons donc agrandi significativement la distance de séparation entre proies et prédateurs (ces paramètres sont nommés *desiredSeparationPrey* et *desiredSeparationPredator* dans la classe Prey). Ensuite nous nous sommes tournés vers le comportement des troupeaux d'espèces. Afin de pouvoir étudier l'impact de la stratégie de mélange des zèbres avec les gnous, nous avons donc ajouté un paramètre d'association pour laisser l'utilisateur choisir les conditions initiales.

C. Comportement des prédateurs

Concernant le comportement des prédateurs, nous avons d'abord implémenté la chasse du prédateur, ainsi engendrant la mort d'une proie. Tout d'abord, nous avons choisi un modèle simple où le chasseur cherche la proie la plus proche, et la cible ensuite. Cela ne nous paraissait pas assez réaliste, donc nous avons décidé de mettre en place un rayon maximal de vision pour les prédateurs. Ainsi, si aucune proie ne se trouve dans ce rayon, le prédateur va se déplacer dans une direction aléatoire, ou bien vers le barycentre de l'ensemble des proies.

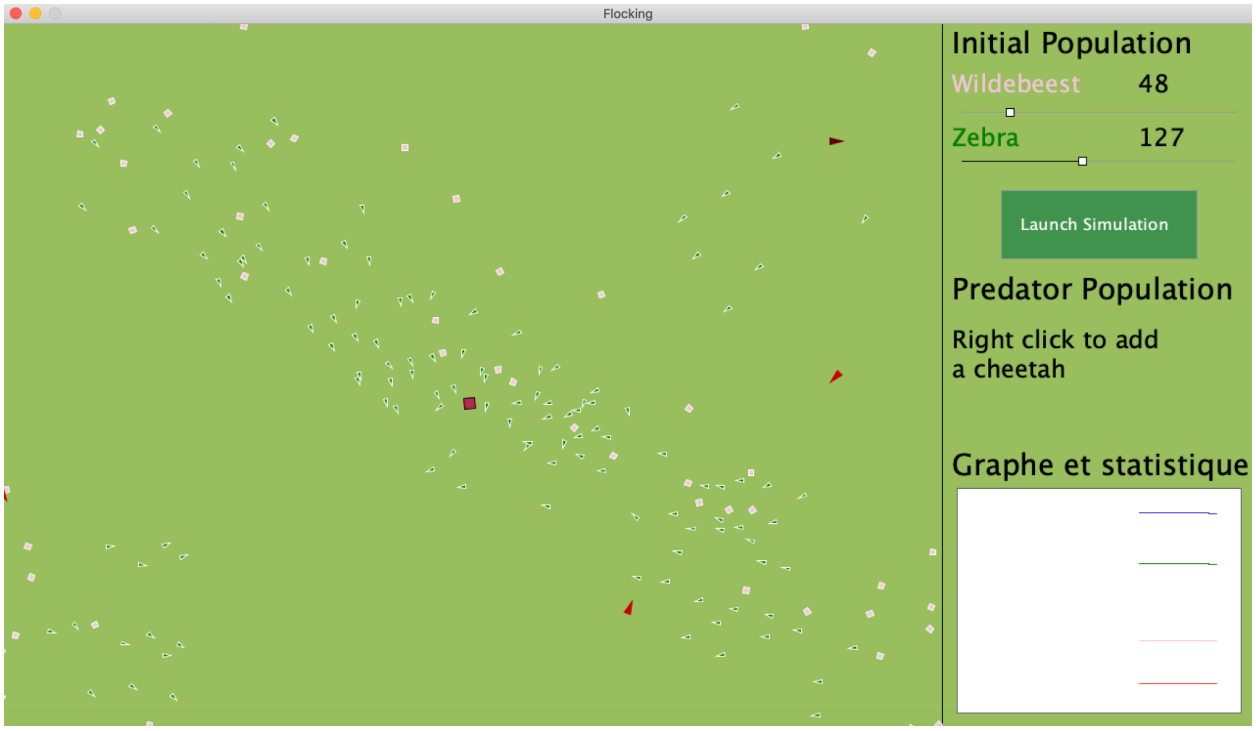

Figure 2 : lancement de la simulation
Ici le barycentre est le carré rouge, et ne prend en compte seulement les gnous.
Légende : les triangles rouges sont les prédateurs (cheetah), les triangles verts sont les zèbres et les carrés roses sont les gnous.
Les courbes de la partie graphe et statistique représentent les populations des différentes espèces, et la courbe bleu représente la population totale. Elles sont constantes ici car la simulation vient d'être lancée et aucun agent n'a été tué encore.

Dans cette même idée de vouloir rendre la simulation plus vraisemblable, nous avons par la suite ajouté une "barre de faim". Si un prédateur a une barre de faim vide, il meurt de faim. Cette barre est remplie lorsqu'un prédateur mange une proie, et se vide au cours du temps où il cherche une proie. Cela permet aux proies de survivre, et de "vaincre" les prédateurs. Les prédateurs ont aussi un champ de vision, réglé par le paramètre alpha dans la classe Predator.

II. Analyse des résultats

L'interface que nous avons mis en place permet à l'utilisateur de choisir ses propres conditions initiales (en termes de populations de proies) et ensuite d'ajouter au fur et à mesure des prédateurs. La partie graphe de cette interface permet à l'utilisateur de faire lui-même l'analyse de ses simulations, en montrant la répartition de chaque espèce dans la simulation. La barre d'association permet de choisir si les zèbres s'associent aux gnous. Ainsi, en testant sur

les deux extrêmes, on aperçoit que les zèbres survivent plus longtemps lorsqu'ils s'associent aux gnous.

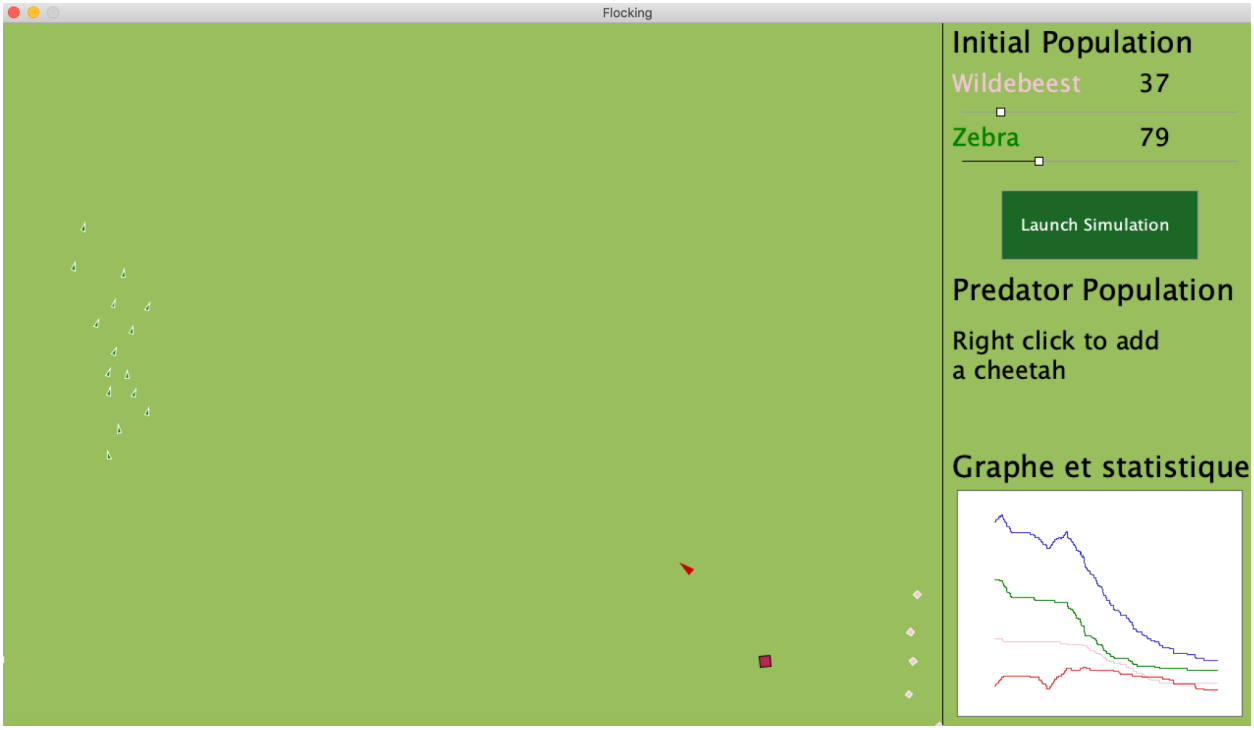

Figure 3 : Résultat possible
Lorsque le nombre de prédateurs est suffisant, le nombre de proie diminue significativement mais reste non nul. En effet, le nombre de proies étant très faible, elles ont tendance à être dispersées et ne se rejoignent pas car elles sont trop loin, et donc les prédateurs ont du mal à les attraper. Ils meurent donc de faim.

Ce résultat était prédictible au vu de notre code. En effet, nous avons codé les prédateurs de façon à ce qu'ils priorisent les gnous lors de la chasse. C'est-a-dire a distance égale d'une proie, les prédateurs se dirigeront vers un gnou et pas un zèbre. Donc si les zèbres ne forment pas leur propres troupeaux, ils auront moins de chance de se faire attaquer.

Cependant si l'utilisateur choisit de ne pas ajouter assez de prédateurs, ces derniers meurent de faim, et on observe le résultat suivant :

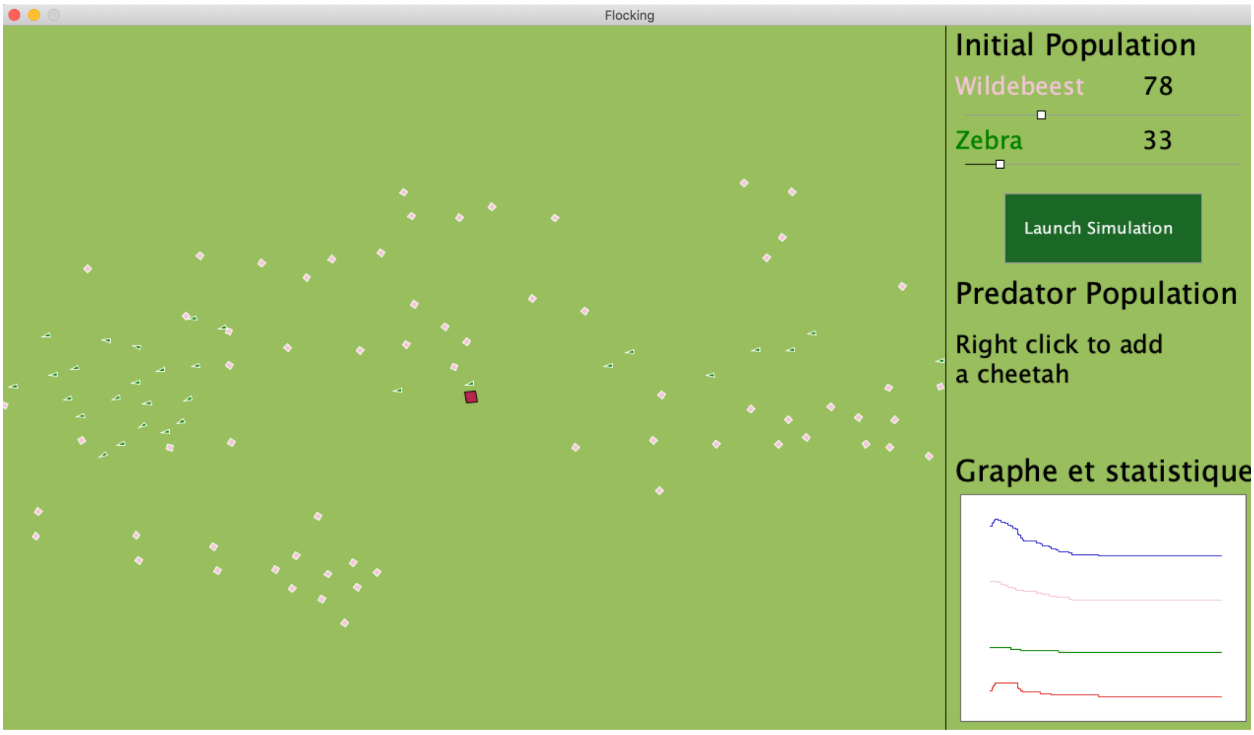

Figure 4 : résultat lorsque le nombre de prédateurs n'est pas suffisant
On observe après un certain temps des populations constantes (nous n'avons pas considéré la mort des proies autres que par un prédateur), et une population de prédateur nulle. Les proies ont donc "survécu".

III. Reflexion

Si ce projet était à refaire, nous travaillerions sûr l'intégration de nos différentes parties plus tôt. Nous avons été freinés dans notre travail par l'apparition de différents problèmes lorsque l'on a "merge" les différentes branches (ralentissement de la simulation par des problèmes de complexité, correction du comportement des agents …) Nous avons aussi rencontré des problèmes au niveau de l'affichage des agents. En effet, la représentation torique a engendré des problèmes. Lorsqu'une proie détecte la présence d'un prédateur pas loin, il lui suffit de se cacher dans l'un des coins de l'écran et ainsi elle réapparaît dans un autre coin de l'écran, loin du prédateur.

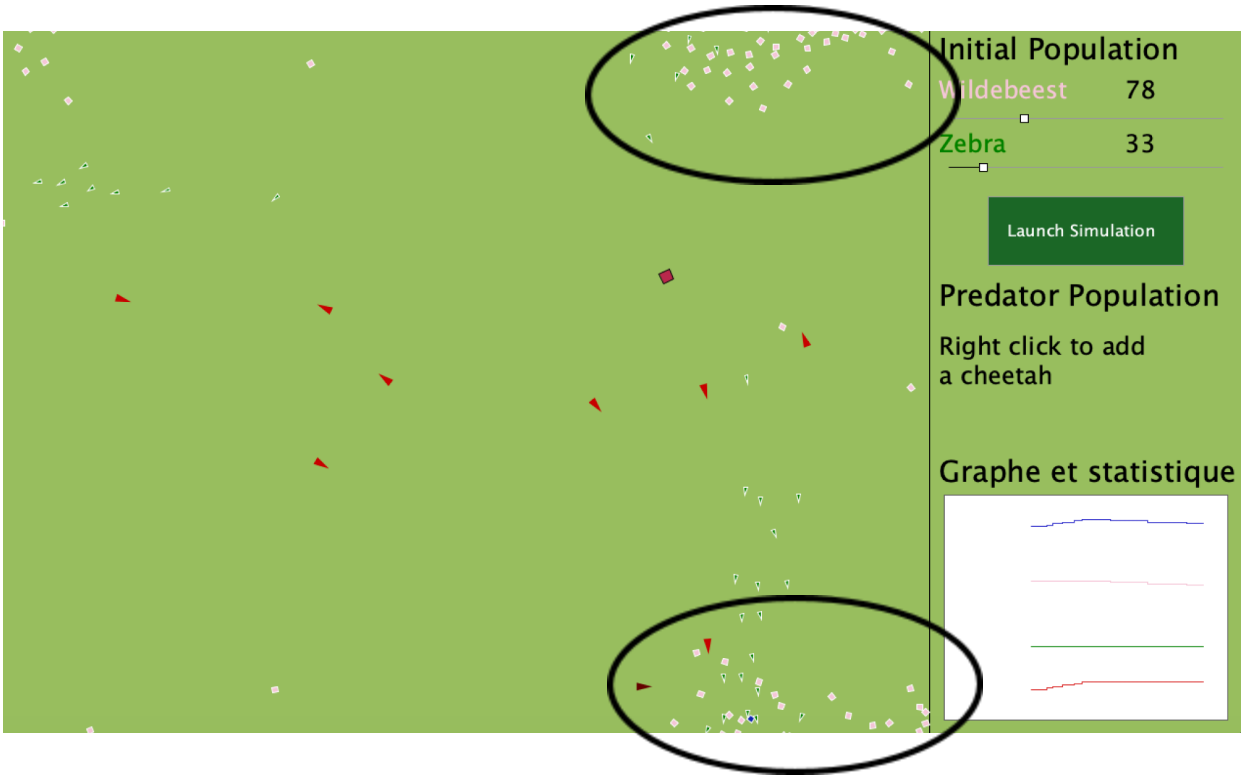

Figure 5 : représentation de la fuite des proies
Ici les gnous (carrés roses) alternent entre le coin en haut à droite de l'écran avec le coin en bas à droite, afin de fuire les prédateurs qui dans une simulation plus réaliste devraient les attraper, du fait que leur vitesse est plus grande. Mais ici la distance entre les deux coins est trop grande pour observer ce même phénomène.

IV. Répartition du travail

Servane- rédaction du poster, et de la présentation, barre d'association, comportement des proies entre espèces

Francois - partie graphe de l'interface, résolution merges / integration

Ilian - comportement des prédateurs

Ines - rédaction du rapport, interface utilisateur (sauf graphique et barre d'association), comportement des proies, résolution merges / integration , code commenté pour la partie proies + visuel