Rapport de TP : Simulation du vol des étourneaux (2D et 3D)
Nom : MOUSSAOUI Lucas
Enseignant : M. TORRES
Date : 23/05/2025

---

1 Introduction

Dans ce TP, nous avons simulé le comportement collectif des étourneaux à l'aide du modèle de **Boids** développé par Craig Reynolds. L'objectif était de modéliser un système dynamique où chaque individu applique des règles locales simples, ce qui permet de faire émerger un comportement global réaliste.

Le travail s'est déroulé en deux étapes : d'abord une simulation en **2D** avec Pygame, puis une extension en **3D** à l'aide de VPython. Des éléments extérieurs (comme un **obstacle fixe** ou une **attaque de faucon/drone**) ont été ajoutés pour observer comment le groupe réagit face à un danger ou une contrainte dans son environnement.

2 Simulation en 2D

2.1 Etude du modèle

Les boids suivent trois comportements principaux qui régissent leur déplacement :

- **Cohésion :** chaque oiseau se déplace vers le centre de masse de ses voisins proches.
- **Séparation :** il maintient une distance minimale pour éviter les collisions avec ses voisins les plus proches.
- **Alignement :** il ajuste sa vitesse et sa direction pour s'aligner sur la vitesse moyenne de son groupe de voisins.

Plusieurs paramètres influencent le comportement collectif : le rayon d'influence (distance maximale à laquelle un oiseau perçoit ses voisins), la vitesse de vol (constante ou maximale autorisée) et le nombre d'oiseaux simulés. Ces paramètres peuvent être ajustés pour modifier la densité du groupe ou la réactivité des étourneaux.

Chaque comportement est représenté par un vecteur de force. Ces vecteurs sont pondérés, puis additionnés pour donner un seul vecteur d'accélération, appliqué à la vitesse du boid à chaque frame.

2.2 Simulation de base

La simulation 2D a été réalisée avec **Pygame**, où chaque boid est représenté par un petit triangle orienté dans la direction de son déplacement. Les calculs de position, vitesse et accélération utilisent le type ‘pygame.Vector2‘, ce qui simplifie les opérations vectorielles.

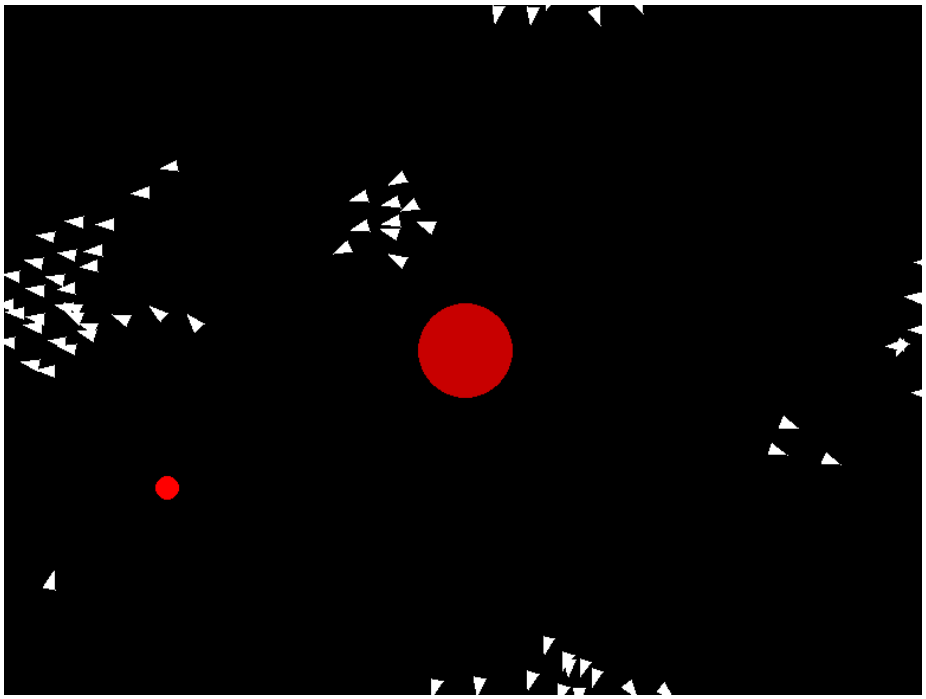

FIGURE 1 – Simulation de Boids en 2D

Les boids avancent à une vitesse constante, mais peuvent ajuster leur trajectoire grâce à un vecteur d'accélération limité. Cela évite les changements de direction trop brusques et rend le mouvement plus fluide.

2.3 Obstacle fixe

Un obstacle rouge a été placé au centre de la scène. Lorsqu'un boid entre dans un rayon critique, une force de répulsion est appliquée pour le faire dévier. Ce comportement fonctionne bien pour les boids lents, qui ont le temps d'ajuster leur trajectoire. En revanche, ceux qui arrivent à pleine vitesse peuvent parfois franchir l'obstacle avant que la correction ne soit calculée, à cause de la fréquence de mise à jour de la simulation.

2.4 Attaque d'un faucon (drone)

Un faucon rouge, contrôlé au clavier, a été ajouté à la simulation. Lorsqu'il s'approche des boids, ces derniers réagissent en appliquant une force de fuite plus intense que celle utilisée pour éviter l'obstacle fixe. Ce comportement permet de simuler une menace mobile, et on observe alors une dispersion rapide et coordonnée du groupe, comme dans la nature face à un prédateur.

3 Simulation en 3D

Pour la version 3D, j'ai utilisé **VPython**, une bibliothèque qui facilite la création d'objets en trois dimensions et la manipulation de vecteurs. Le modèle Boids a été repris tel quel, puis adapté à un espace tridimensionnel.

3.1 Représentation

Dans la version 3D, chaque boid est représenté par une sphère orange pour bien ressortir visuellement. Le fond de la scène est un bleu ciel afin d'évoquer un espace aérien. Un cube transparent encadre l'ensemble de la simulation et sert de repère visuel pour délimiter l'espace dans lequel évoluent les boids. Enfin, la caméra est placée en vue isométrique, ce qui permet d'avoir une vision globale de la scène en 3D sans devoir constamment la manipuler.

3.2 Adaptation des règles

Les trois comportements (alignement, cohésion et séparation) sont conservés tels quels dans la version 3D. La logique reste la même, mais elle s'applique cette fois à des vecteurs en trois dimensions (vector(x, y, z)). Les calculs de distance utilisent mag() (pour obtenir la norme d'un vecteur) et la direction est normalisée avec norm(). Cela permet aux boids de se déplacer de manière fluide dans l'espace, en respectant les mêmes principes que dans la version 2D.

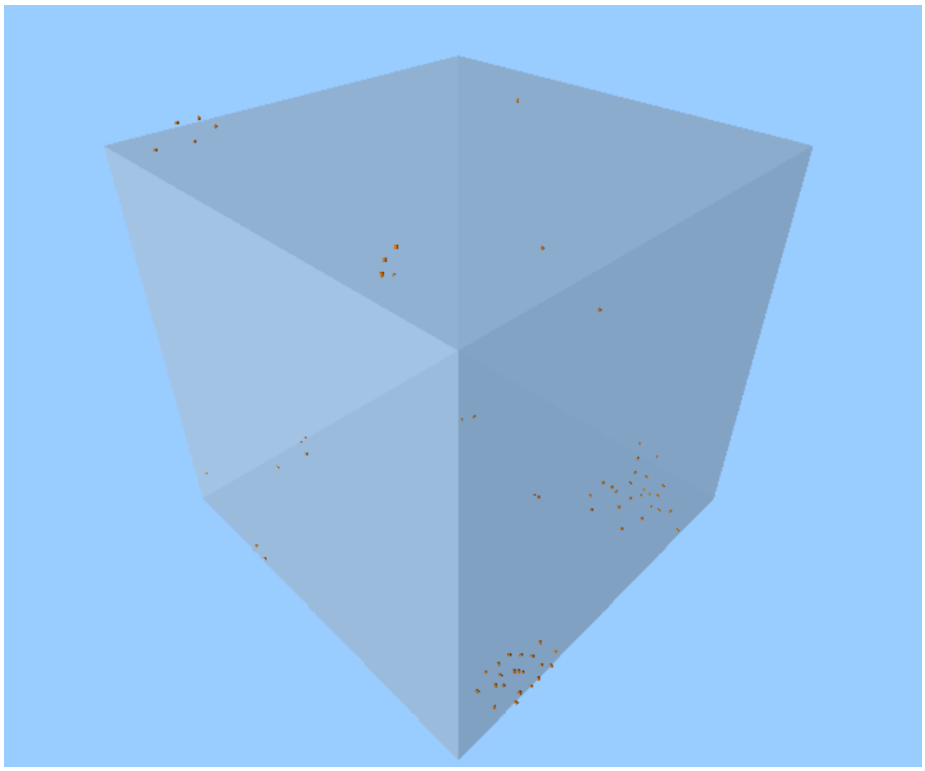

FIGURE 2 – Simulation Boids en 3D

3.3 Rebonds

Dans la version 2D, les boids sortaient d'un côté de l'écran pour réapparaître de l'autre (effet "wrap-around"). En 3D, j'ai préféré utiliser un système de rebonds contre les parois. À chaque fois qu'un boid atteint une limite du cube, sa vitesse est inversée sur l'axe concerné. Cette méthode évite qu'ils sortent du champ de vision et rend la simulation plus facile à suivre visuellement.

3.4 Améliorations possibles

Ajout d'un prédateur en 3D : comme pour la version 2D, il serait intéressant d'introduire un faucon ou un drone en 3D que les boids fuiraient dynamiquement.

Caméra mobile : suivre un boid en temps réel avec la caméra permettrait de mieux observer le comportement du groupe du point de vue d'un individu.

Affichage des axes : afficher les axes X, Y et Z dans la scène aiderait à mieux visualiser l'espace et l'orientation des déplacements.

4 Extension à un banc de poissons

Le modèle des boids utilisé pour simuler le vol des étourneaux peut aussi servir à représenter un banc de poissons, à condition d'adapter certains paramètres au contexte aquatique.

Pour commencer, les trois règles de base (alignement, cohésion, séparation) restent valables, car elles correspondent bien aux comportements collectifs observés chez les poissons. En revanche, quelques ajustements sont nécessaires pour rendre le déplacement plus réaliste :

Réduire la vitesse : les poissons bougent généralement plus lentement et avec plus de fluidité que les oiseaux. En diminuant la vitesse maximale ('MAXSPEED'), le mouvement devient plus doux et naturel.

Augmenter la densité : dans un banc, les poissons nagent souvent très près les uns des autres. On peut simuler ça en augmentant le nombre de boids et en réduisant le rayon d'interaction ('RADIUS'), ce qui les oblige à rester groupés.

Ajouter un effet de flottement : pour donner une impression de nage, on peut faire varier légèrement la hauteur ('y') de chaque poisson avec une petite oscillation (par exemple en utilisant une fonction sinusoïdale). Ça crée un effet de balancement agréable à regarder.

Réagir à l'environnement : on peut aussi imaginer une source de lumière qui influence le déplacement, ou un prédateur (comme un requin ou un sous-marin) qui ferait fuir le groupe. Ce serait similaire à l'ajout du faucon dans la version 2D.

En résumé, avec quelques modifications simples, on peut facilement adapter le modèle pour donner l'illusion d'un véritable banc de poissons en mouvement.

5 Conclusion

Ce TP m'a permis de mieux comprendre comment un comportement collectif peut émerger à partir de règles simples appliquées localement. J'ai pu adapter le modèle des boids à différentes situations, comme l'ajout d'obstacles, de prédateurs ou encore le passage à la 3D.

Le code étant bien structuré, ça m'a aussi facilité les tests et les modifications. Pygame et VPython se sont révélés très pratiques pour mettre en place des simulations visuelles sans trop de complexité.

Annexe

Le code source est disponible ici : github.com/Lucasmsai/boids$_simulation.git$