# PAF : Multi-scale tracking of Collective Movement

Groupe 16.2:
Mathis DUPONT, Vincent HERFELD, Benjamin TERNOT, Inès VARHOL

June 2022

## Table des matières

# I Prise en main, généralités

Pour utiliser notre simulation, il faut télécharger Processing (`https://processing.org/download`) et notre dépôt git (`https://gitlab.telecom-paris.fr/PAF/2122/gr16-2`).

Il faut ouvrir Main.pde du dossier Main dans Processing, puis lancer le programme. On peut mettre en pause et exporter les données en cliquant dans la fenêtre de simulation. Les données (les positions des barycentres des groupes à chaque frame) sont exportées dans traj.csv qui est également présent dans le dossier Main.

On peut choisir les paramètres suivants (variables globales en haut du fichier main.pde) :

— le nombre de boids

— le nombre de frames par seconde (c'est-à-dire le nombre d'update de positions des boids)

Dans la fonction setup de Main.pde, le positionnement initial des boids est déterminé, ainsi que la taille de la fenêtre de simulation.

Voici la liste des commandes dans notre simulation finale :

— clic de souris : met en pause et exporte les données en .csv

— touche "p" : place un obstacle à l'endroit du curseur

— touche "r" : enlève tous les obstacles

— touche "t" : affiche/cache les trajectoires des groupes

# II Déplacement des boids

## II.1 Réutilisation d'un code

Nous avons réutilisé une simulation présente sur le site de Processing (`https://processing.org/examples/flocking.html` codé par Daniel Shiffman). Cette simulation affiche des boids qui se déplacent sous l'effet de trois forces :

— cohésion : les boids ont tendance à se rapprocher des boids proches

— séparation : les boids gardent une distance minimale entre eux

— alignement : les vecteurs vitesse des boids proches ont tendance à être dans la même direction

Dans Boid.pde, les méthodes `cohesion, separate, align` renvoient un vecteur qui représente la force associée. Les méthodes `flock, applyForce` et `update` mettent à jour la position des boids en fonction des forces.

Les boids se déplacent comme sur un tore, si un boid sort du côté droit, il réapparait du côté gauche (méthode `borders` dans Boids.pde).

## II.2 Vision des voisins

Dans notre simulation, nous gardons ces trois forces, mais nous avons changé la façon de trouver les voisins d'un boid, les calculs de distance entre boids et nous avons ajouté une 4ème force qui modélise des obstacles (voir 6.).

A l'origine, dans le calcul de chaque force pour chaque boid A, les distances entre A et tous les autres boids sont calculées pour déterminer quels sont les voisins de A. Nous utilisons une

carte qui retient les positions de tous les boids à chaque frame. Pour connaître ses voisins, un boid parcourt toutes les cases proches dans la carte grâce à la méthode `searchNeighbours` dans Boid.pde.

### II.3 Distance sur un tore

Nous avons changé le calcul de distance du programme d'origine pour que la distance corresponde à la "réalité physique" du tore.

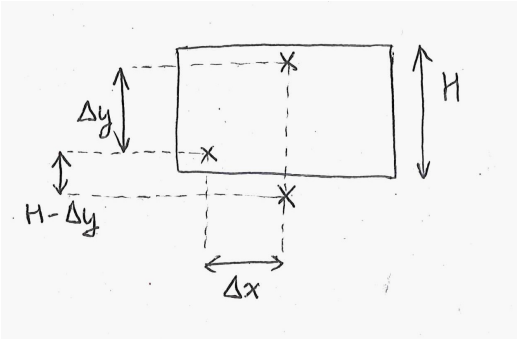


FIGURE 1 – Calcul de la distance entre deux points

Par exemple, dans la figure 1, on a pour les deux points choisis :

— distance classique = $\sqrt{\Delta x^2 + \Delta y^2}$

— distance réelle = $\sqrt{(\Delta x)^2 + (H - \Delta y)^2}$

La formule générale pour calculer la distance sur notre écran de taille (width = $W$, height = $H$) est donc :

$$\mathrm{d} = \sqrt{min(\Delta x, W - \Delta x)^2 + min(\Delta y, H - \Delta y)^2}$$

## III Clustering

Nous avons longuement réfléchi aux règles pour identifier les groupes de boids. Voici les différentes façons que nous avons envisagé : considérer que deux boids suffisamment proches sont dans le même groupe, ou bien considérer en plus si leurs vecteurs vitesse sont suffisamment alignés, ou bien définir les groupes à partir de boids "de coeur" avec un nombre minimal (par exemple 3) de voisins.

### III.1 Clustering par distance uniquement

Nous avons implémenté deux versions de "groupement par distance" qui fonctionnent sur ce modèle. Une première version (commit SHA 05a71e) qui a l'inconvénient de ne pas attribuer un indice unique à chaque groupe, et une deuxième (commit SHA 20f0cc) qui corrigeait ce défaut. Nous avons défini les règles suivantes pour qu'un groupe A conserve le même identifiant :

- il doit contenir la majorité des membres du groupe A du frame précédent
- les anciens membres du groupe A doivent être majoritaires dans le nouveau groupe

L'inconvénient du "groupement par distance", c'est que parfois deux groupes de boids avec des directions opposées peuvent se rapprocher sans pour autant que les groupes ne fusionnent.

### III.2 DBscan

```
public class Factorial
{
    public static void main(String[] args)
    {   final int NUM_FACTS = 100;
        for(int i = 0; i < NUM_FACTS; i++)
            System.out.println( i + "! is " + factorial(i));
    }

    public static int factorial(int n)
    {   int result = 1;
        for(int i = 2; i <= n; i++)
            result *= i;
        return result;
    }
}
```

### III.3 Décentralisation

Que ce soit le groupement par distance ou le groupement type DBscan, nous avons implementé des versions décentralisées du code.

Pour cela, nous utilisons la méthode `searchNeighbours` de Boid.pde qui permet à chaque boid de repérer les boids voisins. Cette méthode se base sur la lecture d'un tableau de boids qui les localise et qui est mis à jour à chaque frame.

De plus, dans le cas du groupement par DBscan, la décentralisation implique

## IV Outils d'analyse

Dans cette section nous allons aborder les différents outils construits permettant d'analyser le comportement des boids une fois les groupes formés.

### IV.1 Calcul de barycentre et vecteur vitesse de groupe

Nous avons décidé dans un premier temps, de caractériser un groupe par son isobarycentre. Pour un groupe de taille $N$ avec $b(i)$ le vecteur position du $i$-ieme boid du groupe :

$$barycentre = \frac{1}{N} \sum_{i=1}^{N} b(i)$$

Toutefois comme nos boids se déplacent sur un espace torique, le calcul du barycentre dans un plan ne correspond pas toujours au centre du groupe que l'on voit sur l'écran. C'est pourquoi nous avons calculé un barycentre adapté.

On considère séparément les abscisses et les ordonnées des points. En temps normal, on place les valeurs sur un segment [0, width] et on calcule leur valeur moyenne : ici, on place les valeurs sur un cercle et on calcule la direction moyenne.

Ainsi, l'abscisse du barycentre d'une liste de points d'abscisse $x_i$ est

$$barycentre(x_i) = \frac{width}{4} + \frac{width}{2\pi} * arctan\left(\frac{\sum_{i=1}^{N} cos(\frac{2\pi x_i}{width})}{\sum_{i=1}^{N} sin(\frac{2\pi x_i}{width})}\right) + \frac{width}{2} * int(\sum_{i=1}^{N} sin(\frac{2\pi x_i}{width}) < 0)$$

On a utilisé la fonction $atan2$ pour implémenter le calcul dans le code.

## IV.2 Conservation des trajectoires

Une fois les barycentres calculés il est simple de conserver les trajectoires de chaque groupe. Il suffit de placer dans un `ArrayList<PVector>` l'ensemble des barycentres associés à chaque groupe. Pour cela nous avons pris un dictionnaire qui a le numéro de groupe pour clef à laquelle on associe la liste des barycentres du groupe.

Lorsque l'on fixe un état initial il peut arriver que l'on place plusieurs boids sur une même case (exemple : séparation de tous les boids en deux points). Sachant que l'algorithme de recherche de voisins `searchNeighbours()` pour un boid $b$ ne considère seulement les boids qui ne sont sur une autre case que celle de $b$, nous avons décidé de négliger les groupes formés qui disparaissent avant la $30^{eme}$ frame de l'acquisition. Après cette frame, nous conservons tous les groupes (qu'ils disparaissent ou non) dans le dictionnaire décrit plus haut.

Une dimension importante pour nous est le temps. Nous avons décidé de rester fidèle à cette notion et pour cela avant la création d'une nouvelle trajectoire de groupe (c'est-à-dire à la création d'un nouveau groupe) nous plaçons dans le dictionnaire des trajectoire au niveau de la clef du nouveau groupe, des points (-10,-10) afin de faire du padding. On place exactement `frameCount - 30` points (-10,-10).

A chaque frame le dictionnaire de rempli, et lorsque l'on clique sur l'écran d'acquisition, la simulation se met en pause et une image (le dictionnaire des trajectoires) de l'état actuel est sauvée dans un document `.csv`.

Ce document pourra donc etre exploité dans un tableur classique de type Excel. Cependant, nous avons aussi mis au point un programme Processing permettant d'observer seulement les trajectoires de groupes acquisent dans un intervalle de temps donné par l'utilisateur. Il suffit de modifier les valeurs de `startFrame` et de `endFrame` dans `Graphs.pde`. A noter que le `endFrame` sera automatiquement limité par la durée de l'acquisition dont est issu le document `traj.csv`.

## IV.3 Mode analyse en temps-réel

Nous avons mis au points un mode d'affichage qui permet d'observer en temps-réel le tracé des trajectoires de groupes ainsi que les vecteurs vitesse de chaque groupe. Pour activer (ou désactiver) ce mode il suffit de tapper sur la touche `"t"` de son clavier.