Projet algorithmique et programmation — Simulation de nuées

Guillaume Martin

IFIE1 – Octobre 2024 – IMT Mines Albi

Table des matières

1 Consignes

Ce travail s'effectue en binôme. Le rendu se fera sur Campus sous la forme d'une archive (`.zip`, `.rar`, `.7z`, `.tgz`...) **portant les deux noms du binôme** (un seul dépôt par binôme).

Contenu attendu dans l'archive :

1. un document (en **PDF**) contenant :
   (a) les noms et prénoms des membres du binôme,
   (b) une analyse des problèmes auxquels vous avez été confrontés,
   (c) les algorithmes mis en œuvre,
   (d) un descriptif de ce que vous avez programmé expliquant les choix que vous avez faits,
2. vos scripts python.

Les éléments qui **ne doivent pas être** dans cette archive sont : votre environnement virtuel (le répertoire `venv`) et d'éventuels répertoires `__pycache__`.

2 Objectif

L'objectif de ce projet est de simuler le mouvement d'une nuée d'oiseaux ou d'un banc de poissons dans un espace en deux dimensions.

3 La simulation

3.1 Physique en 2D

Dans notre simulation, nous considérons un espace en deux dimensions dans lequel se déplacent des entités, qui peuvent être des oiseaux (appelés Boids) par exemple. Cet espace est continu, ce qui signifie que les déplacements des entités se feront selon la logique :

$$\mathbf{v} = \mathbf{a} * dt \tag{1}$$

$$\mathbf{x} = \mathbf{v} * dt \tag{2}$$

avec **a**, **v** et **x** correspondent respectivement à l'accélération, la vitesse et la position. dt correspond au laps de temps écoulé depuis la dernière boucle de simulation.

Vous noterez que vous disposez d'une classe `Vector`. Celle-ci ne contient pas toutes les fonctionnalités nécessaires pour terminer le projet mais nous vous fournissons les noms des méthodes ainsi qu'une description de ce qu'elles doivent faire.

3.2 Représentations cartésiennes et polaires des vecteurs

Pour ce projet, il sera parfois utile de passer d'une représentation cartésienne des vecteurs en (x, y) à une représentation polaire (r, θ) telle que

$$r = \text{norme}(\text{Vecteur}(x, y)) \tag{3}$$
$$\theta = \text{atan2}(y, x) \tag{4}$$

ou bien,

$$x = r\cos(\theta) \tag{5}$$
$$y = r\sin(\theta) \tag{6}$$

Pour plus de détails sur la fonction `atan2`, reportez-vous à sa documentation dans le module math de Python.

3.3 Attributs des boids

Par défaut, les boids possèdent une position, une vitesse et une accélération qui sont tous des instances de la classe `Vector`. Les boids peuvent détecter la présence d'autres boids dans un rayon de détection. Les boids sont également limités par une vitesse maximale qu'il faudra prendre en compte lors de la mise à jour de leur état. Le reste des attributs déjà existants servent à l'affichage des boids.

3.4 Règles de déplacement

Les oiseaux se déplacent selon trois règles, qui représentent chacune une force appliquée à chaque oiseau :

— l'évitement,
— le groupement,
— l'alignement.

Évitement Chaque boid souhaite éviter de rentrer au contact avec les autres boids. Pour cela, il évalue la distance $\vec{d}$ avec les boids $\mathcal{B}$ dans son rayon de détection et calcule

$$\overrightarrow{f_{\text{évitement}}} = \frac{1}{N}\sum_{b\in\mathcal{B}} \frac{\vec{d}}{\|\vec{d}\|^2} \tag{7}$$

Cette force est ensuite divisée par le nombre N de boids dans $\mathcal{B}$.

Groupement Chaque boid cherche à se rapprocher des autre boids. Pour cela, il calcule la position moyenne des autres boids $\mathcal{B}$ dans son rayon de détection et retranche sa propre position $\vec{x}$

$$\overrightarrow{f_{\text{groupement}}} = \frac{1}{N}\sum_{b\in\mathcal{B}} \vec{x}_b - \vec{x} \tag{8}$$

Alignement Chaque boid essaie également d'aligner sa direction avec la direction moyenne des boids qu'il aperçoit. On a alors

$$\overrightarrow{f_{\text{alignement}}} = \frac{1}{N}\sum_{b\in\mathcal{B}} \vec{v}_b \tag{9}$$

La force résultante est donc une somme pondérée des forces précédentes. Il n'y a pas de contrainte sur la valeur des coefficients de pondération de chacune des forces.

3.5 Gestion du voisinage

Chaque boid considérera uniquement les autres boids dans un rayon de détection fourni dans la ligne de commande.

3.6 Traitement des bordures

Par défaut, les boids se déplacent dans un monde torique. Cela signifie qu'un boid qui dépasse une des quatre bordures de la simulation doit ré-apparaître de l'autre côté.

3.7 Gestion de la nuée

Initialisation des boids Il faut générer n boids aléatoirement dans l'espace de simulation, sachant que n est un paramètre fourni par l'utilisateur dans la ligne de commande. Pour générer un boid, il faut tirer uniformément les composantes de sa position dans l'espace, puis les composantes de sa vitesse initiale uniformément dans $[0, 1[$.

Boucle de la nuée Pour ce projet (et pour la majorité des simulations multi-agent), il faut pour chaque boid :

- décider de la force à appliquer au boid par rapport aux autres puis,
- gérer les bordures de la simulation puis,
- mettre à jour l'état des boids via la fonction update,
- et enfin mettre à jour l'affichage via la fonction draw que nous avons rédigée pour vous.

Attention la longueur d'une période de simulation n'est pas fixe : elle dépend d'un pas de temps variable *dt* fourni par la fonction clock de pygame et qu'il faudra prendre en compte dans la mise à jour "physique" des boids.

4 Éléments techniques

4.1 Gestion du hasard

Les choix de placements initiaux des boids sont effectués par des tirages aléatoires. Pour cela, vous utiliserez le module random. Voici quelques fonctions qui pourraient vous servir :

- `random.seed(seed)` – permet de choisir la graine définissant la suite pseudo-aléatoire utilisée (à n'appeler qu'une seule fois en début de script). Le paramètre seed peut être un nombre ou une chaîne de caractères. Pour utiliser une graine différente à chaque exécution, vous pouvez prendre le résultat de la fonction time_ns() du module time.
- `random.random()` – retourne un nombre flottant aléatoire dans l'intervalle $[0, 1[$.
- `random.randint()` – retourne un entier aléatoire dans un intervalle défini par vous.

4.2 L'environnement virtuel

Dans le cadre de ce projet, vous devrez créer un environnement virtuel et y installer les modules pygame (pour la gestion du jeu et de l'affichage) et numpy (pour les produits matriciels qui permettent les rotations en 2D). Il n'est pas nécessaire de mettre cet environnement virtuel dans le rendu final.

4.3 Exécution minimale du script

Le script main.py qui gère le lancement du projet doit accepter au minimum et dans cet ordre les arguments suivants :

- `detection_radius`, le rayon de détection des boids,
- `alignment_weight`, le coefficient de la force d'alignement,
- `cohesion_weight`, le coefficient de la force de groupement,
- `avoidance_weight`, le coefficient de la force d'évitement,
- `num_boids`, le nombre de boids à simuer,
- `seed`, la graine pour les tirages aléatoires

Attention, le projet doit être exécutable en ligne de commande et pas seulement avec la fonction « Debug » de VSCode !

4.4 Utilisation des modules fournis

Vous pouvez réutiliser l'intégralité du code fourni dans l'énoncé dans votre rendu final.

5 Ce qu'il faut produire

Vous serez amenés à écrire du code dans chacun des fichiers fournis. Veillez à ne pas changer les noms des fichiers pour des raisons de facilité de correction, s'il vous plaît. Votre projet sera exécuté via le fichier main, en recevant les arguments en ligne de commande décrits plus haut et dans le même ordre. Ne modifiez pas les fonctions d'affichage et n'utilisez pas d'autres modules que ceux proposés dans l'énoncé !

Il vaut mieux écrire de nombreuses méthodes ou fonctions courtes que peu de fonctions et méthodes très longues (application du précepte « *diviser pour régner* »).

Mises à part d'éventuelles constantes (dont, par convention, la valeur n'est jamais modifiée), vous ne devez pas utiliser de variables globales.

6 Pour aller plus loin...

Pour ceux qui souhaitent aller plus loin, vous pourrez proposer un ou plusieurs autres scripts (dérivés de ceux de base) en ajoutant une ou plusieurs fonctionnalités au script initial.

Voici des idées de fonctionnalités supplémentaires :

1. prendre en compte des arguments supplémentaires sur la ligne de commandes si nécessaire (voir les idées suivantes).
2. remplacer la gestion de base des bordures pour la rendre plus « naturelle » en ajoutant une force d'évitement supplémentaire liée à la distance avec les bords de la simulation. Cette fonctionnalité est plus facile à visualiser en prenant en compte une border plus petite que la taille de la fenêtre de jeu de manière à voir les boids « tourner ».
3. limiter la rotation maximale admissible par chaque boid à chaque mise à jour.
4. remplacer la détection dans un rayon par la détection dans un cone de vision avec un angle de vision défini par vous.
5. ajouter un vent constant influençant tous les boids dans l'espace de simulation.
6. prévoir un arrêt automatique de la simulation après un certain temps en secondes.

7 Évaluation

Pour chacune des fonctions ou méthodes que vous aurez codées vous-même, vous devrez fournir l'algorithme associé.

Pour les schémas bulle, les objets reçus en paramètre et modifiés durant l'exécution des instructions de l'algorithme devront apparaître dans la partie gauche et dans la partie basse du schéma bulle. Ne dupliquez pas votre code Python dans votre rapport (il sera dans votre archive).

Voici des éléments d'appréciation qui nous serviront pour évaluer votre travail.

Concernant le rapport :
- la correction de vos schémas bulle,
- le respect des notations algorithmiques (telles que définies dans les supports de cours et de TP),
- la logique et la qualité de votre analyse et de vos explications,
- la logique et la clarté de vos algorithmes,
- le respect des spécifications de cet énoncé,

Concernant le code Python :
- le respect des spécifications de cet énoncé,
- le respect des bonnes pratiques de Python,

— la qualité des commentaires ou des chaînes de documentation,
— la conformité du code Python avec les algorithmes,
— la pertinence du nommage des variables, des fonctions, des méthodes, des classes...

Nous prendrons aussi en compte les variantes et autres améliorations que vous proposerez (si vous en proposez) ainsi que les analyses ou explications liées aux expérimentations réalisées.