Projet Pansemant Nano-robots

Description

L'objectif de ce projet est de simuler un système sanguin qui contiendrait des nano-robots qui auraient pour mission principale d'aider à la cicatrisation d'anévrismes (de brèches dans des vaisseaux sanguins).

Le comportement des nano-robots serait un système mutli-agent qui contiendrait deux types d'agents :

- Les **Plaquettes** (Platelet en anglais) qui auraient pour mission de colmater les brèches en se regroupant autour de celles-ci. Elles se déplacent en essaims pour rester groupées et couvrir le maximum d'espace pour détecter au mieux les brèches.
- Les **Messagers** qui auraient pour mission de trouver des brèches et de les signaler aux groupes de Plaquettes. Ils se déplacent en solitaire, plus vite et sont capables de cartographier leur route avec un mémoire tampon pour transmettre les routes optimales aux groupes de plaquettes.

Installation

Pour lancer le projet, il vous faudra installer les dépendances contenus dans le fichier `requirements.txt`. Avec la commande suivante :

```
pip install -r requirements.txt
```

Puis lancer :

```
python main.py
```

Utilisation:

- `ESC` : Quitter
- `Espace` : Pause/Play
- `v` : Afficher/Masquer la vision des agents
- `l` : Afficher/Masquer les noms des agents (Fait ralentir la simulation)
- `n` : Afficher/Masquer les lignes de voisinage
- `o` : Afficher/Masquer les lignes d'obstacles
- `s` : Afficher/Masquer la vélocité des agents de la simulation
- `h` : Afficher/Masquer les commandes dans le HUD
- `LEFT/RIGHT` : Diminuer/Augmenter la vitesse des agents
- `UP/DOWN` : Diminuer/Augmenter la vitesse de lecture de la simulation

Fonctionnement

Les Plaquettes

Les Plaquettes se déplacent selon un algorithme de **Flocking** qui leur permet de rester groupées et de se déplacer ensembles à une bonne distance les uns des

autres pour couvrir le plus de surface possible.

L'algorithme de Flocking que j'utilise les règles suivantes :

- **Séparation** : Les agents s'écartent des autres agents trop proches.

Pour chaque voisin, si la distance entre l'agent et le voisin est inférieure à une distance de séparation que l'on veut (90% du rayon de vision de l'agent), on soustrait le vecteur formé par la position de l'agent et la position du voisin à la vélocité de l'agent.

- **Cohésion** : Les agents se dirigent vers le centre de masse de leurs voisins.

Pour chaque voisin, on ajoute la position du voisin à un vecteur somme. On divise ce vecteur par le nombre de voisins pour obtenir le centre moyen des voisins, puis on soustrait la position de l'agent à ce centre de position pour obtenir un vecteur qui pointe vers ce centre moyen.

- **Alignement** : Les agents essaient de s'aligner avec la direction de leurs voisins pour se déplacer dans la même direction.

Pour chaque voisin, on fait la somme de leur vecteur de vélocité que l'on norme par leur nombre pour obtenir un vecteur moyen de vélocité. Ce vecteur obtenu nous sert de direction que l'on ajoute au vecteur de vélocité de l'agent pour qu'il s'aligne avec ses voisins.

- **Esquive des obstacles** : Faire en sorte que les agents évitent les obstacles.
- **Limite de vitesse** : Limiter la vitesse des agents pour éviter que leur vitesse ne s'accèlère pas à l'infini puisque nous faisons la somme des vecteurs de vélocité pour avoir la direction que l'agent doit prendre à la prochaine itération.

Les obstacles

Les obstacles sont des objets issus de la même classe parents que les agents mobiles. La différence est qu'ils ont une vélocité nulle qui n'influe donc pas sur leur position. Ainsi seul la règle de **séparation** s'applique dans leur intéraction avec la vélocité des Plaquettes. Ces obstacles sont sencés être les parois des vaisseaux sanguins. Cependant aucune "carte" n'a été implémentée pour tester les interactions entre les agents et obstacles de manière plausible.

Les Messagers et les Trous

Les messagers et les trous n'ont pas été implémentés. L'idée est de créer un système de communication entre les agents pour qu'ils puissent s'affecter une brèche à colmater en transmettant sa taille et sa position (ou le chemin pour y accèder) qui serait actualisé par les aller-retours des messagers qui exploreraient des chemins à la recherche d'itinéraires plus courts et actualiseraient à

chaque passage le nombre de plaquettes requises pour optimiser la répartition de Plaquettes dans le système sanguin.

Fichiers

- `main.py` : Fichier principal qui lance la simulation en initiant et actualisant les agents et les obstacles.
- `boid.py` : Contient la définition de classe des agents de la simulation. Comme les Plaquettes, les obstacles et supposemment les Messagers et les brèches.
- `utils.py` : Contient des fonctions utilitaires pour la simulation.
- `variables.py` : Fichier de configuration de la simulation. Contient les variables globales ainsi que les poids des règles de Flocking.