UNIVERSITE DE NANTES
proposé par Eric LANGUÉNOU
UFR Sciences – Dpt d'Informatique

Émergence et agents autonomes. Distanciel

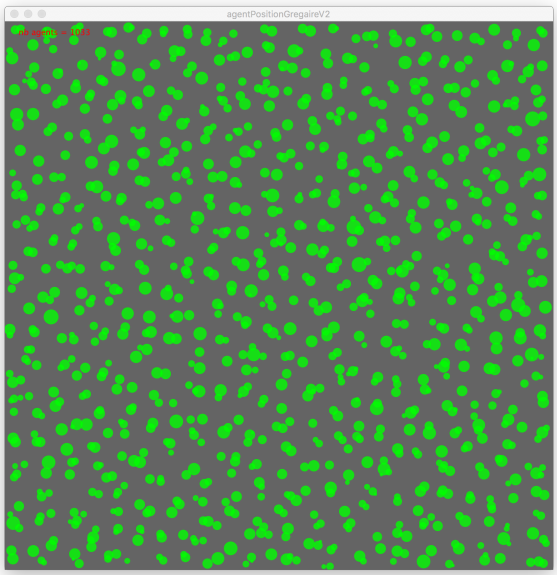

FIGURE 1 – Agents grégaires

1 Travail en distanciel

Ce travail est à faire en autonomie, à partir du cours magistral. Il est à rendre sur madoc dans la zone de dépôt correspondante, partie création numérique.

2 Sujet : agents grégaire et phagocytose

1. Il s'agit d'étudier et de coder un algorithme sous processing qui implémente un comportement multi-agent simplifié.
2. les agents ne possèderont pas de vitesse, ni d'accélération, mais uniquement une position et un diamètre.
3. mise à jour des agents :
   — chaque agent se déplacera vers l'agent le plus proche d'un pourcentage de la distance les séparant.
   — si deux agents sont à une distance en dessous d'une limite (à fixer), il y a phagocytose. Les deux agents sont remplacés par un seul agent dont le diamètre est tel que la somme des aires des deux agents donne l'aire de l'agent résultant.

3 Affichage statique de la population

Dans la suite, et pour commencer, on étudiera et codera :
— le stockage de la population
— son affichage

3.1 Stockage de la population

1. on utilisera deux listes, une pour la position et une pour le diamètre. La position de l'agent i sera donnée par le i^e élément de la liste de positions. Le diamètre par le i^e élément de la liste de diamètre. Ci dessous, quelques exemples d'utilisation de listes (pour cette question et les suivantes) :

```
//.............declaration
List<PVector> posList; // la liste des positions des agents
List<Float> sizeList; // la liste des tailles des agents
//.............  initialisation (new arrayList() efface aussi la liste au besoin)
posList = new ArrayList<PVector>() ;
sizeList = new ArrayList<Float>() ;
//.............  ajout d'un élément dans une liste
posList.add(P); // si P est un PVector
//.............  récupération de l' élément à la place i
P = posList.get(i)
//.............  modification de la position de l'élément i
posList.get(i).add(addP);// si addP est un PVector
//.............  suppression de l'élément à la place i
posList.remove(i);
```

2. écrire une fonction `initAgents` qui allouera les listes et les remplira. Bien entendu, si vous êtes à l'aise avec les classes java, n'hésitez pas à en créer.

3.2 Affichage de la population

1. écrire une fonction `drawAgents` qui affiche tous les agents de la population.

4 Déplacement des agents

Dans une nouvelle version de l'algorithme, on ajoute les déplacements des agents.

1. analyser et écrire une fonction qui, pour un agent donné, rend l'index de l'agent le plus proche ;
2. analyser et écrire une fonction `updateAgent` qui déplace un agent donné par son index ;
3. analyser et écrire une fonction `updateAllAgents` qui déplace tous les agents de la population ;
4. appeler cette dernière fonction à la fin de la méthode `draw()` de processing ;

Pendant le développement, il peut être utile de représenter graphiquement certaines informations lors de ces calculs, comme par exemple dessiner une ligne entre un agent et celui le plus proche. La méthode `line()` de processing peut ainsi être appelée dans `updateAgent`. Faire les tests avec très peu d'agents au début.

Examiner le comportement émergent. Etait-il prévisible ?

5 Phagocytose

Dans la dernière version de l'algorithme, on ajoute la phagocytose.

Deux agents situés à une distance inférieure à une limite à fixer sont transformés en un seul agent. On respectera la conservation des aires, i.e. la somme des aires des 2 agents donne l'aire de l'agent résultant. Pour la position de l'agent résultant, on peut prendre la moyenne des positions ou celle d'un des agents. Si si ils sont très proches, la différence ne se verra pas.

Le plus simple est de supprimer les deux agents et d'en créer un nouveau. Attention, il faut supprimer d'abord celui qui possède l'index le plus grand et ensuite l'autre. Si on supprime l'index le plus petit, l'index le plus grand va changer dans la liste après suppression. Dans les meilleurs cas, on supprime un agent qui ne le devait pas, dans le pire, cet index n'existe plus et le programme plante : accès en dehors de la liste.

6 Travail optionnel : Ajout de couleurs, ajout d'agents

1. On ajoute à présent la propriété couleur à chaque agent. Un tirage aléatoire de couleur à chaque création d'agent, une nouvelle liste de couleur. Lors de la phagocytose, on prend la moyenne des couleurs ou une moyenne pondérée par les aires.
2. Ajouter la création d'un nouvel agent (ou plusieurs) lors d'un clic de souris. La fonction `mouseClicked()` est automatiquement appelée lors d'un click de souris. Il suffit donc, comme pour keyPressed(), d'ajouter cette fonction dans le code.

```
void mouseClicked(){
   //mettre ici les opérations voulues
}
```

7 Rendu du travail

Le travail devra être rendu sur la zone dépôt *distanciel* de madoc, partie création numérique. Il contiendra votre répertoire de sketch sous forme compressée. Vous respecterez les consignes ci-dessous :

— nommage des variables consistant avec leur contenu ;
— nommage des méthodes indiquant ce qui s'y fait ;
— commentaires expliquant les zones de codages et les points critiques ;

Si vous le désirez, vous pouvez aussi rendre un texte explicatif.