Projet C++ 3BIM 2013
Mouvement Collectif : Les Boids

–

10 janvier 2013

1 Présentation du projet

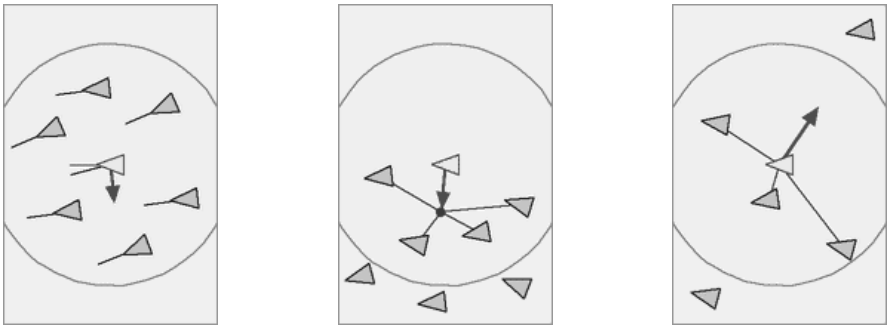
Les simulations de mouvement collectif – les boids – sont apparues à la fin des années 80 comme un exemple de Vie Artificielle. Basée sur des règles d'interaction locales (entre les individus), cette modélisation a permis de mettre en évidence une propriété globale de déplacement en apparence concerté. Ce modèle a été appliqué avec différents succès à des modèles de flottes d'oiseaux, de bancs de poissons, de troupeaux etc.

Le but du projet est de concevoir puis de programmer un simulateur permettant d'étudier ces systèmes dans deux cas. Le cas standard abondamment étudié dans la littérature, et celui avec perturbation introduite par un ou plusieurs prédateurs.

2 Le Modèle

Il s'agit de modéliser une propriété *émergente* issue d'un mécanisme local : à savoir les règles d'interactions entre les agents.

Elles peuvent se résumer par le diagramme suivant



Alignement Cohésion Séparation

Règle 1 Les agents s'alignent sur la vitesse moyenne du rayon de perception

Règle 2 Les agents vont se diriger vers le centre de masse du groupe

Règle 3 Les agents s'éloignent des autres agents trop près (ainsi que des obstacles)

Mathématiquement, nous supposerons que nous avons N agents formant une population dans un environnement 2D. Chaque agent (indexé par i) possède une position $\overrightarrow{x_i} \in \mathbb{R}^2$ et une vitesse $\overrightarrow{v_i}$. Les agents possèdent un rayon de perception r.

2.1 Déplacements des agents

Il faut appliquer certaines règles pour décider du mouvement de chaque agent. Il y a tout d'abord une règle évidente :

$$\overrightarrow{x_i}(t+dt) = \overrightarrow{x_i}(t) + dt\overrightarrow{v_i}(t) \tag{1}$$

La vitesse de l'agent résulte ensuite de l'application d'un ensemble de M règles (ou comportements) que doit suivre l'agent :

$$\overrightarrow{v_i}(t+dt) = \overrightarrow{v_i}(t) + dt\sum_{j=1}^{M}\gamma_j\overrightarrow{a_i^j}(t) \tag{2}$$

Avec $\gamma_j \in \mathbb{R}^+$ des paramètres du modèle permettant d'ajuster l'importance relative de chacun des comportements.

Tous les agents devront suivre trois règles de base ($\overrightarrow{a_i^1}$, $\overrightarrow{a_i^2}$ et $\overrightarrow{a_i^3}$) dont la valeur sera calculé en fonction des autres agents dans le rayon de perception de l'agent ou d'éventuels obstacles.

Règle 1 Les agents s'alignent sur la vitesse moyenne du rayon de perception. Soit

$$\overrightarrow{a_i^1}(t) = \frac{1}{K} \sum_{j, j \neq i} (\overrightarrow{v_j}(t) - \overrightarrow{v_i}(t)) \tag{3}$$

les agents j sont ceux qui sont tels que $||\overrightarrow{x_i} - \overrightarrow{x_j}|| < r$ et sont au nombre de K.

Règle 2 Les agents vont se diriger vers le centre de masse du groupe Soit

$$\overrightarrow{a_i^2}(t) = \frac{1}{K} \sum_{j, j \neq i} (\overrightarrow{x_j}(t) - \overrightarrow{x_i}(t)) \tag{4}$$

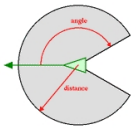
les agents j sont ceux qui sont tels que $||\overrightarrow{x_i} - \overrightarrow{x_j}|| < r$ et sont au nombre de K.

Règle 3 Les agents s'éloignent des autres agents trop près (ainsi que des obstacles) soit c la distance de contact (a priori $c < r$) alors

$$\overrightarrow{a_i^3}(t) = -\frac{1}{K'} \sum_{k, k \neq i} (\overrightarrow{x_k}(t) - \overrightarrow{x_i}(t)) - \frac{1}{O} \sum_{o} (\overrightarrow{o} - \overrightarrow{x_i}(t)) \tag{5}$$

les agents k sont ceux qui sont tels que $||\overrightarrow{x_k} - \overrightarrow{x_j}|| < c$ et sont au nombre de K'. Les objets o sont les obstacles à distance $< c$ de l'agent i (et sont au nombre de O).

Optionnel : On pourra étudier le comportement du modèle lorsque le voisinage de chaque agent est restreint à un angle de vue α et une distance r, les agents hors de l'angle de vue n'étant pas pris en compte leur du calcul des règles de déplacement.



2.2 Espace

Les agents se déplacent dans un espace 2D. Dans un premier temps, pour calibrer les paramètres et vous assurer que votre simulateur fonctionne bien, vous utiliserez un espace rectangulaire clos (les bords de la fenêtre de simulation). Afin que les agents n'en sortent pas, vous devrez ajouter *un comportement de répulsion supplémentaire* $\overrightarrow{a_i^4}$ dont la formulation est libre.

Optionnel : Si le reste du projet fonctionne bien (cas standard et introduction d'un prédateur), vous testerez le comportement du modèle dans un espace 2D avec conditions périodiques aux limites (i.e. un tore). L'on veillera à correctement adapter les calculs de centre de masse et de distance entre agents : deux agents se trouvant aux bords droits et gauche de l'espace affiché (la fenêtre de l'application) sont en fait proches dans l'espace simulé.

2.3 Obstacles

On ne considérera que des obstacles circulaires de rayon r_o possiblement variable dont la répartition est libre.

2.4 Populations

Les agents seront rassemblés en populations reliées par de relations de prédations et pouvant avoir différentes propriétés (vitesse de déplacement, paramètres γ_j, etc.). On utilisera donc une seule population d'agents pour le cas standard et deux populations (la deuxième ne contiendra qu'un seul agent) dans le cas du prédateur.

Optionnel : Étudier les déplacements d'agents répartis en 3 populations ou plus ; la population 2 étant prédatrice de la population 1 et proie de la population 3 par exemple.

3 Cas étudiés

Nous allons étudier deux principaux cas de simulations. Pour ces deux cas :
- Les simulations se feront en dimension 2.
- Les agents seront donc dans un espace rectangulaire de dimension $W \times H$ (avec conditions aux limites périodiques optionnelles)
- Il faudra une sortie graphique *en temps réel* de chaque simulation (pour cela vous utiliserez la classe `bwindow`)

3.1 Simulation Standard

Vous implémenterez la simulation pour obtenir des comportements d'ensembles des agents. Il s'agit donc de critères visuels. Il faudra donc bien penser à implémenter la partie graphique pour faire les tests. Dans ce cas, il n'y a rien de particulier par rapport à la situation décrite dans le modèle. Il faut juste déterminer les différents γ_j ainsi que leur influence sur le comportement global. Une fois cette simulation mise au point, vous pourrez ajouter des obstacles. A titre de rendu « fixe », vous imprimerez le champ de vitesse moyen avec et sans obstacles.

3.2 Introduction de prédateurs

Dans cette simulation, nous introduisons un nouvel agent : le prédateur (il doit être rattaché à une population dans la simulation). Celui-ci va avoir des règles très simples
- S'il n'y a personne autour de lui, il se déplace aléatoirement (de manière isotrope)
- S'il y des proies autour de lui, il se dirige avec une vitesse constante $\overrightarrow{v_P}$ vers la plus proche.

Les proies vont bien sûr essayer d'éviter le prédateur. Pour cela il suffira d'ajouter un comportement supplémentaire $\overrightarrow{a_i^4}$et un paramètre γ_4 de sorte que

$$\overrightarrow{a_i^4} = -\frac{(\overrightarrow{x_P} - \overrightarrow{x_i})}{||(\overrightarrow{x_P} - \overrightarrow{x_i})||} \tag{6}$$

On considèrera que le prédateur attrape la proie lorsqu'il est à une certaine distance (petite mais pas trop) de l'agent. A ce moment là, le prédateur reste immobile un certain temps puis recommence. La formulation de $\overrightarrow{a_i^4}$ pourra être modifiée dans le cas de plusieurs prédateurs.

L'objectif est de regarder le mouvement collectif lorsque celui-ci est perturbé par l'ajout de prédateur. Notamment, en plus de la sortie visuelle, vous imprimerez le champs de vecteur vitesse moyen à partir de l'attaque du prédateur.

3.3 Variables à étudier

En plus du comportement visuel et des champs de vecteurs vitesses moyens, on pourra étudier l'évolution des variables suivantes en fonction des différents paramètres (γ_j) et du cas de simulation (standard, avec simple prédateur, avec population de prédateurs, etc.) :
- Vitesse moyenne (et écart type) des agents (pour chaque population)
- Nombre moyen (et écart type) de sous-groupes (agents ayant des directions semblables et étant proches les uns des autres)

Vous êtes libres d'implémenter les mesures qui vous paraissent pertinentes.

4 Le Projet

Le projet se fera de la manière suivante :

1. Vous serez par trinôme **inhomogène**.
2. Il est à finir pour le début du mois de février 2013.
3. Il se compose d'un tarball contenant le code source, le diagramme des classes et d'éventuelles images présentant les résultats de votre étude (champs de vecteur vitesses, comportement spécifique, etc.).

4. Une validation sera organisée lors de la dernière séance, vous me présenterez votre programme, son architecture, une partie de son code ainsi que vos conclusions (effets des différents paramètres, comportements particuliers, etc.).

4.1 Le code

Le code sera obligatoirement en C++. Les sources principales (classes et main) se trouveront dans un répertoire `src` avec un `makefile` créé par vos soins (supportant au minimum les commandes `make` et `make clean`).

Les différents cas de simulations doivent pouvoir être chargés dans votre programme depuis un fichier en utilisant l'option `-load path`, avec `path` le chemin d'accès vers un fichier. Ces fichiers seront dans un répertoire `sim` et porteront des noms un minumum descriptifs (e.g. `cas_standard_obstacles_1.txt`, ou `predateur_1.txt`, mais pas `fichier_1_lol.txt`).

Vous ajouterez un répertoire `doc` contenant votre diagramme des classes, les images que vous aurez pu générer pendant votre étude, et un fichier `readme.txt` présentant succinctement votre programme, les fichiers de simulations (du dossier `sim`), et les images présentes dans le dossier `doc`.

Je vous conseille de commencer tôt à faire les spécifications et la description des classes. Je vous conseille aussi de vous répartir le code une fois les spécifications complètes. Vous avez toute latitude pour le modèle de votre projet (en termes de classes).

En outre, je n'ai pas précisé la valeur des paramètres. Ce sera à vous d'explorer et d'expliquer leur influence. Notez aussi qu'il faudra réfléchir à d'éventuelles optimisations pour que le code ne soit pas trop lent.

4.1.1 Les fichiers de simulations

On préfèrera des fichiers au format ASCII (lisibles par un éditeur de texte) et contenant au minimum les informations suivantes :

- le nombre de populations ;
- le nombre d'agents pour chaque population ;
- les paramètres des agents ;
- le nombre d'obstacles.

Vous êtes bien sûr libres de sauvegarder plus d'informations dans ces fichiers (positions exactes des agents, des obstacles, type de conditions aux limites, etc.) afin d'augmenter la reproductibilité de vos simulations. La sauvegarde / lecture de ces fichiers sera faite par votre programme au moyens des options `-save path` et `-load path`. Vous pouvez utiliser la bibliothèque de gestion de fichiers de votre choix. Je vous conseille personnellement d'utiliser les objets `ifstream` et `ofstream` de la STL (cf. la doc.).

4.2 La validation

On organisera des validations lors de la dernière séance (et peut être un peu au delà si nécessaire). Cela consistera simplement à me décrire l'architecture de votre programme (diagramme des classes, tests unitaires, algorithmes principaux, etc.), à présenter quelques cas de simulations en temps réel (avec l'option `-load path` de votre programme), puis à discuter de l'influence des différents paramètres et cas de simulations.

La validation et le tarball seront notés et détermineront une part de votre note de DS comprise entre 5 et 10 (la part, pas votre note).

N'hésitez pas à apporter les modifications qui vous plaisent au modèle, à explorer l'influence de différents paramètres et à implémenter des algorithmes pour mesurer des variables qui vous paraissent pertinentes.

Bon projet !