Simulation d'un vol de boids

Introduction à OpenGL/GLUT : TP

Nikolas Stott

16 février 2018

Le but de ce TP est de simuler en C++/OpenGL/GLUT un vol de boids (ou bird-oids, similaire à des oiseaux) contraint à des règles de déplacements, d'environnement et de comportement. La première partie constitue le coeur du TP : l'implémentation des classes nécessaires au fonctionnement du programme et l'affichage de l'évolution du vol avec OpenGL/GLUT. La seconde partie propose plusieurs pistes possibles pour améliorer la simulation : meilleur modèle de boid, nouvelles règles de déplacement, interaction avec l'utilisateur, etc.

Afin de simuler le mouvement coordonné d'animaux, tels les vols d'oiseaux ou les bancs de poissons, Craig Reynolds a introduit en 1986 la notion de *boid* (bird-oid). L'idée centrale du comportement des boids est qu'à chaque instant, la direction qu'un boid choisit de prendre ne dépend que de la position et de la vitesse de ses voisins.

Le programme `main` aurait une structure similaire à :

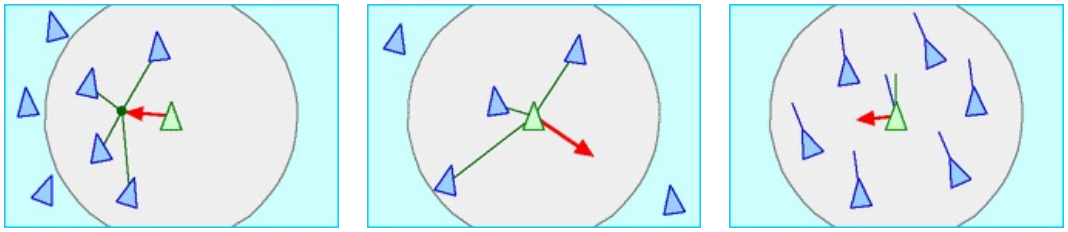
```
initialise_positions()
LOOP
        draw_boids()
        move_all_boids_to_new_positions()
END LOOP
```

La fonction `initialise_positions()` dispose les boids à leur position initiale, la fonction `draw_boids` appelle le dessin d'une image (ou *frame*) de l'animation et la fonction `move_all_boids_to_new_positions`, qui contient l'algorithme effectif, déplace les boids avant de reprendre la boucle.

1 Les boids : 3 règles simples pour se déplacer

Dans un premier modèle, nous pouvons considérer que les boids ne sont soumis qu'à 3 règles de comportement :

- **règle de cohésion** : les boids tendent à se déplacer vers la position moyenne de leurs camarades,
- **règle de séparation** : les boids ne souhaitent pas entrer en collision,
- **règle d'alignement** : les boids tendent à se déplacer dans la direction moyenne de leurs camarades.



(a) Cohésion (b) Séparation (c) Alignement

FIGURE 1 – Les 3 règles des boids

En pseudo-code,cela donne :

```
PROCEDURE move_all_boids_to_new_positions()
        Vector a1, a2, a3
        Boid b
        FOR EACH BOID b
                a1 = rule_cohesion(b)
                a2 = rule_separation(b)
                a3 = rule_alignment(b)
                b.acceleration = limit_acceleration(a1+a2+a3)
                b.speed = limit_speed(b.speed + dt *b.acceleration)
                b.position = b.position + dt * b.speed
        END
END PROCEDURE
```

On pourra dans un premier temps inclure la totalité des boids camarades dans le calcul de chacune des règles, puis dans un second temps limiter ce calcul aux camarades les plus proches uniquement.

1.1 Règle de cohésion :

La règle de cohésion permet, comme son nom l'indique, au vol de rester groupé et (si l'on ajoute des obstacle) de permettre aux boids éloignés de rejoindre le groupe. Si N

voisins du boid $\mathcal{B}$ sont $\mathcal{B}'_1,\ ...\ \mathcal{B}'_N$, on définit le centre de masse perçu par $\mathcal{B}$ par

$$c = \frac{1}{N}\big(\mathcal{B}'_1.position + ... + \mathcal{B}'_N.position\big)$$

La contribution de la règle de cohésion s'écrit alors (constante de proportionalité de l'ordre de 0.01) :

$$a_{cohesion} \ \propto \ (c - \mathcal{B}.position)$$

1.2 Règle de séparation :

Afin d'éviter les collisions, chaque boid $\mathcal{B}$ va chercher s'il y a d'autres boids dans son voisinage (plus proche que quelques fois la taille d'un boid), par exemple $\mathcal{B}'$. Afin d'obtenir un effet de répulsion lisse au cours du temps, nous allons faire agir une force répulsive sur $\mathcal{B}$ (et $\mathcal{B}'$ subira la même lorsque le calcul s'appliquera à lui) de type élastique (constante de proportionalité de l'ordre de 0.5) :

$$a_{separation}[\mathcal{B}' \to \mathcal{B}] \ \propto \ \big(\mathcal{B}.position - \mathcal{B}'.position\big)$$

1.3 Règle d'alignement :

Cette règle permet d'apporter une inertie de groupe aux boids, qui ne se contentent plus de rester autour d'un centre de masse fixe mais qui se déplace dans l'espace. Cette règle est très similaire à la règle de cohésion, mais à la place de moyenner les positions, nous moyennons les vitesses. Si N voisins du boid $\mathcal{B}$ sont $\mathcal{B}'_1,\ ...\ \mathcal{B}'_N$, on définit la vitesse de groupe perçue par $\mathcal{B}$ par

$$v = \frac{1}{N}\big(\mathcal{B}'_1.speed + ... + \mathcal{B}'_N.speed\big)$$

La contribution de la règle d'alignement s'écrit alors (constante de proportionalité de l'ordre de de 0.2) :

$$a_{cohesion} \ \propto \ (v - \mathcal{B}.speed)$$

2 Des règles supplémentaires

2.1 Tendance à se diriger vers une cible

Nous allons donner un objectif aux boids : se déplacer vers une cible fixe (un nid ?). Pour cela, il suffit d'ajouter une nouvelle règle de comportement au calcul d'évolution. Si la cible est notée $\mathcal{C}$, on peut par exemple influencer le déplacement des boids en modifiant leur vitesse par (constante de proportionalité de l'ordre de 0.1) :

$$v_{target} \ \propto \ \frac{\mathcal{C}.position - \mathcal{B}.position}{\|\mathcal{C}.position - \mathcal{B}.position\|}$$

2.2 Autres idées de règles (Optionnelles)

D'autres idées de règles de comportement peuvent être :
— ajouter du vent (constant ou aléatoire),
— imposer une vitesse maximale aux boids,
— contraindre la position à un espace restreint (une boite par exemple),
— éviter des obstacles,
— séparer les boids en prédateurs/proies et simuler une chasse,
— si le boid touche le sol, il se pose et attend avant de re-décoller,
— etc.

3 Implémentation en C++/OpenGL/GLUT

3.1 Interaction utilisateur

Afin d'ajouter un élément d'interactivité, nous proposons d'ajouter des boids dans le vol en cliquant dans la fenêtre : le nouveau boid apparaît sur le sol et rejoint ses camarades.
Optionnel : D'autres possibilités sont :
— déplacer la cible/placer des obstacle avec 'clic souris'+'touche clavier appuyée',
— contrôler la caméra dans la scène : faire orbiter la caméra autour d'un point central (la cible par exemple),
— etc.

3.2 Modélisation avec OpenGL/GLUT

Dans un premier temps, afin de tester la mise en place d'OpenGL/GLUT et la correction de l'implémentation, il peut être intéressant de ne modéliser les boids que par une forme géométrique simple (un triangle, un quadrilatère).

Dans un second temps, nous proposons d'utiliser les fonctions OpenGL/GLUT pour modéliser les éléments de notre scène : boids, cible, obstacles, etc.
Voici quelques suggestions de modèles simples à recréer :
— **Boids** : tétraèdre, modèle simple d'oiseau (corps ovale, ailes triangulaires), etc,
— **Cible** : Cube, boule, nichoir, etc,
— **Obstacle** : Cube, Sphere, Cylindre, arbre, etc.

Ensuite, on peut orienter les boids dans leur direction de déplacement, leur faire battre des ailes, coloriser la scène, ajouter des éclairages, un cycle jour-nuit, ...

Références

1. http ://www.kfish.org/boids/pseudocode.html
2. http ://www.red3d.com/cwr/boids/