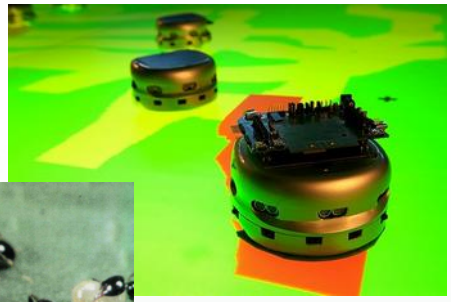
CentraleSupélec

Mineure, 3ème année

Vie artificielle (TP 2)

Déplacement réactif en essaim



Alexis Scheuer & *Olivier Simonin*

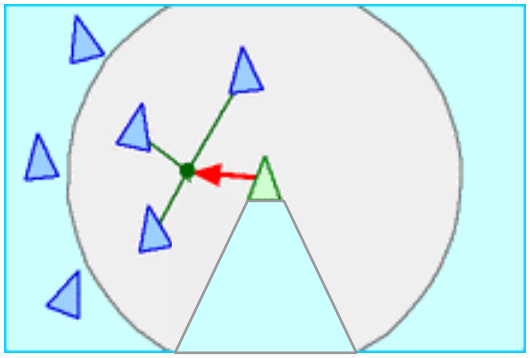
Maître de conférences UL (FST) / Loria

UNIVERSITÉ DE LORRAINE

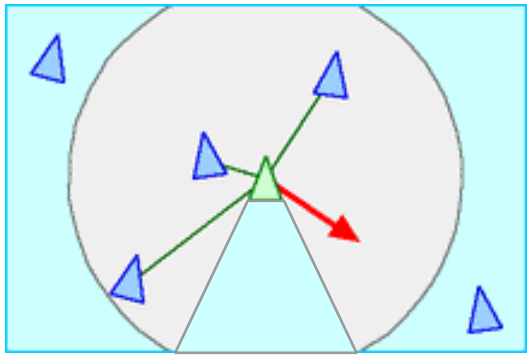
Loria
Laboratoire lorrain de recherche en informatique et ses applications

Flocking

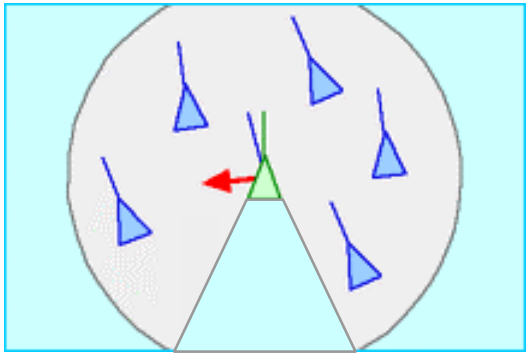
- Les boïds de Reynolds → reproduire les nuées d'oiseaux (1987)
- Combinaison de trois forces / perceptions locales



cohésion



séparation



alignement

Principes du Flocking

Problème : naviguer en essaim

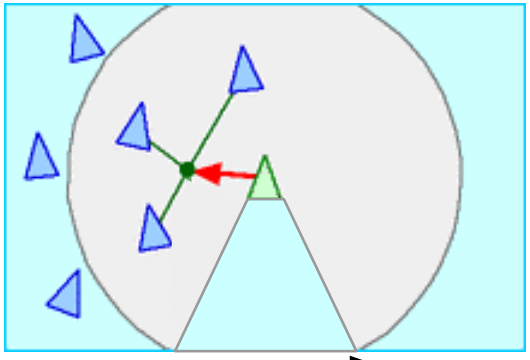
Navigation réactive : $\vec{V} = \vec{Att} + \Sigma_i g_i.\vec{Rep}_i$

Flocking : $\vec{V} = g_1.\vec{C} + g_2.\vec{A} + g_3.\vec{S}$

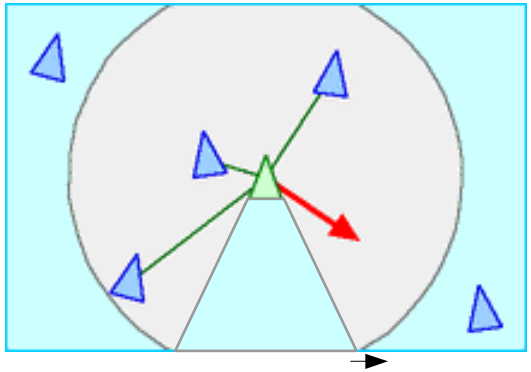
Objectifs

Calcul des vecteurs $\vec{C}$, $\vec{S}$ et $\vec{A}$

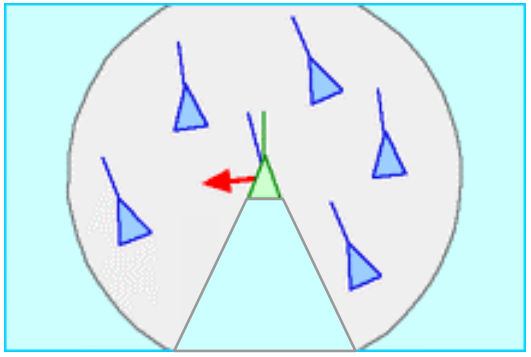
Recherche expérimentale des coefficients g_1, g_2 et $g_3 \in [0,1]$ ($g_1 + g_2 + g_3 = 1$)



cohésion : $\vec{C}$



séparation : $\vec{S}$



alignement : $\vec{A}$

Comportement réactif

Calcul vectoriel :

- Coût calculatoire très léger
- Hypothèses fortes sur les perceptions embarquées
- Ajout d'un bruit → brisure d'équilibres

Boucle de contrôle :

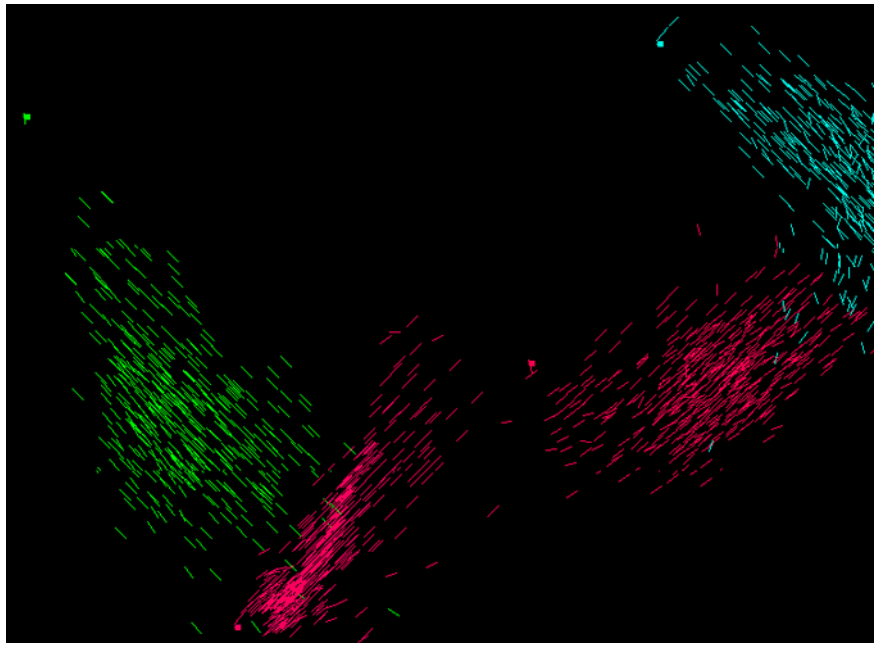
- Haute fréquence
- Perception → Décision nouveau cap → Orientation

Simulation : outil multi-agent TurtleKit

Démonstration *Bees* du simulateur Madkit :

[Ferber, Gutknecht, Michel] `www.madkit.net`

- Espace 2D continu
- Lancements d'agents *Queen*
  → déplacement aléatoire
- Lancements d'agents *Bee*
  → suivent la reine



Des abeilles aux robots mobiles

Modification de la démonstration *Bees* :

- Espace 2D continu
- Agents *Queen* & *Bee* → robots leaders ou suiveurs

Modification apportée aux suiveurs :

si (leader)
alors évitement de collision
sinon comportement Flocking

si (! leader)
alors ajouter cohésion & alignement
ajouter séparation

Codage des robots suiveurs

Comportement :

Modification de la méthode `react()` de la classe `Robot`

Calcul vectoriel :

Définition des 3 vecteurs `vectors[]` (sépar., cohés. & align.) et des coefficients au démarrage

Perceptions et actions :

- Perceptions fournies par l'environnement (dans le repère du robot) : position, orientation & couleur des voisins
  `Detection[] perceptions = robotWorld.getPerceptions(id);`
- Vecteurs pondérés & combinés, transformé en accélérations et envoyé à l'environnement…

Étapes de codage et d'expérimentation

1. Calculer **la direction de répulsion** → `vectors[0]`

   tester et régler avec plusieurs suiveurs (`vectorsCoef` = {1, 0, 0})

2. Calculer **la direction moyenne de cohésion** → `vectors[1]`

   bis (`vectorsCoef` = {0, 1, 0} ou plutôt {1, 1, 0})

3. Calculer **la direction d'alignement** → `vectors[2]` (ter)

4. Étudier **le comportement** en fonction des paramètres

   - Faire varier les coefficients `vectorsCoef`, avec un puis plusieurs leaders
   - Modifier le rayon de perception `Rpercep`