HMIN233 - Algorithmes d'exploration et de mouvement

Flocking et évitement

Suro François
(adaptation des cours de Jacques Ferber)

Université de Montpellier
Laboratoire d'informatique, de robotique
et de microélectronique de Montpellier

Janvier 2021

Agents réactifs

Agent

- Situé: point de vue et porté locale
- Incarné: contraintes de l'environnement
- Autonome: poursuit ses buts propres

Réactif

- Pas de représentation symbolique
- Pas de raisonnement abstrait

Pas de planification, pas de carte du monde, règle ses problèmes localement, comme ils se présentent.

Mouvements de foule

Types de mouvements de foule

Coordination de mouvements

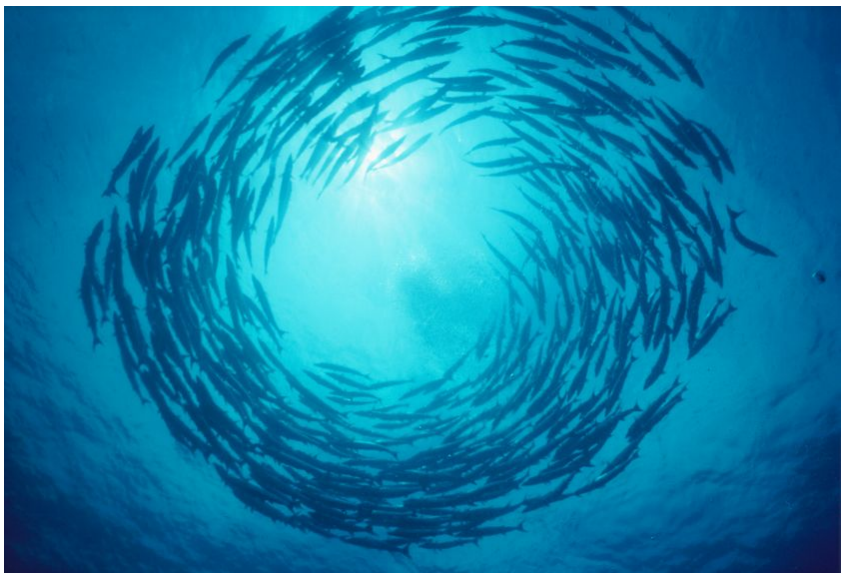
- Suivre
- Entourer
- Flocking (déplacement en formation)
- Évitement
- Rejoindre

Formation coopérative dépendant de la nature des agents

- Physique (Formations en V, en file ...)
- Défense (front, carré, tortue ...)
- Exploration (sondage avalanches ...)

Flocking: mouvement de troupeau

Définition: (flock) un groupe d'oiseaux, de poissons de mammifères, d'humains (d'agents) qui avancent ensemble en formation.



Boids [Reynolds, 1986]

- “boids” vient de “bird-oids” - Article fondateur de Craig Reynolds en 1986.
- Mécanisme semblable à des systèmes de particules, mais avec une orientation
- Mouvement dirigé par le comportement (Behavior-based motion), Algorithme distribué, pas de calcul centralisé.
- Vitesse constante
- Contrainte uniquement en terme de rotation

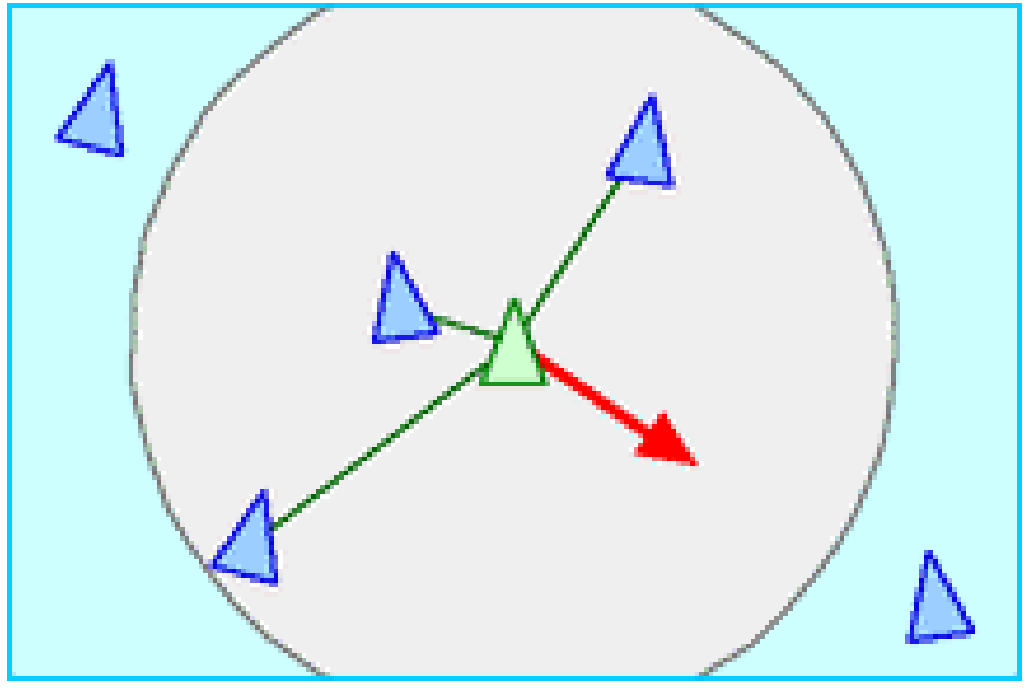
`https://www.red3d.com/cwr/boids/`

Flocking: mouvement de troupeau

- Les Boids doivent se coordonner avec leur voisins.
- Deux tendances principales:
  - Rester près des autres
  - Éviter les collisions avec les autres
- D'autres objectifs peuvent être ajoutés:
  - Prédation, combat, jeux ...
  - Trouver de la nourriture, ressource, énergie ...
  - Reproduction, recherche, regroupement ...

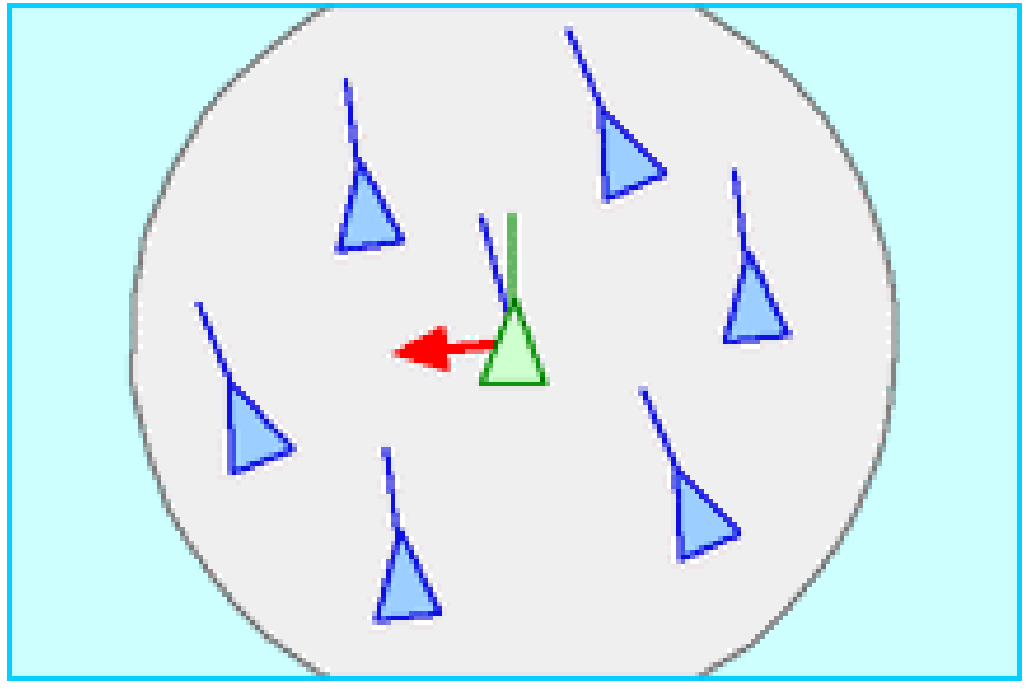
Flocking: 1 - éviter les collisions

Éviter les collisions avec les voisins



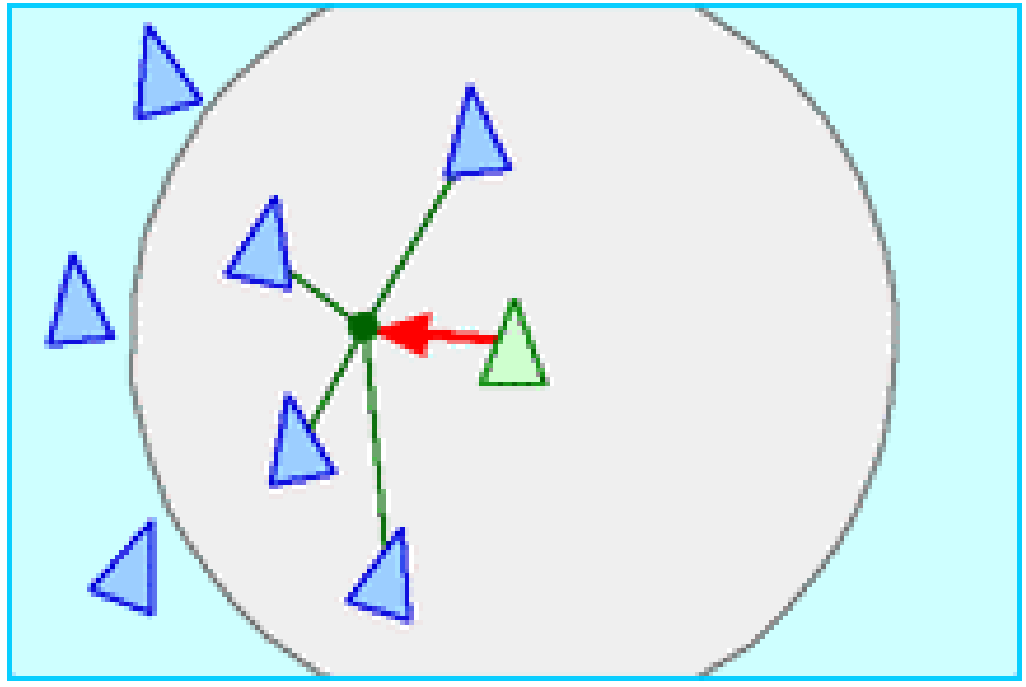
Flocking: 2 - s'aligner avec les autres

Essayer de s'adapter à la direction des voisins



Flocking: 3 - cohésion du groupe

Essayer d'être le plus près des voisins, d'être plus au centre du troupeau..



Flocking: Technique naïve

```
1  A <- agents a portee
2  a <- agent a portee le plus proche
3
4  Si Distance(a) < distance evitement
5     eviter()
6  Sinon Si DistanceCentre(A) > distance cohesion
7     cohesion()
8  Sinon
9     aligner()
```

Alternatives

On peut ajouter d'autres règles pour représenter des comportements basés sur d'autres contraintes

Formation en V des oiseaux migrateurs: utilisation du déplacement d'air de l'oiseau de tête pour économiser son énergie.

Formation en V: le vol des oies sauvages

1. Si agent est trop loin des autres agents, il accélère pour se rapprocher du plus proche.
2. Si un agent est suffisamment près d'un autre, il va venir sur l'un de ses côtés, pour que sa vue ne soit pas obstruée.
3. Si un agent est trop proche d'un autre, il ralentit.
4. Quand les trois autres conditions sont remplies, l'agent adapte sa vitesse et direction à ses voisins visibles

Nathan, A. & Barbosa, V. C (2008)
NetLogo models library: Flocking Vee Formations

Outre les contraites physiques, on voit que ce comportement minimise la consomation d'énergie.

Flocking en V

```
1  to adjust ;; ajuster la direction et position par rapport aux autres.
2     set closest-neighbor min-one-of visible-neighbors [distance myself]
3     let closest-distance distance closest-neighbor
4     ;; if I am too far away from the nearest bird I can see, then try to
           get near them.
5     if closest-distance > updraft-distance [
6        turn-towards (towards closest-neighbor)
7        set speed base-speed * (1 + speed-change-factor)
8        set happy? false
9        stop]
10    ;; if my view is obstructed, move sideways randomly.
11    if any? visible-neighbors in-cone vision-distance obstruction-cone [
12       turn-at-most (random-float (max-turn * 2) - max-turn)
13       set speed base-speed * (1 + speed-change-factor)
14       set happy? false
15       stop]
16    ;; if i am too close to the nearest bird slow down.
17    if closest-distance < too-close [
18       set happy? false
19       set speed base-speed * (1 - speed-change-factor)
20       stop]
21    ;; if all three conditions are filled, adjust.
22    ;; to the speed and heading of my neighbor and take it easy.
23    set speed [speed] of closest-neighbor
24    turn-towards [heading] of closest-neighbor
25    set happy? true
26 end
```

Approche vectorielle

Idée:

- Composer l'ensemble des motivations et décisions d'actions sous la forme de vecteurs.
- Utiliser de simples compositions vectorielles linéaires (la plupart du temps) ou non-linéaires pour déterminer le mouvement.

Le Comportement B est déterminé par la somme des comportements élémentaires β_i.

$$\vec{B} = \omega_1\vec{\beta_1} + \omega_2\vec{\beta_2} + ... + \omega_n\vec{\beta_n}$$

$$\vec{B} = \sum_{i=1}^{n} \omega_i\vec{\beta_i}$$

Application au flocking

Flocking: Behaviour arbitration [Reynolds, 1986]

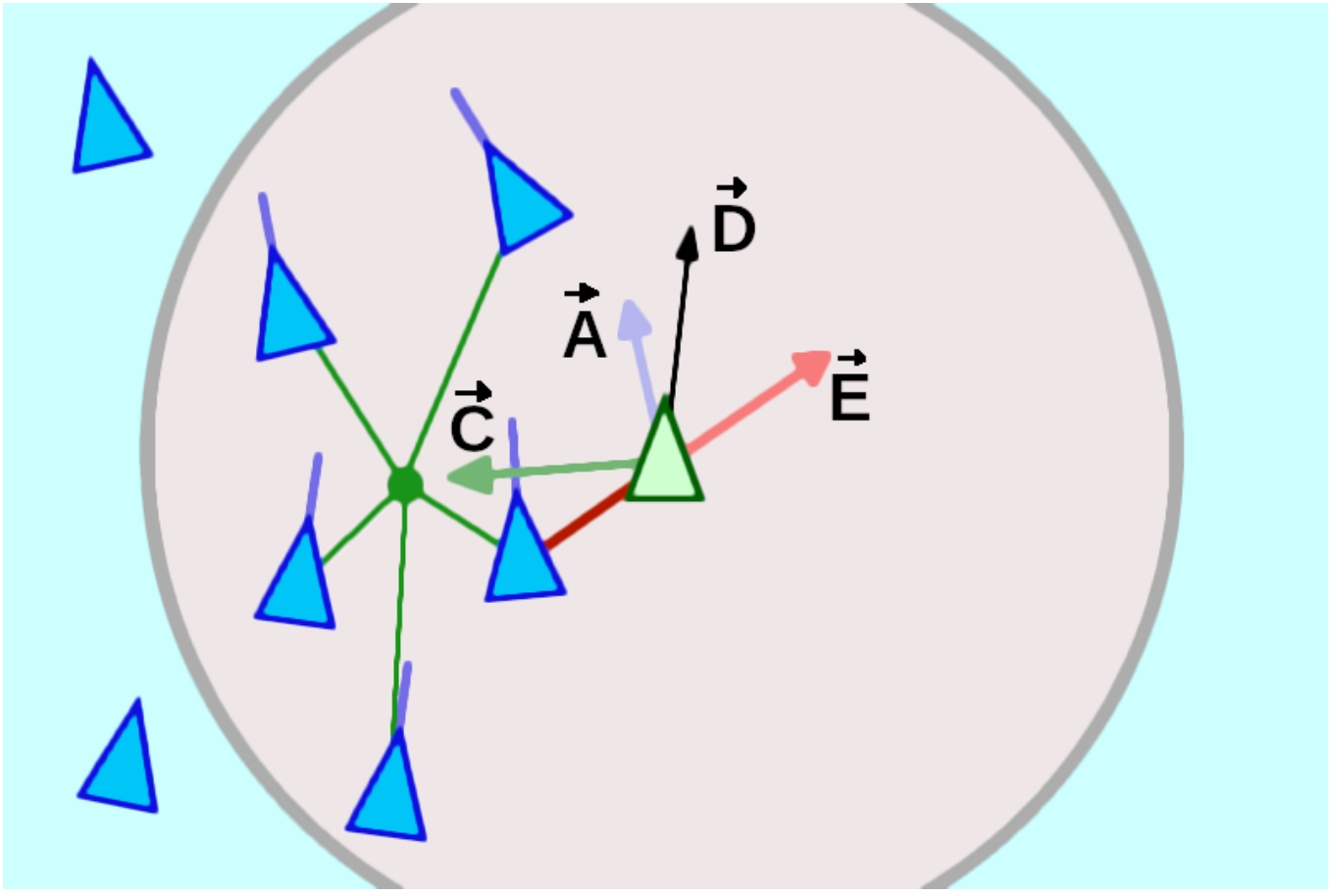
Chaque comportement (évitement, alignement,centrage) définit un vecteur

On somme ces vecteurs en fonction de leur importance:

$$\vec{D}_{irection} = \omega_{align} \cdot \vec{\beta}_{align} + \omega_{evite} \cdot \vec{\beta}_{evite} + \omega_{cohes} \cdot \vec{\beta}_{cohes}$$

Application au flocking

```
Direction = importanceEviter*Eviter()
          + importanceAligner*Aligner()
          + importanceCohesion*Cohesion()
```



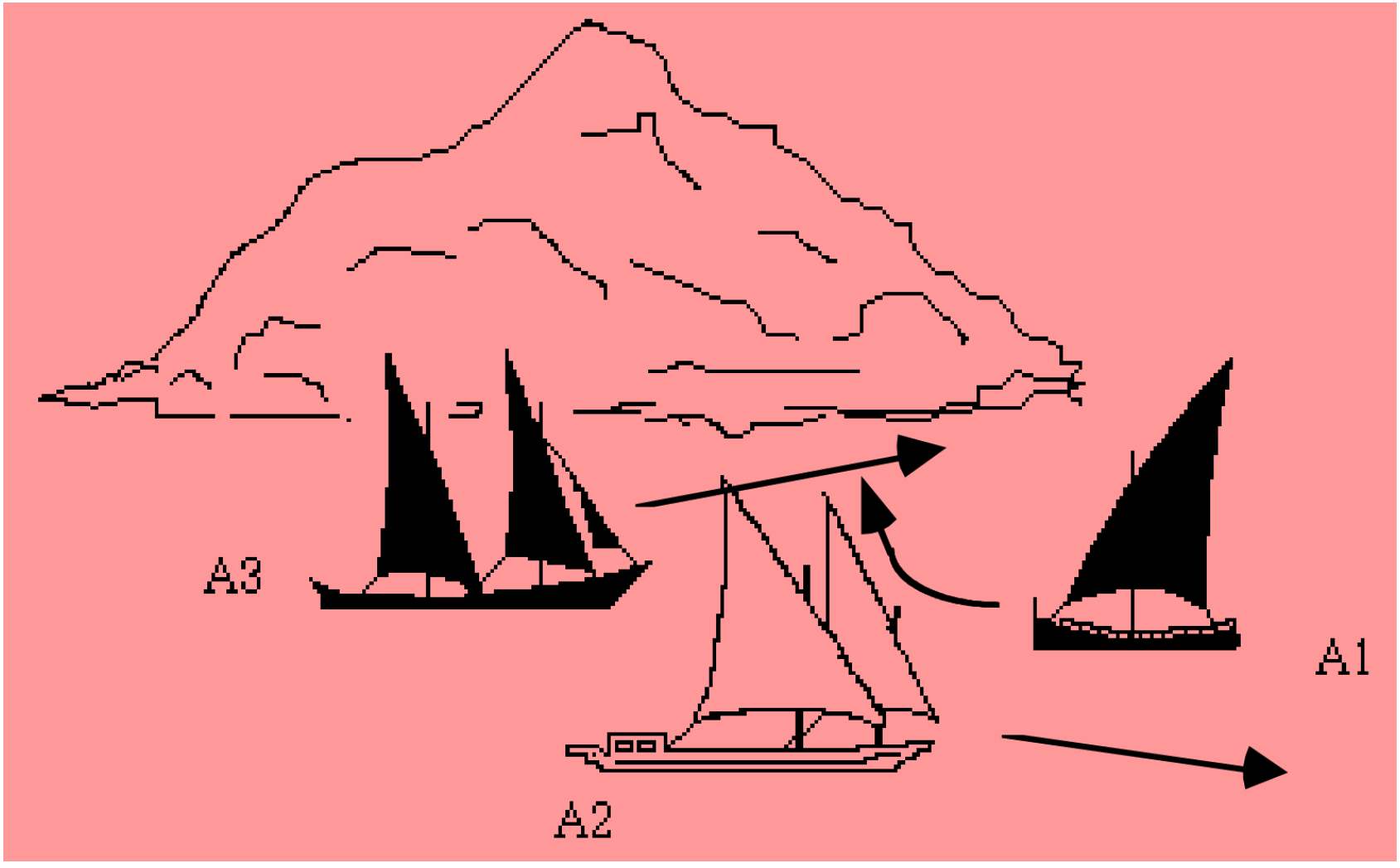
Fonctions vectorielles en NetLogo

```
1  to-report angleFromVect [vect]
2     let a atan item 0 vect item 1 vect
3     report a
4  end
5
6  to-report vectFromAngle [angle len]
7     let l (list (len * cos angle) (len * sin angle))
8     report l
9  end
10
11 to-report multiplyScalarvect [factor vect]
12    report (list (item 0 vect * factor) (item 1 vect * factor))
13 end
14
15 to-report additionvect [v1 v2]
16    report (list (item 0 v1 + item 0 v2) (item 1 v1 + item 1 v2) )
17 end
```

Exemple de fonction en version vectorielle

```
1  to-report vectCohere
2     let x-component mean [sin (towards myself + 180)] of flockmates
3     let y-component mean [cos (towards myself + 180)] of flockmates
4     report (list x-component y-component)
5  end
6
7  to-report vectAlign
8     let x-component sum [dx] of flockmates
9     let y-component sum [dy] of flockmates
10    report (list x-component y-component)
11 end
```

Évitement de collision



Évitement d'obstacles statiques

1 ère solution: répulsion/fuite

Deux approches

- Priorité: d'abord flocking, ensuite répulsion, ou l'inverse.
- Combinaison de vecteurs

Par priorité

Algorithme général

```
1  to flock
2     find-flockmates
3     if any? flockmates
4     [ find-nearest-neighbor
5       ifelse distance nearest-neighbor < minimum-separation
6        [ separate ]
7        [ align
8          cohere ] ]
9     avoid-obstacles ;; ce qui change.
10 end
```

Solution d'évitement simple: la fuite

Fuir: partir dans le sens opposé

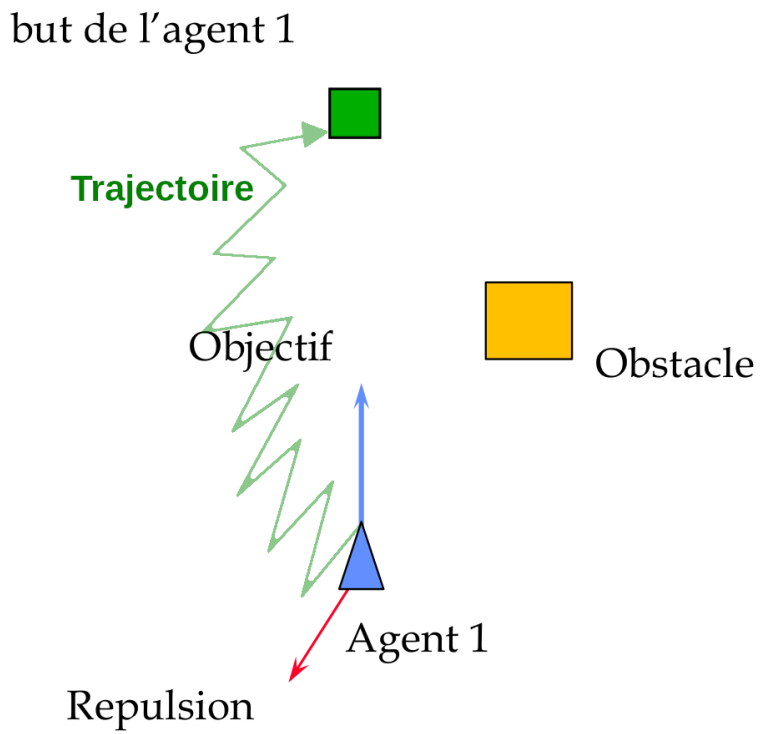
```
1  to avoid-obstacles
2     ; avoid anything nearby that is not black
3     set obstacles patches in-cone vision angle-avoidance with [pcolor !=
          black]
4     if (any? obstacles)
5        [flee]
6  end
7
8  to flee
9     let obstacles-in-front obstacles in-cone 3 angle-flee
10    if (any? obstacles-in-front)
11    [
12       rt 180
13       rt random 10
14       lt random 10
15    ]
16 end
```

Smart avoidance

Mais si pas possible, on fait au mieux (et il y a des crashes).

```
1  to smart-avoidance
2     turn-at-most 180 max-avoidance-turn
3     flee
4  end
```

Par priorité



L'agent hésite entre répulsion et aller vers l'objectif.

Évitement d'obstacles statiques

1 ère solution: répulsion/fuite

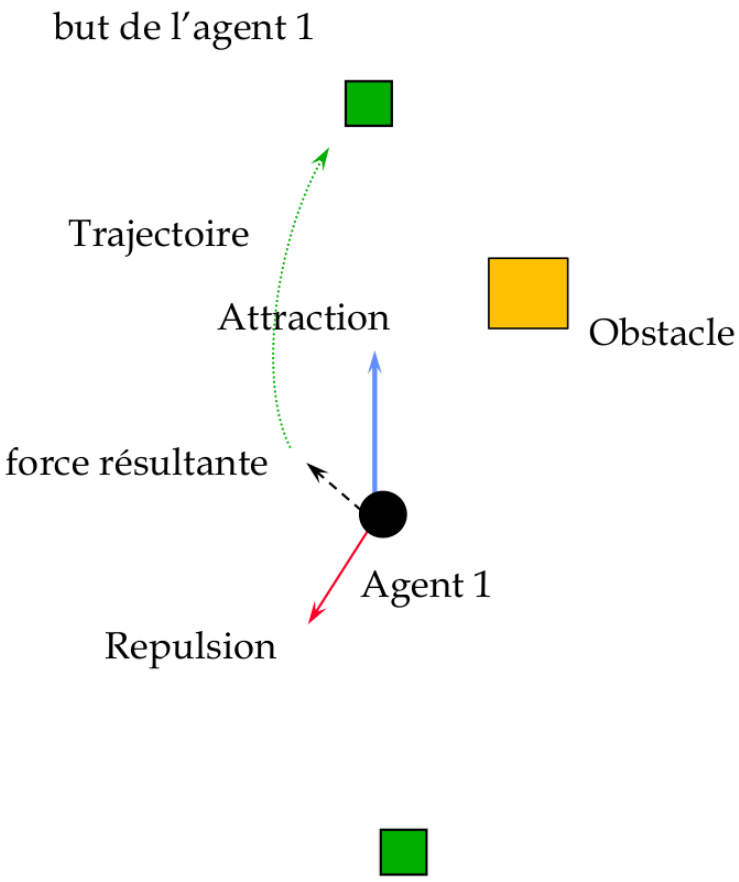
Deux approches

- Priorité: d'abord flocking, ensuite répulsion, ou l'inverse.
- Combinaison de vecteurs

$$\vec{D} = a\vec{R} + (1–a)\vec{F}$$

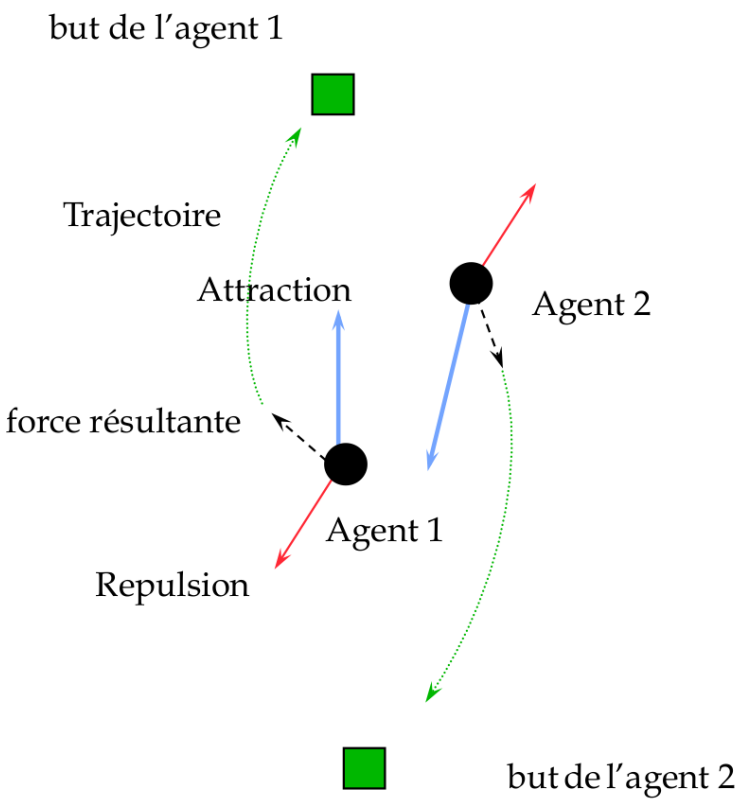
Où $\vec{R}$ est le vecteur de répulsion et $\vec{F}$ celui du flocking. **a** est le coefficient de contrôle. Peut varier en fonction inverse de la distance à l'obstacle (quand l'obstacle est droit devant)
$a = k/dist(self, obstacle)$.

Combinaison vectorielle



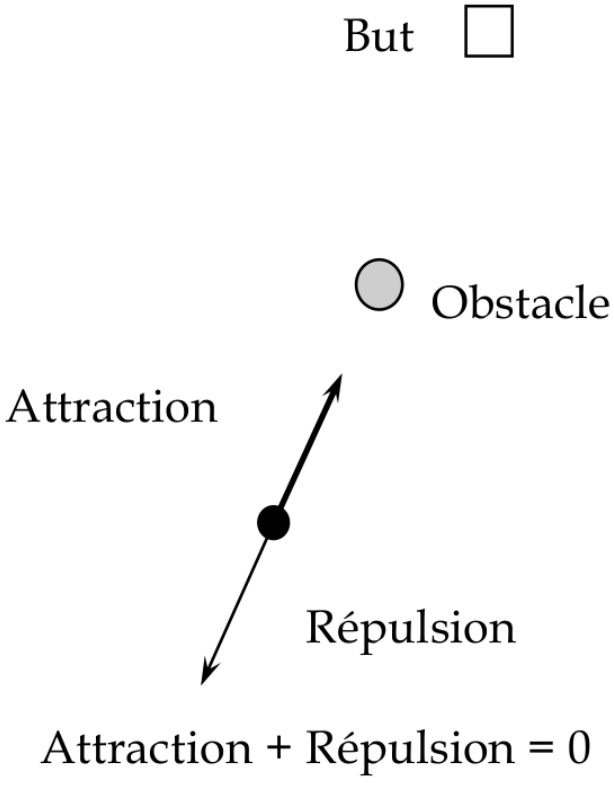
L'agent combine deux vecteurs, chacun correspondant à un comportement.

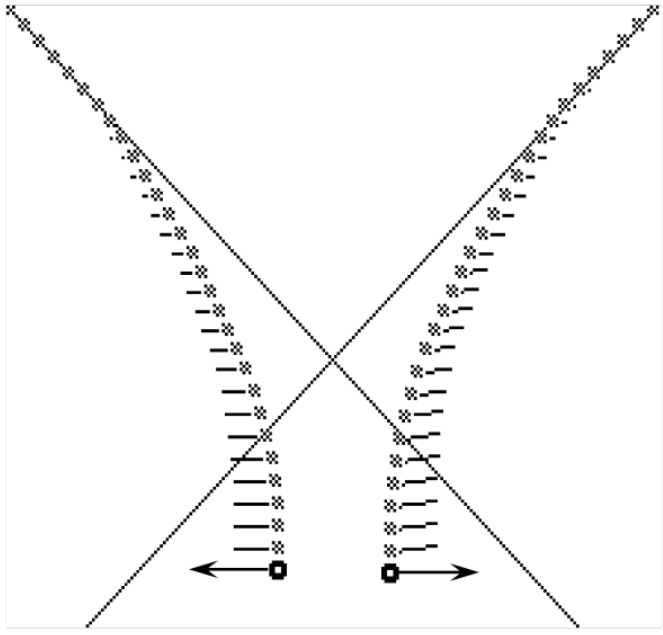
Combinaison vectorielle



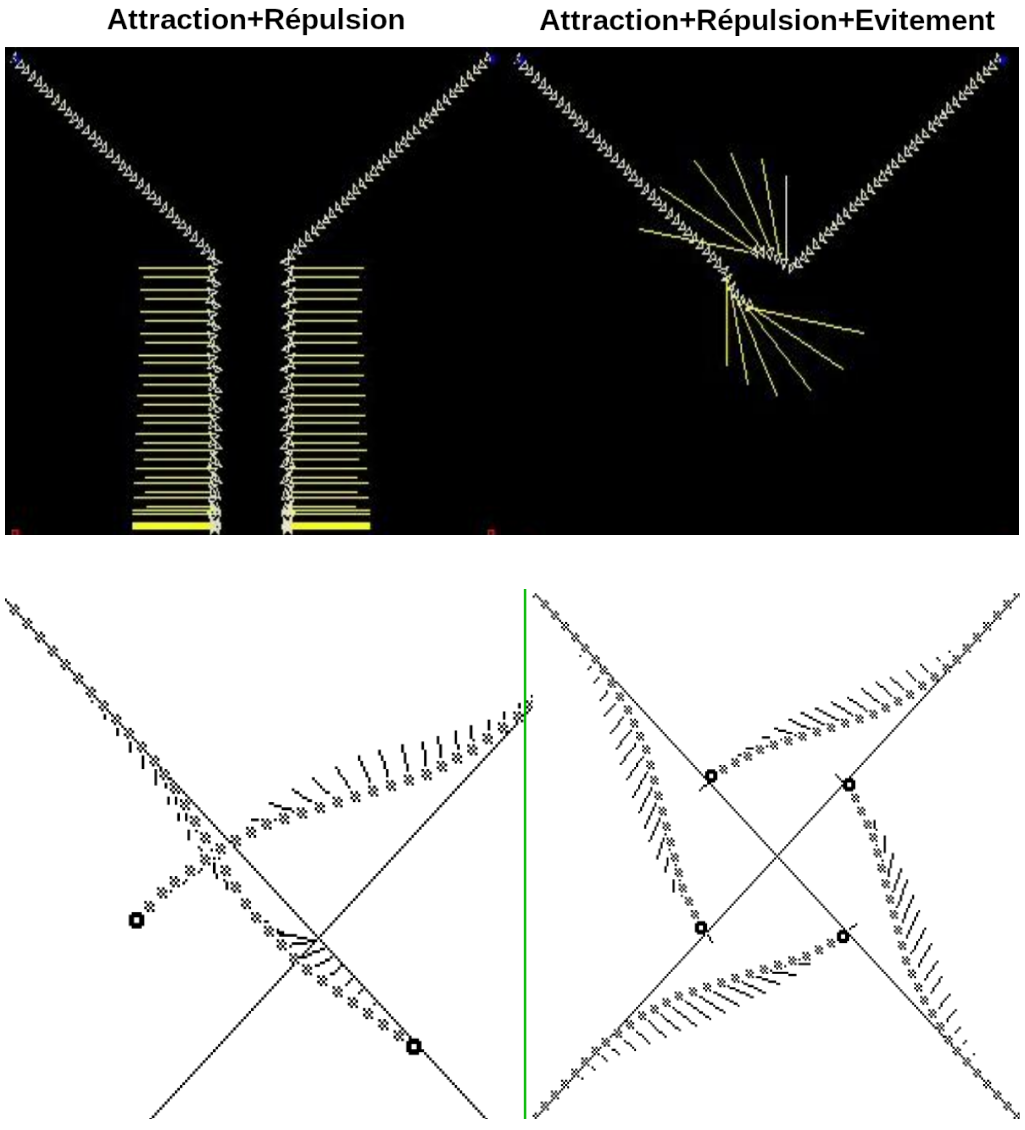
Chaque agent est considéré comme étant un obstacle pour l'autre.

Mais la répulsion n'est pas l'évitement





Ce que l'on souhaiterai ...



TP: le flocking

- Flocking simple
- Technique vectorielle
- Flocking en V