

HMIN108 - Programmation orientée agents

Partie 1: Agents réactifs

2 - Situation, environnement, communication

Suro François

(adaptation des cours de Jacques Ferber)

Université de Montpellier

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

Septembre 2020

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.

Comportements collectifs

Types de comportements collectifs

- ▶ Comportements coopératifs
- ▶ Comportements collectifs de défense et d'attaque



Formations coopératives

Coordination de mouvements

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

Formation dépendant de la nature des agents

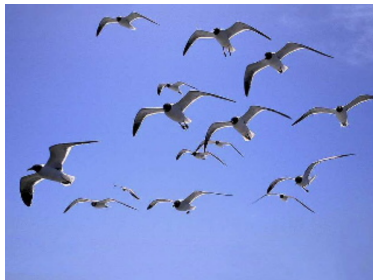
- ▶ Physique (Formations en V, en file ...)
- ▶ Défense (front, carré, tortue ...)
- ▶ Exploration (sondage avalanches ...)

Formations dynamiques

- ▶ Recrutement d'agents
- ▶ Subdivision

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

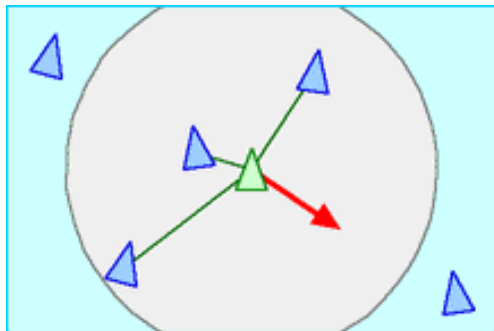
<https://www.red3d.com/cwr/boids/>

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
- ▶ Dans la nature, les mouvements ont évolués:
 - ▶ Prédation
 - ▶ Trouver de la nourriture
 - ▶ Rencontre amoureuse/sexuelle (mating)

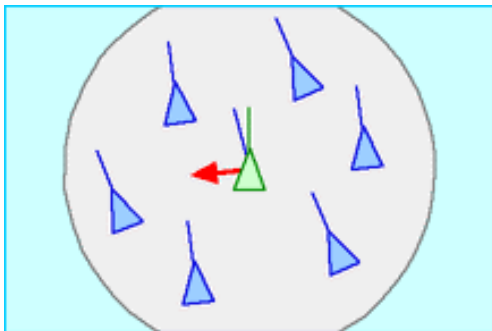
Flocking: 1 - éviter les collisions

Éviter les collisions avec les voisins



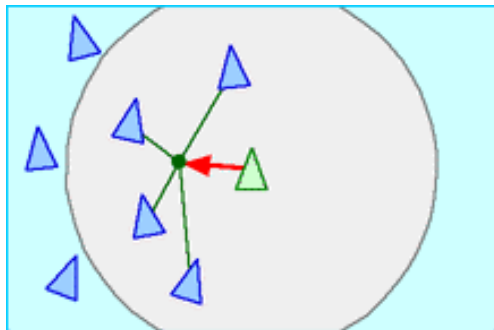
Flocking: 2 - s'aligner avec les autres

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..



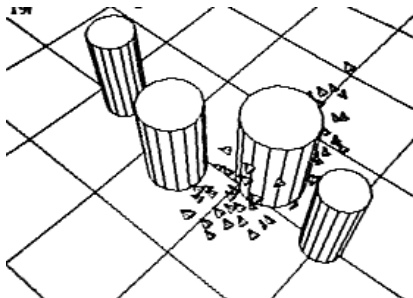
Flocking: Technique naïve

```
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()
```

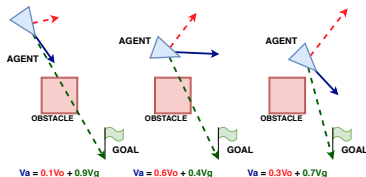
Flocking: Behaviour arbitration [Reynolds, 1986]

- ▶ Chaque comportement (évitement, alignement, centrage) définit un vecteur.
- ▶ On somme ces vecteurs en fonction de leur importance.

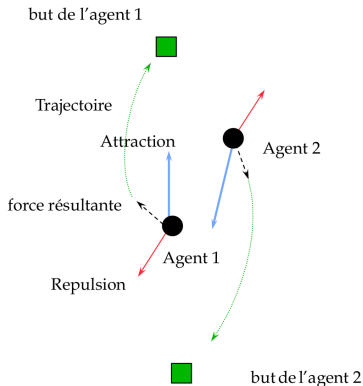
```
direction = importanceEviter*eviter()  
           + importanceAligner*aligner()  
           + importanceCohesion*cohesion()
```



Technique par composition de vecteurs



Approche des Boids:
l'importance de chaque vecteur
est pondéré.



Approche par champs de force:
une simple somme, la valeur de
chaque force change en fonction
de la position.

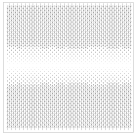
Approches par champs de force



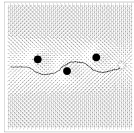
(A) Move-to-goal



(B) Avoid-static-obstacle



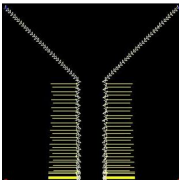
(C) Stay-on-path



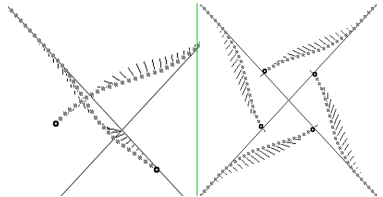
(A)+(B)+(C)

A gauche: l'architecture AuRA
En bas: cas particulier du modèle
attraction+répulsion qui ne fonctionne
pas. Solution avec
attraction+répulsion+évitement.

Attraction+Répulsion

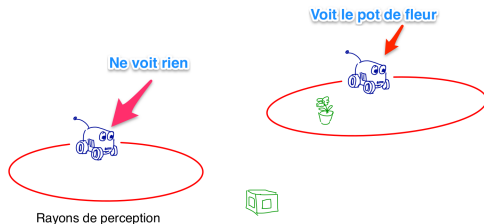


Attraction+Répulsion+Evitement



Les limites de la perception limitée ?

Jusqu'ici nos agents se coordonnent localement avec des agents dans leur champ de perception.



Comment se coordonner avec des agents hors de mon champ de perception limité ?

Rejoindre d'autres agents:

- ▶ Pas de perception directe
- ▶ Agents en déplacement: pas de position connue à l'avance
- ▶ Pas de système de communication directe (téléphone, télépathie ...)

L'importance de l'environnement

L'environnement contient plein d'informations

- ▶ Informations naturelles:
Végétation, paysage (types et propriétés)
- ▶ Informations ajoutées:
Volontairement: marques, balises,
phéromones Involontairement: traces de
pas, odeurs
- ▶ Systèmes de communications:
Signaux



Principe général d'un environnement qui contient des indices :

- ▶ On suit les indices en espérant qu'ils nous conduisent au but en nous faisant éviter les obstacles
- ▶ Les indices sont des substituts à ce vers quoi on se dirige
Ex: les traces des animaux pour un prédateur, chemins de phéromones pour les fourmies ...

Se diriger grâce à un signal

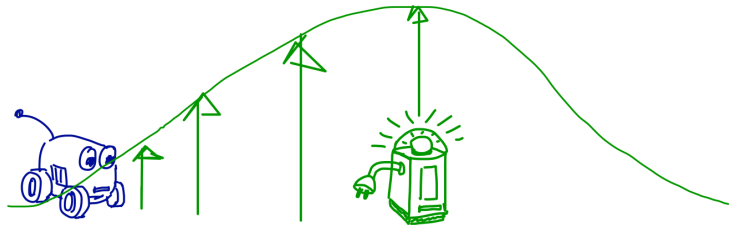
les indices et traces sont interprétés comme des signaux pour aller vers un but

L'agent est attiré par le signal émis par la borne



Aller vers les valeurs les plus grandes d'un champ

Suivre le **gradient** d'un **champ de potentiel**



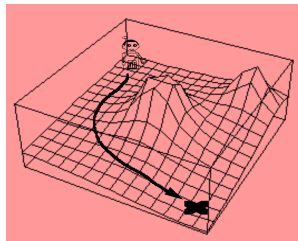
VALEURS DU CHAMP

Suivre un gradient de potentiel

Suivre un gradient de potentiel

Les forces sont définies comme le gradient d'un champ de potentiel

$$F(p) = \text{grad}(U(p))$$



Les buts sont représentés comme des champs attractifs, les obstacles sont représentés comme des champs répulsifs. Le mouvement est obtenu par une combinaison de champs attractifs et répulsifs:

$$U(p) = U_{\text{attr}}(p) + U_{\text{repul}}(p)$$

comparable aux champs de force, mais avec les valeurs du voisinage au lieu de vecteurs

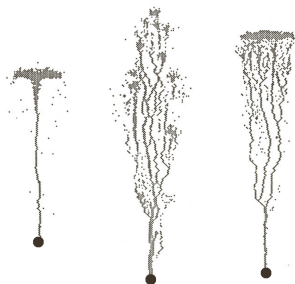
Coordination entre agents en utilisant l'environnement ?

Pistes de phéromones:

Déposées par un autre agent

Diffusion et évaporation des phéromones.

Suivre les phéromones les plus "fraîches".



NetLogo: rappels

Les tortues ont un mouvement local défini à partir d'une "géométrie tortue": tourner à droite, tourner à gauche ...

```
1 to cercle
2   repeat 360 [fd 1 rt 1]
3 end
```

S'orienter : set heading towards *entité*

```
1 set heading towards patch 0 0
```

Faire face : face *entité*

```
1 face one-of reines
```

NetLogo: rappels

Aléatoire: *random valmax*

ex: tourner à gauche aléatoirement entre 0 et 30 degrés

```
1 lt random 30
```

Perception: *agentset in-radius rayonperception*

```
1 Reines in-radius 3
```

Filtrer un agentSet: *one-of*, *min-one-of*, *max-one-of*

ex: se diriger vers la reine la plus proche dans un rayon de 5:

```
1 Let r min-one-of reines in-radius 5 [distance myself]
2 if r != nobody [
3   face r
4   fd 1
5 ]
```

NetLogo: rappels

Déclarer des attributs pour un patch :

```
1 patches-own [hauteur-herbe humidite]
```

Pour aller vers le patch avec la plus grande valeur d'un attribut:

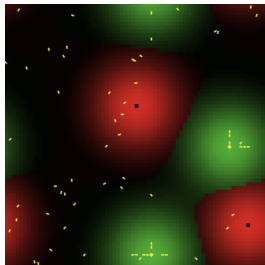
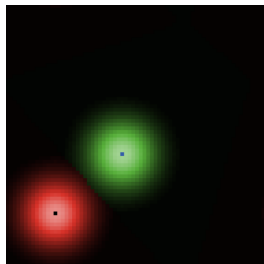
```
1 face max-one-of patches in radius 8 [hauteur-herbe]
2 fd 1
```

```
1 set heading towards max-one-of neighbors [humidite]
2 fd 1
```

NetLogo: les champs de potentiel

Gestion de l'environnement:

- ▶ Diffusion: partage sa valeur avec ses voisins
- ▶ Évaporation: dissipation, dégradation des anciennes traces.



Suivi de gradient:

- ▶ Primitive uphill: "monter une colline", suivre le gradient ascendant.
- ▶ Fonction de suivi de gradient sur mesure.

NetLogo: diffusion

Primitive de diffusion: *diffuse variable coeff*

Attention: primitive *observer*, appelée un fois par tick, pas par chaque patch.

```
1 diffuse chemical 0.40
```

chaque patch diffuse 40% de sa variable *chemical* à ses 8 patches voisins. Donc, chaque patch voisin reçoit $1/8$ de 40% de la variable *chemical* (chaque patch voisin reçoit 5% de la valeur du patch diffusant)

NetLogo: évaporation

La vitesse de disparition des odeurs, phéromones ...

A chaque tour, le patch perd `taux` de sa valeur d'odeur.

Exemple avec un pourcentage:

```
1 set odeur odeur * (100 - taux) / 100
```

NetLogo: suivi du gradient

uphill attribut - downhill attribut

Avance la tortue dans le patch dont la valeur de l'attribut est la plus élevée/faible.

Attention: le comportement des ces primitives est équivalent à (pour uphill):

```
1 let p max-one-of neighbors [ <variable> ]
2 if [<variable>] of p > <variable>
3 [
4   face p
5   move-to p
6 ]
```

Ce qui posera problème

NetLogo: suivi du gradient

Nous utiliserons notre propre procédure à la place de uphill

```
1 let p max-one-of neighbors [ <variable> ]  
2 if [<variable>] of p > <variable>  
3 [  
4   face p  
5   fd 1  
6 ]
```

Une belle simulation

Comportements naturels

vivants, organiques ... pas
nécessairement optimal.

