Steering

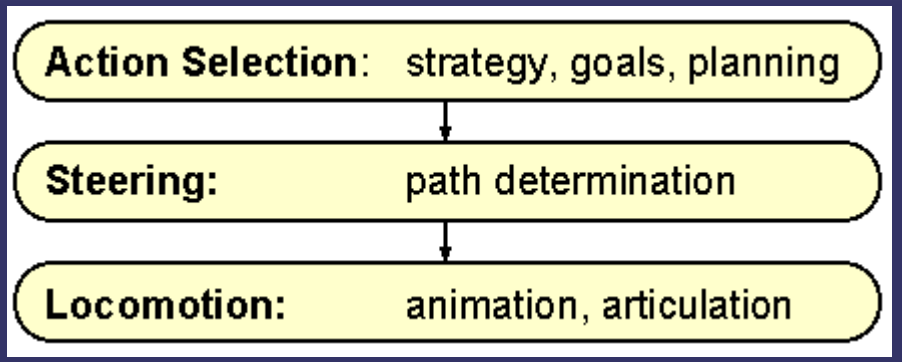
Colin Bruneau
CREAJEUX

D'après Craig W. Reynolds http://www.red3d.com/cwr/papers/1999/gdc99steer.html

Présentation

- Le Steering apporte des solutions réalistes à plusieurs problèmes de mouvements de personnages autonomes:
  - mouvement improvisé en temps réel
  - mouvement indépendant du moyen de locomotion
  - atteinte d'objectifs de haut niveau:
    - éviter des obstacles
    - suivre un chemin
    - joindre un groupe
    - etc.

Hiérarchie de comportements



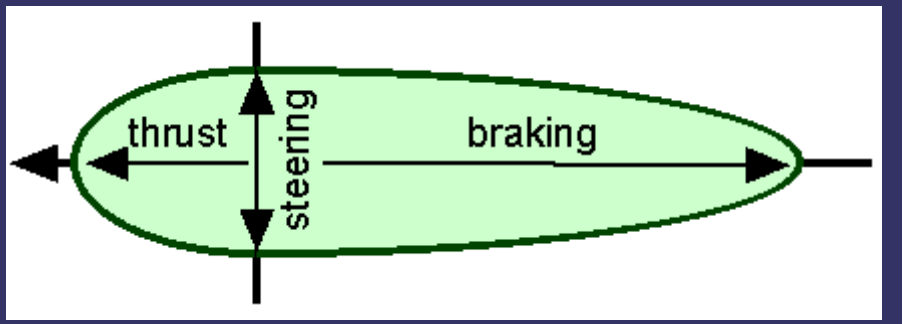
- Hiérarchie pour un comportement de mouvement
- Exemple du troupeau, une vache s'éloigne...
  - Action: le boss dit à son cowboy d'aller chercher la vache
  - Steering: le cowboy détermine son chemin en décomposant en sous-objectifs:
    - se rapprocher de la vache
    - éviter les obstacles
    - attraper la vache
  - Locomotion: le cheval mène le cowboy à la vache

Locomotion

- La couche Locomotion convertit des signaux des décisions du Steering en mouvement
- Le mouvement est soumis aux contraintes de physiques du véhicule:
  - momentum
  - forces
  - limites
- Indépendant du steering pour permettre d'interchanger le véhicule au besoin

Modèle simple de véhicule

- Approximation par Point de masse
  - mass scalar
  - position vector
  - velocity vector
  - max_force scalar
  - max_speed scalar
  - orientation N basis vectors



- Le Steering donne une steering_direction, d'où:

steering_force = truncate (steering_direction, max_force)

acceleration = steering_force / mass

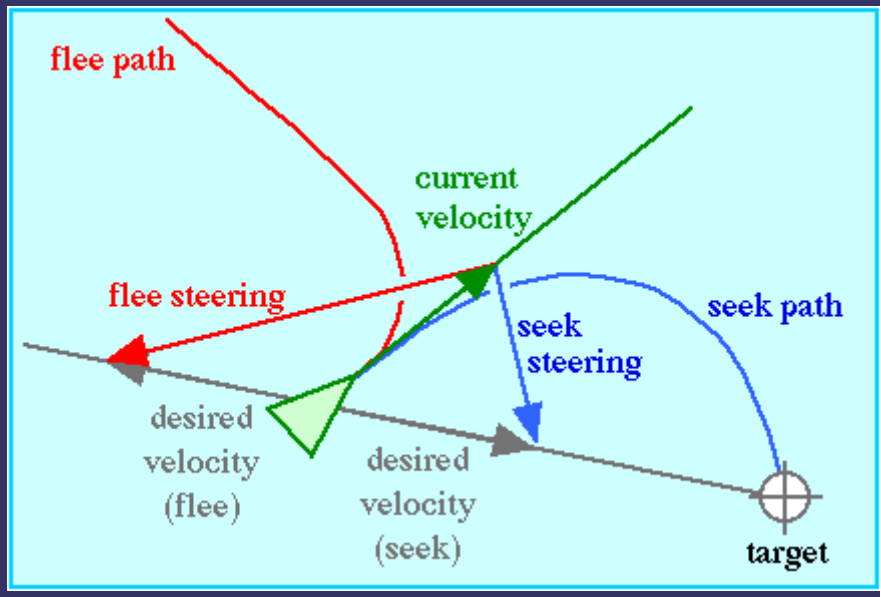
velocity = truncate (velocity + acceleration * dt, max_speed)

position = position + velocity * dt

Steering behaviors

- Seek / Flee
- Pursuit / Evasion
- Arrival
- Obstacle avoidance
- Wander
- Containment / Path following
- Flow field following
- Unaligned collision avoidance
- Separation, Cohesion, Alignment
- Flocking
- Leader following

Seek/Flee



- Seek
  - Aller vers une cible fixe
  - Ajuste la direction de la vélocité

desired_velocity = normalize (target - position) * max_speed

steering = desired_velocity – velocity

- Flee
  - Fuir une cible fixe

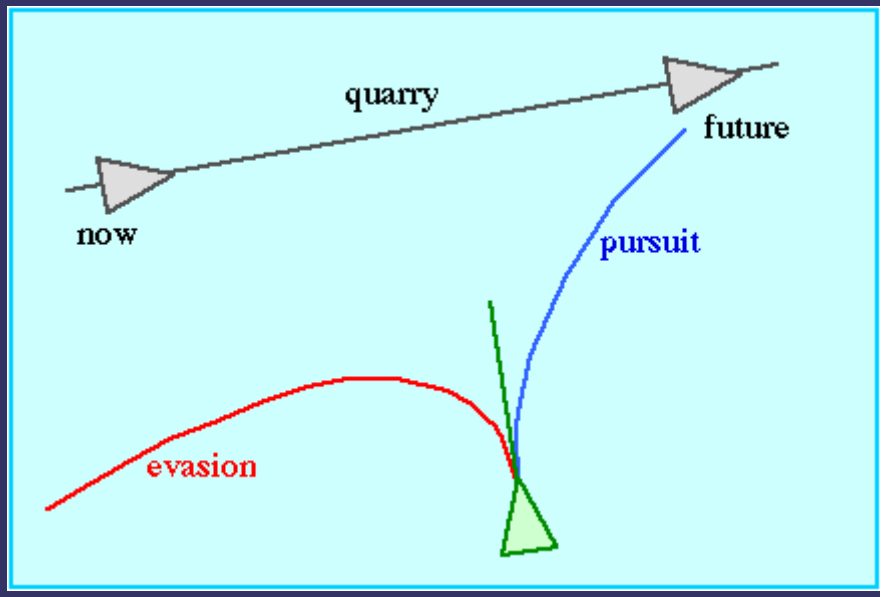
desired_velocity = normalize (position - target) * max_speed

steering = desired_velocity - velocity

Pursuit / Evasion

- Pursuit
  - Aller vers une cible mobile
  - Utilise Seek
  - Prédiction linéaire de la position

de la cible P' = P + Vtarget*T

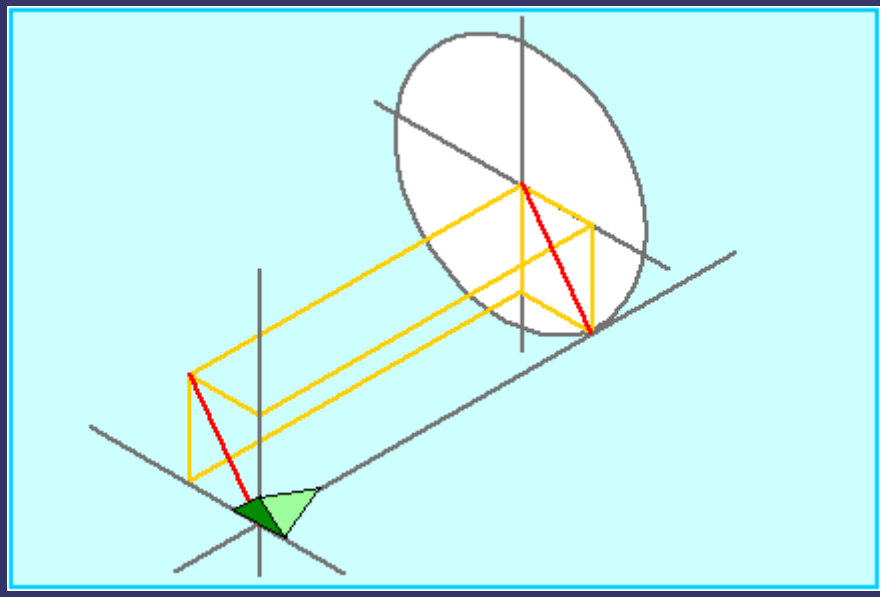


  - T est le temps prédit du rattrapage (avec un maximum)

    T = min(Tmax, distance(entity-target) / Ventity)

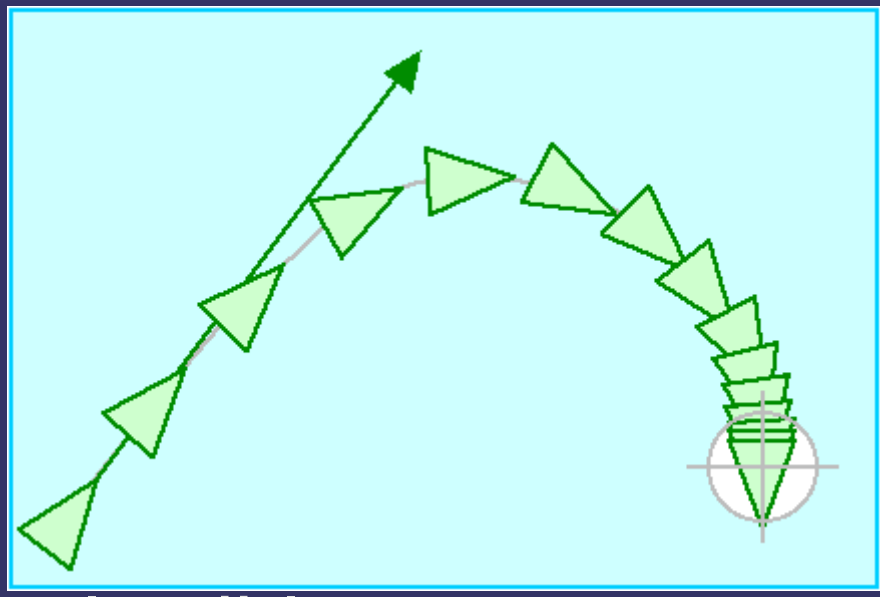
- Evasion
  - Fuir une cible mobile
  - Utilise Flee

Offset pursuit



- Offset sur la position future de la cible
- Utilise Seek
- Calculer dynamiquement une position qui est à une distance R de la cible
  - Projetée de la position cible sur plan Side-Up
  - Normalisé puis multiplié par R
  - Ajouté à la position de la cible

Arrival



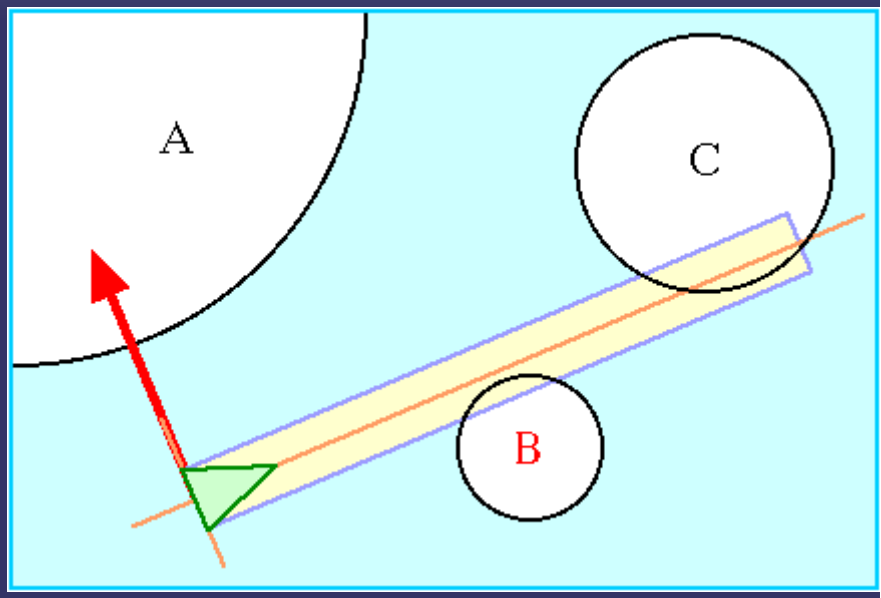
- Ralentissement sur la cible
- Similaire au Seek
- Mais vitesse réduite

linéairement à l'intérieur

d'un cercle de rayon R centré sur la cible

```
target_offset = target - position
distance = length (target_offset)
ramped_speed = max_speed * (distance / slowing_distance)
clipped_speed = minimum (ramped_speed, max_speed)
desired_velocity = (clipped_speed / distance) * target_offset
steering = desired_velocity - velocity
```

Obstacle Avoidance



- Évite des obstacles
- Comme un Flee
- Mais uniquement quand obstacle juste devant
- Approximation des obstacles avec des sphères

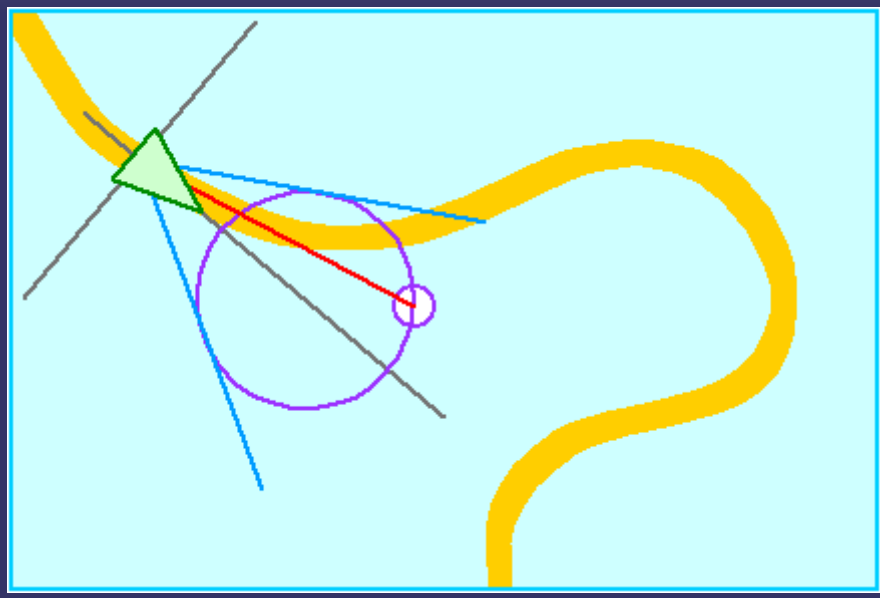
Parcourir chaque obstacle

Si l'obstacle croise le cylindre de mouvement du véhicule

Conserver le plus proche des obstacles

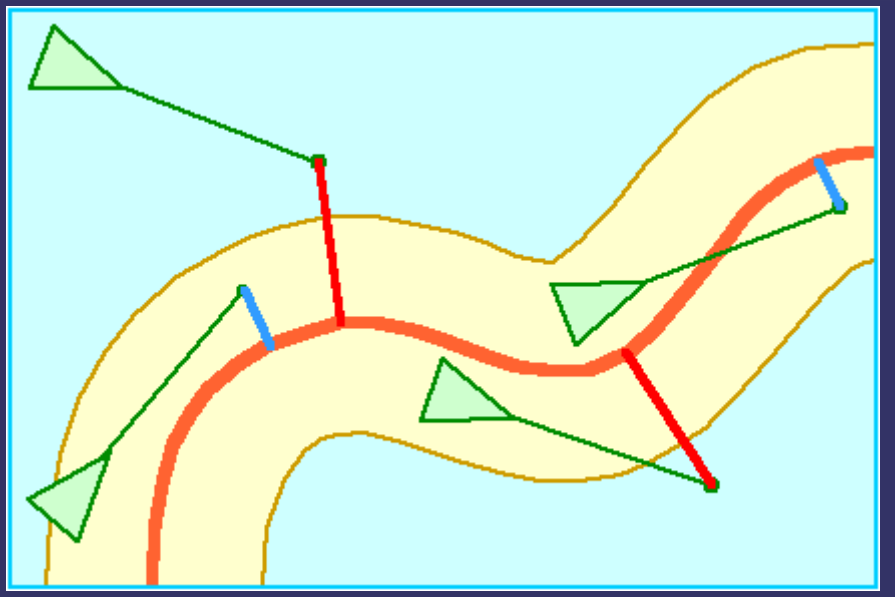
Calculer projection du centre de l'obstacle sur le plan side-up steering_direction inverse de la projection

Wander



- Steering aléatoire
- Déplacement plus réaliste
- Conserve la force de steering
- Ajoute des déplacements aléatoires (petit cercle)
- Contraint la force de steering à un cercle situé devant le perso (grand cercle)
- Permet les comportements Explore et Forage

Path Following



- Suivre un chemin avec une largeur: corridor, tunnel, route
- Contraint la position au rayon du cylindre
- Si le perso est loin, il se rapproche du chemin

Calculer la position future

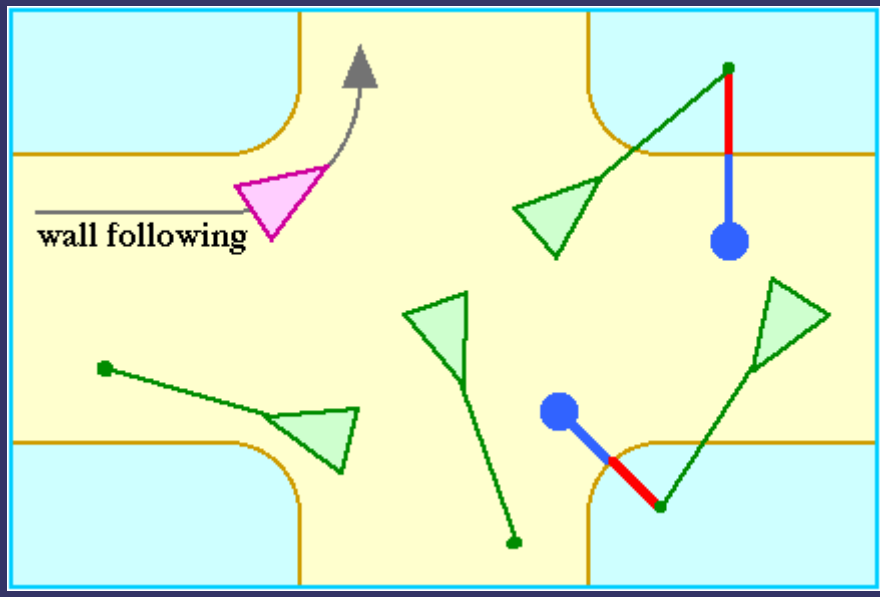
Projeter position sur le point le plus proche du chemin

Si distance > R, on ajoute la force de steering pour rapprocher du chemin

- Adaptable à un chemin orienté

Containment / Wall Following

- Containment est la restriction du mouvement à une zone
- Wall Following est un type de Path Following avec offset
- Path Following est un type de Containment de zone restreinte à la route



Déterminer la position future du perso

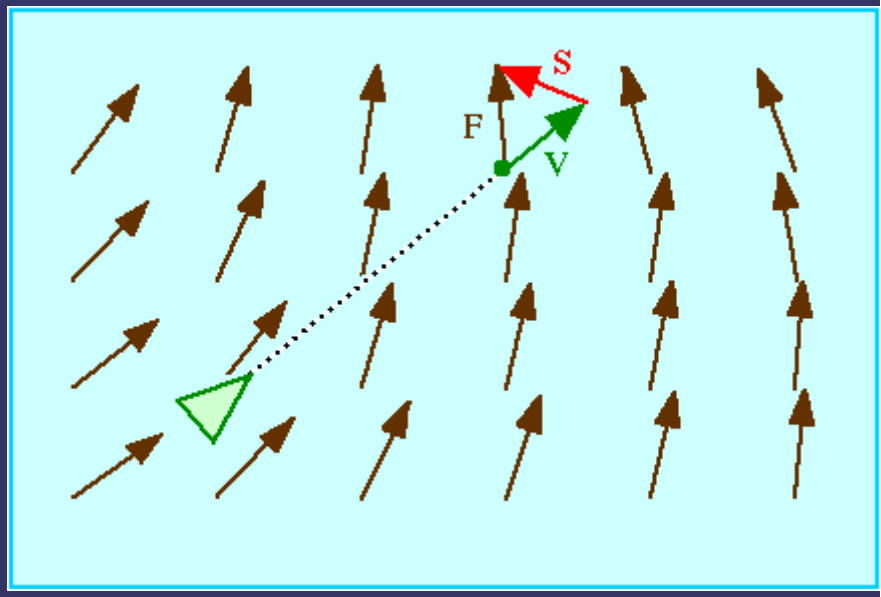
Si position hors de la zone

Steer vers la zone de direction

(projection sur la zone - position future) ou (normale à l'intersection)

Field Following

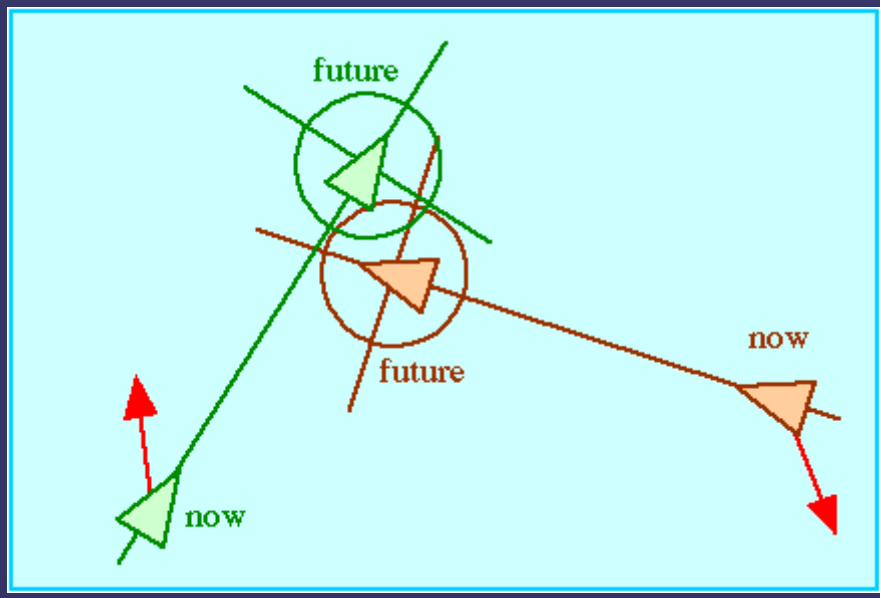
- Mouvement autonome dans un champ de forces
- La vélocité désirée est la force local à la position future du perso



Déterminer la position future du perso

steering_direction = F - V

Unaligned collision avoidance



- Évite les collisions entre entités de directions arbitraires
- Altère la position en fonction de prédiction de positions futures

Déterminer les distances d'approche future pour chaque perso

Conserver le perso de distance plus petite

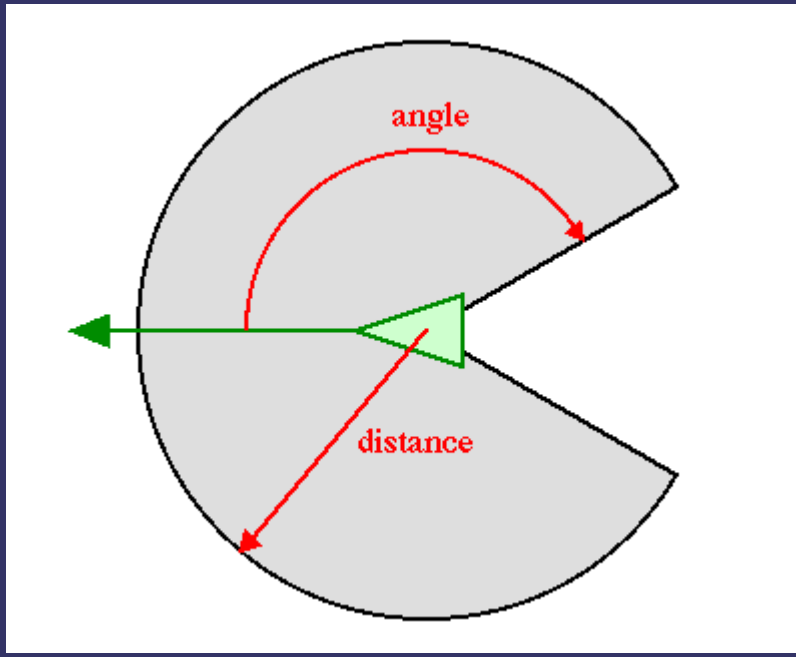
Si distance < R

steering_direction 1 = position future 1 – position future 2

steering_direction 2 = position future 2 – position future 1

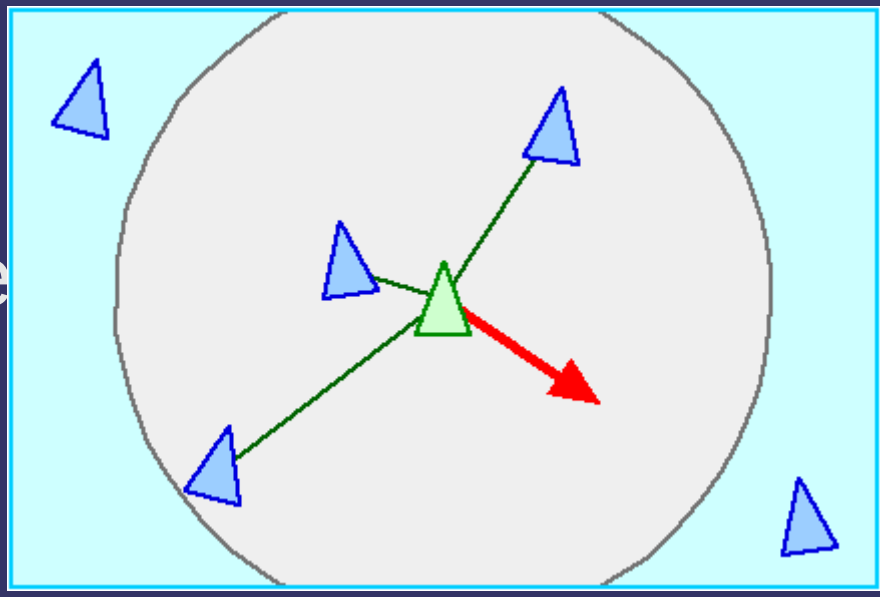
Voisinage

- Le voisinage est défini par:
  - Une distance
  - Un angle de vision



- Les comportements de groupe suivants décrivent le steering lorsque le voisinage est occupé par une autre unité

Separation



- Le comportement Separation permet aux unités de garder une distance par rapport aux autres
- Évite les regroupements

Rechercher les autres unités dans le voisinage

Pour chaque unité dans le voisinage

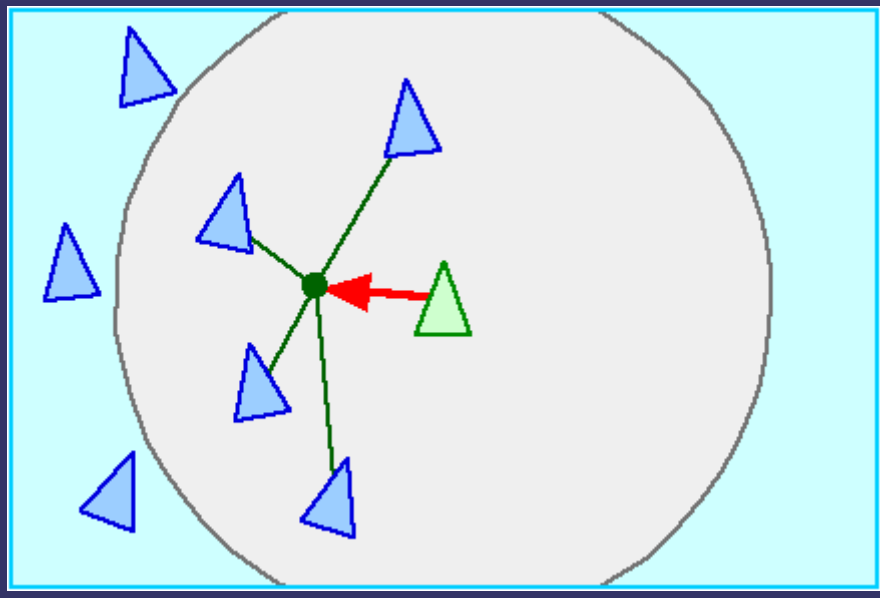
Calculer la force de répulsion de l'unité:

Normalisation(pos. unité – pos. voisin) * 1 / d

Ajouter à la somme des forces de steering

Cohesion

- Le comportement Cohesion permet à l'unité de s'approcher et d'adhérer à un groupe



Rechercher les autres unités dans le voisinage

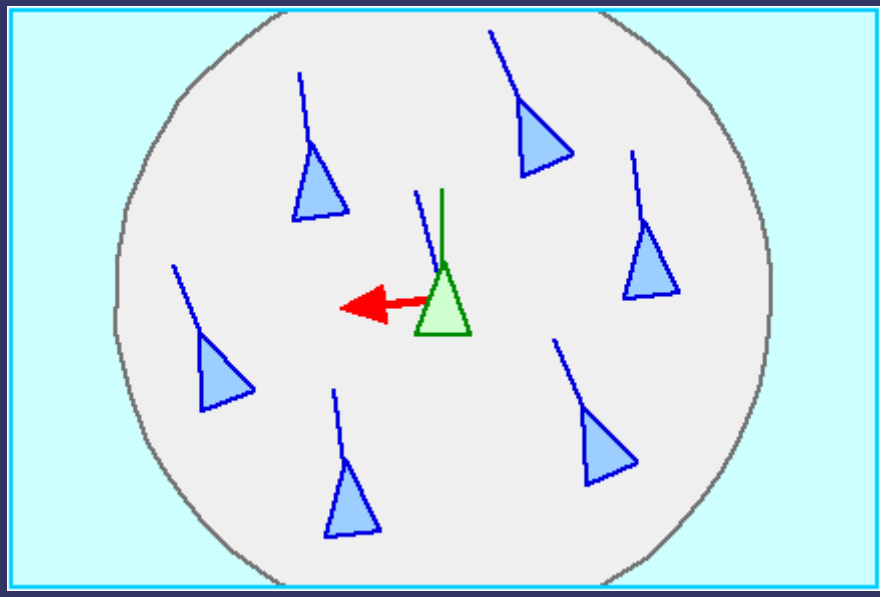
Calculer la position moyenne des unités (centre de gravité)

Calculer la force de répulsion de l'unité:

steering_direction = pos. centre de gravité – pos. unité

Alignment

- Le comportement Alignment permet à l'unité de s'aligner avec les unités d'un groupe



Rechercher les autres unités dans le voisinage

Calculer la vélocité moyenne des unités (vélocité désirée)

Calculer la force de répulsion de l'unité:

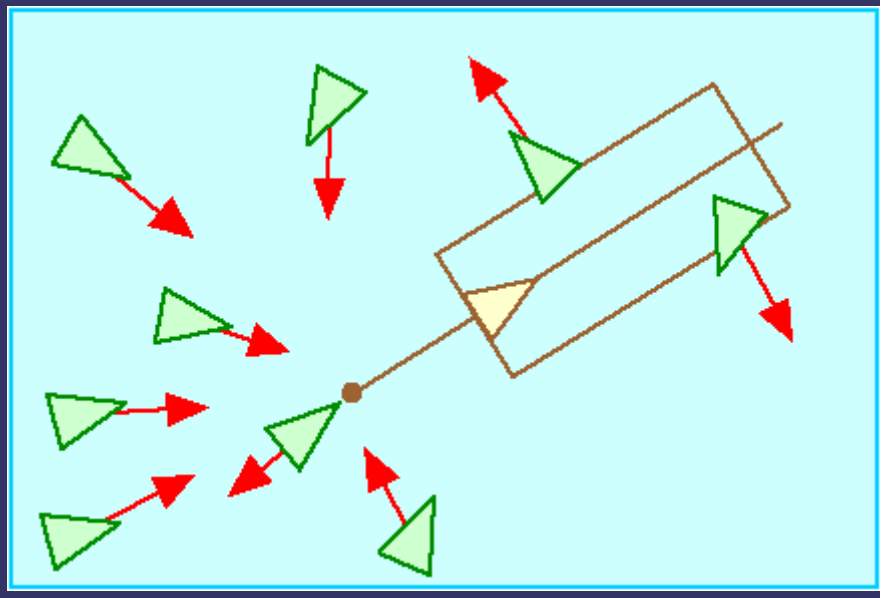
steering_direction = desired_velocity - velocity

Flocking

- Le Flocking simule le mouvement naturel de groupes d'animaux (oiseaux, poissons)
- On peut combiner Separation, Cohesion et Alignment pour trouver une force de stee-ring pour le flocking
- Chaque comportement est déterminé par:
  - poids (steering = normalisation(steering)*poids)
  - Angle et distance (voisinage)

Leader Following

- Les unités suivent un point en arrière du leader et s'écartent sur son passage
- Comportements utilisés:
  - Arrival sur le point en retrait
  - Flee sur la zone en avant
  - Separation pour éviter les collisions



Extensions

- Autres comportements:
  - Interpose (se positionner entre 2 unités mobiles)
  - Docking (comme Arrival + ajuster l'orientation)
  - Hide (vers un point de l'autre coté d'un obstacle)
- Combinaisons
  - Au niveau de la couche Action (séquences)
  - Au niveau de la couche steering (en parallèle)
    - ex: Evasion + Obstacle avoidance
    - Soit par Blending, Priority, Prioritized Dithering