Μοντελοποίηση συμπεριφοράς σμήνους με boids #2

Κ. Φουτζόπουλος[1], Ιανουάριος 2021

για το μάθημα "Υπολογιστική Βιοφυσική"

1. Τμήμα Φυσικής, ΑΠΘ, ΠΜΣ Υπολογιστικής Φυσικής, <kfoutzop@auth.gr>

1 Προσομοίωση

```
mutable struct Boid
  id; pos; vel
  dv; av; ds
  w0; ws; wa; wc
  vmax
end

dis(bi, bj) = norm(bi.pos - bj.pos)
ang(bi, bj) = acos(dot(bi.vel, bj.pos)/(norm(bi.vel)*norm(bj.pos)))
dim(b) = length(b.pos)

B = initialize()

while true
  draw(B)
  flock!(B)
end
```

```
function flock!(set B, time tau)
  B0 = copy(B)
  for bi in B
    deli = neighors(bi, B0)

    si = seperate(bi, deli)
    ai = align(bi, deli)
    ci = cohere(bi, deli)

    bi.vel += bi.ws*si + bi.wa*ai + bi.wc*ci
    bi.pos += tau*bi.vel
  end
end
```

$$\partial_i = \{b_{j\neq i} \in \mathcal{B} : |\mathbf{p}_i - \mathbf{p}_j| < d_\nu \wedge \text{ang}(\mathbf{u}_i, \mathbf{p}_j) < a_\nu\} \quad \text{ang}(\mathbf{u}_i, \mathbf{p}_j) = \cos^{-1}\frac{\mathbf{u}_i \cdot \mathbf{p}_j}{|\mathbf{u}_i||\mathbf{p}_j|}$$

```
function neighbors(boid bi, set B)
  deli = [bj for each bj in B if
    bi != bj and dis(bi, bj) < bi.dv and ang(bi, bj) < bi.av]
  return deli
end
```

$$\mathbf{s}_i = -\sum_{b_j \in \partial_i : s_{ij} < d_s} (\mathbf{p}_j - \mathbf{p}_i) \quad s_{ij} = |\mathbf{p}_j - \mathbf{p}_i|$$

```
function seperate(boid bi, set deli)
  si = zeros(dim(bi))
  for each bj in deli
    if dis(bi, bj) < bi.ds
      si -= bj.pos - bi.pos
    end
  end
  return si
end
```

$$\mathbf{a}_i = \frac{1}{|\partial_i|} \sum_{b_j \in \partial_i} \mathbf{u}_j - \mathbf{u}_i$$

```
function align(boid bi, set deli)
  ai = zeros(dim(bi))
  if isempty(deli)
    return ai
  end
  for each bj in deli
    ai += bj.vel
  end
```

```
  return ai/length(deli) - bi.vel
end
```

$$\mathbf{c}_i = \frac{1}{|\partial_i|} \sum_{b_j \in \partial_i} \mathbf{p}_j - \mathbf{p}_i$$

```
function cohere(boid bi, set deli)
  ci = zeros(dim(bi))
  if isempty(deli)
    return ci
  end
  for each bj in deli
    ci += bj.pos
  end
  return ci/length(deli) - bi.pos
end
```

```
function flock!(set B, time tau)
  for each bi in B
    ...
    wd = wind(wind_accel)
    wr = wander(bi)
    bp = bound_position(bi)

    bi.vel += bi.ws*si + bi.wa*ai + bi.wc*ci + wd + wr + bp
    limit_velocity!(bi)
    ...
  end
end
```

$$|\mathbf{u}_i'| > u_{\max} \rightarrow \mathbf{u}_i' \rightarrow (\mathbf{u}_i'/|\mathbf{u}_i'|)u_{\max}$$

```
function limit_velocity!(boid bi)
  vmag = norm(bi.vel)
  if vmag > bi.vmax
    bi.vel *= bi.vmax/vmag
  end
end

function wind(vec accel)
  return accel
end

function wander(boid bi)
  return bi.wf*bi.vel*rand(dim(bi))
end

function bound_position(boid bi)
  dv = zeros(dim(bi))
  for (i, pi) in enumerate(bi.pos)
    if pi < bi.pmin[i]
      dv[i] = bi.vmax
    else if pi > bi.pmax[i]
      dv[i] = -bi.vmax
```

```
    end
  end
  return dv
end
```

$$\partial_i^{\mathrm{nn}} = \{b_{j\neq i} \in \mathcal{B} \colon |\mathbf{p}_i - \mathbf{p}_j| \leqslant d_{\mathrm{nn}}(n_c)\} \quad d_{\mathrm{nn}}(n) = \max\left(d \colon \#\{b_{j\neq i} \in \mathcal{B} \colon |\mathbf{p}_i - \mathbf{p}_j| \leqslant d\} = n\right)$$

```
function neighbors(boid bi, set B)
  deli = partialsort(B, 2:(bi.nc + 1), by = bj -> dis(bi, bj))
  return deli
end
```

```
function flock!(set Ba, set Bb, time tau)
  for each bi in Ba
    ...
    pe = evade(bi, Bb)
    bp = bound_position(bi)

    bi.vel += bi.ws*si + bi.wa*ai + bi.wc*ci + ev + bp
    limit_velocity!(bi)
    ...
  end

  for bi in Bb
    si = seperate(bi, Bb)
    pu = pursuit(bi, B0, c=tau)
    bp = bound_position(bi)

    bi.vel += bi.ws*si + pu + bp
    limit_velocity!(bi)
    ...
  end
end
```

$$\Delta \mathbf{u}_i = \frac{\mathbf{p}'_{ic}}{|\mathbf{p}'_{ic}|} u - \mathbf{u}_i \quad \mathbf{p}'_{ic} = \boldsymbol{c}'_g - \mathbf{p}_i \quad \boldsymbol{c}'_g = \sum_{j:|\mathbf{p}_{ij}|<d_p} (\mathbf{p}_j + c\,|\mathbf{p}_{ij}|\,\mathbf{u}_j)$$

```
function pursuit(bi, O; c)
  cg = zeros(dim(bi))
  for each bj in O
    if dis(bi, bj) < bi.ed
      cg += bj.pos + c*dis(bi, bj)*bj.vel
    end
  end
  if cg != zeros(dim(bi))
    pit = cg - bi.pos
    dv = (pit/norm(pit))*bi.vmax - bi.vel
    return dv
  else
    dv = zeros(dim(bi))
    return dv
  end
end
```

$$\Delta \mathbf{u}_i = -\frac{\mathbf{p}'_{it}}{|\mathbf{p}'_{it}|} u - \mathbf{u}_i \quad \mathbf{p}'_{ij} = c\,|\mathbf{p}_{ij}|\,\mathbf{u}_t - \mathbf{p}_i \quad j = \min\left(k: |\mathbf{p}_{ik}| < d_e\right)$$

```
function evade(bi, O; c)
```

```
  bj = partialsort(O, 1, by = bj -> dis(bi, bj))
  if dis(bi, bj) < bi.ed
    pit = (bj.pos + c*dis(bi, bj)*bj.vel) - bi.pos
    dv = -(pit/norm(pit))*bi.vmax - bi.vel
    return dv
  else
    dv = zeros(dim(bi))
    return dv
  end
end
```

2 Ρεαλισμός

τάξη ή πόλωση ομάδας

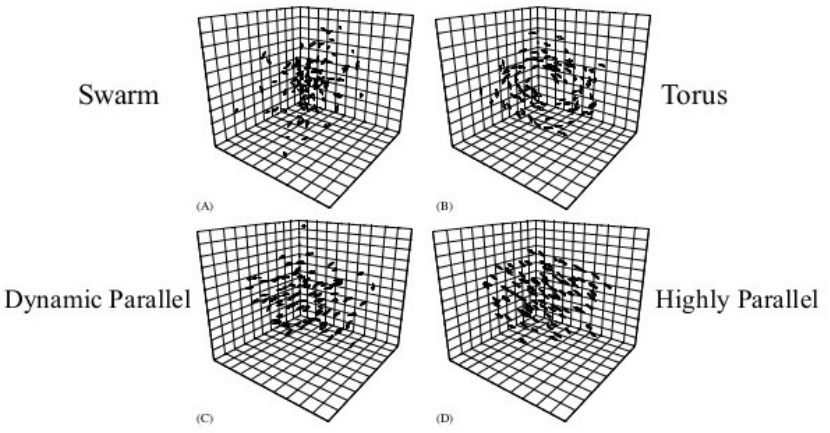
$$p_g(t) = \frac{1}{n}\left|\sum_i \frac{\mathbf{u}_i(t)}{|\mathbf{u}_i(t)|}\right|$$

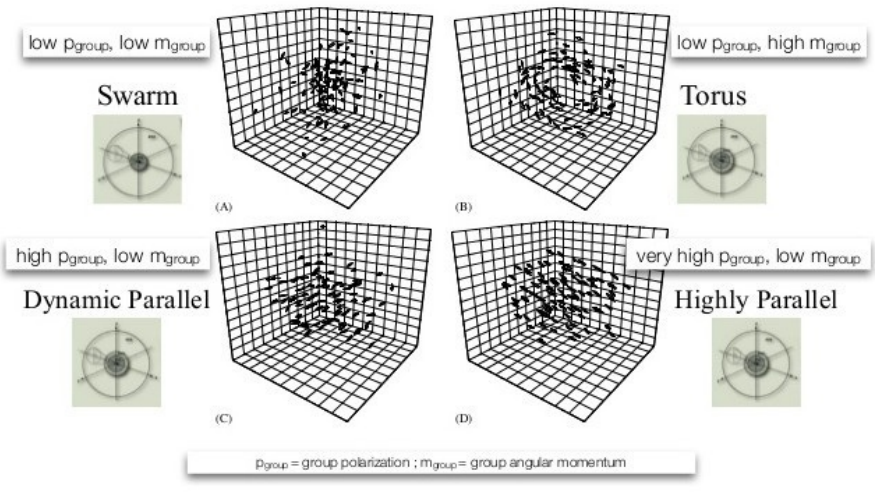
τροαχιακή ορμή ομάδας

$$m_g(t) = \frac{1}{n}\left|\sum_i \mathbf{r}_{ic}(t) \times \mathbf{u}_i(t)\right| \quad \mathbf{r}_{ic} = \mathbf{r}_i - \mathbf{c}_g \quad \mathbf{c}_g(t) = \frac{1}{n}\sum_i \mathbf{r}_i(t)$$

επέκταση σμήνους

$$e(t) = \frac{1}{n}\sum_i |\mathbf{r}_i - \mathbf{c}_g| \quad \mathbf{c}_g(t) = \frac{1}{n}\sum_i \mathbf{r}_i(t)$$


Swarm
Torus
(A)
(B)
Dynamic Parallel
Highly Parallel
(C)
(D)


low p_{group}, low m_{group}
low p_{group}, high m_{group}
Swarm
Torus
(A)
(B)
high p_{group}, low m_{group}
very high p_{group}, low m_{group}
Dynamic Parallel
Highly Parallel
(C)
(D)
p_{group} = group polarization ; m_{group} = group angular momentum

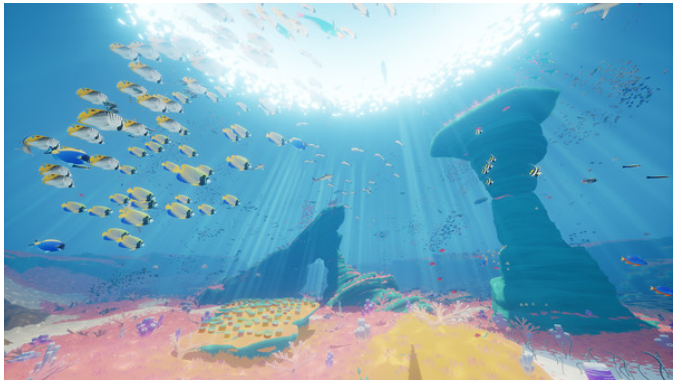
3 Εφαρμογές

– κινούμενα γραφικά (animation)

ταινίες, π.χ. Batman Returns (1992)

τηλεοπτικές σειρές, π.χ. Doctor Who, "A Christmas Carol"

βιντεοπαιχνίδια, π.χ. Abzû



– ρομποτικά σμήνη (swarm robotics)

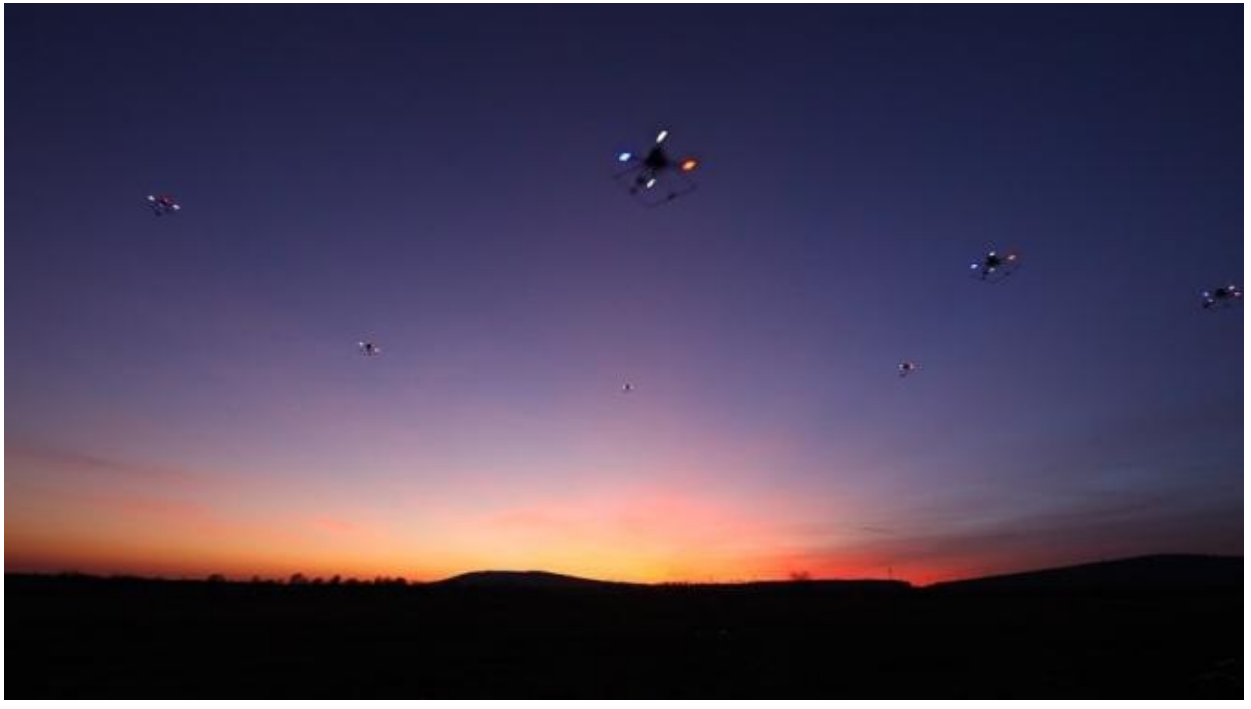
π.χ. μη-επανδρωμένα ιπτάμενα οχήματα (drones)

*γενικότερα εφαρμογές νοημοσύνης σμήνους

- βιοϊατρική, π.χ. καταστροφή καρκινικών κυττάρων
- αεροδιαστημική μηχανική κι εξερεύνηση

  σχηματισμούς νανοδορυφόρων, αυτο-συναρμολόγηση κι ανατροφοδότηση μεγάλων δορυφόρων, κατανεμημένα διαστημικά τηλεσκόπια, και χαρτογράφηση πλανητών
- παρακολούθηση τοπίων
- γεωργική άρδευση και ράντισμα
- εκτύπωση σμήνους

Τεχνητά ρεαλιστικά σμήνη με με αυτόνομα drones



Προβλήματα σε σχέση με εικονικά σμήνη είναι ο θόρυβος και η χρονοκαθυστέρηση.

Ρομποτικό σμήνος με φυσική συμπεριφορά μόλις μερικά χρόνια πριν.

Για εξωτερικούς χώρους η κίνηση εμπνέεται από πτηνά.

Για εσωτερικούς η κινηματική πληροφορία μπορεί να εξαχθεί από μύγες.