Boids — A Simple Way to Simulate How Birds Flock in Processing

Takuma Kakehi Follow
Nov 28, 2019 · 4 min read ★

It is mysterious how birds fly together in groups, heading in unpredictable directions at the same time. This behavior of flocks is not the result of one leader passing instructions to others, nor is it a result of all the birds knowing where they are heading to- instead, it is the cumulation of each bird's reaction to its own immediate neighbor. For instance, birds try in general to stay close to their neighbors to protect themselves from their predators. However, they, of course, try to avoid crashing into each other.

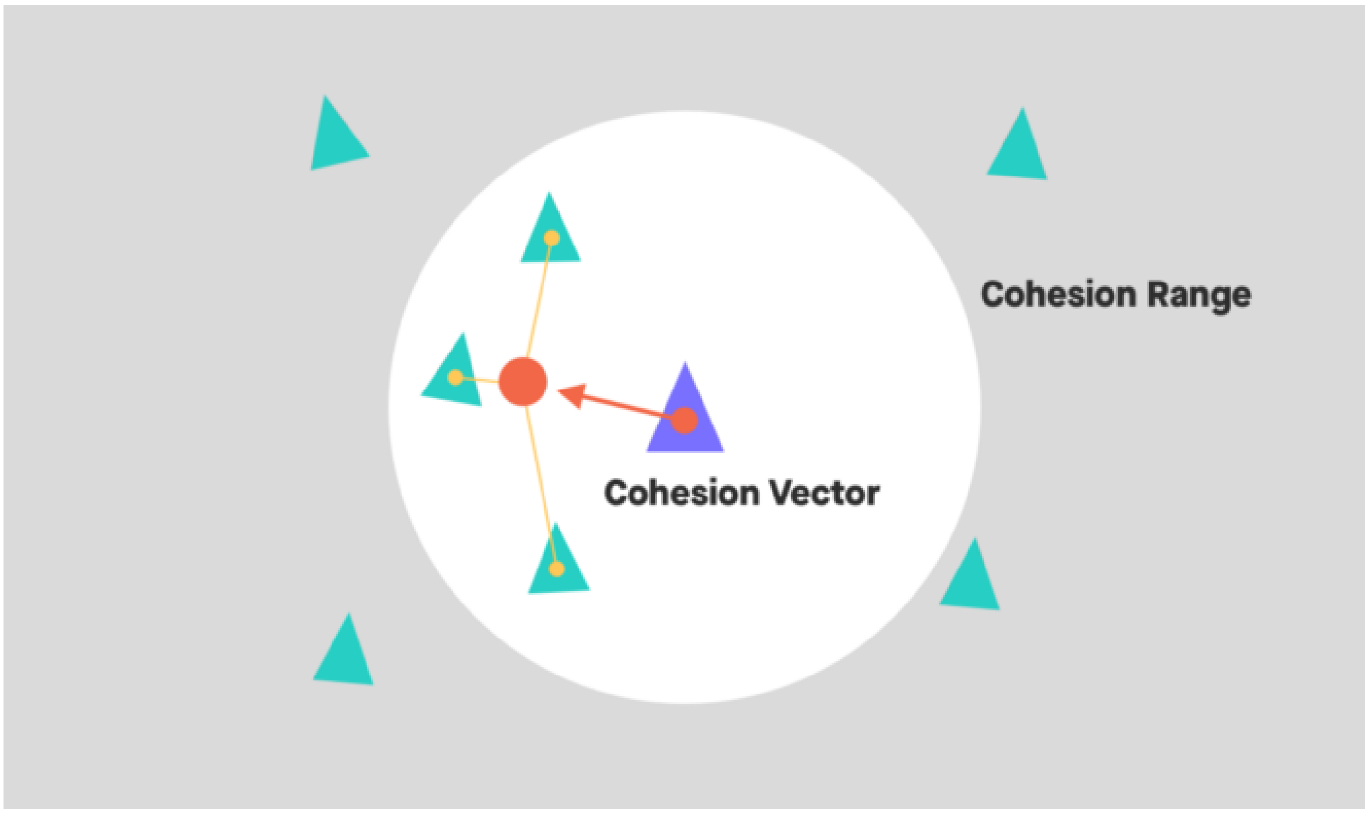
This rather emergent behavior is known as the flocking system. I have probably tried creating flocking systems over ten times in various coding languages throughout my career. However, I've never had the chance to document how a simple flocking system works and can be simulated. In this entry, I would like to document how the most simple flocking system can be recreated.

Boids

Boids, "bird-oid objects", is the combination of simple rules that simulates flock behavior. *Boids* was originally introduced by computer graphics expert, Craig Reynolds, who also worked on scenes for the original Tron movie from 1982 and Batman Returns in 1992. It consists of three fundamental rules: ***Cohesion, Alignment,*** and ***Separation***. Just like many other emergent behaviors, each bird can only register and apply these rules to its immediate neighbors within the limited ranges. The following explains how each rule affects each bird.

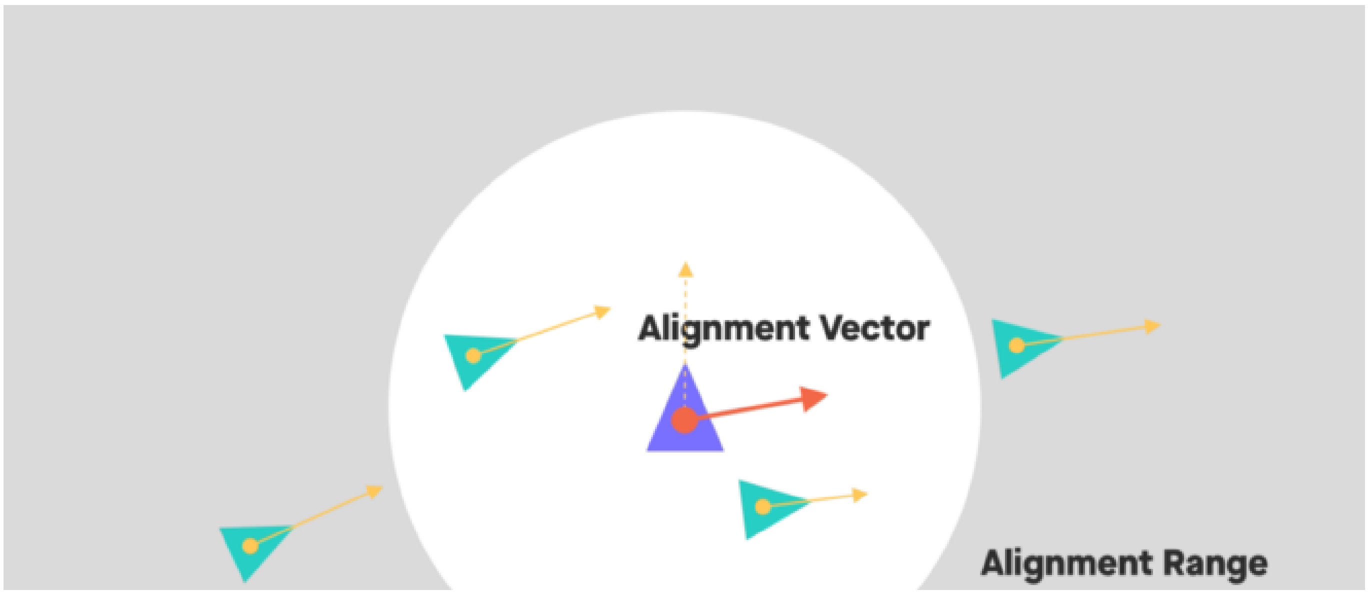
Cohesion

Each bird tries to stay close to the other birds in the mass. When they register their neighbors, this rule tries to get each to come to the center of their neighbors defined space.



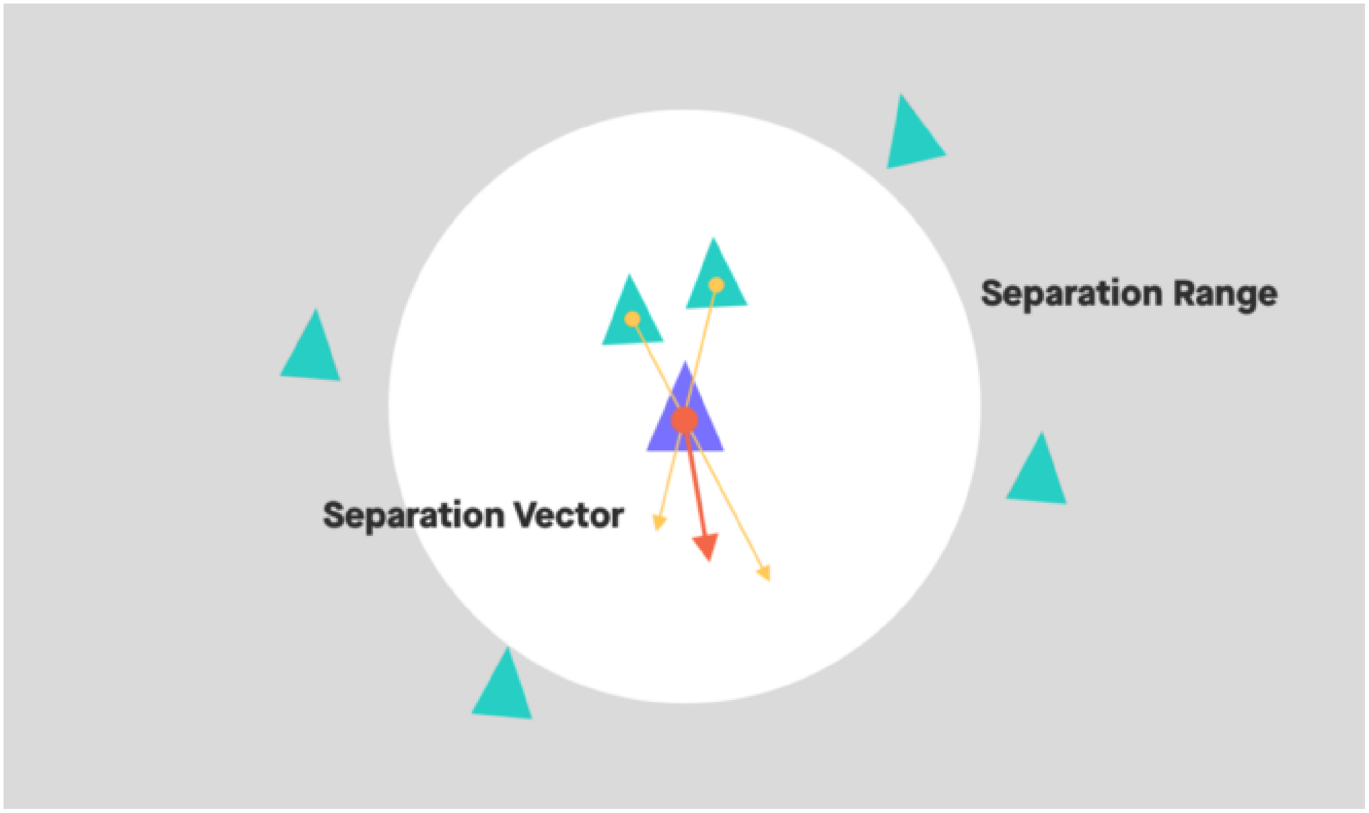
Alignment

Each bird is flying in some direction. When they see others, this rule gets each bird to try to align their direction based on their immediate neighbor.



Separation

When birds get too crowded, this rule tries to keep enough of a boundary around each bird to avoid collisions.



Pseudo Code

Below pseudo code simply explains what each boid is/does.

void Setup

```
Initialize x numbers of **Boid** class
```

class Boid

```
Find its neighbor boids that are in the range and run the following
methods.

Run Cohesion method and get the vector value

Run Separation method and get the vector value

Run Alignment method and get the vector value

Combine the Cohesion, Separation and Alignment vectors as an
acceleration

Add acceleration to its vector

Move its position with its vector
```

vector Cohesion (this boid's neighbors)

```
If there are no neighbors, return 0 vector.

Find the center position of its neighbors,
then find a new vector from this boid's position to this center
position.

return vector.
```

vector Separation (this boid's neighbors)

```
If there are no neighbors, finish here and return 0 vector.

For each of its neighbor boid,
find a vector directing from each of its neighbor boid's position to
this boid's position.

If the above vector is greater than 0,
the force should be inversely proportional to the distance.

Combine all forces calculated for each neighbor boid

return vector.
```

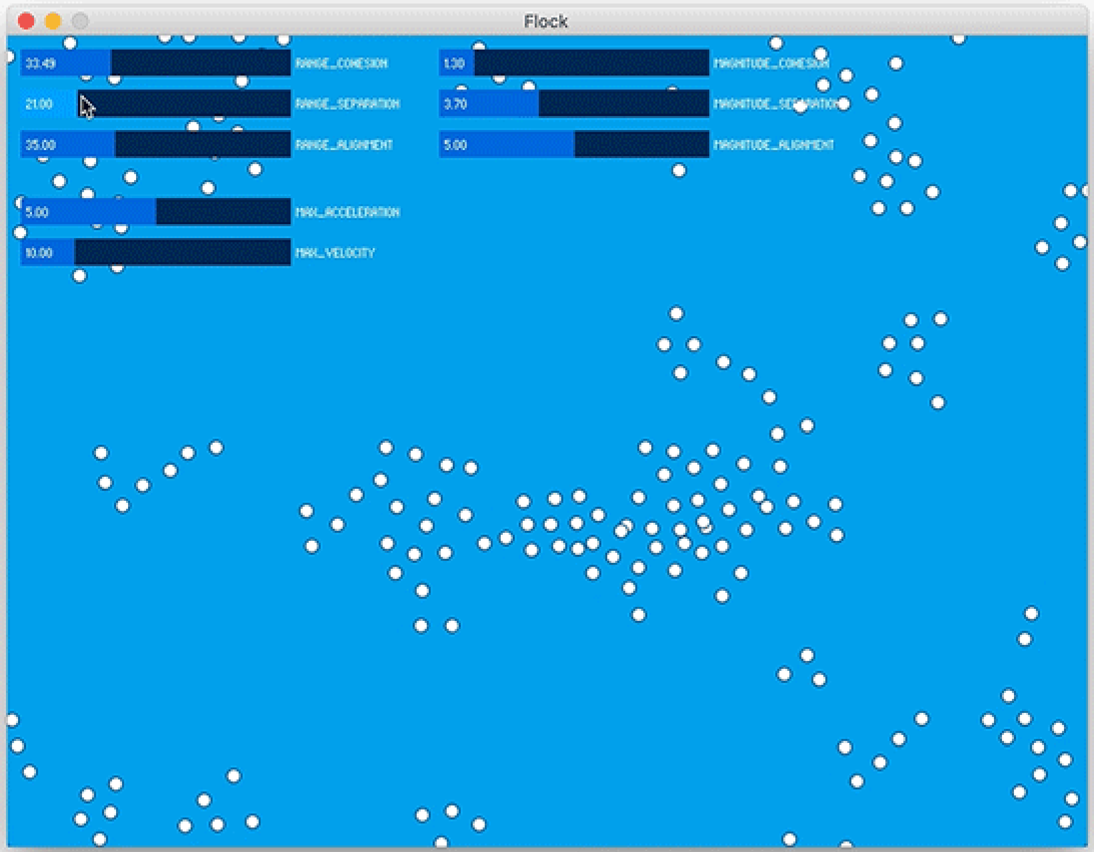
vector Alignment (this boid's neighbors)

```
If there are no neighbors, finish here and return 0 vector.

For each of its neighbor boid,
get its current vector and combine all to find the average vector
within its neighbors

return vector;
```

Final codes

These are my final codes for the project (pde file on github). I also uploaded the version that runs in Unity3D. The version in Unity3D was made possible by the Udemy lecture: Game Devs Unleash Artificial Intelligence: Flocking Agents by Razvan Pistolea. Please subscribe to his lecture to fully comprehend the flocking system. In addition, the lecture provides an advanced version of the flocking system.

Cohesion



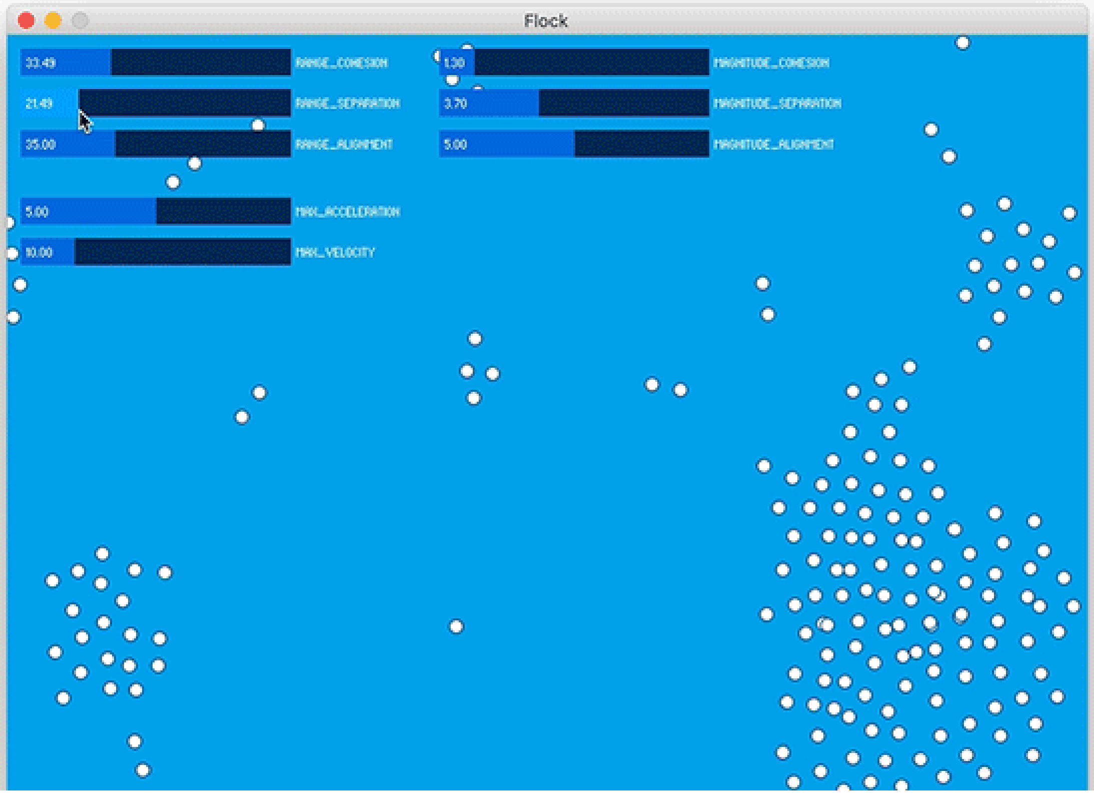
```
33.49  RANGE_COHESION        1.30  MAGNITUDE_COHESION
21.00  RANGE_SEPARATION      3.70  MAGNITUDE_SEPARATION
35.00  RANGE_ALIGNMENT       5.00  MAGNITUDE_ALIGNMENT

5.00   MAX_ACCELERATION
10.00  MAX_VELOCITY
```

Increasing the magnitudes of Cohesion keeps units together.

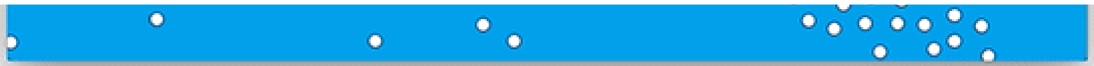
```
PVector cohesion(ArrayList<Boid> neighbors){
  PVector r = new PVector();

  // if there are no neighbors, finish here and return 0 vector
  if (neighbors.size() == 0) {
    return r;
  }

  // find the center position of this boid’s neighbors
  for (Boid other : neighbors) {
    r = PVector.add(r, other.pos);
  }
  r = PVector.div(r, neighbors.size());

  // find a new vector from this boid’s position to this center position
  r = PVector.sub(r, this.pos);

  // return the normalized vector
  return r.normalize();
}
```

boids-cohesion.pde hosted with ❤ by **GitHub** view raw

Separation



```
33.49   RANGE_COHESION        1.30   MAGNITUDE_COHESION
21.49   RANGE_SEPARATION      3.70   MAGNITUDE_SEPARATION
35.00   RANGE_ALIGNMENT       5.00   MAGNITUDE_ALIGNMENT
5.00    MAX_ACCELERATION
10.00   MAX_VELOCITY
```

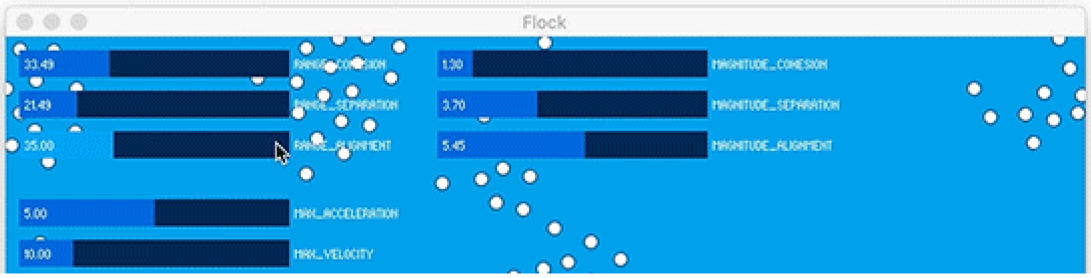


Increasing the range of Separation disrupts the aligned movements.

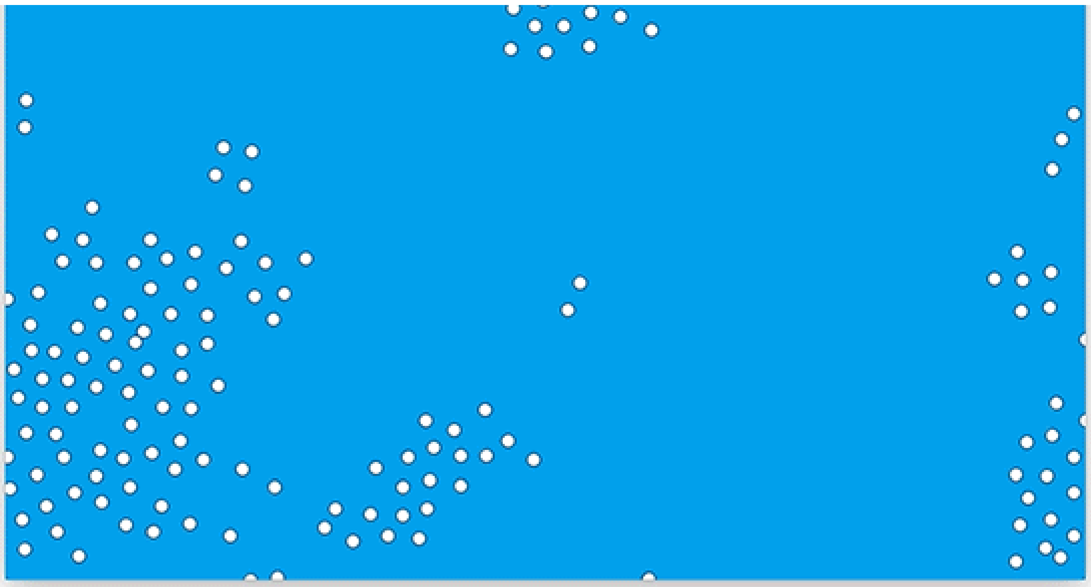
```
PVector separation(ArrayList<Boid> neighbors) {

    PVector r = new PVector();

    // if there are no neighbors, finish here and return 0 vector
    if (neighbors.size() == 0) {
      return r;
    }

    // add the contribution of each neighbor towards me
    for (Boid other : neighbors) {

      PVector towardsMe = new PVector();
      towardsMe = PVector.sub(this.pos, other.pos);

      // force contribution will vary inversely proportional
      if (towardsMe.mag() > 0) {

        // to distance or even the square of the distance
        r = PVector.add(r, PVector.div(towardsMe.normalize(), towardsMe.mag()));
      }
    }

    // return normalized vector
    return r.normalize();
}
```

boids-separation.pde hosted with ❤ by GitHub view raw

Alignment



33.49 RANGE_COHESION
21.49 RANGE_SEPARATION
35.00 RANGE_ALIGNMENT
5.00 MAX_ACCELERATION
10.00 MAX_VELOCITY
1.30 MAGNITUDE_COHESION
3.70 MAGNITUDE_SEPARATION
5.45 MAGNITUDE_ALIGNMENT



Increasing the magnitude of Alignment makes the movements more straight.

```
PVector alignment(ArrayList<Boid> neighbors) {

    PVector r = new PVector();

    // if there are no neighbors, finish here and return 0 vector.
    if (neighbors.size() == 0) {
      return r;
    }

    // for each of its neighbor boid, get its current vector and combine all to find the
    for (Boid boid : neighbors) {
      r = PVector.add(r, boid.v);
    }

    // return normalized vector
    return r.normalize();
}
```

boids-alignment.pde hosted with ❤ by GitHub view raw

Original post: http://www.ta-kuma.com/programming/boids-a-simple-way-to-simulate-how-birds-flock-in-processing/

References

Boids: Flocking made simple | Hermen de Weerd

Why do birds flock together? | How it works

Boids — Stanford Computer Science

Game Devs Unleash Artificial Intelligence: Flocking Agents | Udemy

Flocking Bois 1920 Simulation Processing Unity3d