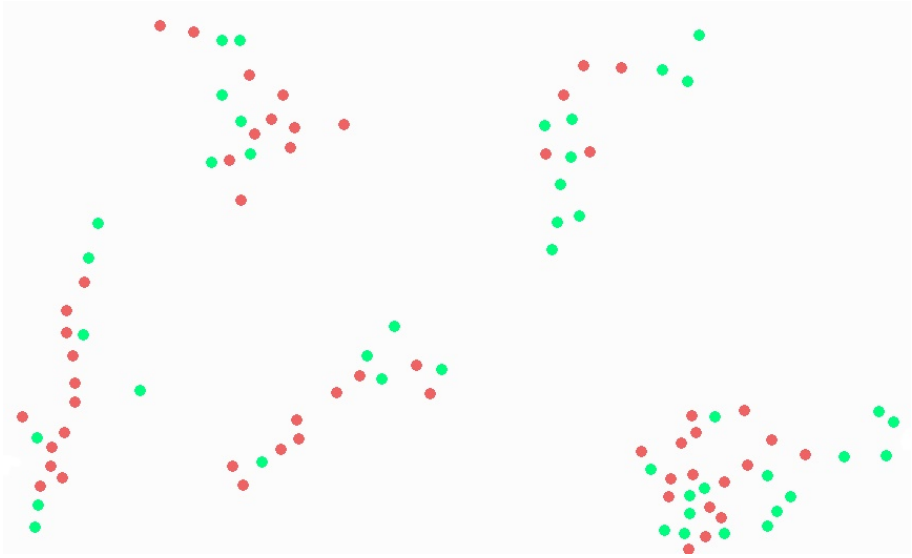
# Assignment: Swarm Robots
## Biologically Inspired Autonomous Systems

Karina Bay Bloch (Kablo17) & Nicoline L. Thomsen (nthom18)

18/05-2022



## Contents

# 1 Introduction

Swarms are what is seen in nature as groups of creatures acting according to each other to work effectively together. This kind of synchronised group behaviour can be seen with ants and birds. When birds fly in flocks, they fly as to not collide with each other, and the waves seen from the outside is the result of nothing but individual birds following, yet avoiding, each other.

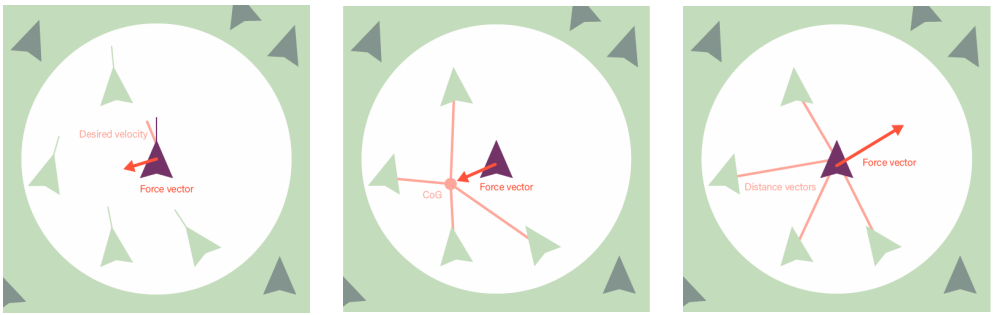
These behaviours are what is sought after in swarm robotics. If each robot is responding depending on its neighbours with its own "brain", instead of all robots responding with the same "brain", the system will be able to flow more smoothly and redundant.

For this assignment, there will not be physical robots in play, but simulations of such instead called boids. Here, a simulation platform, that is constructed in Python, will be used to show the behaviours of simplistic 2D boids. The simulation environment is 2D to allow fast simulations, and will be a toroidal environment, so that the tests are consistent in swarm sizes. These sizes are self-chosen together with the neighbourhood distances, which is the distance in which each boid can see neighbouring boids.

All code used in this project can be found in the GitHub repository `https://github.com/PositiveBeat/BioAs-Projects-F22.git`, in the folder "Swarm Robotics".

# 2 2D Boids

The Boids Algorithm implement behaviour rules the agents, or "boids", follow. The three rules are alignment, cohesion and separation, illustrated in figure 1. The concepts of the boids algorithm in source [1] are described mathematically below.




(a) Alignment (b) Cohesion (c) Separation

Figure 1: Three rules of the boids algorithm, the red arrow is the applied force vector

## Alignment

This behaviour aims to align all agents, making them move in the same direction with the same speed. The calculation is seen in equation 1.

$$\vec{F_a} = \frac{1}{n-1} \sum_{i=1}^{n-1} (\vec{\mathrm{V}}_i) - \vec{V}_{\text{current}} \tag{1}$$

$\vec{F_a}$ is the alignment force, $n$ is the number of perceivable boids including itself, $\vec{V_i}$ is the velocity vector of the individual boids in the flock and $\vec{V}_{\text{current}}$ is the velocity vector of the current boid.

## Cohesion

Cohesion makes the agents stay in close proximity together by creating a force that points toward the centre of mass of perceived flock mates. The equations 2 and equation 3 describes how this is calculated.

$$\vec{\text{com}} = \frac{1}{n-1}\sum_{i=1}^{n-1}(\vec{P}_i) - \vec{P}_{\text{current}} \tag{2}$$

$\vec{\text{com}}$ is the vector pointing to the centre of mass, $n$ is the number of boids in the local flock including self, $\vec{P}_i$ is the position vector of the individual boids in the flock and $\vec{P}_{\text{current}}$ is the position vector of the current boid.

The steering vector itself is then calculated by subtracting the vector pointing to CoM with the current velocity vector.

$$\vec{F}_c = \vec{\text{com}} - \vec{V}_{\text{current}} \tag{3}$$

$\vec{F}_c$ is the cohesion force and $\vec{V}_{\text{current}}$ is the velocity vector of the current boid.

### Separation

To avoid collisions and ensure a spread between the agents, the separation rule pushes the agents away from each other. This calculation is described in equation 4.

$$\vec{F}_s = \sum_{i=1}^{n-1}\left(\frac{\vec{P}_{\text{current}} - \vec{P}_i}{|\vec{P}_{\text{current}} - \vec{P}_i|^2}\right) \tag{4}$$

$\vec{F}_s$ is the separation force, $n$ is the number of perceivable boids including itself, $\vec{P}_i$ is the position vector of the individual boids in the flock and $\vec{P}_{\text{current}}$ is the position vector of the current boid.

### Net Force

The three rules are weighed and summed to obtain the final force vector.

$$\vec{F}_{\text{net}} = w_a \cdot \vec{F}_a + w_c \cdot \vec{F}_c + w_s \cdot \vec{F}_s \tag{5}$$

$\vec{F}_{\text{net}}$ is the net force vector, $w$ is the scalar weighting for each rule.

## 2.1 Behaviour Implementation

The code simulating the boids are divided into two classes, one drawing the visuals and containing variables like position, velocity and acceleration. The other is handling the behavioural aspect of the boids, iteration through them all and applying update rules depending on the current state, describing the surrounding boids. Each rules for the boid algorithm, alignment, cohesion and separation is individually calculated, normalised and scaled according to the maximum force. These boids are for simulation purposes only, so collisions between boids is not a concern. Therefore the update rule is simply a summation of the output of these three rules, and no priority based behaviour was implemented. Each boid has a perception radius, within which neighbouring boids are included in the force calculations. The separation works with a perception radius of one fourth of the full perception.

The world the boids inhabit is a toroidal square of 864x864 pixels, and each boid is a point of a single pixel. The visual representation however makes them appear larger to view them properly.

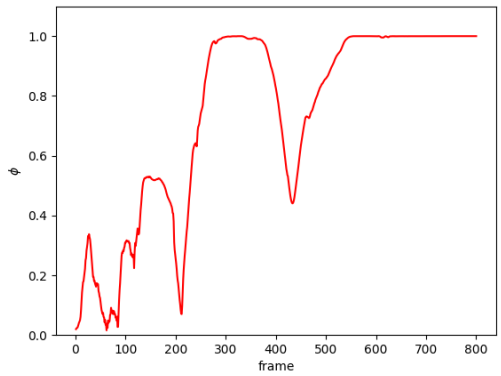
## 2.2 Simulation Tests

To experiment how different weights of the three rules are affecting flocking behaviour, a test script was written, that runs the simulation with combinations of different parameters, including flock size and perception radius. To determine when flocking occurs, the metric in equation 6[2] is

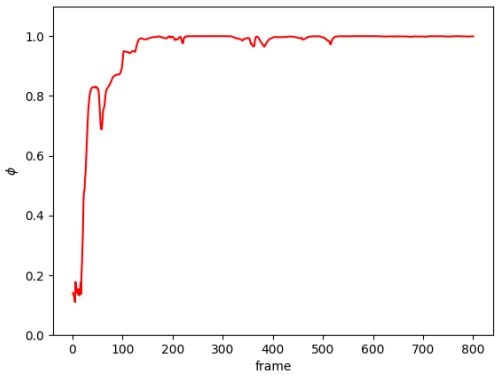
logged over time. It quantifies the flocking behaviour using the normalised average of velocity at any given time. A value of one is high flocking, and a value of zero would be random movement.

$$\phi = \frac{1}{N} \left| \sum_{i=1}^{N} \frac{V_i}{|V_i|} \right| \tag{6}$$

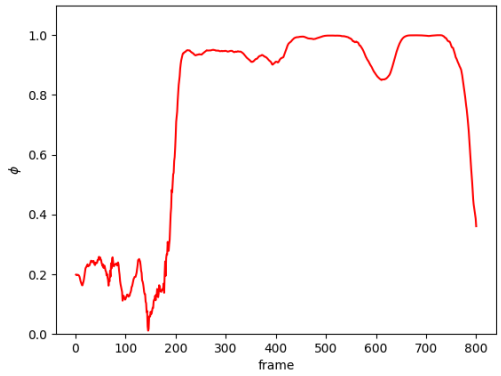
In total 9 tests simulations was run to study the different parameters effect on a randomly initialised system. The first four tests explores flock size and perception radius, with constant boid parameters, all set to an equal value, in this case 10. The two flock sizes tested was of 25 and 100 boids, and the perception radius was 200 and 1000 pixels. Figure 2 shows the graphs of those four tests.



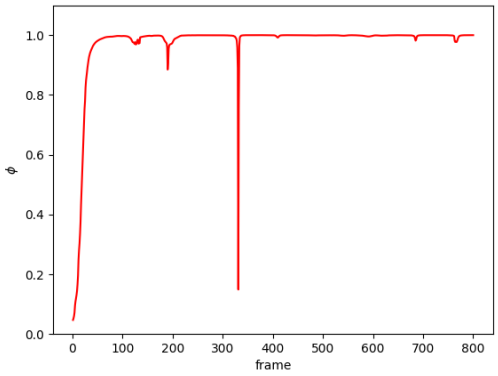

(a) Flock size: 25, Perception radius: 200px




(b) Flock size: 25, Perception radius: 1000px




(c) Flock size: 100, Perception radius: 200px

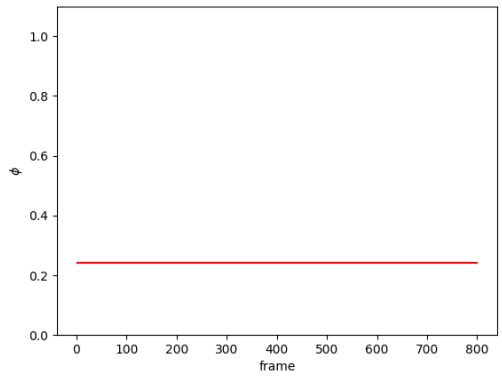



(d) Flock size: 100, Perception radius: 1000px

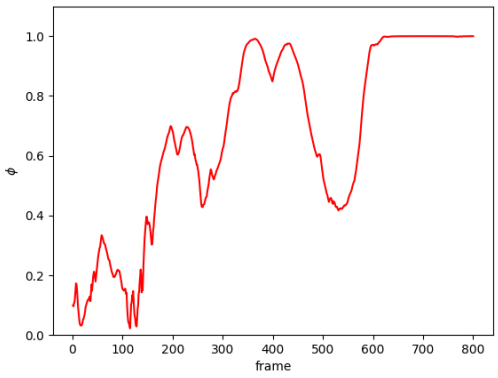
Figure 2: Flocking measure $\phi$ over 800 frames of simulation for different combinations of flock sizes and perception radii

From these tests it is seen that a higher perception will result in good, steady flocking behaviour, and reaching it fast. This is as excepted, as for lower perception the flock is more likely to split into smaller groups, and when they collide cause the flocking measure $\phi$ to drop. The flock size does not seem to have a great effect when the perception is high, but for lower perceptions more flock mates seem to improve overall performance, but with higher risk of causing drops in $\phi$.

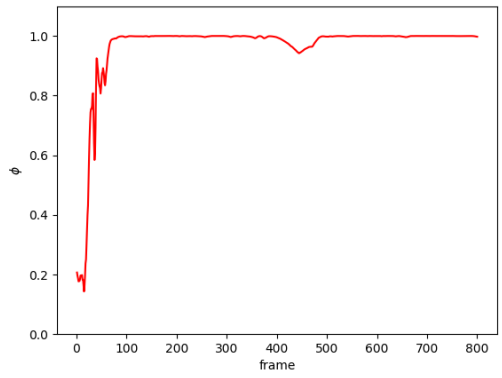
The remaining five tests are to study the impact of changing the weights of the three boids rules, where flock size and perception are fixed. Values for those are chosen to be a flock size of 25, and a perception of 200. The five graphs from this test is shown in figure 3.



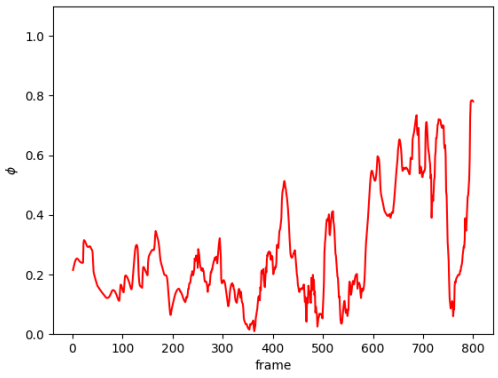

(a) Alignment: 0, cohesion: 0, separation: 0



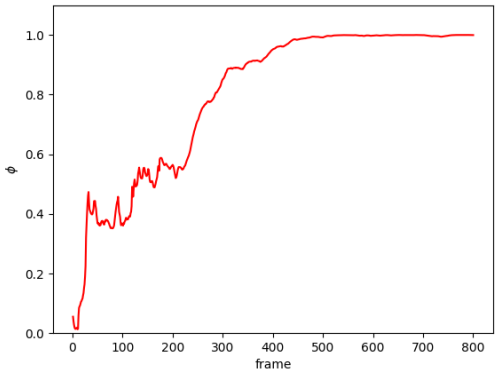

(b) Alignment: 10, cohesion: 10, separation: 10




(c) Alignment: 20, cohesion: 5, separation: 5




(d) Alignment: 5, cohesion: 20, separation: 5



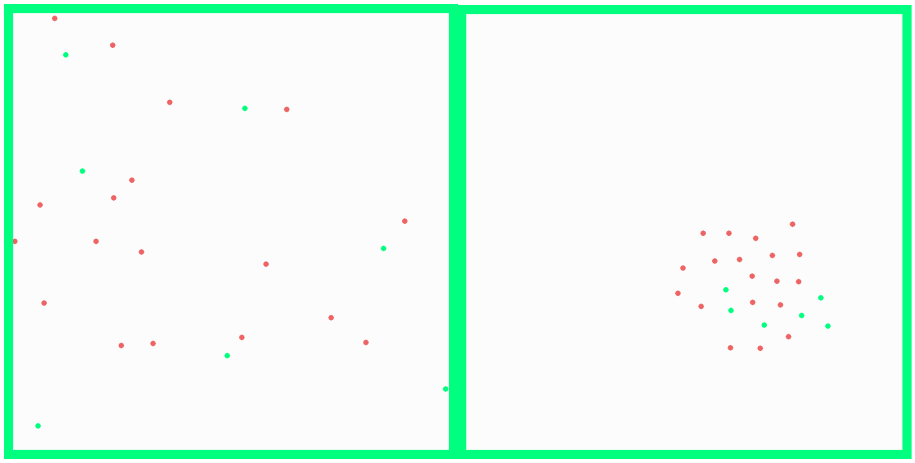

(e) Alignment: 5, cohesion: 5, separation: 20

Figure 3: Flocking measure $\phi$ over 800 frames of simulation for different combinations of weights

The first test in figure 3a confirms that without any flocking rules, the value of $\phi$ is a constant which value is at the mercy of the randomly initiated velocities of the system. With all weights being equal, it takes a long time to achieve flocking, but it converges to a value of before the end of the simulation. With alignment being the dominant factor the flocking is quick and consistent. However from investigating screenshots taken during the test, it is seen that all boids are indeed all moving in the same direction, but they are spread over the board, and not trying to form clusters. If this is considered good flocking will be application specific. With cohesion being the highest weight, they never achieve good flocking, as they all seek the middle of the flock with too much

momentum to stop when they reach it. This causes high fluctuation of $\phi$ in this test. The last test has alignment and cohesion balanced, and separation as the highest value. Flocking is achieved at a somewhat slow rate, all the boids however end up forming one cluster moving as one.

The two best tests are the ones in figure 3c and 3e. To visualise the state of the flock after flocking has occurred, screenshots from the simulation are shown in figure 4.



(a) Test in figure 3c (b) Test in figure 3e

Figure 4: Screenshot from simulation at frame 500. All boids in this frame are moving in the same direction. Green and red dotes are treated indifferently

# 3 Firefly-inspired synchronization

## 3.1 Behaviour Implementation

The second part of the assignment were to assign a firefly-like flashing behaviour to the constructed boids. They are supposed to blink for a short amount of time, where they will afterwards try to synchronise with each other, when getting close to and after spotting their differing blinking patterns. To make this work, they will all act as discrete pulse-coupled oscillators, by the formula in equation 7 from [3].

$$x_i(n+1) = x_i(n) + \frac{1}{T} + \epsilon \cdot \alpha_i(n) \cdot h(x_i(n)) \tag{7}$$

There are some constants that needs to be defined, such as the $T$ being the period between flashes of an isolated boid, $\epsilon$ defining the strength of the coupling between between the oscillating boids. $\alpha_i(n)$ defines the number of flashing boids seen by the boid i at control step n.

This implementation is built as an expansion of the boids simulation. The boids now has an added attribute for synchronisation that contains the value of $x_i(n)$, and a function to visualise the blink. The behaviour script has an added function that implements equation 7. Table 1 shows the chosen values for this implementation.

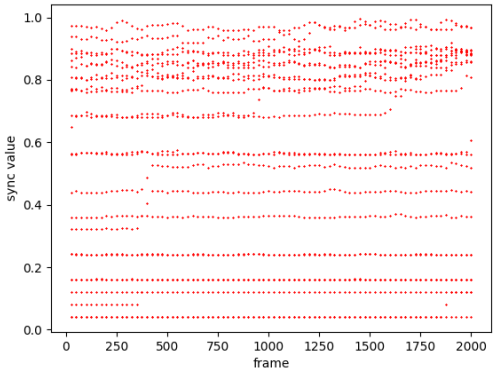
| | |
|---|---|
| $T$ | 25 |
| $\epsilon$ | 0.01 |

Table 1: Chosen values of variables in equation 7

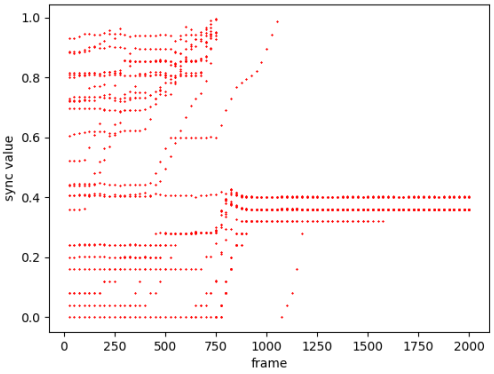
## 3.2 Simulation Tests

The purpose, of testing the synchronisation abilities of the boids, is to see and notify, how long it takes to achieve global synchronisation in the swarms. These tests will have either the swarm

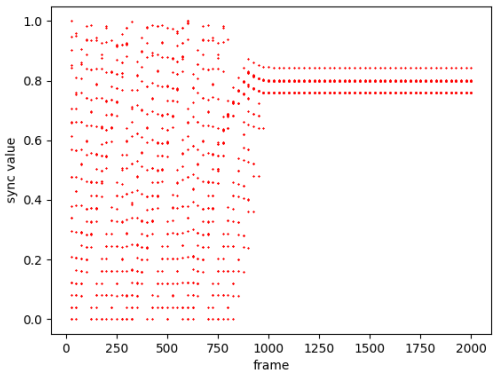
size, the perception radius or the three behaviours changed each time, so that difference between the synchronisations will be spotted more easily. In 5a there is a flock size of 25 boids, that have a perception of 200 pixels, with all other values being 0. Although the time is passing, the boids are not really synchronising, and there is not much change to be seen. As soon as behaviour values are added, the boids start synchronising within 750 frames, as seen in 5b. However, if the perception radius is increased to something unusual such as 1000 pixels in 5c, which is a radius, that is exceeding the world size they inhabit. The boids aren't capable of synchronising at first, but seems to catch on to it after 800 frames. If behaviour values are added, the synchronisation time is greatly reduced to 250 frames.



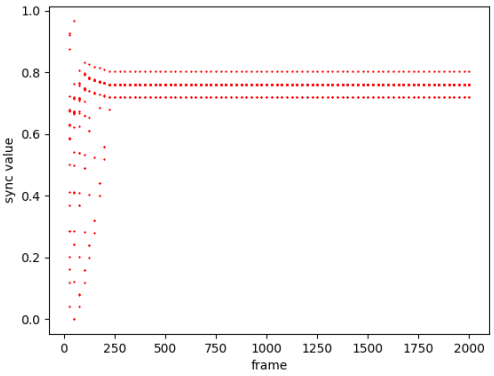

(a) Perception radius: 200px, alignment: 0, cohesion: 0, separation: 0




(b) Perception radius: 200px, alignment: 5, cohesion: 5, separation: 20



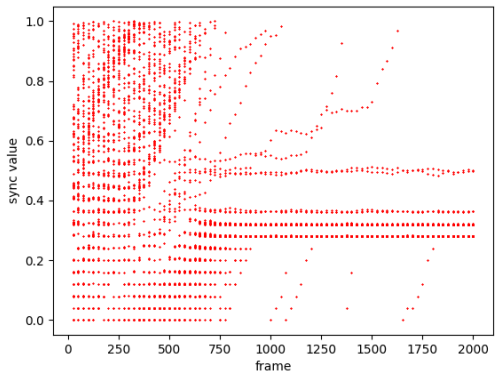

(c) Perception radius: 1000px, alignment: 0, cohesion: 0, separation: 0



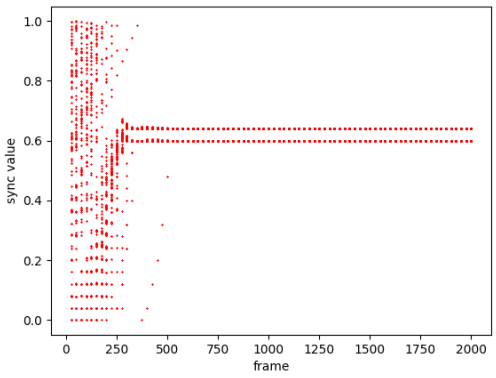

(d) Perception radius: 1000px, alignment: 5, cohesion: 5, separation: 20

Figure 5: Sync value for each boid sampled at the blink rate period. These tests share a flock size of 25

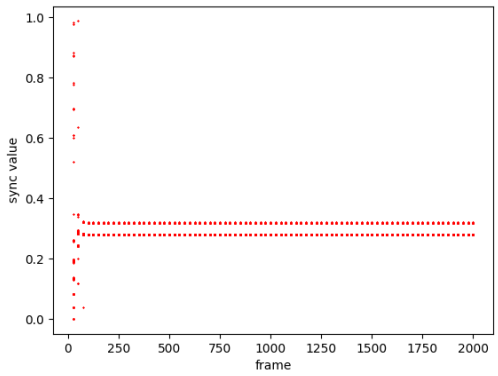
If the flock size were to be changed into 100 boids instead, the matching tests from figure5 seems to be working more effectively, which is most likely because of the increased amount of possible synchronisation targets, and the chaotic behaviour in 6c and 6d looks more precise and controlled.



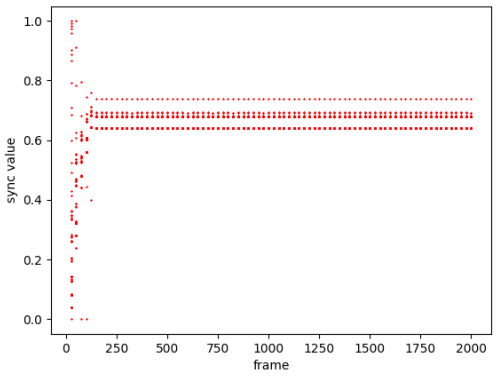

(a) Perception radius: 200px, alignment: 0, cohesion: 0, separation: 0




(b) Perception radius: 200px, alignment: 5, cohesion: 5, separation: 20




(c) Perception radius: 1000px, alignment: 0, cohesion: 0, separation: 0




(d) Perception radius: 1000px, alignment: 5, cohesion: 5, separation: 20

Figure 6: Sync value for each boid sampled at the blink rate period. These tests share a flock size of 100

# 4 Bibliography

[1] Craig W. Reynolds. Steering behaviors for autonomous characters. `https://www.red3d.com/cwr/papers/1999/gdc99steer.html`.

[2] Vijay Kumar and Rumi De. Efficient flocking: metric versus topological interactions. `https://royalsocietypublishing.org/doi/10.1098/rsos.202158`. Online; accessed 06 May 2022.

[3] From fireflies to fault tolerant swarms of robots. 2008. Anders Lyhne Christensen, Rehan O'Grady and Marco Dorigo. IEEE Transactions on Evolutionary Computation 13(4):754-766.



(ノ°□°)ノ ⌒┻━┻