Flocking

creating a basic flocking simulation in violet

python flocking algorithm

flocking based on pseudo code provided

Algorithm 1 Flocking Algorithm

1: **procedure** CHANGE POSITION
2: Check neighbour in radius R
3: **if** *NoNeighbours* **then**
4: Wandering
5: **end if**
6: $Alignment \leftarrow V_N - V_{boid}$
7: $Separation \leftarrow \frac{1}{|N|} \sum_{i \in N} (X_{boid} - X_i)$
8: $Cohesion \leftarrow f_C - V_{boid}$
9: $f_{total} \leftarrow \frac{\alpha Alignment + \beta Separation + \gamma Cohesion}{M_{boid}}$
10: $Move \leftarrow Move + f_{total}$
11: **if** $Move > MaxVelocity$ **then**
12: $Move.Normalize() * MaxVelocity$
13: **end if**
14: $Position \leftarrow Position + Position * \Delta t$
15: **end procedure**

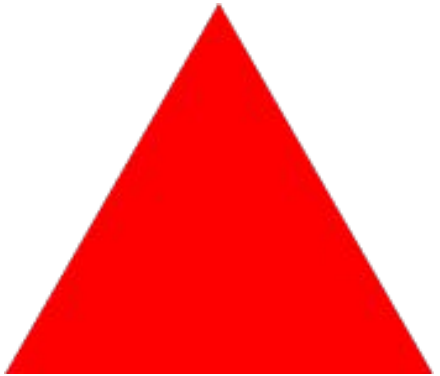
```
if self.in_proximity_accuracy().count() >= 1:
    neighbours=list(self.in_proximity_accuracy())
    #allignment
    sum_v=Vector2()
    sum_x=Vector2()
    sum_difference=Vector2()
    for i in neighbours:
        object=i[0]

        v=object.move
        sum_v+=v
        x=object.pos
        sum_x+=x
        sum_difference+=(self.pos-x)

    n=len(neighbours)
    avg_v=sum_v/n
    alignment_force=avg_v-self.move

    avg_x=sum_x/n
    cohesion_force=avg_x-self.pos
    separation=sum_difference/n

    alpha, beta, gamma = self.config.weights()

    total=(alpha*alignment_force)+(beta*separation)+(gamma*cohesion_force)
    ftotal=total/self.config.mass

    self.move += ftotal

    if Vector2.length(self.move) > MAX_VELOCITY:
        self.move=Vector2.normalize(self.move) * MAX_VELOCITY

    self.pos=self.pos+(self.move*self.config.delta_time)

else:
    self.pos+=self.move
```

obstacle implementation

making the environment interactive,
with api functions:

```
# for obstacle in self.obstacle_intersections:
if (list(self.obstacle_intersections())): #if not empty
    self.move += (self.pos - Vector2(400,300)).normalize()
```

```
sim.batch_spawn_agents(50, Bird, images=["images/bird.png"])
sim.spawn_obstacle("images/triangle@200px.png", 400, 300)
sim.run()
```



border control

the beginning of the change_position() function
copied from violet api

```
if not self._moving:
    return

changed = self.there_is_no_escape()

prng = self.shared.prng_move

# Always calculate the random angle so a seed could be used.
deg = prng.uniform(-30, 30)

# Only update angle if the agent was teleported to a different area of the simulation.
if changed:
    self.move.rotate_ip(deg)
```

observations

When bouncing off certain parts of the obstacle the birds increases velocity.

When a bird collides with the obstacle, its velocity is adjusted to move it away from the obstacle. This adjustment is added to the bird's current velocity, which could result in an increase in speed, especially if the bird was already moving in the direction of the adjustment.

Birds get more clumped towards the end and slower.

As birds flock together in the simulation, the interplay of separation, alignment, and cohesion forces can cause a slowdown. The separation force weakens as birds cluster, reducing speed. The alignment force can slow birds if they match pace with slower neighbors. The cohesion force, pulling birds towards their group's center, contributes to slowdown as birds reach a stable state of proximity.

Small mass lead to bigger flocks.

When there is a smaller mass the birds are more responsive to the forces from their neighbors, leading to bigger groups flocking together.

future improvements to the code

- prevent the birds overlapping
- more obstacles
- bounce off sides of simulation
- add user input to change direction of flocks