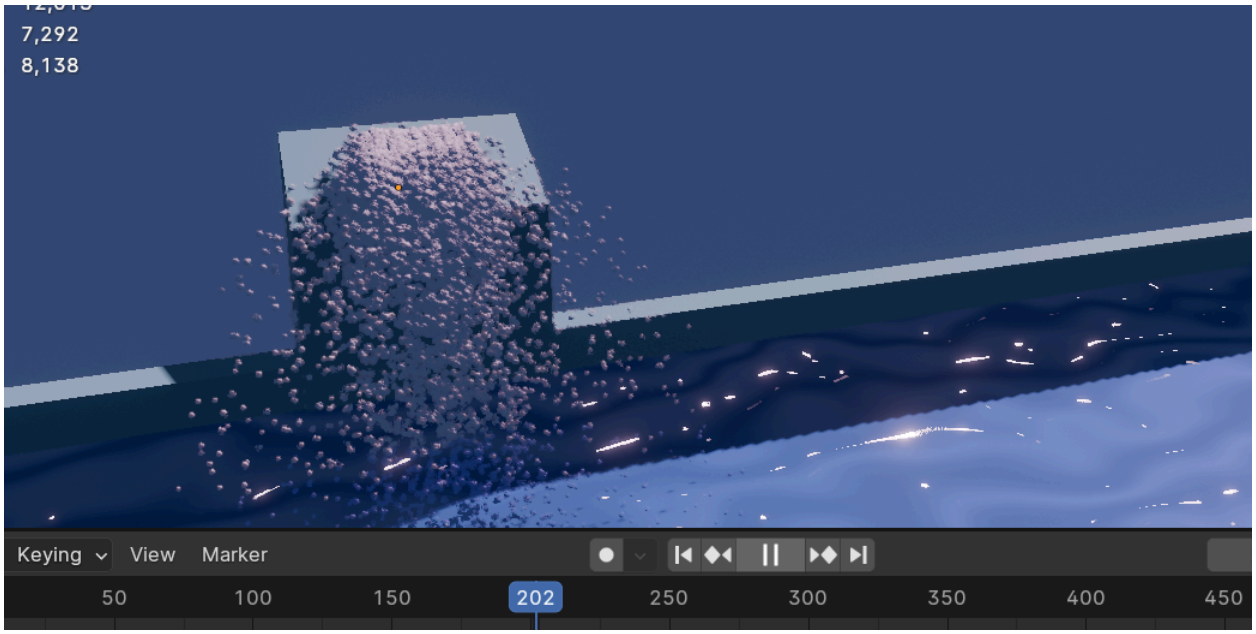
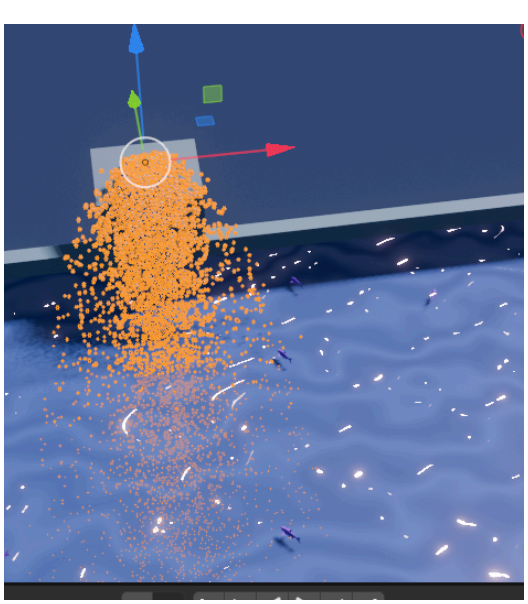
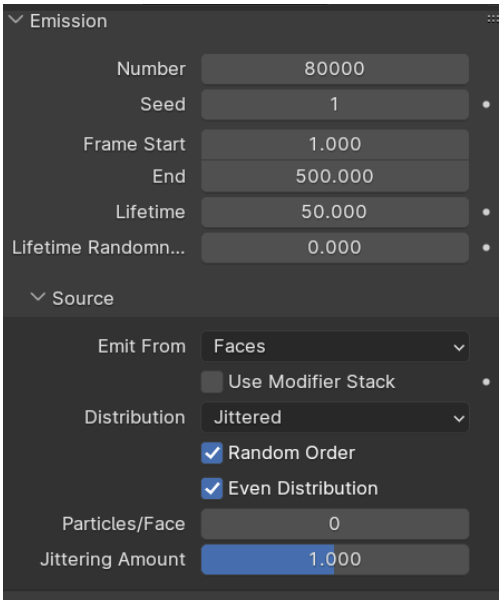
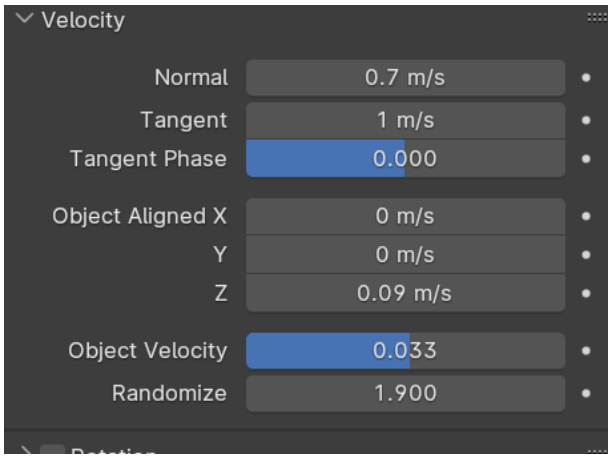
Generative methods, Group 5, Week 7

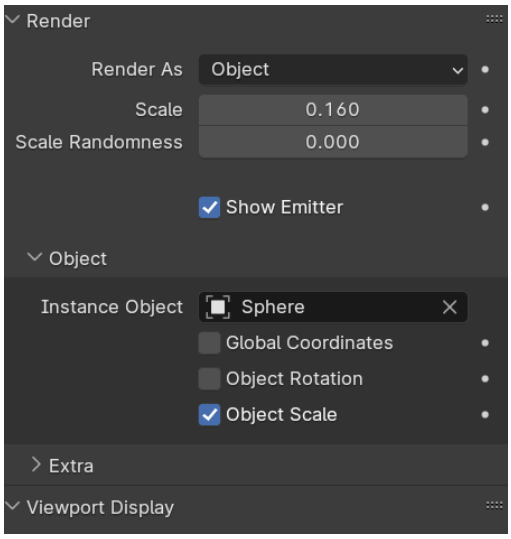
Results of part one:



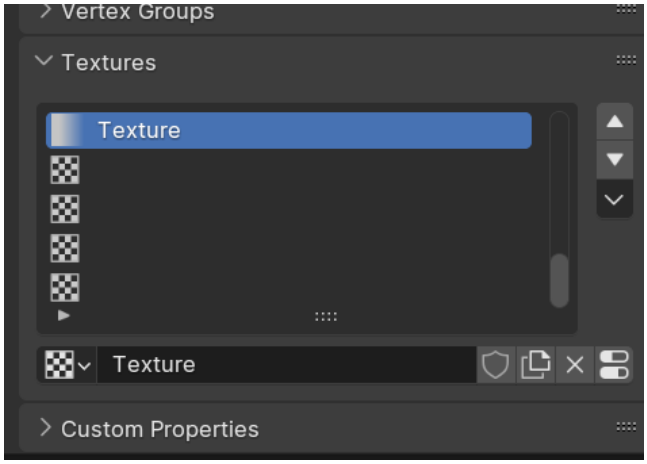


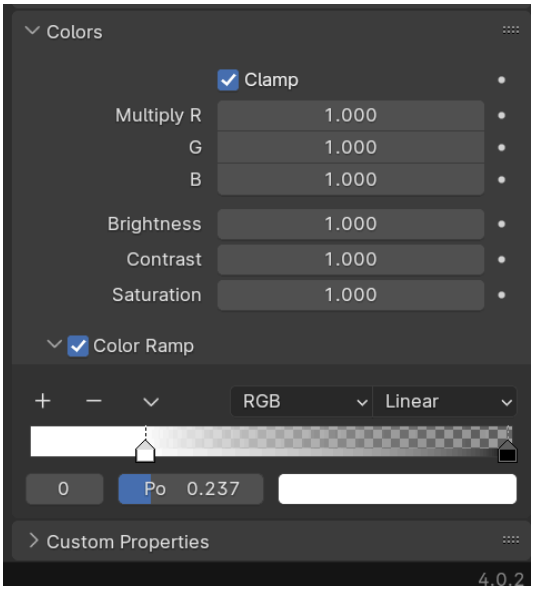


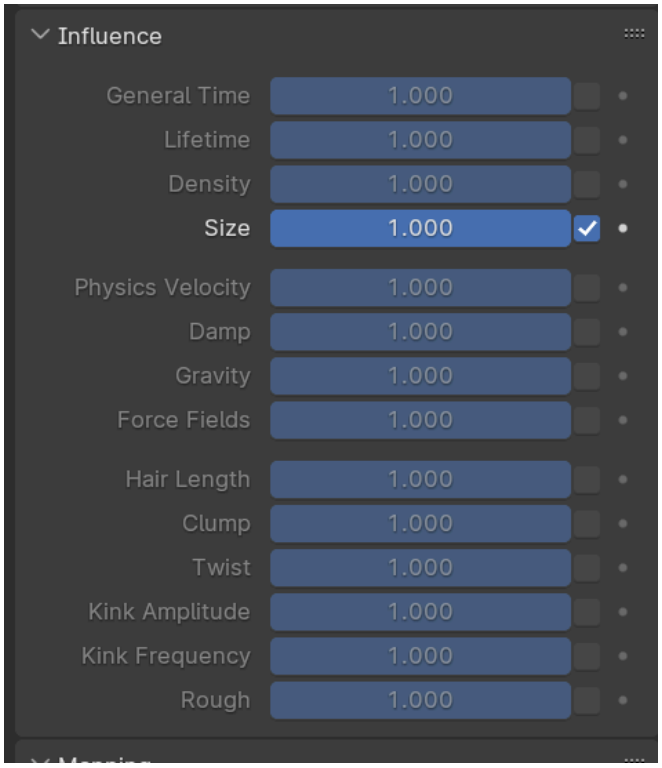




The particles, numbering 80,000, animate over 500 frames, drifting upward at 0.7 m/s while also moving tangentially at 1 m/s. Their motion is infused with randomness at a value of 1.9 to create an organic, varied flow.
We render them as a sphere, and gave the sphere light blue material which looks like water.







Then we add a texture to the particles, use this ramp color to influence their size, which can make them smaller when they are close to the ground.

Results of part 2:

IMPLEMENTATION SUMMARY:
For each fish, we check if it's close to any wall. If so, we apply a repulsion force to steer the fish away from the wall by adjusting the x and y components of the fish's velocity.
We then calculate the change in angle between the current velocity vector and the new velocity vector. This change in angle is applied to rotate the fish object accordingly so that they don't swim backwards.
For avoidance, we calculate a repulsion vector that ensures the fish steer away from nearby neighbors to avoid collision.
For alignment, for every neighboring fish we accumulate its velocity attenuated with a gaussian function that depends on the distance between the fish.
Alignment and avoidance vectors are then added to the fish's current velocity.

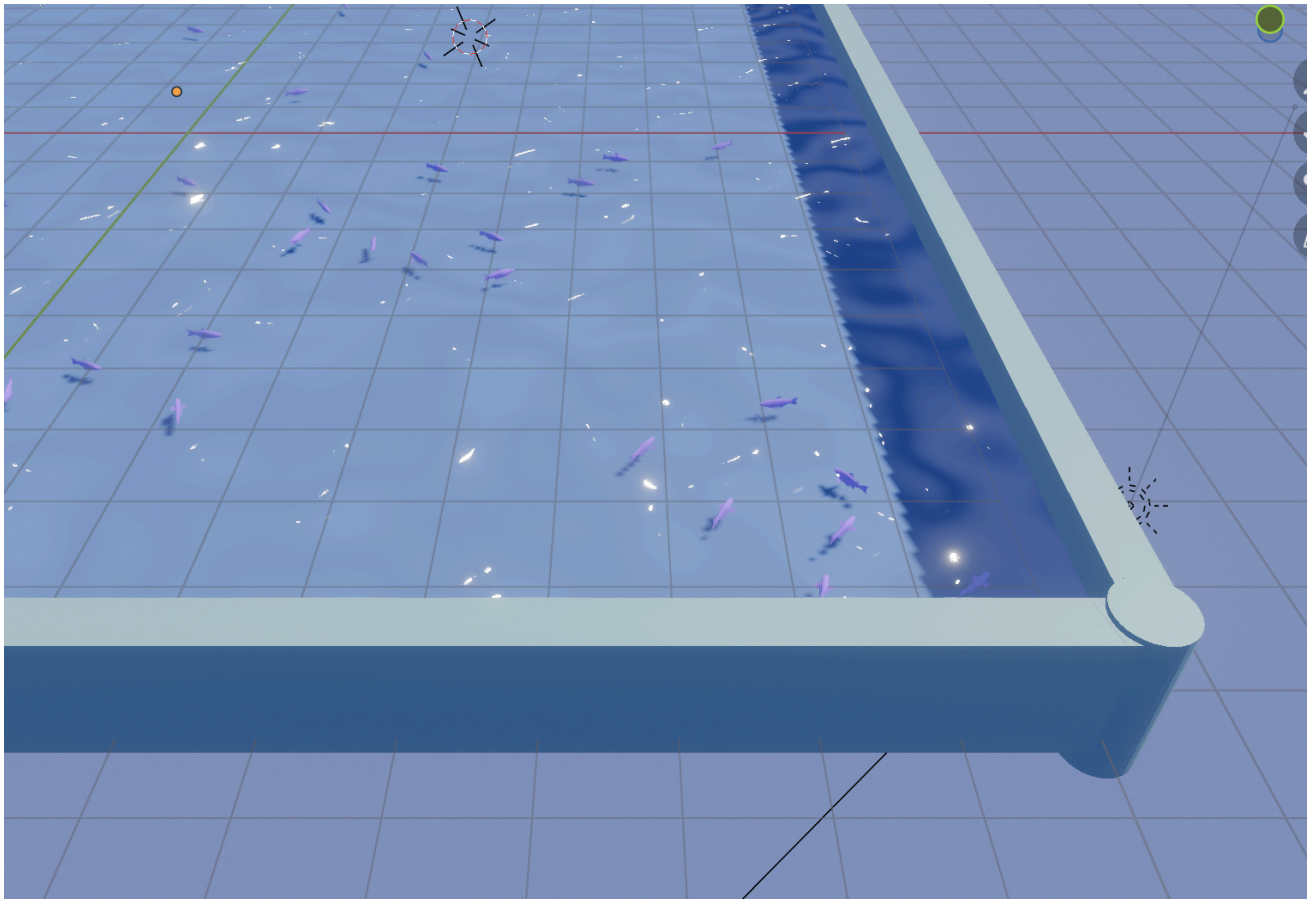
PARAMETERS for avoiding basin walls:
wall : determines the area of the square in which the fish are "allowed" to swim. If very small, then the flocking of the fish will concentrate in a small square in the middle of the pool.
wall_repulsion_strength : how quickly the fish change direction when approaching the wall.
Delta_angle: rotation angle when a fish has to change direction and turn away from a wall.

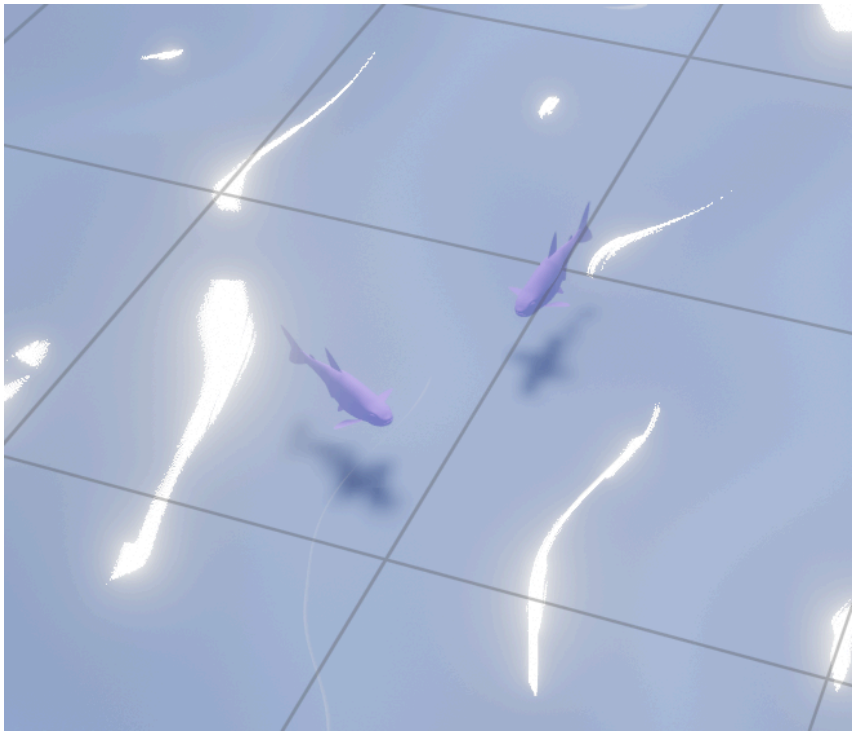
PARAMETERS for fish avoidance:
Repulsion_zone_width: describes the distance from a fish within which other fish are made to change direction.
repulsion_strength : describes how much fish want to avoid each other.

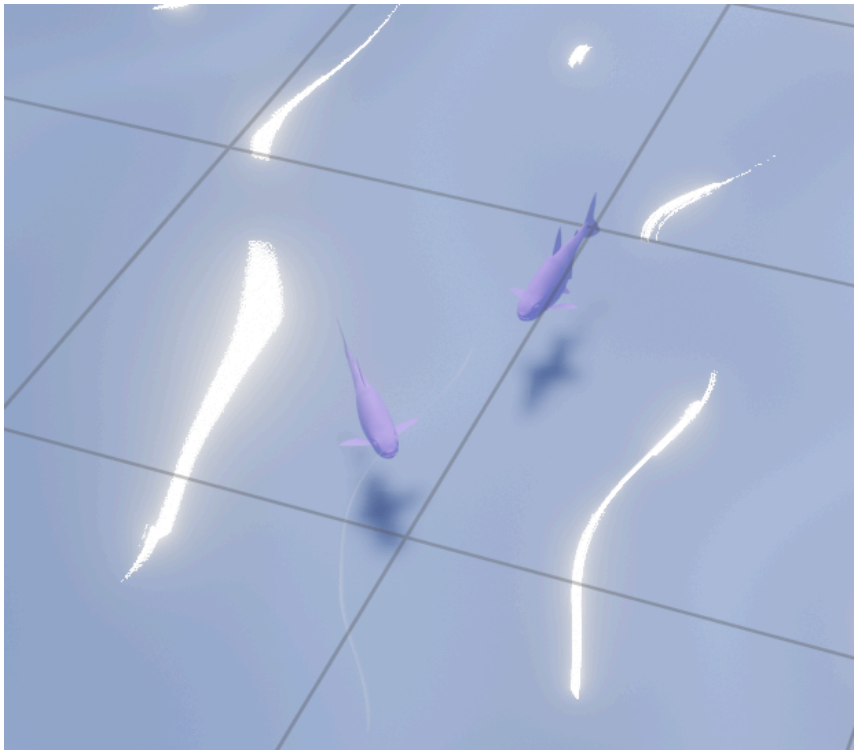
PARAMETERS for fish avoidance:
alignment_zone_width: describes the distance from a fish within which other fish are made to change their speed and influence each other.
Alignment_strength: describes how much fish will influence each other's speeds.

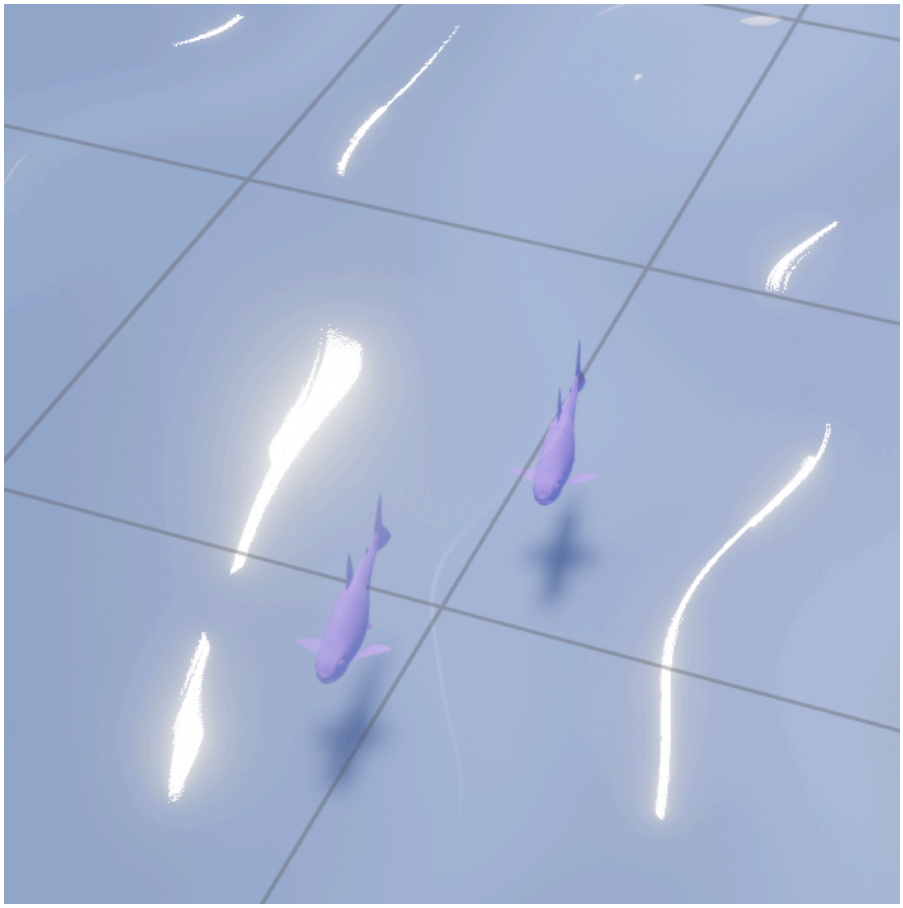
The first picture is to illustrate that fish stay within the pool and do not swim into the walls and out.

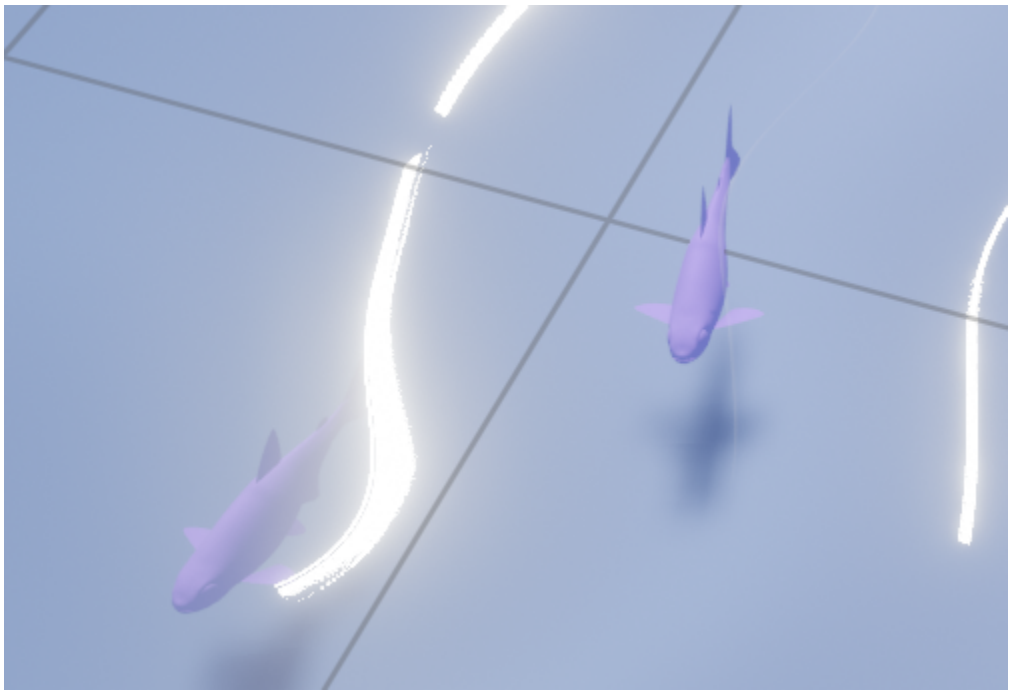
The other pictures try to capture the moment fish approach each other and then adjust direction to avoid each other.











CODE:

```
import bpy
from mathutils import Vector as vec
from math import exp, cos, sin, pi, atan2, fmod, acos
from random import uniform
import numpy as np

wall = 9.5
speed = 0.25
repulsion_strength=0.25
wall_repulsion_strength=0.5
alignment_strength=0.1
repulsion_zone_width=0.8
alignment_zone_width=1.0
keyframe_interval=10
frames=2000
std = 1

Fish = bpy.data.collections['Fish'].objects
velocities = [vec([0,0,0])] * len(Fish)

def angle_between_vectors_radians(v1, v2):
    dot_product = v1.dot(v2)
    magnitude_product = v1.length * v2.length
    dot_product = min(1.0, max(-1.0, dot_product / magnitude_product))
    cross_product = v1.cross(v2)
    if cross_product.z > 0 :
        angle_rad = acos(dot_product)
    else:
        angle_rad = -acos(dot_product)
    return angle_rad

for i,f in enumerate(Fish):
    ang = uniform(0, 2*pi)
    v = speed*vec((cos(ang), sin(ang), 0))
    velocities[i] = v
    f.location = vec((uniform(-wall,wall), uniform(-wall,wall), 0))
    f.keyframe_insert(data_path="location", frame=1)
    f.rotation_euler[2] = atan2(v.y, v.x) + pi
    f.keyframe_insert(data_path="rotation_euler", frame=1)

for frame_no in range(1,int(frames/keyframe_interval)):
```

```
    for i,f in enumerate(Fish):

        v = velocities[i].copy()
        p = f.location

        if p.x > wall:
            v.x -= wall_repulsion_strength
        if p.x < -wall:
            v.x += wall_repulsion_strength
        if p.y > wall:
            v.y -= wall_repulsion_strength
        if p.y < -wall:
            v.y += wall_repulsion_strength

        alignment_v = vec((0, 0, 0))
        repulsion_v = vec((0, 0, 0))
        for i_n, f_n in enumerate(Fish):
            if i!=i_n:
                dist = (f.location - f_n.location).length
                if dist < repulsion_zone_width:
                    repulsion_v += exp(-dist/std) * (f.location - f_n.location)/(f.location -
f_n.location).length * repulsion_strength

                if dist <= alignment_zone_width:
                  alignment_v += velocities[i_n] * exp(-dist/std) * alignment_strength

        v += alignment_v + repulsion_v
        v /= v.length
        v *= speed
        p += v

        delta_angle = angle_between_vectors_radians(velocities[i], v)
        f.rotation_euler[2] += delta_angle
        velocities[i] = v
        f.keyframe_insert(data_path="location", frame=frame_no*keyframe_interval)
        f.keyframe_insert(data_path="rotation_euler", frame=frame_no*keyframe_interval)
```