Implementing the predator fish

Project specification for the course, DD1354
Royal Institute of Technology KTH

Johannes Ljungberg || johlj@kth.se

1 Background

The very first lab in the course, DD1354, Models and Simulation, involved implementing a simulated flocking behavior, where modeled fish swam in an area according to a few coded behaviors. These were the separation behavior, where the fish avoid each other should they get too close, the alignment behavior that compels the fish to swim in the same direction as their neighbors, and the cohesion behavior that makes the fish swim in groups.

The lab also included ideas on how the project could be advanced further, one of which caught my eye as a possible course project. The idea was to implement a larger, predator fish, that chases the smaller ones while they try to escape if the predator comes too close.

1.1 Why this project?

Throughout the years I have worked extensively with Unity, although I never built a physics environment from scratch. This was why this first lab left such a big impression. So far I'd only produced movement using Unity's build-in Rigidbody component but had always felt a strong desire to look deeper into it and truly figure out how it worked. I just didn't know where to start. Then, in this course, one of the very first things I get to do is use raw, mathematical formulas to produce a seamless, remarkably realistic simulation that barely used any of Unity's built-in components.

Once I finally got all three behaviors to work properly, I felt a remarkable rush of satisfaction, and I'm fairly sure I giggled out loud like a bratty child.

In conclusion, I'm choosing this project because the "flocking behavior" simulation is the coolest damn thing I've ever implemented in Unity and I'd like to further advance my understanding of it.

2 Implementation

2.1 Predator

For implementing the predator fish it seems easiest to simply have it rotate towards the nearest fish, rather than using some mathematical formula to achieve the same. Once the predator is rotated correctly, one can use transform.right (or whichever axis points towards the nose of the

predator) to get a directional vector that determines where it should move during the next timeStep. This, because the value of transform.right is determined by the rotation of the Transform.

2.2 The rotation script

I have already created a script that rotates an object to point towards another object. This was implemented in 2D space, but changing to 3D should be as easy as adding the same code for the y-axis.

Here is the code:

First, we determine in what quadrant our rotatable object is in relation to the target it should rotate towards.

```
int WhatKvadrant(float xPos, float yPos, float targetXPos, float targetYPos) {
    if (targetXPos > xPos && targetYPos > yPos) return 1;
    else if (targetXPos < xPos && targetYPos > yPos) return 2;
    else if (targetXPos < xPos && targetYPos < yPos) return 3;
    else if (targetXPos > xPos && targetYPos < yPos) return 4;

    return 0;
}
```

Next, we calculate the angle between the two objects, using basic unit circle formulas.

```
float CalculateAngle(int kvadrant, float xPos, float yPos, float targetXPos, float targetYPos) {
    float yDistance = Mathf.Abs(targetYPos - yPos);
    float xDistance = Mathf.Abs(targetXPos - xPos);
    float angle = Mathf.Atan(yDistance/xDistance);
    angle = angle * Mathf.Rad2Deg;
    if (kvadrant == 2) angle = (90 - angle) + 90;
    else if (kvadrant == 3) angle = angle + 180;
    else if (kvadrant == 4) angle = (90 - angle) + 270;

    return angle;
}
```

It's also important to remember that the unit circle returns to zero once it passes 360 degrees. This must be taken into account before we rotate towards the target.

```
float angle = tran.eulerAngles.z;

if (angle > targetAngle + 180) angle = -(360 - angle);
else if (targetAngle > angle + 180) targetAngle = -(360 - targetAngle);
```

Note that the targetAngle parameter in the above code is the value that was returned in the CalculateAngle method, while the angle variable is simply the *current* rotation of the object.

Finally, we rotate the object, ensuring it stops once it points towards the target.

```
if (Mathf.Abs(targetAngle - angle) > rotationSpeed*3) {
    if (targetAngle > angle) tran.Rotate(Vector3.forward, rotationSpeed);
    else if (targetAngle < angle) tran.Rotate(Vector3.forward, -rotationSpeed);
}
```

This script is superior to other, built-in unity methods that handle look at rotation, such as transform.LookAt(), since you can't set its rotation speed. In comparison, the rotation speed of this script is determined by the rotationSpeed variable.

2.3 Prey

The mathematical formula for avoiding objects has already been implemented in the original lab, so all we need to do is re-use the code, but for the predator; increasing the separation force factor and radius.

Here is the formula used, taken from the instructions of lab 1.

$$\vec{F}_{separate} = k_{separate} \sum_{p_i \in Adj} \frac{r_{separete} - d}{d} (\vec{p_i} - \vec{p})$$

where $k_{separate}$ is the separation force factor, $r_{separate}$ is the radius of the neighbor area and d is the distance between $\vec{p}$ and $\vec{p_i}$, $d = |\vec{p_i} - \vec{p}|$

If I understand the formula correctly, this will compel the fish to swerve to the side to avoid the predator.

3 Risks

3.1 General help sessions

One potential risk is a lack of help this late in the course. There will only be three more general help sessions, one of which is too close to the lab assignment deadline to be used for questions about the project, rather than the lab. Therefore it is critical that I do not miss the two final help sessions, and use the time as effectively as possible.

To ensure this, I have added the dates of the general help sessions to my phone's calendar. Additionally, each time I run into a problem I will take note of it on a separate document, detailing what the problem is along with its priority level so that I can easily identify the most important questions during the help sessions.

3.2 The separation behavior

It's possible the code for the separation behavior described in section (2.3) won't be enough to create a realistic prey behavior. If I understand the formula in (2.3) properly, the separation behavior prevents collision if the two are already heading towards each other, which won't always be the case.

What may be needed is a formula that compels the fish to swim in the opposite direction of the predator. One should also add a level of randomization so that the fish tries to swerve to the side while heading away from the predator to make the movement look more realistic. I'm not sure how something like this would be implemented. I will consult a TA during the next Zoom help session.

4 Degree of simulation

I will attempt to build as much as possible from the ground up, taking inspiration from how the behaviors of the fish have already been implemented. This especially means never utilizing Unity's built-in Rigidbody component, nor its methods and properties.

5 Extensions

Here are a few additions to the project that might be added if I have the time.
1. Ensure the implementation can handle more than one predator, specifically so that predators can simply be copied and added without changing the behavior of the fish.
2. Making use of the spring formula from lab 2 to allow the user to click on a predator to add a rope of sorts that restricts the predator in an area for a limited time. An anchor will be created where the user clicked and the predator will gain an invisible, stretchable rope around it that prevents it from swimming too far from the anchor.

6 References

My blog detailing the project's progress.
(Link to blog will be added before the proper hand-in.)

A link to the instructions of the first lab.
https://docs.google.com/document/d/16T-NZEGOOFZ2VQIf5KFdhJfAyJdRG7uLkAsGEJ-DLbA/edit?usp=sharing

Questions for whoever reads through this (will not be included in the final hand-in)
1. What are the maximum number of allowed pages? I seem to remember it said 1-2 but the example project specification had 3 so I assume more than 2 is okay. I landed on 4. Is this too much?
2. Some of the formulas in the link to the lab instructions (see section 6) won't show up correctly. Is there a way to directly embed a pdf into a google doc so it can be downloaded by the reader? After some googling, I can't seem to find a way to do it.
3. I'm unsure what the references are supposed to be. Should they be used to reference things in *this* document? So if I mention the transform.LookAt() method, I should add a link that describes what it is, as a reference?