CS 480 Task 2: The Wonderful World of Dolphins

Description

This task serves as an extension to the basic behaviors investigated in Task 1. The single bird exhibited static behavior in that it did not interact with the world. It also was not pursing any goal with its fixed actions. This task considers basic interaction with respect to performing dynamic actions appropriate for a simple goal. It also introduces multiple agents.

The scenario is a dolphin playing with some fish by chasing them. Its goal is to meet up with the fish, then continue on to others indefinitely. It does not eat them or in any way modify the world. However, it does interact with the world in that it can see the fish and therefore decide how to swim toward them. The fish exhibit entirely random behavior in their swimming, basically as the birds did in Task 1, but they must avoid each other.

Requirements

You must satisfy the following requirements in a single Java program based on the framework provided in `CS480Task2Template`. Be sure to comment your code so it is clear where each part is handled. Use `cs480viewer-task2.jar` (or later) in whatever fashion worked for you in Task 1.

Section 1: Modeling

Assume all agents initially start in random positions with random yaw orientations. The degrees of freedom for fish are only yaw; the dolphin has both yaw and pitch.

Part 1

Implement a set of five basic fish, with identifiers `fish1` through `fish5`, based conceptually on Parts 3 and 5 of Task 1. Do not precompute the lines; instead, compute the next one once the fish reaches the end of the current one or has to change direction. The noise aspect from Part 4 is omitted here for simplicity, but you are welcome to implement it.

The fish swim along the surface of the water, so their *y* value will always be 0 (but the viewer will show the fish above the water for better appearance). If the fish reaches the edge of the world, randomly reorient it. It is not necessary to smooth this transition.

Part 2

Although five fish can hardly cause a traffic jam, you must implement some kind of collision detection and avoidance. It is not necessary to model vision in any way, so you may "cheat" by calculating mathematical proximity and an appropriate evasive maneuver. The maneuver need not be smooth (but do enforce proper yaw), and a fish that avoids another fish need not continue on its original course.

Part 3

The dolphin (with identifier `dolphin`) uses its eyes to detect fish. The horizontal field of view is 70 degrees in total (35 degrees to each side); there is no vertical field of view. The field extends out 100 meters. If the dolphin sees a fish, it will turn toward it and swim (in a straight line). The turn need not be smooth. The speed of the fish is your choice, but the dolphin should be 1.5 times as fast. It is acceptable to swim right through the fish upon arrival. The fish does not avoid the dolphin.

If the dolphin reaches the edge of the world, randomly reorient it. It is not necessary to smooth this transition.

Part 4

Part 3 expects straight-line movement, the way the fish swim. However, since the dolphin is really happy, it moves by leaping in and out of the water. Implement this movement in terms of a relatively smooth vertical arc. The horizontal and vertical distances are your choice, but keep them proportional to reflect dolphin-like leaps based on the scale we have.

While the dolphin is in the air, it cannot see any fish. Therefore, it cannot change direction until it reenters the water. Spend some random time in the water between leaps and vary the leap distance. It is acceptable to leap out of the world, provided that the dolphin randomly reorients itself and comes back. Again, it is not necessary to smooth this transition.

Section 2: Summary of Approach

There are relatively few well-defined solutions to AI problems, which means that your implementation can be anything you want. I will evaluate it primarily on its performance and your explanation of how it works.

For each part in Section 1, describe at minimum the following:

- what you did
- how well it worked
- what its pros and cons are
- what you might improve

This summary does not need to be extensive. A few sentences for each bullet should be adequate. Be honest: your summary must correspond to the actual performance of your model.

Submission

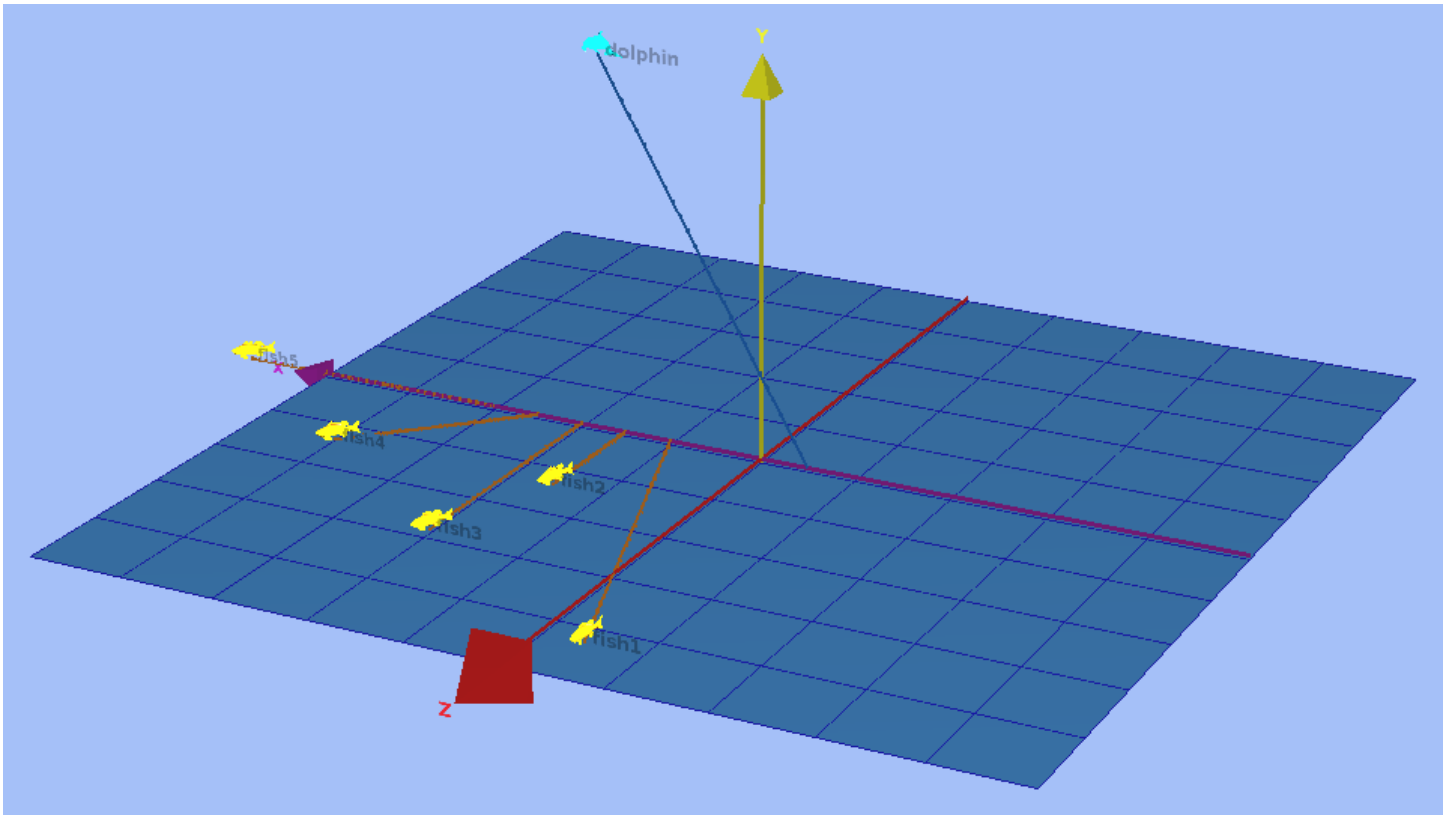
Submit your source file(s), a text readme file that explains how to compile and run your solution, and the summary document in PDF format through the link on the course web page.

For each part in Section 1, submit two representative track files of your output called `track2_n_m.trk`, where *n* is the part number (1 through 5) and *m* is the example number (1 or 2).

Grading

10 Section 1, Part 1
10 Section 1, Part 2
10 Section 1, Part 3
10 Section 1, Part 4
12 Section 2

`CS480Task2Template` generates the following output:



See `viewer.prop` in the jar file for details on the viewer keyboard commands.