Ship Collision Avoidance AI

Korben DiArchangel

December 21, 2022

1 Introduction

There are various rules that ships must abide by in order to get them to their destination safely. Roads may be created so that ships don't run into shallow water, and there are various rules in place on how to handle collisions. Some of the rules are as follows:

- If two ships are to run into each other directly, both ships should turn to starboard (turn right). This is called a Head-On situation.
- If two ships are to run into each other indirectly, the ship on the port (left) side of the other ship must turn to starboard (turn right). This is called a Crossing situation.
- If one ship is trying to pass another ship from behind, the ship trying to pass must turn to avoid the other ship, while the other ship must stay still. This is called an Overtaking situation.

It is important that ships follow these rules, as some of these rules rely on other ships not changing their course in order to avoid collisions.

I have created a simulation in Unity that will randomly generate several ships on a road. The goal of this project is to program AI for these ships so that they avoid colliding with other ships while still following the rules. This will make it useful for practice simulations that teach people what to do in certain situations.

2 AI

Artificial intelligence can be used to allow ships to navigate the roads and to avoid other ships in the simulated environment. Various methods of AI were used in this project, including A* and Potential Fields.

2.1 A*

The ship must be able to navigate a road that isn't a straight line. These roads will require the ship to turn at some point in order to avoid moving outside of the road or moving into the other lane.

This navigation can be done with the help of A*. A rectangular graph can be generated across the ocean, and adjacent nodes will be connected. Each path to a node can have length 1 (as all nodes are equidistant), and the heuristic can be determined by a taxicab distance (The x distance between the nodes plus the y distance between the nodes) to the goal point. This will lead to an A* algorithm that isn't computationally expensive, even with thousands of nodes; no distances need to be calculated, and calculating the value of each path only requires integer values.

The ships must be on the right side of the road in order to avoid potential collisions. This is done by having three separate graphs; two of them will not have any nodes on one of the lanes, and the other will have nodes on both of the lanes. If the ship wants to navigate across the road to the other side, it will choose the graph that omits the ship's "left" lane. This will also allow ships to make U-turns into the other lane if it wants to turn around.

There is also a "flatten" function after the A* path is made so that ship movement significantly smoother. It will remove unnecessary points that can be avoided by taking a direct path from one point to another. This will make the ship move in four straight lines instead of hundreds of horizontal and vertical movements.

2.2 Potential Fields

The ships should also be able to avoid any obstacles thrown at them while still moving closer to the goal. Potential fields allow the ship to better avoid obstacles by using a force-like equation to determine where the ship should be heading. These forces can be significantly stronger when the ship is closer to it, making it useful for making last-minute decisions. A positive potential field will be created at the next point the ship will have to travel to (points after it do not have a potential field). Positive potential fields are also created when a collision is detected; it will be placed in a position so that the ships that need to take action will turn appropriately:

- For Head-On situations, the potential field will be placed on the port-side (left) of each ship.
- For Crossing situations, a potential field for the ship that needs to turn will be placed behind the other ship's stern (back).
- For Overtaking situations, the overtaking ship will have a potential field created for it on the other ship's starboard (right).

These potential fields allow the ship to follow the rules when avoiding other ships.

2.3 Collision Detection

Each ship must be able to recognize when a collision is going to happen so that it can take appropriate action. It also must be able to recognize the type of collision, as the AI needs to place the potential field in the correct position.

The Closest Point of Approach (CPA), which is the point at which the two ships are closest to each other, is repeatedly calculated as the game runs. This point can be used to determine the distance between the two ships when they reach that point, as well as the time it will take to reach the point. This is calculated through the combination of relative velocity and trigonometry; a line is created based on the other ship's relative position and velocity from the ship, a perpendicular to that line is created from (0,0), and the point is found through the intersection of these lines. The distance between each ship is calculated from the distance from (0,0) to the CPA, and the time it takes is calculated through the relative velocity. If the distance and the time is low enough, the AI will detect a collision and will create the potential field to avoid it.

If the possibility of a collision is found, the AI will then figure out what type of situation is occurring. This is done by finding the difference of the angles of the heading of both ships; if the angle is around 180°, it will be identified as a Head-On situation, if the angle is around 0°, it will be identified as an Overtaking situation, and it will identify as a Crossing situation otherwise. It can also use this angle to determine which ship needs to turn to starboard in a Crossing situation. The speed difference of the two ships is used to determine which ship needs to move in the Overtaking situation.

3 Gameplay

There are two states to the gameplay; one consists of a bird's eye view that allows you to view the whole ocean and move each ship to a specified point, and the other consists of a "first person" view where you get to control an individual ship.

3.1 Controls

In the bird's eye view mode:

- **W/A/S/D** to pan the camera
- **R/F** to move the camera up and down
- **Z/X** to rotate the camera left and right
- **T/G** to rotate the camera up and down

- **Space** to move the camera faster
- **C** to create a ship
- **Q** to control the selected ship and enter the ship control mode
- **Left Click** to select a ship
- **Left Click + Shift** to select multiple ships
- **Right Click** to move selected ships to a target location
- **Right Click + Shift** to move the selected ships to multiple target locations

In the ship control mode:

- **Up/Down Arrow Keys** to speed up or slow down the ship
- **Left/Right Arrow Keys** to turn the ship
- **Q** to stop controlling the ship