ABM – Week 10 – Seminar – LVL2

Purpose

This task will allow you to understand the concepts of traffic modelling

Model

Open the model Traffic_TPOE_BASIC.nlogo. You will develop this model so that vehicles respect separation distances by accelerating and decelerating:

Take a couple of minutes to look at the code and take particular note of the following things:

- What are the global variables and which ones are actually used?
- How are cars generated and how is the inflow rate respected?
- How are cars removed from the system?

Step-By-Step Tasks

Rewrite the find-nh-ahead reporter

Notice that cars have a nh-ahead variable (neighbour ahead). This is intended to store the identity of the nearest turtle in front of them. Rewrite the find-nh-ahead reporter, so that it reports the turtle in front of the current turtle. If there are no turtles in front of the current turtle, the reporter should return 'nobody'.

Useful primitives: with, xcor, ifelse, any?, min, distance, …

Rewrite the set-target-speed reporter

This reporter sets a car's ideal speed, taking into account the maximum speed and the cars in front. It is in car context, so each car will run this reporter to determine their ideal speed.

Start by adding code that simply reports the maximum permissible speed if a car has no one in front of it.

If there *are* cars in front…

- Create temporary variables to store the speed of the car in front and their distance from the current car.
- Create a temporary variable to store the predicted distance to the car in front at the next tick (assuming that the current car did not move).

- Report a speed that would not allow the current car to get within 'separation-distance' of the car in front. Make sure this does not exceed the maximum speed.

Useful primitives: let, of, distance, min, …

Rewrite the change-speed procedure

This procedure uses the target-speed calculated at the previous step to alter the speed of the current car, while respecting the maximum acceleration and deceleration. Again, the procedure is in car context, so each car runs it to adjust their own speed.

- If the target-speed exceeds the current speed, increase the speed by the maximum acceleration, but do not exceed the target-speed or the maximum speed.
- If the target-speed is smaller than the current speed, decrease the speed by the maximum deceleration, but do not go below the target speed, or below zero.

Useful primitives: ifelse, min, max, …

Rewrite the fallen-tree procedure

This procedure is intended to place a tree at a random location on the road. Create a new breed to represent the tree and complete the procedure to achieve this goal. The tree should be a green circle. You will also need to create a tree variable called speed (always equal to zero), so that cars understand that the tree is a stationary obstacle.

Useful primitives: sprout–<breed> [Compare with the way that cars are generated], …

Try running 'fallen-tree' while the simulation is running to observe the effect.

EXTENSION

Your cars should now adjust their speed according to the traffic around them and a tailback should form when you add the fallen tree. However, you will notice that if you increase the maximum speed or reduce the maximum deceleration or the separation distance, cars will start to crash into the tree and one another. Come up with a more effective deceleration algorithm to avoid crashes, where a driver looks further ahead than a single tick and decelerates earlier.