F.A.S.T. Documentation – prepared by Georges M. Arnaout

Relevant Classes (brief description)

Car
Constants
IDM
IDMCar
IDMTruck
LaneChange
MicroStreet
MiscroSim2_0
Moveable
OnRamp
SimCanvas

Class Car

java.lang.Object
public class Car extends java.lang.Object implements Moveable
Car (double x, double v, int lane, MicroModel model, LaneChange lanechange, double length, java.awt.Color color, int number)
x: initial position of the vehicle generated
v: the speed of entry of the vehicle
lane: the lane (randomly decided out of 4) that the vehicle will be positioned on.
model: the type of vehicle to be generated (i.e. truck or car)
lanechange: The nature of the lane change being performed (cars are inconsiderate and trucks are polite)
length: The actual length of the vehicle (depending on its type)
color: the color of the vehicle (trucks are black, manual cars are red, and CACC cars are red)
number: is a unique id generated for each vehicle (or agent id)

Class Constants

public interface Constants
The constants influencing the global appearance and functionality of the model. In many cases, the model can be calibrated according to special needs (e.g. a different applet size, different time steps, different acceleration/deceleration, different bumper-to-bumper gap, etc) by simply changing the numbers and recompiling the program.

Class IDM
public abstract class IDM extends java.lang.Object, implements MicroModel
Basis class for the microscopic traffic model IDM (intelligent-driver model, see M. Treiber, A. Hennecke, and D. Helbing, Congested Traffic States in Empirical Observations and Microscopic Simulations, Phys. Rev. E 62, 1805 (2000)].

Class IDMCar
public class IDMCar extends IDM, implements MicroModel
This class contains all the cars parameters such as desired velocity *v0*, average acceleration *a*, average deceleration *b*, desired safety time headway *T*, minimum bumper-to-bumper gap allowed *s0*, and acceleration exponent *delta*.

Class IDMTruck
public class IDMTruck extends IDM, implements MicroModel
This class contains all the trucks parameters such as desired velocity *v0*, average acceleration *a*, average deceleration *b*, desired safety time headway *T*, minimum bumper-to-bumper gap allowed *s0*, and acceleration exponent *delta*.

Class LaneChange
public class LaneChange extends java.lang.Object, implements Constants
Implementation of the lane-changing model MOBIL ("Minimizing Overall Brakings Induced by Lane-changes"), see M. Treiber and D. Helbing, Realistische Mikrosimulation von Straßenverkehr mit einem einfachen Modell , 16. Symposium "Simulationstechnik ASIM 2002" Rostock, 10.09 - 13.09.2002, edited by Djamshid Tavangarian and Rolf Gr\"utzner pp. 514--520.

Class MicroStreet
public class MicroStreet extends java.lang.Object, implements Constants
This class is the core of the traffic simulator F.A.S.T.. The main functions defining the agents behavior in the four-lane freeway are described in this class. The main elements of MicroStreet are:

street, a vector of Moveable's representing the agents,

The update method invoked in every time step. It is responsible of updating the system throughout the simulation.

Methods for moving the vehicles (translate), accelerating them (accelerate) and performing the lane changes (changeLanes).

A sorting routine sort for re-arranging the vehicle order in street in the order of decreasing longitudinal positions

The method ioFlow responsible for the arriving agents into the system.

Class MiscroSim2_0

public class MicroSim2_0 extends java.applet.Applet implements Constants

This class is responsible in starting the model with the parameters chosen in the Constants class and in the user interface (sliders). Think of it as the main in our program.

Class Moveable

public interface Moveable

Moveable represents a general agent object from class Car with its parameters (position, velocity, lane, etc) In each time step, the objects are updated by moving them forward (method translate), by changing the velocity (method accelerate), etc

Class OnRamp

public class OnRamp extends MicroStreet

This class describes the length, behavior, positioning, and other related fields concerning the ramp merging into the freeway.

Class SimCanvas

public class SimCanvas , extends java.awt.Canvas , implements java.lang.Runnable, Constants

This class represent the graphical part of the model. All graphical functions related to drawing the vehicles, drawing the highway, and even relating the drawings with the simulation clock, are represented in this class.

Relevant Functions

accelerate(double, Moveable) - Method in class Car

- This function is responsible for the acceleration for the agents on the freeway.

change(Moveable, Moveable, Moveable) - Method in class Car

- This function is responsible for using the LaneChange class and performing necessary lane changes. It takes three variables, front vehicle on own lane, back vehicle on the new lane, and

the front vehicle on the new lane. Additionally, the LaneChange class is affected by a politeness factor.

density() – Method in class SimCanvas

- this function is responsible of collecting the density on the freeway. The approach used in this function is by creating two points separated by 1000 m (positioned before the ramp) then adding the number of vehicles positioned within the range of those two points.

showResults() – Method in class MicroStreet

- This function is used to collect various statistics about the model such as average speed, variance, standard deviation, etc.

ioFlow(double, double, double, int) - Method in class MicroStreet

- This function responsible for the arriving agents into the system. It is also responsible of regulating the arrival rate of the agents in case the demand exceeded the capacity and the queues became stretched out (i.e. avoid queue explosion).

sort() - Method in class MicroStreet

- This function is used to sorting or re-arranging the agents' order in street in the order of decreasing longitudinal positions.

stop() - Method in class SimCanvas

- This function is responsible of terminating the simulation.

start() – Method in class SimCanvas

- This function is responsible of initiating the simulation.

translate(double) - Method in class Car

- This function is responsible of moving agents on the freeway (changing their positions dynamically).

timeSpent() – Method in class SimCanvas

- This function is used to collect the average travel time of each and every agent in the system. The approach used in this function is by creating an entry point and an exit point on the freeway, and recording these times for each agent, then averaging the difference

update(double, int, double, double, double, double, double) - Method in class MicroStreet

- This function is in used to update the system throughout the simulation. In case throughout the simulation some parameters changed (such as the truck percentage, the arrival rate, etc), this functions is responsible of allowing the change to take effect into the simulation.