C2X EXAMPLE (COM)

Short description

Modelling car-to-car communication with support of an integrated Python script.

Requirements

- Vissim module: COM interface
- For Python scripting to be available, the following software needs to be installed on your computer:
  - PTV Vision Python (https://cgi.ptvgroup.com/php/vision-setups/)
    or a manual installation of Python including pyWin:
  - Python 2.7 (https://www.python.org/downloads/)
  - pyWin Build 218 or higher (http://sourceforge.net/projects/pywin32/files/)

Objective

This Car2X (C2X) example demonstrates how to model communication between vehicles. At simulation second 200, there is a breakdown of a vehicle. At the time of breakdown, the vehicle sends out a warning message. Vehicles receiving this message will drop their speed and adjust their driving behavior until they passed the incident.

To start the example, load CAR2X.INPX and run the simulation. The colors of the vehicles are based on the C2X status:

- white (status 0): no C2X equipment
- yellow (status 1): no active C2X message
- green (status 2): received C2X message
- red (status 3): sending C2X message.

Modelling technique

COM function ‚GetByLocation'

For modelling C2X in PTV Vissim, the COM function Vissim.Net.Vehicles.GetByLocation(posX, posY, distDistr) is provided. This function returns a VehicleCollection containing a subset of all vehicles inside a certain distance from the world coordinates (posX / posY). That distance is the maximum distance of the distance distribution distDistr which is passed as third parameter. A distance distribution is a new network object which specifies a probability depending on a distance, in this case the probability of the vehicle being *not* included in the collection. The probability is zero at distance zero (exactly at the passed world coordinates) and increases to 1 at the maximum distance of the distribution, with the detailed form of the distribution defined as curve in the distribution dialog. This method can be used to model lost Car2X messages sent from certain world coordinates, with the probability of a lost message increasing with the distance from that location.

The event-based script `Car2X Script.py` uses this function.

Incident modelling

The incident is modelled with a parking lot in combination with a parking routing decision. At simulation second 200, one vehicle of vehicle class 70 (vehicles equipped with C2X) is assigned to park at parking lot #1 with a parking duration of 200 s (= duration of incident). Due to the blocked lane #1, all vehicles have to use lane 2 and 3. In order to force them to switch to lane 2 and 3, a partial vehicle route (#2 of partial vehicle routing decision #1) is set which lead over the connector #10000 (lane 2 and land 3 only). During the incident, all vehicles do follow this partial route.

Adjusting speed & driving behavior

If a vehicle received a warning message from a location in driving direction, the vehicle will adjust its speed and driving behavior. The speed is set via COM in the event-based script `Car2X Script.py`. Before setting the new reduced speed, the current desired speed of the vehicle is stored in a user-defined attribute in order to set this desired speed after the vehicle has passed the incident location.

For adjusting the driving behavior, a separate driving behavior for vehicle class #80 (C2X active message) has been added to the link behavior type #2 of the main link. To activate this driving behavior, the vehicle type of all vehicles receiving the C2X message changes to vehicle type #102. Having this vehicle type, the vehicles adjust the driving behavior because this vehicle type is assigned to vehicle class #80. The change of the vehicle class is done within the event-based script `Car2X Script.py`. After a vehicle passed the incident location, the vehicle type of this vehicle is changed again to vehicle type #101. Hence the vehicle changes its driving behavior to the origin behavior.