Adding Different Vehicle Types to the Plexe Environment

Karthikeyan Deenadayalan

May 29, 2017

1 Introduction

This document explains you to understand how vehicles of different type are added to plexe-viens and it clearly states, what modifications were done to the existing files. For this, a good understanding of how vehicles of same type are added to the platoon would be helpful(i.e PlatoonsTrafficManager class). This document demonstrates how a platoon of 8 vehicles, of three different types are injected in a highway scenario. It is also to be made sure that you are working in developement branch of the plexe for both sumo and viens.

2 Modifications in Plexe-Veins/examples folder

Under this folder `platooning/sumocfg/freeway.rou.xml` file is modified to add new vehicle type. As we can see a vehicle is defined by vType-tag. We add a new vehicle by, adding the following lines of code to the XML file.

```
<vType id="vtypeheter" accel="2.5" decel="6.0" sigma="0.5"
    width="2.3" height = "2.5" vClass="trailer" length="5"
    minGap="0" maxSpeed="41" color="1,0,1" probability="1">
  <carFollowing-CC tauEngine="0.5" omegaN="0.2" xi="1"
    c1="0.5" decel="9" lanesCount="4" ccAccel="1.5"/>
</vType>
```

```
<vType id="vtypebus" accel="2.5" decel="6.0" sigma="0.5"
    width="2.3" height = "2.5" vClass="bus" length="6"
    minGap="0" maxSpeed="41" color="0,1,0" probability="1">
  <carFollowing-CC tauEngine="0.5" omegaN="0.2" xi="1"
    c1="0.5" decel="9" lanesCount="4" ccAccel="1.5"/>
</vType>
```

So now we have added vehicle type truck named **vtypeheter** which is our second type of vehicle and vehicle type bus named **vtypebus** which is our third type of vehicle. Similarly we can modify this file to add any number of vehicles.

3 Modifications in Plexe-Veins/src folder

The source files are located in the `plexe-veins/src/veins/modules/application/platooning`. I have created a new class called Heterogenous, which will inject the vehicle types specified in the ***freeway.rou.xml***.

3.1 SimpleHeterogenous Class file

Under this source file we have added new function to insert the vehicles in the platoon:

```
void Heterogenous::insertHeterogeneousVehicles(){
/*Modified Part*/
        switch(vehicleInOrder.at(i)){
        case 1:
            automated.position = currentPos + laneOffset[l];
            automated.lane = l;
            addVehicleToQueue(0, automated);
            currentPos -= (carLength + distance);
            break;
        case 2:
            truck.position = currentPos + laneOffset[l];
            truck.lane = l;
            addVehicleToQueue(0, truck);
            currentPos -= (truckLength + distance);
            break;
        case 3:
            bus.position = currentPos + laneOffset[l];
            bus.lane = l;
            addVehicleToQueue(0, bus);
            currentPos -= (busLength + distance);
            break;
    }
   currentCar++;
}
```

Here the **vehicleInOrder** is a integer vector containing the order of vehicleTypes to be inserted into the platoon(i.e 1 corresponds to vtypeauto, 2 corresponds to vtypeheter, 3 corresponds to vtypebus) this integer vector can be changed in the initialization file according to the user's need. So which ever the number is the case, that corresponding Vehicle type is added to the queue, which is then added to the platoon.

There is an another function which helps us to convert the String of vehicleInOrder variable that we have specified in the initialization file is converted into a integer and placed in the vector.

```
void Heterogenous::convertStringToVector(String param)
```

3.2 NED file for the SimpleHeterogenous class

We should not forget to create a Heterogenous.ned file. Here all the variables are added which are declared in the Heterogenous class.

3.3 Modification in Omnet.ini file

This file can be located in `plexe-veins/examples/platooning`. The variable that are added to this file include (nTrucks, nBus, vehicleTypeInPlatoon, platooningVTypeTruck, platooningVTypeBus). Important thing to notice is to change the traffic type to Heterogenous.

3.4 Modification to the PostionHelper class

This modification can done either in the PositionHelper class or by creating a new class. Since we are interested only in finding the ID of the current vehicle we are adding a function to the class.

```
static int getIdFromExternalId2(std::string externalId){
   if(externalId.substr(5,4).compare("auto") == 0){
        int dotIndex = externalId.find_last_of('.');
        std::string strId = externalId.substr(dotIndex + 1);
        int num = strtol(strId.c_str(), 0, 10);
        int count=0;
        int index=0;
        std::vector<int>::iterator it = vehTypeInOrder.begin();
        while(it != vehTypeInOrder.end()){
            if(*it == 1){
                if(count == num){
                    return index;
                }
                count++;
            }
            index++;
            it++;
        }
    }
    else if(externalId.substr(5,3).compare("bus") == 0){
        int dotIndex = externalId.find_last_of('.');
        std::string strId = externalId.substr(dotIndex + 1);
        int num = strtol(strId.c_str(), 0, 10);
```

```
        int count=0;
        int index=0;
        std::vector<int>::iterator it = vehTypeInOrder.begin();
        while(it != vehTypeInOrder.end()){
            if(*it == 3){
                if(count == num){
                    return index;
                }
                count++;
            }
            index++;
            it++;
        }
    }
    else if(externalId.substr(5,5).compare("heter") == 0){
        int dotIndex = externalId.find_last_of('.');
        std::string strId = externalId.substr(dotIndex + 1);
        int num = strtol(strId.c_str(), 0, 10);
        int count=0;
        int index=0;
        std::vector<int>::iterator it = vehTypeInOrder.begin();
        while(it != vehTypeInOrder.end()){
            if(*it == 2){
                if(count == num){
                    return index;
                }
                count++;
            }
            index++;
            it++;
        }
    }
    return -1;
}
```

4 Modification in the Mobility/Traci Folder

Since we are interested in finding the width and height of the current vehicle for designing the channel model, we have made changes to the TraciCommandInterface class and added two getter functions. These function retrieve the width and height respectively.

```
double TraCICommandInterface::Vehicle::getVehicleHeight()
double TraCICommandInterface::Vehicle::getVehicleWidth()
```

NOTE: It is to be clearly noted that the height method works only for Sumo

version 0.28.0 or higher.