Writeup for Path Planning Project

Rubric points:

√ The code compiles correctly.
√ The car is able to drive at least 4.32 miles without incident.
√ The car drives according to the speed limit.
√ Max Acceleration and Jerk are not Exceeded.
√ Car does not have collisions.
√ The car stays in its lane, except for the time between changing lanes.
√ The car is able to change lanes.

Reflection:

I added 3 functions to the code to implement path panning for the car:

1-calculateNextXYValues

This function basically encapsulates classroom project walkthrough.
The input parameters to this function are: desired velocity, desired lane, localization data, previous path data from simulator and car's current coordinates.
This function calculates the next_x_vals and next_y_vals which will be sent to the simulator as trajectory in main function.

It creates a vector of 5 points, 2 points from current car position or the previous path and 3 points with 30 meters distance starting from car's current position . To make calculations simpler, we convert the vector point coordinates to car coordinates and then use spline library to calculate the trajectory curve passing these 5 points.

It populates next_x_vals and next_y_vals with previous path points which car hasn't gone through them yet and calculates rest of the points to satisfy the desired velocity. Meaning, every 0.02 seconds, it uses remaining points from previous path and adds few new points to the end of the points.

In summary, this function gets the desired velocity and lane calculated by **calculateLaneAndSpeed** function and generates the trajectory to get to desired velocity and lane.

2-calculateLaneAndSpeed

This function calculates the next desired lane and speed for the function **calculateNextXYValues**, based on sensor fusion data.
The input parameters are car position, sensor fusion data and previous path and it calculates lane and velocity. it uses **getLaneCarsData** function to get distance from cars in back and front in each line. Based on the distance from closest cars in current lane, it decides to stay on the lane or change to another lane. Fro change lane it decides if it is safe based on the closest cars on the desired target lane. Also this function determines the desired velocity. If the car gets too close to front car, it decrease the velocity and otherwise it increases the velocity not more than the speed limit and also not causing acceleration and jerk change more than desired values.

3-getLaneCarsData

This function gets the sensor fusion data and the lane and finds the closest cars on the lane from the self driving car. It iterates on all cars from sensor fusion data and compares the Frenet coordinate to determine which cars are on the lane in question. It predicts the location of each car on the lane and determine the closest front and back cars on the lane by comparing the coordinates. The closest distances calculated by this function will be used in **calculateLaneAndSpeed** function.

In main function, every time the message from simulator is received, **calculateLaneAndSpeed** is called which calculates desired lane and velocity and then **calculateNextXYValues** is called which calculates the trajectory populating the next_x_vals and next_y_vals variables which will be fed to the simulator.