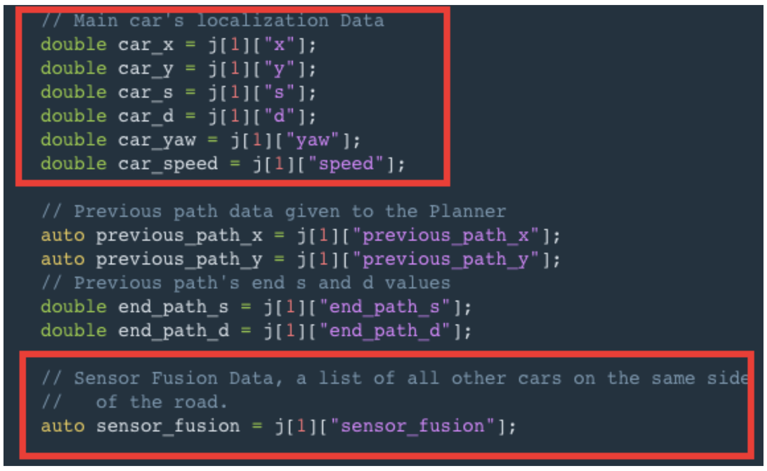
WRITEUP HIGHWAY DRIVING

FLOWCHART

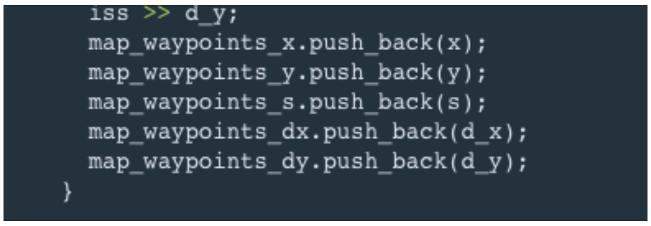
START

SENSOR FUSION AND LOCALISATION DATA
FROM SIMULATOR

MAP DATA



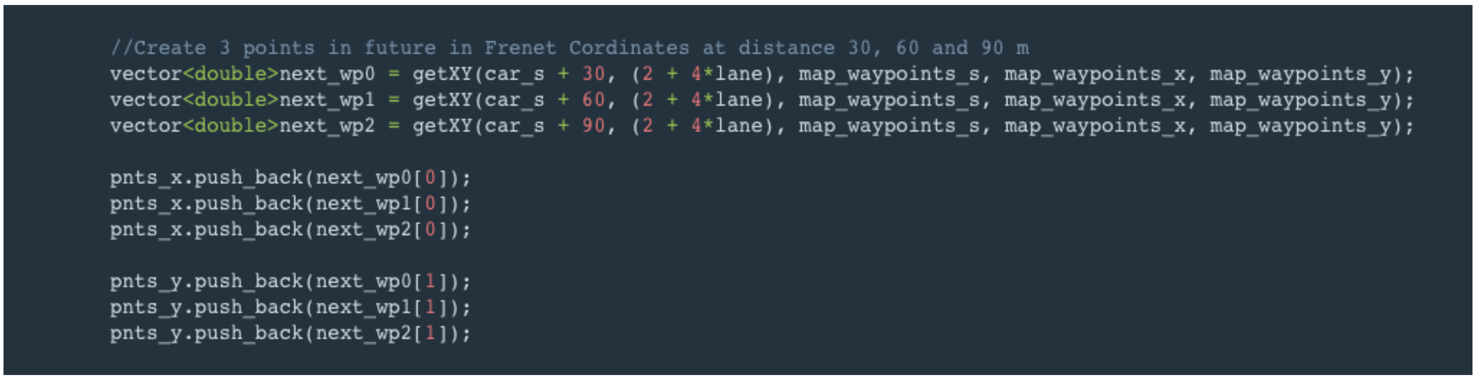
```
// Main car's localization Data
double car_x = j[1]["x"];
double car_y = j[1]["y"];
double car_s = j[1]["s"];
double car_d = j[1]["d"];
double car_yaw = j[1]["yaw"];
double car_speed = j[1]["speed"];

// Previous path data given to the Planner
auto previous_path_x = j[1]["previous_path_x"];
auto previous_path_y = j[1]["previous_path_y"];
// Previous path's end s and d values
double end_path_s = j[1]["end_path_s"];
double end_path_d = j[1]["end_path_d"];

// Sensor Fusion Data, a list of all other cars on the same side
//   of the road.
auto sensor_fusion = j[1]["sensor_fusion"];
```



```
      iss >> d_y;
      map_waypoints_x.push_back(x);
      map_waypoints_y.push_back(y);
      map_waypoints_s.push_back(s);
      map_waypoints_dx.push_back(d_x);
      map_waypoints_dy.push_back(d_y);
    }
```

Data is available form simulator as JSON in every 20ms updates

PREDICTION AND TRAJECTORY PLANNING



```
//Create 3 points in future in Frenet Cordinates at distance 30, 60 and 90 m
vector<double>next_wp0 = getXY(car_s + 30, (2 + 4*lane), map_waypoints_s, map_waypoints_x, map_waypoints_y);
vector<double>next_wp1 = getXY(car_s + 60, (2 + 4*lane), map_waypoints_s, map_waypoints_x, map_waypoints_y);
vector<double>next_wp2 = getXY(car_s + 90, (2 + 4*lane), map_waypoints_s, map_waypoints_x, map_waypoints_y);

pnts_x.push_back(next_wp0[0]);
pnts_x.push_back(next_wp1[0]);
pnts_x.push_back(next_wp2[0]);

pnts_y.push_back(next_wp0[1]);
pnts_y.push_back(next_wp1[1]);
pnts_y.push_back(next_wp2[1]);
```

As the case of Highway driving Prediction is done using Frenet Coordinates. Way points at S+30, +60 and +90 are chosen. Trajectory involves a set of 50 points.

Inorder to smoothen the vehicle transition on corners and lane changes spline function is used for interpolation. The spacing between the coordinates is chosen so that Jerk is with 1G and Tangential jerk is also minimum.

WHILE THE TRAJECTORY CALCULATIONS ARE INITIALLY IN FRENET COORDINATES IT IS PASSED AFTER CONVERTING TO CARTESIAN COORDINATES BEFORE PASSING TO SIMULATOR

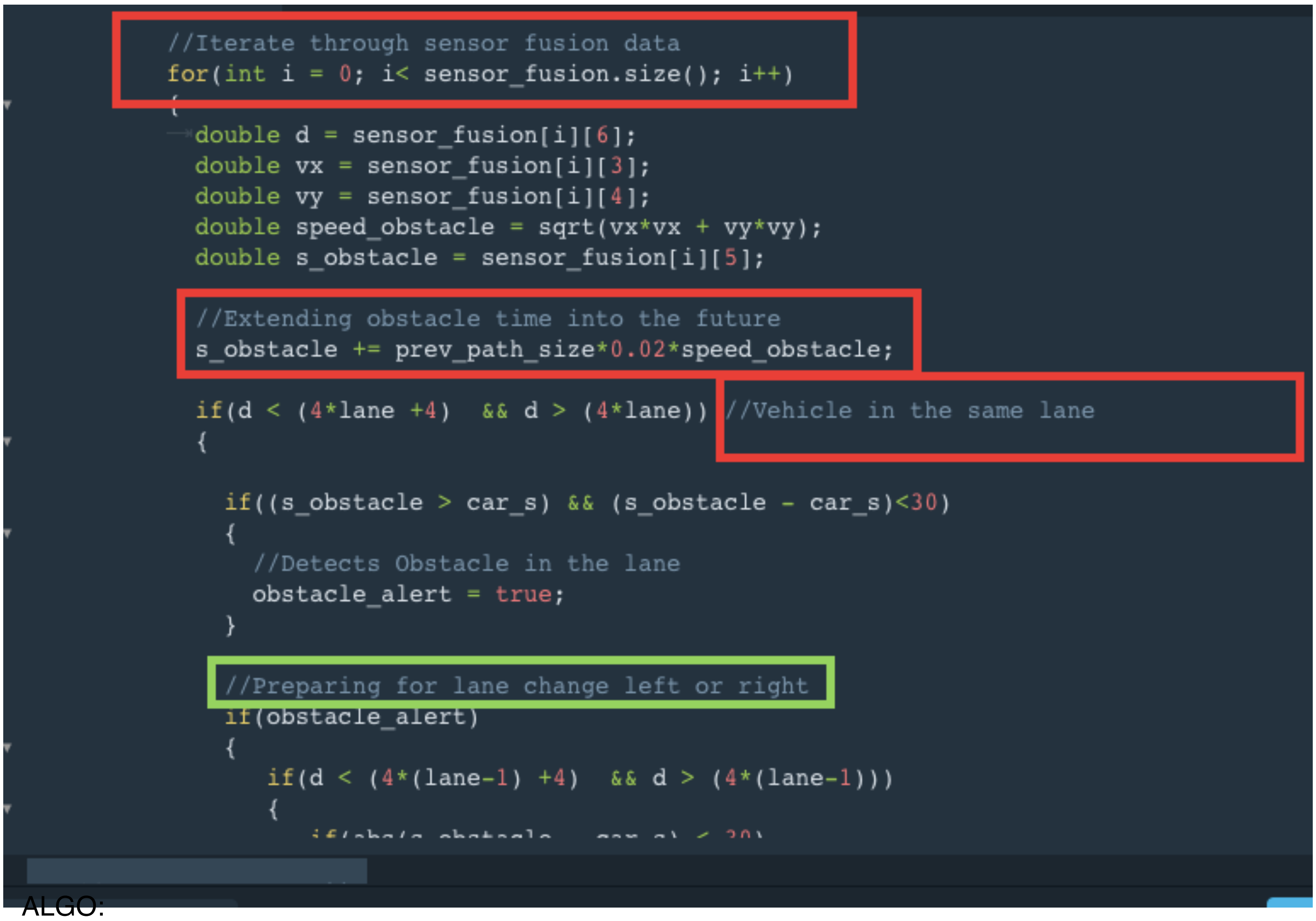
BEHAVIOUR PLANNING

1. ACCELERATION AND DECELERATION

1.1 Vehicle has to accelerate and keep speed limit 49.5 (<50)
1.2 Incase of vehicle on safe lane need to decelerate when S distance becomes less than 30. To avoid collision.

2 LANE CHANGING

2.1 Incase of slower cars on same lane. Vehicle has to consider a lane change.
2.2 Lane change has to be done safely by checking for any incoming traffic or vehicles ahead on the intended lanes.



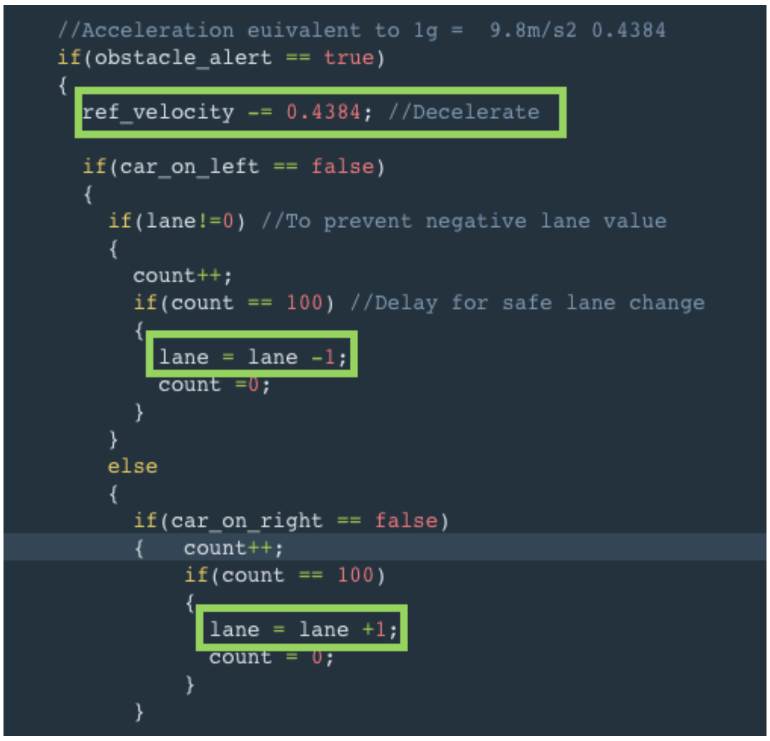
```
//Iterate through sensor fusion data
for(int i = 0; i< sensor_fusion.size(); i++)
{
  double d = sensor_fusion[i][6];
  double vx = sensor_fusion[i][3];
  double vy = sensor_fusion[i][4];
  double speed_obstacle = sqrt(vx*vx + vy*vy);
  double s_obstacle = sensor_fusion[i][5];

  //Extending obstacle time into the future
  s_obstacle += prev_path_size*0.02*speed_obstacle;

  if(d < (4*lane +4)  && d > (4*lane)) //Vehicle in the same lane
  {

    if((s_obstacle > car_s) && (s_obstacle - car_s)<30)
    {
      //Detects Obstacle in the lane
      obstacle_alert = true;
    }

    //Preparing for lane change left or right
    if(obstacle_alert)
    {
      if(d < (4*(lane-1) +4)  && d > (4*(lane-1)))
      {
```

ALGO:

1. ITERATE THROUGH THE SENSOR FUSION.
2. IDENTIFY VEHICLE LANE USING FRENET D COORDINATE
3. IDENTIFY SAFE DISTANCE BY USING FRENET S COORDINATE (USING CAR VELOCITY TO EXTRAPOLATE INTO THE FUTURE PATH)
4. SET CORRESPONDING FLAGS FOR OBSTACLE DETECTED, OR LANE CHANGE RIGHT OR LEFT

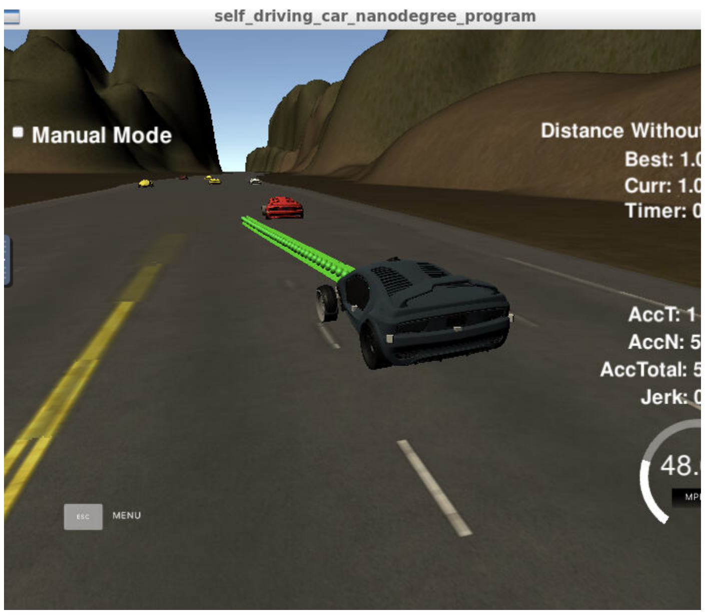
MOTION CONTROL

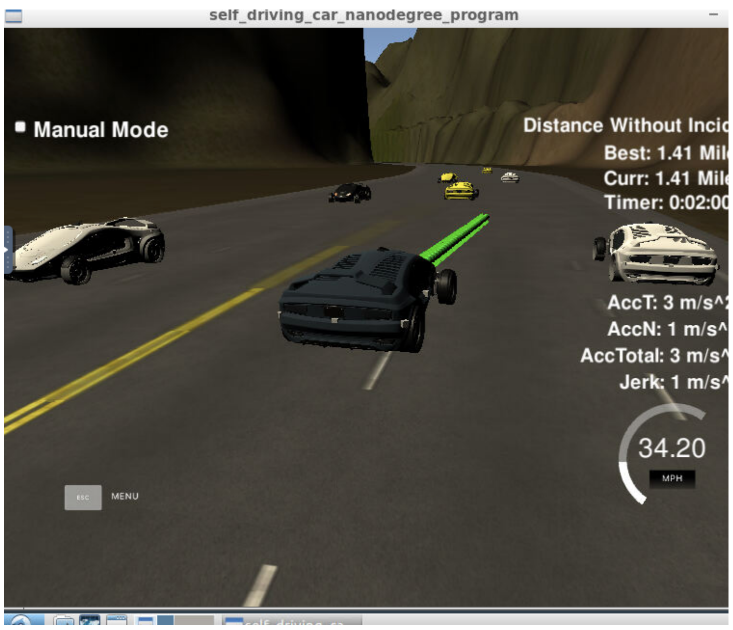
Change acceleration and deceleration, and lane changes based on the flags set by behavioural planning. A count is introduced to provide delay incase of safe lane changing. As vehicle might change left and right lane together in very close instance.



```
//Acceleration euivalent to 1g =  9.8m/s2 0.4384
if(obstacle_alert == true)
{
 ref_velocity -= 0.4384; //Decelerate

  if(car_on_left == false)
  {
    if(lane!=0) //To prevent negative lane value
    {
      count++;
      if(count == 100) //Delay for safe lane change
      {
        lane = lane -1;
        count =0;
      }
    }
    else
    {
      if(car_on_right == false)
      {   count++;
          if(count == 100)
          {
            lane = lane +1;
            count = 0;
          }
      }
```

LEFT AND RIGHT LANE CHANGES





CONCLUSION: Objectives of Rubrics are met.