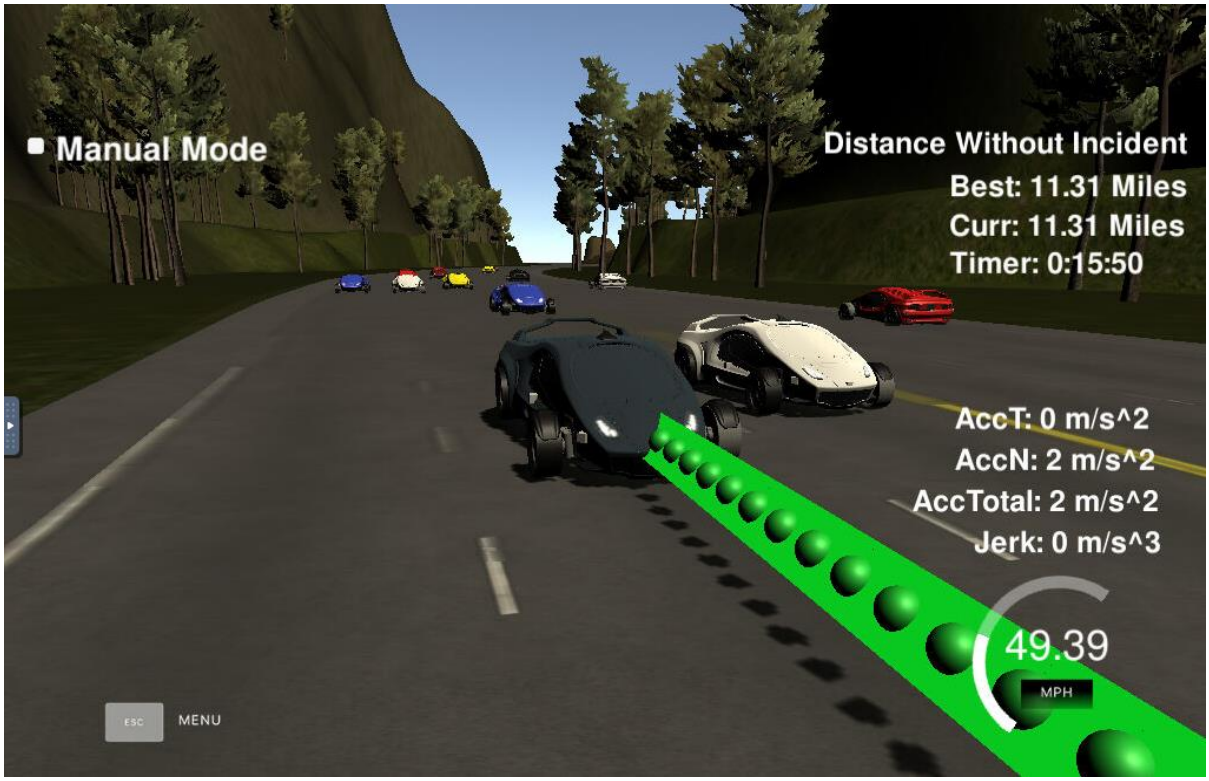
Path planning project

Overview:

In this project we need to implemented path planning algorithm to drive a car on highway on a simulator provided by Udacity. The simulator sends cars localization details (x, y, s, d, yaw, speed) and sensor fusion information (d, s, velocity etc) about rest of the cars on the highway. We need to pass a set of points spaced in time at 0.02 seconds representing the cars trajectory. The car should try to go as close as possible to the 50 MPH speed limit, which means passing slower traffic when possible, note that other cars will try to change lanes too. The car should always avoid hitting other cars at all cost as well as driving inside of the marked road lanes, unless going from one lane to another. The car should be able to make one complete loop around the 6946m highway. Since the car is trying to go 50 MPH, it should take a little over 5 minutes to complete 1 loop. Also the car should not experience total acceleration over 10 m/s^2 and jerk that is greater than 10 m/s^3.
The communication between the simulator and project is achieved using uWebSocket.



Build steps:

In order to build and compile the code, from within the main repository directory:

- `mkdir build && cd build` to create and enter the build directory
- `cmake .. && make` to compile your project
- `./path_planning` to run the code

Click on the "Simulator" button in the bottom of the Udacity workspace, which will open a new virtual desktop. You should see a "Simulator" icon on the virtual desktop. Double-click the "Simulator" icon in that desktop to start the simulator.

Important: You need to open a terminal before attempting to run the simulator and keep the GPU mode on.

Rubric points:

Compilation:

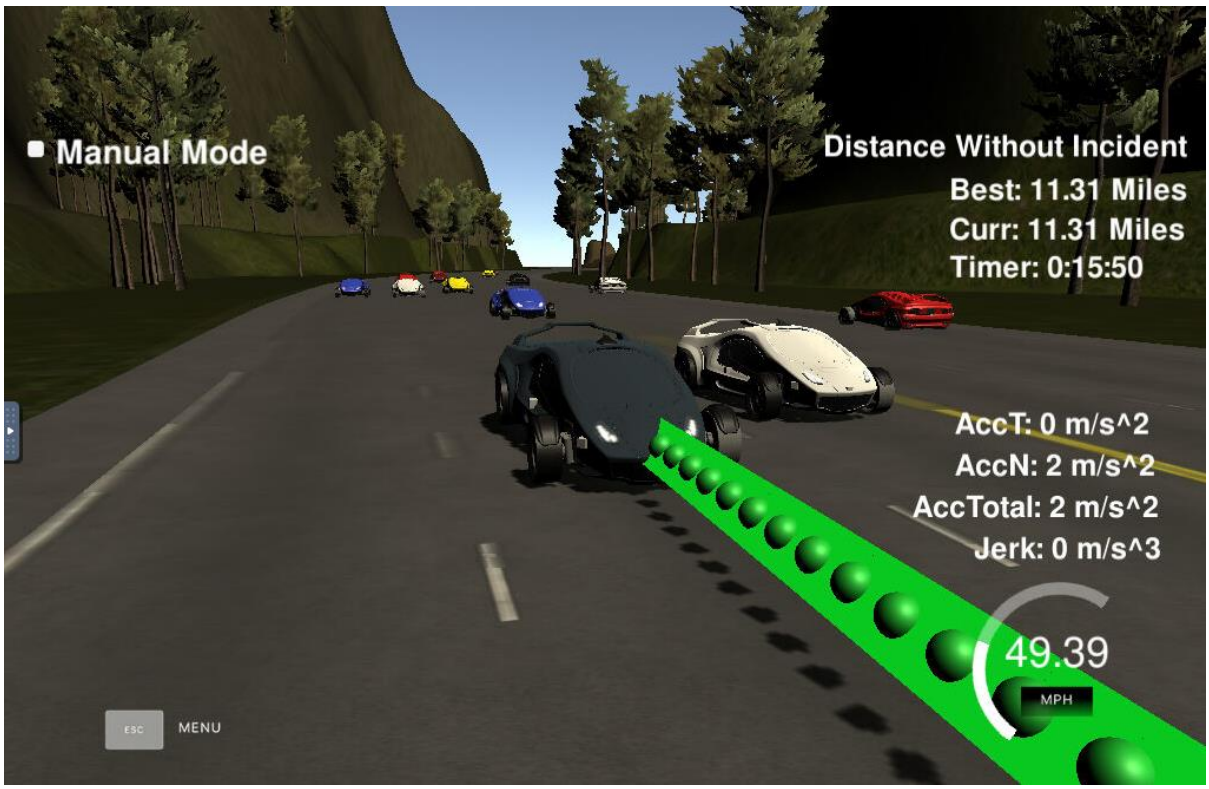
Code compiles correctly. A new flag *set(CMAKE_BUILD_TYPE RelWithDebInfo)* is added to CMakesLists.txt to make spline.h anonymous namespace become referenceable. Otherwise the path planning project will generate below error:

Path-planning: src/spline.h:288: void {anonymous}::tk::spline::set_points(const std::vector<double>&, const std::vector<double>&, bool): Assertion `x.size()==y.size()' failed.
Aborted (core dumped)

Valid trajectories:

The car is able to drive at least 4.32 miles without incident:

I ran the simulator for ~15 mins without any incidents like exceeding acceleration/jerk/speed, collision, and driving outside of the lane lines.



The car drives according to the speed limit:

Car doesn't drive more than expected speed limit 50 MPH.

Max Acceleration and Jerk are not Exceeded:

The car does not exceed a total acceleration of 10 m/s^2 and a jerk of 10 m/s^3.

Car does not have collisions:

Car doesn't collide with other cars on the road.

The car stays in its lane, except for the time between changing lanes:

The car stays in its lane most of the time but when it changes lane because of traffic or to return to the center lane.

The car is able to change lanes :

The car change lanes when the there is a slow car in front of it, and it is safe to change lanes (no other cars around) or when it is safe to return the center lane.

Code Reflection:

Base code provided by Udacity is extended by reading sensor fusion information to the know details of other cars on the neighbor lanes and to decide to either keep lane or to change lanes, followed by behavior planning and generating trajectory for the ego vehicle.

Prediction: [Line 108-164]

In this part of code, we use cars telemetry and sensor fusion information to figure out the following situations to act accordingly:

- Is there car ahead of my ego vehicle on the same lane and distance between these 2 cars is less than 30 meters?
- Is there a car on my left lane making lane shift unsafe?
- Is there a car on my right lane making lane shift unsafe?

Behavior planning: [Line 166-201]

In this section we take decisions on target lane and speed adjustment.

If there is car ahead and no car on left lane and we are on right lanes – then we do a left lane shift.

If there is car ahead and left lane shift is unsafe and no car on right lane and we are not on rightmost lane – then we do right lane shift.

If there is a car ahead and left/right lane shift is unsafe and if we are not in a safe distance of 30 m -- then we reduce the speed by 0.224 mph, which is used to update ref_vel in future.

If there is no car ahead on the same lane and if we are on left most or right most lane we are moving to center lane and if ref_vel is lesser than max speed limit we increase it ref_vel by 0.224 mph.

Trajectory generation: [Line 204-317]

This code generates a trajectory (next x and y values) based on the car localization information received from simulator and speed with lane output from the behavior planning and previous path points.

First, the last two points of the previous trajectory (or the car position if there are no previous trajectory) are used in conjunction with three points at a distance of 30 m to initialize the spline calculation. To make the work less complicated to the spline calculation based on those points, the coordinates are transformed (shift and rotation) to local car coordinates.

In order to maintain continuity, previous path points are added to next x and y values list. We generate new (50-prev_size) number of points. Ex: if car has driven 10 points out of 50 and we only have to generate now 10 more points to keep it to 50 points in total.

Using mathematical equation N*0.02*ref_velocity = d, we calculate the number of points spaced in 30 m distance and using spline we get corresponding y values and then we are transforming output coordinates and adding to next x, y values.

Conclusion:

Current implementation has simple algorithm to drive car safe. To improvise the performance and for selecting a best trajectory, cost functions can be introduced.