UDACITY

- Logout
-

Project Specification

Path Planning

Compilation

Criteria	Meets Specifications
The code compiles correctly.	Code must compile without errors with `cmake` and make. Given that we've made `CMakeLists.txt` as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.

Valid Trajectories

Criteria	Meets Specifications
The car is able to drive at least 4.32 miles without incident..	The top right screen of the simulator shows the current/best miles driven without incident. Incidents include exceeding acceleration/jerk/speed, collision, and driving outside of the lanes. Each incident case is also listed below in more detail.
The car drives according to the speed limit.	The car doesn't drive faster than the speed limit. Also the car isn't driving much slower than speed limit unless obstructed by traffic.
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.	The car must not come into contact with any of the other cars on the road.
The car stays in its lane, except for the time between changing lanes.	The car doesn't spend more than a 3 second length out side the lane lanes during changing lanes, and every other time the car stays inside one of the 3 lanes on the right hand side of the road.
The car is able to change lanes	The car is able to smoothly change lanes when it makes sense to do so, such as when behind a slower moving car and an adjacent lane is clear of other traffic.

Reflection

Criteria	Meets Specifications
There is a reflection on how to generate paths.	The code model for generating paths is described in detail. This can be part of the README or a separate doc labeled "Model Documentation".

Suggestions to Make Your Project Stand Out!

Create a path planner that performs optimized lane changing, this means that the car only changes into a lane that improves its forward progress.

Rubric

For this project the task was to build a program to drive the simulator.

I found it was easiest to use the Frenet coordinates as these seem to be the most intuitive.

A simple Finite State Machine was used to change lanes on the condition there was no car on the left or the right by checking the sensor fusion data which gave the relative positions of the other cars on the road. Car targeted staying in the middle lane.

Steps.

Previous points of cars movement were added to waypoints

Future waypoints were calculated at distance of 30, 60 and 90 meters

Convert points into car coordinates

Create spline to fit waypoints

Calculate target 30 meters ahead

Interpolate points ahead with spline

Car performs reasonably well in the simulator, with runs over 1 hour without any collision incidents.