Akshay Khole
Udacity SDCND
Path Planning Project

To implement the path planning project, we take the following steps:

1. **Drive the car in its own lane**
   To drive the car in it's own lane, we make use of the Frenet coordinates. On a basic level, if we keep the 'd' value, i.e. distance from the solid double yellow line constant, we can ensure car stays in it's lane. We generate a set of waypoints that the car needs to follow. We space the waypoints evenly along the lane. The distance of each waypoint from the yellow double line is given by the 'd' frenet coordinate. By generating a set of such waypoints, the car stays in the assigned lane.

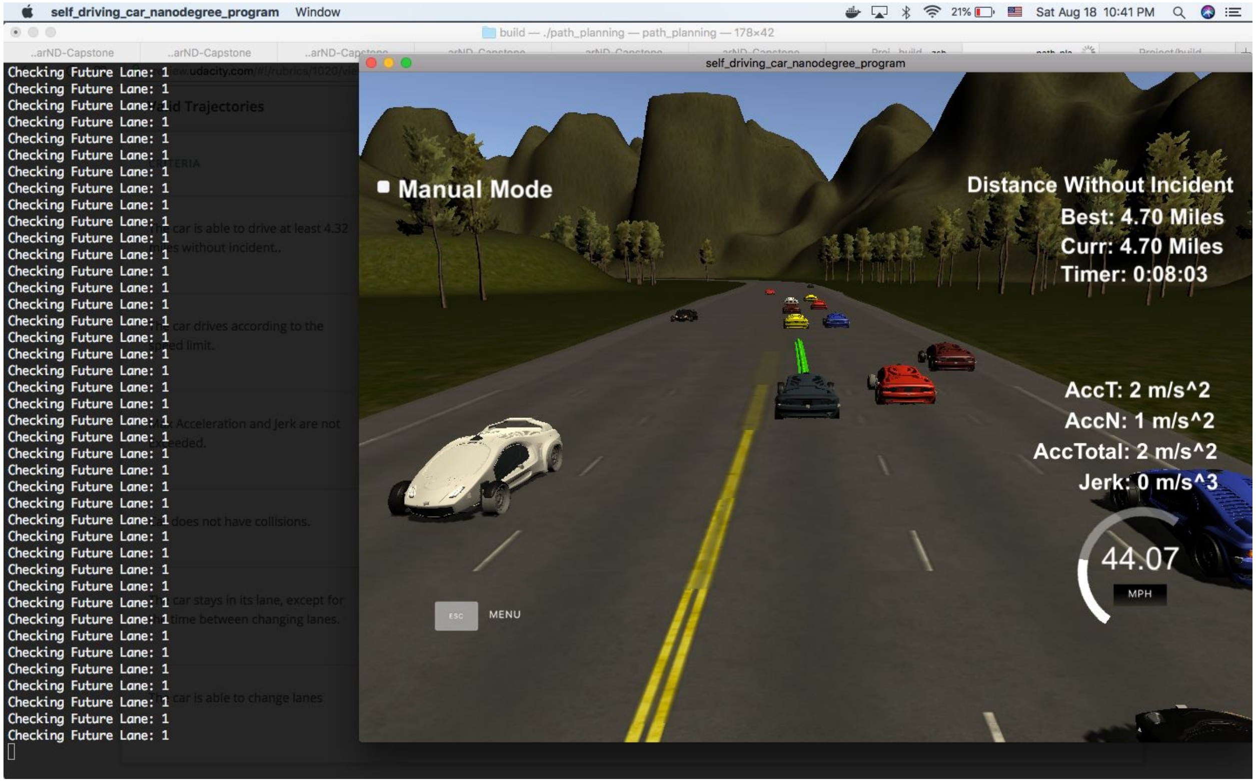
2. **Maintain speed of ~50mph with smooth acceleration and deceleration**
   The car tries to accelerate until it reaches a speed of 49.5mph and then tries to maintain that speed. We set the value as 49.5 mph as the speed limit is 50 mph, so we don't want to go over the limit. We start off with a speed of 0mph and if there are no cars ahead of us, then we move forward and increase acceleration in steps of 0.224 until we reach desired speed. When we detect a car head of us, we decelerate in steps of 0.224 until we are at a safe distance from the car. The increment and decrement in steps helps us minimize jerks. The value of 0.224 is optimal for minimal jerks.

3. **Avoid collisions**
   To avoid collisions, we make use of sensor fusion data, the sensor fusion module gives us a list of all the vehicles around us and their frenet coordinates. Using some simple calculations, we are able to determine positions of other vehicles with respect of our car. We accelerate and decelerate based on whether there is a vehicle ahead of us.

4. **Change lanes if possible, when car ahead is moving slow**
   When the car head of us is moving slow, we want to change lanes so we can maintain our speed of 50mph for as much time as feasible. When we detect a slow moving car ahead of us, we use sensor fusion data to check if any of the applicable neighboring lanes have any vehicles that would prevent us from moving into that lane. The 'can_move_into_lane' function returns a boolean based on whether it is safe to move into that lane. The function ensures there are no cars within a safe distance ahead of behind the car, should we move in that lane.

Test drive results:

- The car is able to navigate the track for 4.7 miles without incident.


self_driving_car_nanodegree_program
Window
21%
Sat Aug 18 10:41 PM
build — ./path_planning — path_planning — 178×42
..arND-Capstone
..arND-Capstone
..arND-Capstone
self_driving_car_nanodegree_program
ESC
MENU

```
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
Checking Future Lane: 1
```

□ Manual Mode

Distance Without Incident
Best: 4.70 Miles
Curr: 4.70 Miles
Timer: 0:08:03

AccT: 2 m/s^2
AccN: 1 m/s^2
AccTotal: 2 m/s^2
Jerk: 0 m/s^3

44.07
MPH