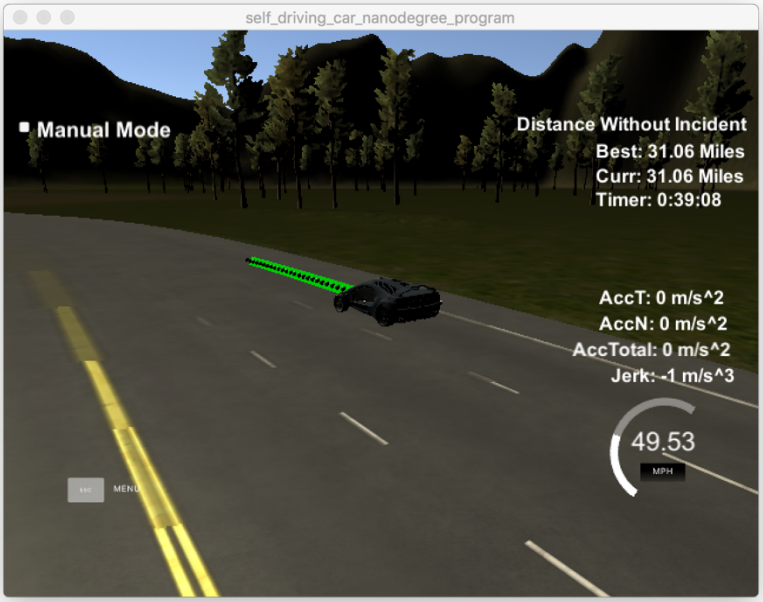
Behavior Planning

Compilation

Program compiles correctly with cmake, make in a MacBook Pro retina with High Sierra Operating System.

Valid Trajectories

- The program is capable to run more than 4.32 miles without incidents.



- The car drives according to the speed limit. It does averages of 47/49 mph depending traffic.
- I have made some tests driving it at 60 mph and seems reacts correctly. Of course the speed violations continuously prevents detecting collisions if not looking permanently to the screen.
- Max Acceleration and jerk are not exceeded. When driving straight they are between 0 and 1 m/s2. When changing lanes about 3 m/s2.
- Car does not have collisions. Normally. Sometimes some crazy driver changes lane just in-front of you. Usually manages it quite ok but sometimes is near impossible.
- To solve this problem, a car is considered an obstacle always that is in your lane or is more than 1m offset from the center of his lane towards our car.
- The car stays in the lane except for the time changing lanes. It works correctly but as I have preferred to be a bit more reactive if I block the computer when changing to lane 0 I may get an out of lane. It does not happens if I made it less fast in response by incrementing the number of previous points used. That problem is also present if I increase resolution or quality of the simulator.
- The car is able to change lanes. It changes lanes without problems to the fastest lane. Sometimes it may get caught between slow cars. With current algorithm is difficult as it always looks ahead.

Reflection

The planner has 2 stages. In first one information from the environment (fusion) cars is used to get an idea about which lanes would be ok to change into and which speed is associated to each lane.

A lane is ok to change if there is no car in a space around our s position. The program defines two constants, one for the distance from the previous car which is increased in case this car has more speed than us and another for the next car.

Speed for a lane is computed as the slowest car in the lane in an defined horizon (60m.).

A lot of work has been done to get quite good speed control. To be able to get consistent and smooth response the program detects if there is a slower car at some distance. Then adjusts our target speed to that of the car. That way when following it we have minimal distance variations.

Once we get the environment the algorithm is very easy:

- If our lane is the fastest just continue in the lane
- If not and the lane in the direction of the fastest lane is free change to that lane

Also if speed is slower than set speed accelerate and if too much just decelerate.

The path is generated as in the presentation. All the code is in a function to be able to implement different parametrized paths but is not used in the submitted program. There is another one that generates 9 different paths and selects the best with a cost function but I have not been able to get it working correctly. I have some fuzzy ideas about when a car collides (where is the center point of a car and what's it's size) which made it difficult to

Parameters

From line 375 to 385 there are some parameters to get different behaviors, more or less aggressive.

- int prev_path_max_size = 5; // Number of points from previous path used. Bigger means softer trajectories, smaller more reactive.
- double max_vel = 49.5/ 2.24 ; // Desired maximum speed
- double delta = 0.02; // simulator delta t
- double max_forward_acc = 5.0 ; // in m/s
- double max_brake_acc = 8.0; // in m/s . Sometimes 9.0 also works
- double pts_space = 30.0; // Distance between reference points to compute spline
  double next_car_distance = 10.0; // Minimum distance to previous car. > 15 is secure.
- double prev_car_distance = 15.0; // Same with previous car.
- double speed_horizon = 30.0; // Horizon to compute speed of lanes. Sensors?
- bool europe = true; // Then try to drive to the rightest lane for your speed and NEVER advance by the right. Not implemented

Some ideas

Create a tree that may be searched by something like a Hybrid A*. This may solve the blocking problem and instead of arriving to an end it may use some metric as how to optimize the s value with a fixed number of steps. Will try, some ideas I am working into.