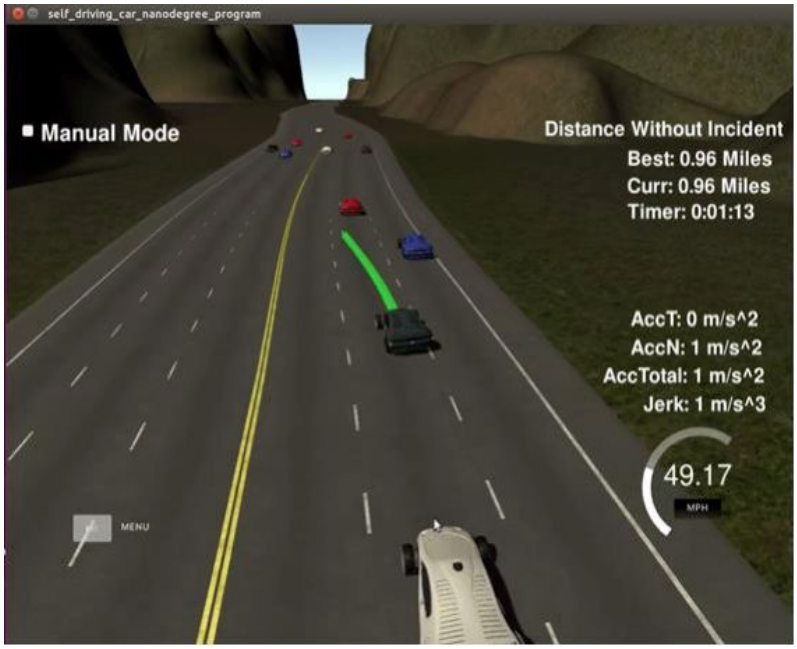
Path Planning

The Goal of this project is to drive a car around a simulated highway for at least 4.32 miles without any incident such as collision, exceeding speed limit, jerk etc. Car should drive smoothly under 50mph.

The highway has 6 lanes in total, - 3 heading in each direction. Each lane is 4 m wide and the car should only ever be in one of the 3 lanes on the right-hand side. The car should always be inside a lane unless doing a lane change. The various criteria and how it's handled are discussed below.



The implementation comprises of mainly two stages,

1. Trajectory generation □

2. Behavior planning □

Link to video : https://youtu.be/FlefgcLZbdE

Trajectory generation:

As a first step, as mentioned in the Q&A video, care was take to allow maximum velocity to the ego car, by setting it to 49.5mph. The jerk limit is 10 /m^3. Care was taken to have minimum jerk trajectory motion due to any increase in sudden speed of the ego car from 0 mph to 49.5 mph in time T. To have a jerk free and smooth trajectory, third party spline method was used to fit a 5^{th} degree polynomial.

The tk::spline function takes translated local X/Y coordinates as inputs. These local coordinates are translated from global X/Y coordinates, which got converted from the car's Frenet s, d coordinates. Converting to local coordinates helps in projecting the Ego vehicles waypoints at 30, 60, and 90m once the spline is computed.

Behavior Planning:

Behavior planning helps to drive car efficiently and safely. Driving efficiently, means finding the best lane to reach the destination safely. Following three points are taken into considering in this project.

1. Fewer lane changes.
2. Larger the distance to the moving object ahead, for better safe maneuvers.
3. Greater the velocity of the object ahead, the faster the car can drive in the lane

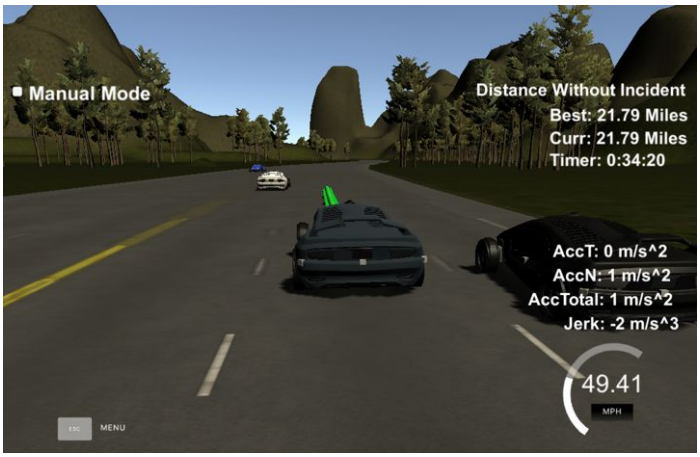
Rubric points:

The code compiles correctly.

A new file, which is cubic spline implementation, was added to the src directory. It is a single header file "single .h". It was used instead of polynomials.
Ref: suggestion from the classroom QA video.

1. The car is able to drive at least 4.32 miles without incident..
I ran the simulator for 15 and 20 miles without incidents.

2. The car drives according to the speed limit.
Car drives within 50mph



3. **Max Acceleration and Jerk are not exceeded.**
Max jerk was not exceeded and no red message was seen.

4. **Car does not have collisions.**
No collisions were seen in 20 miles drive.

5. **The car stays in its lane, except for the time between changing lanes**
The car stays in its lane most of the time except. Only changes lanes either to avoid collision or move away from slow moving traffic.

6. **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.

Reflection

Starter code that was provided for this project is in src/main.cpp. For this project I have only worked in the main.cpp. My code focuses on trajectory and behavior planning. As a future work I would like to create and work on more structured code that should have been modeled based on the suggestions provided in classroom, using cost function, JMT etc.

Data From Sensor Fusion:

Data from sensor fusion was used to perceive the region surroundings of the ego vehicle. Sensor fusion provides [id, x, y, s, d, vx, vy] data of the other cars in the simulator. Lane changing trajectories take this data into consideration and provide and propose "safe" behaviors for the car to avoid collision with other vehicles.

This part of the code deals with the telemetry and sensor fusion data. (line no 274- 280)

```
float d = sensor_fusion[i][6];
double vx = sensor_fusion[i][3];
double vy = sensor_fusion[i][4];
double check_speed = sqrt(vx*vx + vy*vy);
double check_car_s = sensor_fusion[i][5];
```

Setting Speed Limit:

The maximum velocity of the car is set as 49.5mph so the ego car wont exceed the speed limit of 50mph. But we will have a starting problem where the car speed increases from 0mph to 49.5 in time T. This will exceed the jerk limits of 10/m^3. To avoid Jerk , ego car slowly accelerates at every time step at the rate of 50mph which is equivalent to 22.4 meters/sec.

If the reference velocity of the ego car is greater than the maximum allowed velocity then decelerate by the same amount.

```
if (too_close)
                    {
                    //if there is vehicle ahead reduce the speed
                    ref_vel -= .224 * 2;
```

```
else {
                    if(ref_vel < 49.5) {
                    ref_vel += .224 * 2;
                    }
```

Check if there are cars around the ego car (lines 294-336)

A car is considered "dangerous" when its distance to our car is less than 30 meters in front or behind us.

- Is there any car in front of ego car either on left or right side?

```
// if the observed car is ahead and distance is too close
                    if ((dist_between > 0) && (dist_between < 30))
                    {
                      // if an observed car is on the left side
                      if (obsvd_car_lane == (lane-1)){
                              if (space < space_on_left) {
                            space_on_left = space;

                              }
                        left_lane_free = false; // left lane is not free for lane change
                      }
                      // if an observed car is on the right side
                      if (obsvd_car_lane == (lane+1)){
                              if (space < space_on_right) {
                            space_on_right = space;

                              }
                        right_lane_free = false; // right lane is not free for lane change
                      }
                      }
```

- Is there a car behind either on left or right?

```
// if car is at the back, within 20m
                    else if ((car_dist < 0) && (car_dist > -30))
                    {
                    // if an observed car is on the left side
                    if (lane_obsvd == (lane-1)){
                       left_is_free = false;
                    }
                           // if an observed car is on the right side
                    if (lane_obsvd == (lane+1)){ // if an observed car is on the right side
                       right_is_free = false;
                    }
                    }
```

- Is there a car in our lane?

```
                    if ((d<2+4*lane+2) && d > (2+4*lane-2))
                    {
                    // if observed car is too close
                    if ((check_car_s > car_s) && abs(car_dist) < 30)
                    {
                       too_close = true;
                       // if an observed car is on the left side
                       if (lane_obsvd == (lane-1)){
                          left_is_free = false;
                     }
                     // if an observed car is on the right side
                       if (lane_obsvd == (lane+1)){
                          right_is_free = false;
                       }
                    }
                     }
```

For Behavior planning program decides to do: (lines 340-380)

- If there is any car in front/behind of us within 30 meters range, what action we take?

```
// if car is too close


                    if (too_close)
                    {
                    //if there is vehicle ahead reduce the speed
                  ref_vel -= .224 * 2;
                    //change the lanes safely if lane is available
                    //if the lane is far most left lane and Right lane is free
                    if ((lane==0) && right_is_free)
                    {
                   lane = 1;
                }
                    // If the lane is middle one
                    else if (lane==1)
                    {
                   //both left and right lanes are free
                   if (left_is_free && right_is_free)
                   {
                      //check the available space for lane change
                      if (space_on_right > space_on_left)
                      {
                              lane += 1;
                      } else {
                              lane -= 1;
                      }
                   } else if (left_is_free) {// if only left lane is free
                      lane -= 1;
                        } else if(right_is_free) { // if only right lane is free
                      lane += 1;
                   }
                    }
                    else if ((lane==2) && left_is_free)
                        {
                   lane = 1;
                    } else if (lane>2) //fail fast condition
                    {
                   lane = 2;
                    }
                    } else {
                    if(ref_vel < 49.5) {
                   ref_vel += .224 * 2;
                    }
                    }
```