# Self-Driving Car Engineer Nanodegree Program

## Path Planning

**John Reilly**

**Due date May 2019**

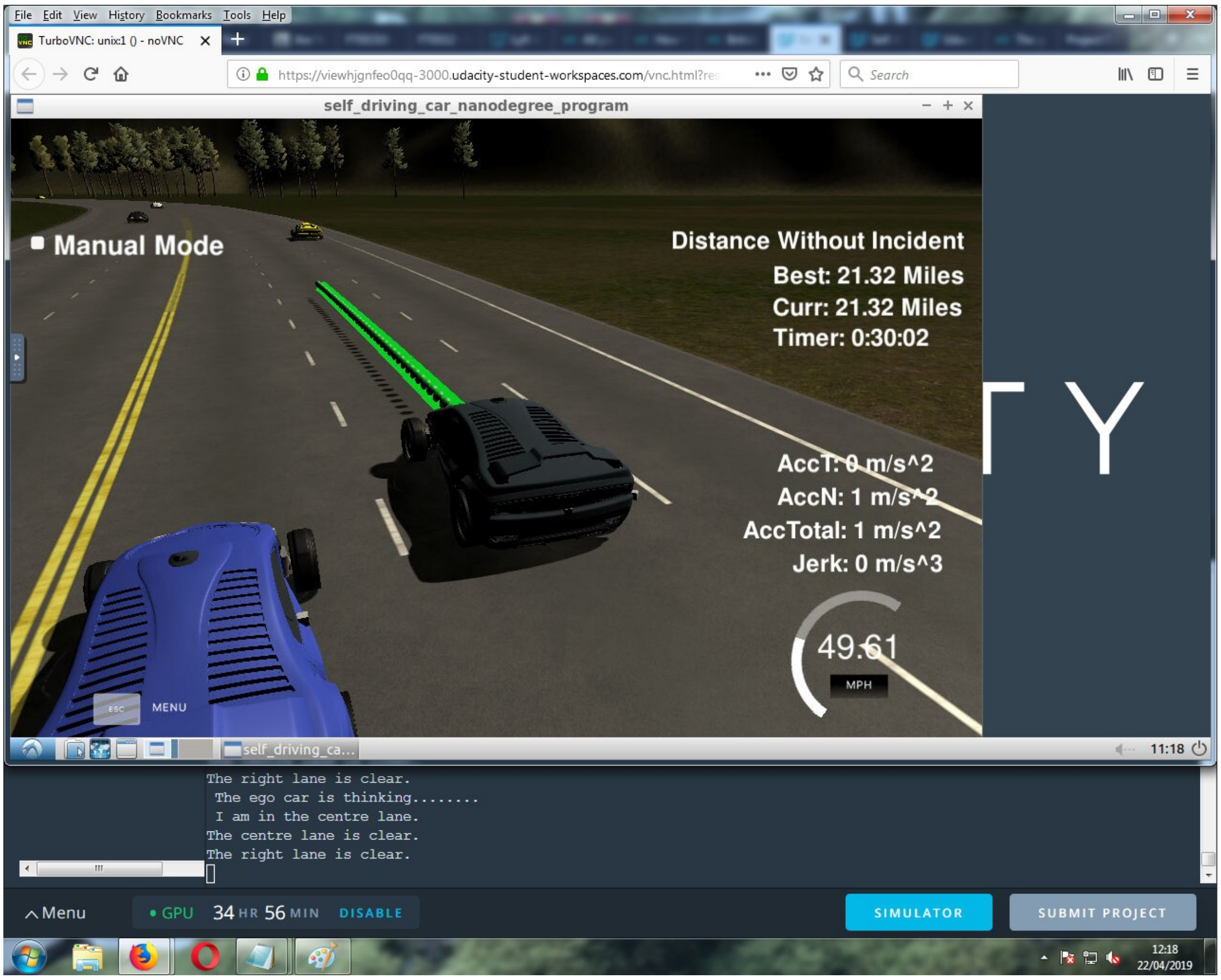
## Table of Contents:

**Rubric point, Compilation:**

The code compiles correctly using the *Cmake .. &&* make command and files that were provided by Udacity. I did note that upon restarting the Workspace environment sometimes *install-ubuntu.sh* or cmake would have to be installed. Having said that the program compiles with the dependencies and environment provided.

**Rubric Point, The car is able to drive at least 4.32 miles without incident:**

The car was easily able to do more than 20 miles without incident at the final version and was stopped due exceeding the requirement rather than an incident. See image below.



```
The right lane is clear.
 The ego car is thinking........
 I am in the centre lane.
The centre lane is clear.
The right lane is clear.
```

**Rubric point, The car drives according to the speed limit.:**

As shown above the car cruised at just below the speed limit. The following line at *main.cpp line 263* controlled the upper limit to just below the speed limit. The smoothing effect allowed the speed to exceed this level slightly but it never exceeded 50mph

e*lse if (ref_vel < 49.5 )*

**Rubric point, Max Acceleration and Jerk are not Exceeded. :**

The car can drive for several minutes without exceeding Max Acceleration and Jerk, (which is sudden acceleration or deceleration).

However it was noted that it was possible for a car to cut in front to the ego car or for a car in the lane ahead to brake and in this situation it was possible to have a collision with the deceleration rate as provided in the Q+A video which was 0.224.

While a collision and a hard deceleration both count as an event requiring the distance timer to be reset I thought it was better to brake in the name of safety that collide. So a number of deceleration rates were used. The first to slow down or accelerate gently in normal use the second to slow down at an increased rate risking a “jerk” event, the next a further harder deceleration and the final and extreme deceleration rate. However in tests it was difficult to recreate the situation where these extra rates were needed so it was not possible to verify these rates work as expected but I think it's more realistic to have varying rates of acceleration and deceleration in the program logic as shown below line 58.

*double gentle_acceleration = 0.224 ; // this was Q+A provided level*

*double jerk_limit_acceleration = 0.300 ;// higher level risky but I am getting caught by lane cut off vehicles and need to slow down faster*

*double brake_rate = 0.4 ;// I just picked a higher number than above ....*

*double emergency_stop = 0.6;*

***Rubric point: Car does not have collisions.***

The car could go many miles without collisions. However there were some rare scenarios where it was more likely and steps were taken to address this. The first scenario was if the ego car approached a much slower group of cars blocking the road. The ego car did not always decelerate fast enough to avoid collision. To improve this multiple rates were devised with 1 being near the limit of a warning and the other being very near risking a warning. These 2 rates of deceleration were enough but a further 2 rates for more dramatic braking were available to avoid a collision at the expensive of a sudden deceleration.

Antoine scenario that caused a collision on one occasion was another car cutting in front of the ego car without enough room for the ego car to slow down. This happened before additional rates were introduced but the event I witnessed have have needed an extreme braking manover to prevent collision and that's why the extra rates were added. This only happened once and could not be tested again as the situation did not arise in many minutes of testing.

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

This worked fine for many tests but at one point gave problems. The ego car would sit perfectly between two lanes and trigger a warning. The problem was when there were 2 equally good choices the car would pick one and move towards that lane then pick the other and move toward the other lane and ending up between two lanes as it it had chosen the exact mid point. The cause of this was not understood until I build debugging outputs to show what was happening and it was then clear that the lane value would oscillate between 0 and 1 for example and that gave the midpoint between the two lanes. This began the development of the debug outputs into an overview of what the car was "thinking".

**Rubric point The car is able to change lanes :**

As per the Udacity Q+A video the Spline library was used to generate smooth points for the car to track and the lane variable which came from a Frenet coordinate the car could smoothly move between lanes. In fact I saw it change from left to right so smoothly that it looked like a double change but the outputs showed it choose the middle lane then the right almost immediately and the change was still smooth. So the spline technique worked really well.

## Reflection:

**Rubric There is a reflection on how to generate paths.**

As a general point the student work is from line 107 to 290 with a few lines around 55. This reflects the huge amount of material supplied by Udacity that made the project possible and also the point of the project was to focus on path planning not 3D car simulators etc. Also the work is contained in a relatively small area to make it easier to correct and more straightforward to write. In a real world scenario the same work would be organised into neat separate functions or classes. Here it is all together making reading somewhat easier.

The first idea implemented was to track which lanes were open and by that I mean distance ahead and behind suitable for moving into and also some booleans for tacking was there are car in my lane if so how near and I had 4 booleans for that. As the cars got nearer different booleans became true.

At line 140 is the loop that cycles through the entire sensor fusion vector which has information about all the cars that the ego car can detect. I check each lane against the sensor fusion data to see if it means there is a car in the lane if yes then I set the boolean and this way the fact that I cannot move into that lane is remembered. Distance of 50m ahead and 20 m behind were considered safe to move into. The figure 50 metres ahead was an increase of 30m from the Q+A video and was an experiment that worked well. Ideally many more levels would be checked but it was hard to consistently test ideas with the random nature of the simulated traffic. It could be considered that these values are conservative but at least they have proved very safe while still allowing the car to change lanes and hit near maximum speeds regularly.

Once each lane is checked the sensor fusion data is checked against the ego car's current lane. And if the sensor fusion data suggested a car is in the same lane further checks will determine the distance of the car. Ranges of within 40 meters, within 30 meters , within 15 metres and within 5 meters are detected and stored in booleans.

Once the status of the lanes and the distances of the cars in the same lanes are determined the next step is to decided what to do. The car can accelerate or slow down in 4 rates two of which should not trigger a “jerk” event 2 likely will. The car can also change lane one lane at a time. It was considered to add a 2 lane change option but the car will do this anyway. For example if the car is in the left lane and it is better to be in the centre lane and better again to be in the right lane it will make a manover that looks like it went two lanes in one but actually it decided the centre land and then the right but it happens so smoothly it looks like it did two at a time. So no double lane change was in fact needed.

The main decision statement on line 220 handle most scenarios. When the car in front is too close either because it is moving slow or just slower than the ego car the ego car will change lane if possible.

The other scenario is when there is another car in the lane and it is only slightly slow maybe near 45mph and the ego car is less than the maximum speed. The ego car can gradually slow down while the car it is following is far ahead and it car drop it's speed and remain following the other car at a distance of more than 30m but still 45 mph or less. If the car that is holding up the ego car was a lot slower the ego car would either not slowdown fast enough and it would get within 30m or less and therefore change lanes but there were occasions when the ego car would adjust speed sufficiently fast to keep the car far ahead without slowing much. Sometime the ego car stabilized at around 45 mph with the other car at around 35 or more metres ahead. This situation was acceptable but the preference was to make good progress if possible and move to a faster lane if possible. Also if the other car was doing more than 49.5 mph the ego car would not have caught it so if the other cars gets within 30m the ego car is slowing down and could go faster. Its amazing how much this line achieves.

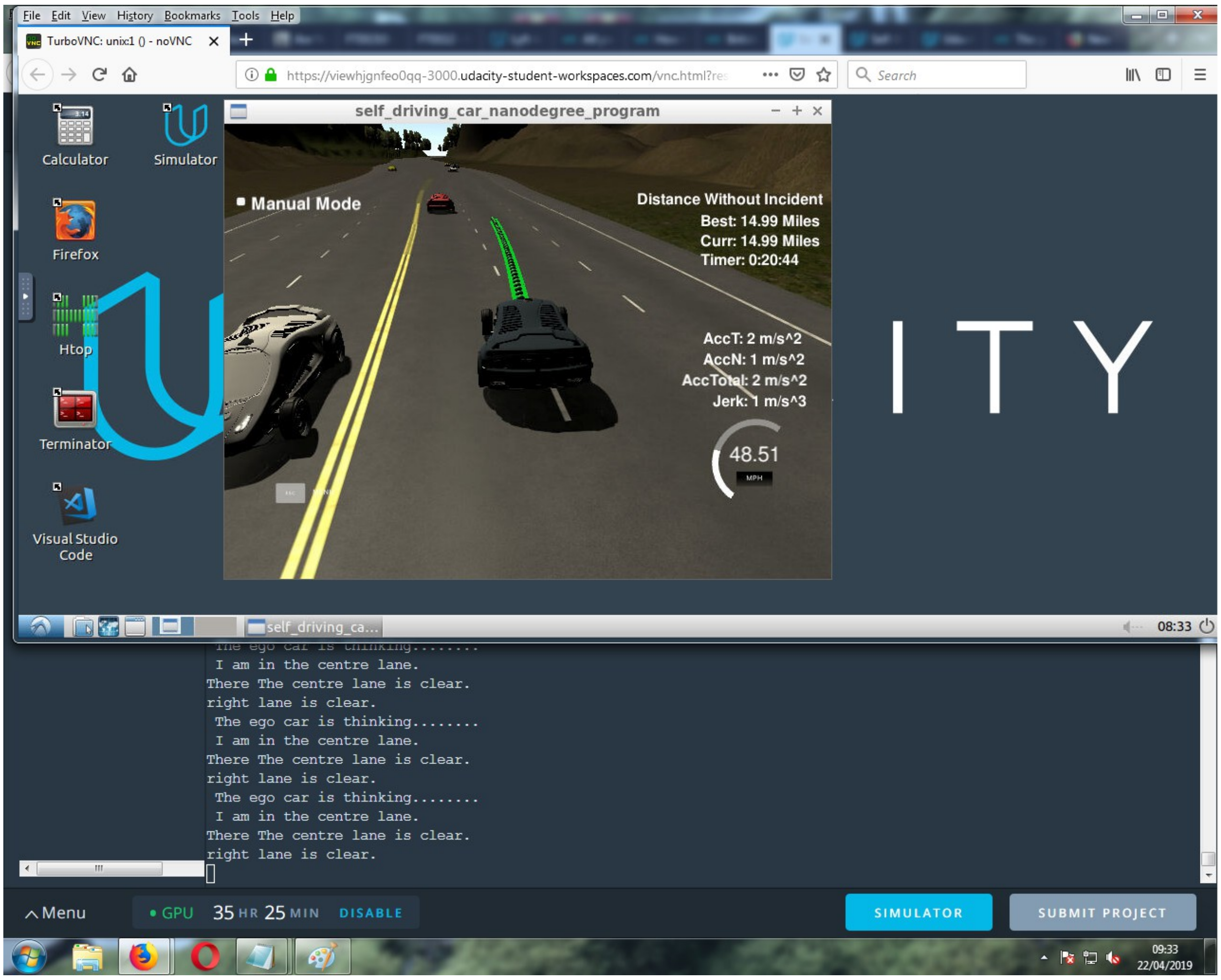
```
if( (another_car_in_my_lane == true && ref_vel < 45) || (within_30m == true ) )
```

**The stand out requirement** was to make the car choose a lane that improves forward progress and the above conditions help move the ego car past reasonably flowing traffic but not maximum speed traffic.

Line 246 to 270 has the acceleration and deceleration parts that will adjust the speed based on how close other cars are. I was not able to sufficiently test the braking and emergency stop rates as the right situation did not arise but I think it was good to have the options available.

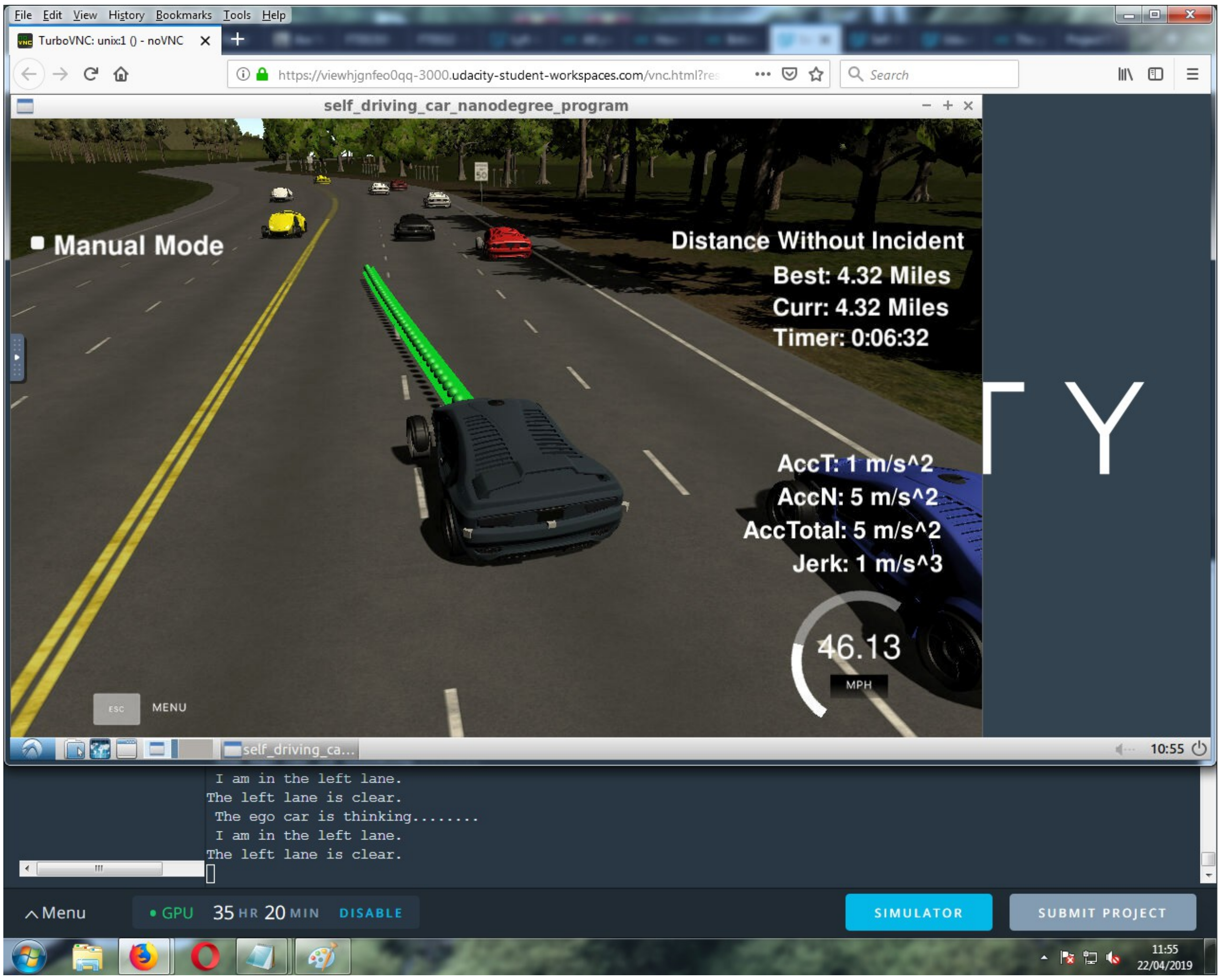
Line 272 to 290 is the output debug information. As the project evolved I was getting different output messages for different reasons and decide it would be fun to get the output statements to say what the car was seeing from its point of view. So the outputs will say “I am in the left lane”, “the centre land is clear” etc and this I think is a fun way of adding to the sense of artificial intelligence while still giving useful information.

## Sample images:



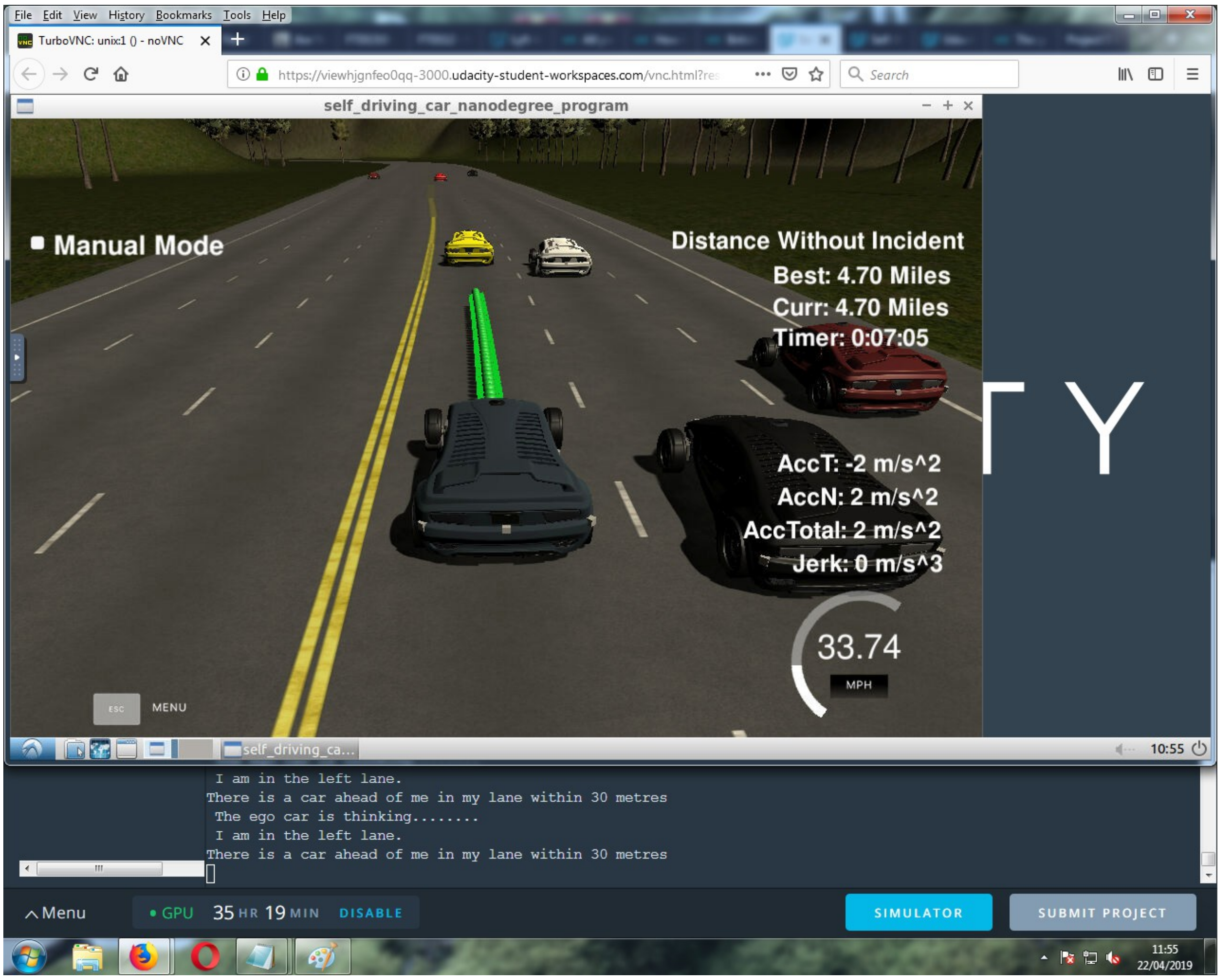
```
 The ego car is thinking........
 I am in the centre lane.
There The centre lane is clear.
right lane is clear.
 The ego car is thinking........
 I am in the centre lane.
There The centre lane is clear.
right lane is clear.
 The ego car is thinking........
 I am in the centre lane.
There The centre lane is clear.
right lane is clear.
```

Above Changing into a better lane with car ahead still far away while maintaining near maximum speed.



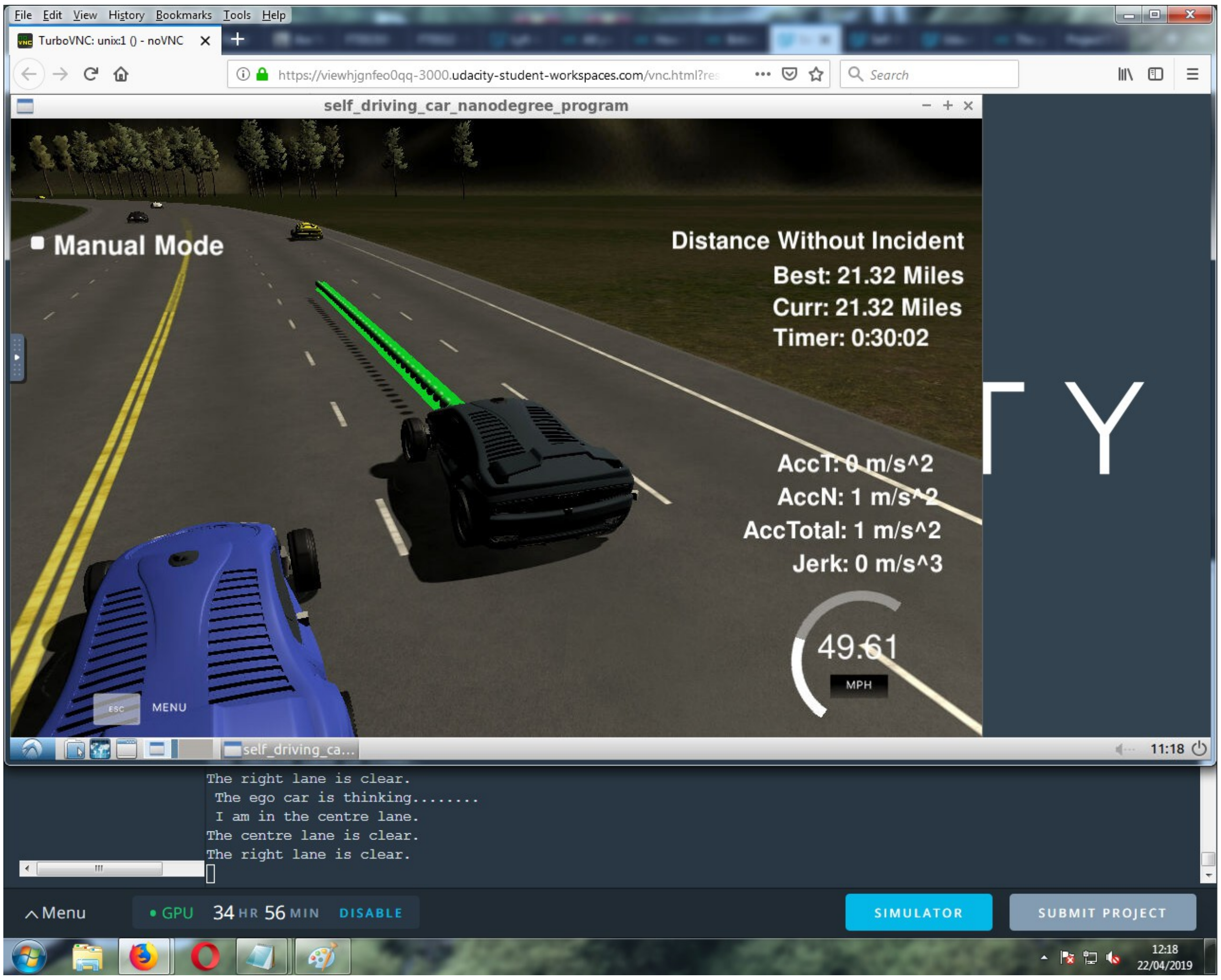
```
 I am in the left lane.
The left lane is clear.
 The ego car is thinking........
 I am in the left lane.
The left lane is clear.
```

Above nice lane change as it finds clear lane to allow small speed increase.



```
 I am in the left lane.
There is a car ahead of me in my lane within 30 metres
 The ego car is thinking........
 I am in the left lane.
There is a car ahead of me in my lane within 30 metres
```

Above: Look closely at the text output the car has detected the car ahead of it and reduced speed while maintaining a safe distance. This is very good as the car has reduced speed also the ego car has not labelled the other lanes as clear or safe to move into and so it slows down and waits, exactly as it should.



```
The right lane is clear.
 The ego car is thinking........
 I am in the centre lane.
The centre lane is clear.
The right lane is clear.
```

After 30 minutes and over 20 miles of safe driving just under the speed limit with open road......Life is good!