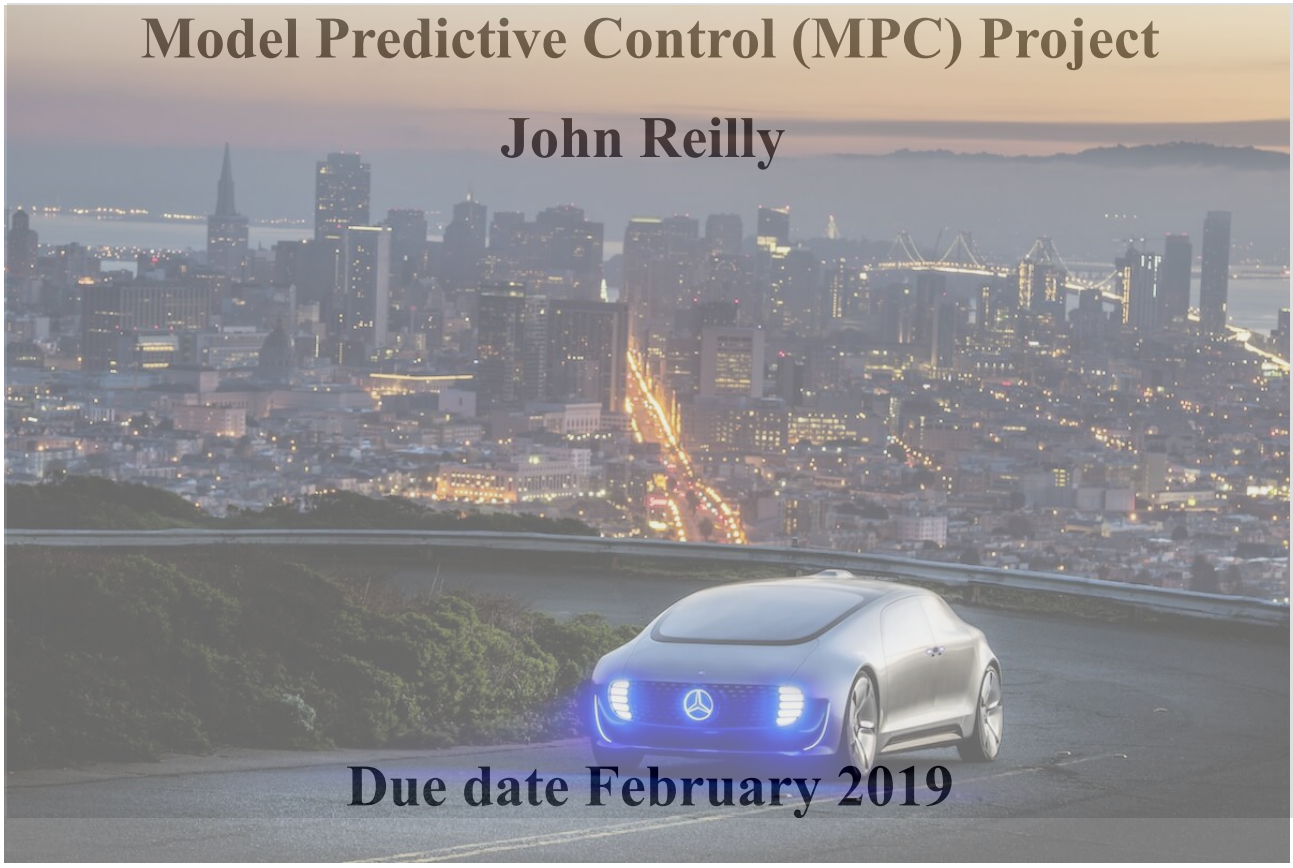
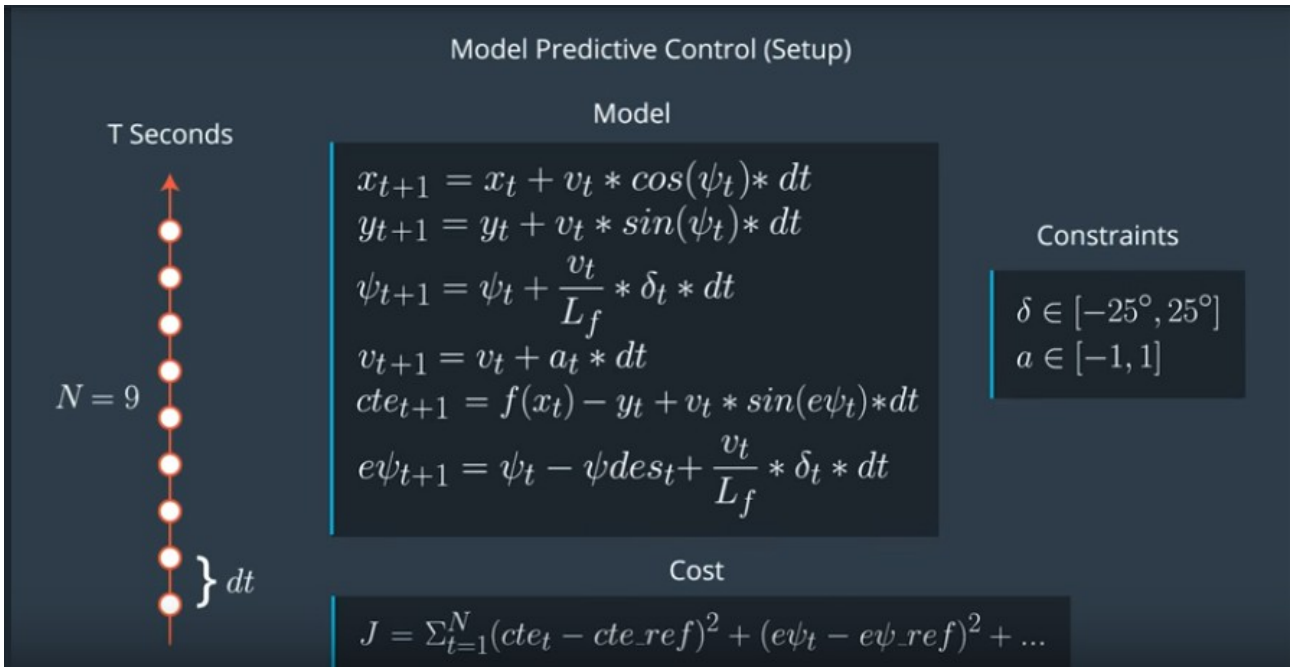

Model Predictive Control (MPC) Project
John Reilly
Due date February 2019

Rubric Points:

The Model:
Student describes their model in detail. This includes the state, actuators and update equations.

The model implemented in the project is the MPC or Model Predictive control model as presented in the video classes and shown in the video screenshot below.



Source Udacity Self Driving Car Nano Degree Term 2 , Lesson 19 Video 6

The MPC model uses the above equations to take the cars postion (x and y), the cars heading (psi) the cars veloctiy(v) the cross track error (cte) and the orientation error (epsi)

The outputs are the cars accleration (a) and the steering angle (delta)

Timestep Length and Elapsed Duration (N & dt) :
Student discusses the reasoning behind the chosen N (timestep length) and dt (elapsed duration between timesteps) values. Additionally the student details the previous values tried.

Timestep lengths of T = - 5 , 10 and 20 combined with Elapsed Duration dt of 0.1, 0.1 and 0.05 were tested.

The best combination was the values given in the Q+A video of Timestep 10 and dt 0.1 however some values were reasonable but not as good and many were much worse. Full details of experiments are later in report.

Polynomial Fitting and MPC Preprocessing :

A polynomial is fitted to waypoints. If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.

A polynomial is fitted using polyeval and polyfit these funtions are in the helpers.h file and provided by Udacity in the project repo. No further code was added to these functions and they are called to generate the fitted polynomial lines.

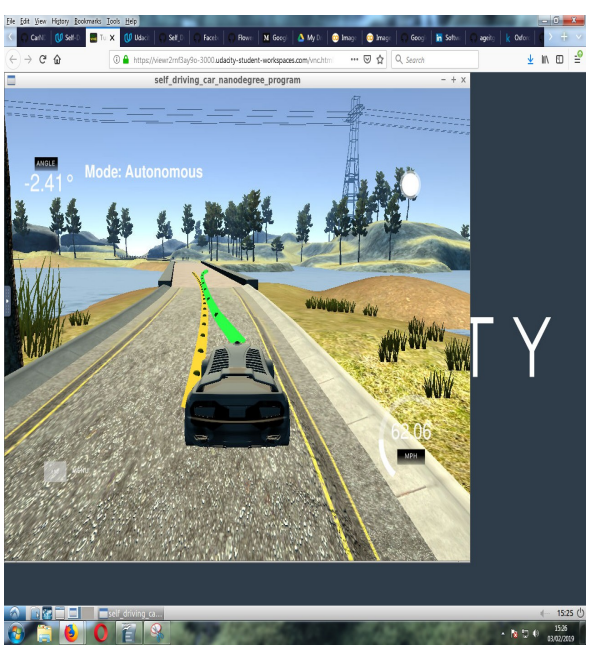
MPC preprocessing is done to take waypoints and convert them to the same co-ordinate system as the car or expressed another way to take the way poionts and traslate them to the cars point of view. This was presented in the Q+A video and implemented the same way in the project.

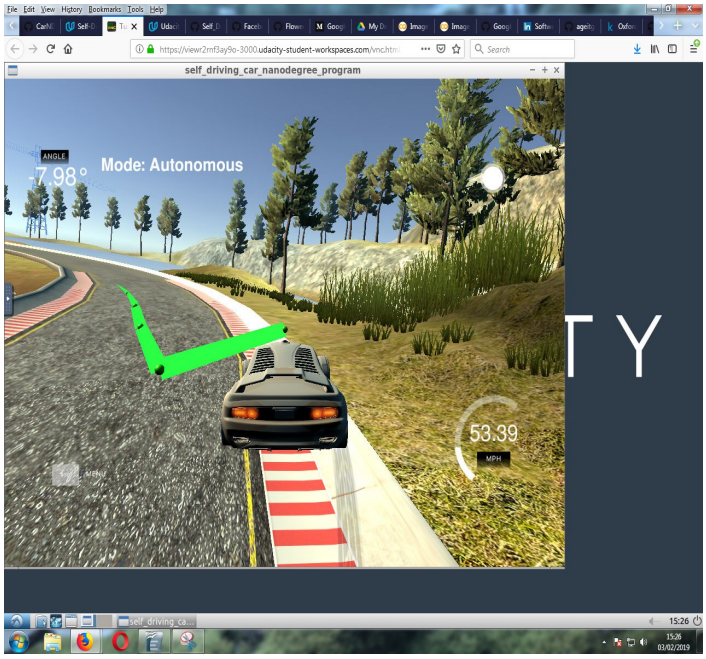
Model Predictive Control with Latency :
The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.

The MPC with latency was implemented using a simple delay in the main loop using the std::this_thread::sleep_for() function. The delay of 100 millisecond could be adjusted at this point but the delay seemed to work OK and the function allows for a real time delay in millisecond as opposed to a simple loop to count to a certain number which may vary in delay according to the speed of the executing processor. Therefore it was concluded that this simple method was sufficient and seemed to work well.

Below follow the screenshots of the numerous tested cost levels and N and dt values. A screenshot shows the values tested and a sample picture of the cars performance is shown with comments.

```
//from Q+A video
fg[0] = 0;
////////////////
// This is the cost variables for below listed here for convience
//reference state costs
double cte_ref_state_cost = 2000 ; //1000 //2000 in Q+A
double epsi_ref_state_cost = 2000 ;//1000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 200 ;//250000 // 200 in Q+A
double a_plus_cost = 10 ;//5000 // 10 in Q+A
```

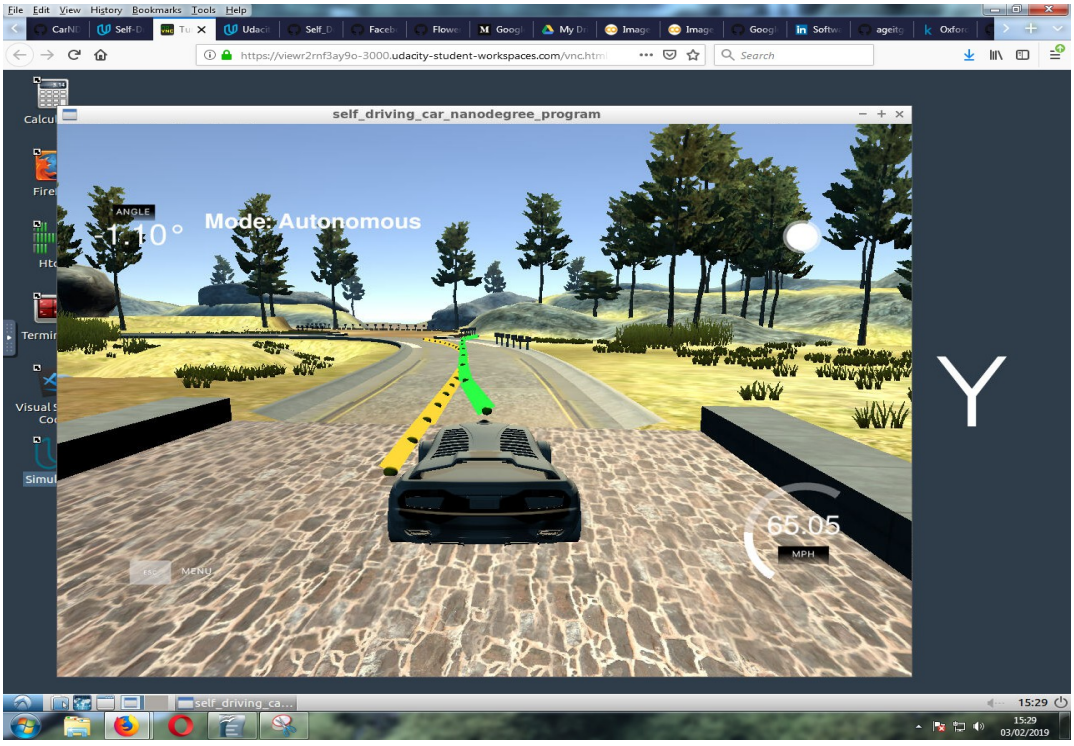




Comments : A bit harsh but goes around but with over steering and overreaction after a few laps crashes

```
////////////////
// This is the cost variables for below listed here for convience
//reference state costs
double cte_ref_state_cost = 4000 ; //1000 //2000 in Q+A
double epsi_ref_state_cost = 2000 ;//1000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 200 ;//250000 // 200 in Q+A
double a_plus_cost = 10 ;//5000 // 10 in Q+A

for(int i = 0; i < N ; i++)
{//2000 below is high weight
```

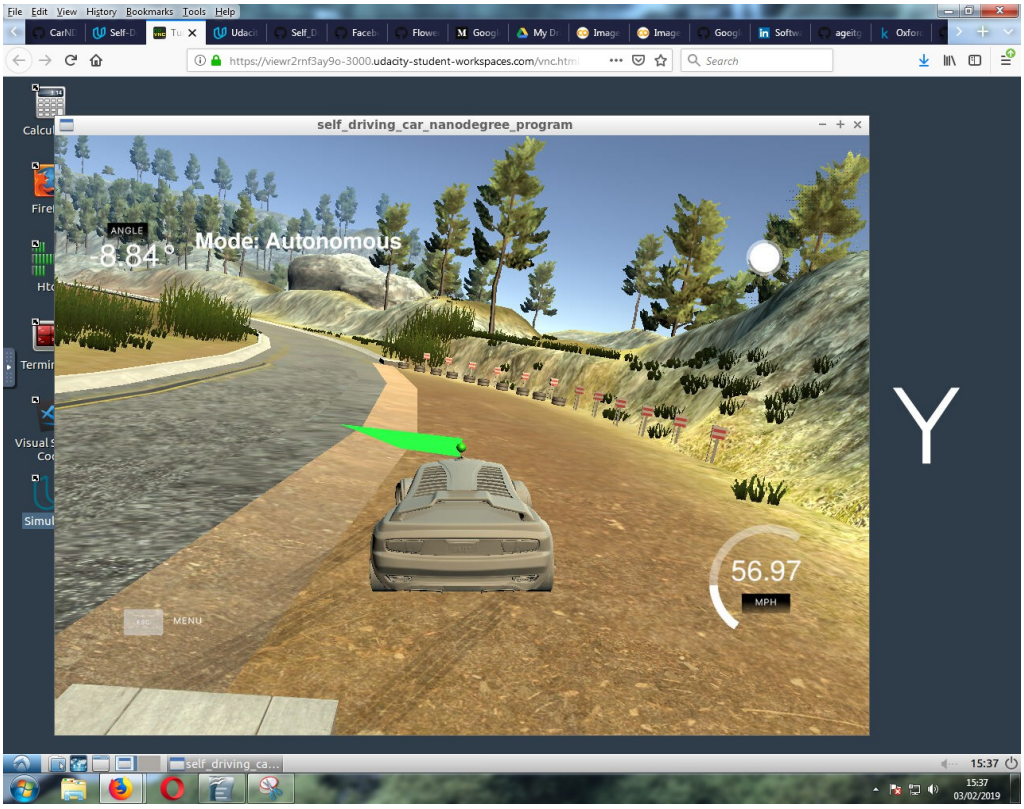


Comment:
It seems to snap back to centre line too harshly and more continuously oscillate

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 2000 ;//1000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 200 ;//250000 // 200 in Q+A
double a_plus_cost = 10 ;//5000 // 10 in Q+A
```

Comment:
Some snap back to centre but less harsh seemed better but crashed

```
// This is the cost variables for below listed here for convience
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 500 ; //4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 2000 ;//1000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 200 ;//250000 // 200 in Q+A
double a_plus_cost = 10 ;//5000 // 10 in Q+A
```

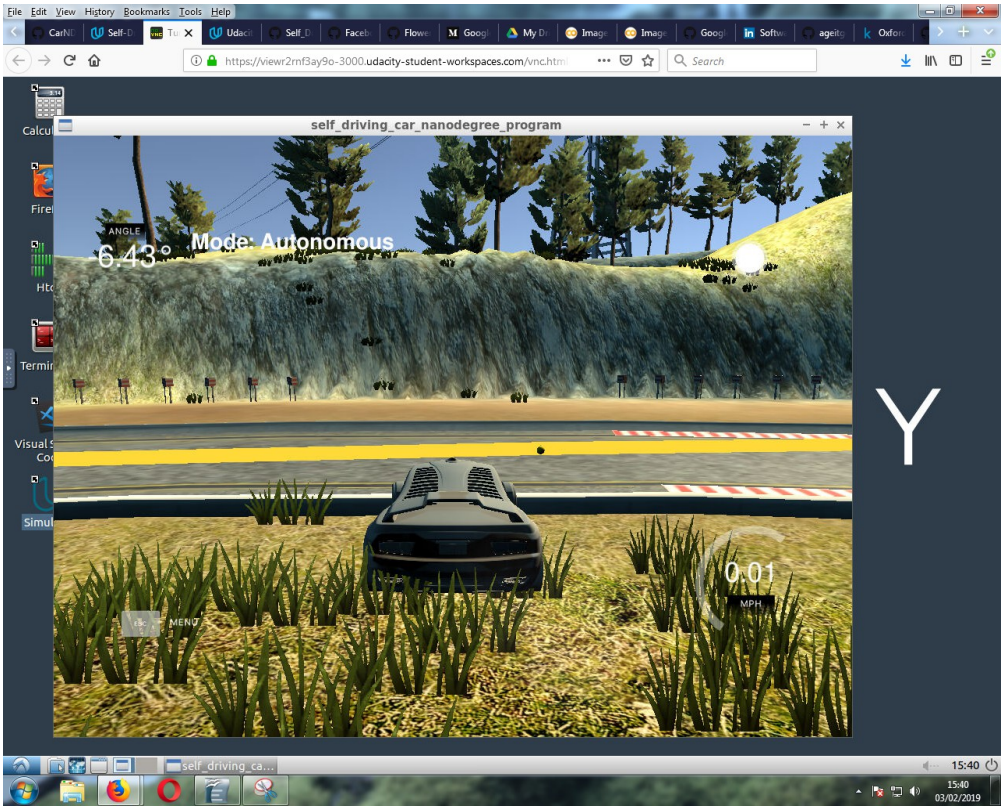


Comment:
Started well smoother curves but crashed

Adjusting all levels as CTE very low now compared to others

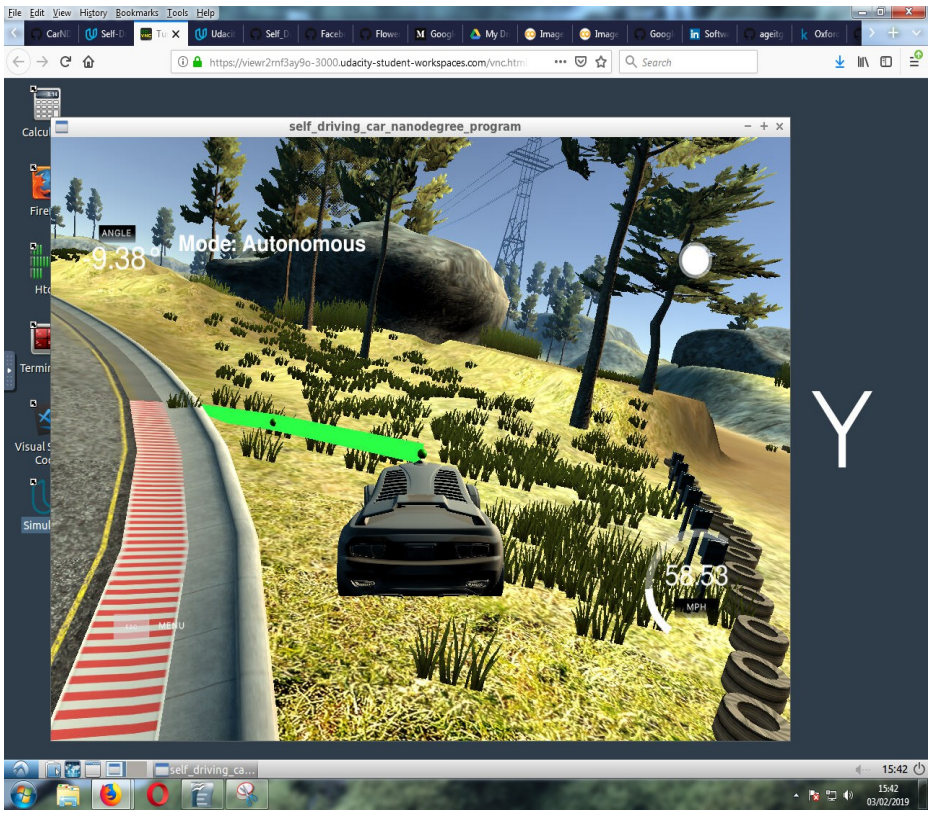
```
// This is the cost variables for below listed here for convience
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 500 ; //4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 500 ;//1000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 200 ;//250000 // 200 in Q+A
double a_plus_cost = 10 ;//5000 // 10 in Q+A

for(int i = 0; i < N ; i++)
{//2000 below is high weight
```



Comment:
About the same as last one

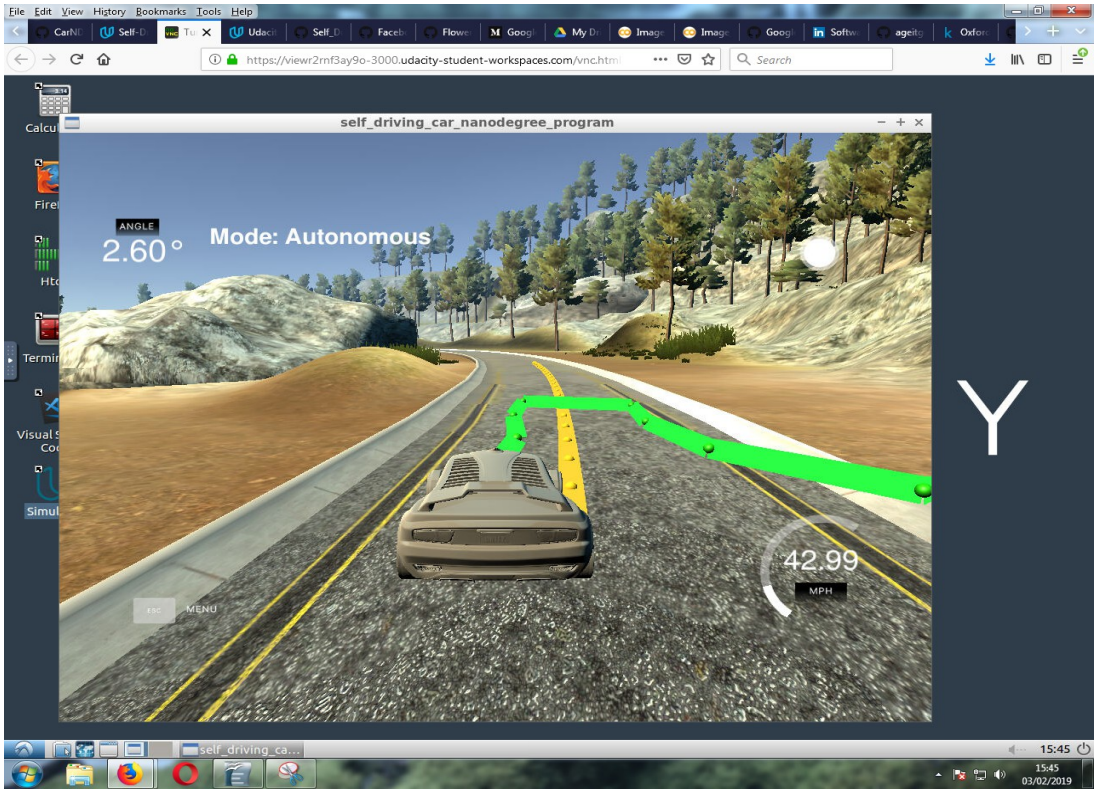
```
fg[0] = 0;
////////////////
// This is the cost variables for below listed here for convience
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1 ; //4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1 ;//1000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 200 ;//250000 // 200 in Q+A
double a_plus_cost = 10 ;//5000 // 10 in Q+A
```



Comment:
Too fast into corner after bridge.

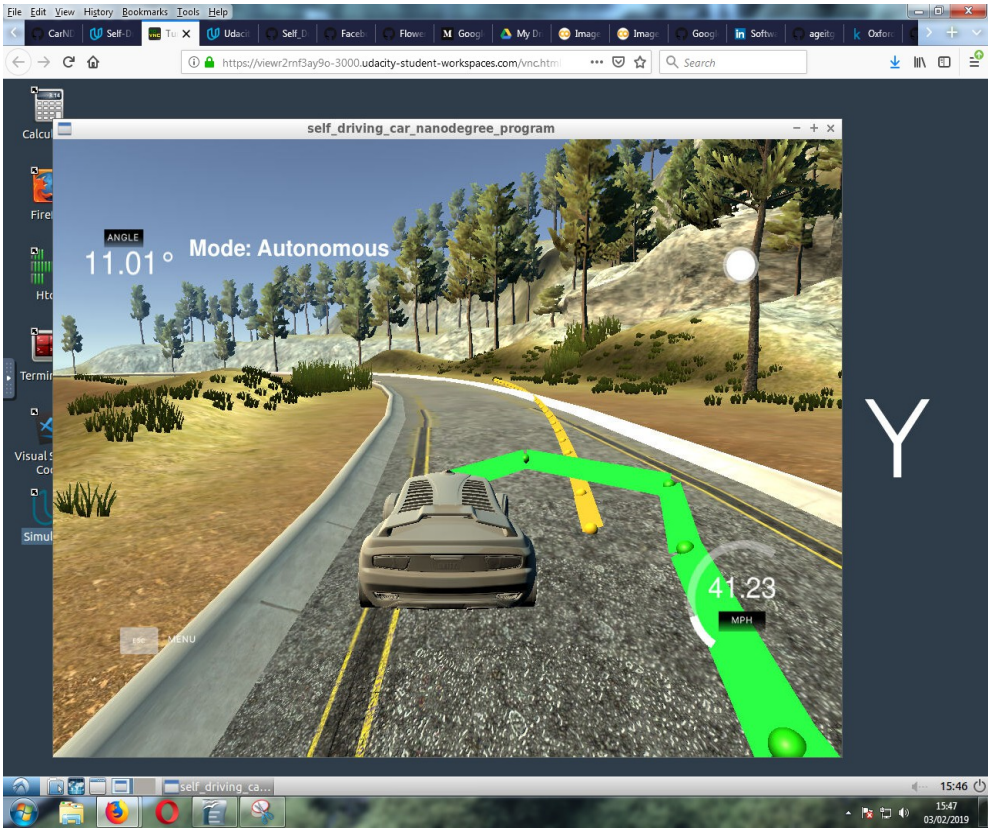
```
////////////////
 // This is the cost variables for below listed here for convience
 //reference state costs
 // I am using Q+ levels then doubleing and then halving them on first set of tests
 double cte_ref_state_cost = 1 ; //1000//4000 // 1000 //2000 in Q+A
 double epsi_ref_state_cost = 1 ;//1000 //4000 //2000 in Q+A
 double v_ref_state_cots = 1 ; //1 in Q+A
 //Acuators Cost
 double delta_cost = 1 ;//50 in Q+A
 double a_cost = 1 ; //50 in Q+A
 //Acuators Cost for time ahead t + 1
 double delta_plus_cost = 1 ;//250000 // 200 in Q+A
 double a_plus_cost = 1 ;//5000 // 10 in Q+A

 for(int i = 0; i < N ; i++)
 {//2000 below is high weight
```



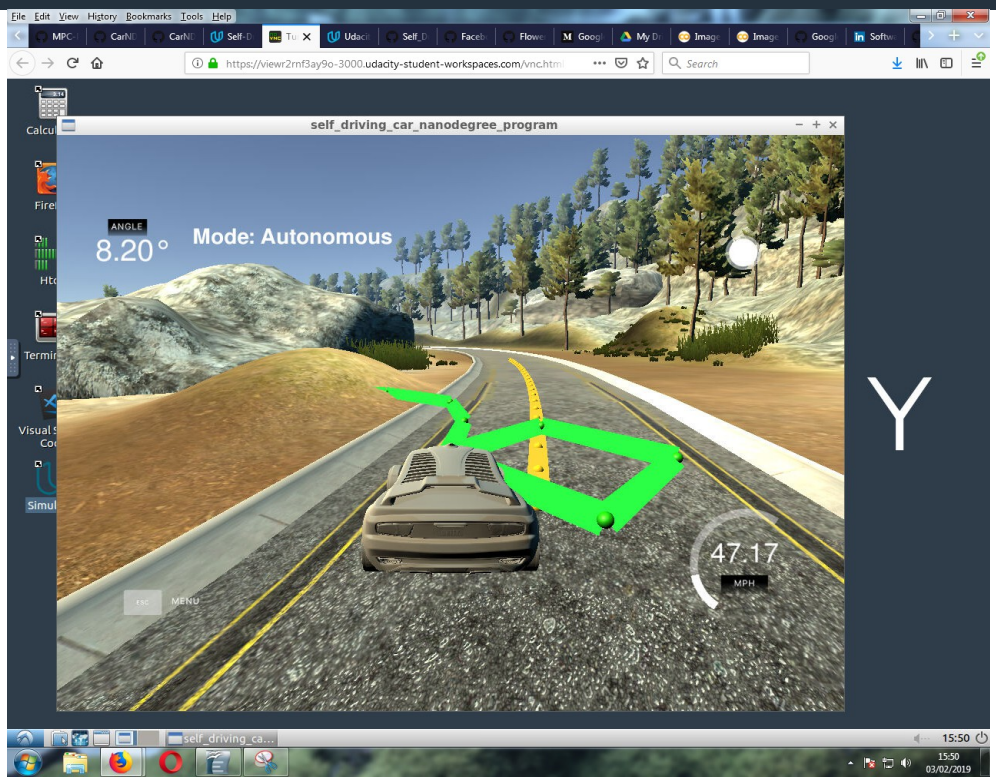
Comment: great for two seconds then.....as soon as steering needed to change the above happened. Crash an instant after screenshot.

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 1 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10 ;//1//250000 // 200 in Q+A
double a_plus_cost = 10 ;//1//5000 // 10 in Q+A
```



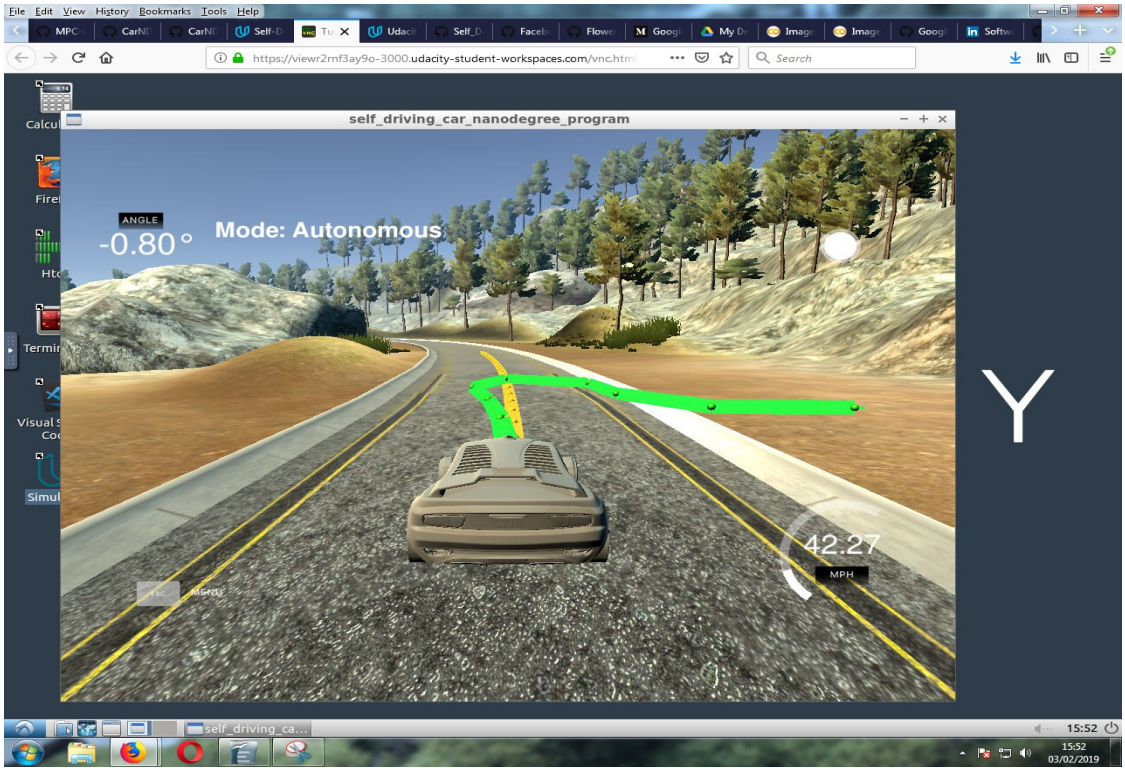
Comment:
Same as last one....

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10 ;//1//250000 // 200 in Q+A
double a_plus_cost = 1 ;//1//5000 // 10 in Q+A
```



Comment:
Even worse than previous two similar problem after couple of seconds.

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 10 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10 ;//1//250000 // 200 in Q+A
double a_plus_cost = 1 ;//1//5000 // 10 in Q+A
```

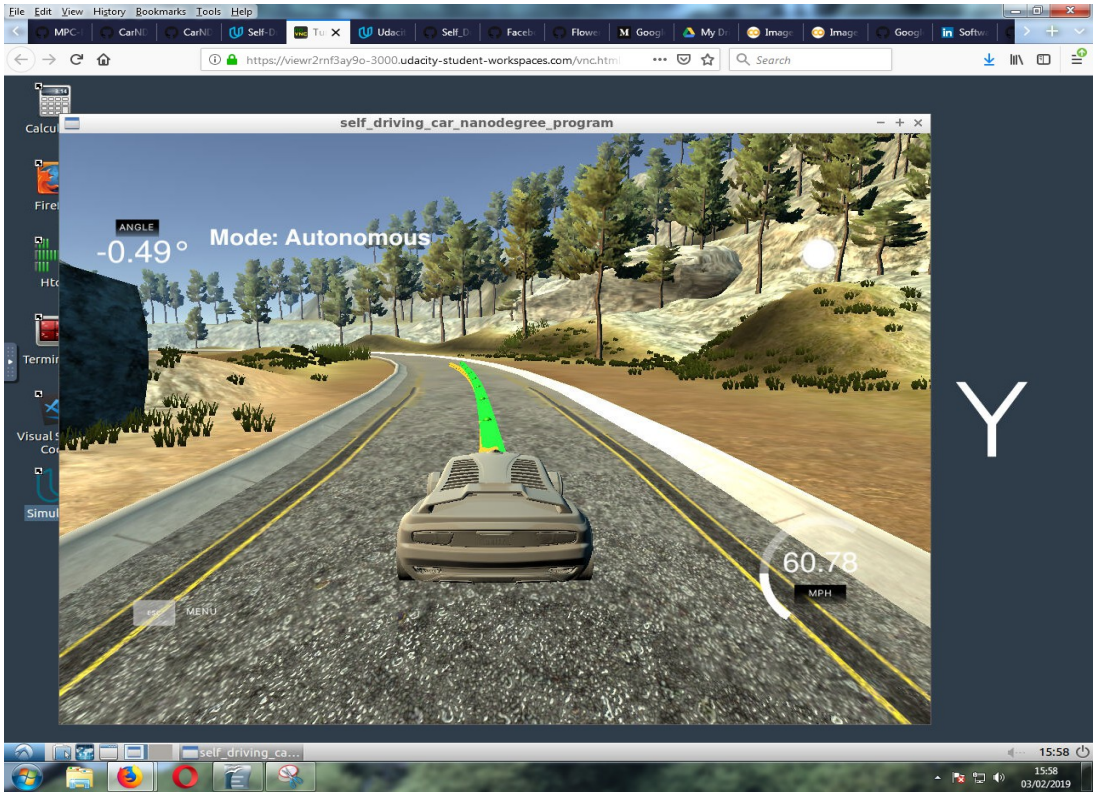


Comment:
Same as previous 3

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 100 ;//1//250000 // 200 in Q+A
double a_plus_cost = 10 ;//1//5000 // 10 in Q+A
```

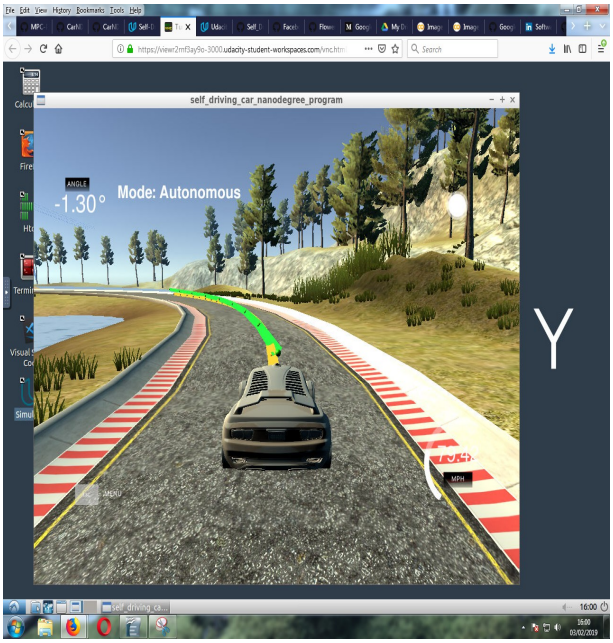
Comment:
Big improvement over last few but oversteering and too fast

```
// This is the cost variables for below listed here for convience
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 1000 ;//1//250000 // 200 in Q+A
double a_plus_cost = 100 ;//1//5000 // 10 in Q+A
```



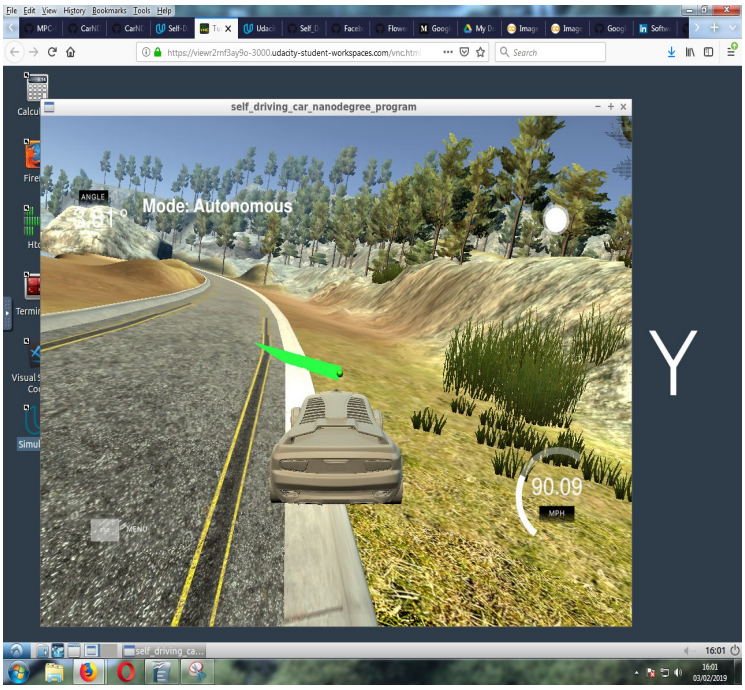
Comment: Not bad but crashed

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000 ;//1//250000 // 200 in Q+A
double a_plus_cost = 100 ;//1//5000 // 10 in Q+A
```

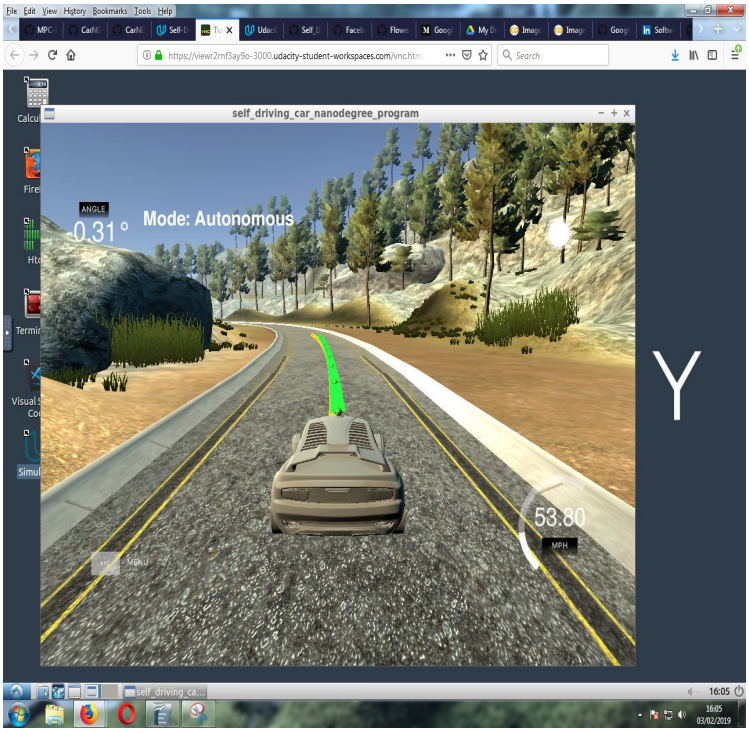


Comment: Not bad but crashed.

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 100000;//1//250000 // 200 in Q+A
double a_plus_cost = 100 ;//1//5000 // 10 in Q+A
```



```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 10 ;//50 in Q+A
double a_cost = 1 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 100000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000 ;//1//5000 // 10 in Q+A
```



```
//////////////////
// This is the cost variables for below listed here for convience
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 100 ;//50 in Q+A
double a_cost = 10 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 100000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000 ;//1//5000 // 10 in Q+A
```

not the worst.....

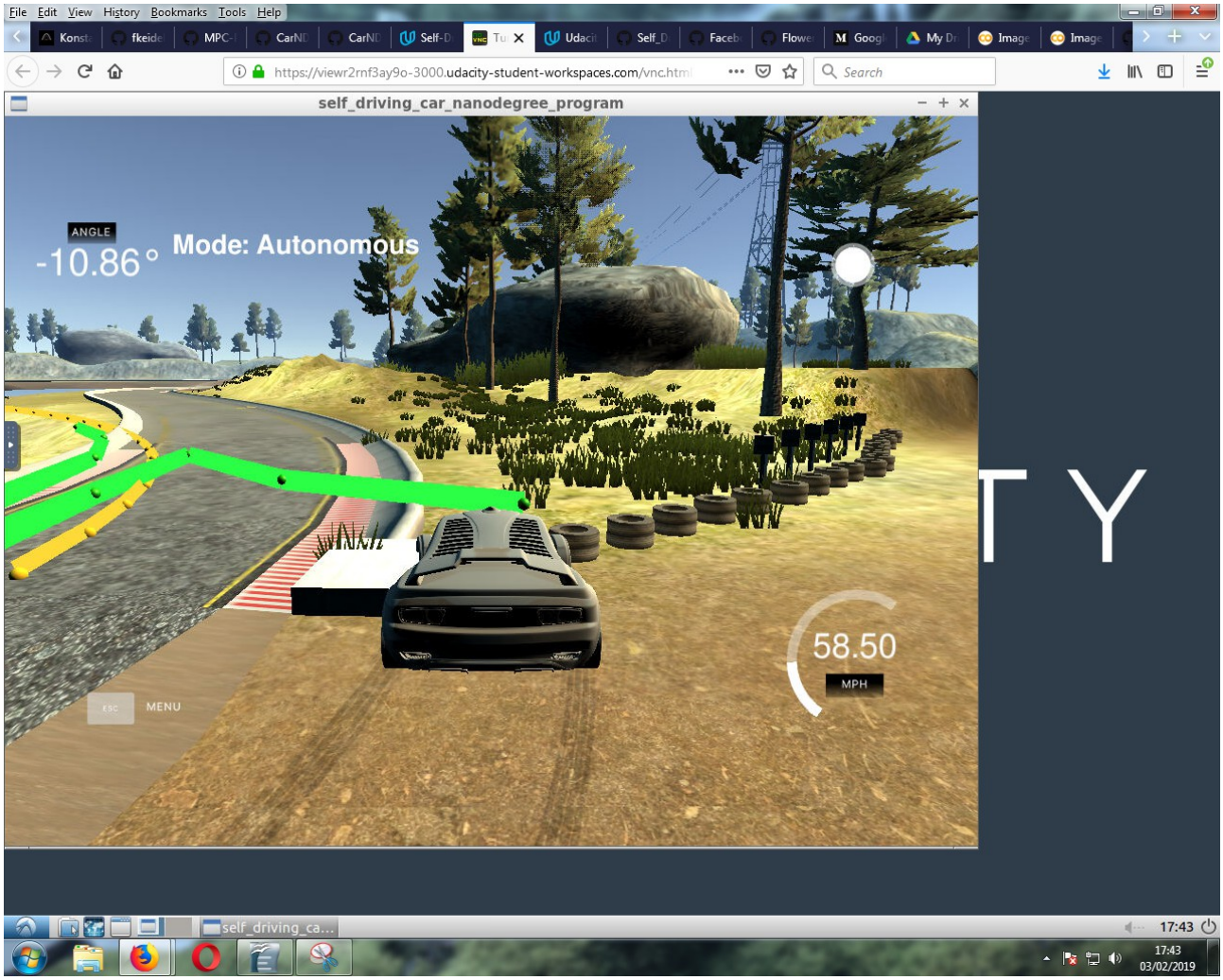
```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 100 ;//50 in Q+A
double a_cost = 10 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 100 ;//1//5000 // 10 in Q+A
```

started well but...

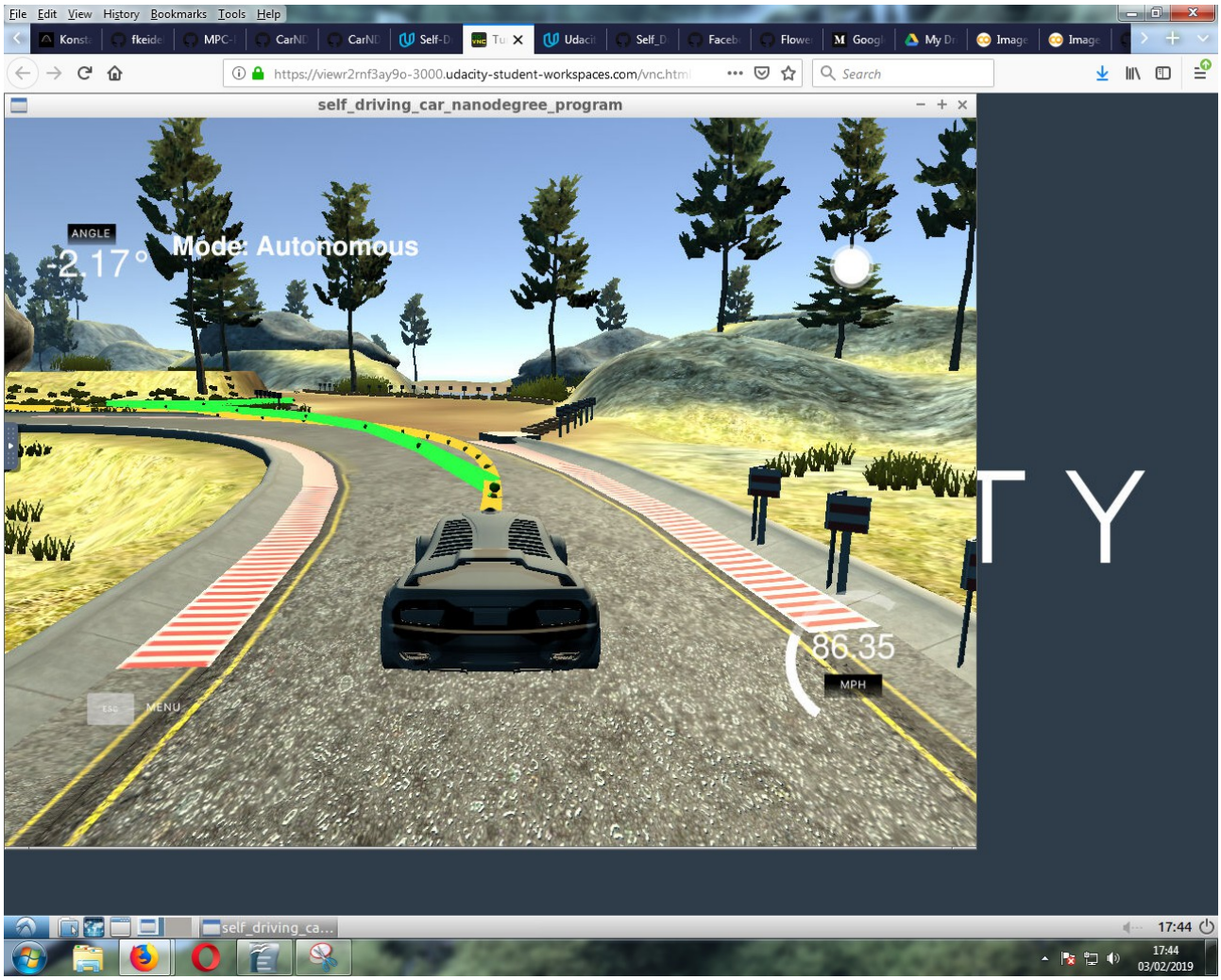
```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 1000 ;//50 in Q+A
double a_cost = 10 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 10 ;//1//5000 // 10 in Q+A
```

Comment:
started well then I think it was going to fast

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 1000 ;//50 in Q+A
double a_cost = 10 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 100;//1//5000 // 10 in Q+A
```

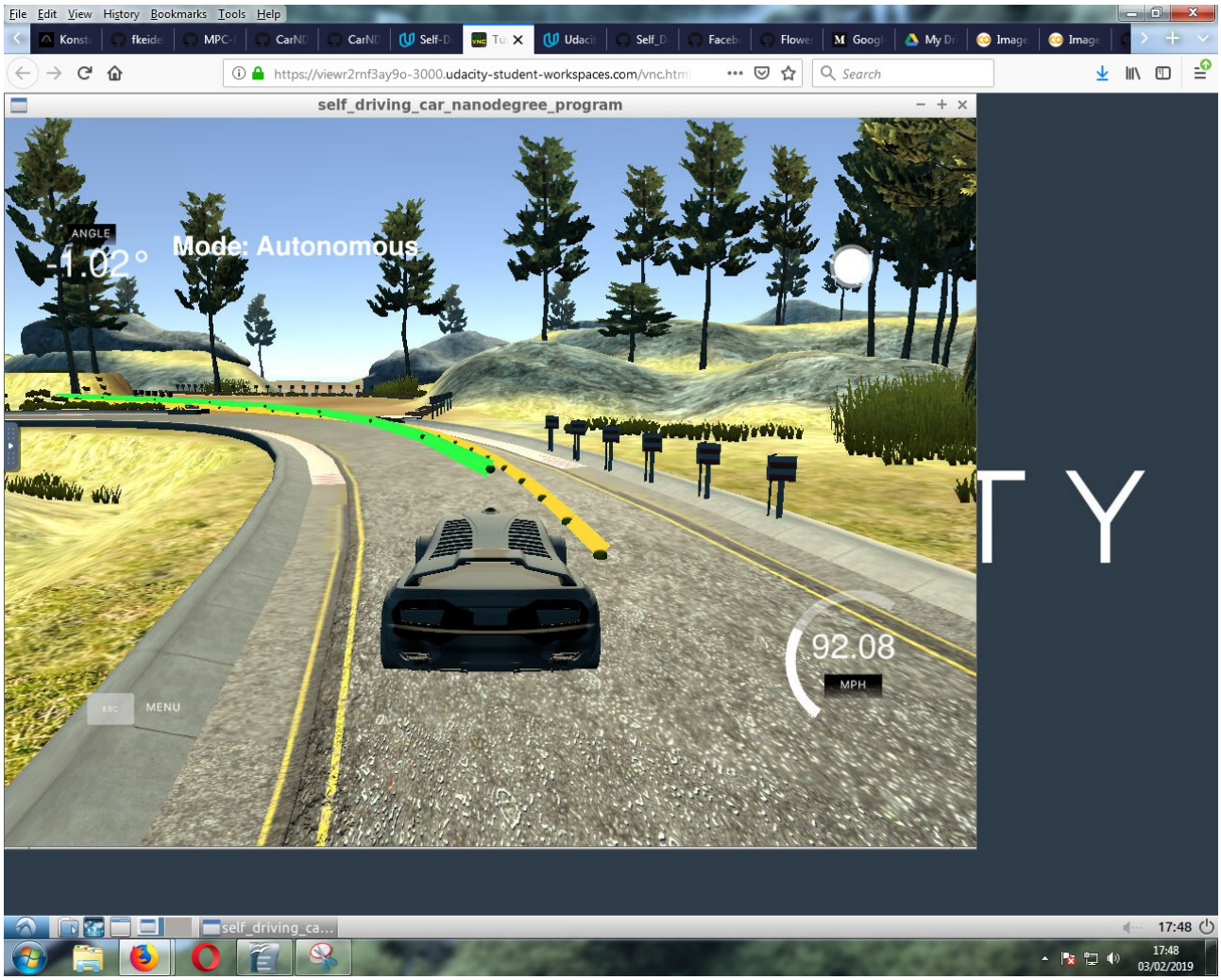


Comment:
Recent attempts are trying to go around this corner too fast 80mph +


File Edit View History Bookmarks Tools Help
https://viewr2rnf3ay9o-3000.udacity-student-workspaces.com/vnc.html
Search
self_driving_car_nanodegree_program
ANGLE
-2.17°
Mode: Autonomous
86.35
MPH
MENU
self_driving_ca...
17:44
17:44
03/02/2019

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 1000 ;//50 in Q+A
double a_cost = 10 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 10;//1//5000 // 10 in Q+A
```

Comment:
making now and future actuator costs equals to see if I can reduce speed around corner



it actually went into corner faster.....

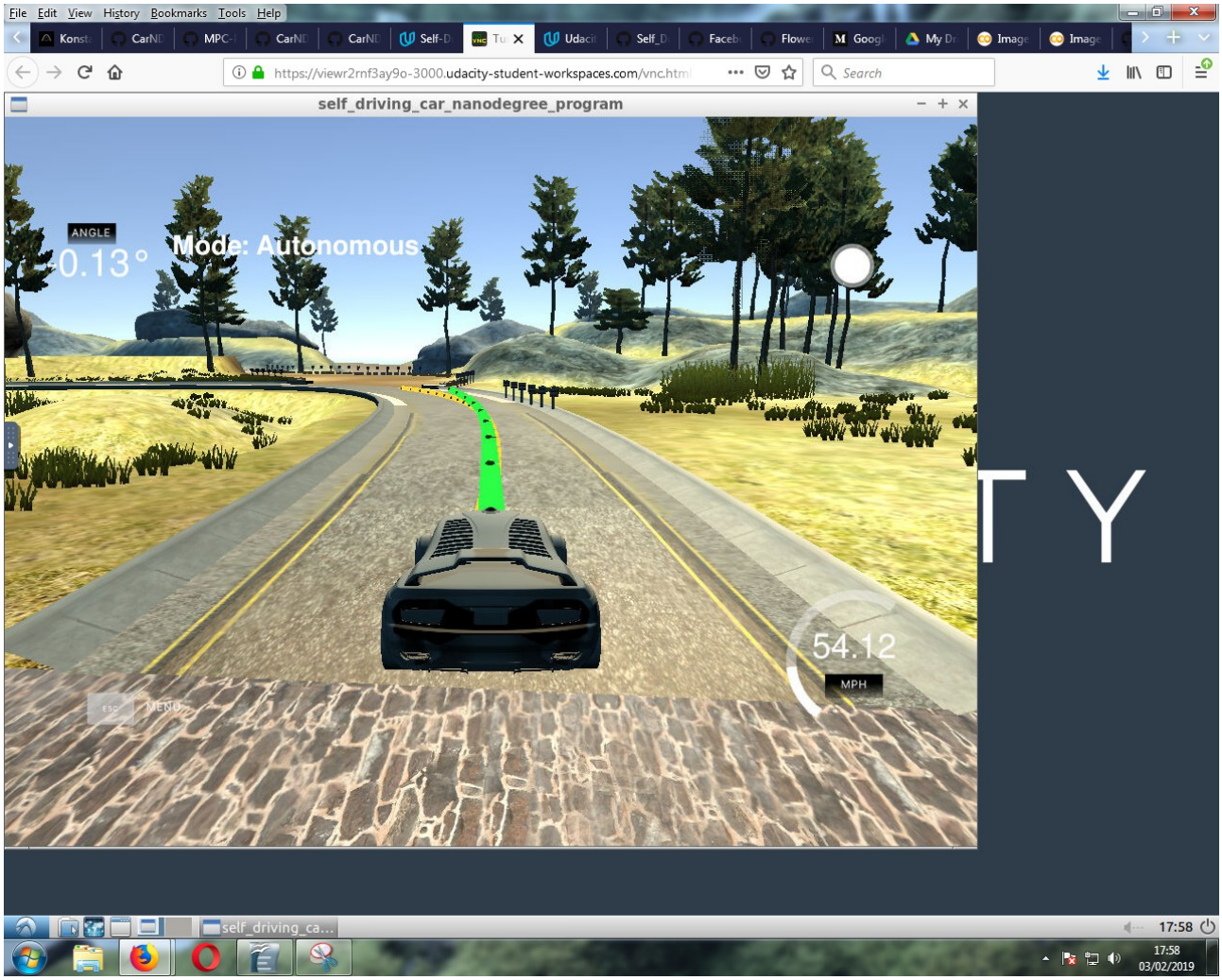
```
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 10;//1//5000 // 10 in Q+A

for(int i = 0; i < N ; i++)
```

Comment:
back to 50 and 50

note speed into corner better but crashed on second attempt on this corner went around once

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 10000;//1//5000 // 10 in Q+A
```

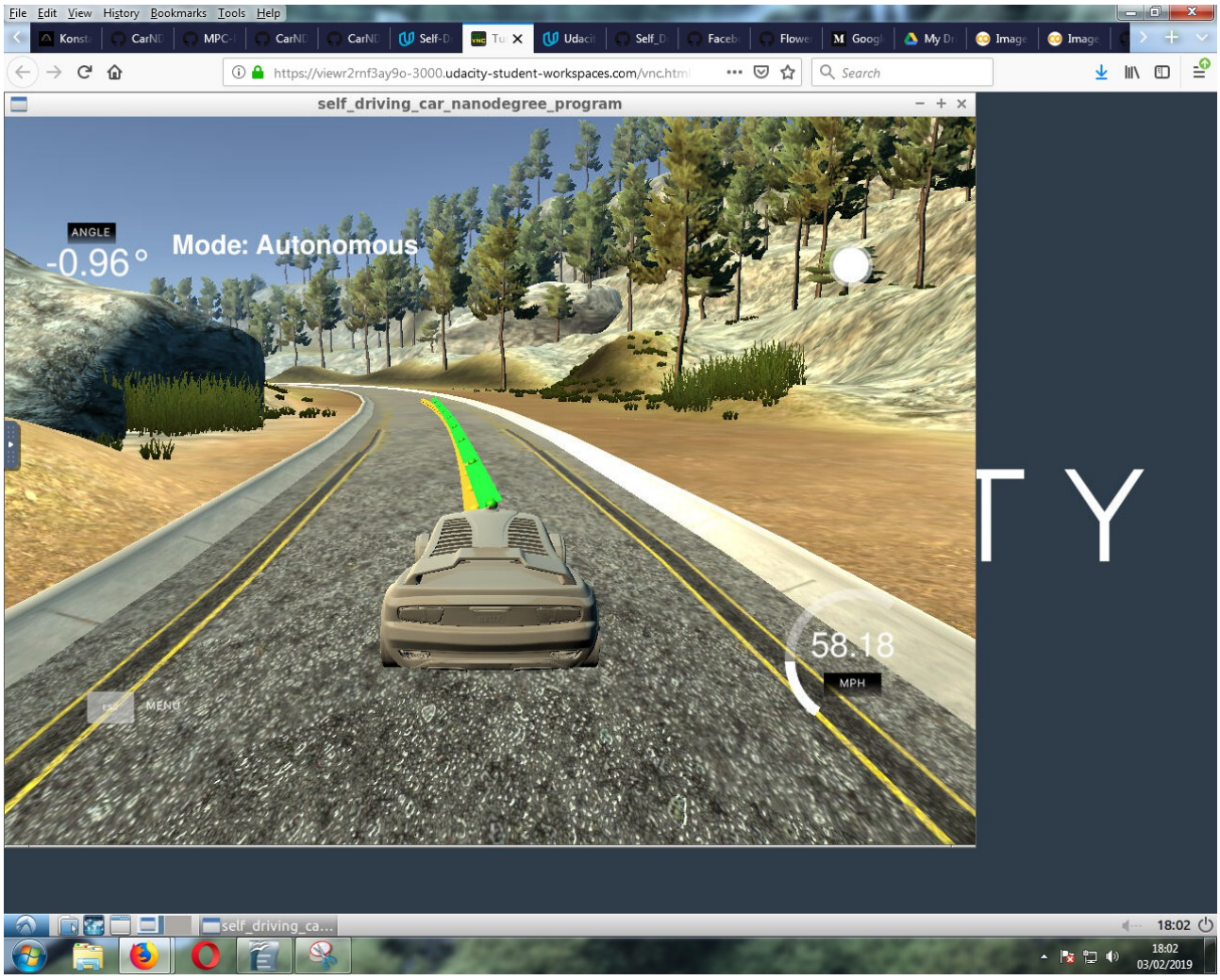


Note speed into this corner is about good not too fast.

best so far not too fast and close to centre line will do many laps

```
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 50 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000;//1//5000 // 10 in Q+A
```

taking 10 off a_plus_cost to see what happens

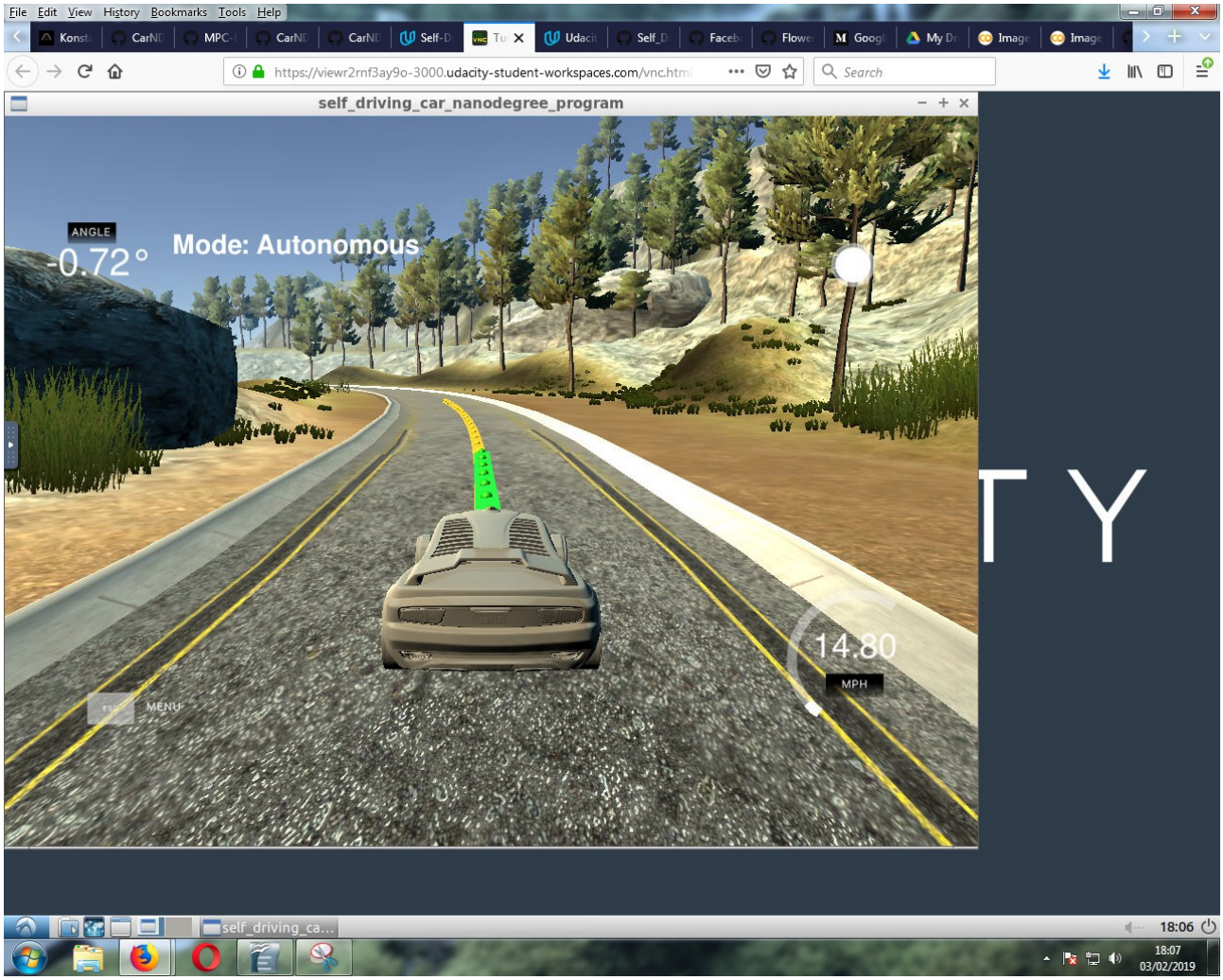


Comment: This green line only does this for a split second then continues fine note speed into corner OK these setting work despite this moment.

almost good enough touches red kerb

```
//reference state costs
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 500 ;//50 in Q+A
double a_cost = 500 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000;//1//5000 // 10 in Q+A
```

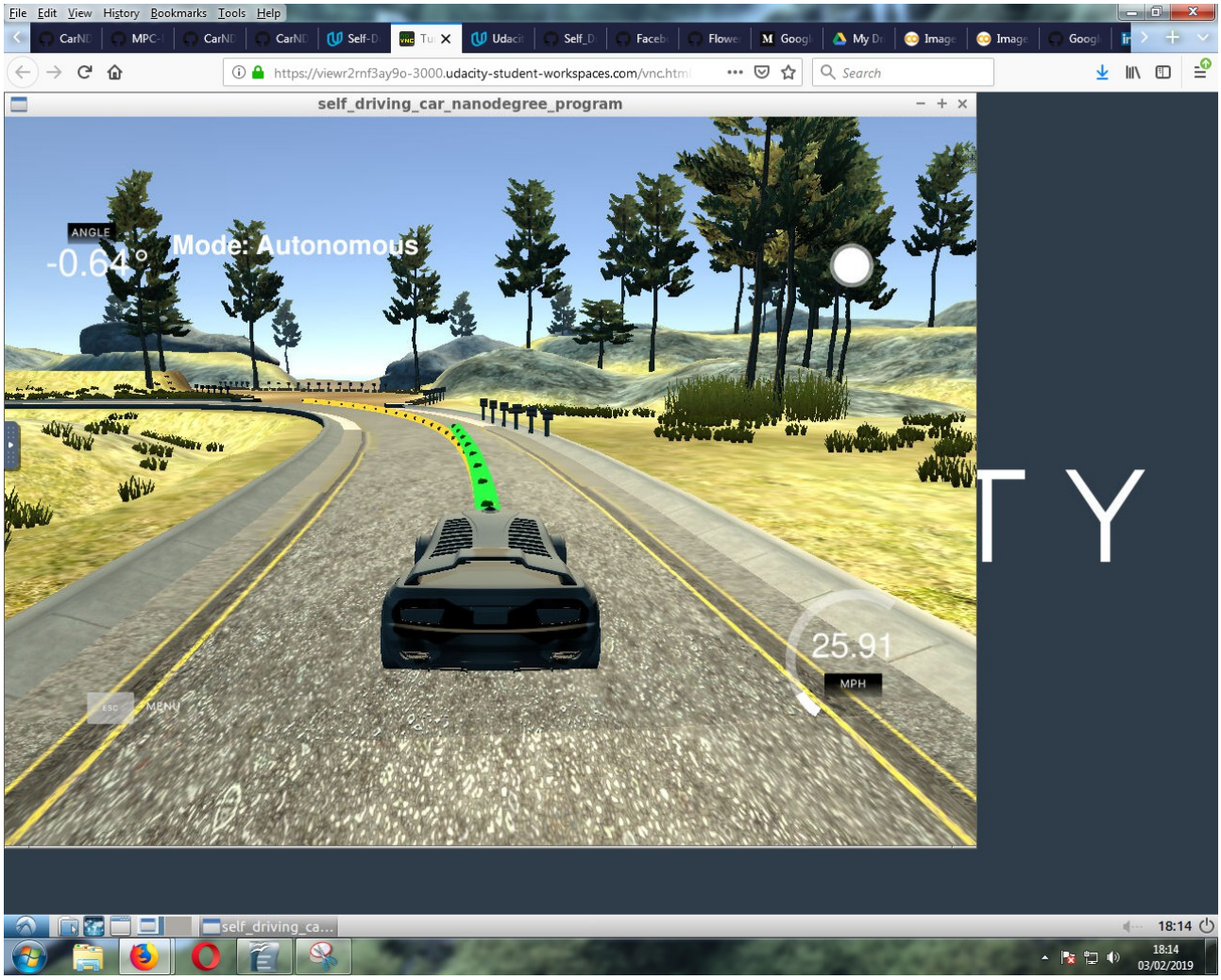
changing actuators to see...result is very slow but very accurate



Note 15 mph into corner 40 to 50 seems optimum.

Near perfect centre line but very slow will try faster

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 500 ;//50 in Q+A
double a_cost = 250 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000;//1//5000 // 10 in Q+A
```



near perfect centre line with 25mph....nearly good enough to submit

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 250 ;//50 in Q+A
double a_cost = 250 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000;//1//5000 // 10 in Q+A
```

Comment: Quite similar to last one


File Edit View History Bookmarks Tools Help
https://viewr2rnf3ay9o-3000.udacity-student-workspaces.com/vnc.html
Search
self_driving_car_nanodegree_program
ANGLE
5.87°
Mode: Autonomous
25.58
MPH
MENU
self_driving_ca...
18:18
18:18
03/02/2019

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 250 ;//50 in Q+A
double a_cost = 125 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 1000;//1//5000 // 10 in Q+A
```

hoping this will go faster....yes it does note speed into corner.


File Edit View History Bookmarks Tools Help
https://viewr2rnf3ay9o-3000.udacity-student-workspaces.com/vnc.html
self_driving_car_nanodegree_program
ANGLE
-5.78°
Mode: Autonomous
37.68
MPH
MENU
self_driving_ca...
18:21
18:21
03/02/2019

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 250 ;//50 in Q+A
double a_cost = 125 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 125; //1000;//1//5000 // 10 in Q+A
```

Comment:
Experimentating making a_cost and a_plus_cost the same

Comment:
a little wobble but a little faster and better overall


File Edit View History Bookmarks Tools Help
https://viewr2rnf3ay9o-3000.udacity-student-workspaces.com/vnc.html
Search
self_driving_car_nanodegree_program
ANGLE
6.47°
Mode: Autonomous
40.09
MPH
ESC
MENU
self_driving_ca...
18:26
18:26
03/02/2019

```
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 2500 ;//50 in Q+A
double a_cost = 125 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 125; //1000;//1//5000 // 10 in Q+A
```

Comment: put up delta cost out of curiosity

Comment: good speed good steering

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 2500 ;//50 in Q+A
double a_cost = 50 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 125; //1000;//1//5000 // 10 in Q+A

for(int i = 0; i < N ; i++)
```

Comment: Faster but goes onto red lines

Crashed on lap 3, slightly strange that it did 2 laps OK but crashed on third.

```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 2500 ;//50 in Q+A
double a_cost = 75 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 125; //1000;//1//5000 // 10 in Q+A
```

Comment: good speed, steering and just short of red and white kerbs probably the faster/best

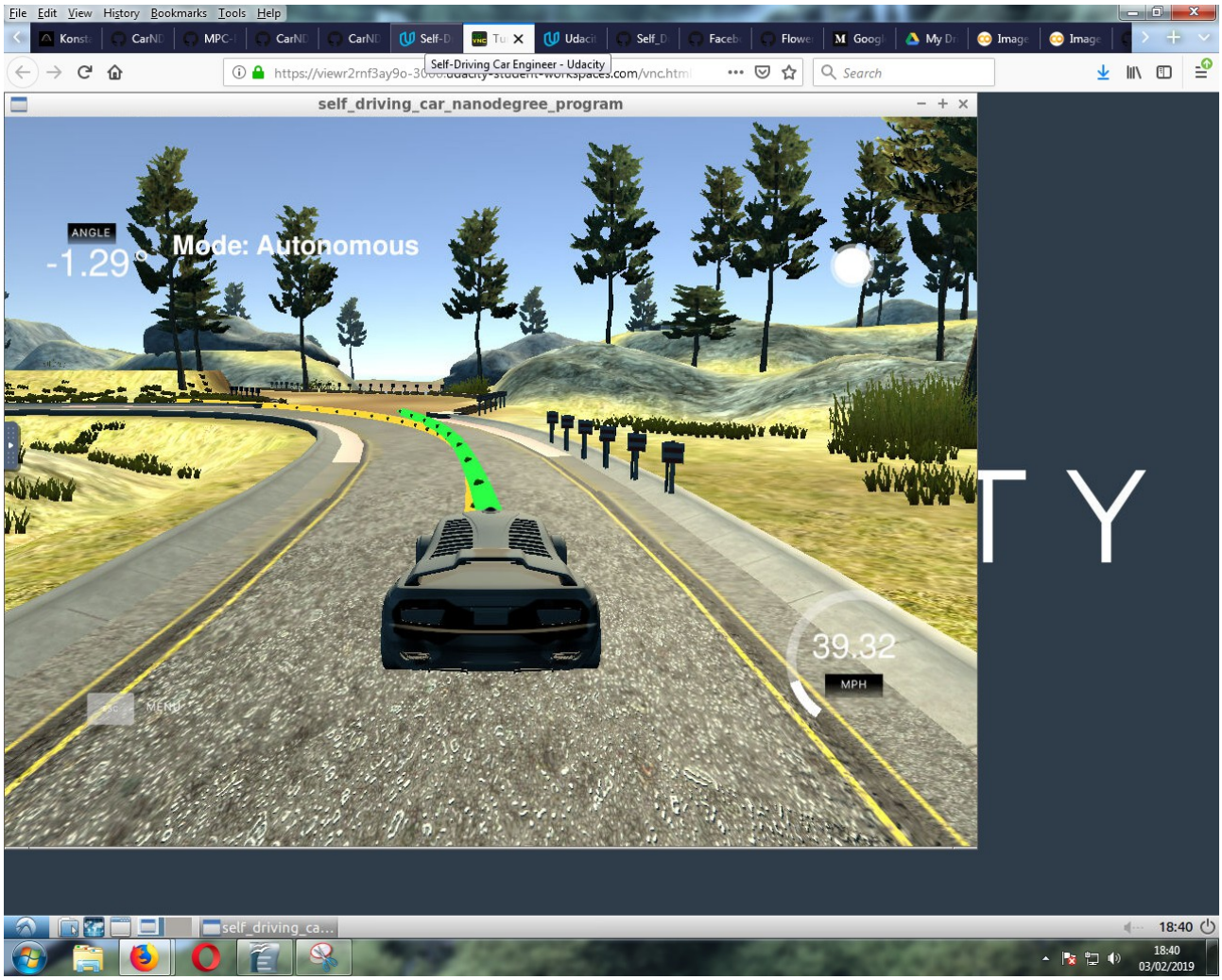

File Edit View History Bookmarks Tools Help
https://viewr2rnf3ay9o-3000.udacity-student-workspaces.com/vnc.html
Search
self_driving_car_nanodegree_program
ANGLE
3.53°
Mode: Autonomous
52.58
MPH
MENU
self_driving_ca...
18:35
18:35
03/02/2019

Reducing level a little to stop it going onto red kerbs

```
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 2500 ;//50 in Q+A
double a_cost = 100 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 125; //1000;//1//5000 // 10 in Q+A
```

Comment: a little slower and safer than previous

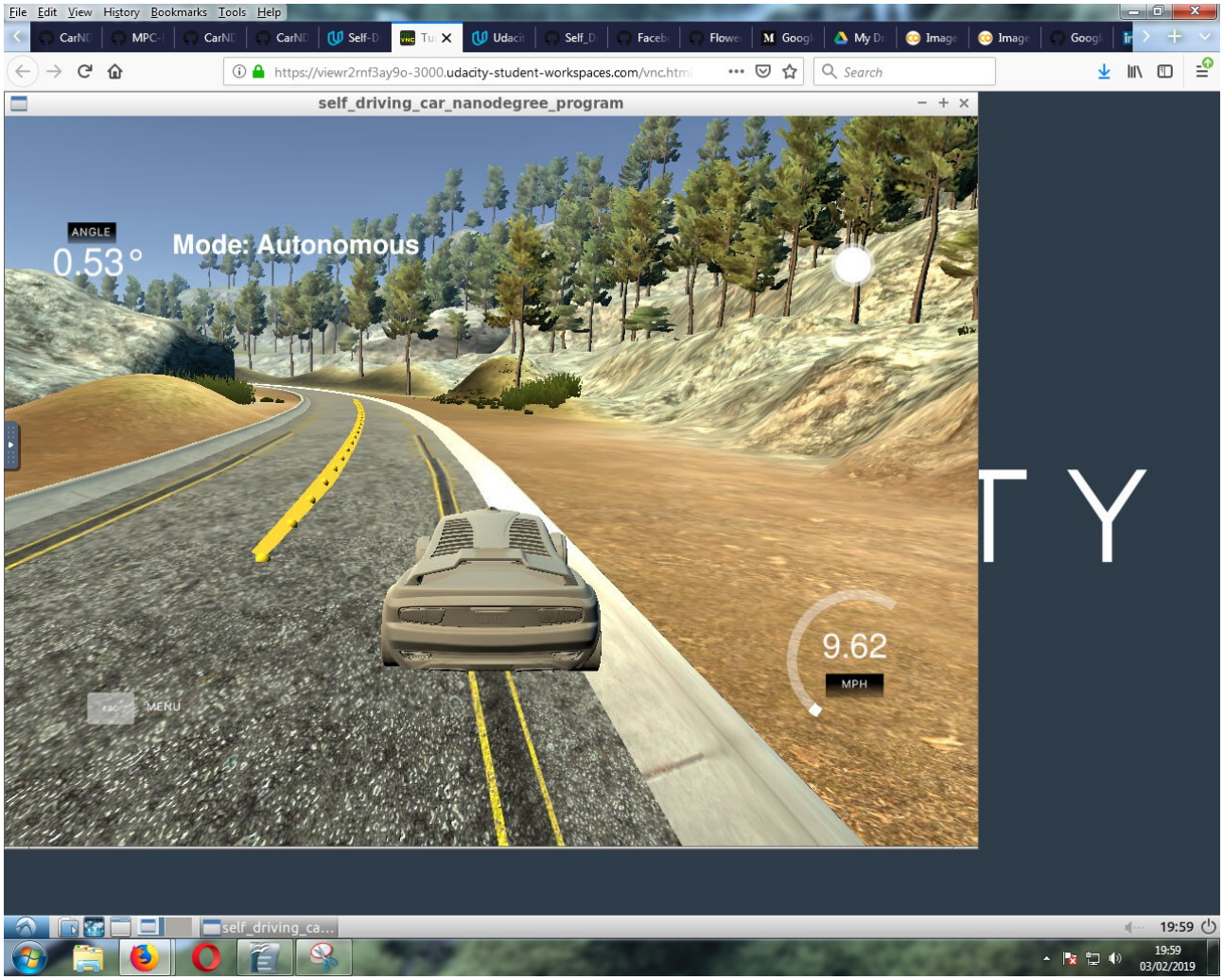
```
// I am using Q+ levels then doubleing and then halving them on first set of tests
double cte_ref_state_cost = 1000 ; //1000//4000 // 1000 //2000 in Q+A
double epsi_ref_state_cost = 1000 ;//1000 //4000 //2000 in Q+A
double v_ref_state_cots = 1 ; //1 in Q+A
//Acuators Cost
double delta_cost = 2500 ;//50 in Q+A
double a_cost = 150 ; //50 in Q+A
//Acuators Cost for time ahead t + 1
double delta_plus_cost = 10000;//1//250000 // 200 in Q+A
double a_plus_cost = 150; //1000;//1//5000 // 10 in Q+A
```



Comment:
Does not touch red and white here when most of them did so this should pass where others may not...submitting this one.

Changing N and dt to see if it helps as suggested in Rubric

```
size_t N = 5;//10;
double dt = 0.05; //0.1
```



Comment: quite odd no green line just drifted to side as road turned

```
24
25
26  size_t N = 20;//10;
27  double dt = 0.1; //0.1
28
```

Comment : note the length of green line

Comment: Started OK but got erratic

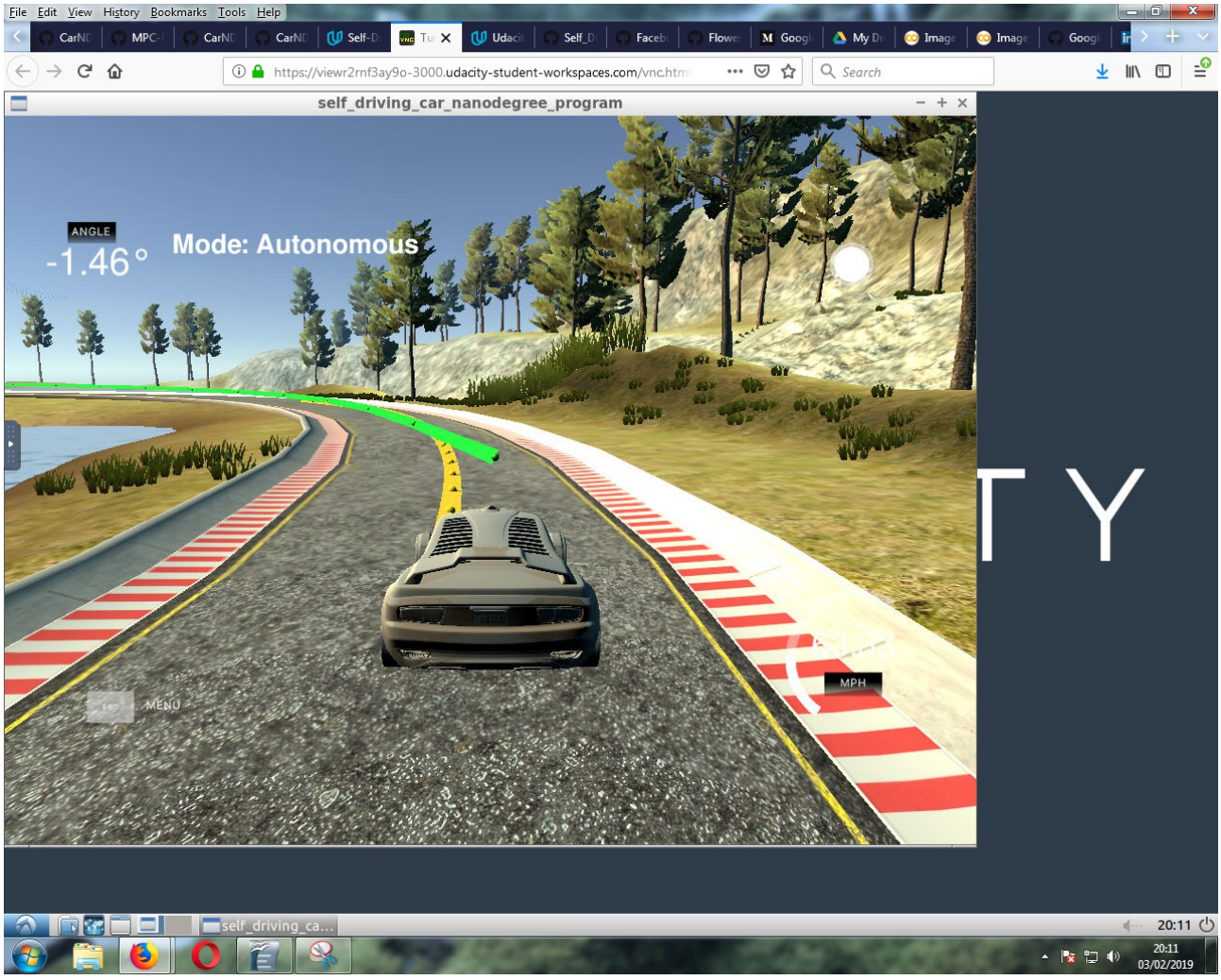
```
23
24
25
26  size_t N = 20;//10;
27  double dt = 0.05; //0.1
28
29  // This value assumes the model presented in th
```

Comment: Started ok

then got very erratic didn't make the bridge


File Edit View History Bookmarks Tools Help
https://viewr2rnf3ay9o-3000.udacity-student-workspaces.com/vnc.html
Search
self_driving_car_nanodegree_program
ANGLE
-2.15°
Mode: Autonomous
MPH
MENU
self_driving_ca...
20:07
20:07
03/02/2019

```
25
26  size_t N = 20;//10;
27  double dt = 0.2; //0.1
28
29  // This value assumes the model presented in the classroo
30  //
```

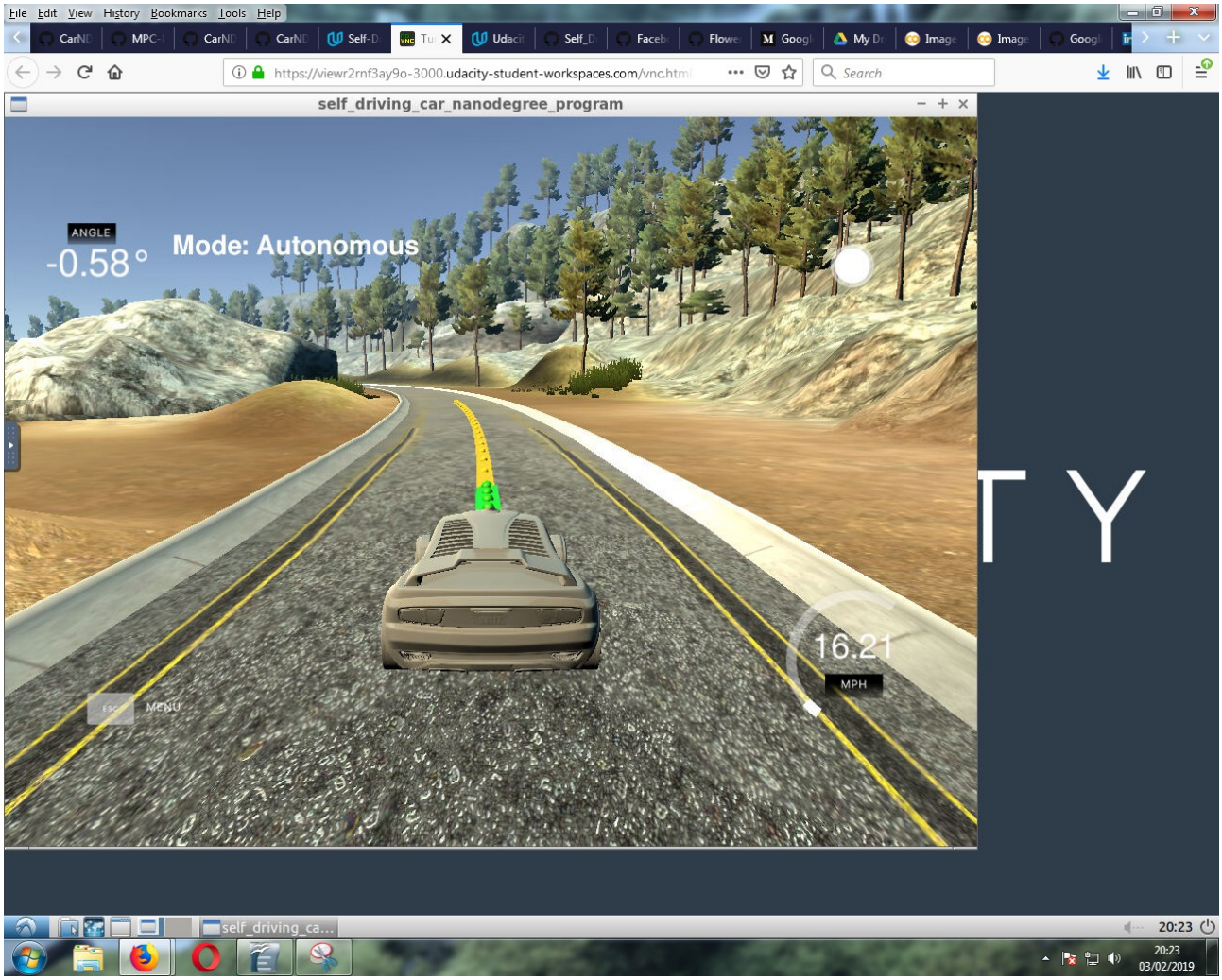


Comment: Crash !

```
24
25
26  size_t N = 10;//10;
27  double dt = 0.2; //0.1
28
```

Comment: Very slightly worse then N = 10 dt = 0.1 touched red and white line here

```
26  size_t N = 10;//10;
27  double dt = 0.05; //0.1
28
```



Comment note the short green line. This one is good accuracy just slow.

Conclusion: N 10 and dt 0.1 are the best settings for speed and accuracy.