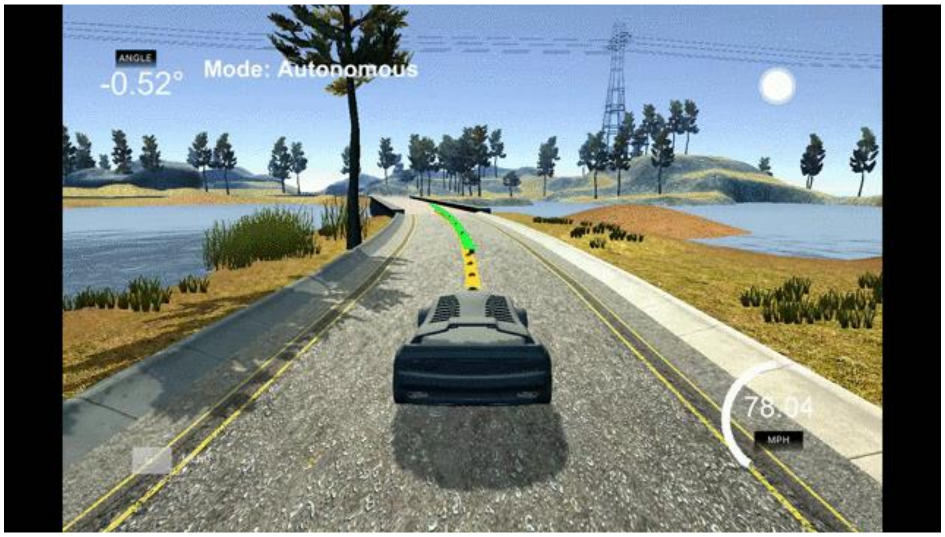
In this project you'll implement Model Predictive Control to drive the car around the track. This time however you're not given the cross track error, you'll have to calculate that yourself! Additionally, there's a 100 millisecond latency between actuations commands on top of the connection latency.

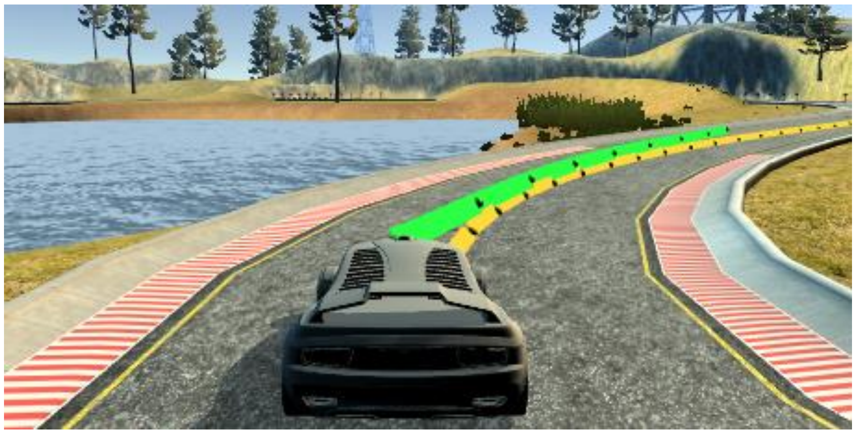
Setup

1. The **project repo**.
2. The simulator. Downloadable from the **releases page**. Make sure to get the one with MPC!
3. The **rubric**.



The yellow is a polynomial fitted to waypoints and the green line represents the *x* and *y* coordinates of the MPC trajectory.

Tips and Tricks for the MPC Project



Displaying the MPC trajectory path in green, and the polynomial fitted reference path in yellow.

Simulator Misc

- A singular unity unit in the simulator in equivalent to 1 meter.
- For details on the data sent back from the server, read **this file**.

ψ Updates

In the classroom we've referred to the ψ update equation as:

$\psi_{t+1}=\psi_t+L_f v_t*\delta_t*dt$

Note if δ is positive we rotate counter-clockwise, or turn left. In the simulator however, a positive value implies a right turn and a negative value implies a left turn. Two possible ways to get around this are:

1. Change the update equation to $\psi_{t+1} = \psi_t - \frac{v_t}{L_f} * \delta_t * dt$
2. Leave the update equation as is and multiply the steering value by -1 before sending it back to the server.

Visualization

When working on the MPC project it helps to visualize both your reference path and the MPC trajectory path.

You can display these connected point paths in the simulator by sending a list of optional x and y values to the `mpc_x`,`mpc_y`, `next_x`, and `next_y` fields in the C++ main script. If these fields are left untouched then simply no path will be displayed.
The `mpc_x` and `mpc_y` variables display a line projection in green. The `next_x` and `next_y` variables display a line projection in yellow. You can display these both at the same time, as seen in the image above.
These (x,y) points are displayed in reference to the vehicle's coordinate system. Recall that the x axis always points in the direction of the car's heading and the y axis points to the left of the car. So if you wanted to display a point 10 units directly in front of the car, you could set `next_x = {10.0}` and `next_y = {0.0}`.
Remember that the server returns waypoints using the map's coordinate system, which is different than the car's coordinate system. Transforming these waypoints will make it easier to both display them and to calculate the CTE and Epsi values for the model predictive controller.

IPOPT Installation on Mac

When executing `brew install ipopt`, some Mac users have experienced the following error:

```
Listening to port 4567
Connected!!!
mpc(4561,0x7ffff1eed3c0) malloc: *** error for object 0x7f911e007600: incorrect checksum for
freed object - object was probably modified after being freed.
*** set a breakpoint in malloc_error_break to debug
```

This error has been resolved by updrading ipopt with `brew upgrade ipopt --with-openblas`per this **forum post**.

Project Submission

1. Clone/fork the project's template files from the **project repository**. (Note: Please do not submit your project as a pull request against our repo!)
2. Choose your state, input(s), dynamics, constraints and implement MPC.
3. Test your solution on basic examples.
4. Test your solution on the simulator!
5. When the vehicle is able to drive successfully around the track, submit! ***Remember to include a separate file in .txt, .md, .html, or .pdf format addressing the questions in the rubric.***
6. Try to see how fast you get the vehicle to **SAFELY** go!