Design PID (proportional / integral / differential) Controller

The purpose of this project was :

1. to "build a PID controller. Then
2. tune the PID hyperparameters by applying general processing flow in lectures. Finallly,
3. test the PID solution using the car simulator.

Note:

1. The simulator provides cross-track error (CTE), speed, and steering angle data via local websocket.
2. The PID controller will send to the simulator the steering and throttle commands to drive the car reliably around the simulator track.

The effect each of the P, I, D components had in your implementation.

A PID controller is a control loop feedback controller which is widely used in different control systems. The PID stands for (Proportional, Integral, Derivative)

The Error is an input variable for the controller. In our case,

```
* cte = desired_state - measured_state
```

The P stands for "proportional" component.

It causes the car to steer proportional (and opposite) to the car's CTE (cross track error) which is the distance from the lane center.

- If the car is far to the right it steers hard to the left,
- If it's slightly to the left it steers slightly to the right.
- If there is no eror, there is no corrective response.

The D stands for "differential" component.

- It counteracts the P component's tendency to ring and overshoot the center line.
- It is effectively seeking to reduce the effect of the error (CTE) by exerting a control influence generated by the rate of error change.

The I stands for "integral" component.

- It accounts for past error values and integrates them over time to produce the I term.
- The integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error.
- When the error is eliminated, the integral term will cease to grow.
- This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.
- It counteracts a bias in the CTE such as a steering drift (as in the Control unit lessons),

CRITERIA #1 : The PID procedure follows what was taught in the lessons.

hyperparameter tuning/optimization. However, the base algorithm should follow what's presented in the lessons.

CRITERIA #2 : Describe the effect each of the P, I, D components had in your implementation.

The description is in "The effect each of the P, I, D components had in your implementation."

CRITERIA #3 : Describe how the final hyperparameters were chosen.

The final PID is (0.1, 0.001, 1.0) and it is manually selected. Using the concept TWIDDLE() we learned, I started out with :

1. PID = (1,0,0) -- car drive fast wildly left and right.
2. PID = (0.1, 0, 0) -- make car move slow, it is easier to control
3. PID = (0.1, 0. 1.0) -- run ~47,000 loop without any problem but total error increase
4. PID = (0.1, 1.0. 1.0) -- crash after a little run
5. PID = (0.1, 0.1. 1.0) -- crash after a little run (~2,700)
6. PID = (0.1, 0.01. 1.0) -- stable.
7. PID = (0.1, 0.001. 1.0) -- very stable

With this foundation, I continue to tune and decide the default PID value is (0.1, 0.001, 1.0)

CRITERIA #4 : The vehicle must successfully drive a lap around the track.

The vehicle run over night without rolling over.