1. *In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?*

In the simulation, I can see that the car is doing exactly what I programmed it to do: choose a random direction (None, forward, left, right) and go in that direction if the "light" is green. Eventually, it does make it to the target location, but since the driving is random, it can take a while and I have to hope to get lucky.

2. *Justify why you picked these set of states, and how they model the agent and its environment.*

The states I picked were: next_waypoint, traffic light status, oncoming traffic, and traffic to the left. I chose the 'oncoming traffic' because the project description says "On a green light, you can turn left only if there is no oncoming traffic at the intersection coming straight. On a red light, you can turn right if there is no oncoming traffic turning left or traffic from the left going straight," both of which deal with the status of oncoming traffic. I chose the status of the traffic light because obviously, you can't do anything if the light is red. I chose next_waypoint because I wanted to compare what the GPS tells the agent to do and compare that to what the driver should do; if the GPS says go straight for example, a better solution might turn out to be driving to the right. Lastly, I used the status of traffic to the left because the rule "On a red light, you can turn right if there is no oncoming traffic turning left or traffic from the left going straight," requires the agent to yield to traffic from the left. This set of states models the agent and its environment because they include all possible combinations of traffic and lights and the path to the destination, while incentivizing the agent to follow the rules of the laws.

There are other input variables, such as deadline and the status of traffic to the right, that I could have included but I have decided against. I chose to exclude the 'deadline' variable because we are trying to reach the destination as soon as possible; we don't care about how much time is left per say, but more so of where the GPS is telling us to go and the traffic laws. I decided against the status of traffic to the right because traffic laws don't have to worry about traffic to the right at all, just oncoming traffic and traffic to the left.

3. *What changes do you notice in the agent's behavior?*

The agent explores for a little bit until it discovers pretty much all of the 128 possible states. Then, it is pretty confident when using the Q-table and can correctly choose the best action from its current state to reach its destination nearly every single time without breaking any traffic laws.

4. *Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform? Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties?*

	α = 0.0	α = 0.1	α = 0.2	α = 0.3	α = 0.4	α = 0.5	α = 0.6	α = 0.7	α = 0.8	α = 0.9	α = 1.0
Reward	-571.0	3152.5	2714.5	2903.5	2831.0	2833.5	2998.0	2808.0	2846.0	2837.5	2936.0
Wins	0	1	88	97	95	94	94	96	96	97	96

For my Q-learning algorithm, instead of just using the reward for the Q-table, I implemented a discount factor, represented by gamma, which is multiplied by the max Q-value of the possible next states. In addition, I added a learning rate. After a series of test runs from the table above, I decided to go with a learning rate α of 1.0 because it's near 100% success with one of the highest total rewards. Lastly, I

replaced random exploration with optimistic initialization, setting all Q-values to 10 initially and over time every state-action pair will be chosen while in random exploration, that might not necessarily be the case. With these changes, my implementation performs very well, reaching its destination ~96 times out of 100!

Yes, the agent is very good at avoiding penalties and reaching its destination with a ton of reward in an optimal amount of time. In a test simulation, across the last 40 trials, there was a total of 1 error/penalty by the agent; this can be accounted to one state-action pair not being discovered until that trial and it just so happened to be a bad state-action pair.