QUESTION: *Observe what you see with the agent's behavior as it takes random actions. Does the **smartcab** eventually make it to the destination? Are there any other interesting observations to note?*

Since the deadline check is set to false, the learning agent navigates forever within one single trial until it reaches the destination without any training algorithm being involved. Our test car violates traffic rules.

QUESTION: *What states have you identified that are appropriate for modeling the **smartcab** and environment? Why do you believe each of these states to be appropriate for this problem?*

A unique state consists of combinations of NextWaypoint, Light, Left and Oncoming (as shown in my implementation of State.__eq__) and each of such combination defines what action can be taken – traffic rule. Ex. Light is green, no oncomging traffic, then it is ok to make a left turn, any of the premise missing, then taken action might receive negative reward.

OPTIONAL: *How many states in total exist for the **smartcab** in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

State total = light combinations * NextWaypoint combinations * Left combinations * Oncoming combinations

The amount of states in total is smaller than 100 – indicating the state spaces is small, in stochastic exploration, if we are lucky, it only takes less than 100 steps covering all the combinations.

If we take "Deadline" into account, what we might do is that when there are only a few steps left, then the agent can violate traffic rules heading straight to the destination, which means that violating traffic rules receive positive reward instead of negative one. The MDP context should be consistent and static, this kind of change is not allowed

Another excluded factor is input['right'], because having a car on the right has no impact at all for making decision according to eh U.S. Right-of-Way rules.

QUESTION: *What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?*

Our test car stays at the same location more often; the reason is that the QValue for staying at the same location might be higher than other actions. After adding nextwaypoint to part of the state, the car does not stuck on the same place anymore.

QUESTION: *Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?*

Epsilon greedy refers to the chances of a random action, therefore given a specific start and destination, agent will be able to explore multiple routes. http://www.wearepop.com/articles/secret-formula-for-self-learning-computers

there is no epsilon implementation in my code, only gamma (discount rate) and alpha (α – learning rate) are used. As discussed in class material, gamma is something that we can ignore actually, the historical original for this variable comes from economic sector. For alpha – learning rate, the higher the value is the quicker the current state qvalue gets overridden.

I run the program, using different alpha values, 0.1, 0.3, 0.5, after 60 trials for training, run another 39 trials gathering following data

total positive reward and negative reward

alpha=0.1 - total positive reward: 854.0 / total negative reward: -6.0

alpha=0.3 - total positive reward: 848.0 / total negative reward: -4.0

alpha=0.5 - total positive reward: 826.0 / total negative reward: -6.5

the difference is not much, maybe alpha 0.3 is the best fit here.

QUESTION: *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? How would you describe an optimal policy for this problem?*
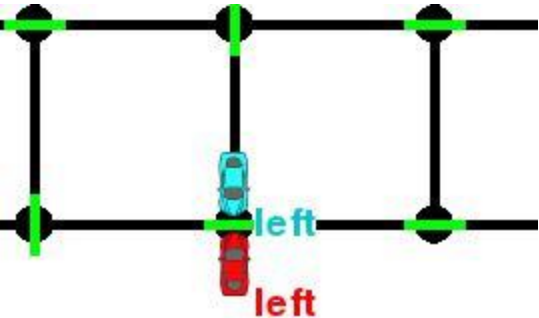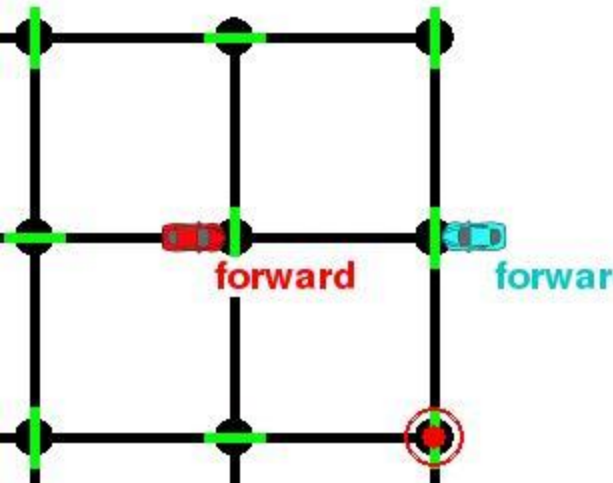
An optimal policy should behave so that no violation of traffic rules should happen; Overall, if we combine the total positive/negative reward + the number of mistakes, the policy is closed to be optimal but not optimal, because every single mistake can cause life

I have added a method called "`printMistakeStats`", when next_waypoint is different from the action taken then it is counted as a mistake, I start recording these mistakes after 60 trials. The method name here is a bit misleading, because the taken action being different from next_waypoint does not indicate violation of traffic rules. Anyway, I notice that most of the received negative rewards are -0.5 which happens when light is red, next_waypoint is forward, and the taken action is either None or right, in such cases, the agent is actually obeying the traffic rule, but get a negative reward. An example is shown below

⊿ ● mistakeSeries	[[[...]], [[...]], [[...]], [[...]], [[...]], [...]]]
⊿ ● [0]	[[72, 12, {...}, 'forward', 'right', False]]
⊿ ⊘ [0]	[72, 12, {'left': None, 'light': 'red', 'oncoming': 'forward', 'right': None}, 'forward', 'right', False]
● [0]	72
● [1]	12
▷ ● [2]	{'left': None, 'light': 'red', 'oncoming': 'forward', 'right': None}
● [3]	'forward'
● [4]	'right'
● [5]	False

at trial 72, step 12, input being `{'left': None, 'light': 'red', 'oncoming': 'forward', 'right': None}`, the next_waypiont being "forward", taken action is "right"

And there are occasions violating traffic rules as well, as shown below

Inputs	Next_waypoint	Action	Reward	
{'left': None, 'light': 'red', 'oncoming': 'left', 'right': None}	Forward	left	-0.5	
{'left': None, 'light': 'red', 'oncoming': 'left', 'right': None}	Left	forward	-1	

This indicates that mistakes happen when "oncoming" contains value other than None. Our agent does not land on these states that often because it will require two cars locating at specific position; and therefore the qvalue for these states sometimes are not converged yet.

On the other hand, I have created a class named "QTableSnabShotPerTrial", it makes a deep copy of all states between trial 60 and 99. Q-values are normalized within same trial, then I compare how much the total q-value changes between trials, don't know if it is something worth to do, but the changes in percentage is between -15% to 15%, many of them are lower than 7%. I guess when the change percentage is low, it means it is converged