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?*

The smartcab reaches the destination but it takes a long time and it is not the optimal path. We can also see that the car several times loses points in the top left information in the game window.

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?*

I considered that the “self.next_waypoint” , the “inputs[‘light’]” , the “inputs['oncoming']” and “inputs['left']” should be used. I need to know my next step, the light, if there is a car on coming or on coming from left, from the rules you can exclude cars coming from the right. The “deadline” is useless since it will not help arriving faster in the target position, so I didn’t declare it.

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?*

The total states are (3x2x4x4=) 96. For the Q function it is dependent of the states and actions, there is 4 possible actions, so 384 possibilities. The number is reasonable to make the decisions, considering that is expected to train the algorithm it is possible to let it running and learn about the decisions to make. Also for a given state, when the algorithm gets the positive result, it is not necessary anymore to test this possibility.

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?*

At first the car was moving to the expected direction but he had to do a movement different than normal (turn or a turn that it has not done before) it would just stop. After I changed the q max formula from “greater than” to “greater or equal to” and it started working perfectly. It was happening because to be stopped would not generate and penalty and it would not update its reward. When I changed to “greater or equal to”, it would prefer to test anything than to be stopped (in the list possible_actions, None is the first), unless it has already learned that anything else is bad.

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?*

The parameters to measure performance were number of steps for the total test and number of errors.

Alpha	Gama	Epsilon	Total steps	Total errors
0.5	0.0	0.0	1226	28

0.2	0.0	0.0	1193	36
0.8	0.0	0.0	1336	27
0.95	0.0	0.0	1252	31
0.8	0.5	0.0	1478	40
0.8	0.8	0.0	2050	665
0.8	0.0	0.2	1585	222
0.8	0.0	0.05	1288	78

The final agent is performing very well and it will perform very well for several parameters.

Analyzing the parameters:

- Alpha: since the Q is starting as 0, for any alpha between 0 and 1 if the reward is negative it will never be done again (unless by random), so value of alpha is really irrelevant. The variation in the table is probably only the variation of the test.
- Gamma: is applied for the future Q, but as mentioned above the main value here is the reward, we could set the gamma to 0. In most of the cases the future Q will be 0 (since the case is 0 any way). A Gamma different than 0 only get the performance worse in this case.
- Epsilon: Since we are testing all the movements before choosing to stop (what is 0 reward), having random moves is in fact worse. We only move when we know it is good to move, random moves are terrible for the algorithm, so in this case the best is epsilon 0. Including random moves, we can see that the model really had a worse performance.

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?*

The agent gets to the optimal policy since it tests all the cases (or at least all the positive ones). The case is to every time possible do the recommended move except when there is a penalty, in this case, stop. In 100 runs it is performing around 30 errors and around 12.5 steps per run it is an excellent performance.