**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.

**Inform the Driving Agent**

Now that your driving agent is capable of moving around in the environment, your next task is to identify a set of states that are appropriate for modeling the **smartcab** and environment. The main source of state variables are the current inputs at the intersection, but not all may require representation. You may choose to explicitly define states, or use some combination of inputs as an implicit state. At each time step, process the inputs and update the agent's current state using the self.state variable. Continue with the simulation deadline enforcement enforce\_deadline being set to False, and observe how your driving agent now reports the change in state as the simulation progresses.

***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?*

***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?*

**Implement a Q-Learning Driving Agent**

With your driving agent being capable of interpreting the input information and having a mapping of environmental states, your next task is to implement the Q-Learning algorithm for your driving agent to choose the *best* action at each time step, based on the Q-values for the current state and action. Each action taken by the **smartcab** will produce a reward which depends on the state of the environment. The Q-Learning driving agent will need to consider these rewards when updating the Q-values. Once implemented, set the simulation deadline enforcement enforce\_deadline to True. Run the simulation and observe how the **smartcab** moves about the environment in each trial.

The formulas for updating Q-values can be found in **[this](https://classroom.udacity.com/nanodegrees/nd009/parts/0091345409/modules/e64f9a65-fdb5-4e60-81a9-72813beebb7e/lessons/5446820041/concepts/6348990570923" \t "_blank)** video.

***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?*

**Improve the Q-Learning Driving Agent**

Your final task for this project is to enhance your driving agent so that, after sufficient training, the **smartcab** is able to reach the destination within the allotted time safely and efficiently. Parameters in the Q-Learning algorithm, such as the learning rate (alpha), the discount factor (gamma) and the exploration rate (epsilon) all contribute to the driving agent’s ability to learn the best action for each state. To improve on the success of your **smartcab**:

* Set the number of trials, n\_trials, in the simulation to 100.
* Run the simulation with the deadline enforcement enforce\_deadlineset to True (you will need to reduce the update delay update\_delayand set the display to False).
* Observe the driving agent’s learning and **smartcab’s** success rate, particularly during the later trials.
* Adjust one or several of the above parameters and iterate this process.

This task is complete once you have arrived at what you determine is the best combination of parameters required for your driving agent to learn successfully.

***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?*

***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?*