Implement a basic driving agent

The agent randomly moves on the grid. It takes lots of penalties. It eventually makes it to the target location but a lot of time after deadline is passed (generally 20 to 40 turns after deadline is passed).

Identify and update states

Initially I reported the state as follows:

- self.state = (inputs[0],inputs[1],inputs[2],inputs[3],self.next_waypoint,deadline)
- inputs[0] : provides information about the color of the traffic light at the current cross road
- inputs[1],inputs[2],inputs[3] : let the agent know about incoming traffic. With only three other cars on the road. Those inputs are most of the time set to None. But with enough training, the car will be able to behave accordingly.
- self.next_waypoint : provides the planner direction to reach destination as quickly as possible (but does not take into account other cars and traffic lights)
- deadline : provides the time left to reach destination

Ideally, the car would collect as many rewards as possible while reaching destination before deadline. Nonetheless, it multiplies the number of states to such an extend that I decided to remove it from the state definition so that the state space can be learned in a decent amount of time.

With the remaining state space provides all required information for the agent to reach destination as quickly as possible while respecting traffic rules.

Implement Q-learning

Implementing Q-learning without randomly choosing an action from time to time does not perform better than the random policy. The agent always chooses the action it has previously selected in the same state.

Adding some random exploration in the decision related to each step eventually leads to an agent following the planner while respecting traffic rules.

I chose epsilon = 0.1, to get even closer to the optimal policy, epsilon should slowly decay with time.