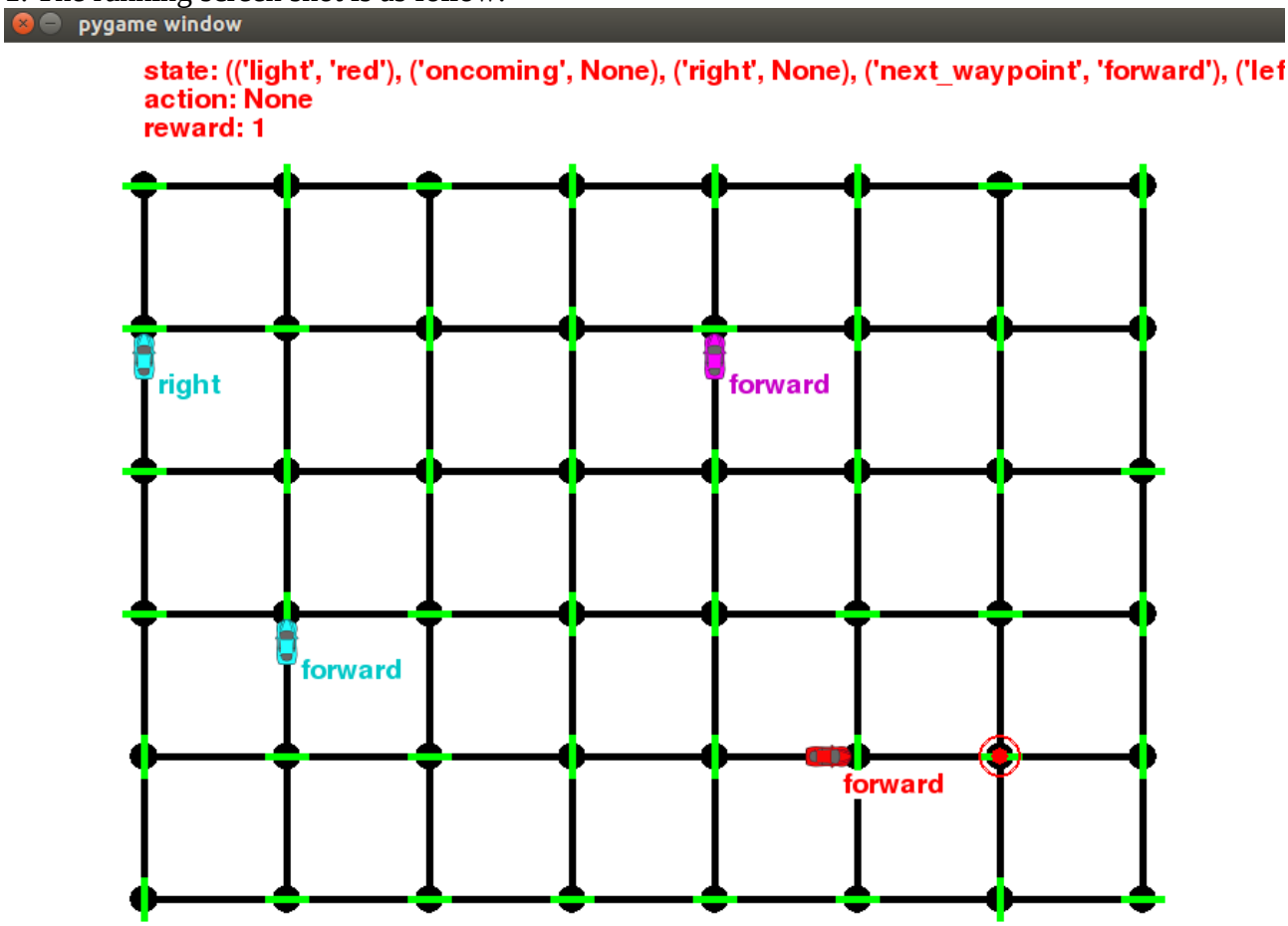
1. The running screen shot is as follow:




```
Simulator.run(): Trial 0
Environment.reset(): Trial set up with start = (3, 2), destination = (6, 4), deadline = 25
RoutePlanner.route_to(): destination = (6, 4)
LearningAgent.update(): deadline = 25, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = left, reward = -1
LearningAgent.update(): deadline = 24, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = right, reward = 2
LearningAgent.update(): deadline = 23, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = None, reward = 1
LearningAgent.update(): deadline = 22, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = None, reward = 1
LearningAgent.update(): deadline = 21, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = right, reward = 2
LearningAgent.update(): deadline = 20, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'forward', 'left': None}, action = None, reward = 1
LearningAgent.update(): deadline = 19, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'forward', 'left': None}, action = None, reward = 1
LearningAgent.update(): deadline = 18, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'forward', 'left': None}, action = None, reward = 1
LearningAgent.update(): deadline = 17, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'forward', 'left': None}, action = left, reward = -1
LearningAgent.update(): deadline = 16, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'forward', 'left': 'right'}, action = left, reward = 0.5
LearningAgent.update(): deadline = 15, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = None, reward = 1
LearningAgent.update(): deadline = 14, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = forward, reward = 0.5
LearningAgent.update(): deadline = 13, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = right, reward = 2
LearningAgent.update(): deadline = 12, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'next_waypoint': 'forward', 'left': None}, action = left, reward = 0.5
LearningAgent.update(): deadline = 11, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = None, reward = 1
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 10, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'next_waypoint': 'right', 'left': None}, action = right, reward = 12
```

The red car is taking action randomly, but after a long time, it can reach the destination.

2.
State includes "light", "oncoming", "right", "left" and "next_waypoint". "Light", "oncoming", "right", "left" is important, these give agent whether it will get punishment by taking some actions. "Next_waypoint" gives the agent the right direction to go tothe destination as soon as possible.

3. The agent will tend to follow the traffic rule and take positive reward actions and walk to the destination.

4. I have tried different combinations of alpha and gema, [(1, 1), (0,1,0,1), (0.2,0,1), (0.5, 0)], and find that (0.2, 0.1) can give good result, but it dosn't guarantee perfect destination reached, sometimes the car is always turning right and getting stucked. (0.5, 0) is a great choice, the car can find a good strategy to get to the destination very fast and achieve perfect performance.
The car basically find the best strategy, get to the destination very fast, and never get a negative reward.