<u>FAI - Project Proposal</u>

Project Name: Self-driving Smart Cab

Proposed Milestone Deliverables:

- Milestone 1 - Getting familiar with PyGame and other libraries and setting up the environment. Start implementing the Q-Learning Algorithm.

Our Progress:

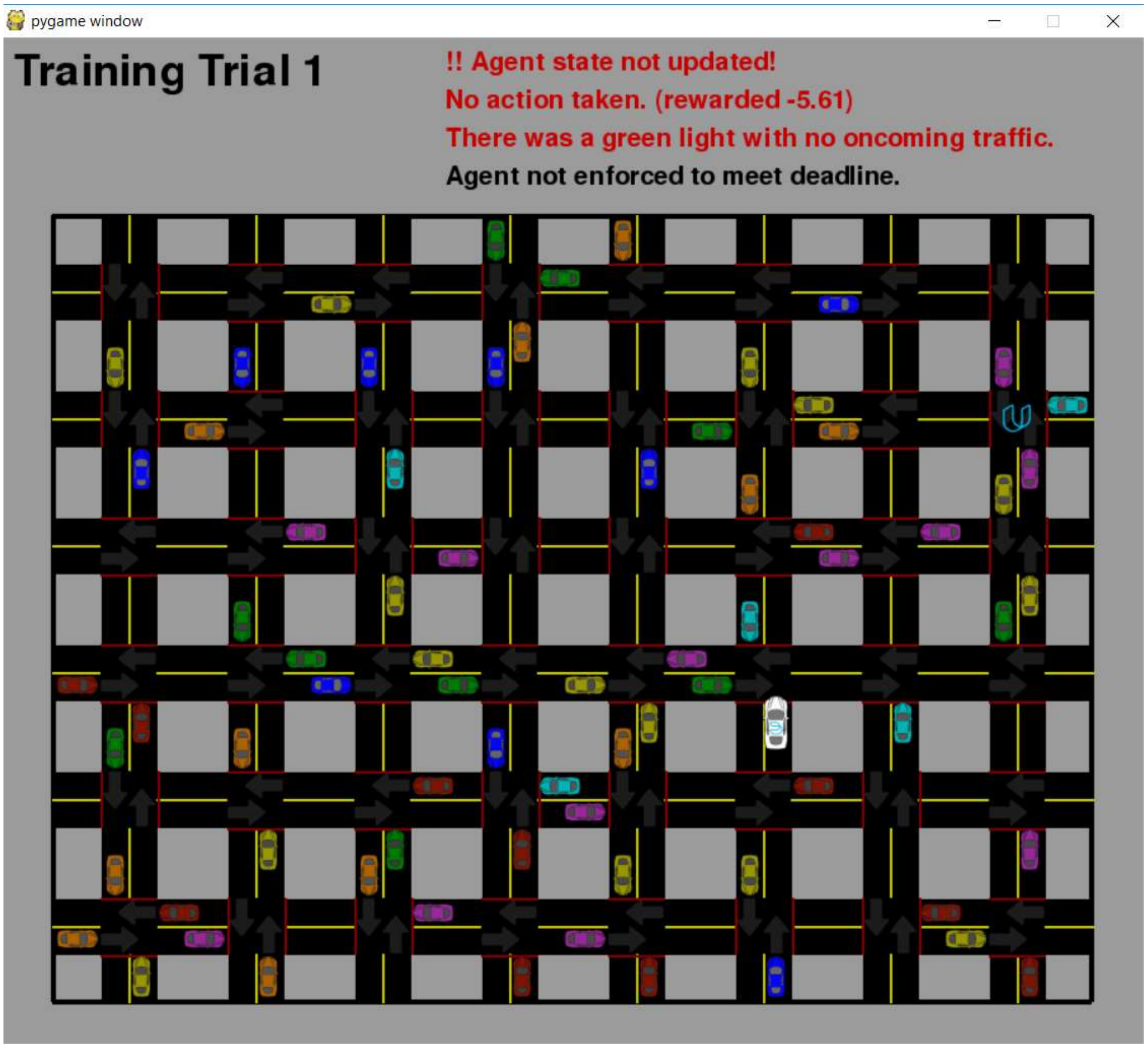
- We have studied and gotten familiar with the working and different implementations of PyGame, numpy and matplotlib. These are the libraries that we will be using for this project at different steps of our implementations.

- We have installed PyGame, numpy and matplotlib along with the Anaconda python distribution. After this we set up the environment in our systems and ran the simulator to understand the problem better.

- We have studied the problem and identified the steps for this project.
    - Understanding the environment – For this we run the environment with an uninformed agent which was not taking any action at any instant i.e. a stationary agent. This allowed us to understand the rewards that the agent was receiving for not moving at different scenarios e.g. not moving when there is a green light, moving at a red light etc. An example of the agent receiving the corresponding rewards upon running the environment is given below.
        - Agent properly idled at a red light. (rewarded 2.59)
        - Agent properly idled at a red light. (rewarded 1.73)
        - Agent properly idled at a red light. (rewarded 2.98)
        - Agent idled at a green light with no oncoming traffic. (rewarded -5.26)
        - Agent idled at a green light with no oncoming traffic. (rewarded -5.68)
        - Agent idled at a green light with no oncoming traffic. (rewarded -4.47)
        - Simulation ended. . .

So here we can see that since the agent is not moving and the traffic lights are getting updated in the environment, depending on the status of the light and the incoming traffic the agent is being rewarded positive or negative rewards. This analysis gives us a basic understanding of the environment. Below is a screenshot of the environment in which the agent is functioning. The white car is the (idle) agent.

The Environment:



Training Trial 1

!! Agent state not updated!

No action taken. (rewarded -5.61)

There was a green light with no oncoming traffic.

Agent not enforced to meet deadline.

- Implementing a basic (uninformed) driving agent – Initially the agent is set to be idle at all instances. So even if there were a green light and no oncoming traffic, the agent would not move and resultantly would receive a negative reward for not taking the right action. Our goal in this step was to make the agent take a random action based on the list of valid actions. To implement this, we updated the agents choose_action method to choose a random action with the following code.

```
•   random_action = random.SystemRandom()
•   return random_action.choice(self.valid_actions)
```

  This now allowed the agent to move randomly and in an uninformed manner in the environment which in turn allows us to observe the interactions and the rewards with greater variability. An example of a trial run after this update is given below.

  - Agent attempted driving left through a red light. (rewarded -9.26)
  - Agent followed the waypoint right. (rewarded 2.50)
  - Agent drove right instead of forward. (rewarded 0.90)
  - Agent properly idled at a red light. (rewarded 1.30)
  - Agent attempted driving left through a red light. (rewarded -10.24)
  - Agent properly idled at a red light. (rewarded 2.75)
  - Agent idled at a green light with no oncoming traffic. (rewarded -4.87)
  - Agent idled at a green light with no oncoming traffic. (rewarded -5.76)
  - Agent drove forward instead of left. (rewarded 0.62)
  - Simulation ended. . .

  Here we can see that the agent does take actions. But most of the times they are not the ideal options since it is completely random.

Milestone One Shortcomings:

- One of the goals that we had set for Milestone 1 was to start implementing a functioning Q-Learning agent. However, we failed to have a functioning version of a Q-Learning agent yet since it took a long time for us to figure out how to set up the environment and get the simulator up and running. However, we have already started implementing the Q-learning algorithm and hope to be done with it soon. We also have to explore the different flag options that the environment provides us to better understand our agent’s actions over time.