CORNELL UNIVERSITY

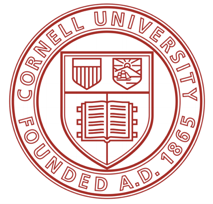
SIBLEY SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

Cornell Autonomous Systems Lab

|  |
| --- |
| “Reinforcement Learning for Car Control”  Final Report |

|  |  |
| --- | --- |
| *Author:*  Min Young Chang | *Professor:*  Mark Campbell  Hadas Kress-Gazit |
| *NetID:*  mc968 | *Advisor:*  Adam Pacheck  Scott Hamill |

May 18th, 2019



**I. ABSTRACT**

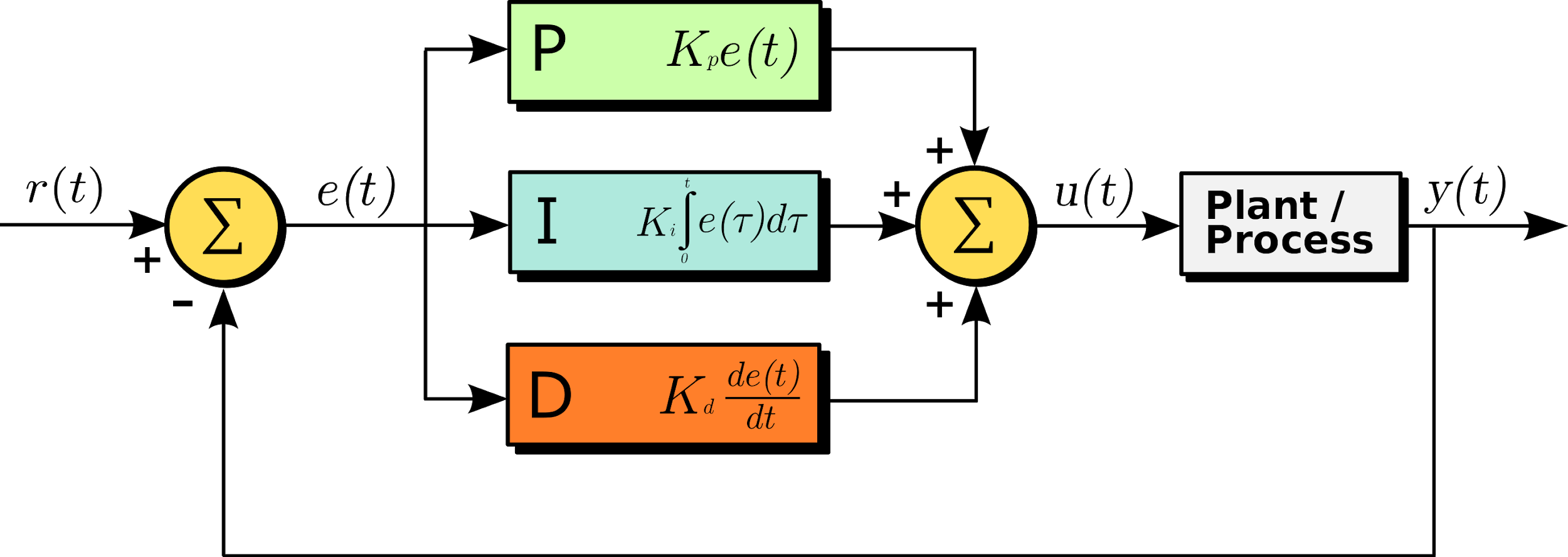
The goal of this project is to implement reinforcement learning for car control, in replacement of the traditional feedback control. Through the process of learning, the simulated car is designed to learn how to go toward right while getting as close as possible to the flag. The process of training and testing the neural network was done in simulation with python code, which included the rendering of the simulation that visualizes the process. After hundreds of training episodes, the model learned how to achieve the goal mentioned above.



*Figure 1. Rendering of the Simulation*

**II. Introduction**

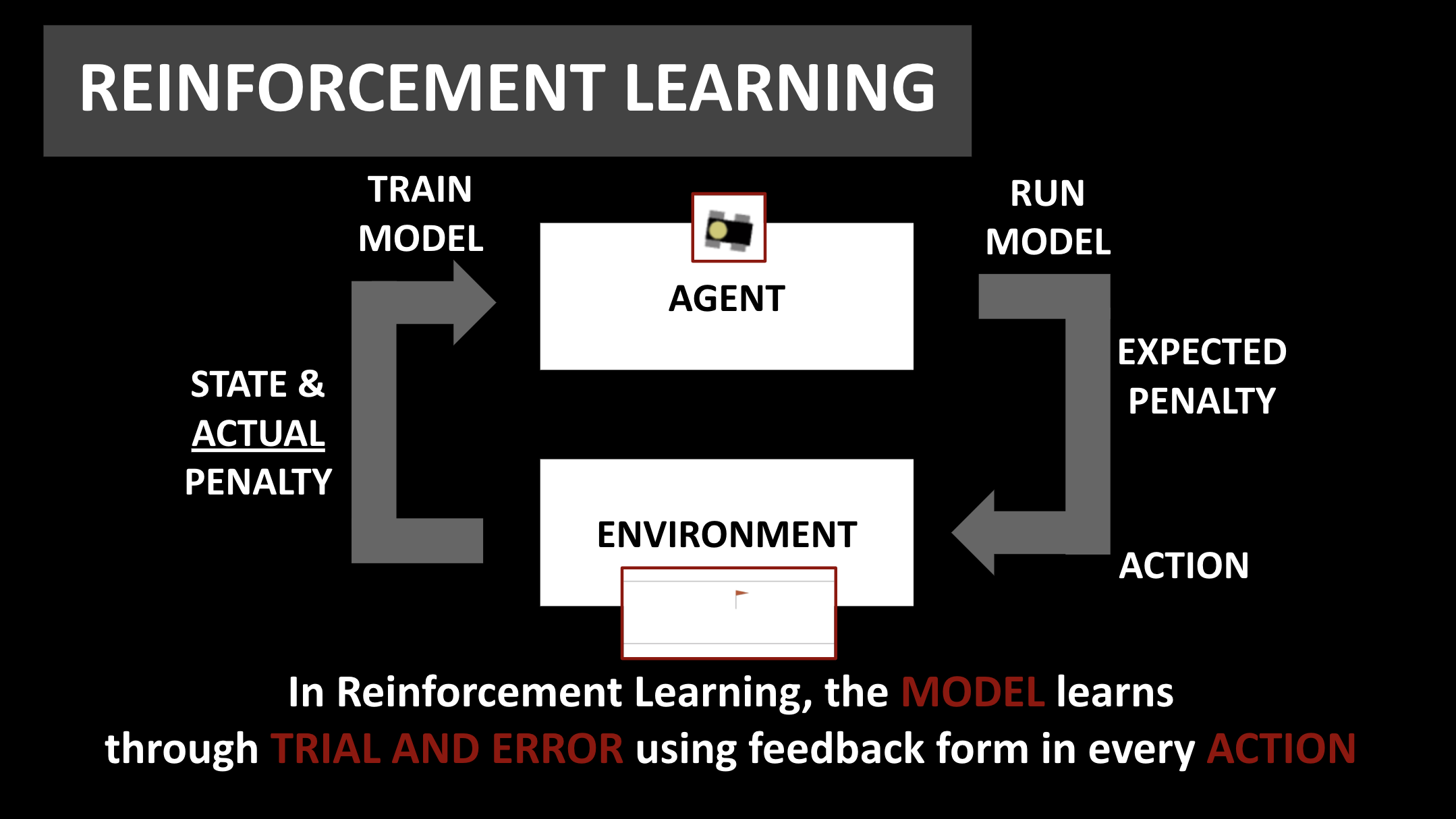
The ultimate goal of the project is to replace the traditional PID Feedback Control, which was originally implemented in “duckiebot” as the main control system. The problem with the PID Feedback Control is that its performance depends on the three parameters—Kp, Ki, and Kd—which are very difficult to be perfectly adjusted. Instead of the PID controller, the new model that is trained through reinforcement learning will figure out the best control system that has even better performance than the original PID controller.



*Figure 2. Diagram of PID Feedback Control*

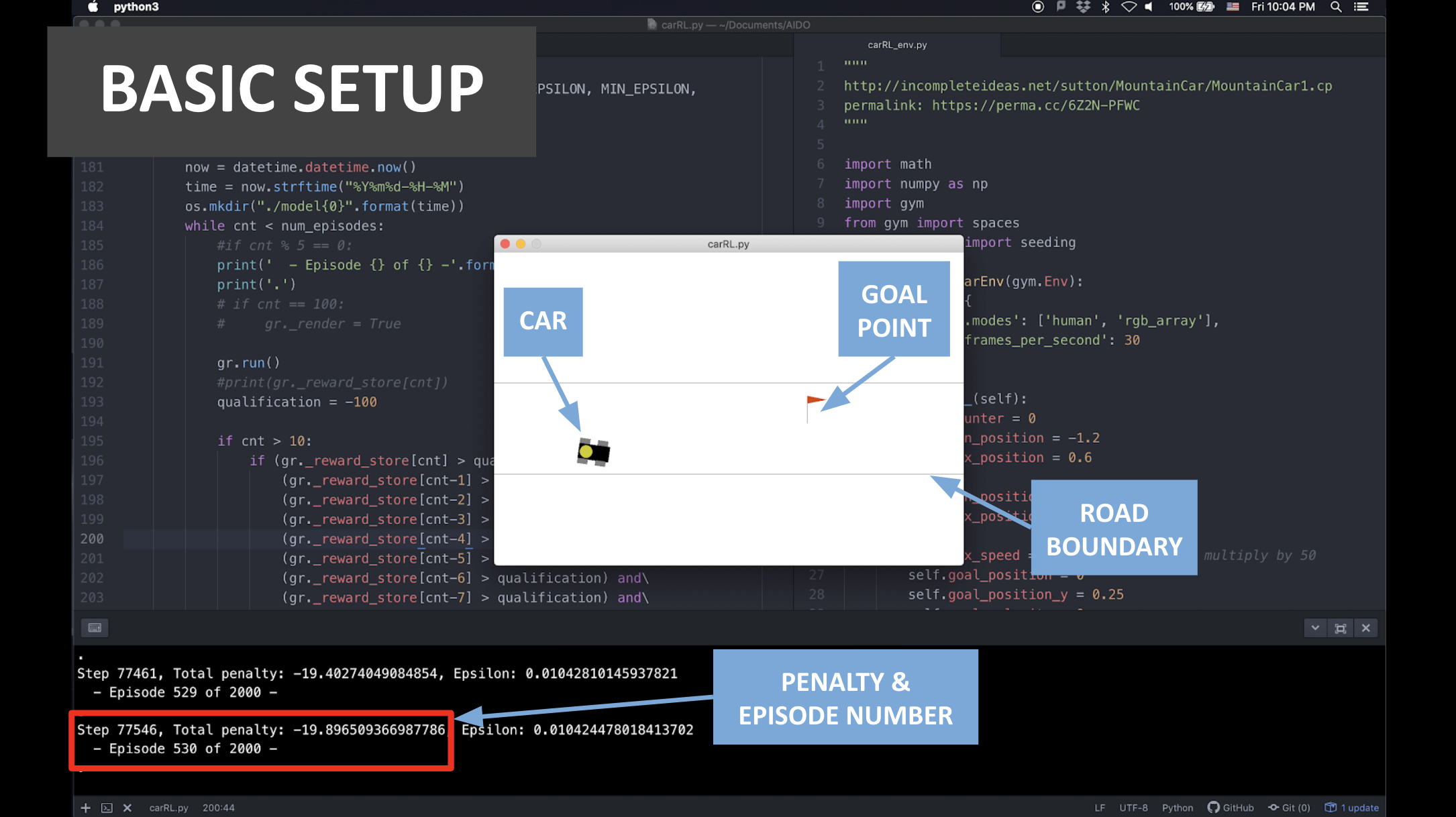
**III. Concept**

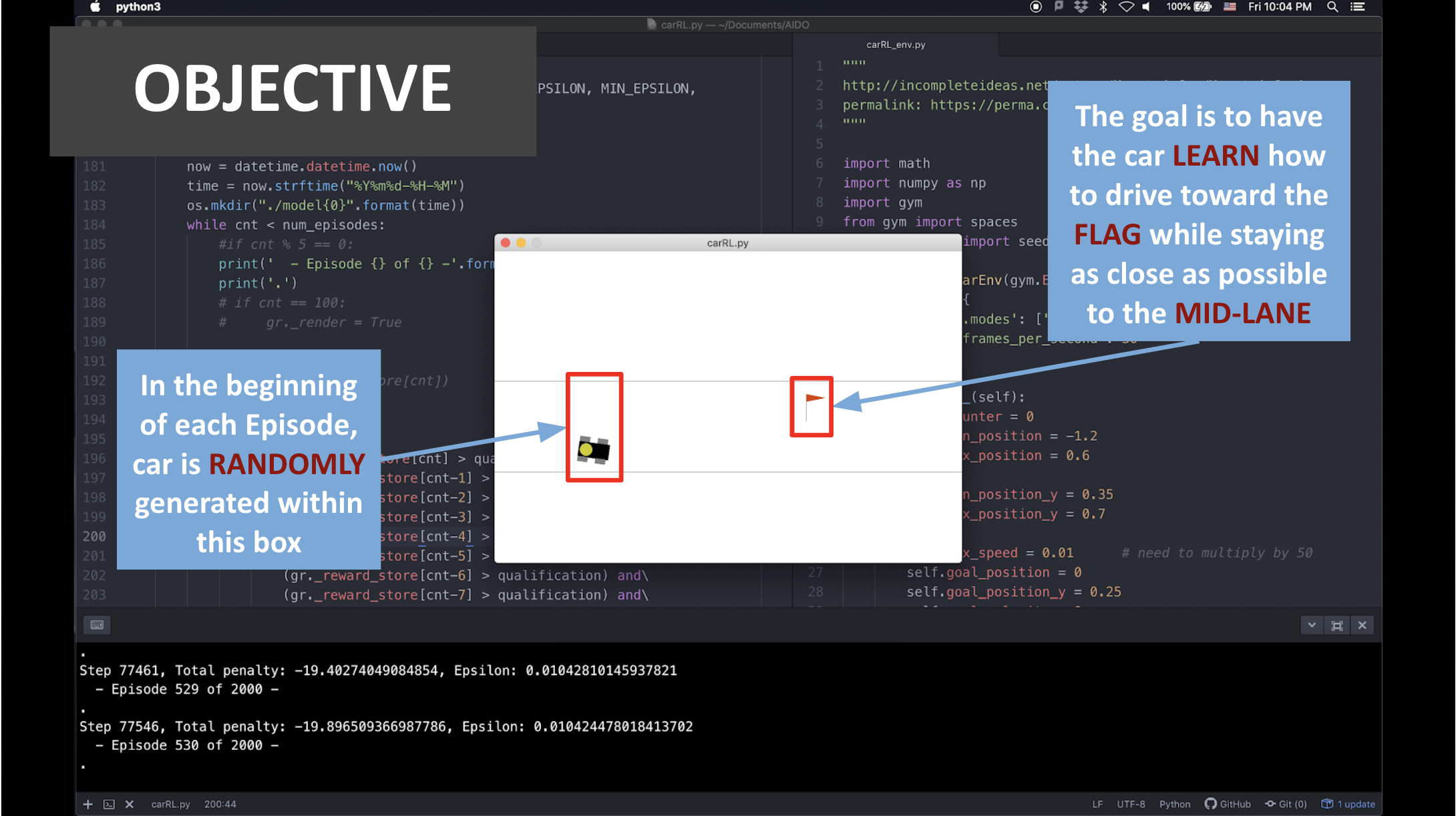
Reinforcement Learning is a type of Machine Learning. A typical Machine Learning process consists of three main steps: acquiring training data, training the model, and testing the model. However, reinforcement learning combines the three steps into a single repetitive process. In the beginning of the process, there is no training data — unlike other usual Machine Learning processes. The model chooses actions randomly, because the model is not trained yet. However, such random actions result in corresponding penalties that was originally set in the given environment. Here, the model received the current state of the car as the input, the current model calculated the actions based on it — although it is quite meaningless yet — and corresponding penalty was given by the environment. These three pieces of information combine into “a piece of training data.” In other words, each step of the learning process results in this single piece of training data. Based on the data acquired, the model gets trained, and calculate next action based on the updated state of the car. The model is not trained much in the early phase of the process, but after a few thousand steps in the learning process, the model is trained with a few thousand training data and become much more accurate in calculating.



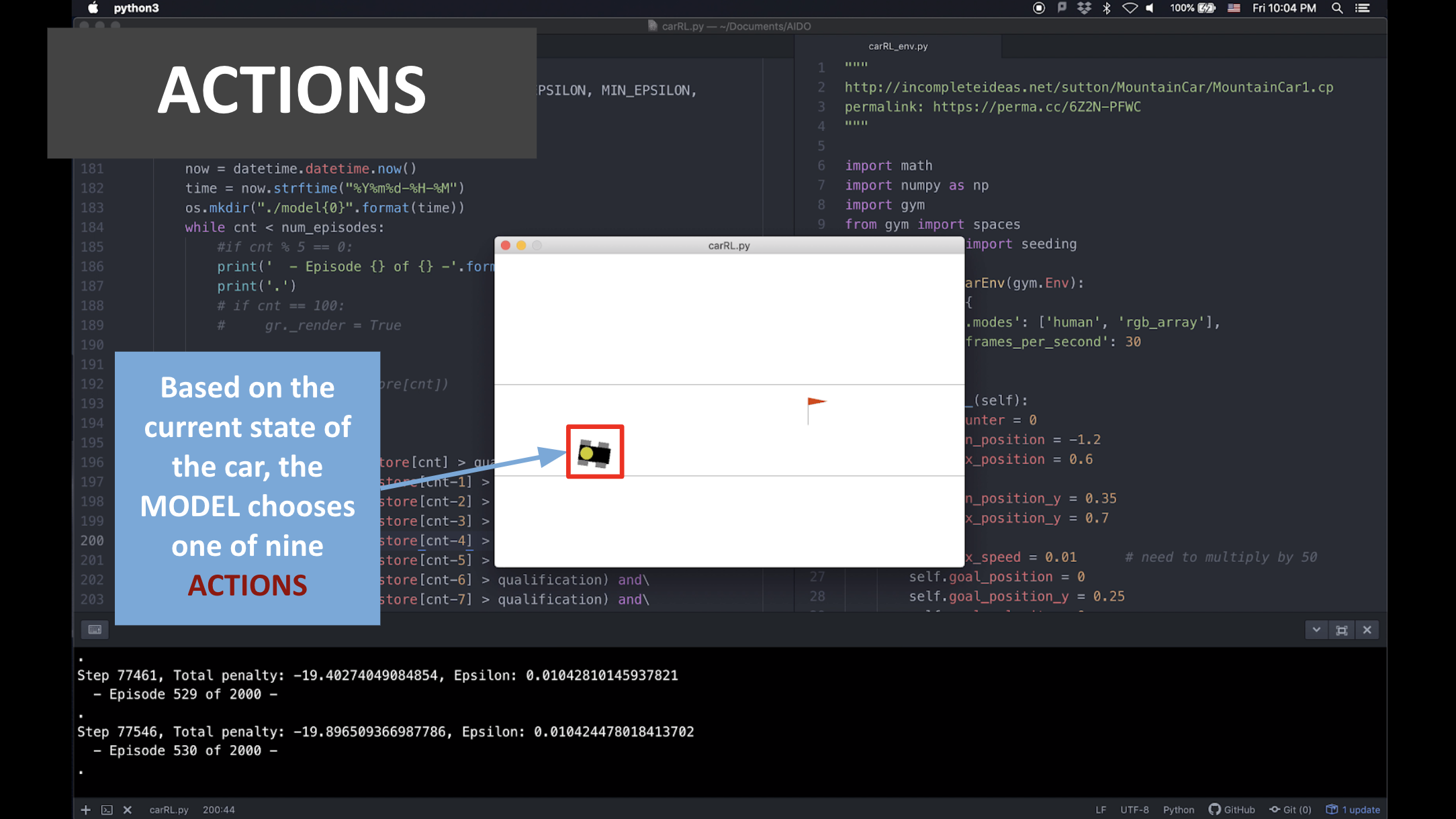
*Figure 3. Diagram of Reinforcement Learning Process*

**IV. Process**

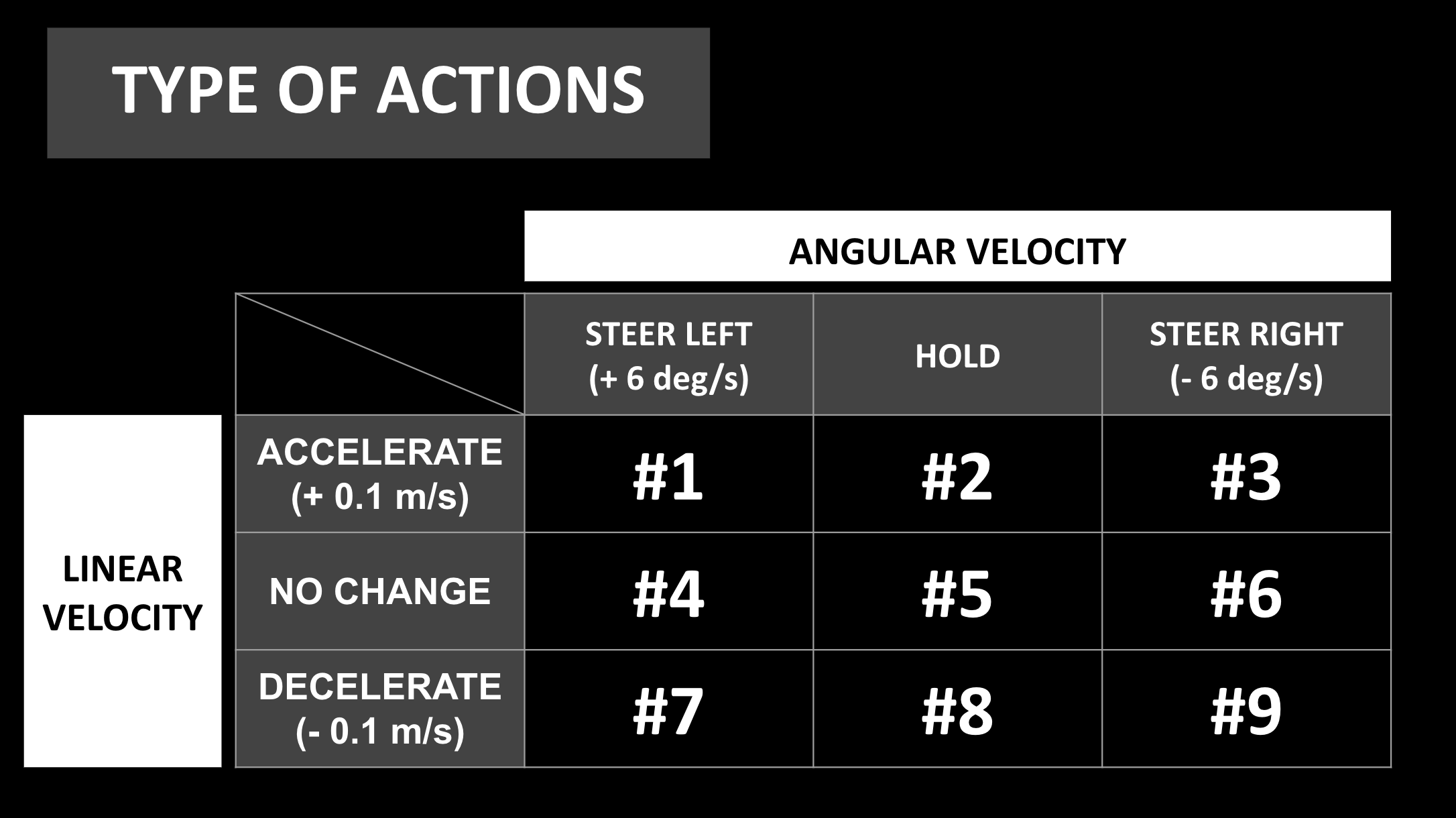




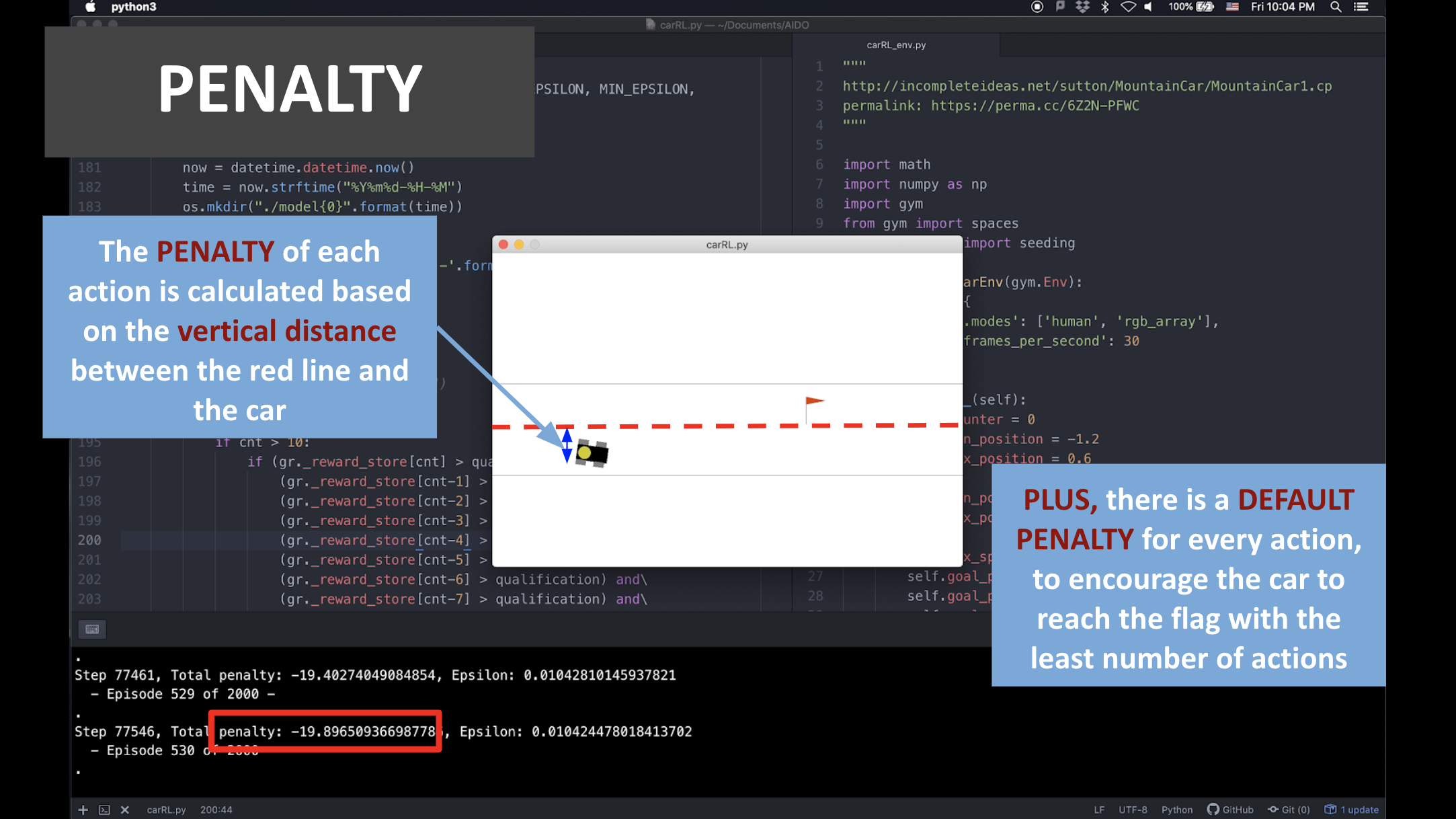
Staying as close as possible to the mid-lane of the road is one of the most important requirements of autonomous car.



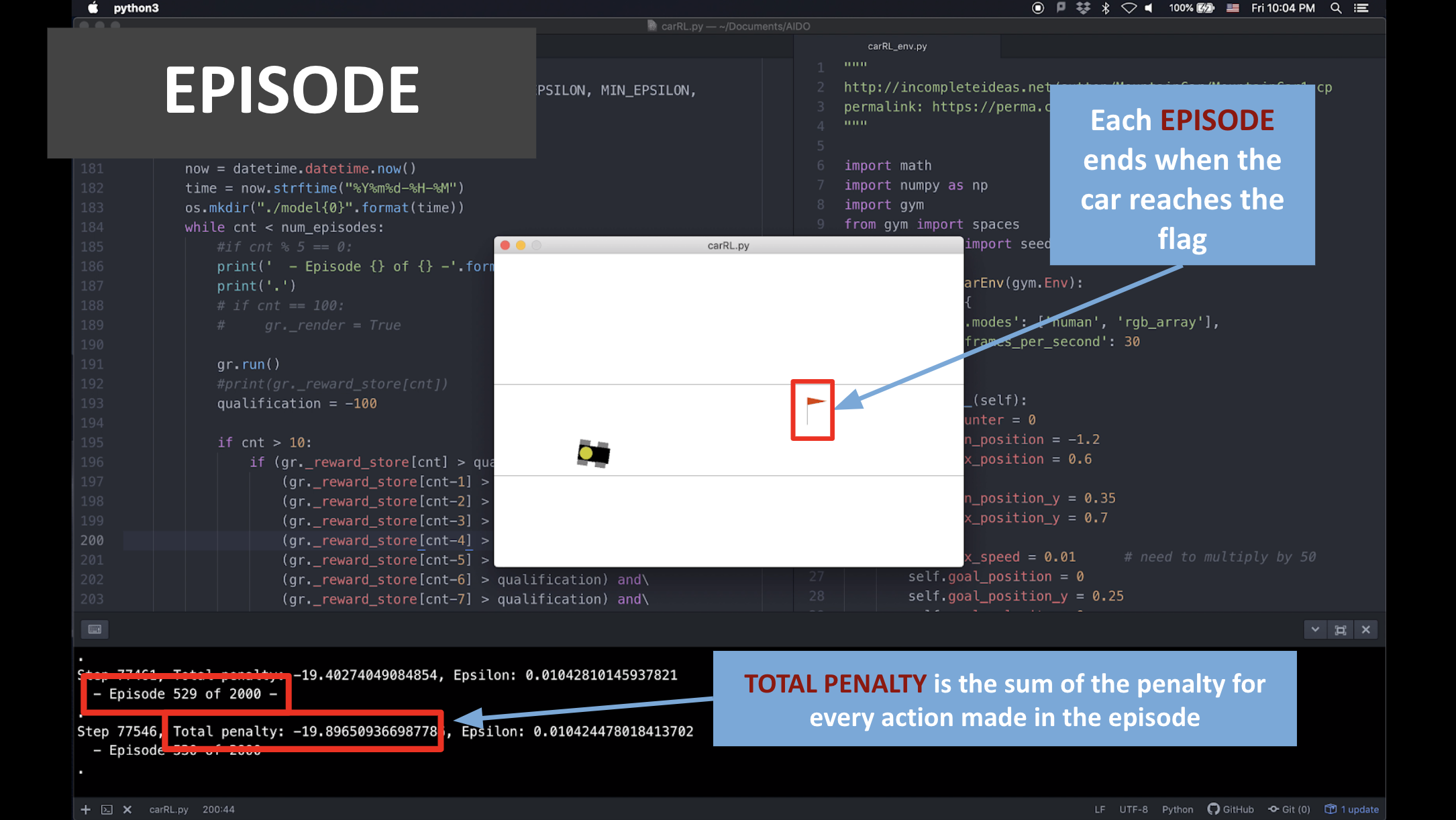
A current state is consisted of 6 factors: x-position, y-position, velocity, angular velocity, x-goal, y-goal. The first four factors get updated based on the action made, but the x,y position of the goal point does not change.

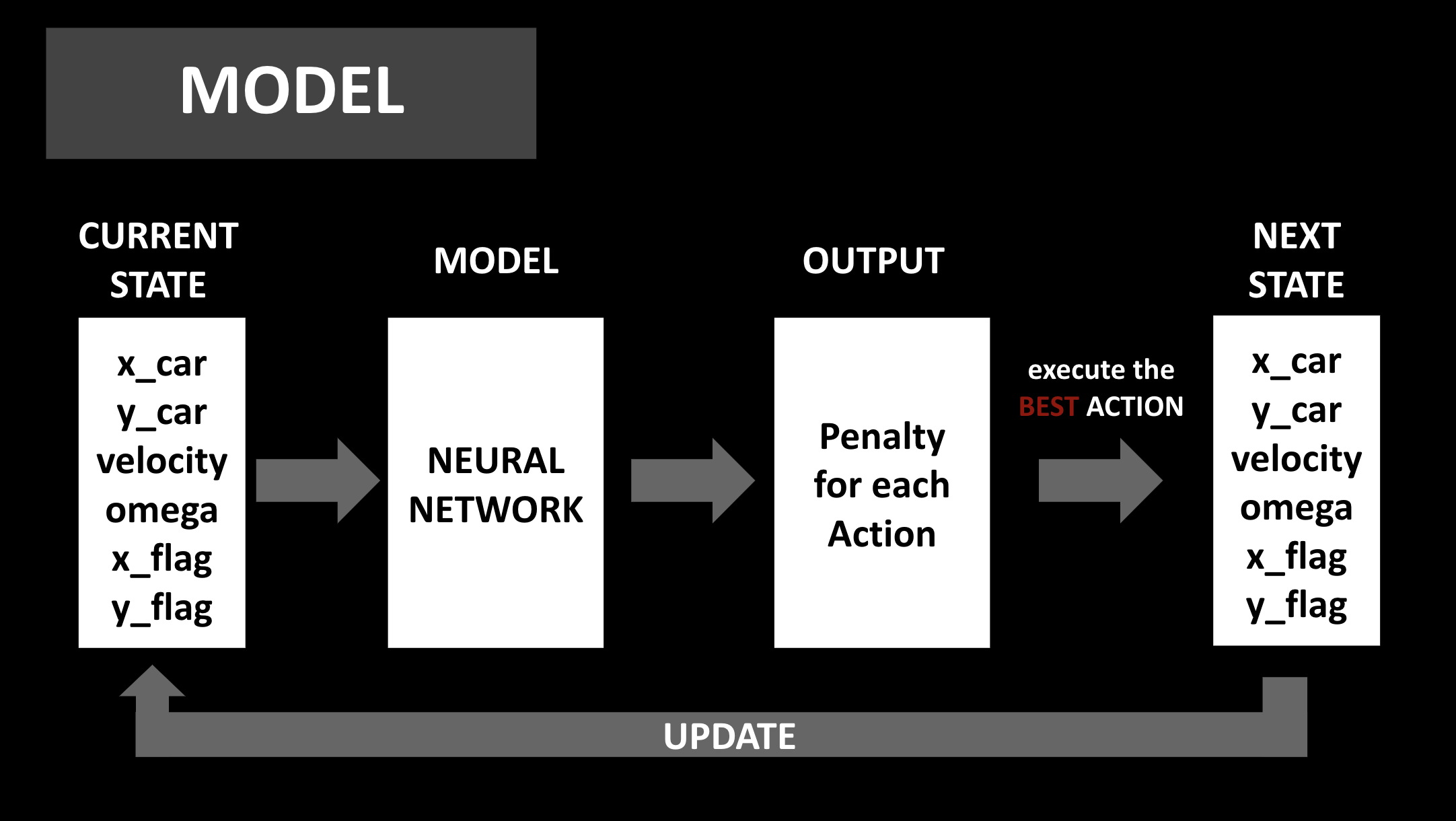


As mentioned above, there are a total of nine different actions. For example, action number 1 would have the care steer to the left while increasing its velocity.

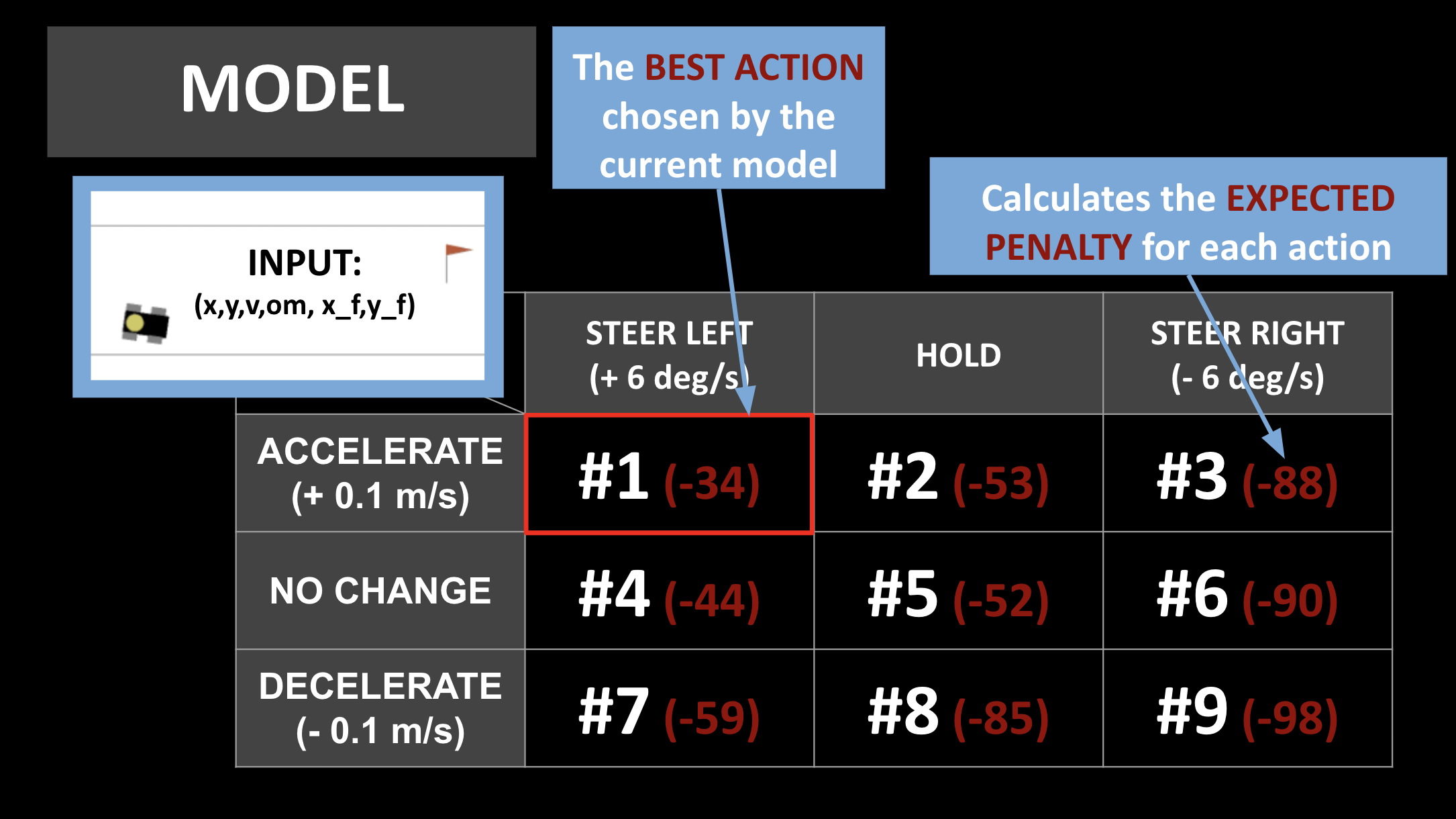


The penalty based on the vertical distance between the car and the mid-lane was to encourage the car to stay as close as possible. The default penalty is, as explained in the slide, to encourage the car to drive as fast as possible.





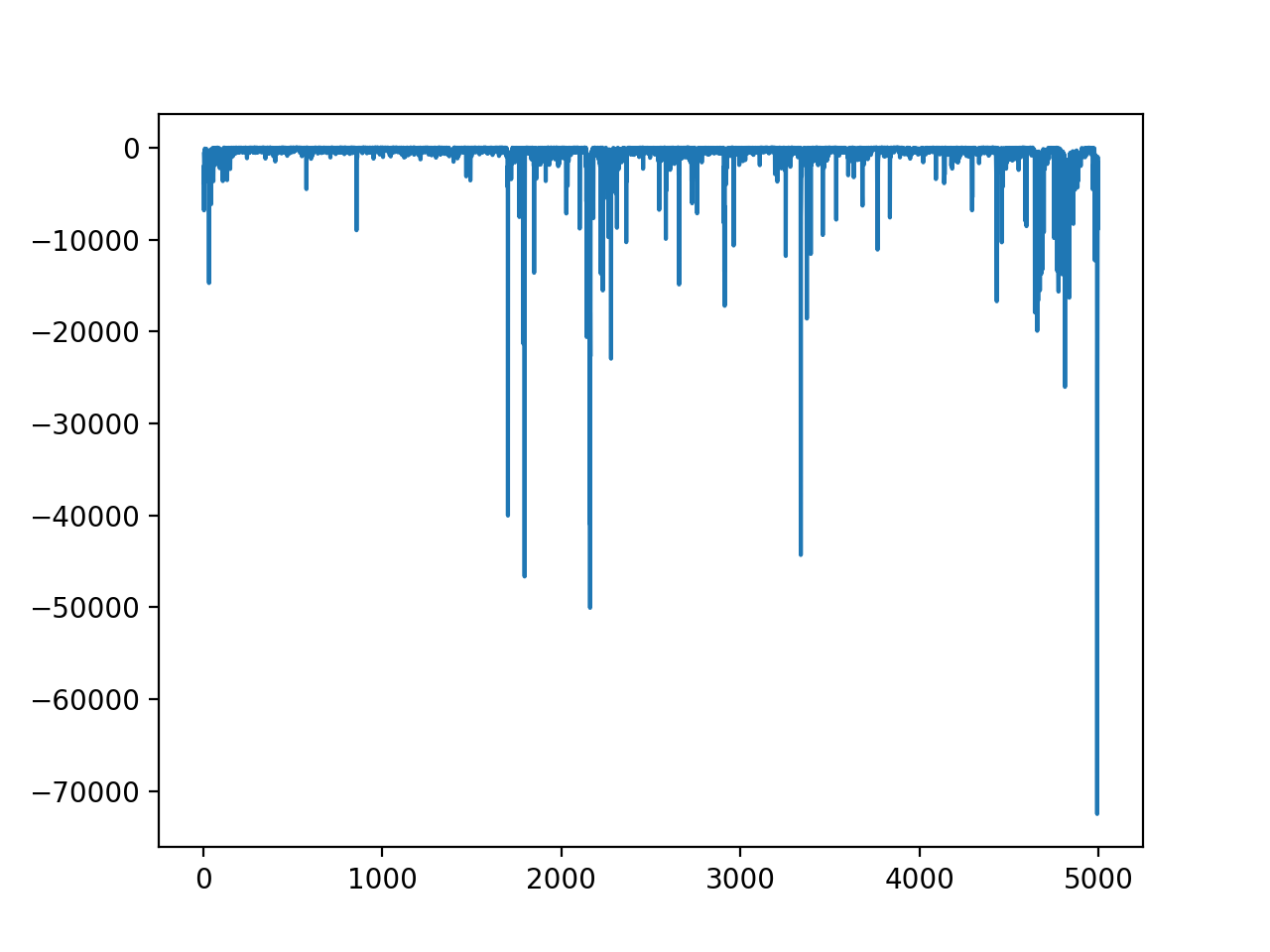
This is a diagram that explains how the actions are chosen and the states are updated based on those.



**V. Result and Discussion**

Video for process and result: <https://www.youtube.com/watch?v=piQjevcBO6c&t=0s>

After 1500 episodes of training, the model was trained well enough to control the car to drive toward the flag while staying close to the mid-lane of the road. When the model finished the episode with less than 100 total penalty for 10 consecutive episodes, it was extracted as the final trained model.

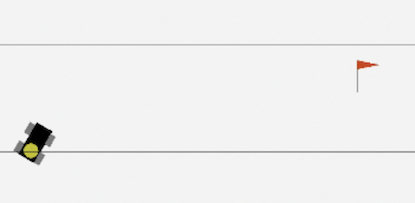


x-axis: number of episode, y-axis: total penalty of the episode

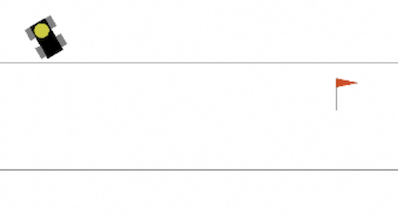
*Figure 4. Penalty Graph*

According to Figure 4, the total penalty converged to almost zero by Episode number 1500. Interestingly, the total penalty starts to diverge after Episode number 1600. This phenomenon is caused by the system of reinforcement learning. As discussed above, reinforcement learning is designed to learn through trial and error. This method definitely helps the model to train almost perfectly. However, after the model is trained well enough and there is not much room left for more improvement, the model starts trying somewhat undesired actions, because it is designed to try something new. For this reason, it is very important to end the process to prevent the model from making meaningless tries and learn from those.

The trained model, which was extracted at episode number 1510, was tested with a couple of extreme cases. During the training, the car was randomly generated in terms of y-coordinate, but it was always facing toward the x-axis. In the two cases of testing, car was generated from the top and bottom of the road, facing almost perpendicular to the x-axis as shown in Figure 5 and Figure 6:



*Figure 5. Test Case 1*



*Figure 6. Test Case 2*

For both of the cases, the car successfully drove toward the red flag while staying as close as possible to the mid-lane of the road. This demonstration is included at the end of the video attached.

The major challenge of this project was on designing the environment, which includes the number of possible actions and the penalty system. At first, I designed the car to have an almost infinite number of actions by letting it freely change its speed and direction within a certain range. However, this resulted in a failure. Since there were too many possible actions, the model took too long to be trained, and even the result of training was very unsuccessful. It was very important to discretize the actions into 9 different options. With less number of actions possible, the learning process became much faster with better results.

The design of the penalty system also greatly influenced the result of the learning process. In this project, the goal has two parts: driving toward the flag, and driving close to the mid-lane. Penalty was assigned separately for each part, and the balance of the two parts was very important. For example, if the penalty for driving far from the mid-lane is too big compared to that for getting to the goal point fast, the learning process only focuses on driving near the mid-lane. In this case, the car did not really accelerated but only focused on getting to the mid-lane and stay there for a long period of time. On the other hand, when the penalty for getting to the goal point late was too big, the car was trained to simply accelerate as much as possible and go straight toward the right, without making any kind of steer toward the mid-lane. In this project, the ideal penalty system was found through many trials.

As a counterpart of the penalty system, a reward system was tried during the project. In the reward system was designed to reward the car when it makes an action that gets the car closer to the mid-lane. This system generated an interesting result: instead of driving toward the flag, the car started driving back and forth while facing downward, in an attempt to receive more rewards. As shown in this trial, a poorly designed penalty/reward system created meaningless results.

**VI. Future Work**

The next step of this project can be applying the trained model to the actual “duckiebot,” through ROS and replace the traditional PID controller. It is very important to design the specification of the simulated car to resemble that of the actual car. The maximum speed, maximum angular speed, rate of acceleration and steering are four the most important factors of the specification. The goal points — for the actual car— will be received from the code of the path-planning team. It will be very interesting to see how the model trained through simulation performs in reality.

Repository: <https://github.coecis.cornell.edu/asl/duckietown-software/tree/master19/catkin_ws/src/11-control/ReinforcementLearning>

**VII. Real-world Autonomous Driving Reflection**

After working on the AIDO project, I was very much intrigued by the field of self-driving. Especially, while working on this particular project, I learned a lot about machine learning. I learned a lot from reading papers and articles on the internet and trying various tutorials posted in githubs. This experience would enable me to self-teaching whatever I would be working on in the future. I am very excited to continue my research on Autonomous Driving, especially on implementing LIDAR sensors and computer vision.