Lab 2: MGM SpeedWay Model

Overview

In this lab you will get 20 minutes to create the model you will be racing with at the MGM SpeedWay. You can always train another model afterwards, but make good use of the time and resources to come up with a great model. From the Reinforcement learning page choose **Create model** and leave all hyperparameter settings as they are, and focus your time on the reward function.

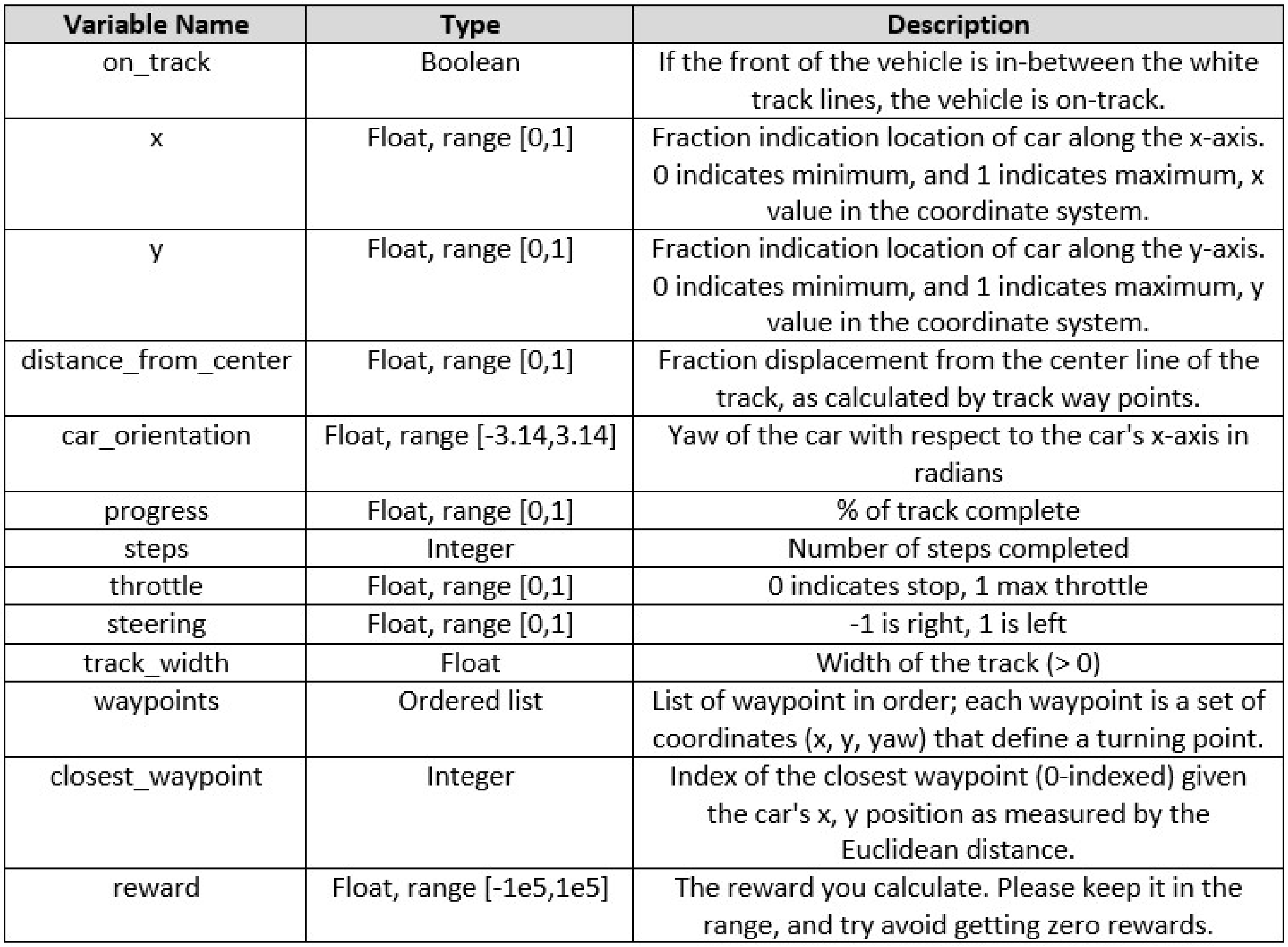
While you don't have to start training your model after 20 minutes, it is adviseable to try get to a point where you can as training will take the better part of an hour and you want to ensure you get to the MGM SpeedWay with your model on your USB stick.

Once your model training is completed, create a folder called "models" on your usb stick, download the model from the AWS DeepRacer console, and copy it to the models folder.

Reward Function Tips

Tip 1: Start off looking at the advanced reward functions for inspiration. We provide a few examples below. Tip 2: Think carefully through the driving behavior you want to incentivize and consider the trade-offs. For example, you can penalize your car for going slow, but if all your can can do is go fast, it may not be the best at taking turns.

Here are the variables you can use in your logic. Furthermore, the following three cells show examples of advanced reward functions.



Variable Name	Type	Description
on_track	Boolean	If the front of the vehicle is in-between the white track lines, the vehicle is on-track.
x	Float, range [0,1]	Fraction indication location of car along the x-axis. 0 indicates minimum, and 1 indicates maximum, x value in the coordinate system.
y	Float, range [0,1]	Fraction indication location of car along the y-axis. 0 indicates minimum, and 1 indicates maximum, y value in the coordinate system.
distance_from_center	Float, range [0,1]	Fraction displacement from the center line of the track, as calculated by track way points.
car_orientation	Float, range [-3.14,3.14]	Yaw of the car with respect to the car's x-axis in radians
progress	Float, range [0,1]	% of track complete
steps	Integer	Number of steps completed
throttle	Float, range [0,1]	0 indicates stop, 1 max throttle
steering	Float, range [0,1]	-1 is right, 1 is left
track_width	Float	Width of the track (> 0)
waypoints	Ordered list	List of waypoint in order; each waypoint is a set of coordinates (x, y, yaw) that define a turning point.
closest_waypoint	Integer	Index of the closest waypoint (0-indexed) given the car's x, y position as measured by the Euclidean distance.
reward	Float, range [-1e5,1e5]	The reward you calculate. Please keep it in the range, and try avoid getting zero rewards.

In [8]:
```
#Advanced Reward Function 1
def reward_function(on_track, x, y, distance_from_center, car_orientation, progress
, steps, throttle, steering, track_width, waypoints, closest_waypoint):

    '''
    '''

    import math

    marker_1 = 0.1 * track_width
    marker_2 = 0.25 * track_width
    marker_3 = 0.5 * track_width

    reward = 1e-3
    if distance_from_center >= 0.0 and distance_from_center <= marker_1:
        reward = 1
    elif distance_from_center <= marker_2:
        reward = 0.5
    elif distance_from_center <= marker_3:
        reward = 0.1
    else:
        reward = 1e-3  # likely crashed/ close to off track

    # penalize reward if the car is steering way too much
    ABS_STEERING_THRESHOLD = 0.5
    if abs(steering) > ABS_STEERING_THRESHOLD:
        reward *= 0.8

    return float(reward)
```

In [9]:
```
#Advanced Reward Function 2
def reward_function(on_track, x, y, distance_from_center, car_orientation, progress
, steps, throttle, steering, track_width, waypoints, closest_waypoint):

    '''
    '''

    import math

    marker_1 = 0.1 * track_width
    marker_2 = 0.25 * track_width
    marker_3 = 0.5 * track_width

    reward = 1e-3
    if distance_from_center >= 0.0 and distance_from_center <= marker_1:
        reward = 1
    elif distance_from_center <= marker_2:
        reward = 0.5
    elif distance_from_center <= marker_3:
        reward = 0.1
    else:
        reward = 1e-3  # likely crashed/ close to off track

    # penalize reward for the car taking slow actions
    THROTTLE_THRESHOLD = 0.5
    if throttle < THROTTLE_THRESHOLD:
        reward *= 0.8

    return float(reward)
```

In [10]:

```
#Advanced Reward Function 3
#Bonus Advanced Reward Function 3
def reward_function(on_track, x, y, distance_from_center, car_orientation, progress
, steps, throttle, steering, track_width, waypoints, closest_waypoint):

    reward = 1e-3
    if distance_from_center >= 0.0 and distance_from_center <= 0.03:
        reward = 1.0

    # add steering penalty
    if abs(steering) > 0.5:
        reward *= 0.80

    # add throttle penalty
    if throttle < 0.5:
        reward *= 0.80

    return reward
```

In [ ]: