AWS DEEP RACER

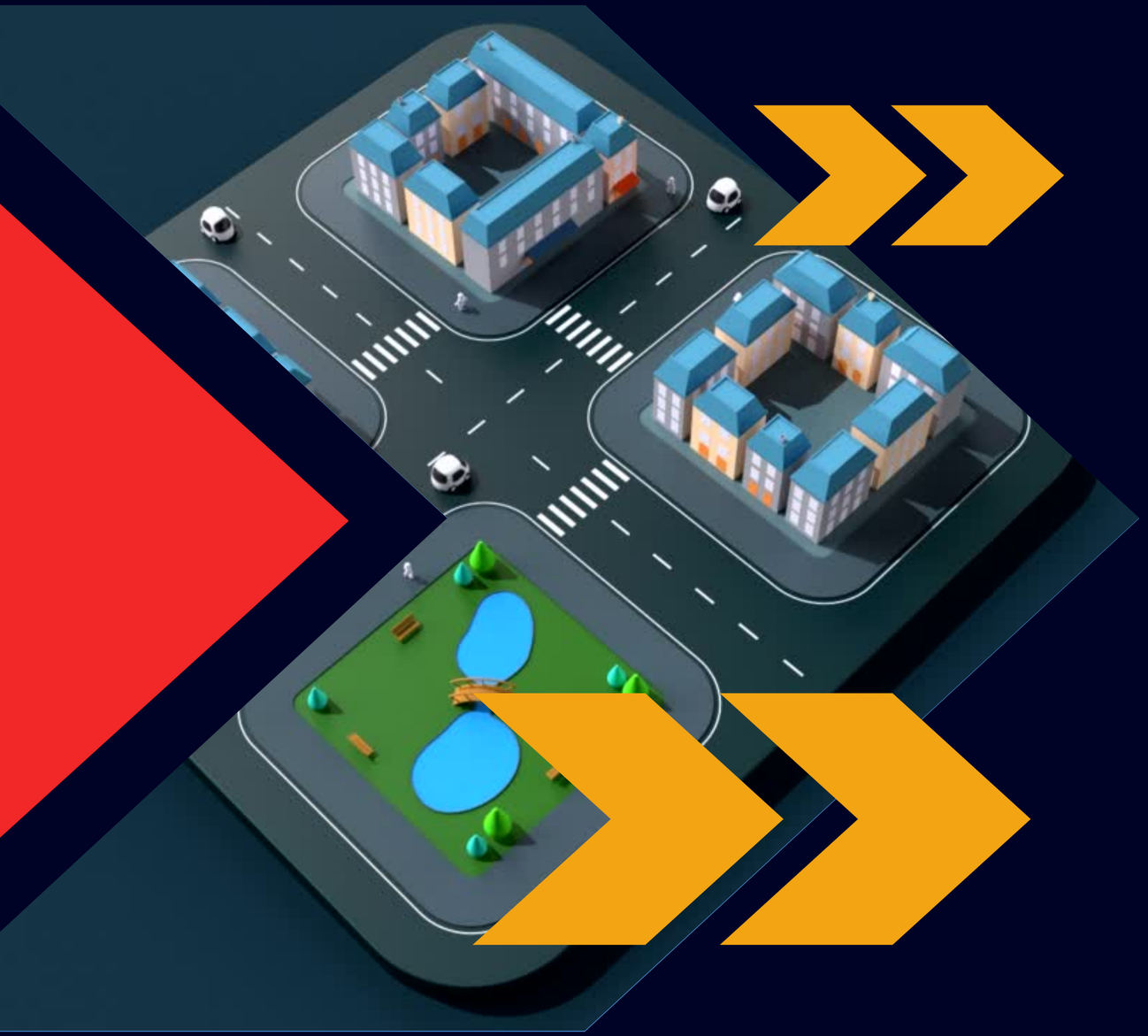
Group 4: Nick Angus, Qichun Yu

Get Started

AWS DEEPRACER

- A fun way to get started with reinforcement learning
- Building and training reinforcement learning model with virtual autonomous car racing
- Compete with autonomous racing league to race for prizes

Simulator **League** **Speed**



REINFORCEMENT LEARNING

- A type of machine learning technique ideal for autonomous driving
- Learning the optimal behavior in an environment to obtain maximum reward

aws

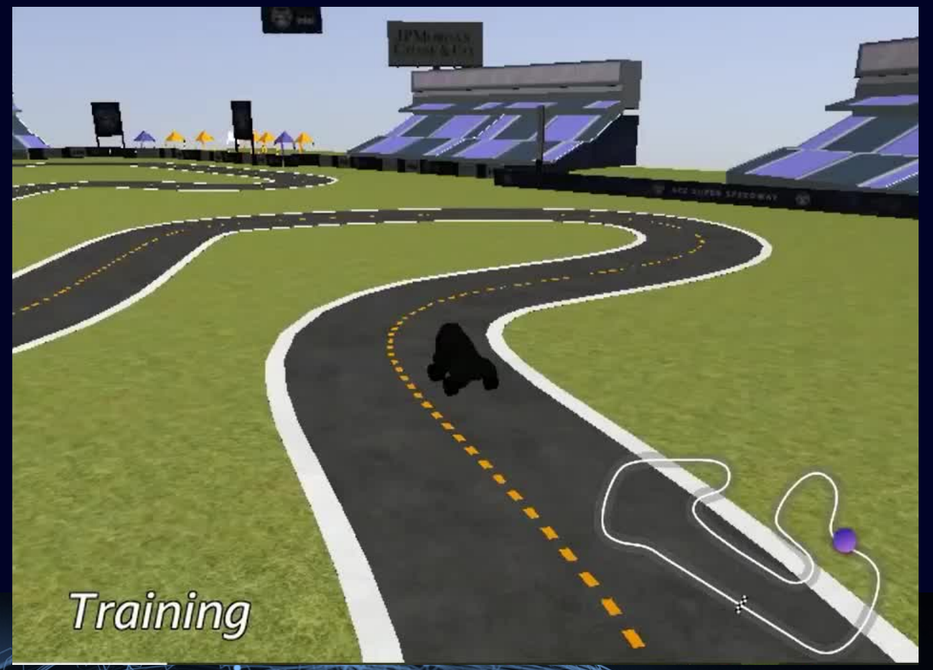
REWARD FUCTION

- The core of reinforcement learning.
- Provides a numerical score based on the state of the environment.
- Incentivize our virtual car to take specific actions to finish a lap as fast as possible

OUR FIRST MODEL

Rewarding the agent to follow center line

```python
def reward_function(params):
    '''
    Example of rewarding the agent to follow center line
    '''

    # Read input parameters
    track_width = params['track_width']
    distance_from_center = params['distance_from_center']

    # Calculate 3 markers that are at varying distances away fro
    marker_1 = 0.1 * track_width
    marker_2 = 0.25 * track_width
    marker_3 = 0.5 * track_width

    # Give higher reward if the car is closer to center line and
    if distance_from_center <= marker_1:
        reward = 1.0
    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

    return float(reward)
```

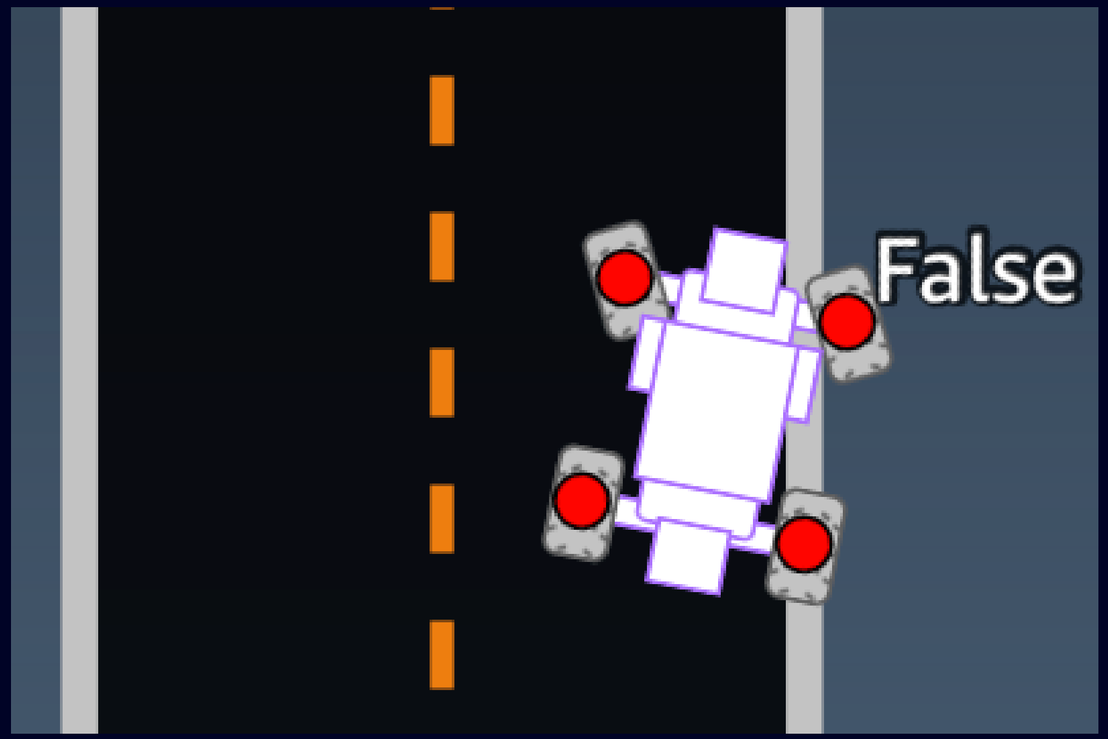


ALL_WHEELS_ON_TRACK

Type: Boolean

Variable:
all_wheels_on_track = params['all_wheels_on_track']

```
# Give higher reward if the car is closer to center line and vice versa
if all_wheels_on_track and distance_from_center <= marker_1:
    reward = 1.0
elif all_wheels_on_track and 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
```

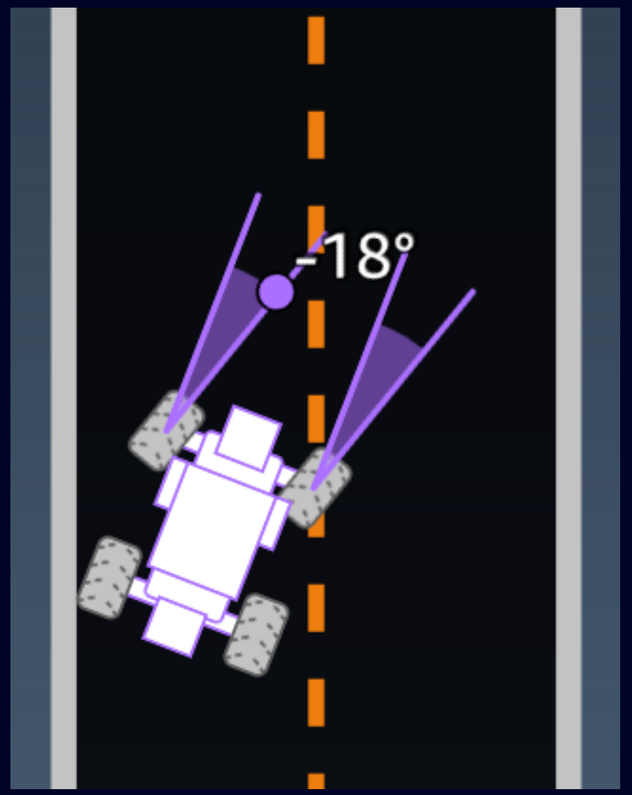


STEERING_ANGLE

Type: float
Range: -30:30

Variable:
abs_steering = abs(params['steering_angle'])

```
# Steering penality threshold, change the numb
ABS_STEERING_THRESHOLD = 15

# Penalize reward if the car is steering too m
if abs_steering > ABS_STEERING_THRESHOLD:
    reward *= 0.8
```



STEPS

Type: int

Variable:
steps = params['steps']
progress = params['progress']

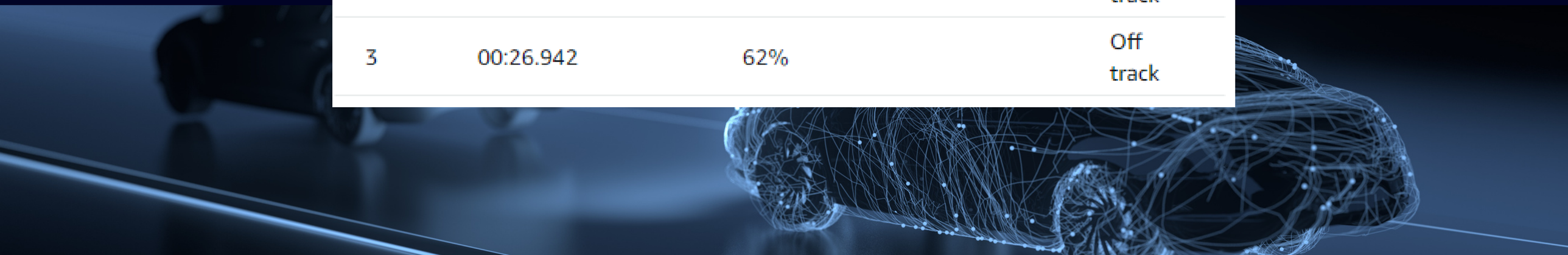
```
# Total num of steps we want the car to finish the lap, it will vary d
TOTAL_NUM_STEPS = 300

# Initialize the reward with typical value
reward = 1.0

# Give additional reward if the car pass every 100 steps faster than e
if (steps % 100) == 0 and progress > (steps / TOTAL_NUM_STEPS) * 100 :
    reward += 10.0
```

OUR FIRST MODEL

Rewarding the agent to follow center line

Evaluation results

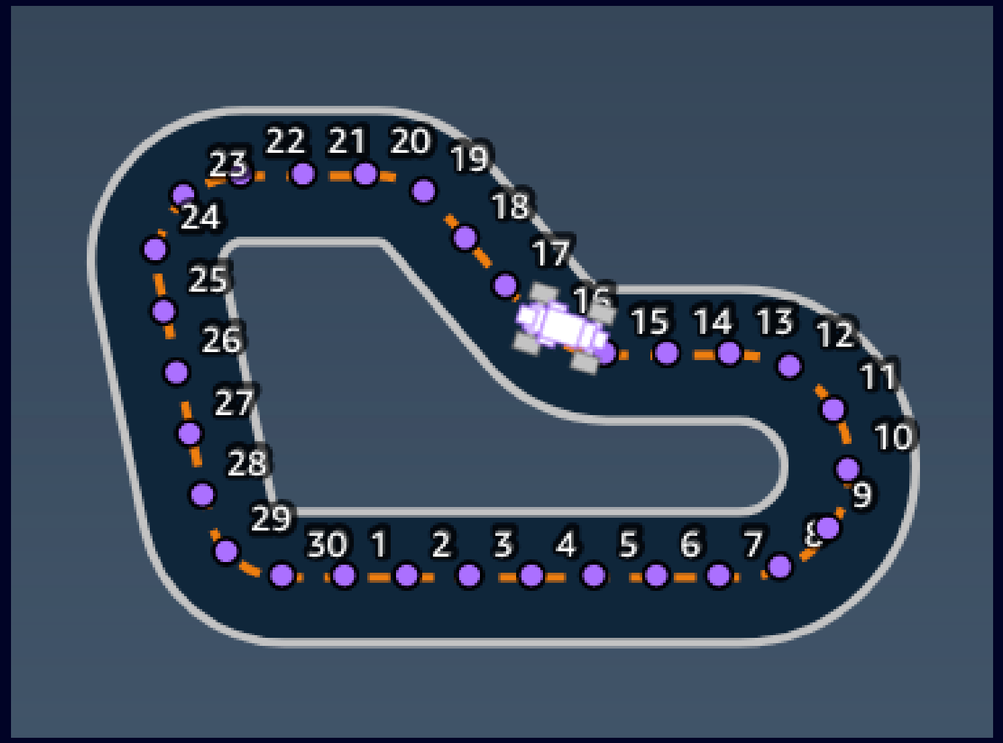
Trial	Time (MM:SS.mmm)	Trial results (% track completed)	Status
1	00:20.057	47%	Off track
2	00:20.474	47%	Off track
3	00:26.942	62%	Off track



WAYPOINTS

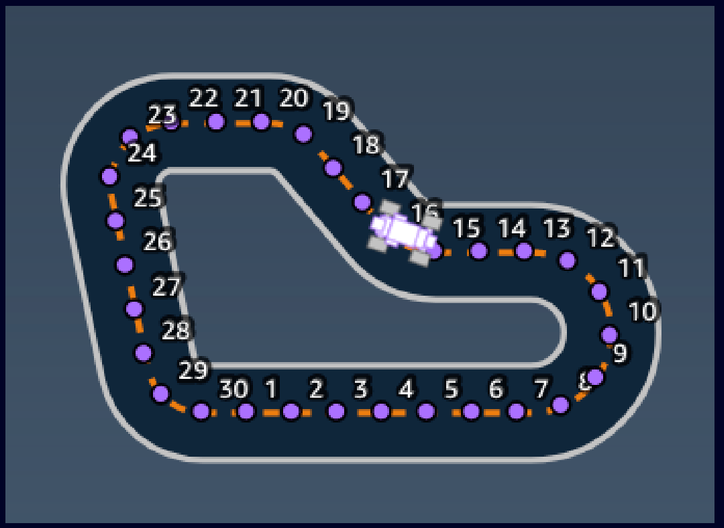
Type: list

Variable:
waypoints = params['waypoints']
closest_waypoints = params['closest_waypoints']
heading = params['heading']



CLOSEST_WAYPOINTS

waypoints = params['waypoints']
closest_waypoints = params['closest_waypoints']
heading = params['heading']




```
# Calculate the direction of the center line based on the closest waypoints
next_point = waypoints[closest_waypoints[1]]
prev_point = waypoints[closest_waypoints[0]]

# Calculate the direction in radius, arctan2(dy, dx), the result is (-pi, pi) in radians
track_direction = math.atan2(next_point[1] - prev_point[1], next_point[0] - prev_point[0])
# Convert to degree
track_direction = math.degrees(track_direction)

# Calculate the difference between the track direction and the heading direction of the car
direction_diff = abs(track_direction - heading)
if direction_diff > 180:
    direction_diff = 360 - direction_diff

# Penalize the reward if the difference is too large
DIRECTION_THRESHOLD = 10.0
if direction_diff > DIRECTION_THRESHOLD:
    reward *= 0.5
```

PROGRESS

Type: float
Range: 0:100

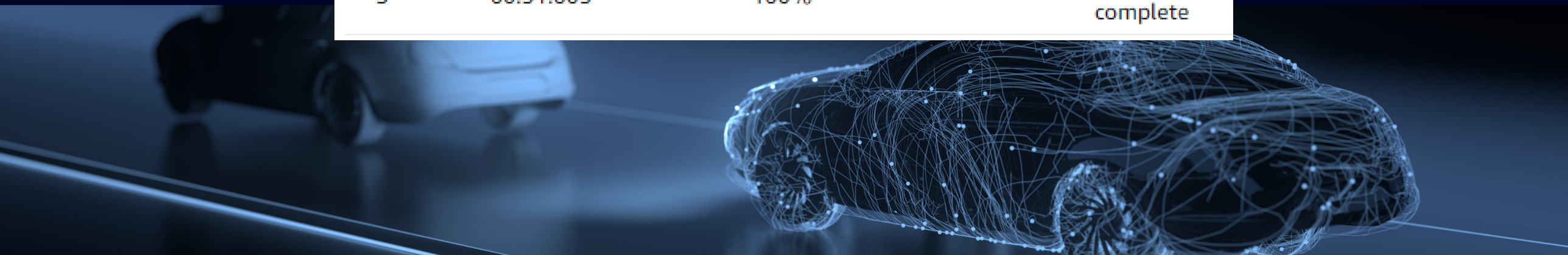
Variable:
progress = params['progress']
is_offtrack=params['is_offtrack']

```
#Get reward if completes the lap
if progress == 100:
    reward += 100
elif is_offtrack:
    reward-=50
```

EVALUATION

Evaluation results

Trial	Time (MM:SS.mmm)	Trial results (% track completed)	Status
1	00:50.254	100%	Lap complete
2	00:31.937	63%	Off track
3	00:51.005	100%	Lap complete



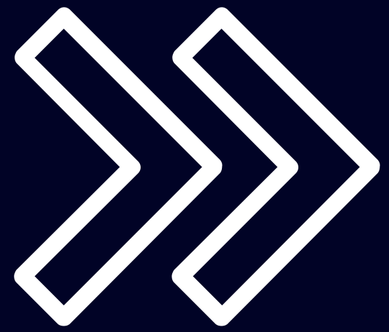
5 MARKRS

```
# Calculate 5 markers that are at varying distances away from the center line
marker_1 = 0.01 * track_width
marker_2 = 0.1 * track_width
marker_3 = 0.2 * track_width
marker_4 = 0.3 * track_width
marker_5 = 0.5 * track_width

# Give higher reward if the car is closer to center line and vice versa
if all_wheels_on_track and distance_from_center <= marker_1:
    reward = 1.0
elif all_wheels_on_track and distance_from_center <= marker_2:
    reward = 0.5
elif distance_from_center <= marker_3:
    reward = 0.2
elif distance_from_center <= marker_4:
    reward = 0.1
elif distance_from_center <= marker_5:
    reward = 0.001
else:
    reward = 1e-3  # likely crashed/ close to off track
```

INCREASE ACTION SPACE SPEED

Action space

No.	Steering angle (°)	Speed (m/s)
0	-30.0	1.25
1	-30.0	2.50
2	-15.0	1.25
3	-15.0	2.50
4	0.0	1.25
5	0.0	2.50
6	15.0	1.25
7	15.0	2.50
8	30.0	1.25
9	30.0	2.50



Action space

No.	Steering angle (°)	Speed (m/s)
0	-30.0	1.10
1	-30.0	1.20
2	-15.0	1.50
3	-15.0	1.90
4	0.0	3.00
5	0.0	2.90
6	15.0	1.70
7	15.0	2.00
8	30.0	1.10
9	30.0	1.20

EVALUATION

Evaluation results

Trial	Time (MM:SS.mmm)	Trial results (% track completed)	Status
1	00:45.129	100%	Lap complete
2	00:44.004	100%	Lap complete
3	00:28.728	64%	Off track

PENALIZE REWARD

Variable:
speed = params['speed']
SPEED_THRESHOLD = 1.5

```
# Penalize reward if the car goes too slow
if speed < SPEED_THRESHOLD:
    reward *= 0.8
```

EVALUATION

Evaluation results

Trial	Time (MM:SS.mmm)	Trial results (% track completed)	Status
1	00:26.195	62%	Off track
2	00:42.800	100%	Lap complete
3	00:42.602	100%	Lap complete

INCREASE ACTION SPACE SPEED & INCREASE SPEED THRESHOLD

Action space

No.	Steering angle (°)	Speed (m/s)
0	-30.0	1.10
1	-30.0	1.10
2	-15.0	2.00
3	-15.0	2.50
4	0.0	3.60
5	0.0	4.00
6	15.0	2.00
7	15.0	2.50
8	30.0	1.10
9	30.0	1.20

```
# Set the speed threshold based your action space
SPEED_THRESHOLD = 2.0
```

REWARD FUCTION

```python
import math

def reward_function(params):
    '''
    Example of rewarding the agent to follow center line
    '''

    # Read input parameters
    all_wheels_on_track = params['all_wheels_on_track']
    speed = params['speed']

    track_width = params['track_width']
    distance_from_center = params['distance_from_center']
    steps = params['steps']
    progress = params['progress']
    is_offtrack=params['is_offtrack']

    abs_steering = abs(params['steering_angle']) # Only need the absolute steering angle

    waypoints = params['waypoints']
    closest_waypoints = params['closest_waypoints']
    heading = params['heading']

    # Total num of steps we want the car to finish the lap, it will vary depends on the track length
    TOTAL_NUM_STEPS = 800

    # Initialize the reward with typical value
    reward = 1.0

    # Set the speed threshold based your action space
    SPEED_THRESHOLD = 2.0

    #Get reward if completes the lap
    if progress == 100:
        reward += 100
    elif is_offtrack:
        reward-=50

    # Calculate the direction of the center line based on the closest waypoints
    next_point = waypoints[closest_waypoints[1]]
    prev_point = waypoints[closest_waypoints[0]]

    # Calculate the direction in radius, arctan2(dy, dx), the result is (-pi, pi) in radians
    track_direction = math.atan2(next_point[1] - prev_point[1], next_point[0] - prev_point[0])
    # Convert to degree
    track_direction = math.degrees(track_direction)

    # Calculate the difference between the track direction and the heading direction of the car
    direction_diff = abs(track_direction - heading)
    if direction_diff > 180:
        direction_diff = 360 - direction_diff

    # Penalize the reward if the difference is too large
    DIRECTION_THRESHOLD = 10.0
    if direction_diff > DIRECTION_THRESHOLD:
        reward *= 0.5

    # Give additional reward if the car pass every 100 steps faster than expected
    if (steps % 100) == 0 and progress > (steps / TOTAL_NUM_STEPS) * 100 :
        reward += 10.0

    # Calculate 5 markers that are at varying distances away from the center line
    marker_1 = 0.01 * track_width
    marker_2 = 0.1 * track_width
    marker_3 = 0.2 * track_width
    marker_4 = 0.3 * track_width
    marker_5 = 0.5 * track_width

    # Give higher reward if the car is closer to center line and vice versa
    if all_wheels_on_track and distance_from_center <= marker_1:
        reward = 1.0
    elif all_wheels_on_track and distance_from_center <= marker_2:
        reward = 0.5
    elif distance_from_center <= marker_3:
        reward = 0.2
    elif distance_from_center <= marker_4:
        reward = 0.1
    elif distance_from_center <= marker_5:
        reward = 0.001
    else:
        reward = 1e-3  # likely crashed/ close to off track

    # Steering penality threshold, change the number based on your action space setting
    ABS_STEERING_THRESHOLD = 15

    # Penalize reward if the car is steering too much
    if abs_steering > ABS_STEERING_THRESHOLD:
        reward *= 0.8

    # Penalize reward if the car goes too slow
    if speed < SPEED_THRESHOLD:
        reward *= 0.8

    return float(reward)
```

FINAL MODEL

- Lap complete X 2
- Best time(MM:SS.mmm): 00:40.539

EVALUATION

Evaluation results

Trial	Time (MM:SS.mmm)	Trial results (% track completed)	Status
1	00:41.397	100%	Lap complete
2	00:27.139	65%	Off track
3	00:40.539	100%	Lap complete

THANK YOU