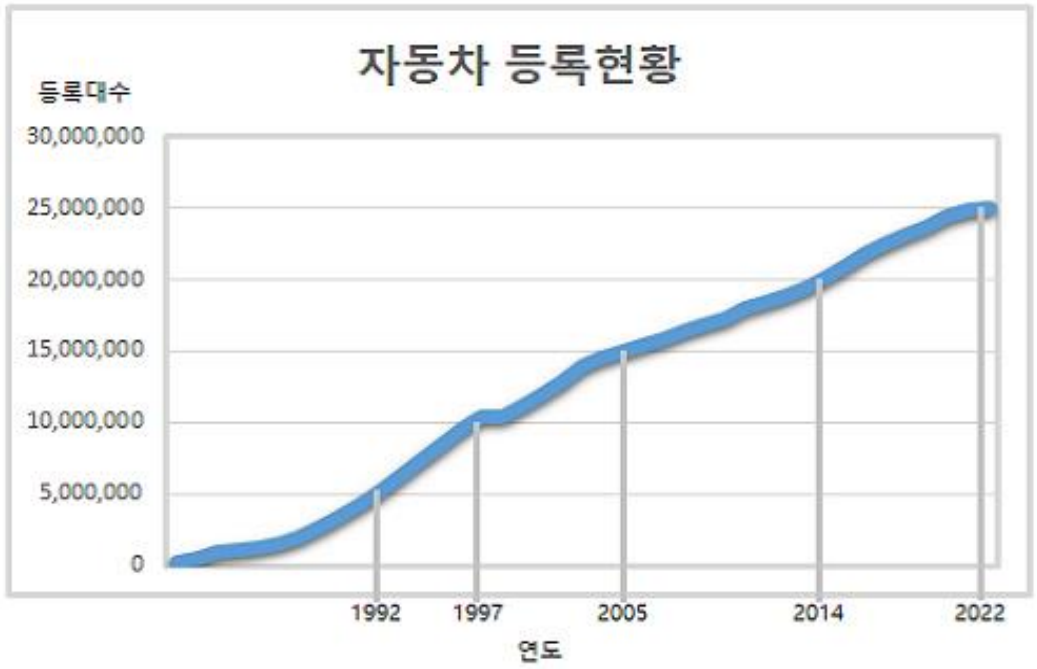
Implementation of Parking Using Reinforcement Learning

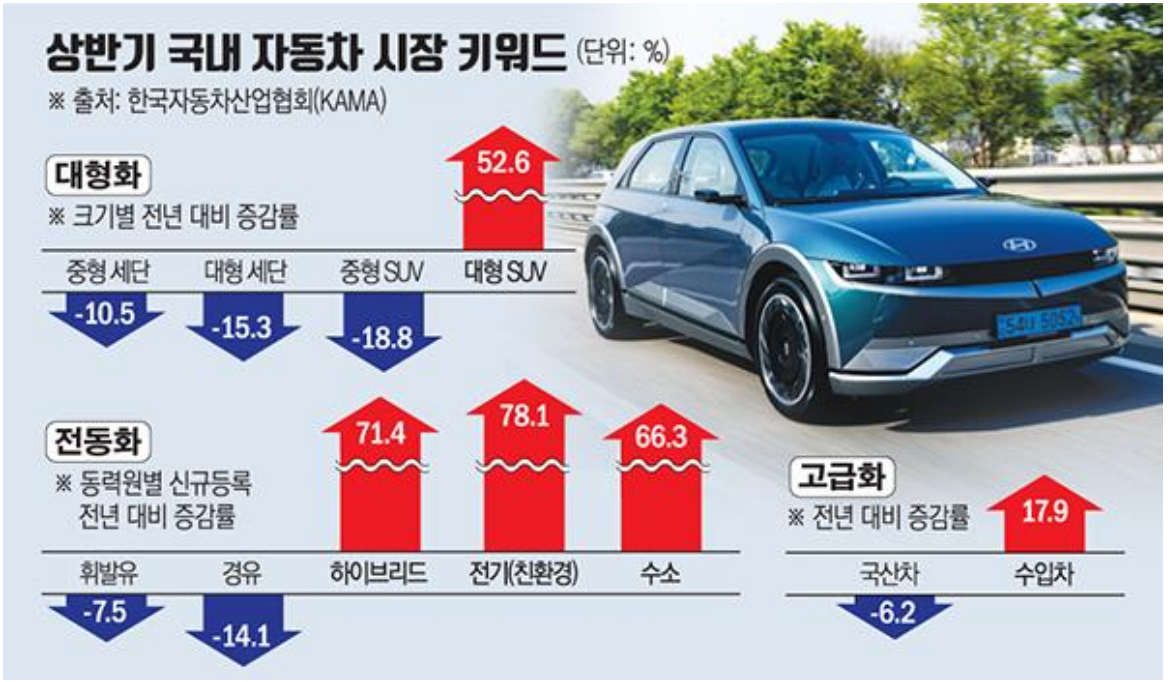
2조 : 김현우
김범진
김창기
안종원

1

Project Outline

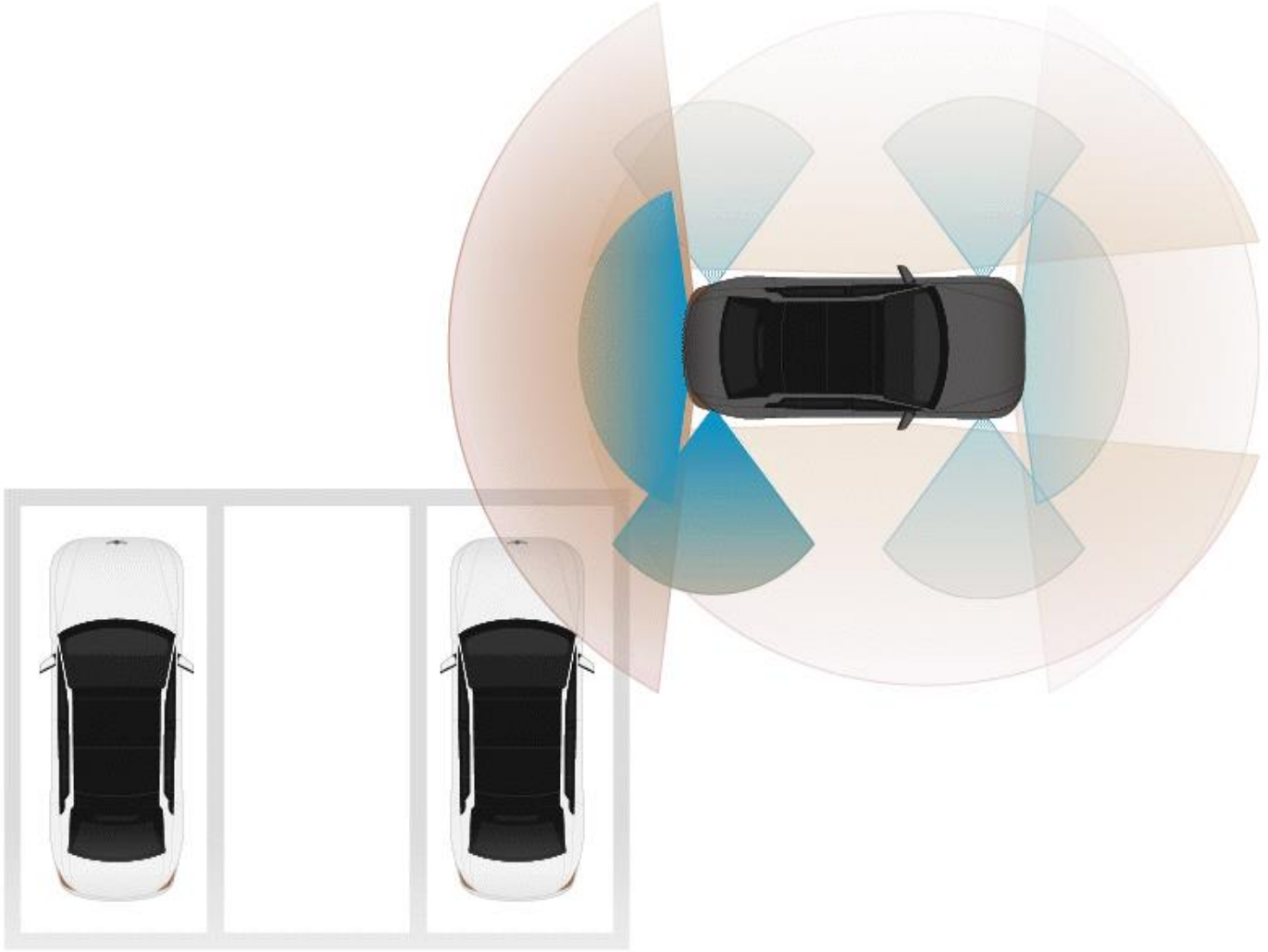
Project Outline





Project Outline

Project Outline

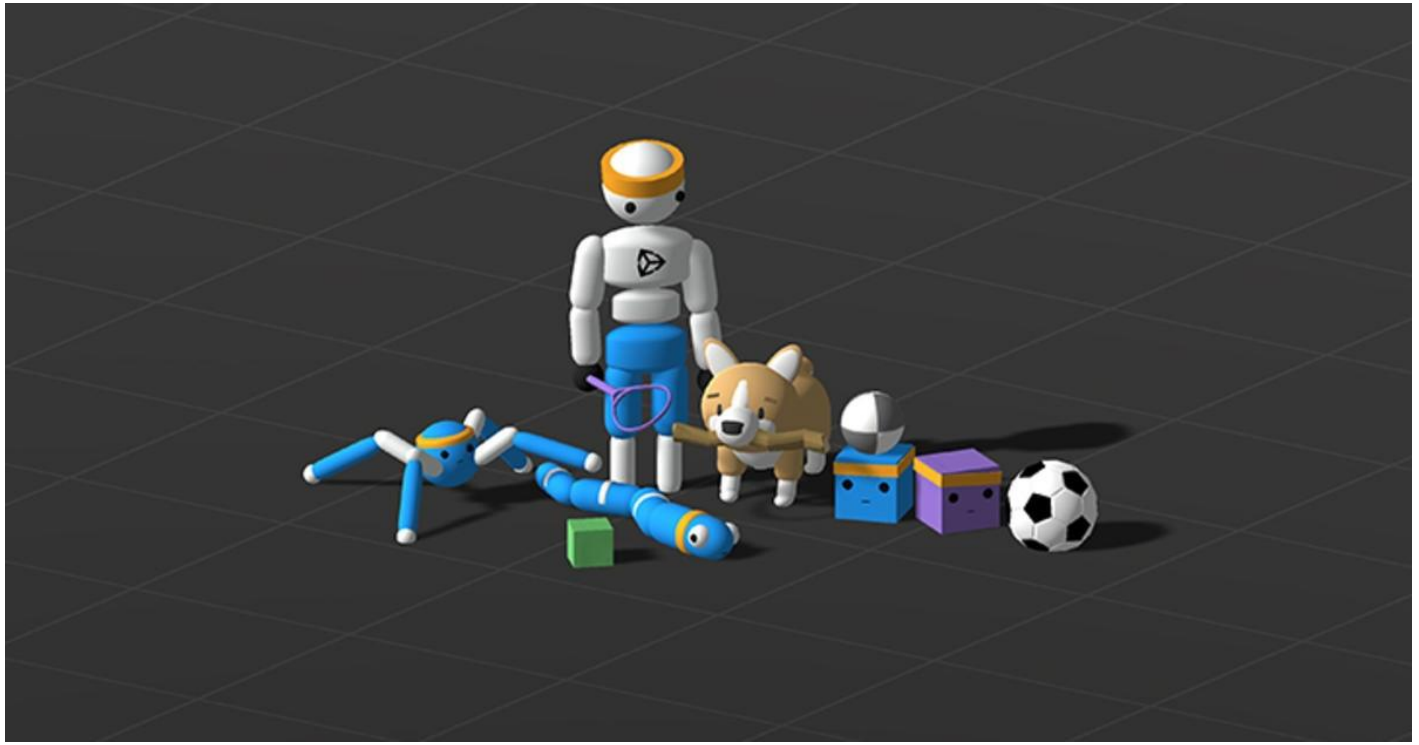


2

Project Implementation

Project Implementation

Use Unity ML-Agent to implement



Project Implementation

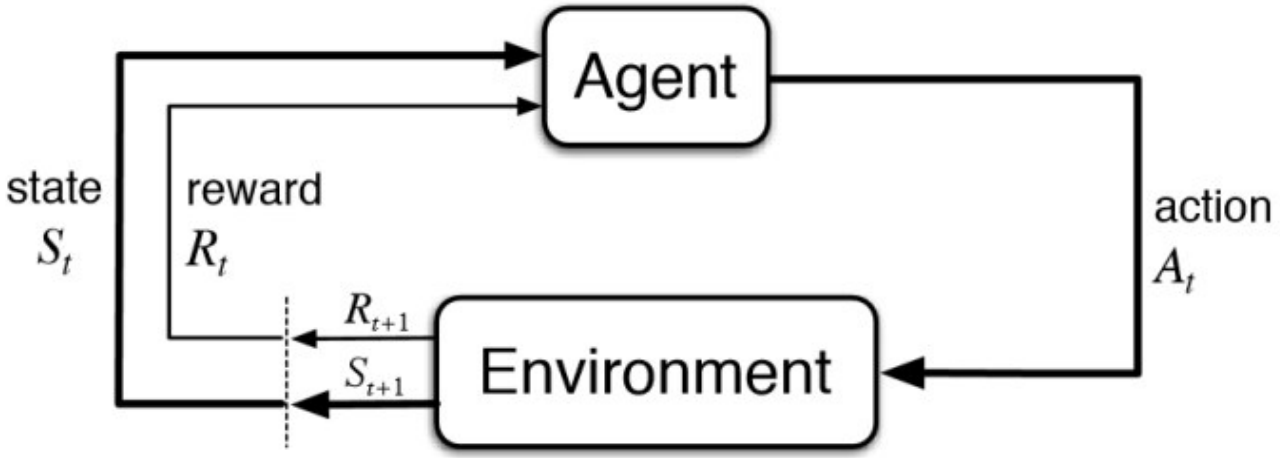
Project Implementation

Use Unity Asset Store's Car asset and town asset.

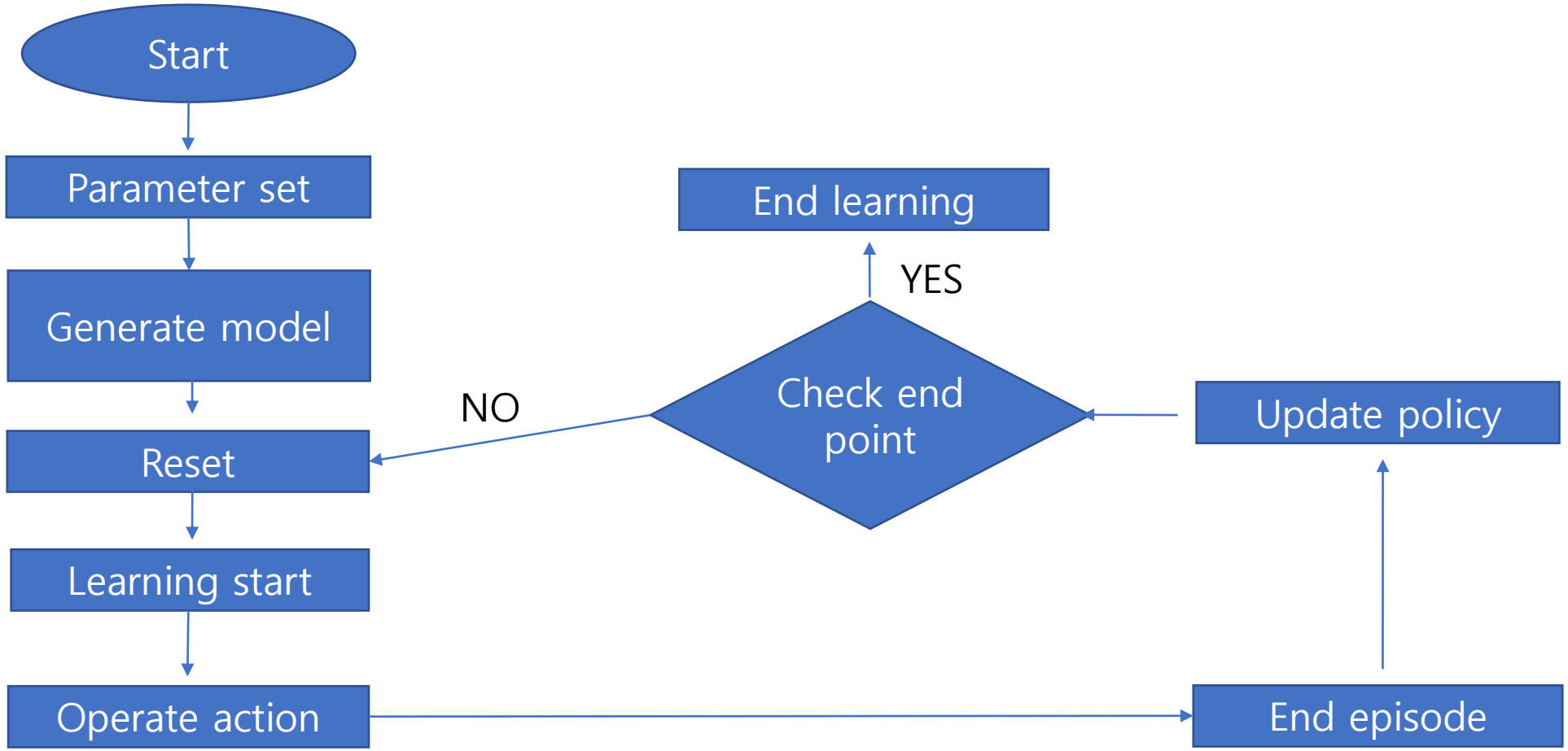
Project Implementation

Reinforcement Learning?

- Agent receives a State from the environment to determine an action
- Environment takes action from the agent, performs, and then outputs the state and the return
- Learn an action that maximizes the expected value by the interaction between the environment and the agent.



Project Implementation



Project Implementation

Reinforcement method : PPO(Proximal Policy Optimization)

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

Project Implementation

Reinforcement method : PPO(Proximal Policy Optimization)

```yaml
behaviors:
  CarBehaviour:
    trainer_type: ppo
    hyperparameters:
      batch_size: 1024
      buffer_size: 5120
      learning_rate: 0.00025
      beta: 0.0020
      epsilon: 0.15
      lambd: 0.95
      num_epoch: 5
      learning_rate_schedule: linear

    network_settings:
      normalize: true
      hidden_units: 264
      num_layers: 3

    reward_signals:
      extrinsic:
        gamma: 0.95
        strength: 0.99

    keep_checkpoints: 15
    checkpoint_interval: 100000
    time_horizon: 264
    max_steps: 100000000
    summary_freq: 100000
```

Project Implementation

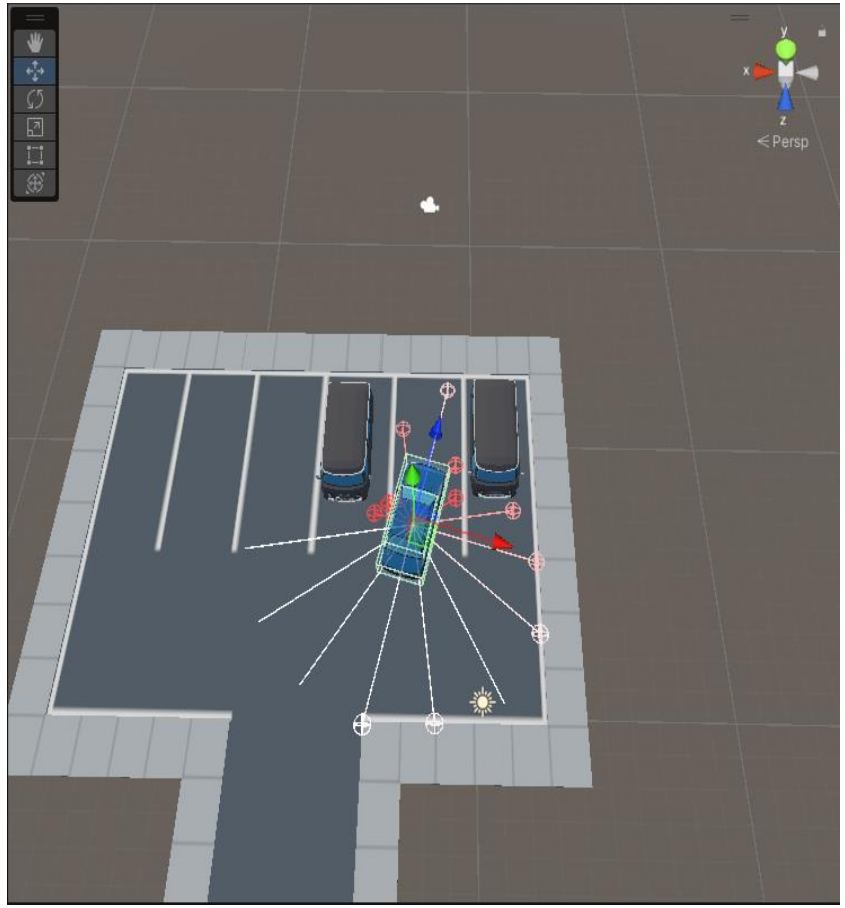
◆ Action : steer, forward, backward

Project Implementation

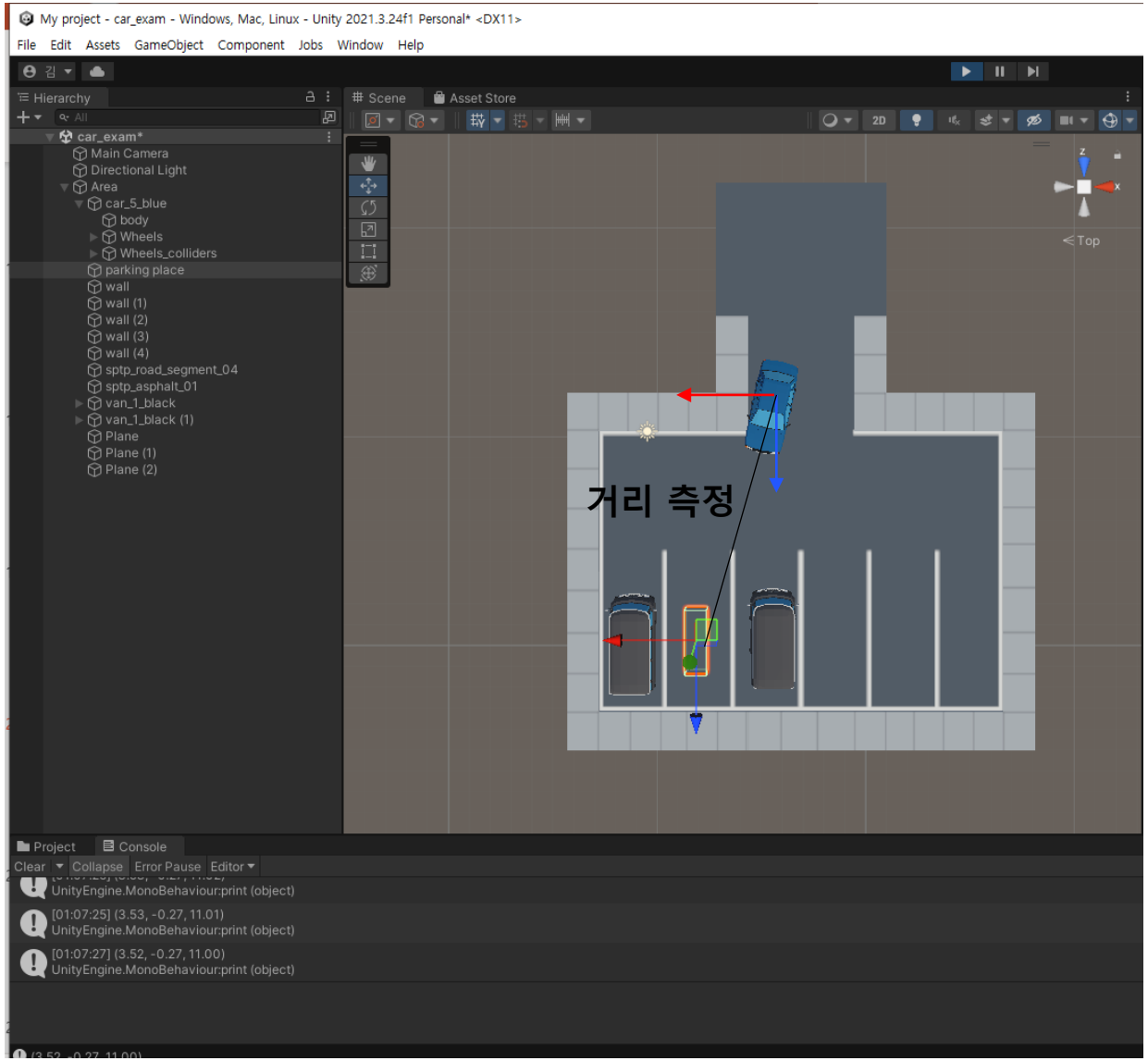
◆ State :

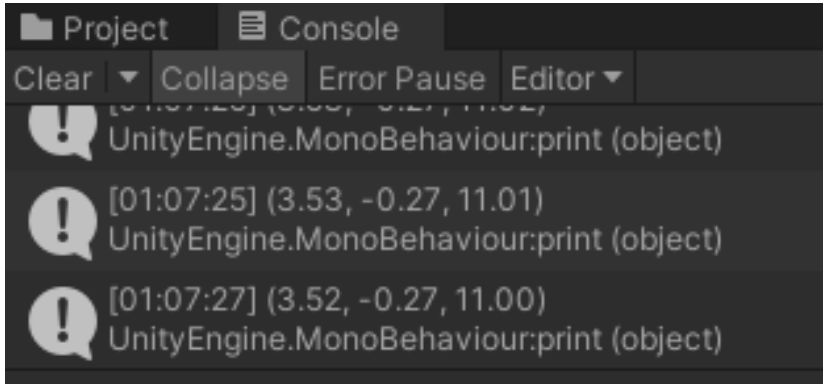
1. Agent's Position – Parking place Position (x,y,z)
2. Agent's angle – Parking place angle
3. Agent's speed

+RayPerceptionSensor 3D sensing

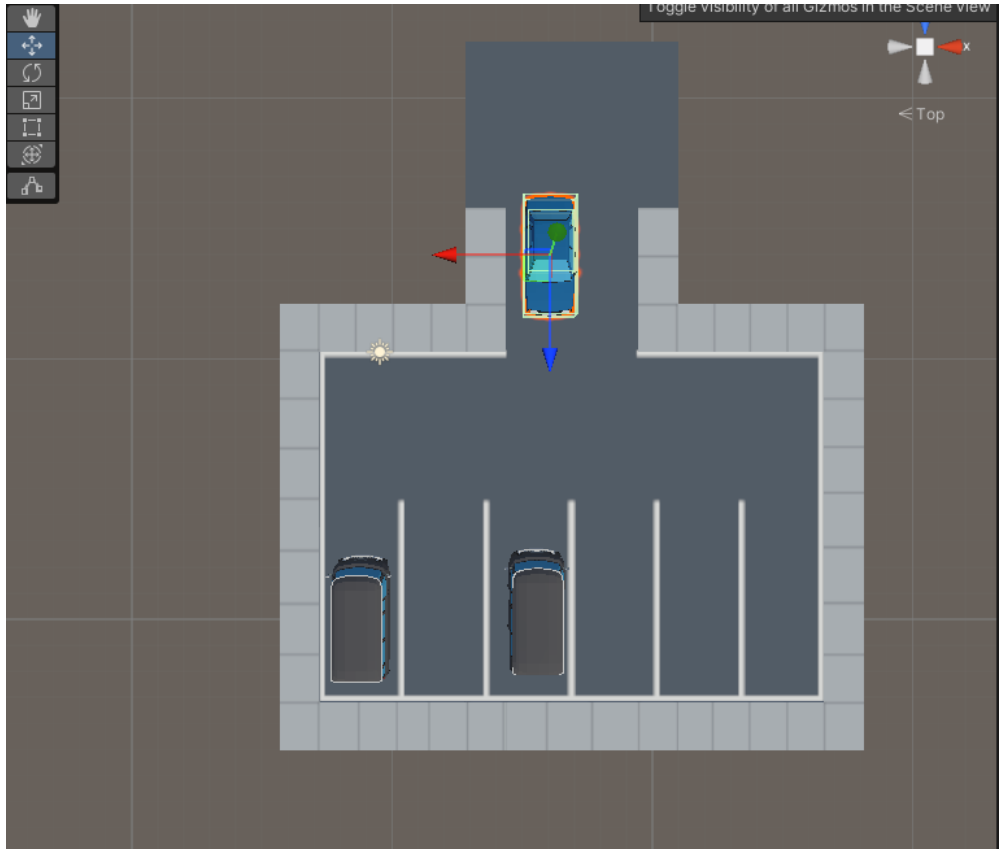


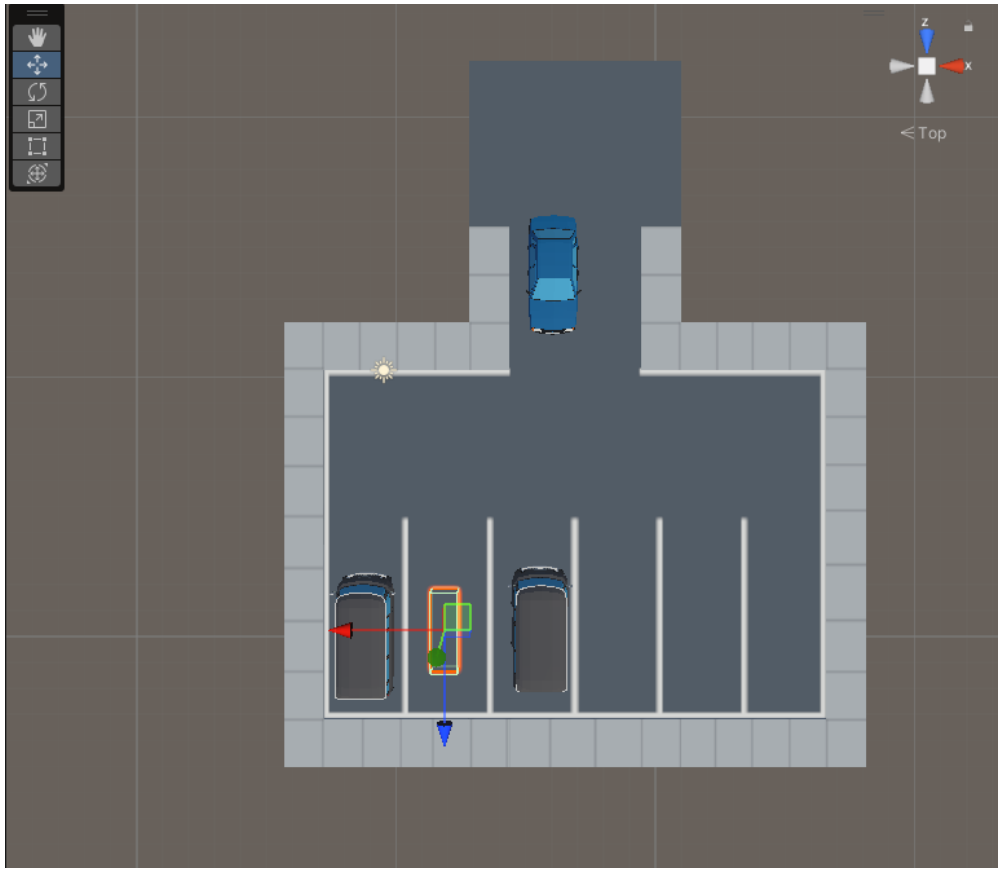
Project Implementation



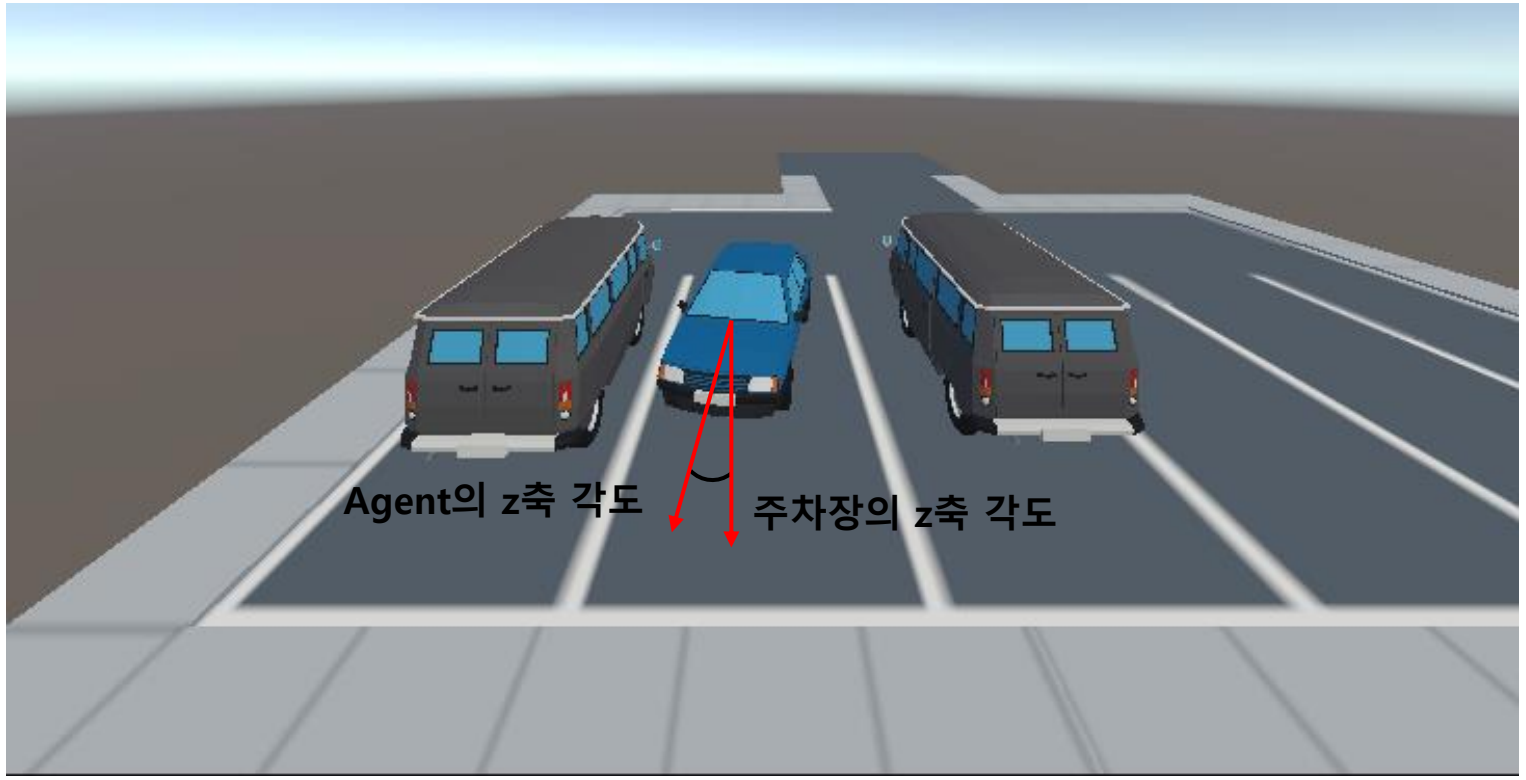


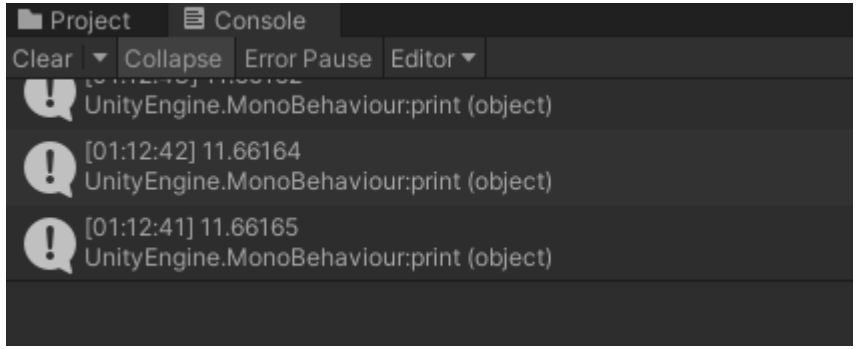
Project Implementation





Project Implementation





Project Implementation

◆ Reward :

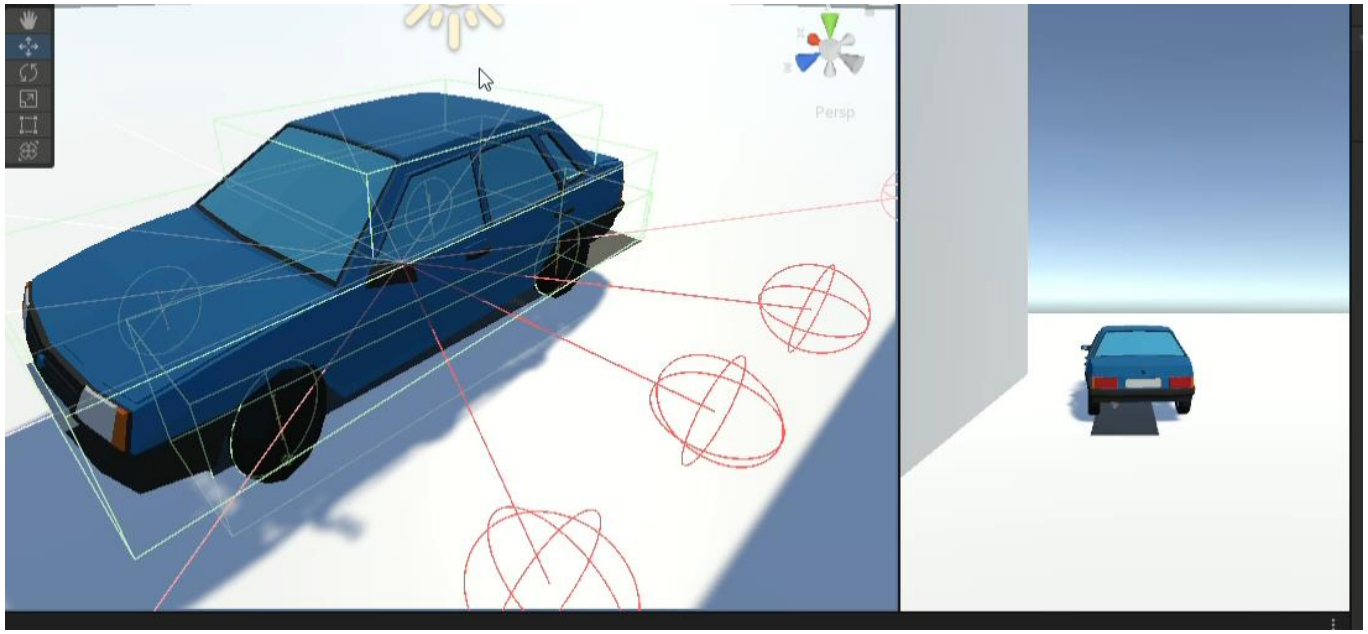
1. Parking success+10reward
2. When Collision –reward depend on Agent speed
3. Agent position – Parking place position
4. Agent angle – Parking place angle

3

Project Result

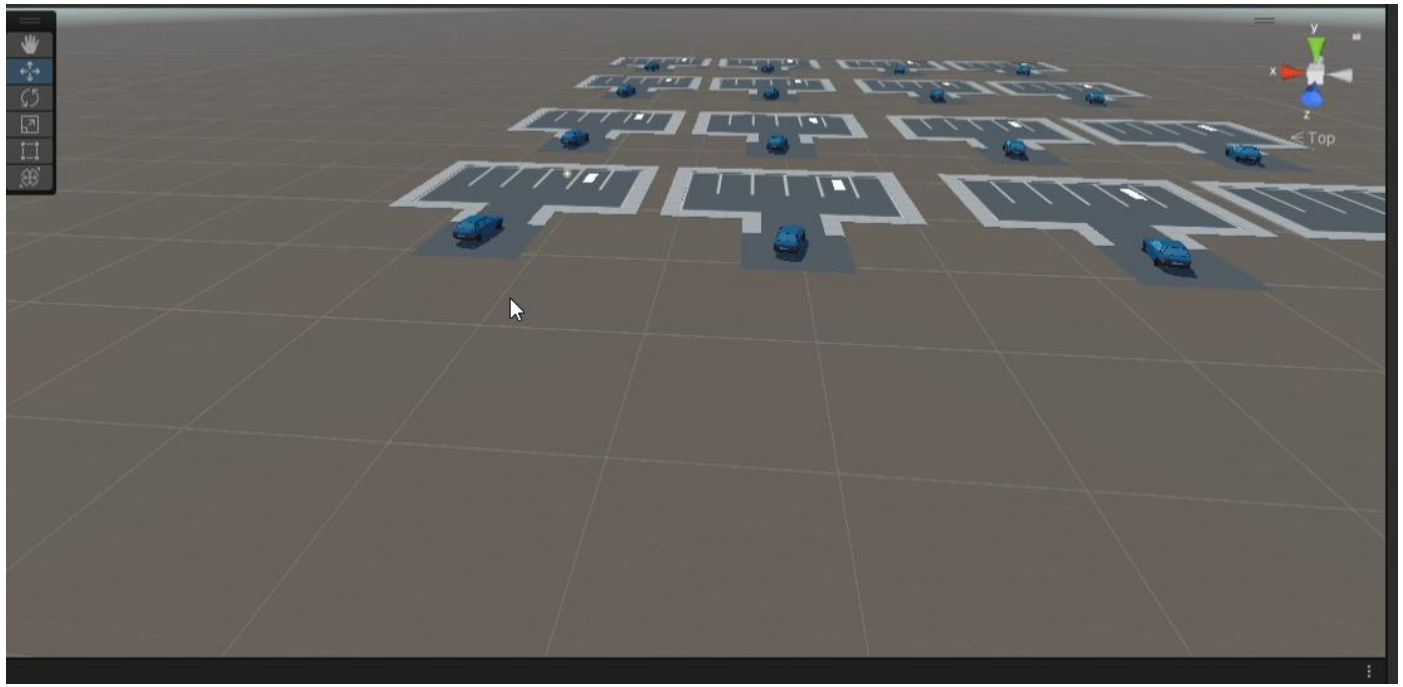
Project Result

Empty Place



Project Result

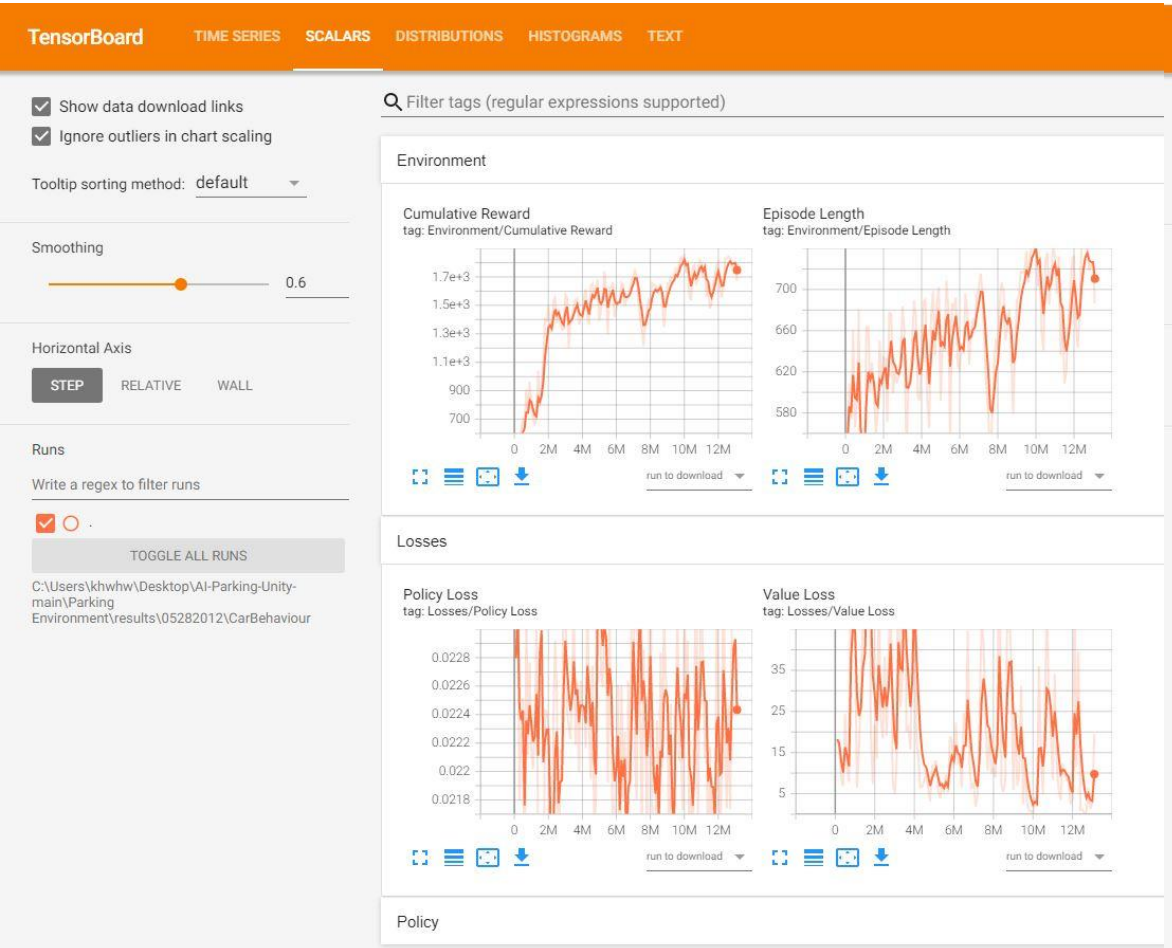
Only Parking line

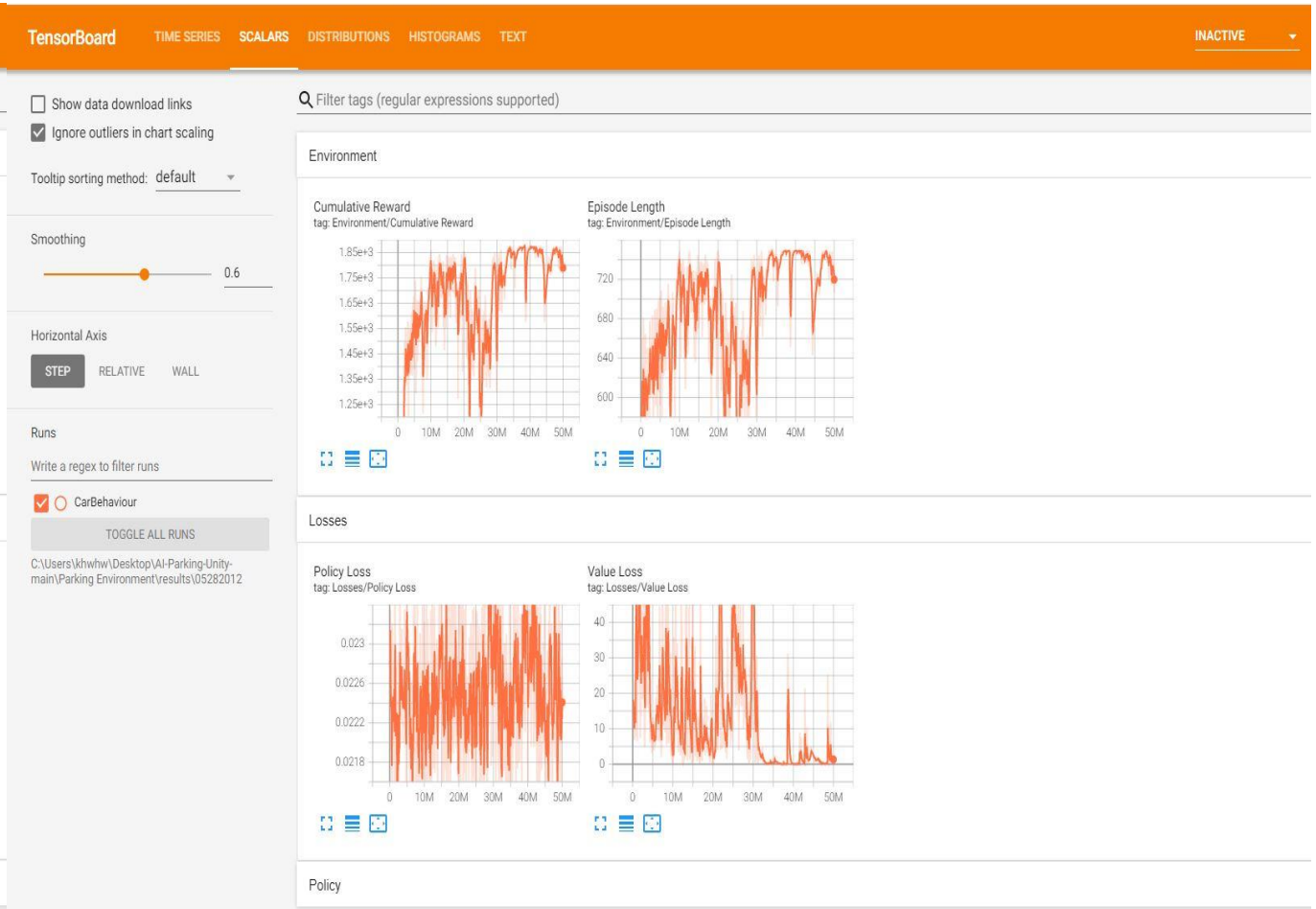


Project Result

In constrained situation(Parking line, other Cars)

Project Result

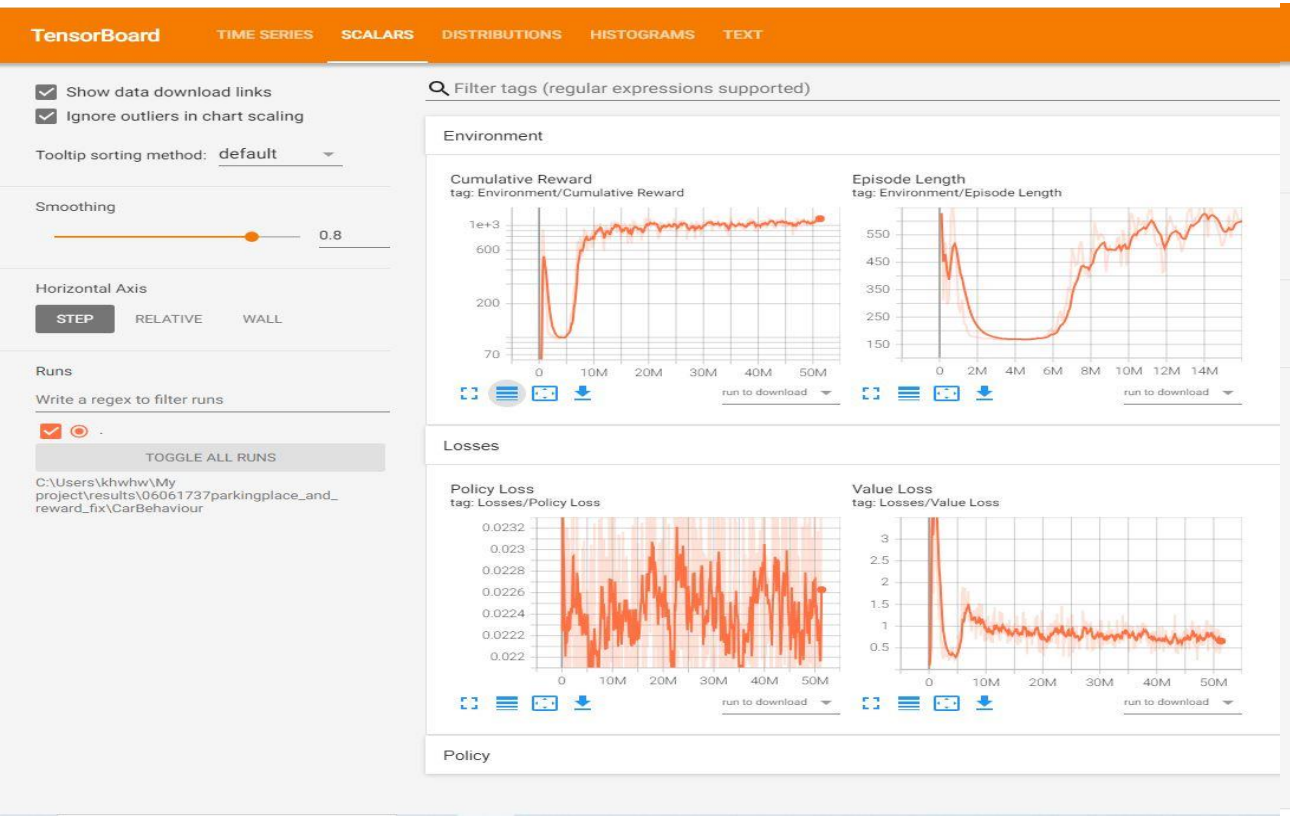


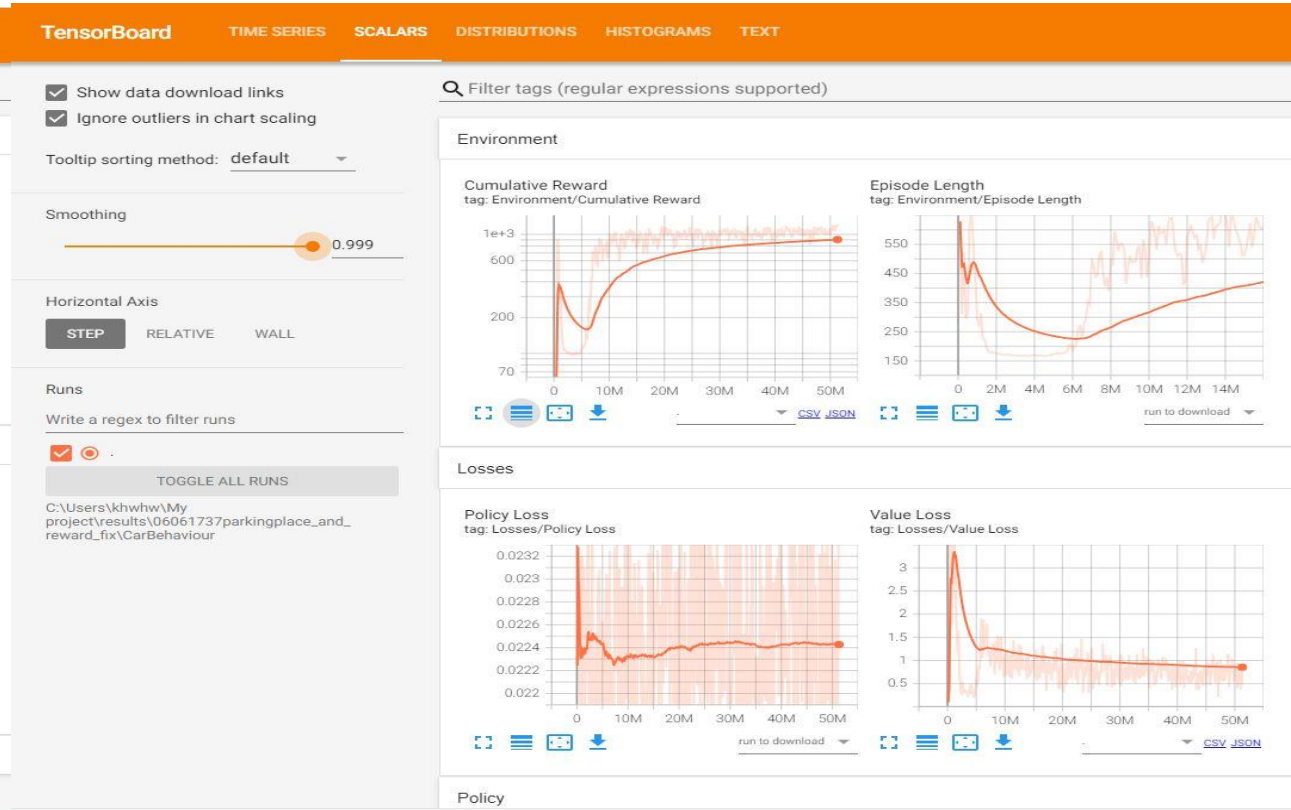


Project Result

Make reward more stably

Project Result





Thank you!