

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Нейроинтерфейсы информационных систем

«Парковка с использованием ИИ в Unity: автоматическая парковка использует набор инструментов Unity ML-Agents, реализующий алгоритмы нейронной сети»

Выполнил студент

Чан Дык Зюи

Группа № P33202

Преподаватель: Штенников Д. Г.

г. Санкт-Петербург

2024

Содержание

Цель работы	3
Программа работы	3
Отчет	5
I. Исследование	5
II. Выполнение	7
Рекомендации	20
Вывод.....	21

Цель работы: Использование набор инструментов Unity ML-Agents для развертывания алгоритмов нейронных сетей.

Программа работы:

Шаг 1: Настройка среды разработки

1. Установка Unity и ML-Agents Toolkit:

- Загрузите и установите Unity Hub и Unity Editor с официального сайта Unity.
- Установите Unity ML-Agents Toolkit, следуя инструкциям из [официальной документации ML-Agents](#).

2. Создание нового проекта:

- Откройте Unity Hub, создайте новый проект и выберите шаблон 3D.

3. Установка необходимых пакетов:

- В Unity откройте "Window" -> "Package Manager" и установите такие пакеты, как ML-Agents, Input System и другие необходимые пакеты.

Шаг 2: Создание симуляционной среды

1. Построение среды:

- Создайте новую сцену и спроектируйте парковочную зону, включая стены, препятствия и парковочные места.

2. Добавление автомобиля:

- Создайте или загрузите модель автомобиля из Unity Asset Store или другого источника.
- Добавьте автомобиль в сцену и настройте такие компоненты, как Rigidbody и Collider.

Шаг 3: Создание агента и академии

1. Создание агента:

- Создайте GameObject для агента (автомобиля) и добавьте компонент **Agent** из ML-Agents.
- Создайте C# скрипт, наследующий от класса **Agent**, и реализуйте методы, такие как **CollectObservations**, **OnActionReceived** и **Heuristic**.

2. Создание академии:

- Создайте GameObject для академии и добавьте компонент **Academy**.

Шаг 4: Определение целей и вознаграждений

1. Определение целей:

- Определите парковочные места на сцене и сделайте их целями для агента.

2. Определение вознаграждений:

- Определите вознаграждения за действия, такие как приближение к парковочному месту, правильная парковка, и штрафы за столкновение с препятствиями или неправильную парковку.

Шаг 5: Обучение модели

1. Настройка обучения:

- Создайте файл конфигурации **.yaml** для настройки параметров обучения, таких как learning rate, максимальное количество шагов и другие параметры.

2. Запуск обучения:

- Используйте команду **mlagents-learn** в терминале для запуска процесса обучения.
- Наблюдайте за процессом обучения и при необходимости корректируйте параметры.

Шаг 6: Тестирование и доработка

1. Тестирование модели:

- После завершения обучения протестируйте модель, запустив симуляцию в Unity.
- Наблюдайте за поведением агента и оценивайте его эффективность.

2. Доработка модели:

- На основе результатов тестирования корректируйте параметры обучения, вознаграждения и компоненты агента для улучшения эффективности.

Шаг 7: Развертывание и публикация

1. Развертывание проекта:

- Упакуйте проект в отдельное приложение или поделитесь им на таких платформах, как Unity Asset Store или GitHub.

2. Написание документации и руководства:

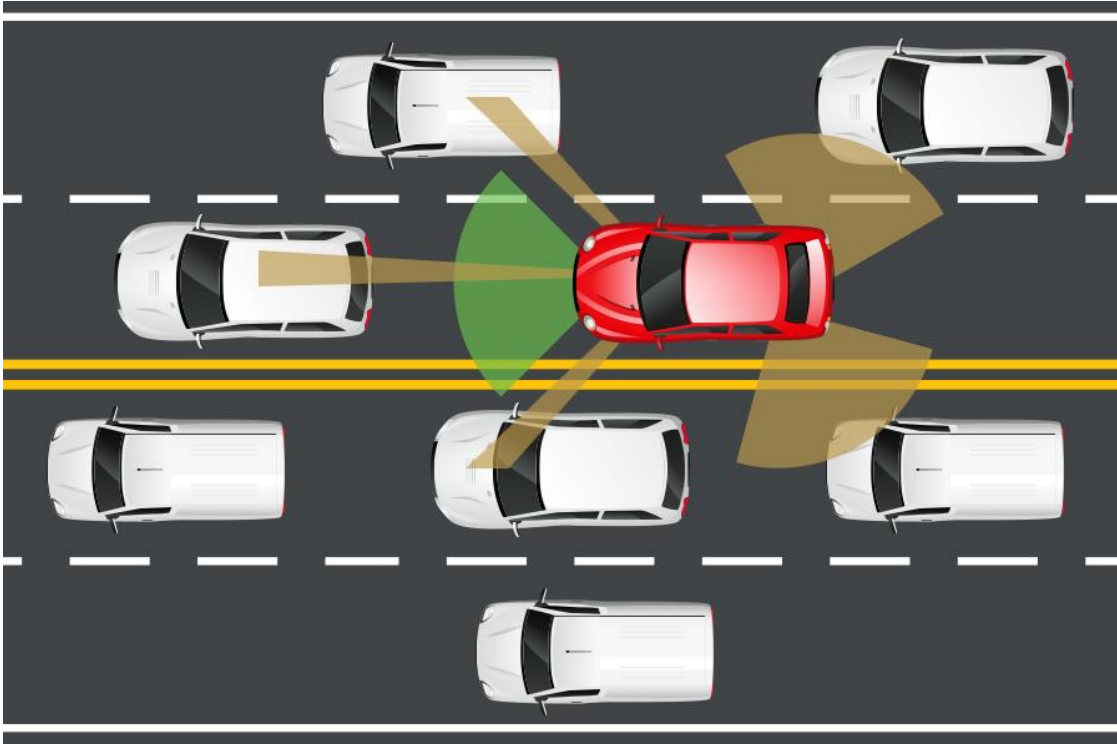
- Напишите документацию по использованию и установке проекта для других пользователей.

Отчет:

I. Исследование

1. Проблема – предотвращение столкновений

- Существующие системы беспилотного вождения используют датчики для предотвращения столкновений.
- В будущем все автомобили могут быть автономными и общаться напрямую.
- Можем ли мы использовать такую связь для повышения безопасности парковки?



2. Проблема – парковка

- Автостоянки – это плотная, опасная среда с множеством сценариев.
- Естественно, мы можем смоделировать автостоянку как мультиагентную систему и использовать RL.
- Исследовать влияние общения, конкуренции и сотрудничества.



3. Задний план

- Автономное вождение — активная область исследований и приложений.
- «Deep RL позволяет решать мультиагентные задачи реальной сложности» [1]
- Недавние работы по теме (21 год):
- Моделирование самостоятельной парковки автомобиля с использованием RL и Unity [2]
- Динамическое предотвращение препятствий и планирование пути на основе RL [3]
- Удивительная эффективность PPO в кооперативных многоагентных играх [4]
- Мультиагентный RL: опрос [1]
- Современная работа, ориентированная на будущее
- Предполагает группы автономных автомобилей.
- Предполагает технологические возможности

4. Мотивация

Благодаря этой работе мы можем:

- Лучше понять производительность и ограничения систем самостоятельной парковки на основе RL и их технологические требования.
- Определить поведенческие модели, возникающие в системах самостоятельной парковки на основе RL.
- Оцените социальную совместимость систем самостоятельной парковки на основе RL.
- Повысьте безопасность и эффективность парковки
- Улучшенный дизайн систем самостоятельной парковки

5. Метод

- Смоделируйте каждого агента с помощью MDP
- Реализуйте MDP, используя Unity в качестве моделирования.
- Обучайте и оценивайте алгоритмы RL с помощью Python и Unity
- Используйте PPO [1], современный алгоритм RL на основе глубокого обучения.
- Используйте систему пакетных вычислений Warwick для обучения в больших масштабах

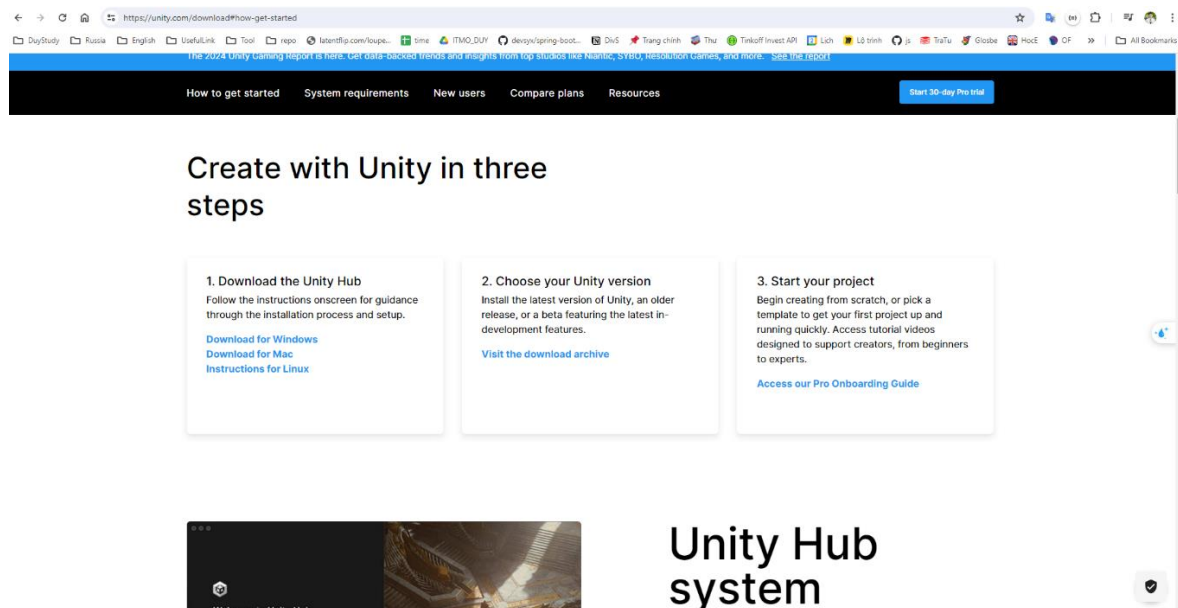
6. МДП

- Довольно стандартный базовый MDP, используемый для робототехники и беспилотного вождения.
- Расширьте нашу проблему:
 - Обмен/отслеживание состояния: найдите состояние фиксированного количества близлежащих автомобилей для предотвращения столкновений и планирования.
 - Выбор и сообщение целей – например, указание
 - Награды за участие в конкурсах и сотрудничестве.
 - Различные функции безопасности...

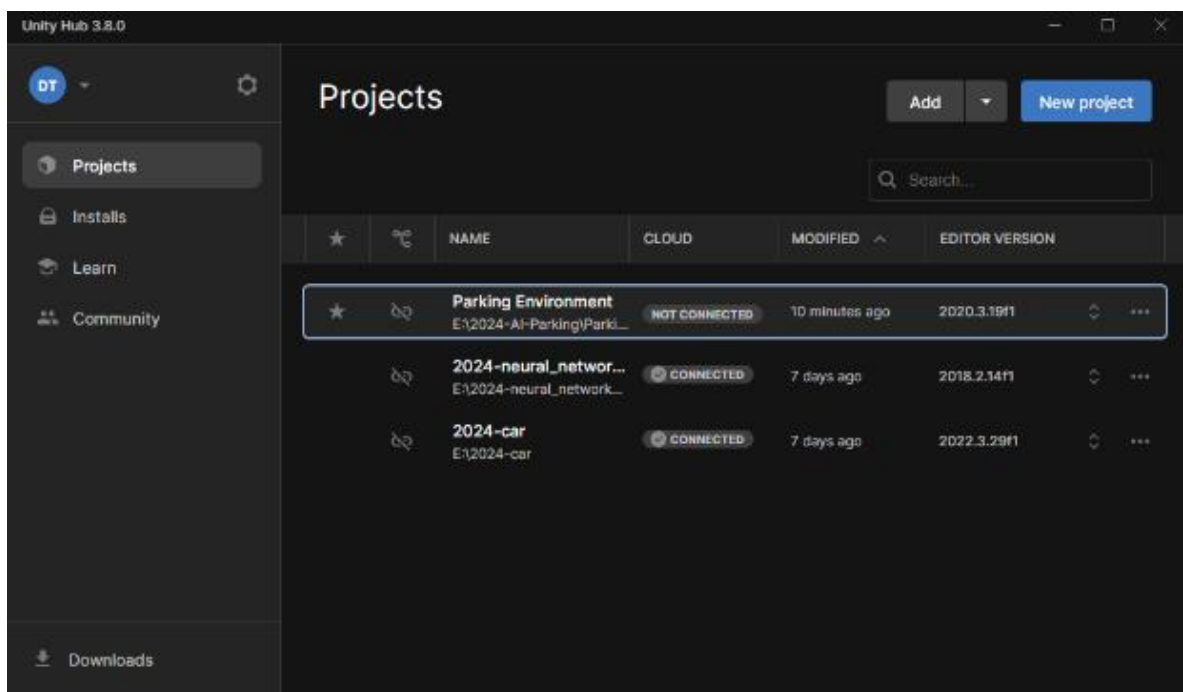
II. Выполнение

Шаг 1: Настройка среды разработки

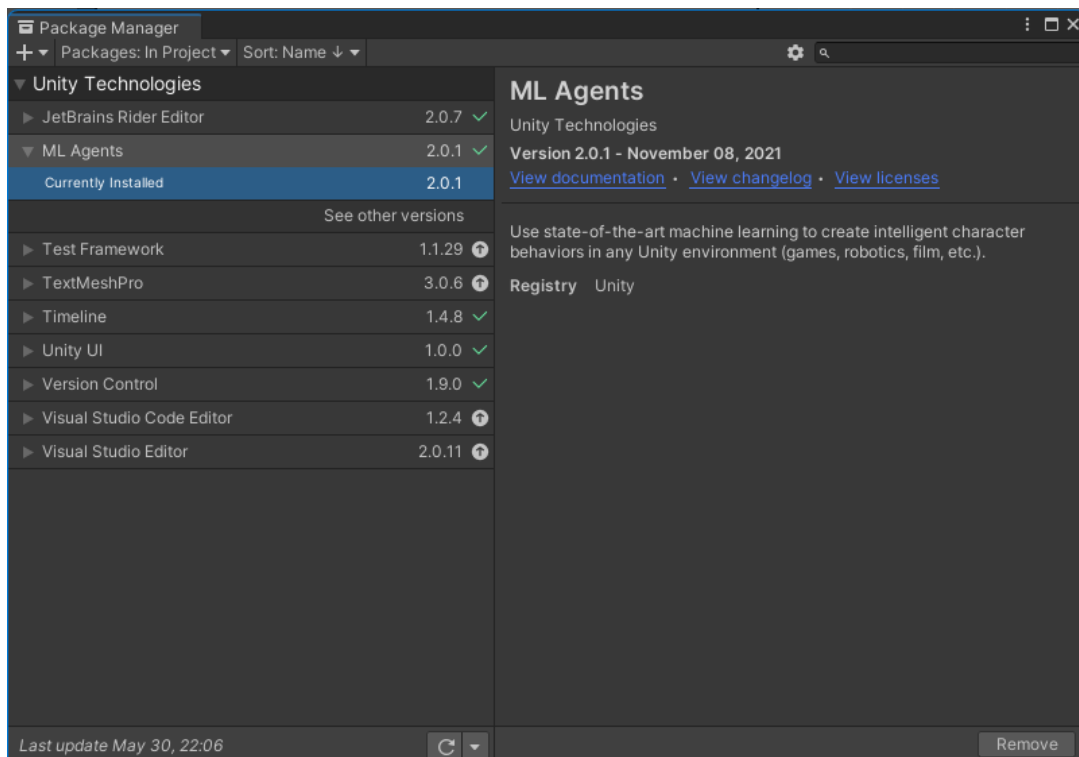
1. Установка Unity и ML-Agents Toolkit:



2. Создание нового проекта:

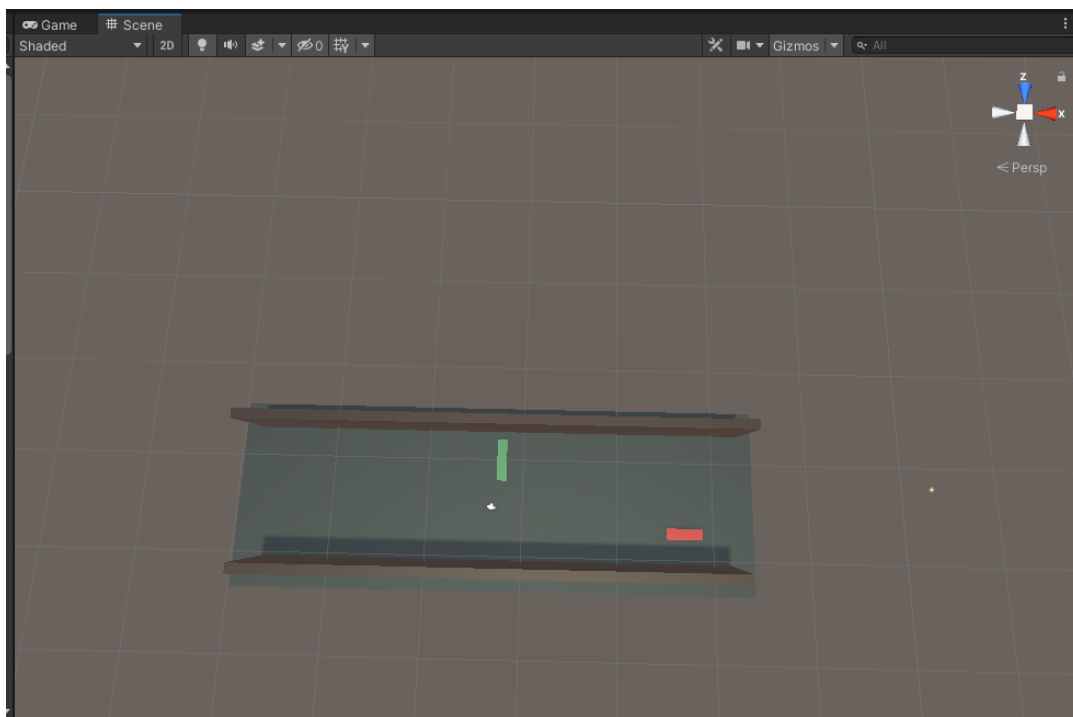


3. Установка необходимых пакетов:

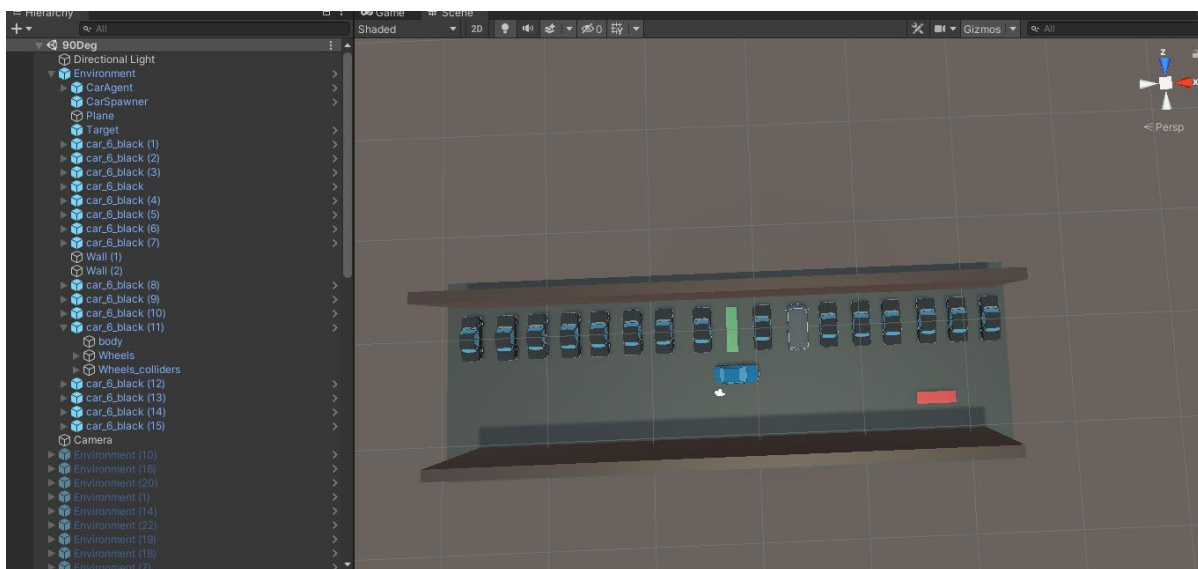


Шаг 2: Создание симуляционной среды

1. Построение среды:



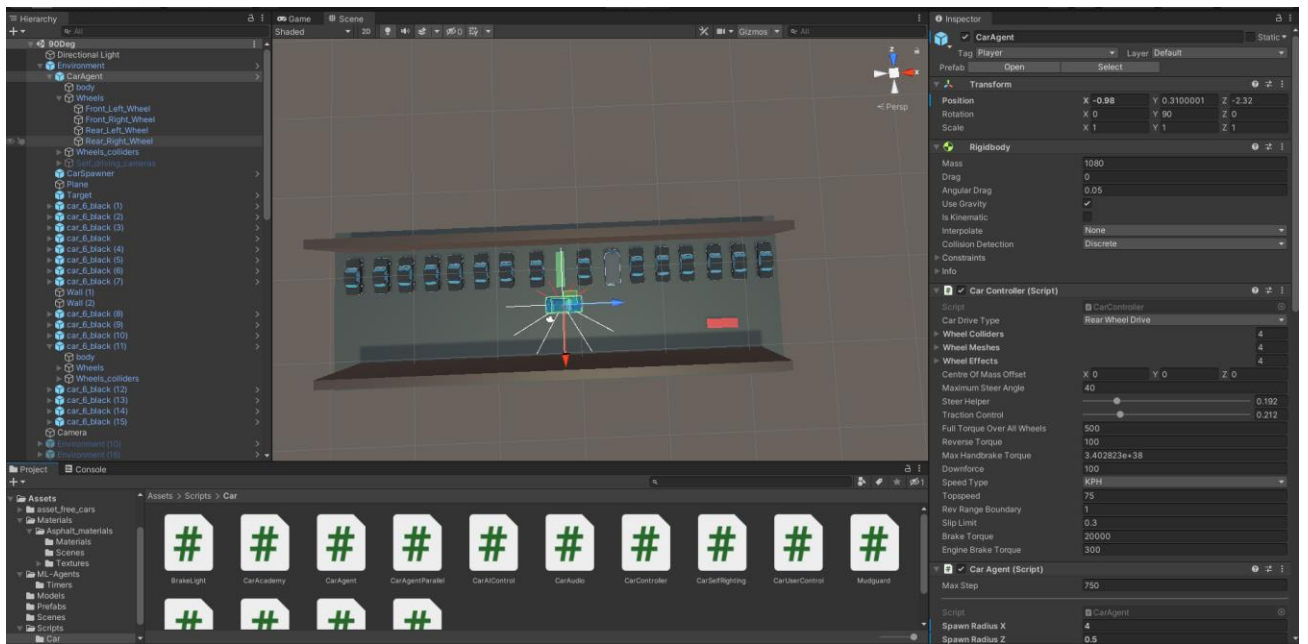
2. Добавление автомобиля:



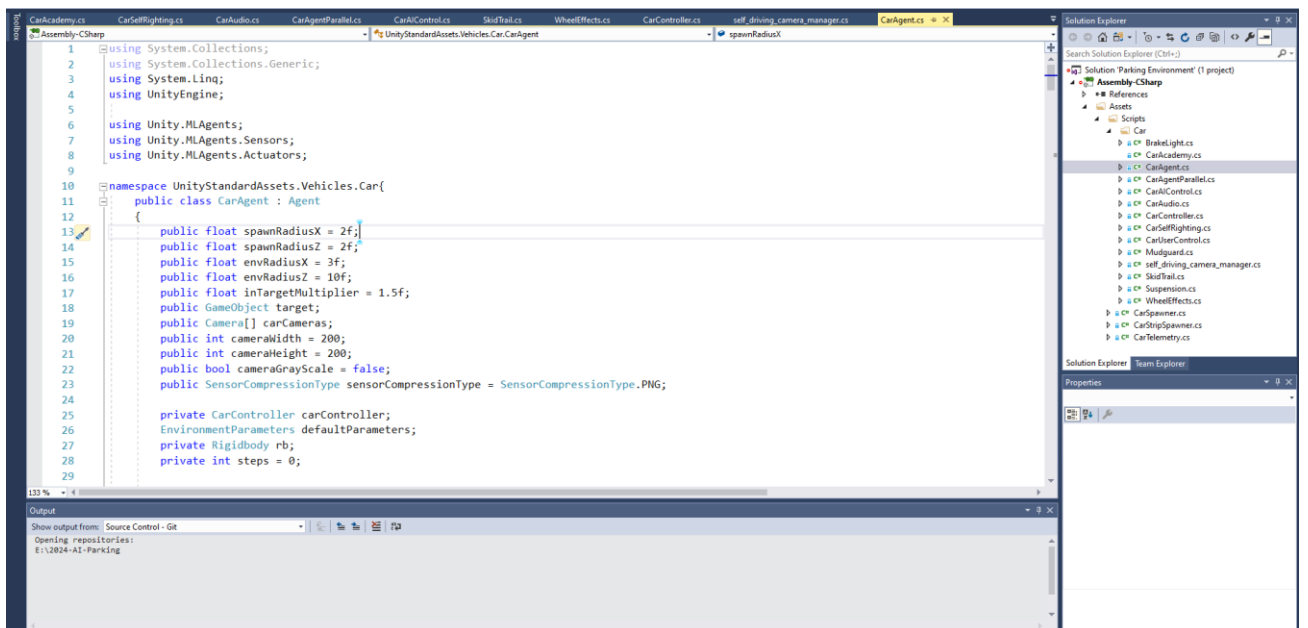
Шаг 3: Создание агента и академии

1. Создание агента:

- Создайте GameObject для агента (автомобиля) и добавьте компонент **Agent** из ML-Agents.



- Создайте C# скрипт, наследующий от класса **Agent**, и реализуйте методы, такие как **CollectObservations**, **OnActionReceived** и **Heuristic**.



2. Создание академии:

- Создайте GameObject для академии и добавьте компонент **Academy**.


```

103 private void FindParkingSpot(){
104     var RpMeasurements = RayPerceptionMeasurements();
105
106     int LeftLikelihoodScore = 0;
107     int RightLikelihoodScore = 0;
108
109     // First check if the left and right perpendicular sensors are detecting a long range
110     if(RpMeasurements.RDistL[2] > 0.9f){
111         LeftLikelihoodScore += 1;
112     }
113     if(RpMeasurements.RDistR[2] > 0.9f){
114         RightLikelihoodScore += 1;
115     }
116
117     // Sum the distances of the sensors for each side to check which side is just seeing space
118     if(RpMeasurements.RDistL.Sum() < RpMeasurements.RDistR.Sum()){
119         LeftLikelihoodScore += 1;
120     }
121     else{
122         RightLikelihoodScore += 1;
123     }
124
125     // Check if sensor observations are symmetrical. This indicates the agent is in the middle of the parking space
126     float RayDiff1 = Mathf.Abs(RpMeasurements.RDistL[0] - RpMeasurements.RDistL[4]);
127     float RayDiff2 = Mathf.Abs(RpMeasurements.RDistL[1] - RpMeasurements.RDistL[3]);
128     float TotalRayDiff = RayDiff1 + RayDiff2;
129
130     if(TotalRayDiff < 0.1f){
131         LeftLikelihoodScore += 1;
132     }
133
134     RayDiff1 = Mathf.Abs(RpMeasurements.RDistR[0] - RpMeasurements.RDistR[4]);
135     RayDiff2 = Mathf.Abs(RpMeasurements.RDistR[1] - RpMeasurements.RDistR[3]);
136     TotalRayDiff = RayDiff1 + RayDiff2;
137
138     if(TotalRayDiff < 0.1f){
139         RightLikelihoodScore += 1;
140     }
141
142 }
143
144 // Can be refactored
145 // If one of the sides met all the requirements, the agent is in the middle of a space
146 if(LeftLikelihoodScore == 3){
147     // Validate the spot is large enough
148     float PredictedSpace = (RpMeasurements.RDistL[1] * Mathf.Cos(60*Mathf.Deg2Rad)) + (RpMeasurements.RDistL[3] * Mathf.Cos(60*Mathf.Deg2Rad));
149     // Distances are in normalised units, so multiply by the ray length to get the actual distance
150     PredictedSpace *= 7;
151
152     if(PredictedSpace > 3f){
153         isLookingForSpot = false;
154         isPositioning = true;
155         predictedSpotSize = PredictedSpace;
156         detectedSpotLocation = new Vector3(transform.position.x, transform.position.y, transform.position.z);
157         Debug.Log("Found spot left");
158     }
159 }
160 else if(RightLikelihoodScore == 3){
161     // Validate the spot is large enough
162     float PredictedSpace = (RpMeasurements.RDistR[1] * Mathf.Cos(60*Mathf.Deg2Rad)) + (RpMeasurements.RDistR[3] * Mathf.Cos(60*Mathf.Deg2Rad));
163     // Distances are in normalised units, so multiply by the ray length to get the actual distance
164     PredictedSpace *= 7;
165
166     if(PredictedSpace > 3f){
167         isLookingForSpot = false;
168         isPositioning = true;
169         predictedSpotSize = PredictedSpace;
170         detectedSpotLocation = new Vector3(transform.position.x, transform.position.y, transform.position.z);
171         Debug.Log("Found spot right");
172     }
173 }
174 else{
175     isLookingForSpot = true;
176 }
177
178 }
179

```

Шаг 5: Обучение модели

1. Настройка обучения:

- Создайте файл конфигурации **.yaml** для настройки параметров обучения, таких как learning rate, максимальное количество шагов и другие параметры.

```
1. behaviors:
2.   CarBehaviour:
3.     trainer_type: ppo
4.     hyperparameters:
5.       batch_size: 1024
6.       buffer_size: 5120
7.       learning_rate: 0.00035
8.       beta: 0.0025
9.       epsilon: 0.3
10.      lambda: 0.95
11.      num_epoch: 5
12.      learning_rate_schedule: linear
13.
14.   network_settings:
15.     normalize: true
16.     hidden_units: 264
17.     num_layers: 3
18.
19.   reward_signals:
20.     extrinsic:
21.       gamma: 0.95
22.       strength: 0.99
23.     gail:
24.       strength: 0.3
25.       demo_path: Demos/90dgRS3s.demo
26.       use_actions: false
27.
28.   behavioral_cloning:
29.     demo_path: Demos/90dgRS3s.demo
30.     steps: 750000
31.     strength: 0.4
32.
33.   keep_checkpoints: 15
34.   checkpoint_interval: 1000000
35.   time_horizon: 264
36.   max_steps: 50000000
37.   summary_freq: 100000
38.   threaded: true
39.
40.   CarBehaviourSAC:
41.     trainer_type: sac
42.     hyperparameters:
43.       batch_size: 128
44.       buffer_size: 50000
```

```

45.     learning_rate: 0.00035
46.     buffer_init_steps: 5000
47.     init_entcoef: 0.5
48.     steps_per_update: 1
49.     learning_rate_schedule: constant
50.
51.     network_settings:
52.         normalize: true
53.         hidden_units: 128
54.         num_layers: 3
55.
56.     reward_signals:
57.         extrinsic:
58.             gamma: 0.75
59.             strength: 0.99
60.         gail:
61.             strength: 0.3
62.         demo_path: Demos/90dgRev3s.demo
63.         use_actions: false
64.
65.     behavioral_cloning:
66.         demo_path: Demos/90dgRev3s.demo
67.         steps: 300000
68.         strength: 0.3
69.
70.     keep_checkpoints: 15
71.     time_horizon: 264
72.     max_steps: 30000000
73.     summary_freq: 20000
74.     threaded: false

```

CarBehaviour (PPO)

- **trainer_type: ppo**
 - Выбор алгоритма обучения. PPO выбран за его стабильность и эффективность при обучении сложных политик.
- **hyperparameters**
 - **batch_size:** Размер batch определяет количество образцов данных, используемых при каждом обновлении весов. Большой размер batch (1024) помогает лучше усреднять градиенты и уменьшать шум в процессе обучения.
 - **buffer_size:** Размер буфера определяет количество сохраняемых опытов перед обновлением. Большой буфер (5120) позволяет хранить больше опытов, помогая модели учиться в различных ситуациях.

- **learning_rate**: Скорость обучения определяет степень изменения весов при каждом обновлении. Низкое значение (0.00035) делает процесс обучения медленным и стабильным, избегая быстрого и нестабильного обучения.
- **beta**: Коэффициент энтропии помогает поддерживать разнообразие в действиях агента, избегая излишней уверенности в своих действиях и потери способности к исследованию.
- **epsilon**: Этот параметр определяет степень допустимых изменений политики в PPO. Большие значения позволяют больше изменений, меньшие поддерживают стабильность.
- **lambda**: Этот параметр влияет на точность Generalized Advantage Estimation (GAE). Значение 0.95 часто выбирается для баланса между точностью и стабильностью.
- **num_epoch**: Количество проходов по всем данным при каждом обновлении помогает эффективно оптимизировать веса.
- **learning_rate_schedule**: График изменения скорости обучения. Линейный график позволяет скорости обучения постепенно снижаться со временем, что делает процесс обучения медленнее при приближении к оптимуму.
- **network_settings**
 - **normalize**: Нормализация входных данных помогает стабилизировать процесс обучения, уменьшая изменчивость входных данных.
 - **hidden_units**: Количество нейронов в каждом скрытом слое определяет сложность нейронной сети. 264 скрытых нейрона обеспечивают достаточную способность для обучения сложным шаблонам.
 - **num_layers**: Количество скрытых слоев определяет глубину нейронной сети. 3 скрытых слоя достаточно для моделирования сложных зависимостей, не усложняя при этом модель.
- **reward_signals**
 - **extrinsic**
 - **gamma**: Коэффициент дисконтирования помогает модели фокусироваться на краткосрочных или долгосрочных наградах. Gamma 0.95 балансирует между краткосрочными и долгосрочными наградами.
 - **strength**: Сила внешнего сигнала награды. 0.99 делает внешние награды значимыми для процесса обучения.
 - **gail**
 - **strength**: Сила сигнала награды GAIL помогает агенту учиться на предоставленных демо.

- **demo_path:** Путь к демо данным, предоставляющим примеры желаемого поведения.
- **use_actions:** Определяет, использовать ли действия из демо в процессе обучения.
- **behavioral_cloning**
 - **demo_path:** Путь к демо данным помогает агенту учиться на примерах поведения.
 - **steps:** Количество шагов обучения для Behavioral Cloning. Большое количество шагов (750000) помогает агенту детально изучить демо данные.
 - **strength:** Сила сигнала Behavioral Cloning. 0.4 делает награды от клонирования значимыми.
- **Другие параметры**
 - **keep_checkpoints:** Сохранение 15 контрольных точек помогает гарантировать возможность восстановления на различных этапах процесса обучения.
 - **checkpoint_interval:** Интервал между сохранениями контрольных точек помогает управлять объемом памяти и частотой восстановления.
 - **time_horizon:** Максимальная длина каждого эпизода взаимодействия агента помогает контролировать длину сохраняемых последовательностей действий.
 - **max_steps:** Максимальное количество шагов обучения помогает контролировать время и ресурсы, затрачиваемые на обучение.
 - **summary_freq:** Частота записи данных обучения помогает детально отслеживать прогресс.
 - **threaded:** Использование многопоточности помогает увеличить производительность обучения.

CarBehaviourSAC (SAC)

- **trainer_type: sac**
 - Выбор алгоритма обучения. SAC выбран за его стабильность и способность оптимизировать сложные поведения.
- **hyperparameters**
 - **batch_size:** Меньший размер batch (128) для более быстрой реакции на изменения в среде.
 - **buffer_size:** Большой размер буфера (50000) для хранения большего количества опытов и обучения в различных ситуациях.

- **learning_rate**: Скорость обучения помогает регулировать степень изменения весов при каждом обновлении.
- **buffer_init_steps**: Количество шагов инициализации для буфера, чтобы обеспечить достаточное количество данных перед началом обучения.
- **init_entcoef**: Начальный коэффициент энтропии помогает контролировать разнообразие в действиях агента.
- **steps_per_update**: Количество шагов между обновлениями помогает контролировать частоту обучения.
- **learning_rate_schedule**: График скорости обучения constant удерживает скорость обучения неизменной на протяжении всего процесса обучения.
- **network_settings**
 - **normalize**: Нормализация входных данных помогает стабилизировать процесс обучения.
 - **hidden_units**: Количество нейронов в каждом скрытом слое определяет сложность нейронной сети.
 - **num_layers**: Количество скрытых слоев определяет глубину нейронной сети.
- **reward_signals**
 - **extrinsic**:
 - **gamma**: Коэффициент дисконтирования помогает фокусироваться на краткосрочных или долгосрочных наградах.
 - **strength**: Сила внешнего сигнала награды.
 - **gail**:
 - **strength**: Сила сигнала награды GAIL помогает учиться на предоставленных демо.
 - **demo_path**: Путь к демо данным, предоставляющим примеры желаемого поведения.
 - **use_actions**: Определяет, использовать ли действия из демо в процессе обучения.
- **behavioral_cloning**
 - **demo_path**: Путь к демо данным помогает агенту учиться на примерах поведения.
 - **steps**: Количество шагов обучения для Behavioral Cloning.
 - **strength**: Сила сигнала Behavioral Cloning.

- **Другие параметры**

- **keep_checkpoints:** Сохранение 15 контрольных точек помогает гарантировать возможность восстановления на различных этапах процесса обучения.
- **time_horizon:** Максимальная длина каждого эпизода взаимодействия агента помогает контролировать длину сохраняемых последовательностей действий.
- **max_steps:** Максимальное количество шагов обучения помогает контролировать время и ресурсы, затрачиваемые на обучение.
- **summary_freq:** Частота записи данных обучения помогает детально отслеживать прогресс.
- **threaded:** Решает, использовать ли многопоточность для увеличения производительности обучения.

2. Запуск обучения:

- Используйте команду **mlagents-learn** в терминале для запуска процесса обучения.
- Наблюдайте за процессом обучения и при необходимости корректируйте параметры.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 1362ms.
(base) PS E:\2024-AI-Parking\Parking Environment> conda create -n mlagents2 python=3.8

(base) PS E:\2024-AI-Parking\Parking Environment> conda activate mlagents2
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> conda init
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> pip install mlagents
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> pip install protobuf==3.20.3
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c
pytorch
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> pip install packaging
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> pip install "numpy<1.24"
(mlagents2) PS E:\2024-AI-Parking\Parking Environment> mlagents-learn trainer_settings.yaml --run-id="run_build" --num-e
nvs=5 --envs=Builds --no-graphics

(mlagents2) PS E:\2024-AI-Parking\Parking Environment> tensorboard --logdir results
```

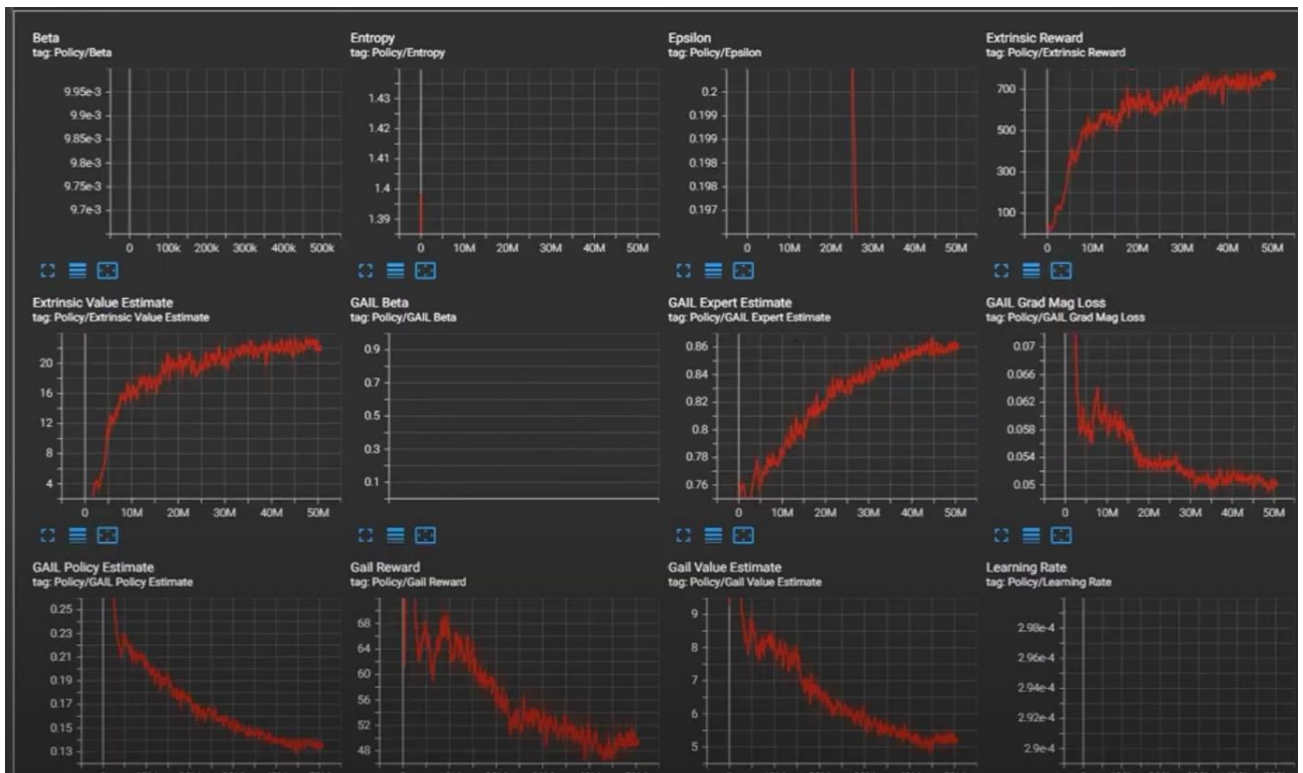
```
D:\miniconda\envs\mlagents2\lib\site-packages\torch\__init__.py:747: UserWarning: torch.set_default_tensor_type() is deprecated as of PyTorch 2.1, please use torch.set_default_dtype() and torch.set_default_device() as alternatives. (Triggered internally at C:\cb\pytorch_100000000000\work\torch\csrc\tensor\python_tensor.cpp:433.)
  _C._set_default_tensor_type(t)
```



```
Version information:
ml-agents: 0.30.0,
ml-agents-envs: 0.30.0,
Communicator API: 1.5.0,
PyTorch: 2.3.0
```

```
Windows PowerShell x Windows PowerShell x + v
[INFO] CarBehaviour. Step: 47500000. Time Elapsed: 50555.567 s. Mean Reward: 493.799. Std of Reward: 513.486. Training.
[INFO] CarBehaviour. Step: 47600000. Time Elapsed: 50658.284 s. Mean Reward: 509.107. Std of Reward: 507.947. Training.
[INFO] CarBehaviour. Step: 47700000. Time Elapsed: 50770.513 s. Mean Reward: 514.435. Std of Reward: 509.211. Training.
[INFO] CarBehaviour. Step: 47800000. Time Elapsed: 50875.178 s. Mean Reward: 588.329. Std of Reward: 503.227. Training.
[INFO] CarBehaviour. Step: 47900000. Time Elapsed: 50980.874 s. Mean Reward: 613.966. Std of Reward: 478.894. Training.
[INFO] CarBehaviour. Step: 48000000. Time Elapsed: 51090.529 s. Mean Reward: 540.151. Std of Reward: 511.090. Training.
[INFO] Exported results\run_build\CarBehaviour\CarBehaviour-47999973.onnx
[INFO] CarBehaviour. Step: 48100000. Time Elapsed: 51195.429 s. Mean Reward: 542.926. Std of Reward: 517.293. Training.
[INFO] CarBehaviour. Step: 48200000. Time Elapsed: 51308.372 s. Mean Reward: 512.750. Std of Reward: 517.724. Training.
[INFO] CarBehaviour. Step: 48300000. Time Elapsed: 51416.391 s. Mean Reward: 585.853. Std of Reward: 495.182. Training.
[INFO] CarBehaviour. Step: 48400000. Time Elapsed: 51525.625 s. Mean Reward: 598.239. Std of Reward: 502.664. Training.
[INFO] CarBehaviour. Step: 48500000. Time Elapsed: 51628.938 s. Mean Reward: 445.519. Std of Reward: 520.905. Training.
[INFO] CarBehaviour. Step: 48600000. Time Elapsed: 51736.054 s. Mean Reward: 551.923. Std of Reward: 519.174. Training.
[INFO] CarBehaviour. Step: 48700000. Time Elapsed: 51834.683 s. Mean Reward: 503.628. Std of Reward: 507.653. Training.
[INFO] CarBehaviour. Step: 48800000. Time Elapsed: 51936.954 s. Mean Reward: 629.846. Std of Reward: 505.641. Training.
[INFO] CarBehaviour. Step: 48900000. Time Elapsed: 52040.373 s. Mean Reward: 531.667. Std of Reward: 492.471. Training.
[INFO] CarBehaviour. Step: 49000000. Time Elapsed: 52139.111 s. Mean Reward: 476.155. Std of Reward: 510.980. Training.
[INFO] Exported results\run_build\CarBehaviour\CarBehaviour-48999981.onnx
[INFO] CarBehaviour. Step: 49100000. Time Elapsed: 52243.755 s. Mean Reward: 492.763. Std of Reward: 521.542. Training.
[INFO] CarBehaviour. Step: 49200000. Time Elapsed: 52342.524 s. Mean Reward: 578.800. Std of Reward: 485.153. Training.
[INFO] CarBehaviour. Step: 49300000. Time Elapsed: 52439.870 s. Mean Reward: 534.973. Std of Reward: 485.097. Training.
[INFO] CarBehaviour. Step: 49400000. Time Elapsed: 52542.637 s. Mean Reward: 513.314. Std of Reward: 512.118. Training.
[INFO] CarBehaviour. Step: 49500000. Time Elapsed: 52640.417 s. Mean Reward: 518.269. Std of Reward: 495.821. Training.
[INFO] CarBehaviour. Step: 49600000. Time Elapsed: 52742.521 s. Mean Reward: 522.289. Std of Reward: 520.819. Training.
[INFO] CarBehaviour. Step: 49700000. Time Elapsed: 52851.440 s. Mean Reward: 475.575. Std of Reward: 511.109. Training.
[INFO] CarBehaviour. Step: 49800000. Time Elapsed: 52950.944 s. Mean Reward: 582.334. Std of Reward: 497.529. Training.
[INFO] CarBehaviour. Step: 49900000. Time Elapsed: 53056.731 s. Mean Reward: 480.138. Std of Reward: 503.426. Training.
[INFO] CarBehaviour. Step: 50000000. Time Elapsed: 53159.284 s. Mean Reward: 453.302. Std of Reward: 495.043. Training.
[INFO] Exported results\run_build\CarBehaviour\CarBehaviour-49999923.onnx
[INFO] Exported results\run_build\CarBehaviour\CarBehaviour-50003435.onnx
[INFO] Copied results\run_build\CarBehaviour\CarBehaviour-50003435.onnx to results\run_build\CarBehaviour.onnx.
(mlagents2) PS E:\2024-AI-Parking\Parking Environment>
```

10°C Cloudy 3:04 PM 5/24/2024



Шаг 6: Тестирование и доработка

1. Тестирование модели:

- После завершения обучения протестируйте модель, запустив симуляцию в Unity.
- Наблюдайте за поведением агента и оценивайте его эффективность.

2. Доработка модели:

- На основе результатов тестирования корректируйте параметры обучения, вознаграждения и компоненты агента для улучшения эффективности.

Шаг 7: Развертывание и публикация

1. Развертывание проекта:

- Упакуйте проект в отдельное приложение или поделитесь им на таких платформах, как Unity Asset Store или GitHub.

Демо: <https://youtu.be/n5qa4nFWkqE>

Рекомендации

- AI Learns to Park - Deep Reinforcement Learning
(https://www.youtube.com/watch?v=VMp6pq6_QjI)

- Multi-Agent Car Parking using Reinforcement Learning Demo (<https://www.youtube.com/watch?v=xMCpVxDpogA>)
- Parallel Parking using Reinforcement Learning - Unity3D & ML-Agents (<https://www.youtube.com/watch?v=QDVwSAgY6cA>)
- Multi-Agent Car Parking using Reinforcement Learning (<https://arxiv.org/pdf/2206.13338>)

Вывод

В процессе работы над проектом я успешно создал разнообразную и сложную среду моделирования, которая включает в себя зоны парковки, препятствия и автомобили, что позволило мне достоверно воссоздать пространство реальной парковки.

Я углубился во множество аспектов разработки автономных систем искусственного интеллекта. Я изучил принципы работы с ML-Agents Toolkit и его интеграцию с Unity, а также освоил методы создания и обучения агентов для выполнения конкретных задач, таких как автоматическая парковка.

Мой опыт включает в себя работу с различными типами данных, определение целей и наград, разработку и оптимизацию алгоритмов обучения с поддержкой глубокого обучения. Я также приобрел практические навыки в настройке параметров обучения и анализе результатов обучения для достижения оптимальной производительности системы.

Кроме того, я углубился в аспекты тестирования и оценки моделей, проводя эксперименты в различных сценариях и корректируя настройки для повышения эффективности и надежности моих решений. Этот проект стал для меня не только возможностью применить теоретические знания, но и научиться решать реальные проблемы в области искусственного интеллекта и автоматизации.