به نام خدا

پروژه پنجم (RL) - هوش مصنوعی **نیایش خانی ۹۹۵۲۱۲۳۵**

چالشی که در این پروژه با آن مواجه بودم نصب کردن dependency ها و لایبرری های لازم برای استفاده از gymnasium بود. با استفاده از لینک قرار گرفته در داکیومنت سوال و یوتیوب، این چالش ها را به سختی برطرف کردم.

توضیحات کد :

در کد ارائه شده، جنبه های یادگیری تقویتی (RL) ابتدا برای طراحی سیستم پاداش و نحوه تعامل محیط با عامل (ماشین در این مورد) قرار داده شده است. بخش‌های مربوط به RL کد شامل محاسبه پاداش، به‌روزرسانی‌های وضعیت و شرایط پایان است را با جزئیات مرور می کنیم:

۱. سیستم پاداش

سیستم پاداش در روش step و در کلاس FrictionDetector تعریف شده است که به این صورت است :

```
class FrictionDetector(contactListener):
    def __init__(self, env, lap_complete_percent):
        contactListener.__init__(self)
        self.env = env
        self.lap_complete_percent = lap_complete_percent
    def _contact(self, contact, begin):
        …
        if begin:
            obj.tiles.add(tile)
            if not tile.road_visited:
                tile.road_visited = True
                self.env.reward += 1000.0 / len(self.env.track)
                self.env.tile_visited_count += 1

                # Lap is considered completed if enough % of the track was
covered
                if (
                    tile.idx == 0
                    and self.env.tile_visited_count / len(self.env.track)
                    > self.lap_complete_percent
                ):
                    self.env.new_lap = True
        else:
            obj.tiles.remove(tile)
```

- پاداش مثبت: هر بار که خودرو از یک بخش جاده جدید بازدید می کند، پاداشی معادل ۱۰۰۰,۰ / len (self.env.track)) دریافت می کند.
- تکمیل دور: اگر خودرو درصد کافی از بخش های مسیر را (که با self.lap_complete_percent تعریف می‌شود) بازدید کند، یک دور جدید تکمیل شده در نظر گرفته می‌شود (self.env.new_lap = True).

همچنین در کلاس step داریم :

```
    def step(self, action: Union[np.ndarray, int]):
        …
        step_reward = 0
        terminated = False
        truncated = False
        if action is not None:  # First step without action, called from
reset()
            self.reward -= 0.1
            # We actually don't want to count fuel spent, we want car to
be faster.
            # self.reward -=  10 * self.car.fuel_spent / ENGINE_POWER
            self.car.fuel_spent = 0.0
            step_reward = self.reward - self.prev_reward
            self.prev_reward = self.reward
            if self.tile_visited_count == len(self.track) or self.new_lap:
                # Truncation due to finishing lap
                # This should not be treated as a failure
                # but like a timeout
                truncated = True
            x, y = self.car.hull.position
            if abs(x) > PLAYFIELD or abs(y) > PLAYFIELD:
                terminated = True
                step_reward = -100

        if self.render_mode == "human":
            self.render()
        return self.state, step_reward, terminated, truncated, {}
```

- پاداش منفی: برای هر فریم، یک پاداش منفی کوچک (۰,۱-) اعمال می شود تا عامل را تشویق کند تا مسیر را به سرعت به پایان برساند.
- پایان: اگر ماشین از زمین بازی خارج شود، یک پاداش منفی بزرگ (۱۰۰-) دریافت می کند و terminate می شود(terminated = True).

۲. State Update

وضعیت محیط در متد step به روز می شود:

```
def step(self, action: Union[np.ndarray, int]):
        …
        self.car.step(1.0 / FPS)
        self.world.Step(1.0 / FPS, 6 * 30, 2 * 30)
        self.t += 1.0 / FPS

        self.state = self._render("state_pixels")
      …
```

- موقعیت خودرو و سایر چیز هایی که داینامیک هستند با استفاده از self.car.step (1.0 / FPS) به روز می شوند.
- وضعیت با self._render ("state_pixels") به روز می شود.

۳. Termination

اپیزودی که در حال اجرا است، می تواند تحت دو شرط خاتمه یابد:

۱. مرز Playfield: اگر ماشین از زمین بازی خارج شود (اگر abs(x) > PLAYFIELD یا abs(y) > PLAYFIELD)، آن اپیزود با یک جریمه قابل توجه پایان می یابد.

۲. تکمیل دور: اگر ماشین یک دور را با بازدید از درصد کافی از بخش ها (self.tile_visited_count == len(self.track) یا self.new_lap کامل کند، موفقیت‌آمیز در نظر گرفته می‌شود.

٤. متد Reset :

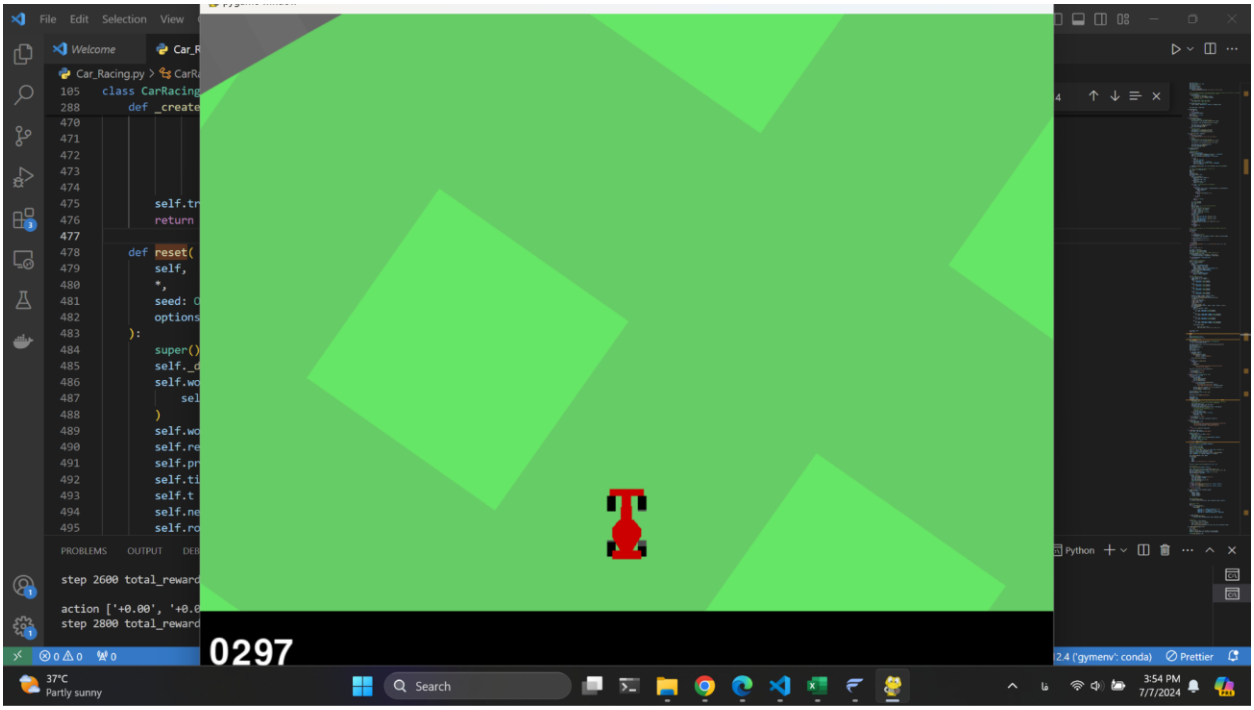
این متد محیط را برای یک اپیزود جدید مقداردهی اولیه می کند:

```
    def reset(
        self,
        *,
        seed: Optional[int] = None,
        options: Optional[dict] = None,
    ):
        …
        self.reward = 0.0
        self.prev_reward = 0.0
        self.tile_visited_count = 0
        self.t = 0.0
        self.new_lap = False
        …
```

```
        while True:
            success = self._create_track()
            if success:
                break
            if self.verbose:
                print(
                    "retry to generate track (normal if there are not
many"
                    "instances of this message)"
                )
        self.car = Car(self.world, *self.track[0][1:4])
            …
        return self.step(None)[0], {}
```

۱. ابتدا پاداش، تعداد بازدید کاشی و سایر متغیرهای لازم را دوباره برابر صفر قرار می دهد تا ریست شوند.
۲. در یک while true تلاش می کند تا یک create_track ایجاد کند تا زمانی که موفقیت آمیز باشد.
۳. قرارگیری خودرو: در آخر ماشین را در موقعیت شروع قرار می دهد.

خروجی کد :



همانطور که مشاهده می شود، تا زمانی که ماشین در مسیر جاده قرار دارد، امتیاز مثبت دریافت می کند و زمانی که از مسیر منحرف شود، امتیاز منفی دریافت می کند.