tanawin-st123975-game-lab05

March 10, 2024

[ ]:
```
import gym
from gym import spaces
import pygame
import numpy as np
import math
import random
```

[ ]:
```
WIDTH, HEIGHT = 800, 600
FPS = 60
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
```

[ ]:
```
class CarRacingEnv(gym.Env):
    def __init__(self):
        super(CarRacingEnv, self).__init__()
        self.action_space = spaces.Discrete(5)  # 5 actions: 0: accelerate, 1:␣
↪decelerate, 2: turn left, 3: turn right, 4: do nothing
        self.observation_space = spaces.Box(low=0, high=255, shape=(HEIGHT,␣
↪WIDTH, 3), dtype=np.uint8)
        self.viewer = None
        self.clock = pygame.time.Clock()
        self.car = None
        self.obstacles = []
        self.score = 0

    def step(self, action):
        if action == 0:
            self.car.accelerate()
        elif action == 1:
            self.car.decelerate()
        elif action == 2:
            self.car.turn_left()
        elif action == 3:
            self.car.turn_right()
        elif action == 4:
            pass  # Do nothing
        self.car.update()
        self._update_obstacles()
        observation = self._get_observation()

        # Check for collision with obstacles
        if self._check_collision():
            done = True
            reward = -100  # Negative reward for collision
        else:
            done = False
            reward = 0

            # Update score
            self.score += 1

        return observation, reward, done, {'score': self.score}

    def reset(self):
        pygame.init()
        self.viewer = pygame.display.set_mode((WIDTH, HEIGHT))
        pygame.display.set_caption("Car Racing")
        self.score = 0

        self.car = Car()
        self.obstacles = []
        self._generate_obstacle()  # Initial obstacle
        observation = self._get_observation()
        return observation

    def _get_observation(self):
        self.viewer.fill(WHITE)
        all_sprites = pygame.sprite.Group()
        all_sprites.add(self.car)
        all_sprites.add(self.obstacles)
        all_sprites.draw(self.viewer)
        pygame.display.flip()
        self.clock.tick(FPS)

        return pygame.surfarray.array3d(self.viewer)

    def render(self, mode='human'):
        pass

    def close(self):
        pygame.quit()
    def _check_collision(self):
        return pygame.sprite.spritecollide(self.car, self.obstacles, False)

    def _update_obstacles(self):
        for obstacle in self.obstacles:
            obstacle.update()
            if obstacle.rect.bottom > HEIGHT:
                self.obstacles.remove(obstacle)

        # Generate a new obstacle periodically
        if random.randint(1, 100) <= 5:  # Adjust the frequency of obstacle␣
↪generation
            self._generate_obstacle()

    def _generate_obstacle(self):
        new_obstacle = Obstacle()
        self.obstacles.append(new_obstacle)
```

[ ]:
```
class Car(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load("racecar.png").convert()  # Load the car␣
↪image
        self.image = pygame.transform.scale(self.image, (50, 70))  # Resize the␣
↪image as needed
        self.rect = self.image.get_rect()
        self.rect.center = (WIDTH // 2, HEIGHT // 2)
        self.speed = 0
        self.angle = 0

    def update(self):
        self.speed *= 0.95  # Add friction
        dx = self.speed * math.cos(self.angle)
        dy = self.speed * math.sin(self.angle)
        self.rect.x += dx
        self.rect.y -= dy

    def accelerate(self):
        self.speed += 0.5

    def decelerate(self):
        self.speed -= 0.5

    def turn_left(self):
        self.angle += 0.1

    def turn_right(self):
        self.angle -= 0.1
```

[ ]:
```
class Obstacle(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((30, 30))
        self.image.fill(RED)
        self.rect = self.image.get_rect()
        self.rect.x = random.randint(0, WIDTH - 30)
        self.rect.y = -30  # Start above the screen

    def update(self):
        self.rect.y += 5  # Adjust the speed of the obstacle
```

[ ]:
```
# Test the environment
env = CarRacingEnv()
observation = env.reset()
done = False
while not done:
    action = env.action_space.sample()  # Sample a random action
    observation, reward, done, info = env.step(action)
    score = info.get('score', 0)
    print(f"Score: {score}")

env.close()
```

```
libpng warning: iCCP: known incorrect sRGB profile
```

```
Score: 1
Score: 2
Score: 3
Score: 4
Score: 5
Score: 6
Score: 7
Score: 8
Score: 9
Score: 10
Score: 11
Score: 12
Score: 13
Score: 14
Score: 15
Score: 16
Score: 17
Score: 18
Score: 19
Score: 20
```

```
Score: 21
Score: 22
Score: 23
Score: 24
Score: 25
Score: 26
Score: 27
Score: 28
Score: 29
Score: 30
Score: 31
Score: 32
Score: 33
Score: 34
Score: 35
Score: 36
Score: 37
Score: 38
Score: 39
Score: 40
Score: 41
Score: 42
Score: 43
Score: 44
Score: 45
Score: 46
Score: 47
Score: 48
Score: 49
Score: 50
Score: 51
Score: 52
Score: 53
Score: 54
Score: 55
Score: 56
Score: 57
Score: 58
Score: 59
Score: 60
Score: 61
Score: 62
Score: 63
Score: 64
Score: 65
Score: 66
Score: 67
Score: 68
```

```
Score: 69
Score: 70
Score: 71
Score: 72
Score: 73
Score: 74
Score: 75
Score: 76
Score: 77
Score: 78
Score: 79
Score: 80
Score: 81
Score: 82
Score: 83
Score: 84
Score: 85
Score: 86
Score: 87
Score: 88
Score: 89
Score: 90
Score: 91
Score: 92
Score: 93
Score: 94
Score: 95
Score: 95
```

```
[ ]:
```