1. Objective

To implement a rule-based agent to navigate a simulated racecar around a track using sensor information (Lidar) and limited velocity information. The agent must make appropriate decisions to optimize forward movement and avoid going off the track.

2. Initial Setup

The starting code base includes:

An environment for Racecar

A cs1graphics-based visualizer

Various agent files: RandomAgent.py, RuleAgent.py

Run script: Run.py

Helper files: Track.py, Visualize.py, Geometry.py

Tracks can be initiated with:

python Run.py RuleAgent 1 -g 1000 -t 0.01

3. Problem Description

The agent must choose from 9 possible action combinations of motion and direction:

Directions: 'left', 'straight', 'right'

Motions: 'accelerate', 'coast', 'brake'

Sensor inputs:

obs['lidar']: 5 Lidar readings (left, left-front, front, right-front, right)

obs['velocity']: scalar float of current velocity

4. Original RuleAgent Design

Initially, the agent used basic position- and velocity-based heuristics to choose among three actions. This was successful but not reactive to the momentary configuration of the track as inferred from lidar feedback.

5. Modifications Made

```
import random

class Agent:

        def chooseAction(self, observations, possibleActions):

                return ('straight', 'coast')

class RuleAgent:

    def __init__(self):

        self.max_velocity = 0.8

    def chooseAction(self, obs):

        lidar = obs['lidar']

        velocity = obs['velocity']


        front = lidar[2]  # center

        left = lidar[0]   # far left

        right = lidar[4]  # far right


        if front > 0.8:

            direction = 'straight'

        elif left > right:

            direction = 'left'

        else:

            direction = 'right'


        if front < 0.3:
```

```
        speed = 'brake'
    elif velocity < self.max_velocity and front > 0.5:
        speed = 'accelerate'
    else:
        speed = 'coast'

    return (direction, speed)
```

6. Design Rationale

Obstacle Avoidance: If the front lidar sees an obstacle in front of it, the agent abruptly turns and brakes to circumvent it.

Path Finding: The agent prioritizes directions with greater lidar distance as safer paths.

Stability: In the absence of a preference, the car coasts straight, saving momentum without violent acceleration.

7. Testing & Validation

The updated agent was tested on track 1 using:

python Run.py RuleAgent 1 -g 1000 -t 0.01

Results:

- The car could make turns more effectively.
- Fewer collisions with track boundaries.
- Average speed increased slightly due to better path planning.

8. Conclusion

The minimal rule-based reasoning was significantly improved through the addition of lidar-based directional selection. The new logic is nearer to reactive behavior and provides a foundation for further improvements (e.g., memory, path prediction, or reinforcement learning).