Simulating Multiple Robots in Python

Overview

This simulation demonstrates how to control multiple robots in a 2D environment using Python and Pygame. Each robot performs a random walk and switches to line-following behavior upon detecting a black line. Collisions with a black obstacle are avoided by reversing direction.

Python Code

```
import pygame
import math
import random

# Initialize Pygame
pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Multiple Robots Simulation")
clock = pygame.time.Clock()

# Robot class
class Robot:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.angle = random.uniform(0, 2 * math.pi)
        self.color = color
        self.speed = 2
        self.angular_speed = random.uniform(-0.05, 0.05)
        self.mode = "random"  # or 'line_follow'

    def update(self):
        # Sensor point
        sensor_x = int(self.x + 10 * math.cos(self.angle))
        sensor_y = int(self.y + 10 * math.sin(self.angle))

        if 0 <= sensor_x < 800 and 0 <= sensor_y < 600:
            color = screen.get_at((sensor_x, sensor_y))
            if color[:3] == (0, 0, 0):  # black line
                self.mode = "line_follow"
            else:
                self.mode = "random"

        if self.mode == "line_follow":
            self.speed = 2
            self.angular_speed = 0
        else:
            # Random walk
            if random.random() < 0.05:
                self.angular_speed = random.uniform(-0.1, 0.1)
```

Simulating Multiple Robots in Python

```
            self.speed = 2

        self.angle += self.angular_speed
        self.x += self.speed * math.cos(self.angle)
        self.y += self.speed * math.sin(self.angle)

        # Check collision with obstacle
        robot_rect = pygame.Rect(self.x - 10, self.y - 10, 20, 20)
        if robot_rect.colliderect(obstacle):
            self.angle += math.pi  # reverse direction

    def draw(self):
        pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), 10)
        pygame.draw.line(
            screen, (255, 0, 0),
            (self.x, self.y),
            (self.x + 20 * math.cos(self.angle), self.y + 20 * math.sin(self.angle)), 2
        )

# Obstacle and Line
obstacle = pygame.Rect(300, 250, 100, 100)
robots = [
    Robot(100, 100, (0, 0, 255)),
    Robot(200, 150, (0, 255, 0)),
    Robot(400, 100, (255, 0, 0))
]

# Main loop
running = True
while running:
    screen.fill((255, 255, 255))

    # Events
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Draw environment
    pygame.draw.rect(screen, (0, 0, 0), obstacle)
    pygame.draw.line(screen, (0, 0, 0), (50, 500), (750, 500), 5)

    # Update and draw robots
    for robot in robots:
        robot.update()
        robot.draw()

    pygame.display.flip()
    clock.tick(30)

pygame.quit()
```