5-Practice

Theme: Simulation of Action Coordination

Ojective: study the concept of action coordination through simulation and to analyze the mechanisms by which multiple agents or objects act in a coordinated manner.

Theory:

Action coordination simulation is the process of modeling or simulating how multiple agents (or objects) combine their actions in a fair and coordinated manner to achieve a common goal. This process is widely used in solving complex problems, traffic systems, robotics, logistics, and other fields.

In Python, action coordination simulations can be designed in a simple and practical way for project-based learning.

Key Elements of a Motion Coordination Simulation

1. **Environment**
   The environment defines constraints and the state in which agents operate.
   For example, spaces in the environment may be occupied or free.
2. **Agents**
   Multiple agents work together to achieve a common goal.
   Each agent has a specific sequence of actions it can perform.
3. **Coordination Mechanism**
   Agents communicate with each other to coordinate their actions and achieve the shared objective.
   Coordination algorithms can use various methods, such as:

- *A Search:** Used to find the optimal path.
- **Contract Net Protocol:** Used for task allocation.
- **Distributed Constraint Optimization (DCOP):** Used to solve complex problems.

4. **Resources**
   Each agent has resources (e.g., time, energy, space).
   Efficient resource coordination is essential for successful task execution.

Simulation Process

1. **Path Planning by Agents**
   Each agent plans its own path using the `plan_path` method.This path is determined optimally to move the agent from its starting position to its goal.
2. **Coordination**
   If agents attempt to move to the same location, one agent waits.This approach helps prevent conflict situations.
3. **Updating Current State**
   In each cycle, agents update their current state and display it.The simulation continues until all agents reach their respective goals.

Practice:

Below is a simple example of a motion coordination simulation. In this example, two agents (Agent A and Agent B) coordinate with each other to choose the most optimal path.

```
import random

class Environment:
    def __init__(self):
        # Room status: 'free' or 'occupied'
        self.grid = {
            (0, 0): 'free', (0, 1): 'free', (0, 2):
'free',
            (1, 0): 'free', (1, 1): 'occupied', (1, 2):
'free',
            (2, 0): 'free', (2, 1): 'free', (2, 2): 'free'
        }
        self.agent_a_location = (0, 0)  # Agent A starting
location
        self.agent_b_location = (2, 2)  # Agent B starting
location

    def get_percept(self, location):
        """Returns the current state to the agent."""
        return self.grid.get(location, 'free')

    def move_agent(self, agent_name, direction):
        """Moves the agent in the given direction."""
        if agent_name == "Agent A":
            x, y = self.agent_a_location
```

```
        elif agent_name == "Agent B":
            x, y = self.agent_b_location
        else:
            return

        if direction == 'right' and y < 2:
            new_location = (x, y + 1)
        elif direction == 'left' and y > 0:
            new_location = (x, y - 1)
        elif direction == 'up' and x > 0:
            new_location = (x - 1, y)
        elif direction == 'down' and x < 2:
            new_location = (x + 1, y)
        else:
            new_location = (x, y)

        # Place the agent if the new location is free
        if self.grid[new_location] == 'free':
            if agent_name == "Agent A":
                self.agent_a_location = new_location
            elif agent_name == "Agent B":
                self.agent_b_location = new_location
            self.grid[new_location] = 'occupied'
            self.grid[(x, y)] = 'free'

    def check_goal(self):
        """Checks if both agents have reached their
goals."""
        return self.agent_a_location == (2, 2) and
self.agent_b_location == (0, 0)


class Agent:
    def __init__(self, name, environment, start_location,
goal_location):
        self.name = name
        self.environment = environment
        self.start_location = start_location
        self.goal_location = goal_location

    def plan_path(self):
```

```
        """Plans the path (simplified version)."""
        path = []
        current_x, current_y = self.start_location
        goal_x, goal_y = self.goal_location

        # Move along the X-axis
        while current_x != goal_x:
            if current_x < goal_x:
                path.append('down')
                current_x += 1
            elif current_x > goal_x:
                path.append('up')
                current_x -= 1

        # Move along the Y-axis
        while current_y != goal_y:
            if current_y < goal_y:
                path.append('right')
                current_y += 1
            elif current_y > goal_y:
                path.append('left')
                current_y -= 1

        return path


# Main program
if __name__ == "__main__":
    env = Environment()  # Create environment
    agent_a = Agent("Agent A", env, (0, 0), (2, 2))  #
Agent A
    agent_b = Agent("Agent B", env, (2, 2), (0, 0))  #
Agent B

    print("Simulation started:")
    step = 0

    while not env.check_goal():
        step += 1
        print(f"\n--- Step {step} ---")
```

```
        # Agents plan their paths
        agent_a_path = agent_a.plan_path()
        agent_b_path = agent_b.plan_path()

        # Determine moves for each agent
        if len(agent_a_path) > 0 and len(agent_b_path) >
0:
            agent_a_move = agent_a_path.pop(0)
            agent_b_move = agent_b_path.pop(0)

            # Coordination: if both agents move to the
same location, make one wait
            if (agent_a_move == 'down' and agent_b_move ==
'up') or \
                (agent_a_move == 'up' and agent_b_move ==
'down') or \
                (agent_a_move == 'right' and agent_b_move
== 'left') or \
                (agent_a_move == 'left' and agent_b_move ==
'right'):
                 print("Coordination: Agents are moving to
the same location. Agent B is waiting...")
                 env.move_agent("Agent A", agent_a_move)
            else:
                 env.move_agent("Agent A", agent_a_move)
                 env.move_agent("Agent B", agent_b_move)

        # Display current state
        print(f"Agent A location: {env.agent_a_location},
Agent B location: {env.agent_b_location}")
        print(f"Room status:\n{env.grid}")

    print("\nSimulation finished! All agents have reached
their goals.")
```

Result

```
Simulation started:
--- Step 1 ---
Agent A location: (0, 0), Agent B location: (2, 2)
```

```
Room status:
{(0, 0): 'free', (0, 1): 'free', (0, 2): 'free', (1, 0):
'free', (1, 1): 'occupied', (1, 2): 'free', (2, 0): 'free',
(2, 1): 'free', (2, 2): 'free'}

--- Step 2 ---
Coordination: Agents are moving to the same location. Agent B
is waiting...
Agent A location: (1, 0), Agent B location: (2, 2)
Room status:
{(0, 0): 'free', (0, 1): 'free', (0, 2): 'free', (1, 0):
'occupied', (1, 1): 'occupied', (1, 2): 'free', (2, 0):
'free', (2, 1): 'free', (2, 2): 'free'}

Simulation finished! All agents have reached their goals.
```