AI LAB | 8 AUGUST 2023

Name: Maloth Aditya

Roll No.: 120CS0124

Vacuum Cleaner

Simple Reflex:

Code:

```
import random

class SimpleVacuumEnvironment():
    def __init__(self):
        # Initialize rooms,room_status,agent_location
        # 1 means clean and 0 means dirty
        self.rooms = {'A': '0','B':'1'}
        self.rooms['A'] = random.randint('0','1')
        self.rooms['B'] = random.randint('0','1')
        self.vacuumLocation = random.choice('A','B')


    def is_dirty(self, room):
        # Returns if the room is dirty or not
        if self.rooms[room]=='1' :
            return False
        else:
            return True

    def clean(self, room):
        # Returns if the room is clean or not
        if self.rooms[room]=='1' :
            return True
        else :
            self.rooms[room]=1
            return False

    def move_agent(self, room):
        # Assigns the agent_location with room
        self.vacuumLocation = room

    def display(self):
        # Display the current state of the environment.
```

```
        print("Status of All Rooms:")
        for Room,Status in self.rooms.items() :
            print(f"{Room} : {Status}")
        print(f"Vacuum Location: {self.vacuumLocation}")
        print()


class SimpleReflexVacuumAgent(SimpleVacuumEnvironment ):

    def __init__(self, environment):
        self.environment = environment
        self.model = {}

    def perceive(self):
        # The agent perceives the current room it is in and whether the room is
dirty or not.
        currentRoom = self.environment.vacuumLocation
        dirtStatus = self.environment.is_dirty(currentRoom)
        if dirtStatus == 1 :
            print(f"{currentRoom} is Dirty")
        else :
            print(f"{currentRoom} is Clean")

    def decide_action(self,dirtStatus):
        #  Based on the dirt status, the agent decides whether to clean or move.
If the room is dirty, it chooses to clean; otherwise, it chooses to move.
        currentRoom = self.environment.vacuumLocation


    def act(self):
        # The agent performs the chosen action. If it decides to clean,
        # it cleans the room and updates the environment's status.
        # If it decides to move, it moves to the other room and updates its
location.
        currentRoom,dirtStatus = self.perceive()
        action = self.decide_action(dirtStatus)

        if action == '1' :
            self.environment.clean(currentRoom)
            print(f"Vaccum cleaned {currentRoom}")
        else :
            otherRoom = 'A' if currentRoom == 'B' else 'B'
            self.environment.move_agent(otherRoom)
            print(f"Vacuum moved from {currentRoom} to {otherRoom}")
```

```
# Example usage
env = SimpleVacuumEnvironment()
env.display()
agent = SimpleReflexVacuumAgent(env)
while True :
    agent.act()
    env.display()
```

Model Based:

Code:

```
import random

class SimpleVacuumEnvironment:
    def __init__(self):
        # Initialize rooms,room_status,agent_location
        self.rooms = {'A' : '0', 'B' : '0'}
        self.rooms['A'] = random.choice(['0','1'])
        self.rooms['B'] = random.choice(['0','1'])
        self.vacuumLocation = random.choice(['A','B'])

    def is_dirty(self, room):
        # Returns if the room is dirty or not
        return self.rooms[room] == '0'

    def clean(self, room):
        # Returns if the room is clean or not
        self.rooms[room] = '1'

    def move_agent(self, room):
        # Assigns the agent_location with room
        self.vacuumLocation = room

    def display(self):
        # Display the current state of the environment.
        print("Status of All Rooms:")
        for Room,Status in self.rooms.items() :
            print(f"{Room} : {Status}")
        print(f"Vacuum Location: {self.vacuumLocation}")
        print()
```

```
class ModelBasedReflexVacuumAgent:
    def __init__(self, environment):
        # Initialize environment and model
        self.environment = environment
        self.model = {}

    def perceive(self):
        # Perceive and return the current room and dirt status
        currentRoom = self.environment.vacuumLocation
        dirtStatus = self.environment.is_dirty(currentRoom)
        return currentRoom,dirtStatus

    def update_model(self, room):
        # The agent updates its model of the environment by marking
        # the current room as clean in its model.
        self.model[room] = '1'

    def decide_action(self, dirt_status):
        # the agent decides whether to clean or move based on its perception and
model.
        # If the room is dirty in the actual environment, it chooses to clean.
        # If the room is clean in the actual environment but marked as dirty in
the model,
        # it still chooses to clean based on the model. Otherwise, it chooses to
move.
        currentRoom = self.environment.vacuumLocation
        if dirt_status :
            return '1'
        elif self.model.get(currentRoom) == '0' :
            return '1'
        else :
            return '2'

    def act(self):
        # The agent performs the chosen action.
        # If it decides to clean, it cleans the room in both the actual
environment
        # and its model, and updates the model. If it decides to move, it moves
to
        # the other room and updates its location.
        currentRoom,dirtStatus = self.perceive()
        action = self.decide_action(dirtStatus)
```

```
        if action == '1' :
            self.environment.clean(currentRoom)
            self.update_model(currentRoom)
            print(f"Vacuum cleaned {currentRoom}")
        else :
            otherRoom = 'A' if currentRoom == 'B' else 'B'
            self.environment.move_agent(otherRoom)
            print(f"Vacuum moved from {currentRoom} to {otherRoom}")

# Example usage
env = SimpleVacuumEnvironment()
env.display()
agent = ModelBasedReflexVacuumAgent(env)
for _ in range(3):
    agent.act()
    env.display()
```

Goal Based:

Code:

```
import random

class SimpleVacuumEnvironment:
    def __init__(self):
        # Initialize rooms,room_status,agent_location
        self.rooms = {'A':'0','B':'0'}
        self.rooms['A'] = random.choice(['0','1'])
        self.rooms['B'] = random.choice(['0','1'])
        self.vacuumLocation = random.choice(['0','1'])

    def is_dirty(self, room):
        # Returns if the room is dirty or not
        return self.rooms[room] == '0'

    def clean(self, room):
        # Returns if the room is clean or not
        self.rooms[room] = '1'

    def move_agent(self, room):
        # Assigns the agent_location with room
        self.vacuumLocation = room
```

```
    def display(self):
        # Display the current state of the environment.
        print('Status of All Rooms:')
        for Room,Status in self.rooms.items() :
            print(f"{Room}: {Status}")
        print(f"Vacuum cleaner location: {self.vacuumLocation}")
        print()


class GoalBasedVacuumAgent:
    def __init__(self, environment):
        # Initialize environment and goals
        self.environment = environment
        self.goals = []

    def set_goal(self, goal):
        # Set goals with actions and priorities
        self.goals.append(goal)

    def prioritize_goals(self):
        # Sort goals based on priority criteria
        self.goals.sort(key=lambda x:x[1])

    def perceive(self):
        # Perceive and return the current room and dirt status.
        currentRoom = self.environment.vacuumLocation
        dirtStatus = self.environment.is_dirty(currentRoom)
        return currentRoom,dirtStatus

    def decide_action(self, dirt_status):
        # The agent decides on the action to take based on the highest
        # priority goal. If there are no goals, the agent will have no action.
        if self.goals :
            return self.goals[0][0]
        return None

    def act(self):
        # The agent performs the decided action. If the action is to clean,
        # it cleans the room. If the action is to move, it moves to the target
room.
        currentRoom,dirtStatus = self.perceive()
        action = self.decide_action(dirtStatus)

        if action == '1':
```

```
            self.environment.clean(currentRoom)
            print(f"Vacuum cleaned {currentRoom}")
        elif action =='2' :
            otherRoom = 'A' if currentRoom == 'B' else 'B'
            self.environment.move_agent(otherRoom)
            print(f"Vacuum moved from {currentRoom} to {otherRoom}")
        else :
            print('Vacuum has no action')


# Example usage
env = SimpleVacuumEnvironment()
agent = GoalBasedVacuumAgent(env)
agent.set_goal(('Clean', 1))  # Priority 1: Clean the room
agent.set_goal(('Move', 2))  # Priority 2: Move to room B
agent.prioritize_goals()
for _ in range(5):
    agent.act()
    env.display()
```

Utility Based:

Code:

```
import random

class SimpleVacuumEnvironment:
    def __init__(self):
        self.rooms = {'Room A': 'Clean', 'Room B': 'Dirty'}
        self.agent_location = random.choice(['Room A', 'Room B'])

    def is_dirty(self, room):
        return self.rooms[room] == 'Dirty'

    def clean(self, room):
        self.rooms[room] = 'Clean'

    def move_agent(self, room):
        self.agent_location = room

    def display(self):
        print("Room Status:")
        for room, status in self.rooms.items():
```

```
            print(f"{room}: {status}")
        print(f"Vacuum Cleaner Location: {self.agent_location}")
        print()

class UtilityBasedVacuumAgent:
    def __init__(self, environment):
        self.environment = environment
        self.utilities = {'Room A': 0, 'Room B': 0}

    def calculate_utilities(self):
        for room in self.utilities:
            if self.environment.is_dirty(room):
                self.utilities[room] = -1
            else:
                self.utilities[room] = 1

    def decide_action(self):
        current_room = self.environment.agent_location
        other_room = "Room A" if current_room == "Room B" else "Room B"
        if self.utilities[current_room] < self.utilities[other_room]:
            return "Clean"
        else:
            return "Move"

    def act(self):
        action = self.decide_action()

        if action == "Clean":
            current_room = self.environment.agent_location
            self.environment.clean(current_room)
            print(f"Agent cleaned {current_room}.")
        else:
            current_room = self.environment.agent_location
            other_room = "Room A" if current_room == "Room B" else "Room B"
            self.environment.move_agent(other_room)
            print(f"Agent moved from {current_room} to {other_room}.")

# Example usage
env = SimpleVacuumEnvironment()
env.display()
agent = UtilityBasedVacuumAgent(env)

for x in range(5):
    agent.calculate_utilities()
    print(f"Utility Value = {agent.utilities}")
```

```
    agent.act()
    env.display()
```