LAB ASSIGNMENT – Intelligent Systems Design

P Kartikeya Raj

2023001492

Title: Program to Control Vacuum Cleaner Movements in an Intelligent Agent Environment

Design and implement an intelligent vacuum-cleaner agent that can move inside a 2×2 or 3×3 grid environment and clean dirty locations based on percepts.

Your program must simulate the Intelligent System Design Process, including:

i. Problem Definition

ii. Environment Modeling

iii. Agent Design (Sensors + Actuators)

iv. Control Strategy

v. Simulation Output

i.Prob definition:

Objective is to make a vacuum cleaner agent that cleans dirt if dirt is there in the room else moves to other room, since it is a grid it has to follow snake pattern like for eg 2x2 grid, 1->2->4->3 then rooms are cleaned.

Each cell may be either:
Clean or Dirty
The agent must:

1. Perceive the environment using sensors.
2. Decide appropriate actions using a control strategy.
3. Act using actuators to move and clean.
4. Terminate when the entire environment becomes clean.

The final goal is to minimize the number of actions, such as movement and cleaning, making the agent rational and efficient.

ii. Environment Modeling

We model the environment as a discrete, deterministic, fully observable grid world.

A. Environment Representation
For 2×2 Grid
A | B

C | D
For 3×3 Grid
A | B | C

D | E | F

G | H | I
B. Attributes of Each Cell
Each cell of the grid has 2 possible states:

Dirty
Clean

The agent has a current location such as A, B, C.

C. Environment Type Based on AI Classification

Property	Explanation
Fully observable	Agent perceives dirt status + location
Static	Environment does not change unless agent acts
Discrete	Grid-based
Deterministic	Same action → same result
Single-agent	Only one vacuum cleaner

iii. Agent Design (Sensors + Actuators)
A. Sensors
The agent uses the following sensors:

1. Location Sensor
    - Detects the current cell (A, B, C…).
2. Dirt Sensor
    - Returns either:
        - Dirty
        - Clean

B. Actuators
The vacuum cleaner can perform these actions:

- Move Up
- Move Down
- Move Left

- Move Right
- Suck (Clean the location)
- NoOp (Do nothing → used when all cells are clean)

iv. Control Strategy
We use a simple reflex agent with a rule-based strategy:
Rule Set

1. IF current cell is dirty → THEN Suck
2. ELSE move to the next unvisited or dirty cell
3. IF all cells are clean → NoOp (stop)

Traversal Strategy
We use a systematic scanning pattern such as:

- Left → Right
- Top → Bottom

(Or serpentine pattern for efficiency.)

Percept (Sensor Input)	**Action (Actuator Output)**	**Description**
Current Location (e.g., A, B, C…)	Move Up	Move one cell upward in the grid
Current Location (e.g., A, B, C…)	Move Down	Move one cell downward in the grid
Current Location (e.g., A, B, C…)	Move Left	Move one cell to the left in the grid
Current Location (e.g., A, B, C…)	Move Right	Move one cell to the right in the grid
Dirt Status = Dirty	Suck	Clean the current cell
Dirt Status = Clean	NoOp	Do nothing (used when all cells are clean)

Simulation output

Code

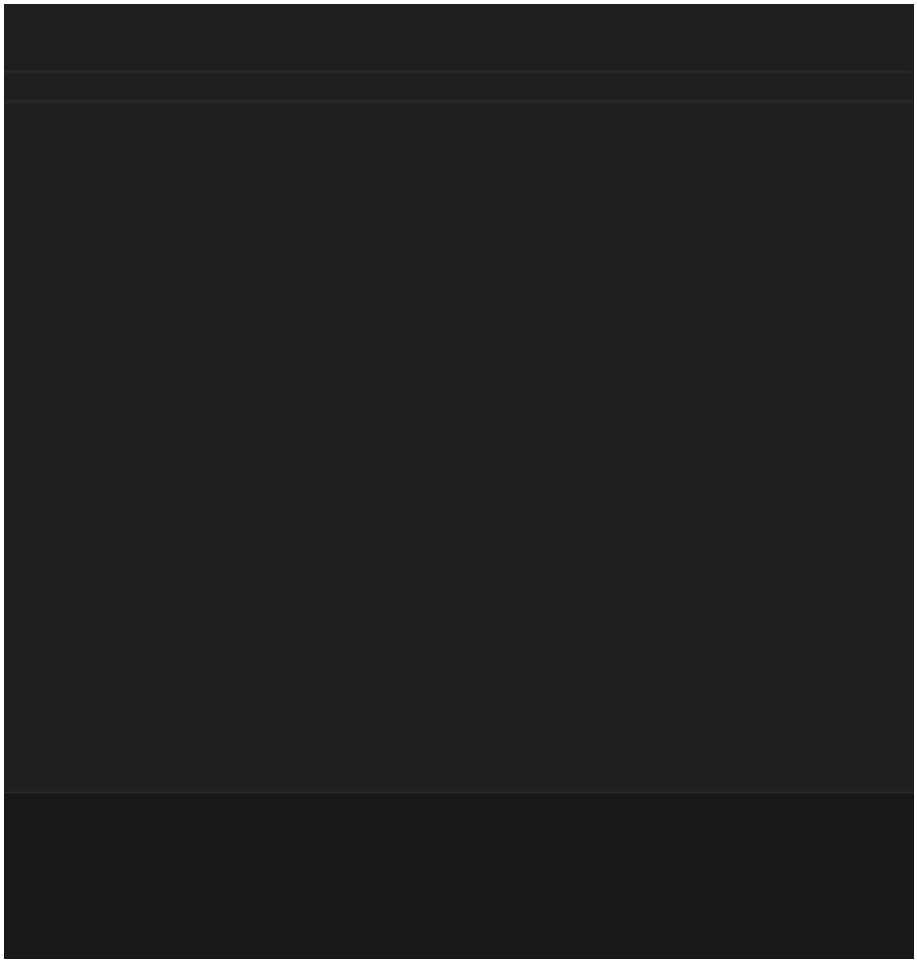
```
import pygame
import random
import time

GRID_SIZE = 3
CELL_SIZE = 150
WINDOW_SIZE = GRID_SIZE * CELL_SIZE

WHITE = (255, 255, 255)
```

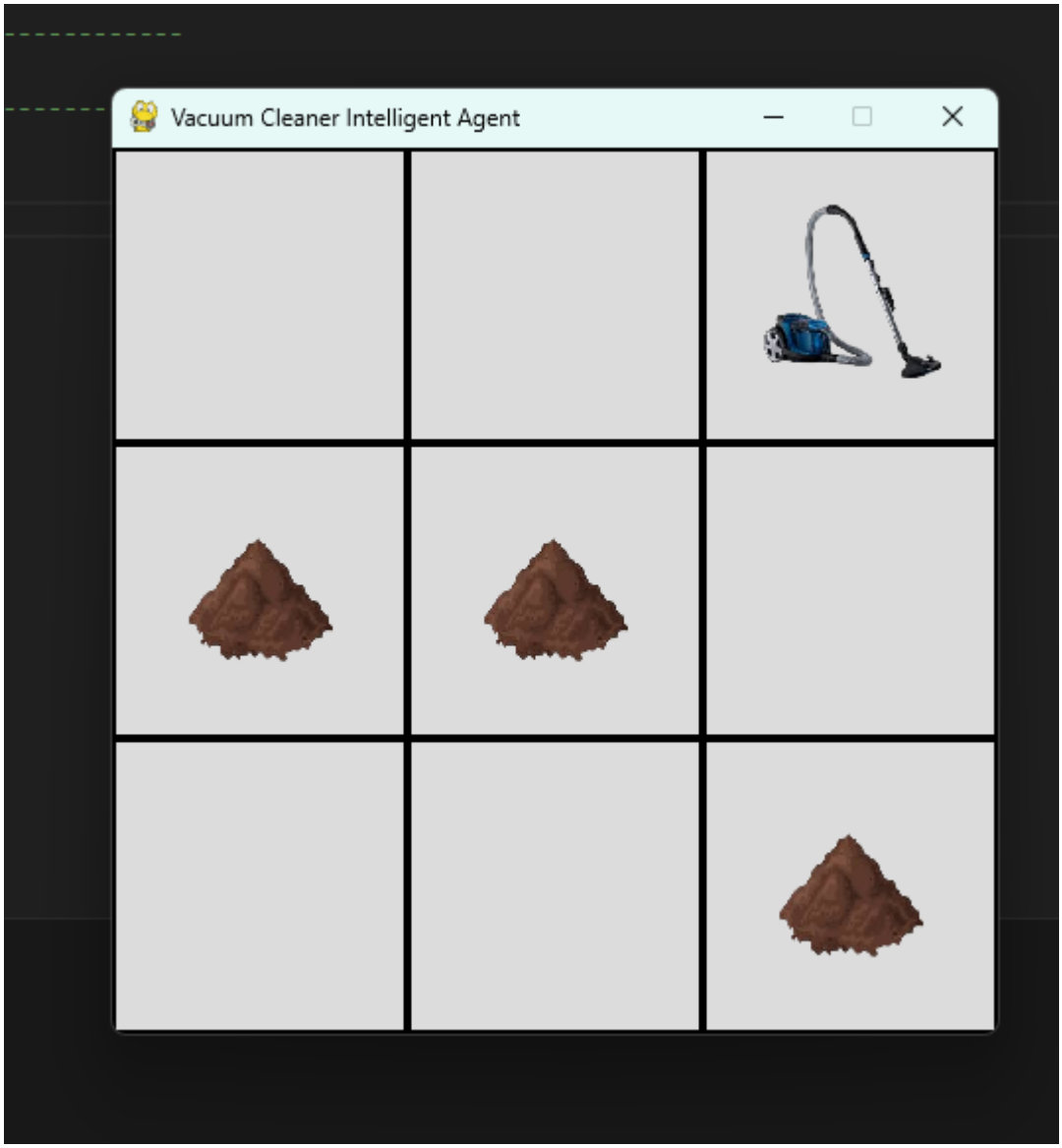
```
BLACK = (0, 0, 0)

pygame.init()
window = pygame.display.set_mode((WINDOW_SIZE, WINDOW_SIZE))
pygame.display.set_caption("Vacuum Cleaner Intelligent Agent")

font = pygame.font.SysFont(None, 32)

vacuum_img = pygame.image.load("vacuum.png")
vacuum_img = pygame.transform.scale(vacuum_img, (100, 100))

dirt_img = pygame.image.load("dirt.png")
dirt_img = pygame.transform.scale(dirt_img, (100, 100))

clean_color = (220, 220, 220)

class Environment:
    def __init__(self, size):
        self.size = size
        self.grid = [[random.choice([0, 1]) for _ in range(size)] for _ in
range(size)]

    def is_dirty(self, x, y):
        return self.grid[y][x] == 1

    def clean(self, x, y):
        self.grid[y][x] = 0

class VacuumAgent:
    def __init__(self, env: Environment):
        self.env = env
        self.x = 0
        self.y = 0
        self.path = self.generate_path()
        self.path_index = 0

    def generate_path(self):
        path = []
        for row in range(self.env.size):
            if row % 2 == 0:
                for col in range(self.env.size):
                    path.append((col, row))
            else:
                for col in reversed(range(self.env.size)):
                    path.append((col, row))
        return path

    def sense_dirty(self):
        return self.env.is_dirty(self.x, self.y)
```

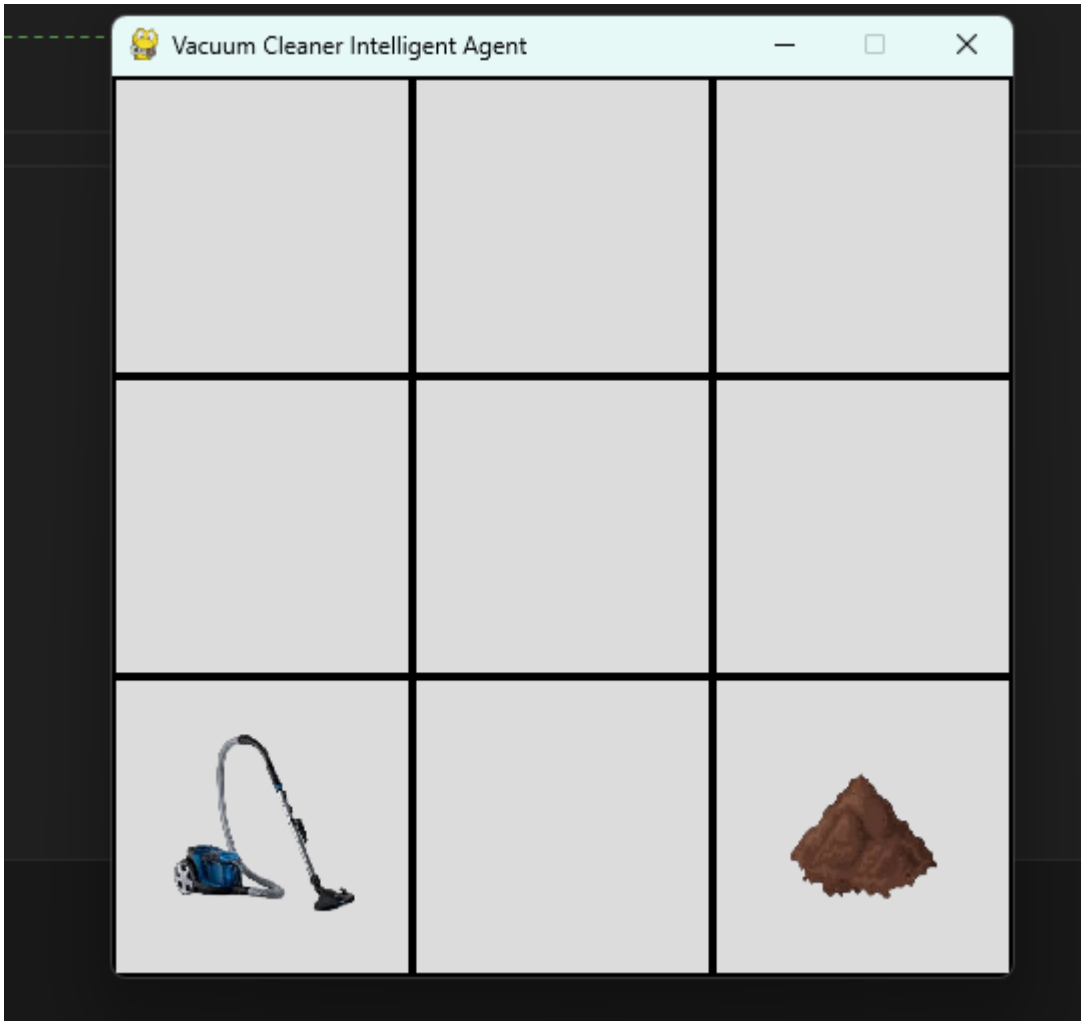
```
    def act(self):
        if self.sense_dirty():
            print(f"Cleaning cell {self.x},{self.y}")
            self.env.clean(self.x, self.y)
            return
        if self.path_index < len(self.path) - 1:
            self.path_index += 1
            self.x, self.y = self.path[self.path_index]
            print(f"Moving to {self.x},{self.y}")

def draw(environment, agent):
    window.fill(WHITE)
    for y in range(environment.size):
        for x in range(environment.size):
            cell_x = x * CELL_SIZE
            cell_y = y * CELL_SIZE
            pygame.draw.rect(window, clean_color, (cell_x, cell_y,
CELL_SIZE, CELL_SIZE))
            pygame.draw.rect(window, BLACK, (cell_x, cell_y, CELL_SIZE,
CELL_SIZE), 2)
            if environment.is_dirty(x, y):
                window.blit(dirt_img, (cell_x + 25, cell_y + 25))
    agent_x = agent.x * CELL_SIZE + 25
    agent_y = agent.y * CELL_SIZE + 25
    window.blit(vacuum_img, (agent_x, agent_y))
    pygame.display.update()

def run_simulation():
    environment = Environment(GRID_SIZE)
    agent = VacuumAgent(environment)
    running = True
    clock = pygame.time.Clock()
    while running:
        clock.tick(1)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
        agent.act()
        draw(environment, agent)
        if all(environment.grid[r][c] == 0 for r in range(GRID_SIZE) for c
in range(GRID_SIZE)):
            print("All cells cleaned! Simulation complete.")
            time.sleep(2)
            running = False
    pygame.quit()

if __name__ == "__main__":
    run_simulation()
```

Output

```
python vacuum-ver2.py

pygame 2.6.1 (SDL 2.28.4, Python 3.13.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
Moving to 1,0
Moving to 2,0
Cleaning cell 2,0
Moving to 2,1
Moving to 1,1
Cleaning cell 1,1
Moving to 0,1
Cleaning cell 0,1
Moving to 0,2
Moving to 1,2
Moving to 2,2
Cleaning cell 2,2
All cells cleaned! Simulation complete.
```




Vacuum Cleaner Intelligent Agent



vacuum.mp4