**AI OEL 4th Semester**

**Submitted BY:**

**Syed Ali Tariq (23-CS-072)**

**Section:**

**“C”**

**Maze Solver Agent – Explanation Document**

**1. Agent Type**

**Model-Based Agent**

* The agent uses a complete internal representation of the environment in a 10×10 grid (grid[][]), distinguishing between wall and free cells.
* Before taking any step, the agent applies the **A\*** search algorithm to compute a full path from the start point to the goal.
* It navigates the environment based on the pre-computed plan.

**2. Decision Logic**

**Heuristic Used:**

* **Manhattan Distance:**

h(n)=∣x1−x2∣+∣y1−y2∣h(n) = |x₁−x₂| + |y₁−y₂|h(n)=∣x1​−x2​∣+∣y1​−y2​∣

**A\* Algorithm Details:**

1. **Open Set:**
   * A priority queue (min-heap) ordered by total estimated cost f = g + h.
2. **Cost Definitions:**
   * g: Cost from the start node to the current node.
   * h: Estimated cost from the current node to the goal using the Manhattan heuristic.
3. **Path Reconstruction:**
   * When the goal node is removed from the open set, the path is reconstructed by backtracking through parent nodes.

**Agent Execution:**

* After computing the path, the agent moves step-by-step through each grid cell in the computed path.
* Each step is logged to the console for traceability.

**3. Game Rules and Objective**

* **Environment:** 10×10 grid with predefined wall cells (black).
* **Start Point:** (0, 0)
* **Goal Point:** (9, 9)
* **Victory Condition:** Agent successfully reaches the goal cell without colliding with any wall.

**Rendering Scheme:**

| **Color** | **Meaning** |
| --- | --- |
| White | Free Cell |
| Black | Wall Cell |
| Green | Computed Path |
| Red | Goal Cell |
| Blue Circle | Agent Position |

**4. Visual Presentation (Screenshots Required)**

Include the following screenshots in your document:

1. **Initial Grid:**
   * Agent at (0, 0), no path shown.
2. **Path Overlay:**
   * After A\* completes, show the green path overlay.
3. **Mid-Run Snapshot:**
   * Agent halfway along the green path.
4. **Final Snapshot:**
   * Agent arrives at the red goal cell (9, 9).

**5. Logging (Console Output)**

Example terminal logs during execution:

vbnet

CopyEdit

Computed Path: [(0, 0), (1, 0), (2, 0), ..., (9, 9)]

Step 0: Agent moves to (0, 0)

Step 1: Agent moves to (1, 0)

...

Step 18: Agent moves to (9, 9)

Agent reached the goal.

This output clearly shows:

* Path planning via A\*.
* Step-by-step movement decisions.

**6. Running & Capturing Screenshots**

**Requirements:**

* Python installed
* Pygame library installed:

nginx

CopyEdit

pip install pygame

**Run the Code:**

nginx

CopyEdit

python maze\_solver.py

**Capture Screenshots:**

You can use either:

* OS-level screenshot tools
* pygame.image.save(...) to programmatically save snapshots

Recommended screenshots to capture:

1. Initial grid
2. Grid with A\* path overlay
3. Mid-run snapshot
4. Final frame with agent on goal cell

**✅ Checklist: Submission Requirements Met**

* Model-Based Agent (internal map + path planning)
* Dynamic Agent Movement (animated using Pygame)
* A\* Search Implementation
* Visual Output (colored grid)
* Logging Output in Console