Lab 1: Intelligent Agent Simulation

Reflex and Goal-Based Agents in a Deterministic Grid World

Class Regulations

1. Individual work is mandatory.
2. Collaboration is encouraged, but copying code from another student is not allowed. The plagiarism assignments will not be scored.
3. One week to conduct each lab session. Late submission will not be accepted.
4. Students are allowed to request clarification within three days of grading, but must provide specific reasons for the re-evaluation request.

This lab session is to:

- Implement of intelligent agents within a simplified Pac-Man environment.
- Explore the behavioral differences between **reflex agents** and **goal-based agents**.

In a classical agent-environment framework, an **agent** perceives its environment through sensors and acts upon it via actuators.

This lab contrasts two agent architectures:

Agent Type	Perceptual Model	Decision Basis	Example Behavior
Simple Reflex Agent	Only the current percept	Rule-based (if–then)	Eats nearby food
Goal-Based Agent	Current percept + goal	Moves towards selected goal	Pursues nearest food (greedy)

Project Structure Overview:

File / Folder	What's Inside	Main Function
`config.py`	Game settings (speed, colors, rewards)	Stores all adjustable parameters for the game.
`main.py`	Main program	Starts the game, loads layout, and chooses agent type.
`engine.py`	`GridWorld` environment	Handles movement, scoring, and game rules.
`ui.py`	Game interface (Tkinter)	Draws the grid, Pac-Man, food, and updates score display.
`layout.py`	Layout reader	Loads `.lay` map files and finds Pac-Man's start position.
`utils.py`	Helper functions	Extra tools for counting food or checking map boundaries.
`agents/`	Folder of agents	Contains all agent scripts (Reflex, Goal-based).
`agents/reflex_agent.py`	`ReflexAgent` class	Moves based on nearby food.
`agents/goal_greedy_agent.py`	`GoalGreedyAgent` class	Chooses and moves toward the nearest food.
`layouts/`	Game maps	Stores maze layouts like `smallClassic` and `mediumClassic`.

Assignment 1 (20 points)

Experience the environment manually to understand movement, scoring, and termination.

To-do:

1) Run human mode:

```
python main.py --agent human --layout layouts/smallClassic.lay
```

Use the **arrow keys** to move Pac-Man. You can pause/resume with **Space**, step with **N**, and **R** to reset.

2) Observe the following rules:

- Food `.` : +10 points.
- Power pellet `o` : +50 points and **power mode** for 15 steps.
- Ghost `G` collision: **+200** if in power mode (ghost removed), **−500** otherwise.
- Every step: −1 point (step penalty).
- Bumping into a wall `%` : you remain in place (optional bump cost = 0 by default).

3) Answer these questions in your report:

- (a) What triggers power mode, and when does it end?
- (b) How does total score evolve over time during exploration?
- (c) What ends an episode (all pellets vs. step limit)?
- (d) One limitation you experienced with manual control.

Assignment 2 (30 points)

Explain how environment rules are encoded in the source code.

To-do:

1) Open **`engine.py`** and locate where the environment handles:

- Eating food `.` and power pellet `o`
- Entering and leaving **power mode**
- Colliding with **ghosts** `G`
- Updating `score`, `steps`, and `eaten`

2) Open **`ui.py`** and confirm:

- How walls `%`, empty spaces, food `.`, power `o`, ghosts `G`, and Pac-Man are drawn
- Why Pac-Man changes color in power mode
- What the information panel reports each frame (agent, mode, score, steps, food left, power state)

3) Write a short explanation:

- How the environment determines **termination**
- Why the world is designed to be **deterministic**
- What `env.neighbors(r,c)` returns and how it prevents walking through walls

Assignment 3 (50 points)

Compare simple reflex agent and goal-based agent through experiments.

Step 1: Execute both agents

1) Run each configuration separately:

```
python main.py --agent reflex --layout layouts/smallClassic.lay
python main.py --agent goal   --layout layouts/smallClassic.lay
```

2) Then repeat on a more complex map:

```
python main.py --agent reflex --layout layouts/mediumClassic.lay
python main.py --agent goal   --layout layouts/mediumClassic.lay
```

Step 2: Collect data (15 points)

1. For each layout and each agent type, run five independent simulations.
2. Record at the end of each run: total score, number of steps, food remaining, and ghosts eaten.

Example Table:

Layout	Agent Type	Average Score	Average Steps Taken	Total Food Remaining	Total Ghosts Eaten
smallClassic	Reflex	87	381	0	0
smallClassic	Goal-Based	94	366	0	0
mediumClassic	Reflex	52	400	4	0
mediumClassic	Goal-Based	77	395	0	0

Step 3: Observations (20 points)

While the simulation runs, note:

- Does the **Reflex Agent** loop or get stuck?
- Does the **Goal-Based Agent** reduce distance to food more consistently?
- Behavior near ghosts/power pellets?
- Which finishes faster or leaves less food?

Step 4: Discussion (15 points)

- How the **Reflex Agent** makes decision when it fails to reach food?
- How the **Goal-Based Agent** differs conceptually (goal selection + greedy step).
- Compare efficiency (score/steps) and robustness (ability to finish).

Submission Requirements

Each student must submit a report (2–3 pages, PDF format) containing:

- Clear and detailed answers for each assignment.
- Screenshots of results for each experiment.