CSC14003 – Artificial Intelligence

PROJECT 02: LOGICAL AGENT

Group :

- 22127014 - Nguyễn Kim Anh
- 22127092 - Lê Bảo Giang
- 22127433 - Nguyễn Ngọc Anh Tú
- 22127460 - Quách Trần Quán Vinh

I. Assignment Plan

- Setup the environment and download the necessary library
- Implement the `Program` class for map handling and pecept updates.
  - Map reading and initalization
  - Percept updates
  - Visualization: `draw_grid` to render the grid
- Implement the `Agent` class for agent behavior and knowledge base updates.
  - Agent movement and actions: `move_forward`, `turn_left`, `turn_right`, `turn_around`
  - Percept handling: `perceive_current_cell`, `infer_surroundings` in `Agent`
  - Knowledge base updates and PL-resolution: `update_KB`, `PL_resolve`, `PL_resolution`
  - Decision making: `make_safe_move`, `explore`, `find_to_start`
- Testing and debugging
- Documentation:
  - Write the report
  - Write the code documentation: comments and docstrings

II. Project information

1. Environment setup

- Python
- Pygame: This library is used for the graphical interface. Install it using pip: `pip install pygame`
- SymPy: This library is used for symbolic mathematics. Install it using pip: `pip install sympy`

2. Running the program

- Execute the main file `main.py` to run the project.

III. Self evaluation

Task	Evaluation	Completion
Finish problem successfully	Finished but there were some difficulties. Mostly from the unclear or unexplained or non existence requirements and cases.	100%
Graphical demonstration of each step of the running process	Finish it quickly after we had finished coding the path finding process.	100%
Generate at 5 maps with difference structures: position and number of Pit, Gold and Wumpus	Finish it quickly after we had finished coding the path finding process.	100%
Report your algorithm, experiment with some reflection or comments	It is done quite quickly as we had already coded all of the requirements.	100%

IV. Basic Implement

Class `agent`

Attribute

Attribute	Description
KB	A list of clauses that the agent knows
start	constant = (1, 1)
pos	constant = (1, 1)
program	Current program
grid_size	Current program size
list tracked_map	A map which has same size with program size for tracking cell, initialized with 0
facing	The current direction, nititialized the first facing is NORTH
set visited	set of visited cell
int point	Score of agent (inititalize 0 at beginning)
list tracked_path	Track the path taken by agent
set unknown_cell	Coordinates of all cells in the grid that the agent has not yet visited or perceived
set safe	Coordinates of cells that the agent has determined to be safe based on its knowledge base (KB) and percepts
set not_unsafe	Coordinates of cells that the agent has determined not necessarily safe, but they are not confirmed to be dangerous either

Method

`perceive_current_cell(self)`

- Input parameter: None
- Output parameter: The percepts of the current cell
- Description: Returns the percepts of the current cell the agent is in

`neighbor_cells(self, x, y)`

- Input parameter:
  - `x` : The x-coordinate of the current cell
  - `y` : The y-coordinate of the current cell
- Output parameter: A list of tuples representing the coordinates of the neighboring cells
- Description: Returns the coordinates of the neighboring cells of the current cell

`infer_surroundings(self, element)`

- Input parameter: The element to infer
- Output parameter: A list of symbols representing the inferred surroundings
- Description: Infers the surroundings of the current cell for the given element

`infer_surroundings(self, element)`

- Input parameter: The element to infer
- Output parameter: A list of symbols representing the inferred surroundings
- Description: Infers the surroundings of the current cell for the given element

`update_KB(self)`

- Input parameter: None
- Output parameter: None
- Description: Updates the knowledge base of the agent based on the current percepts

`turn_left(self, current_direction, action)`

- Input parameter:
  - `current_direction` : The current direction the agent is facing
  - `action` : A boolean indicating whether to perform the action

Output parameter: The new direction after turning left Description: Turns the agent to the left and returns the new direction

`turn_right(self, current_direction, action)`

- Input parameter:
  - `current_direction` : The current direction the agent is facing
  - `action` : A boolean indicating whether to perform the action
- Output parameter: The new direction after turning right
- Description: Turns the agent to the right and returns the new direction

`turn_around(self)`

- Input parameter: None
- Output parameter: The cost of turning around
- Description: Turns the agent around and returns the cost of the action

`move_forward(self)`

- Input parameter: None
- Output parameter: The cost of moving forward
- Description: Moves the agent forward and returns the cost of the action

`opposite_direction(self, direction)`

- Input parameter:
  - `direction`: The current direction the agent is facing
- Output parameter: The opposite direction
- Description: Returns the opposite direction of the given direction

`make_safe_move(self, node)`

- Input parameter: The node to move
- Output parameter: The new node after making a safe move if it is posible
- Description: Makes a safe move from the current node and returns the new node

`align_direction(self, current_direction, desired_direction)`

- Input parameter:
  - `current_direction`: The current direction the agent is facing
  - `desired_direction`: The desired direction to align to
- Output parameter: The new direction after aligning
- Description: Aligns the agent's direction to the desired direction and returns the new direction

`align_direction_cost(self, current_direction, desired_direction)`

- Input parameter:
  - `current_direction`: The current direction the agent is facing
  - `desired_direction`: The desired direction to align to
- Output parameter: The cost of aligning the direction
- Description: Calculates and returns the cost of aligning the agent's direction to the desired direction

`explore(self)`

- Input parameter: None
- Output parameter: The node where the gold is found, or None if exploration ends without finding gold
- Description: Handles the exploration of the grid, updating the agent's knowledge base, and making safe moves

`shoot(self)`

- Input parameter: None
- Output parameter: None
- Description: Handles the action of shooting an arrow in the current direction

`expand(self, node, goal)`

- Input parameter:
  - `node`: The current node
  - `goal`: The goal node
- Output parameter: A list of nodes representing the expanded nodes
- Description: Generates possible moves from a given node, considering the grid boundaries and heuristic cost

`find_path_to_start(self)`

- Input parameter: None
- Output parameter: None
- Description: Finds a path back to the starting position using a priority queue and heuristic-based expansion

`PL_resolve(self, literal, Ci, Cj)`

- Input parameter: The literal to resolve, and the two clauses to resolve
- Output parameter: The resolvent of the two clauses
- Description: Resolves the two clauses and returns the resolvent

`PL_resolution(self, query)`

- Input parameter: The query to resolve
- Output parameter: True if the query is entailed by the knowledge base, False otherwise

- Description: Uses the PL resolution algorithm to determine if the query is entailed by the knowledge base

`die(self)`

- Input parameter: None
- Output parameter: None
- Description: Handles the agent's death and exit the program

Class `Program`

Attribute

Attribute	Description
list map	Read from input file
int size	Size of map (read from input file)
cell_size	const cell_size = 75
int width, height	size of cell

Method

`set_screen_size(self)`

- Input parameter: None
- Output parameter: None
- Description: Set screen size

`load_map(self, input_file)`

- Input parameter: `input_file` (input file)
- Output parameter: None
- Description: Load map from input file and update percepts

`read_map(self, input_file)`

- Input parameter: `input_file` (input file)
- Output parameter: `grid` and `size` of map
- Description: Read map from input file

`update_percepts(self)`

- Input paramenter: None
- Output parameter: None
- Description: Handle cell with multiple percepts by checking specific combinations in a single cell

`mark_visited(self, pos)`

- Input parameter: `pos` (location where agent has visited)
- Output parameter: None
- Description: Mark the position on map at cell that agent has visited by `self.map[self.size - x][y - 1] += '.V'`

`add_percept(self, x, y, percept)`

- Input parameter: `x`, `y`, `percept`
- Output parameter: None
- Description: Add percept to cell at position (x, y) and the adjacent cells by append to cell `'.' + percept`

`remove_gold(self, pos)`

- Input parameter: `pos` (location where agent has found gold)
- Output parameter: None
- Description: Removes the gold from a specified position on the map

`remove_element(self, pos, element)`

- Input parameter: `pos` (location), `element` (element to remove)
- Output parameter: None
- Description: Removes a specified element and its associated percept from the map

`draw_grid(self)`

- Input parameter: None

- Output parameter: None
- Description: Draw grid using `pygame` library

`run(self)`

- Input parameter: None
- Output parameter: None
- Description: Run the program using `pygame`

`get_cell_info(self, pos)`

- Input parameter: `pos` (location)
- Output parameter: return cell information (S, W, P, ...)
- Description: Get cell information `self.map[self.size - x][y-1]`

`move_agent(self, pos, direction, step)`

- Input parameter: `pos`, `direction`, `step` (step to move)
- Output parameter: None
- Description: Moves the agent to a new position and updates the display

`clear_agent(self.pos)`

- Input parameter: `pos` (location)
- Output parameter: None
- Description: Clears the agent's image from the specified position on the screen

`draw_agent(self, pos, direction)`

- Input parameter: pos (position), direction (direction)
- Output parameter: None
- Description: Draws the agent at the specified position and direction on the screen

`add_action(self, action)`

- Input parameter: action (action to add)
- Output parameter: None
- Description: Adds an action to the action log and updates the display

`draw_action_log(self)`

- Input parameter: None
- Output parameter: None
- Description: Draws the action log on the screen

`draw_action_log(self)`

- Input parameter: None
- Output parameter: None
- Description: Draws the action log on the screen

`handle_scroll(self, event)`

- Input parameter: event (scroll event)
- Output parameter: None
- Description: Handles scrolling in the action log

`update_status(self, health, point, healing_potions=0)`

- Input parameter: health (current health), point (current points), healing_potions (number of healing potions, default is 0)
- Output parameter: None
- Description: Updates the status display with the current health, points, and healing potions

`select_button(self, button)`

- Input parameter: button (button to select)
- Output parameter: None
- Description: Selects a map button and updates the display

`draw_buttons(self)`

- Input parameter: None
- Output parameter: None
- Description: Draws the map and control buttons on the screen.

`handle_button_click(self, event)`

- Input parameter: event (button click event)
- Output parameter: None
- Description: Handles button click events for map selection and control buttons

`move_agent_back(self)`

- Input parameter: None
- Output parameter: None
- Description: Moves the agent back one step

`move_agent_forward(self)`

- Input parameter: None
- Output parameter: None
- Description: Moves the agent forward one step

`show_percepts(self, pos)`

- Input parameter: pos (position)
- Output parameter: None
- Description: Displays the percepts for the specified position on the screen

`reset_map(self)`

- Input parameter: None
- Output parameter: None
- Description: Resets the map, agent position, and logs to their initial states

`print_map(self)`

- Input parameter: None
- Output parameter: None
- Description: Print map to console

`update_cellinfor(self, pos, infor)`

- Input parameter: pos (position), infor (information to update)
- Output parameter: None
- Description: Updates the information of the specified cell

`mark_cell_safe(self, pos)`

- Input parameter: pos (position)
- Output parameter: None
- Description: Marks the specified cell as safe

V. Knowledge base and PL - resolution

Knowledge base:

- At first, the KB (Knowledge base) will have:

$$B_{xy} \leftrightarrow (P_{x+1,y} \vee P_{x-1,y} \vee P_{x,y+1} \vee P_{x,y-1})$$

- With each cell travelled, KB will be updated accordingly
- The `update_KB` method updates the KB with new information based on the agent's percepts. For example, if the agent perceives a Breeze at cell (x, y), it adds the proposition $(B_{xy} \leftrightarrow (P_{x+1,y} \vee P_{x-1,y} \vee P_{x,y+1} \vee P_{x,y-1}))$ to the KB.
- Percepts: The agent perceives the current cell to determine if there are any hazards (Breeze, Stench, Whiff, Glow)
- Actions: The agent can move forward, turn left, turn right, turn around, or shoot an arrow
- Neighboring cell: The neighboring cells of a given cell (x, y) are (x+1, y), (x-1, y), (x, y+1), and (x, y-1), provided they are within the grid boundaries
- If there is a Breeze in the current cell, then at least one of the neighboring cells contains a Pit
- If there is no Breeze in the current cell (x, y), then none of the neighboring cells contain a Pit
- If there is a Stench in the current cell (x, y), then at least one of the neighboring cells contains a Wumpus
- If there is no Stench in the current cell (x, y), then none of the neighboring cells contain a Wumpus
- If there is a Whiff in the current cell (x, y), then at least one of the neighboring cells contains a Healing Potion

- If there is no Whiff in the current cell (x, y), then none of the neighboring cells contain a Healing Potion
- If there is a Glow in the current cell (x, y), then at least one of the neighboring cells contains a Poison Gas
- If there is no Glow in the current cell (x, y), then none of the neighboring cells contain a Poison Gas
- If there is a Wumpus or a Pit in the current cell (x, y), the agent dies
- Safe Cells:
  - The current cell (x, y) is safe if it does not contain a Wumpus or a Pit
- Health and status updates:
  - If there is Poison Gas in the current cell (x, y), the agent's health decreases by 25 points
  - The agent's status is updated based on the current health, points, and available health potions

PL - resolution

- The `PL_resolution` method is used to check if a query can be inferred from the KB: if $(A \vee B)$ and $(\neg A \vee C)$ are both true, then $(B \vee C)$ must also be true. The method iteratively applies the resolution rule to pairs of clauses until either a contradiction is found (indicating the query is true) or no new information can be derived (indicating the query is false)

Decision making

- `make_safe_move` in `Agent` class: determining the next safe move for the agent based on its current knowledge base (KB) and percepts:
  - If the agent's health (`hp`) is less than or equal to 50 and there are available healing potions (`available_hp`), the agent uses a potion to heal itself
  - If there are less than or equal to 3 healing potions available and the current cell contains a healing potion (.H_P.), the agent picks up the potion
  - Calculate the cost to align the agent's current direction to the desired direction using `align_direction_cost` then store the moves along with their alignment costs in `moves_with_costs`
  - Sort moves by alignment cost to minimize the number of turn required
  - For each move, check if the target cell is safe using `PL_resolution` and update the agent's points and return a new `Node` representing the move
- `explore` in `Agent` class: exploring the grid to find the gold while ensuring the agent's safety
  - Initialize the frontier list with the starting position of the agent.
  - Update the agent's position and direction based on the node's state and action update the agent's position and the status in UI grid. Then, add the current position to the visited set
  - If the current position is not the start, update the knowledge base (`update_KB`)
  - If the cell contains a poison gas (.P_G.), it is added to not_unsafe; otherwise, it is added to safe.
  - Update `unknown_cells` to mark the explored cell
  - Check for gold (.G.) and update the agent point
  - Call `make_safe_move` for safe move decision
  - For each unknown cell, if it is surrounded by unsafe cells, it is added to `not_unsafe` and removed from `unknown_cells`
- `find_path_to_start` in `Agent` class: finding a path back to the starting position
  - Using `PriorityQueue` and a `reached` dictionary to keep track of reached node
  - Call `expand` function to generate child nodes for given node. This function using heuristic to decide what posible move based on current location. For each candidate move, a new node is created with he updated state, parent, action, and cost

VI. Test cases and results

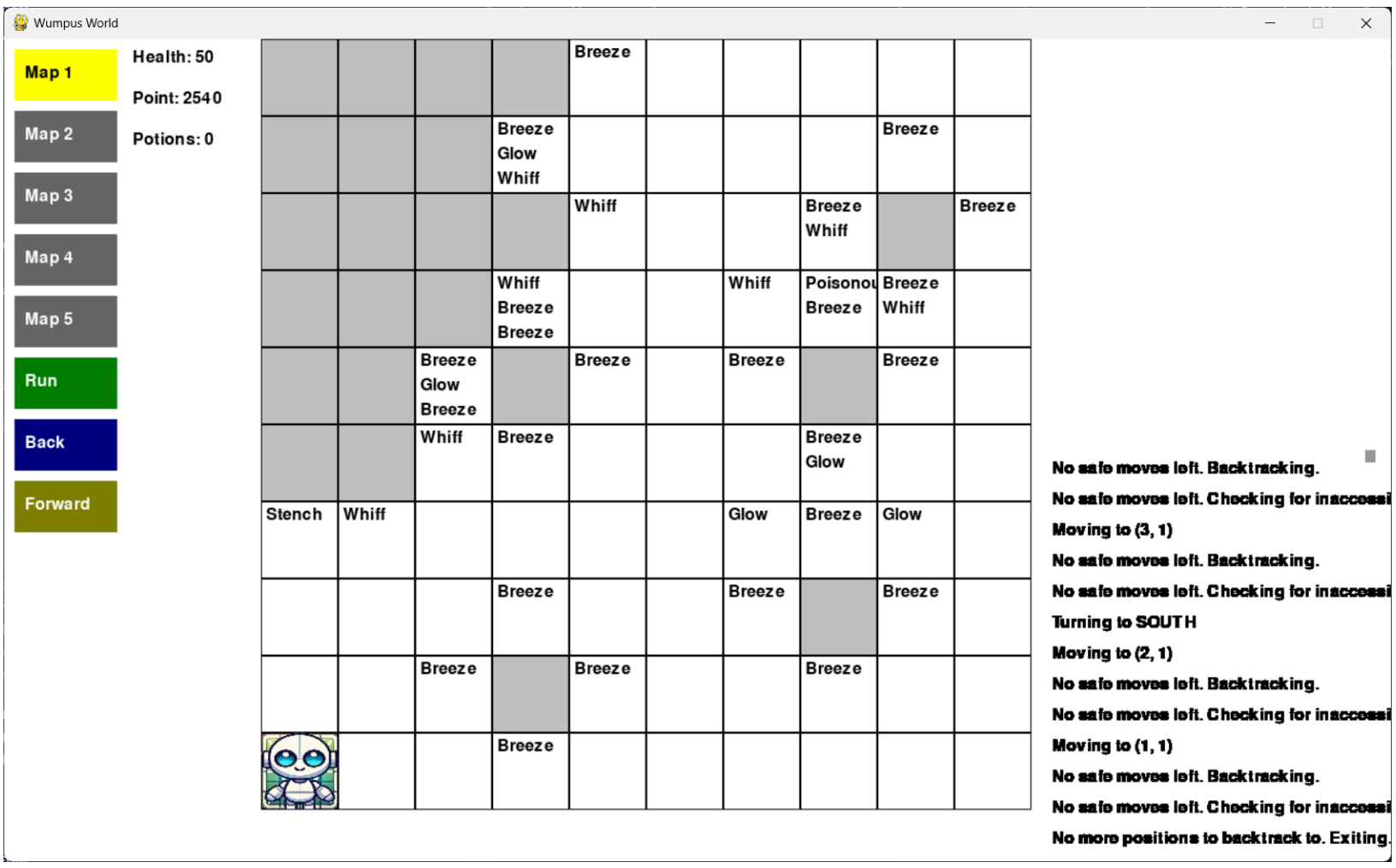
1. Test case 1: Grid 10x10

- File name: `map1.txt`
- Description:
  - 1 Wumpus
  - 1 Gold
  - 5 Pits
  - 3 Healing Potions
  - 3 Poisonous Gas

map1.txt

```
10
-.-.-.P.-.-.-.-.-
-.-.H_P.-.-.-.-.-.-
-.-.-.P_G.-.-.-.-.P.-
-.-.P.-.-.-.-.P_G.-.-
-.H_P.-.P.-.-.-.P.-
W.P_G.-.-.-.-.-.-.-
-.-.-.-.-.-.G.H_P.-.-
-.-.-.-.-.-.-.P.-.-
-.-.-.P.-.-.-.-.-
A.-.-.-.-.-.-.-.-
```

- Result:



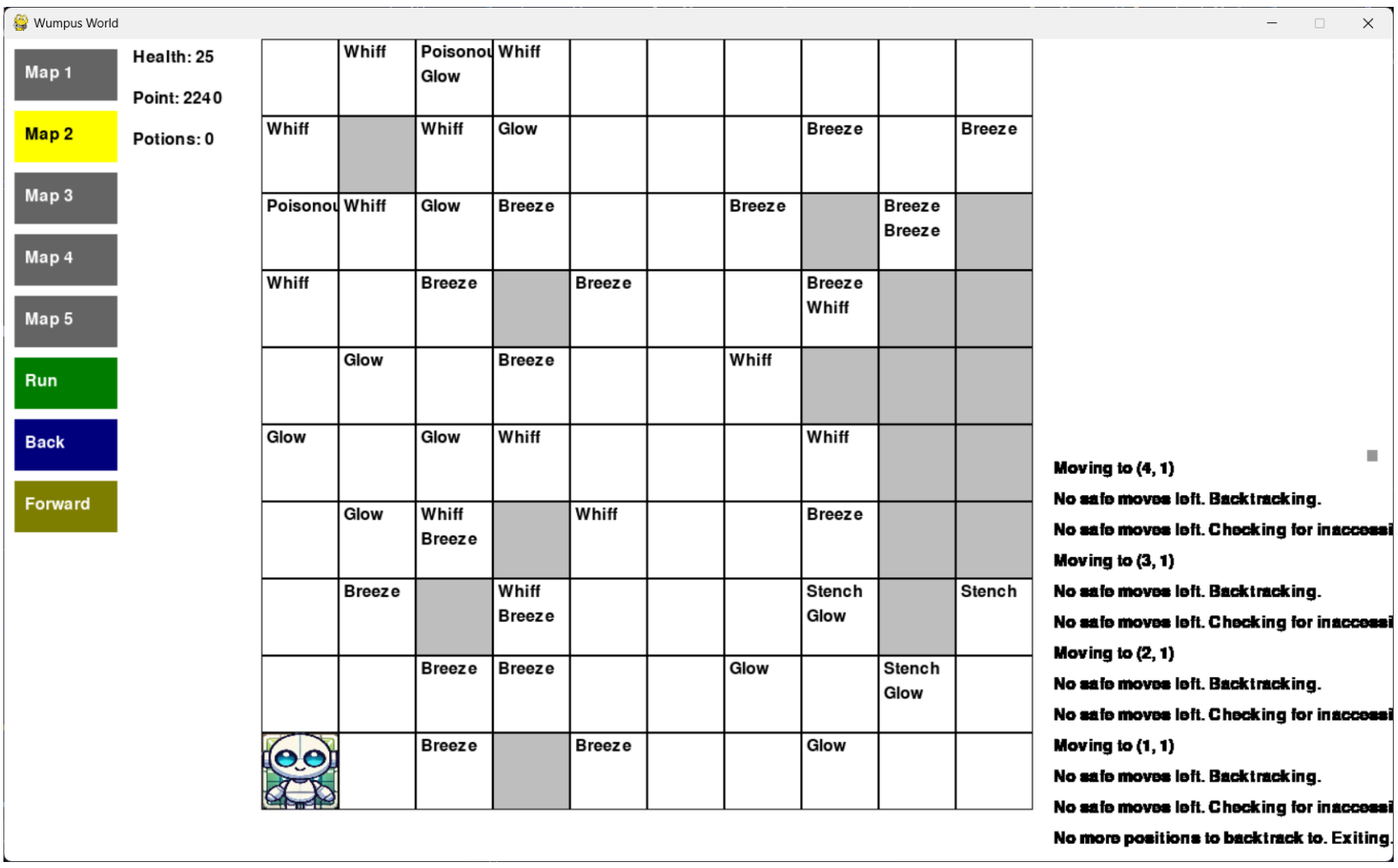
2. Test case 2: Grid 10x10

- File name: `map2.txt`
- Description:
  - 1 Wumpus
  - 1 Gold
  - 6 Pits
  - 3 Healing Potion
  - 4 Poisonous Gas

map2.txt

```
10
-.-.P_G.-.-.-.-.-.-.-
-.-.H_P.-.-.-.-.-.-.-
P_G.-.-.-.-.-.-.P.-.P
.-.-.-.P.-.-.-.-.-.-
-.-.-.G.-.-.-.P_G.-.-
.-.H_P.-.-.-.-.-.-.-.-
-.-.-.P_G.-.-.-.-.P.-
-.-.P.-.-.-.-.-.W.-
.-.-.-.-.-.-.-.H_P.-.-
A.-.-.P.-.-.-.-.-.-
```

- Result:



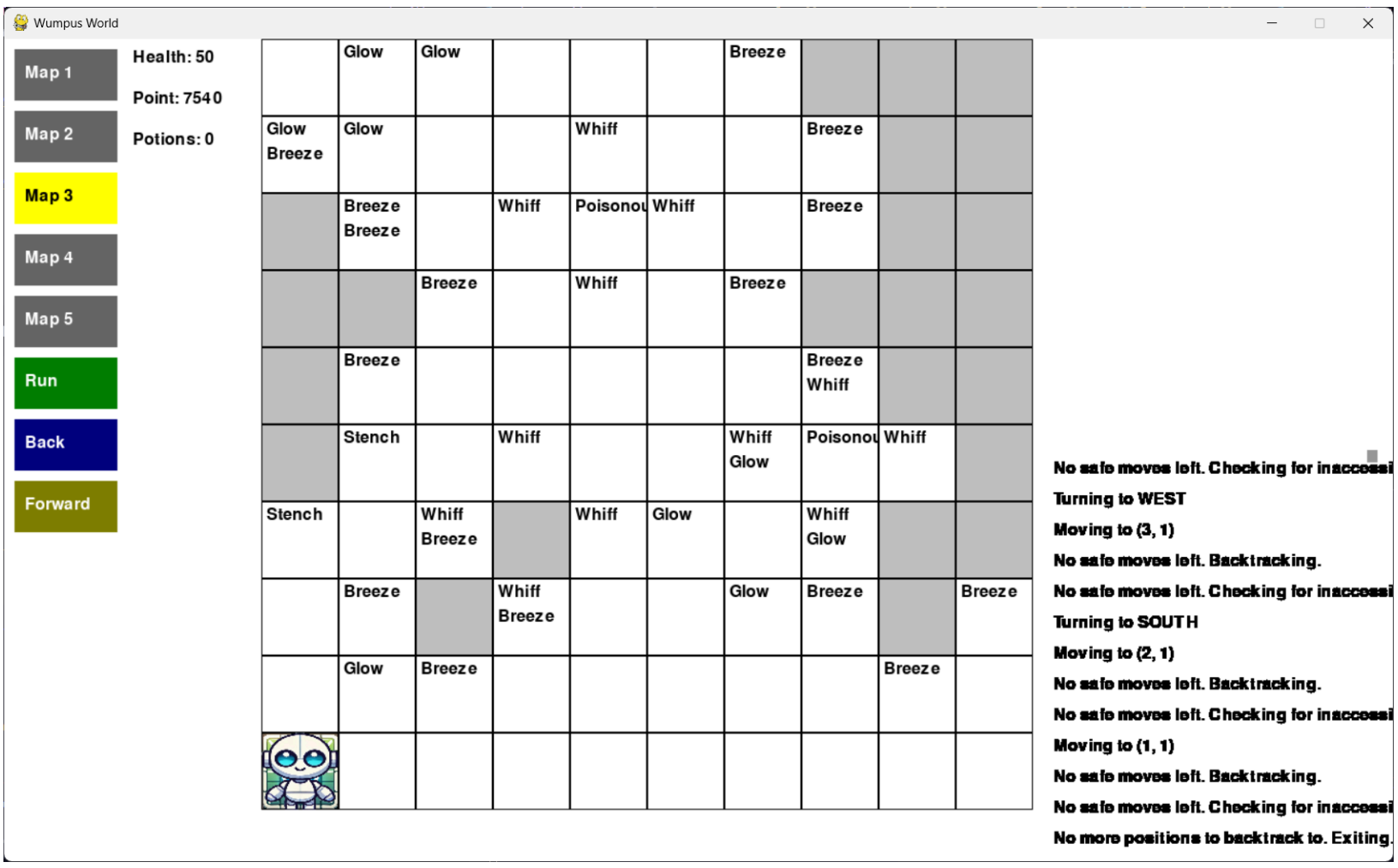
3. Test case 3: Grid 10x10

- File name: `map3.txt`
- Description:
  - 1 Wumpus
  - 1 Gold
  - 6 Pits
  - 4 Healing Potion
  - 3 Poisonous Gas

map3.txt

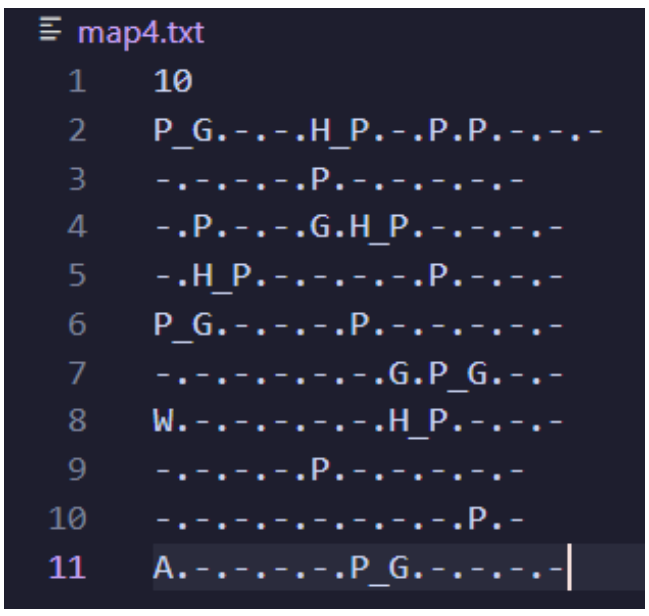
```
10
H_P.-.-.-.-.-.-.P.-.-
-.-.H_P.-.-.-.-.-.-.-
P.-.-.-.P_G.-.-.-.-.-
-.P.-.-.-.G.-.P.-.-
-.-.-.-.-.-.-.-.-.-.-
W.-.-.-.-.-.G.P_G.-.-
-.-.-.P_G.-.-.H_P.-.-.-
-.-.P.-.-.-.-.-.P.-
-.-.H_P.-.-.-.-.-.-.-
A.-.-.-.-.-.-.-.-.-
```

- Result:



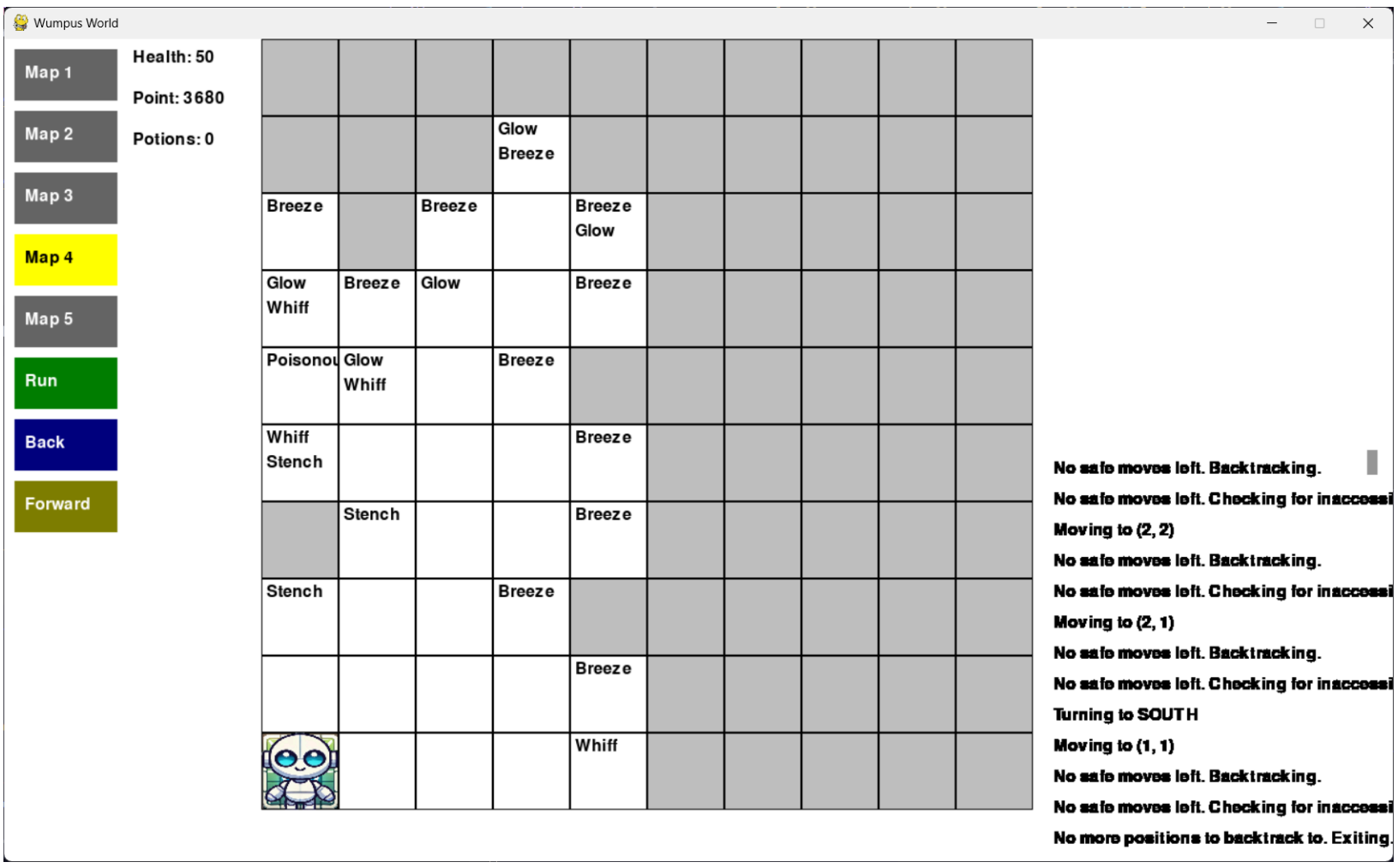
4. Test case 4: Grid 10x10

- File name: `map4.txt`
- Description:
  - 1 Wumpus
  - 1 Gold
  - 9 Pits
  - 4 Healing Potion
  - 4 Poisonous Gas



```
10
P_G.-.-.H_P.-.P.P.-.-.-
-.-.-.-.P.-.-.-.-.-
-.P.-.-.G.H_P.-.-.-.-
-.H_P.-.-.-.-.P.-.-.-
P_G.-.-.-.P.-.-.-.-.-
-.-.-.-.-.-.G.P_G.-.-
W.-.-.-.-.-.H_P.-.-.-
-.-.-.-.P.-.-.-.-.-
-.-.-.-.-.-.-.-.P.-
A.-.-.-.-.P_G.-.-.-.-
```

- Result:



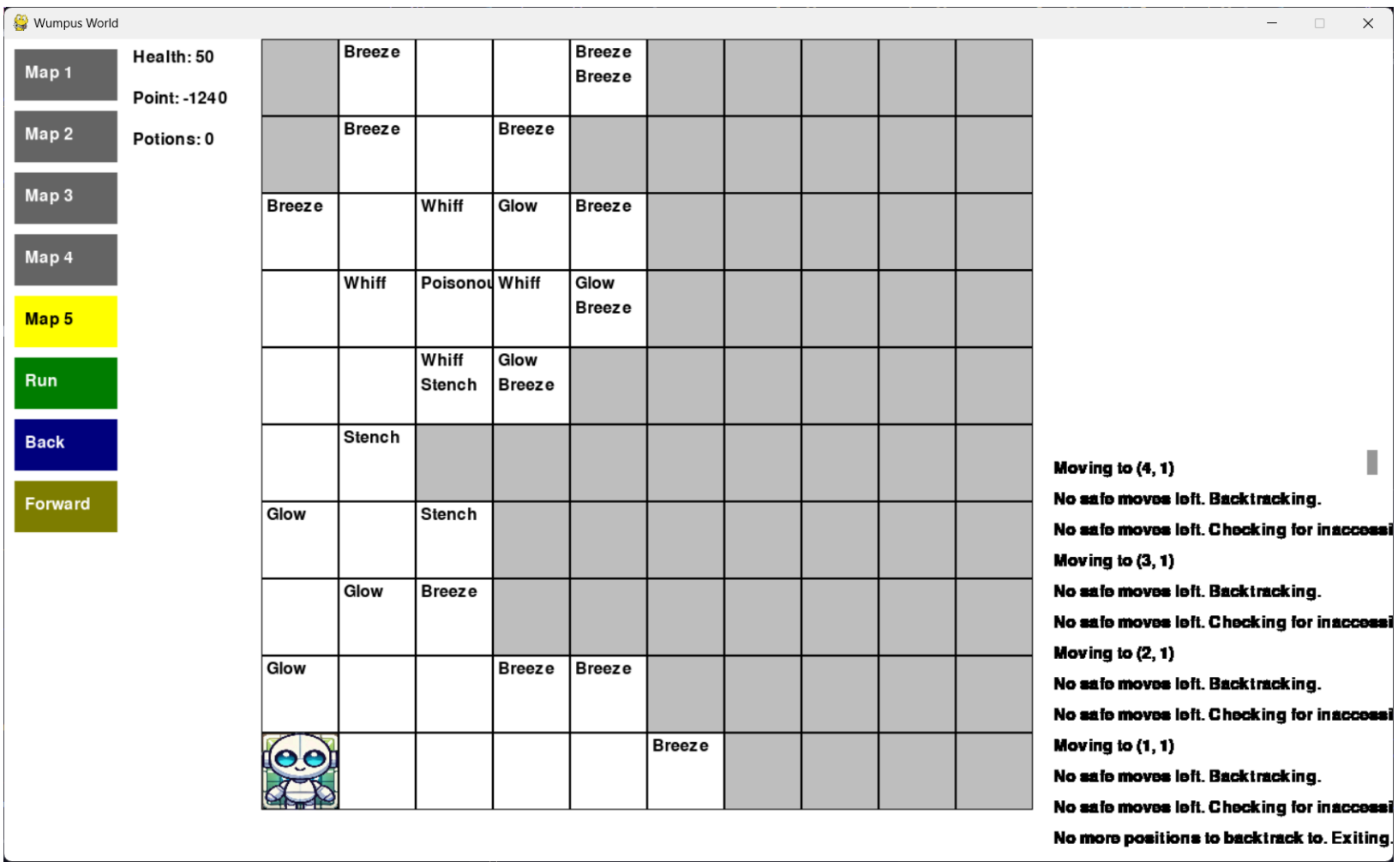
5. Test case 5: Grid 10x10

- File name: `map5.txt`
- Description:
  - 1 Wumpus
  - 1 Gold
  - 11 Pits
  - 5 Healing Potion
  - 4 Poisonous Gas

map5.txt

```
10
P.-.-.-.-.P.-.-.-.-
P.-.-.-.P.-.P.-.-.-
-.-.-.-.-.-.-.-.-.-
-.-.P_G.H_P.-.G.P.P_G.-.-
-.-.-.-.P.-.-.-.-.-
-.-.W.-.P.-.-.-.-.-
-.-.-.-.-.H_P.-.P.-.-
H_P.-.-.P.-.-.-.-.P_G.-
-.-.-.-.-.P.H_P.-.-.H_P
A.-.-.-.-.-.-.-.P_G.-
```

- Result:



VII. References

- Slide from lecturer
- https://www.cs.mcgill.ca/~dprecup/courses/AI/Lectures/ai-lecture08.pdf