Rational Wumpus World Agent

Dawson Pallotta

Artificial Intelligence

May 2024

Introduction

Objective

The objective of this report is to develop a rational agent for the Wumpus World environment, a classic problem in artificial intelligence. The aim is for the agent to explore its surroundings, avoid hazards, and collect the gold within a grid-based world, all while using a reinforcement learning approach to optimize its performance over time. The report also seeks to present evidence of the agent’s rationality by comparing it with a random agent.

Overview

This document is organized into several sections. Section 2, Implementation, details the architecture of the agent, including its logic, decision-making process, and learning methodology. Section 3, Results and Supporting Evidence, presents the outcomes of the agent's behavior and compares it with a random agent to showcase its rationality.

Implementation

Agent Architecture

The agent operates within a 4x4 grid environment, containing various entities, including a dangerous monster known as the Wumpus, hazardous areas called pits, gold as the primary objective, and an exit position. The agent begins in the bottom left corner of the grid, initially facing to the right. Each grid cell provides percepts, which include information about nearby hazards and gold, aiding the agent in deciding its next move. The agent employs a combination of logical inference and reinforcement learning to select its actions.

Decision-Making Process

The decision-making process of the agent follows several key steps. First, the agent checks possible next moves by generating a list of potential actions such as moving forward, turning left, turning right, grabbing, or exiting. For each move, it calculates the new position based on its current facing direction, as implemented in the select_move method. Next, the agent uses model checking to assess the safety of each potential move. The model_check method evaluates the probability of safety for each possible move. If a move is deemed safe, it is added to a list of safe moves.The agent then selects the safest move from the list of safe moves, choosing the one with the highest probability of being safe. If no move is deemed safe, or if the agent wants to explore, it uses an epsilon-greedy strategy to explore other moves. Reinforcement learning is integrated into the agent's decision-making process through Q-learning, which improves the agent's decision-making over time. The action method implements the agent's movement and learning, updating the Q-values based on the outcome of the moves. Lastly, the agent explores new places using the epsilon-greedy strategy, which encourages the agent to explore new positions instead of toggling between known safe places.

Agent Logic

The following pseudocode outlines the main logic of the agent:

```
Function select_move():
    possible_moves = ['move', 'left', 'right', 'grab', 'exit']
    safe_moves = []
    for move in possible_moves:
        next_position = calculate_next_position(move)
        probability_safe = model_check(next_position)
        if probability_safe >= threshold:
            safe_moves.append(move)
    if not safe_moves or random < epsilon:
        return random_move(possible_moves)
    return max_q_value_move(safe_moves)
Function action():
    move = select_move()
    execute_move(move)
    update_q_value(move)
    return move
```

Learning and Adaptation

The agent utilizes reinforcement learning to adapt its behavior based on past experiences and it previously ran simulations. This is achieved through utilizing the reinforcement learning technique of q-learning which updates the agent's knowledge based on rewards received for its actions. The agent keeps a q-table that takes the states and puts them in a pair of actions and rewards to allow the agent to select the actions that maximize its expected reward. The agent uses an epsilon-greedy strategy, balancing exploration and exploitation, where it either selects the action with the highest q-value or explores a new action based on a random choice. The truth table enumeration is incorporated into the agent's learning process as its model-checking mechanism. The rational agent employs logical reasoning to determine the safety of potential moves using a propositional logic representation. The truth table enumeration process involves generating all possible combinations of truth values for the relevant propositions and checking which combinations satisfy the knowledge base and the specific query. This allows the agent to evaluate the probability of safety for each potential move.

Reinforcement Learning and Truth Table Enumeration in Action

In the model checking the agent uses the truth table in order to evaluate the safety of its potential moves. The agent uses the different defined symbols to represent the relevant positions for a given mode and then enumerates all the possible truth assignments. At each truth assignment the agent checks to see if the assignment is able to satisfy both the knowledge base and the query. Doing this allows the agent to determine the probability of the safety of each move, and then updates the q-table. The agent's learning process is further refined through its q-learning algorithm. The action method implements the agent's movement and learning, updating the q-values based on the outcomes of its moves. This method first evaluates the safety of potential moves using the truth table enumeration technique. Then, it selects the move with the highest probability of being safe or explores a new move using the epsilon-greedy strategy. Through reinforcement learning, the agent adapts its behavior to maximize its performance. The agent's learning is guided by its q-values, which are updated based on the rewards it receives. The integration of truth table enumeration allows the agent to accurately assess the safety of potential moves, enhancing its ability to navigate the environment and achieve its objectives.

Results and Supporting Evidence

Experiment Design

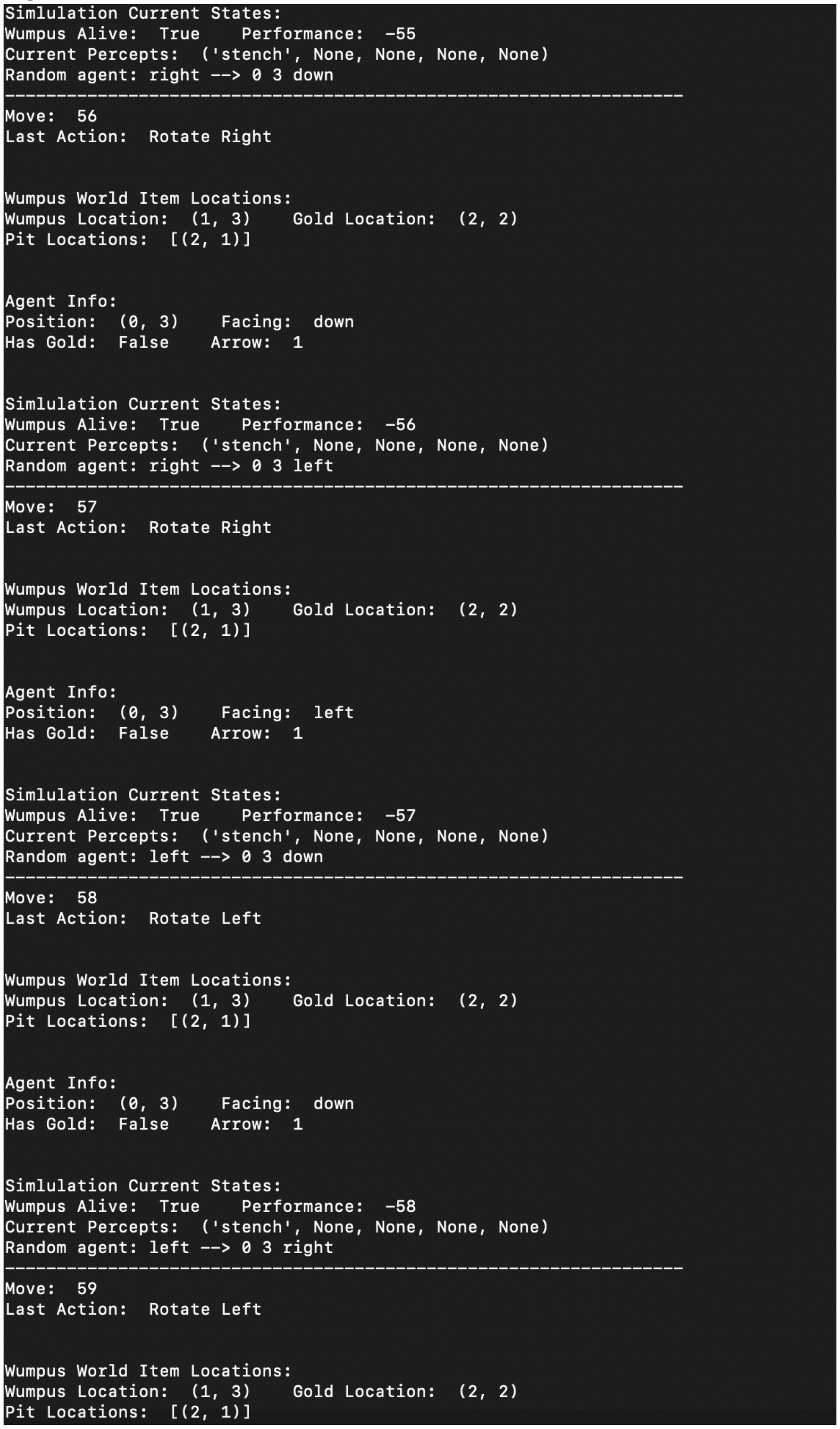
In order to evaluate both the original random agent and the newly created rational agent, the tracking of each of the agents' performance over fifty simulations runs each. .The first test was to run the random agent fifty times, and then the rational agent fifty times. After running the two versions of the wumpus agent the overall performance of the agent was judged by two metrics: the amount of time that agent was able to survive based off of the average survival time, the amount of runs that ended with the agent getting the gold**.** The rational agent consistently performed better in survival time, gold acquisition, and exit achievement compared to the random agent. Out of the 50 times that both agents were ran, the random agent held an average survival time of 13 moves, and only reached the goal 8 times of the 50 runs, whereas the rational agent was able to have a better average survival time by having an average of 19 moves and had acquired the gold in 43 of the 50 runs of the simulation. The figures 1.1-1.5 show some of the ran simulations of the rational agent and its results, and the figures 2.1-2.5 show the rational agent grabbing the gold in most runs and in one instance of it dying as it had fallen into a pit.

Conclusion

The Wumpus World agent developed for this project demonstrated a rational and effective approach to navigating a hazardous environment using logical reasoning and reinforcement learning. The agent's architecture was designed to balance exploration and exploitation, leveraging q-learning and truth table enumeration to inform its decision-making process. By doing so, the agent was able to maximize its performance over time, as evidenced by the experimental results. The decision-making process involved assessing possible moves, evaluating their safety, and then selecting the most promising option based on a combination of logic and learned experience. The agent's learning and adaptation mechanisms allowed it to improve its performance through successive runs, showcasing its ability to learn from past experiences and optimize its actions accordingly. The experimental results provided strong evidence of the agent's rationality, as it consistently outperformed a random agent in key metrics such as survival time, gold acquisition, and successful exit. The rational agent's superior performance highlighted the effectiveness of its design and implementation, demonstrating that it was well-suited to the challenges of the Wumpus World environment. Overall, the development of this rational agent for the Wumpus World problem successfully met the objectives outlined at the beginning of the report. The agent's architecture, decision-making process, and learning methodology were thoroughly examined, and the results provided clear evidence of its rational behavior.

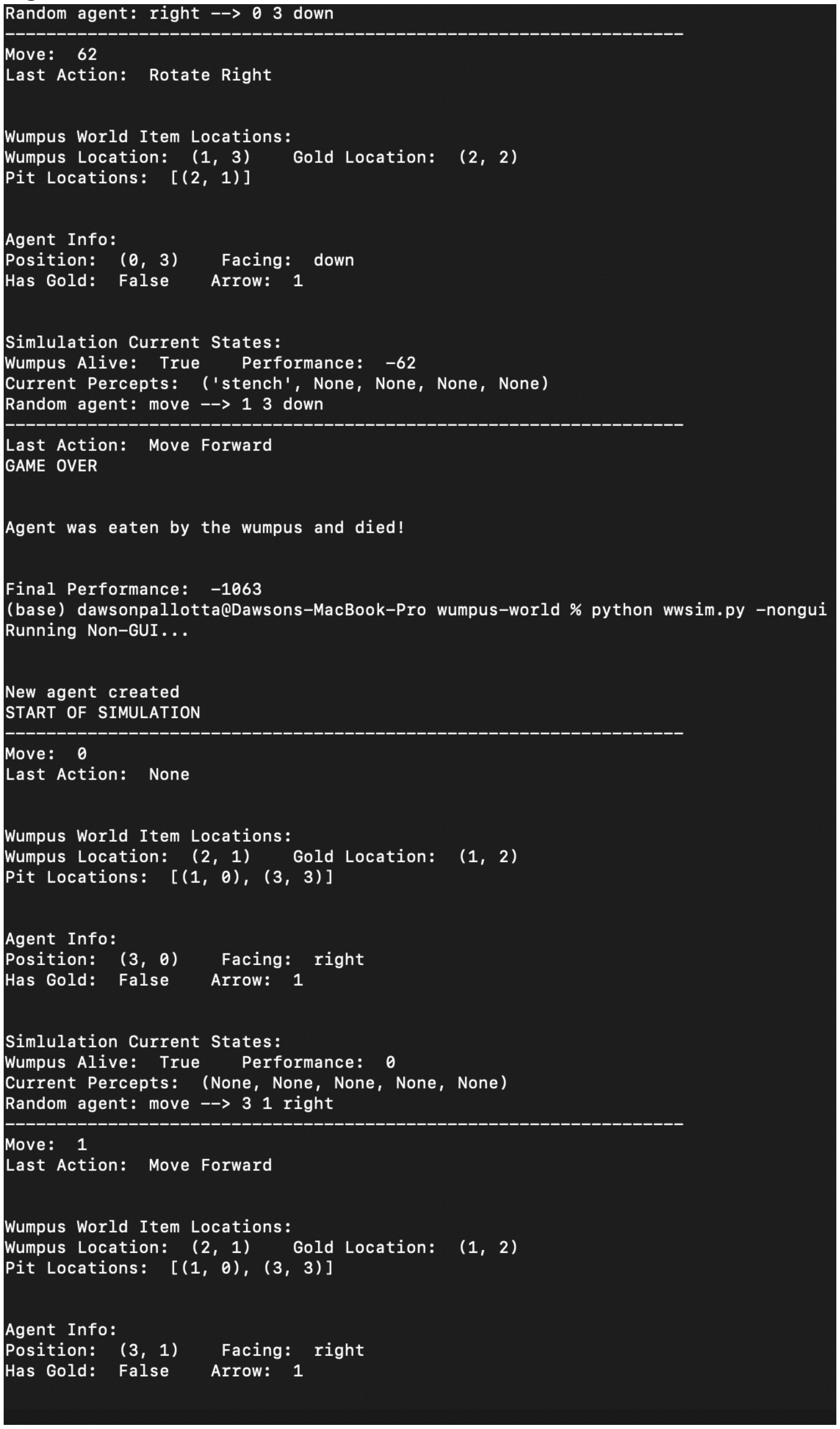
Random Agent Simulation Results

Figure 1. 1



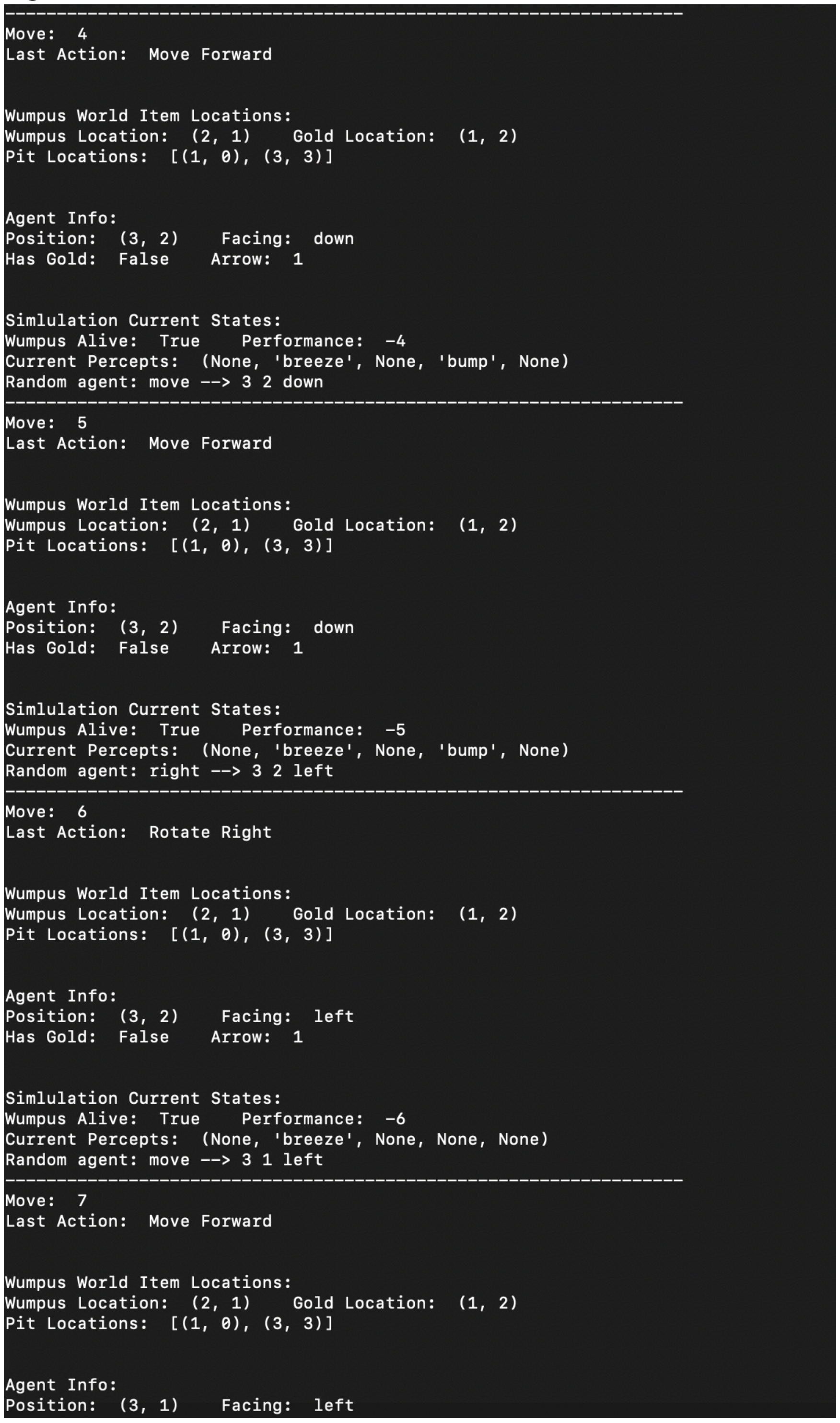
```
Simlulation Current States:
Wumpus Alive:  True    Performance:  -55
Current Percepts:  ('stench', None, None, None, None)
Random agent: right --> 0 3 down
-----------------------------------------------------------------
Move:  56
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 3)    Gold Location:  (2, 2)
Pit Locations:  [(2, 1)]


Agent Info:
Position:  (0, 3)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -56
Current Percepts:  ('stench', None, None, None, None)
Random agent: right --> 0 3 left
-----------------------------------------------------------------
Move:  57
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 3)    Gold Location:  (2, 2)
Pit Locations:  [(2, 1)]


Agent Info:
Position:  (0, 3)    Facing:  left
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -57
Current Percepts:  ('stench', None, None, None, None)
Random agent: left --> 0 3 down
-----------------------------------------------------------------
Move:  58
Last Action:  Rotate Left


Wumpus World Item Locations:
Wumpus Location:  (1, 3)    Gold Location:  (2, 2)
Pit Locations:  [(2, 1)]


Agent Info:
Position:  (0, 3)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -58
Current Percepts:  ('stench', None, None, None, None)
Random agent: left --> 0 3 right
-----------------------------------------------------------------
Move:  59
Last Action:  Rotate Left


Wumpus World Item Locations:
Wumpus Location:  (1, 3)    Gold Location:  (2, 2)
Pit Locations:  [(2, 1)]
```

Figure 1.2:



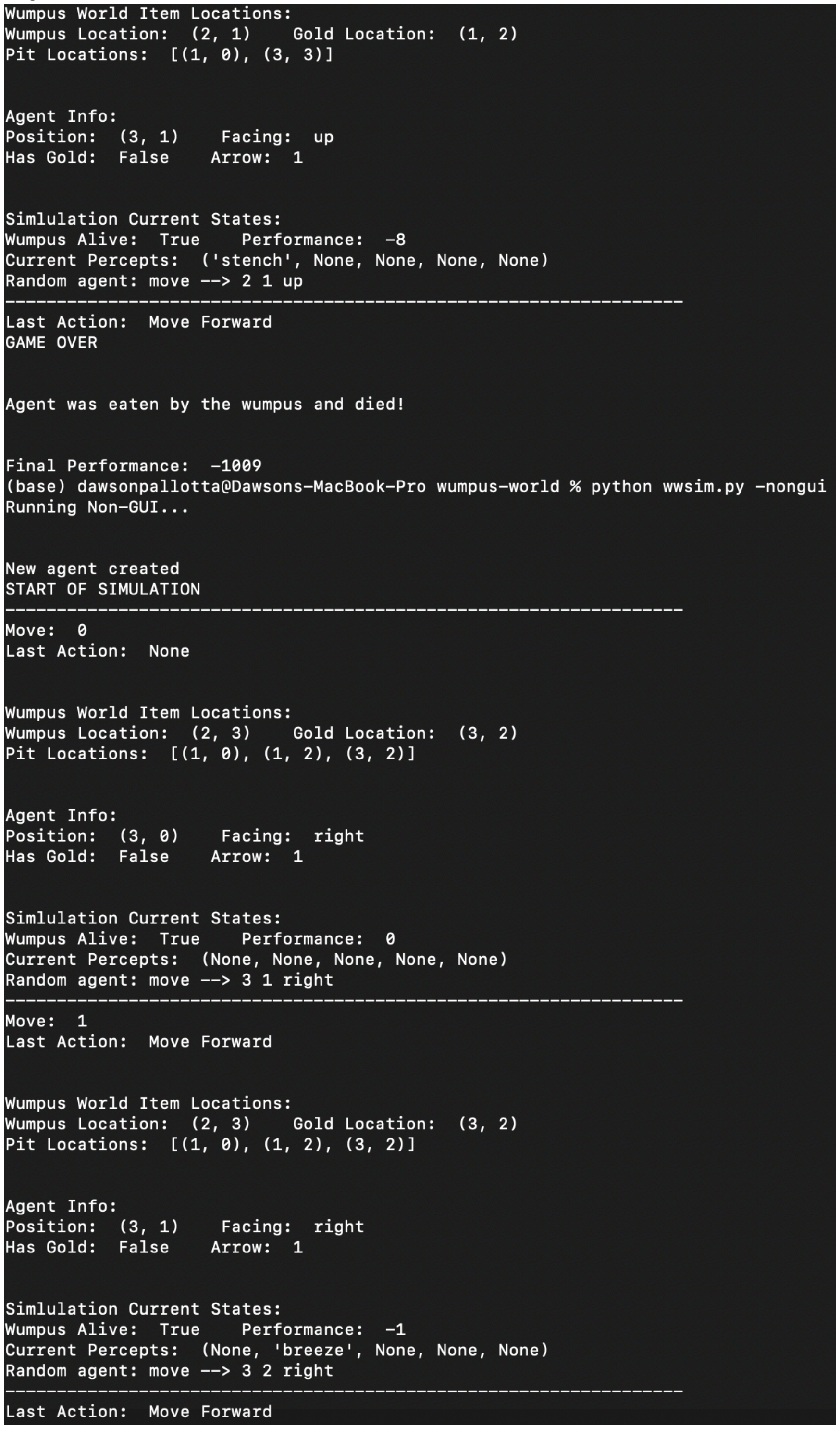
```
Random agent: right --> 0 3 down
-------------------------------------------------------------------
Move:  62
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 3)    Gold Location:  (2, 2)
Pit Locations:  [(2, 1)]


Agent Info:
Position:  (0, 3)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -62
Current Percepts:  ('stench', None, None, None, None)
Random agent: move --> 1 3 down
-------------------------------------------------------------------
Last Action:  Move Forward
GAME OVER


Agent was eaten by the wumpus and died!


Final Performance:  -1063
(base) dawsonpallotta@Dawsons-MacBook-Pro wumpus-world % python wwsim.py -nongui
Running Non-GUI...


New agent created
START OF SIMULATION
-------------------------------------------------------------------
Move:  0
Last Action:  None


Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 0)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  0
Current Percepts:  (None, None, None, None, None)
Random agent: move --> 3 1 right
-------------------------------------------------------------------
Move:  1
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 1)    Facing:  right
Has Gold:  False    Arrow:  1
```

Figure 1.3:



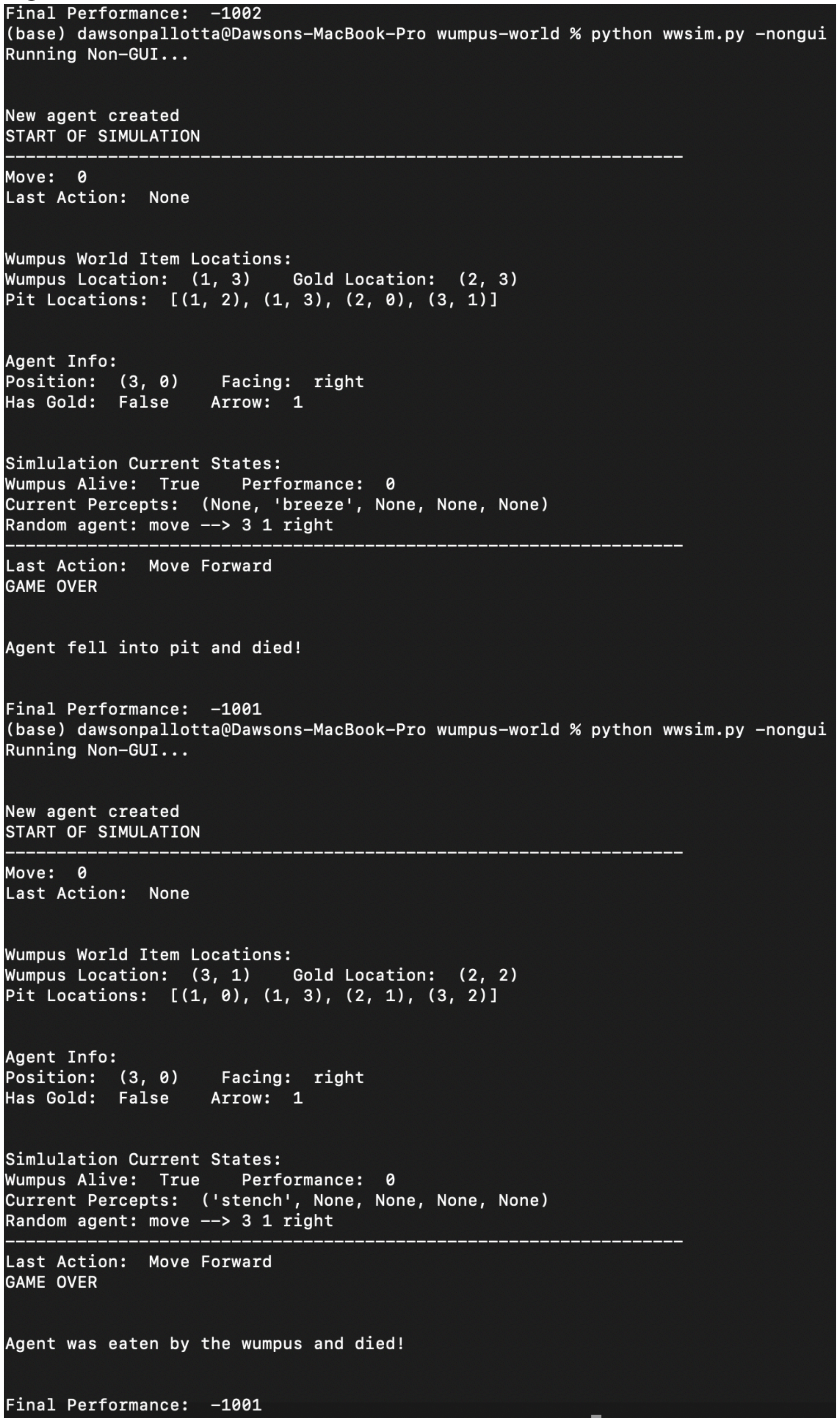
```
-------------------------------------------------------------------
Move:  4
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 2)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -4
Current Percepts:  (None, 'breeze', None, 'bump', None)
Random agent: move --> 3 2 down
-------------------------------------------------------------------
Move:  5
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 2)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -5
Current Percepts:  (None, 'breeze', None, 'bump', None)
Random agent: right --> 3 2 left
-------------------------------------------------------------------
Move:  6
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 2)    Facing:  left
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -6
Current Percepts:  (None, 'breeze', None, None, None)
Random agent: move --> 3 1 left
-------------------------------------------------------------------
Move:  7
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 1)    Facing:  left
```

Figure 1.4



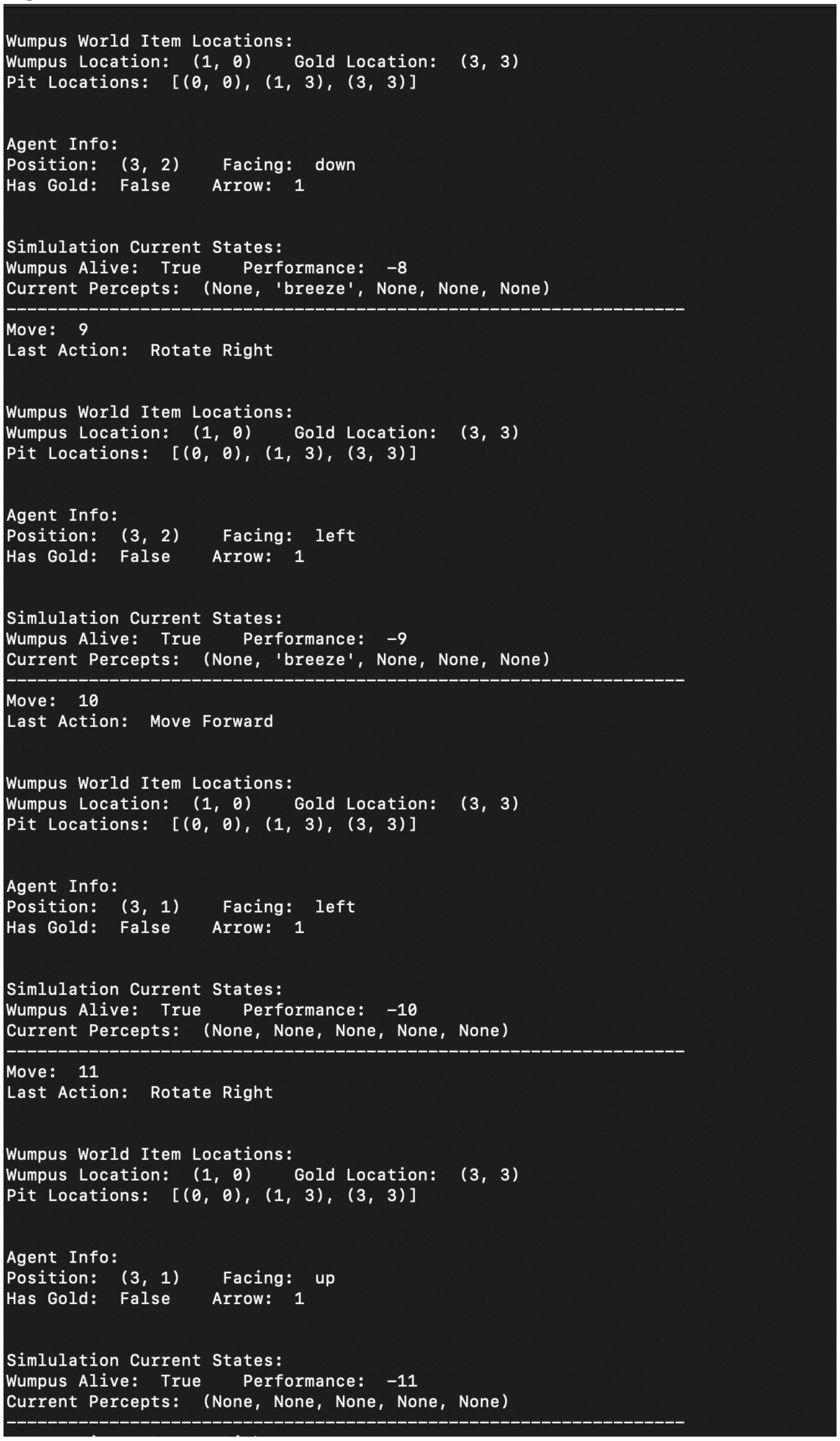
```
Wumpus World Item Locations:
Wumpus Location:  (2, 1)    Gold Location:  (1, 2)
Pit Locations:  [(1, 0), (3, 3)]


Agent Info:
Position:  (3, 1)    Facing:  up
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -8
Current Percepts:  ('stench', None, None, None, None)
Random agent: move --> 2 1 up
------------------------------------------------------------------
Last Action:  Move Forward
GAME OVER


Agent was eaten by the wumpus and died!


Final Performance:  -1009
(base) dawsonpallotta@Dawsons-MacBook-Pro wumpus-world % python wwsim.py -nongui
Running Non-GUI...


New agent created
START OF SIMULATION
------------------------------------------------------------------
Move:  0
Last Action:  None


Wumpus World Item Locations:
Wumpus Location:  (2, 3)    Gold Location:  (3, 2)
Pit Locations:  [(1, 0), (1, 2), (3, 2)]


Agent Info:
Position:  (3, 0)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  0
Current Percepts:  (None, None, None, None, None)
Random agent: move --> 3 1 right
------------------------------------------------------------------
Move:  1
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (2, 3)    Gold Location:  (3, 2)
Pit Locations:  [(1, 0), (1, 2), (3, 2)]


Agent Info:
Position:  (3, 1)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -1
Current Percepts:  (None, 'breeze', None, None, None)
Random agent: move --> 3 2 right
------------------------------------------------------------------
Last Action:  Move Forward
```

Figure 1.5



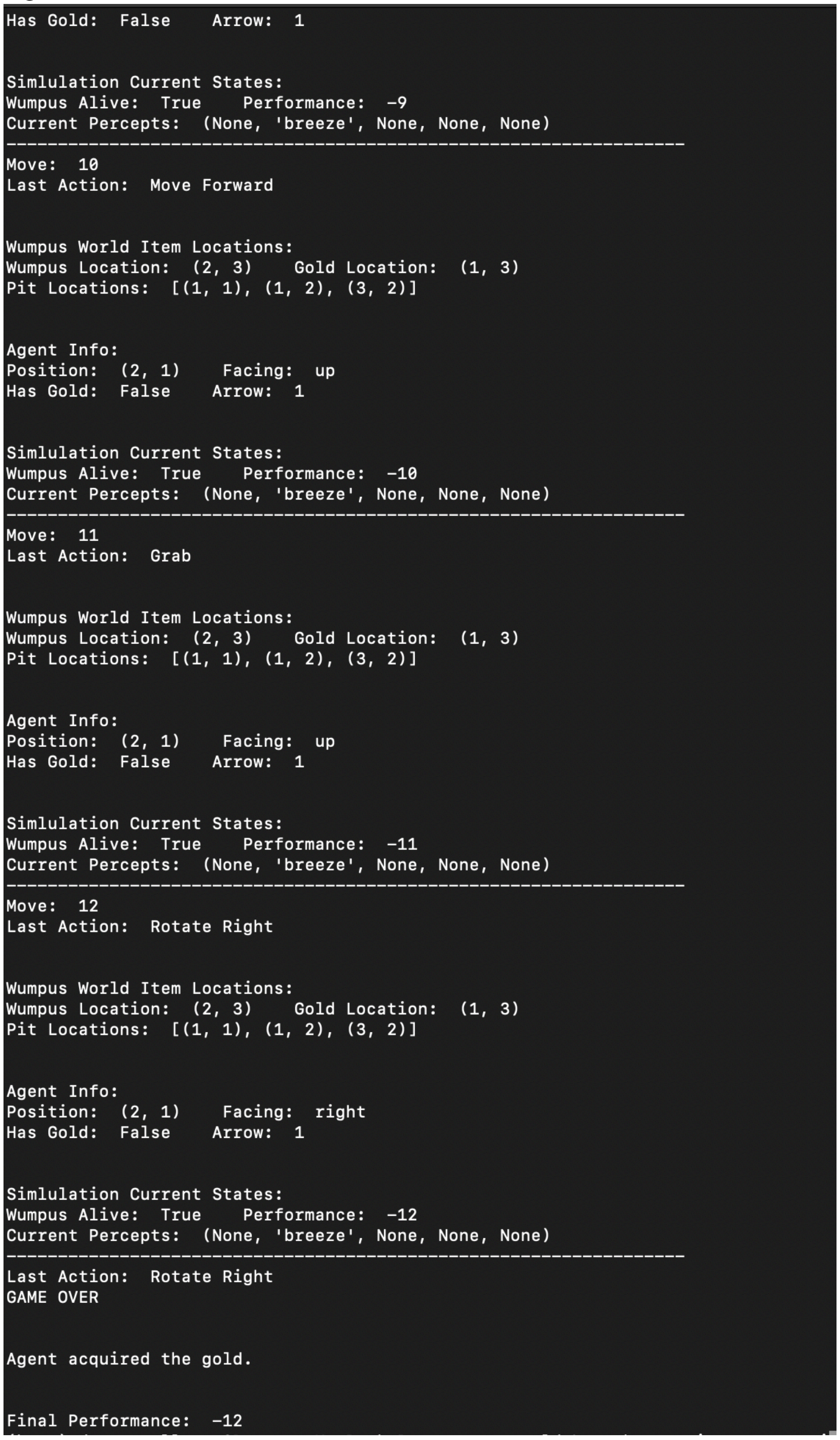
```
Final Performance:  -1002
(base) dawsonpallotta@Dawsons-MacBook-Pro wumpus-world % python wwsim.py -nongui
Running Non-GUI...


New agent created
START OF SIMULATION
------------------------------------------------------------------
Move:  0
Last Action:  None


Wumpus World Item Locations:
Wumpus Location:  (1, 3)    Gold Location:  (2, 3)
Pit Locations:  [(1, 2), (1, 3), (2, 0), (3, 1)]


Agent Info:
Position:  (3, 0)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  0
Current Percepts:  (None, 'breeze', None, None, None)
Random agent: move --> 3 1 right
------------------------------------------------------------------
Last Action:  Move Forward
GAME OVER


Agent fell into pit and died!


Final Performance:  -1001
(base) dawsonpallotta@Dawsons-MacBook-Pro wumpus-world % python wwsim.py -nongui
Running Non-GUI...


New agent created
START OF SIMULATION
------------------------------------------------------------------
Move:  0
Last Action:  None


Wumpus World Item Locations:
Wumpus Location:  (3, 1)    Gold Location:  (2, 2)
Pit Locations:  [(1, 0), (1, 3), (2, 1), (3, 2)]


Agent Info:
Position:  (3, 0)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  0
Current Percepts:  ('stench', None, None, None, None)
Random agent: move --> 3 1 right
------------------------------------------------------------------
Last Action:  Move Forward
GAME OVER


Agent was eaten by the wumpus and died!


Final Performance:  -1001
```

Rational Agent Simulation Results
Figure 2.1



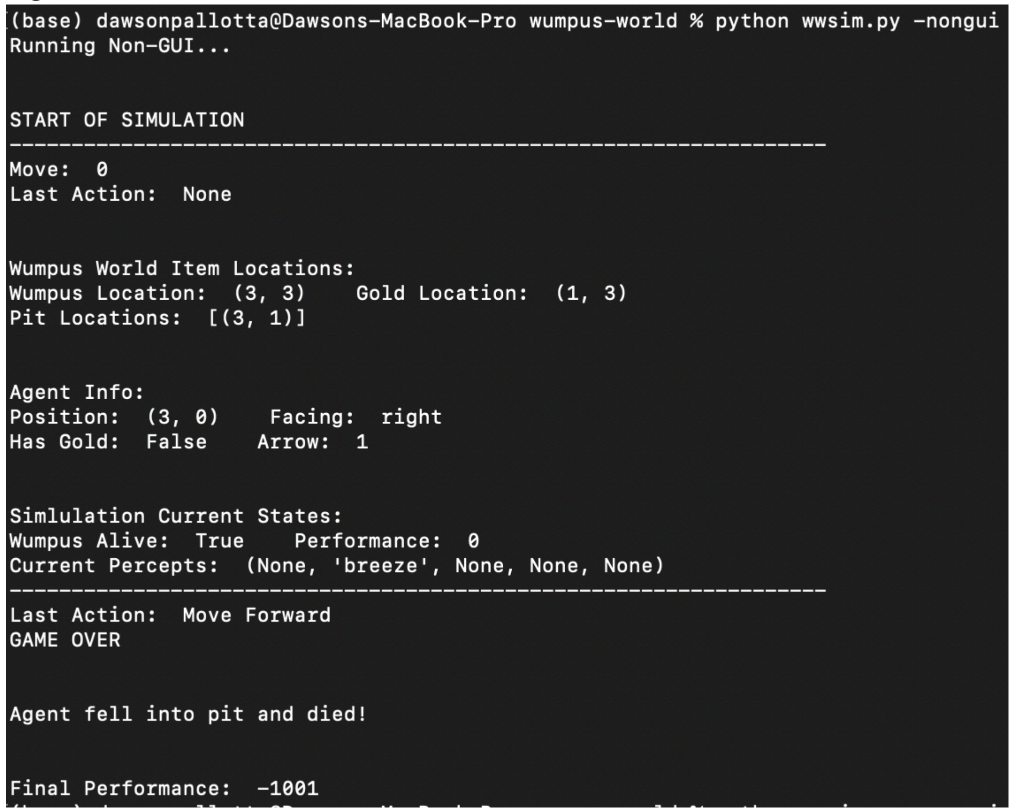
```
Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (3, 3)
Pit Locations:  [(0, 0), (1, 3), (3, 3)]


Agent Info:
Position:  (3, 2)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -8
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Move:  9
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (3, 3)
Pit Locations:  [(0, 0), (1, 3), (3, 3)]


Agent Info:
Position:  (3, 2)    Facing:  left
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -9
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Move:  10
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (3, 3)
Pit Locations:  [(0, 0), (1, 3), (3, 3)]


Agent Info:
Position:  (3, 1)    Facing:  left
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -10
Current Percepts:  (None, None, None, None, None)
------------------------------------------------------------------
Move:  11
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (3, 3)
Pit Locations:  [(0, 0), (1, 3), (3, 3)]


Agent Info:
Position:  (3, 1)    Facing:  up
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -11
Current Percepts:  (None, None, None, None, None)
------------------------------------------------------------------
```

Figure 2.2



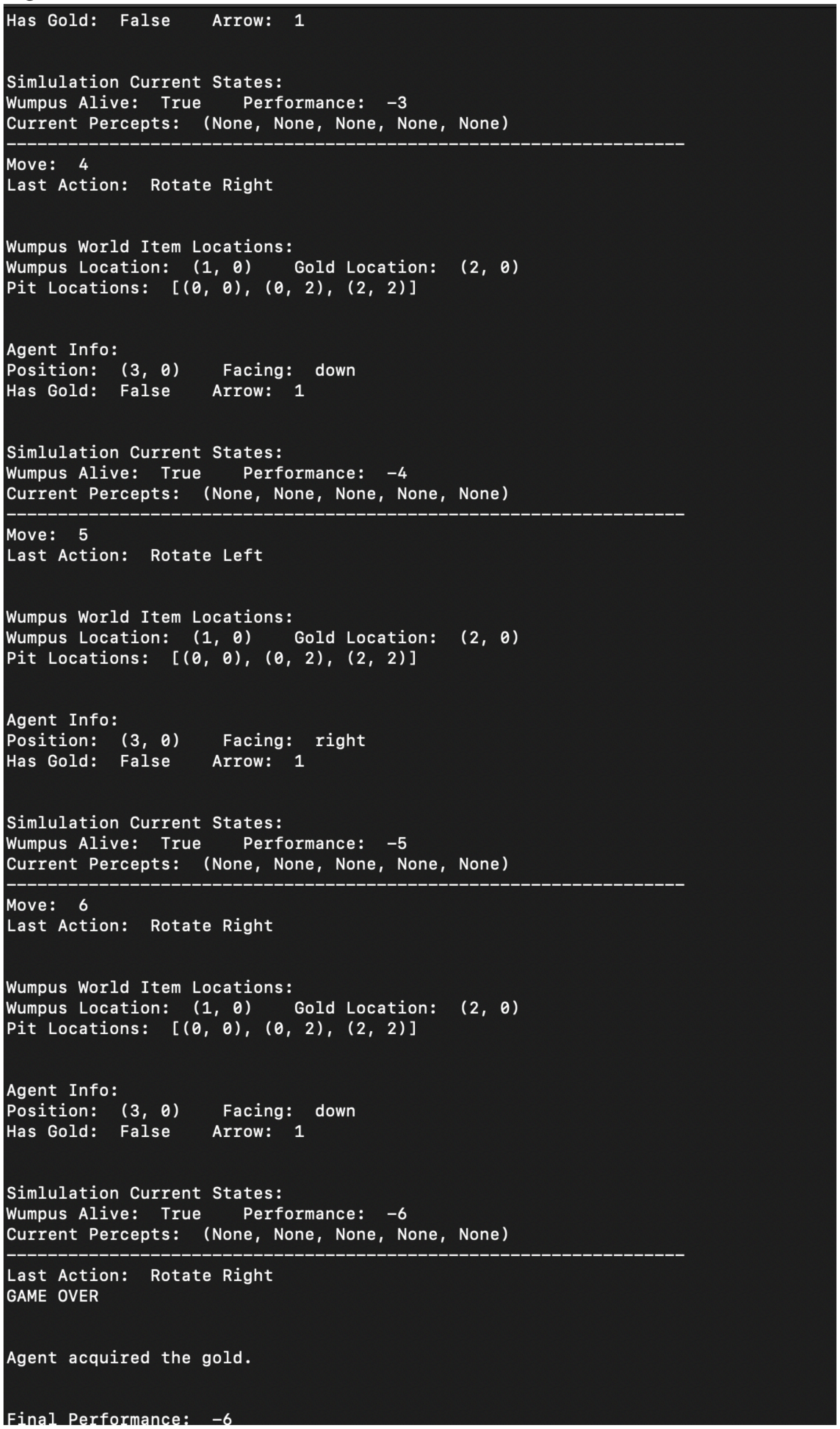
```
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -9
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Move:  10
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (2, 3)    Gold Location:  (1, 3)
Pit Locations:  [(1, 1), (1, 2), (3, 2)]


Agent Info:
Position:  (2, 1)    Facing:  up
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -10
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Move:  11
Last Action:  Grab


Wumpus World Item Locations:
Wumpus Location:  (2, 3)    Gold Location:  (1, 3)
Pit Locations:  [(1, 1), (1, 2), (3, 2)]


Agent Info:
Position:  (2, 1)    Facing:  up
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -11
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Move:  12
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (2, 3)    Gold Location:  (1, 3)
Pit Locations:  [(1, 1), (1, 2), (3, 2)]


Agent Info:
Position:  (2, 1)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -12
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Last Action:  Rotate Right
GAME OVER


Agent acquired the gold.


Final Performance:  -12
```

Figure 2.3



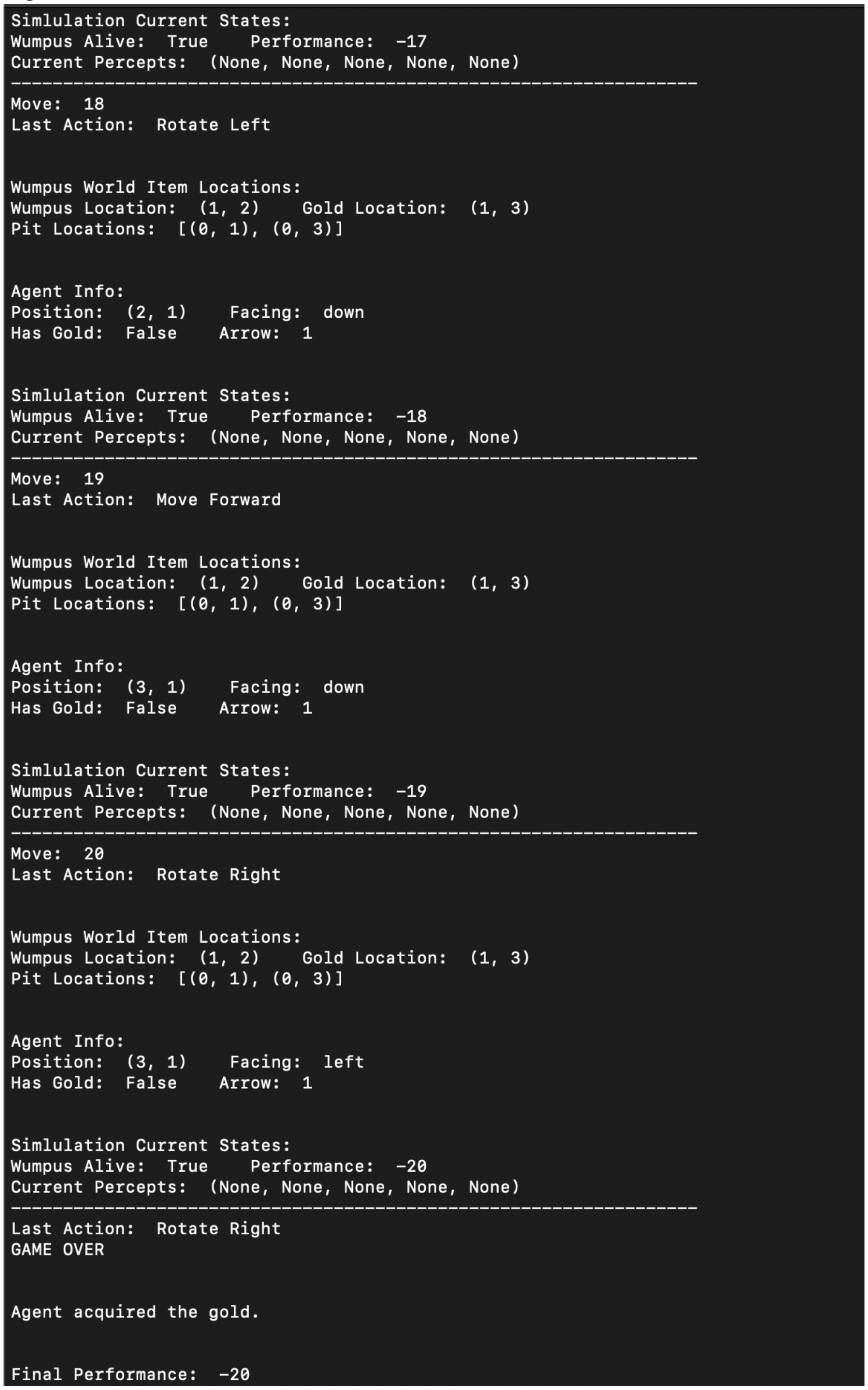
```
(base) dawsonpallotta@Dawsons-MacBook-Pro wumpus-world % python wwsim.py -nongui
Running Non-GUI...


START OF SIMULATION
------------------------------------------------------------------
Move:  0
Last Action:  None


Wumpus World Item Locations:
Wumpus Location:  (3, 3)    Gold Location:  (1, 3)
Pit Locations:  [(3, 1)]


Agent Info:
Position:  (3, 0)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  0
Current Percepts:  (None, 'breeze', None, None, None)
------------------------------------------------------------------
Last Action:  Move Forward
GAME OVER


Agent fell into pit and died!


Final Performance:  -1001
```

Figure 2.4



```
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -3
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Move:  4
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (2, 0)
Pit Locations:  [(0, 0), (0, 2), (2, 2)]


Agent Info:
Position:  (3, 0)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -4
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Move:  5
Last Action:  Rotate Left


Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (2, 0)
Pit Locations:  [(0, 0), (0, 2), (2, 2)]


Agent Info:
Position:  (3, 0)    Facing:  right
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -5
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Move:  6
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 0)    Gold Location:  (2, 0)
Pit Locations:  [(0, 0), (0, 2), (2, 2)]


Agent Info:
Position:  (3, 0)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -6
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Last Action:  Rotate Right
GAME OVER


Agent acquired the gold.


Final Performance:  -6
```

Figure 2.5



```
Simlulation Current States:
Wumpus Alive:  True    Performance:  -17
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Move:  18
Last Action:  Rotate Left


Wumpus World Item Locations:
Wumpus Location:  (1, 2)    Gold Location:  (1, 3)
Pit Locations:  [(0, 1), (0, 3)]


Agent Info:
Position:  (2, 1)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -18
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Move:  19
Last Action:  Move Forward


Wumpus World Item Locations:
Wumpus Location:  (1, 2)    Gold Location:  (1, 3)
Pit Locations:  [(0, 1), (0, 3)]


Agent Info:
Position:  (3, 1)    Facing:  down
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -19
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Move:  20
Last Action:  Rotate Right


Wumpus World Item Locations:
Wumpus Location:  (1, 2)    Gold Location:  (1, 3)
Pit Locations:  [(0, 1), (0, 3)]


Agent Info:
Position:  (3, 1)    Facing:  left
Has Gold:  False    Arrow:  1


Simlulation Current States:
Wumpus Alive:  True    Performance:  -20
Current Percepts:  (None, None, None, None, None)
-------------------------------------------------------------------
Last Action:  Rotate Right
GAME OVER


Agent acquired the gold.


Final Performance:  -20
```

help me take this code and make to do the following, basically help take the current agent and make a Model Checking Agent, a Probabilistic Model checking agent, and a Planning-based Agent(Monte Carlo) : Your replacement code should Check what positions around the current position of the agent are possible next moves Version 1: For each possible move, use model checking to see if that move is safe (i.e., if there is a pit or wumpus there) Version 2: Pick the move that has the highest probability of being safe Version 3: Using RL to improve the move selected the model checking should be done through truth-table enumeration-based entailment (model checking) to estimate probabilities Your agent will move through the Wumpus word and find the gold, winning the game. 2. If there is a 100% guaranteed safe move for the agent from its current position, then it will take that move. 3. If there is no 100% safe move, then the agent will take the safest move it can, based on its estimate of the probability of a wumpus or pit being in the target location. 4. It uses indirect model-based RL to improve its selection of the safest move and overall performance. You have to implement your reasoning using an implementation of truth-table enumeration to determine at the earliest moment possible given the percepts whether a location is safe for the wumpus or not. You cannot use any other method to do this. A large component of the grade will be given for this. You do not need to represent the rules of wumpus world declaratively – you can build them into a procedure for example that calculates the truth table for the Knowledge Base. Version 2: Probabilistic model checking - Model checking will return True if and only if the query alpha is True in the KB and in all models. Probabilistic model checking COUNTS how many models both the KB and alpha is true and divides that by the number of models in which only the KB is true. Modify the model checking code to increment a counter whenever the KB is true and a second counter whenever alpha is also true. Once all models have been checked, just divide one by the other and that is the probability of alpha being true! Version 3: Reinforcement learning - Your Q table should be indexed by 'state' and state should include the location on the grid, probability that the location is safe, and the actions you can perform. The reward for the goal should be positive and for getting killed should be negative. Each empty room can have a small negative reward. You can use MC or TD learning to update the Q table. You should pick a move based on the highest probability of being safe with probability (1-) with based on the Q table with . can start low and should increase with episodes to simplify the model checking: INSTEAD OF isTrue(KB,model), write a function returning True/False that will check whether the model obeys the rules of the game as follows: The model will be a list of assignments of True and False to ALL the propositions, e.g., model=[(b11, True), (b12, False), ……] Write for loops that check model to see if there are breezes in rooms around pits, and pits whenever breezes and ditto for wumpus/stenches Write a for loop that checks all the percepts in self.map and anything you know about pits also holds in this model. Alpha is: that there is no wumpus or pit in the room you want to considering moving to. Let say that is room (i, j). All you need to do is write a function that checks that pij is false in the model and wij is false in the model and only returns True if they are When you carry out an action, you will need to update the agent's class variables yourself to represent where you expect the agent to be. calculateNextPosition – this will calculate what position the agent will be at if it

carries out a move forward. calculateNextDirection – this will calculate where the agent is facing after it carries out the turn action given as argument. Both of these are useful procedures but you don't have to make use of them – you can write your own. __init__ -- initializes the agent class variables. Key variables include the agent's current position, percepts and the map/history of percepts update – updates the map/history of percepts with the current percepts. This is called by wwsim just before each move to give the agent the percept data. action – called by wwsim to generate the next action for the agent. This is the main function you need to modify. The function returns which action the agent has selected to do next: move forward ('move'), turn left ('left), turn right ('right') or grab the gold ('grab).
If you return 'exit' the game will end.
Q-table can be editing into the simulation by changing the last bit of the wwsim file to create a loop that runs the simulation multiple times so that the q-table can save values throught the running sumulations