جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⴰⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN

UNIVERSITY

Artificial intelligence Project 2 Report

Wumpus Project

Chifaa Bouzid

82723

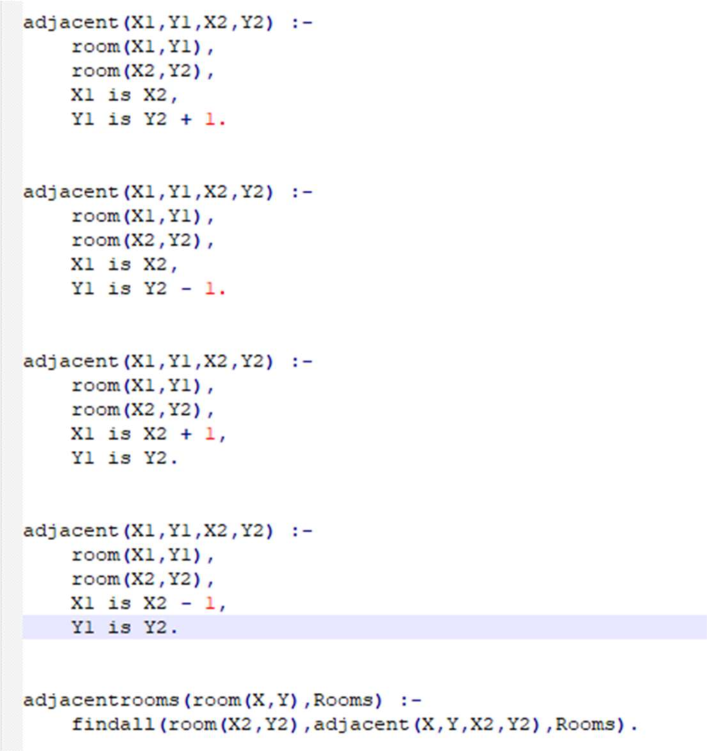
CSC 4301 01

Dr. Tajjeddine Rachidi

March 21, 2022

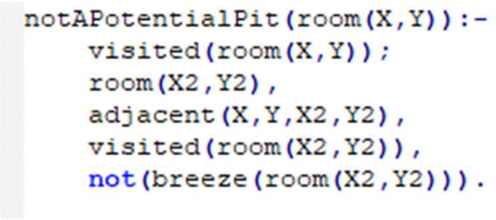
Teammate: Ayoub Mabkhout

Key Predicates and the Meaning of the Variables

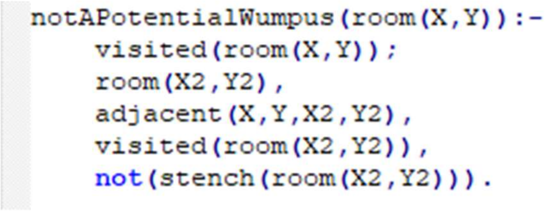


```
adjacent(X1,Y1,X2,Y2) :-
    room(X1,Y1),
    room(X2,Y2),
    X1 is X2,
    Y1 is Y2 + 1.


adjacent(X1,Y1,X2,Y2) :-
    room(X1,Y1),
    room(X2,Y2),
    X1 is X2,
    Y1 is Y2 - 1.


adjacent(X1,Y1,X2,Y2) :-
    room(X1,Y1),
    room(X2,Y2),
    X1 is X2 + 1,
    Y1 is Y2.


adjacent(X1,Y1,X2,Y2) :-
    room(X1,Y1),
    room(X2,Y2),
    X1 is X2 - 1,
    Y1 is Y2.


adjacentrooms(room(X,Y),Rooms) :-
    findall(room(X2,Y2),adjacent(X,Y,X2,Y2),Rooms).
```

The first key predicate is to get the adjacent rooms of a specific room. In the predicate adjacentrooms, the arguments are the room that we would like to get its adjacent rooms and Rooms which is the list in which we want the result to be returned (in this case the adjacent rooms.) On the other hand, adjacent is a predicate that returns the coordinates of the adjacent rooms in X2 and Y2.

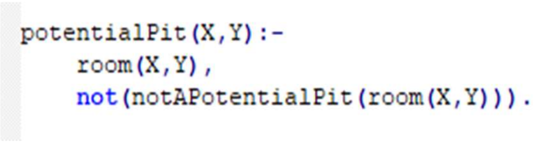


```
notAPotentialPit(room(X,Y)):-
    visited(room(X,Y));
    room(X2,Y2),
    adjacent(X,Y,X2,Y2),
    visited(room(X2,Y2)),
    not(breeze(room(X2,Y2))).
```

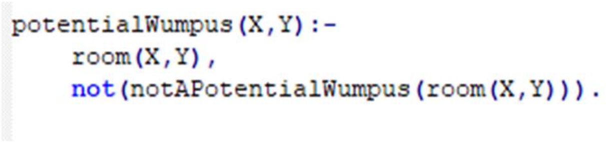


```
notAPotentialWumpus(room(X,Y)):-
    visited(room(X,Y));
    room(X2,Y2),
    adjacent(X,Y,X2,Y2),
    visited(room(X2,Y2)),
    not(stench(room(X2,Y2))).
```

The other two key predicates that share the same concept are notAPotentialPit and notAPotentialWumpus. The logic used here is that if a room is already visited or has an adjacent room which is also visited but does not have any sign (breeze or stench), then the room does not have a potential pit/wumpus. For the variables, X and Y are used for the room that we would lie to check and X2 and Y2 are used for the coordinates of the adjacent room of the concerned room.

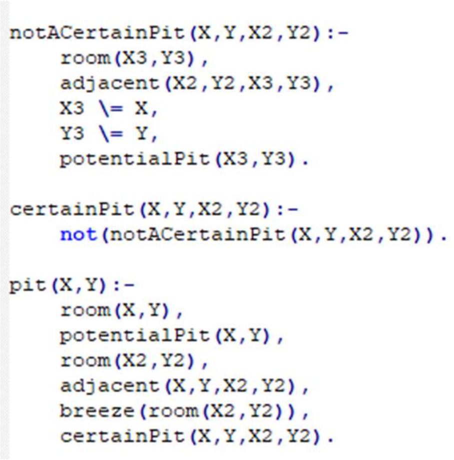


```
potentialPit(X,Y):-
    room(X,Y),
    not(notAPotentialPit(room(X,Y))).
```



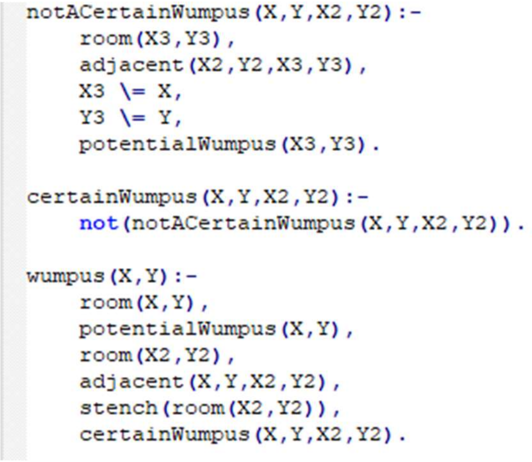
```
potentialWumpus(X,Y):-
    room(X,Y),
    not(notAPotentialWumpus(room(X,Y))).
```

The above predicates check if a room with cooredinate X and Y has a potential pit/wumpus. The logic used the negation of the notAPotentialWumpus predicate since the two predicates are opposites.

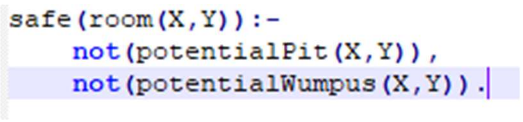


```
notACertainPit(X,Y,X2,Y2):-
    room(X3,Y3),
    adjacent(X2,Y2,X3,Y3),
    X3 \= X,
    Y3 \= Y,
    potentialPit(X3,Y3).

certainPit(X,Y,X2,Y2):-
    not(notACertainPit(X,Y,X2,Y2)).

pit(X,Y):-
    room(X,Y),
    potentialPit(X,Y),
    room(X2,Y2),
    adjacent(X,Y,X2,Y2),
    breeze(room(X2,Y2)),
    certainPit(X,Y,X2,Y2).
```

The above kep predicates are used to see if there is a pit in a room with coordinates X and Y. The pit predicate checks if there is a potential pit in the room, if there is a breeze in its adjacent room, and if it has a certain pit. The latte is anotehr predicate that takes the negation of notACertainPit predicate. The latter checks if the adjacent room (X3, Y3) to the adjacent room (X2, Y2) of the concerned room (X, Y) has a potential pit or not. If it does, then we can only infer that there is no certain pit in room with coordinates X and Y. The same logic is used for wumpus related predicates as shown in the image below:

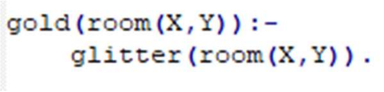


```
notACertainWumpus(X,Y,X2,Y2):-
    room(X3,Y3),
    adjacent(X2,Y2,X3,Y3),
    X3 \= X,
    Y3 \= Y,
    potentialWumpus(X3,Y3).

certainWumpus(X,Y,X2,Y2):-
    not(notACertainWumpus(X,Y,X2,Y2)).

wumpus(X,Y):-
    room(X,Y),
    potentialWumpus(X,Y),
    room(X2,Y2),
    adjacent(X,Y,X2,Y2),
    stench(room(X2,Y2)),
    certainWumpus(X,Y,X2,Y2).
```



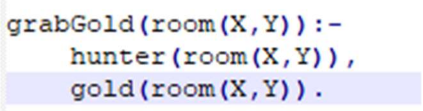
```
safe(room(X,Y)):-
    not(potentialPit(X,Y)),
    not(potentialWumpus(X,Y)).
```

The purpose of the above predicate is to see if a room is safe or not. A room is safe simply if it does not have a potential pit and a potential wumpus.



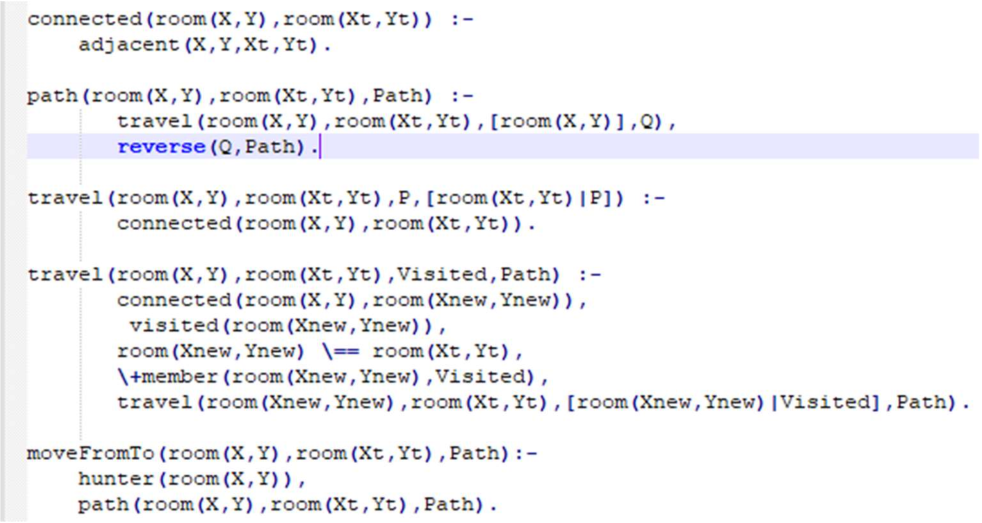
```
gold(room(X,Y)):-
    glitter(room(X,Y)).
```

The above predicate states that there is gold in room with coordinates X and Y if there is glitter in that room.



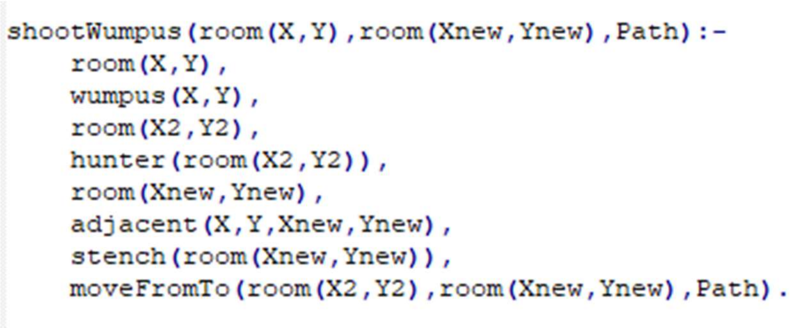
```
grabGold(room(X,Y)):-
    hunter(room(X,Y)),
    gold(room(X,Y)).
```

The agent/hunter grabs gold if it is located in room(X, Y) and there is gold in that room.



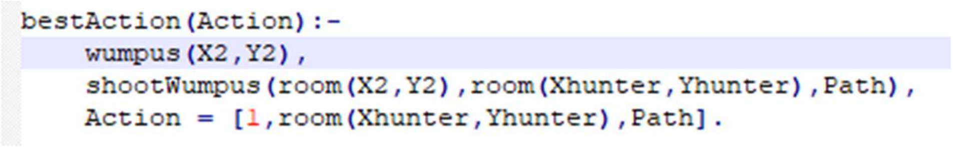
```
connected(room(X,Y),room(Xt,Yt)) :-
    adjacent(X,Y,Xt,Yt).

path(room(X,Y),room(Xt,Yt),Path) :-
        travel(room(X,Y),room(Xt,Yt),[room(X,Y)],Q),
        reverse(Q,Path).

travel(room(X,Y),room(Xt,Yt),P,[room(Xt,Yt)|P]) :-
        connected(room(X,Y),room(Xt,Yt)).

travel(room(X,Y),room(Xt,Yt),Visited,Path) :-
        connected(room(X,Y),room(Xnew,Ynew)),
         visited(room(Xnew,Ynew)),
        room(Xnew,Ynew) \== room(Xt,Yt),
        \+member(room(Xnew,Ynew),Visited),
        travel(room(Xnew,Ynew),room(Xt,Yt),[room(Xnew,Ynew)|Visited],Path).

moveFromTo(room(X,Y),room(Xt,Yt),Path):-
    hunter(room(X,Y)),
    path(room(X,Y),room(Xt,Yt),Path).
```

The set of the above predicates allows the agent to move from a room (X, Y) to a target room (Xt, Yt) and stores the path (in Path). Q is the reverse path: we reverse it and put it back to path.



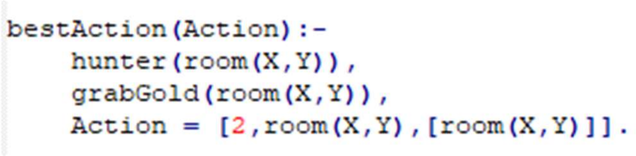
```
shootWumpus(room(X,Y),room(Xnew,Ynew),Path):-
    room(X,Y),
    wumpus(X,Y),
    room(X2,Y2),
    hunter(room(X2,Y2)),
    room(Xnew,Ynew),
    adjacent(X,Y,Xnew,Ynew),
    stench(room(Xnew,Ynew)),
    moveFromTo(room(X2,Y2),room(Xnew,Ynew),Path).
```

We shoot a Wumpus when there is a Wumpus in room with coordinates X and Y, hunter/agent is in room (X2, Y2) and room with coordinates Xnew and Ynew is adjacent to room(X2, Y2) and has a stench. The hunter should move from room(X, Y) to room(Xnew, Ynew) using the Path.



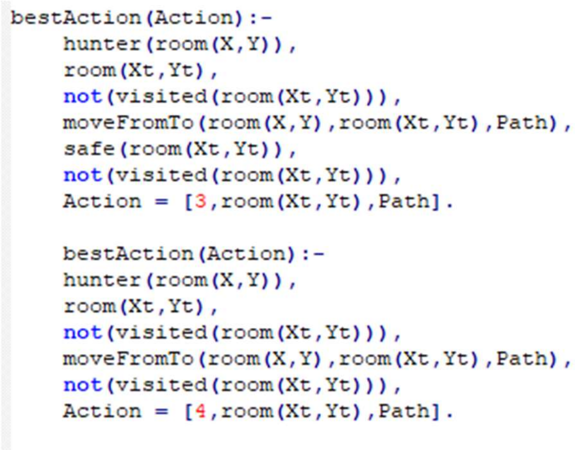
```
bestAction(Action):-
    wumpus(X2,Y2),
    shootWumpus(room(X2,Y2),room(Xhunter,Yhunter),Path),
    Action = [1,room(Xhunter,Yhunter),Path].
```

The first bestAction predicate concerns shooting a Wumpus. We do so if there is a Wumpus in room(X2, Y2). We used integers to refer to the action types. This action type has number 1.



```
bestAction(Action):-
    hunter(room(X,Y)),
    grabGold(room(X,Y)),
    Action = [2,room(X,Y),[room(X,Y)]].
```

The second bestAction is to grab gold in room (X, Y). Note that the predicate grabGold checks if there is gold in room(X, Y).

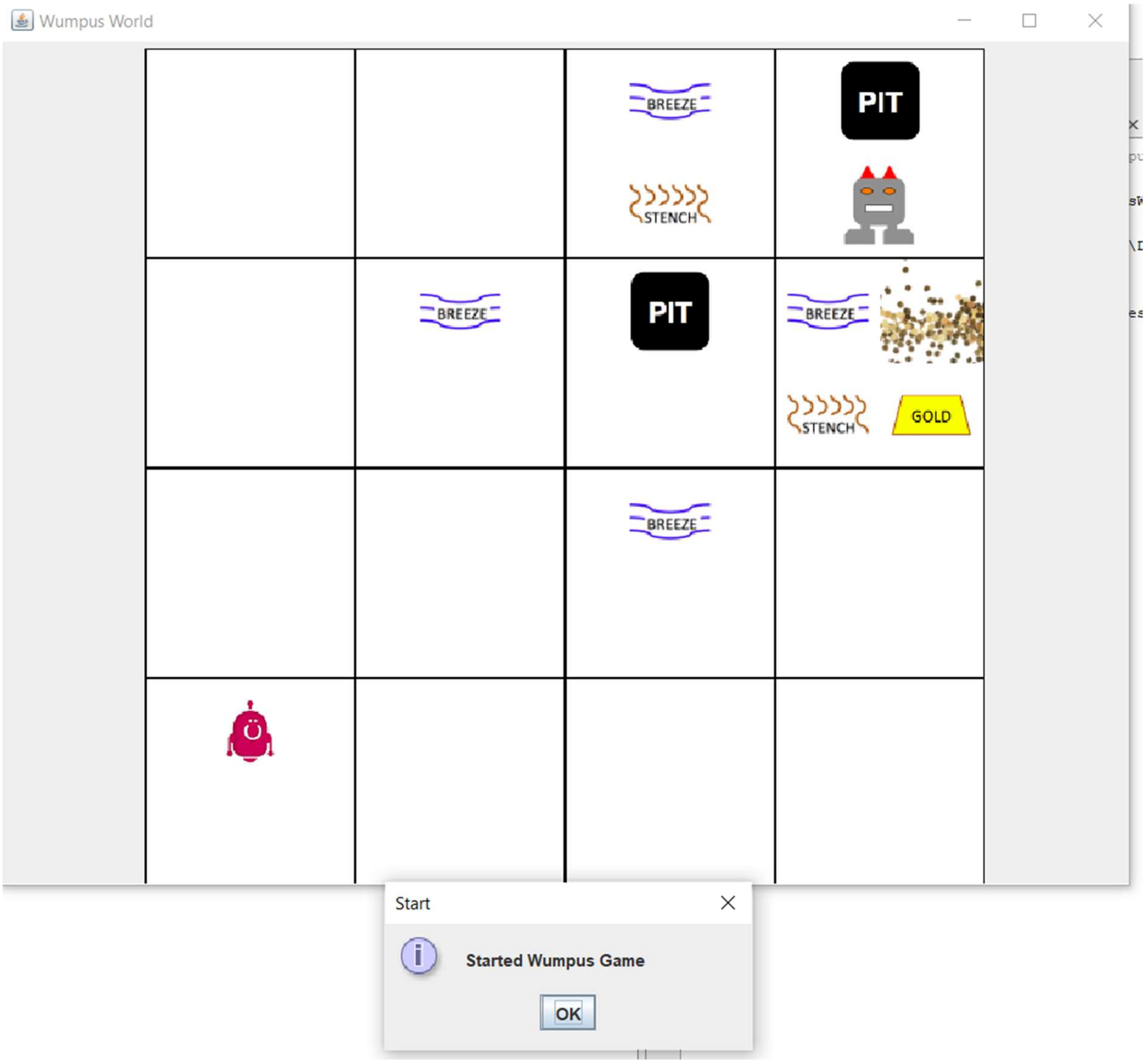


```
bestAction(Action):-
    hunter(room(X,Y)),
    room(Xt,Yt),
    not(visited(room(Xt,Yt))),
    moveFromTo(room(X,Y),room(Xt,Yt),Path),
    safe(room(Xt,Yt)),
    not(visited(room(Xt,Yt))),
    Action = [3,room(Xt,Yt),Path].

    bestAction(Action):-
    hunter(room(X,Y)),
    room(Xt,Yt),
    not(visited(room(Xt,Yt))),
    moveFromTo(room(X,Y),room(Xt,Yt),Path),
    not(visited(room(Xt,Yt))),
    Action = [4,room(Xt,Yt),Path].
```

The above best actions are used to move from a room to another (if the room is not visited yet). The first one out of these two checks if there the room is safe and moves to it that is why it is given the higher priority while the second one directly moves from room(X, Y) to room (Xt, Yt) without checking if it is safe.

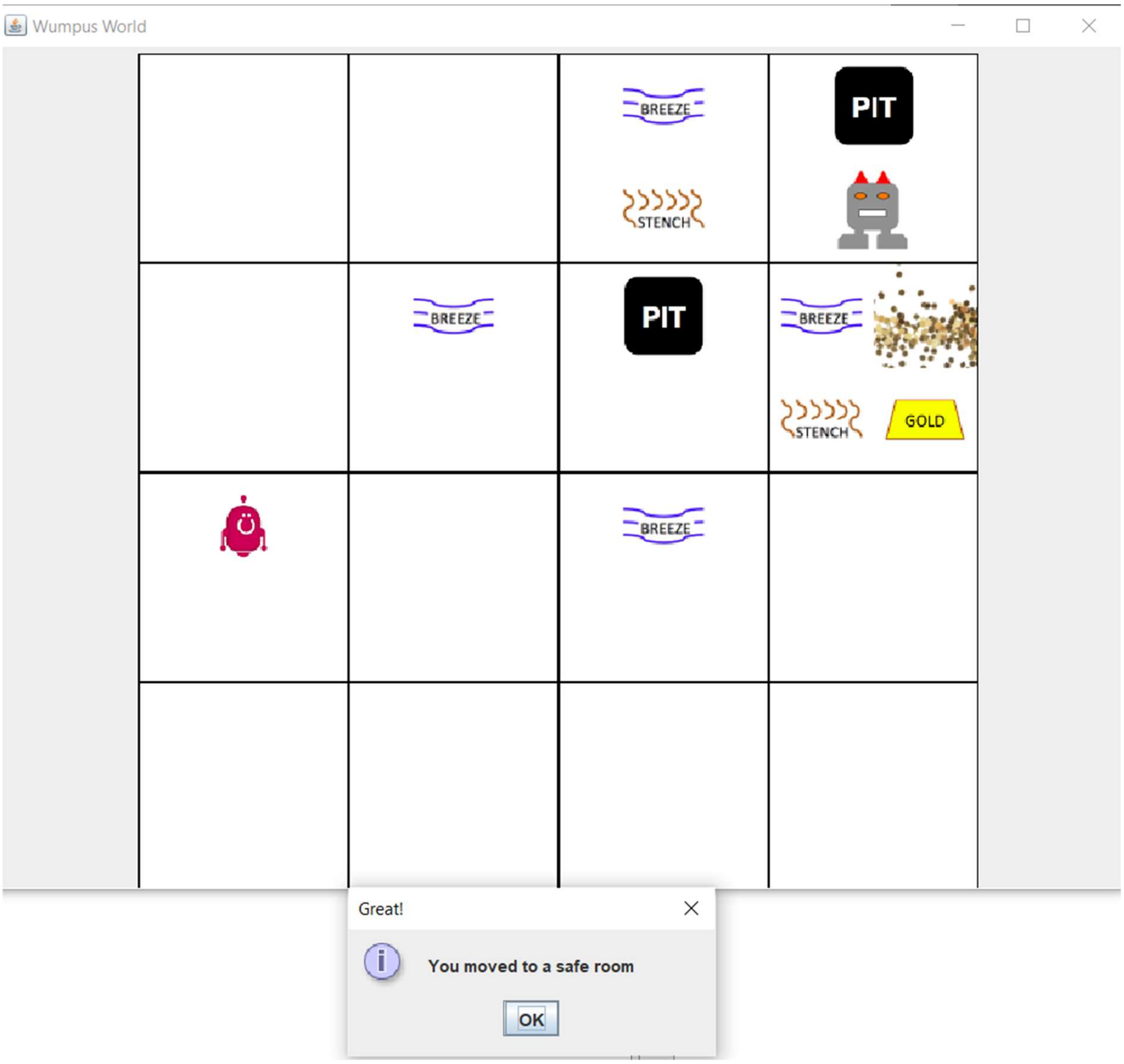
Snapshots of the Experiment with Various Starting Configurations

Experiment 1



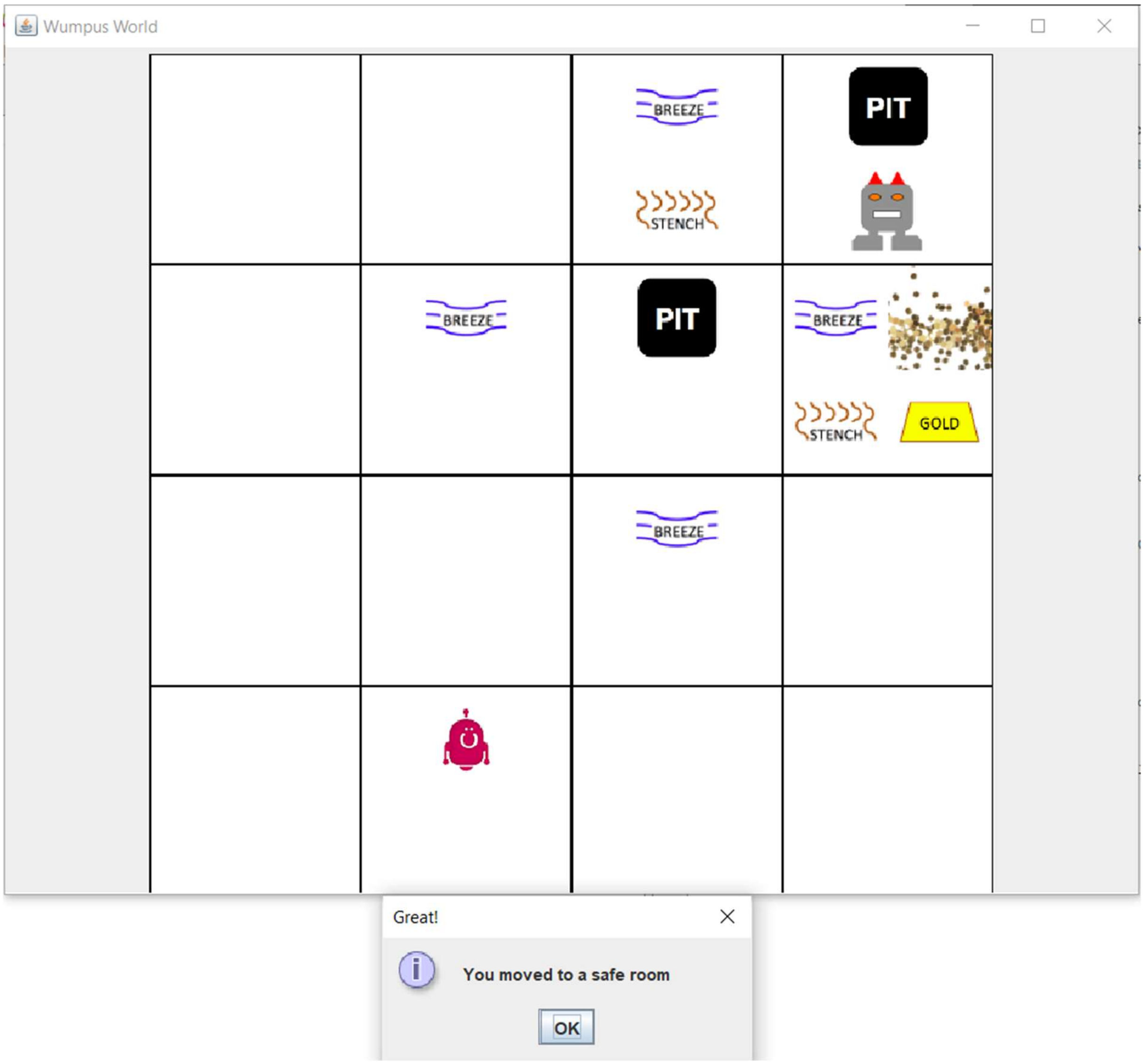
		BREEZE STENCH	PIT
	BREEZE	PIT	BREEZE STENCH GOLD
		BREEZE	

Started Wumpus Game

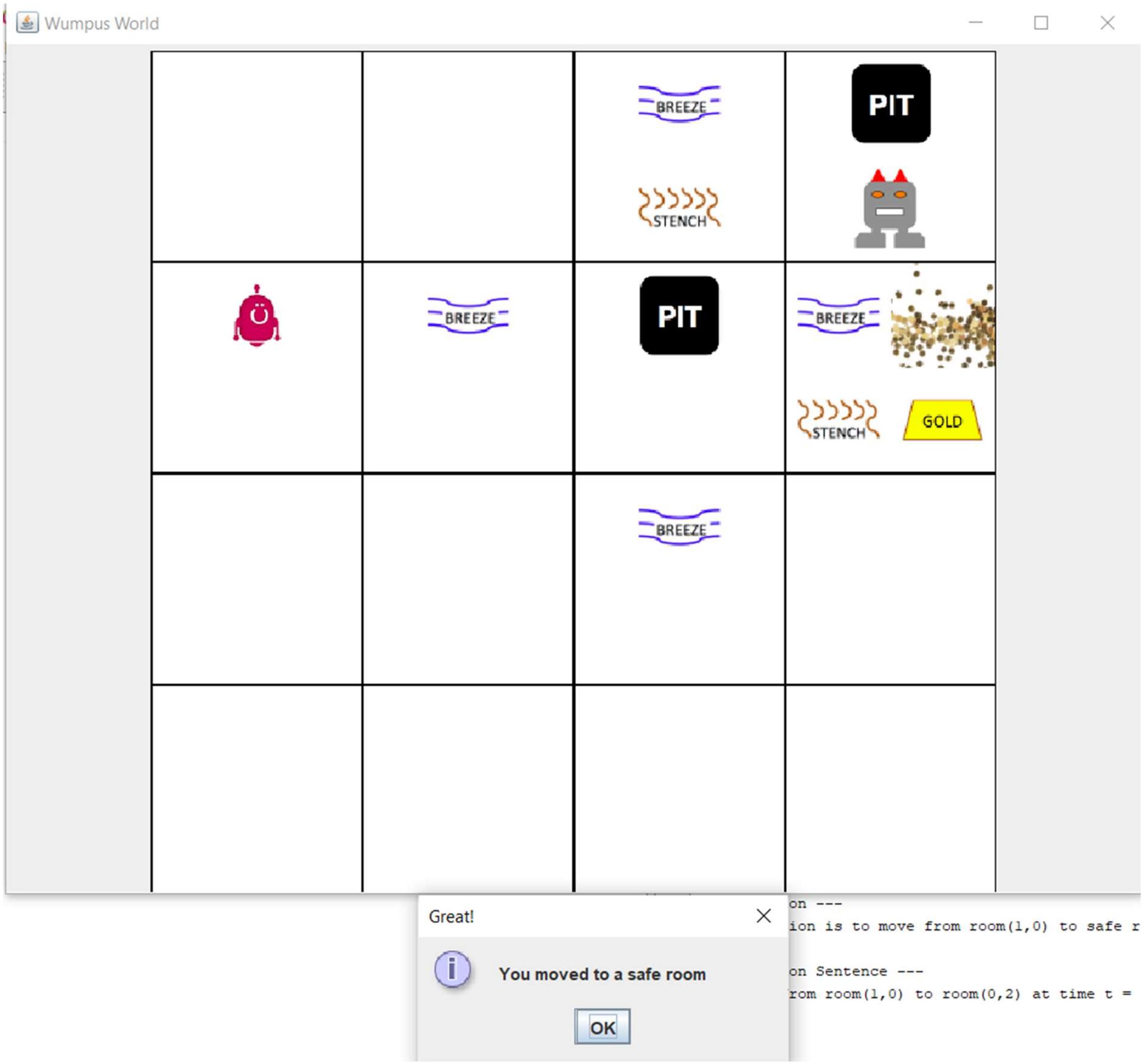


Great!

You moved to a safe room



You moved to a safe room

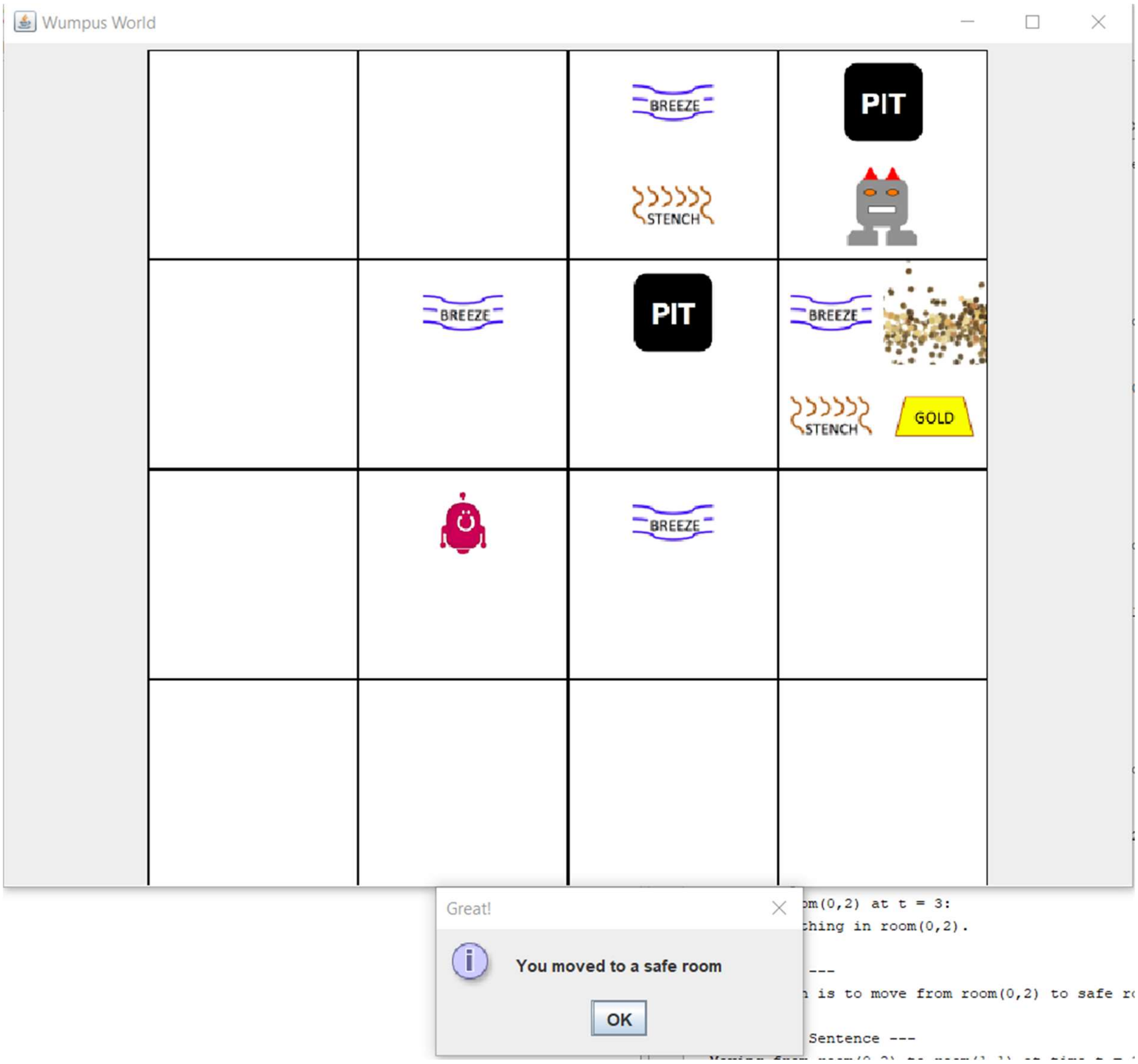


Great!

You moved to a safe room

OK

```
on ---
ion is to move from room(1,0) to safe r

on Sentence ---
rom room(1,0) to room(0,2) at time t =
```

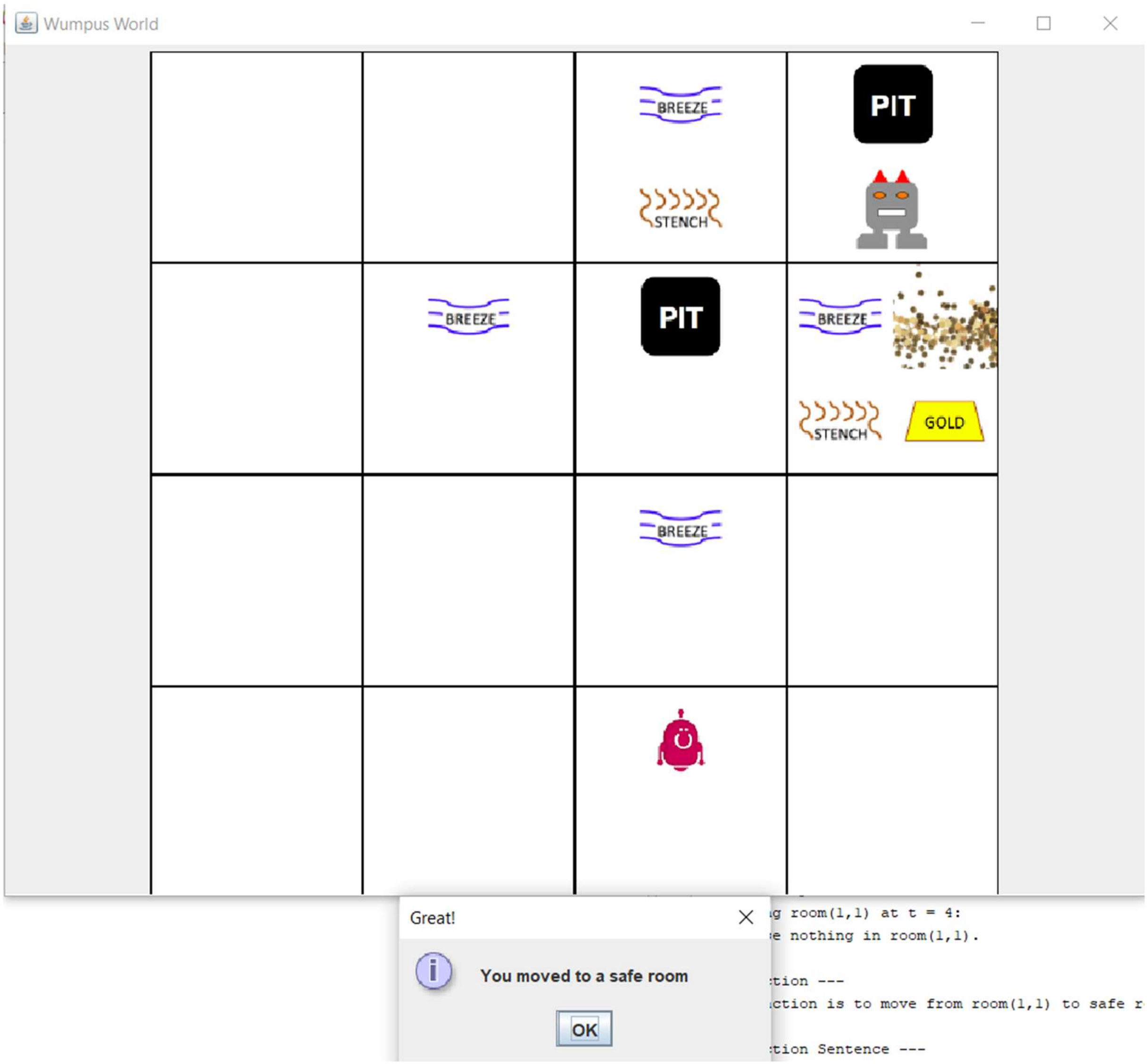


Great!

You moved to a safe room

OK

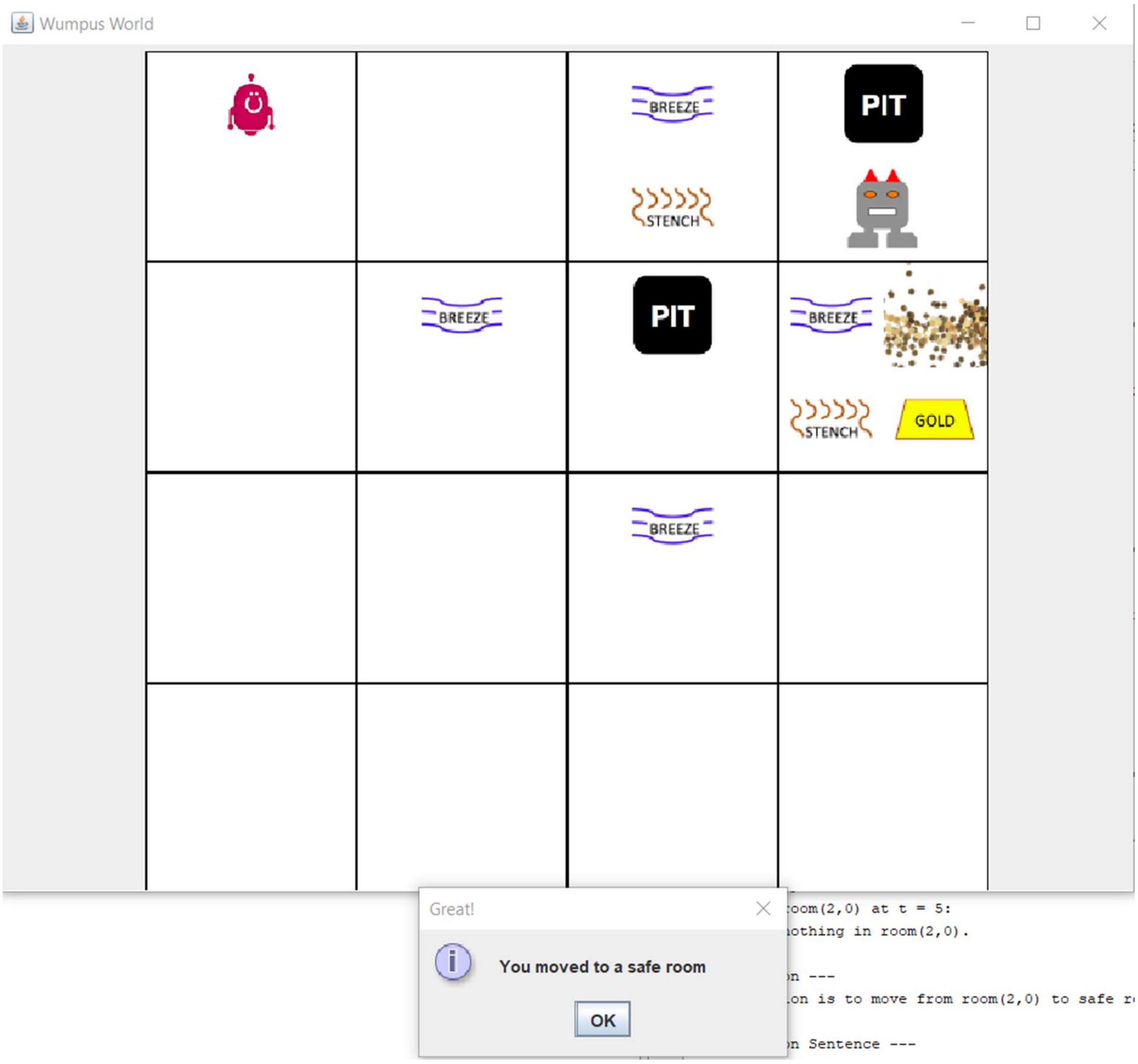
```
om(0,2) at t = 3:
thing in room(0,2).

 ---
n is to move from room(0,2) to safe r

 Sentence ---
Moving from room(0,2) to room(1,1) at time t =
```



Great!

You moved to a safe room

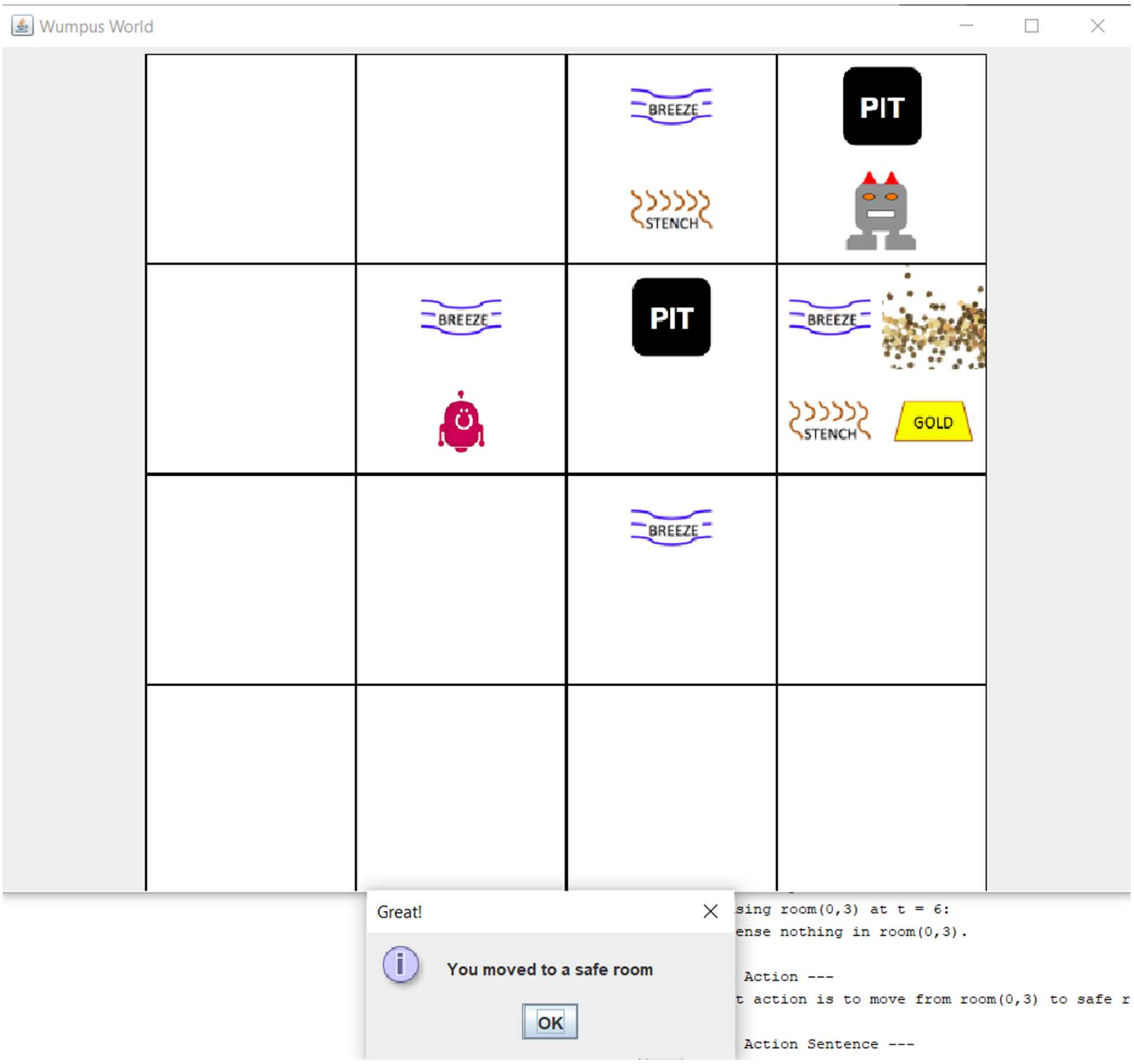
```
g room(1,1) at t = 4:
e nothing in room(1,1).

tion ---
ction is to move from room(1,1) to safe r

tion Sentence ---
```



Great!

You moved to a safe room

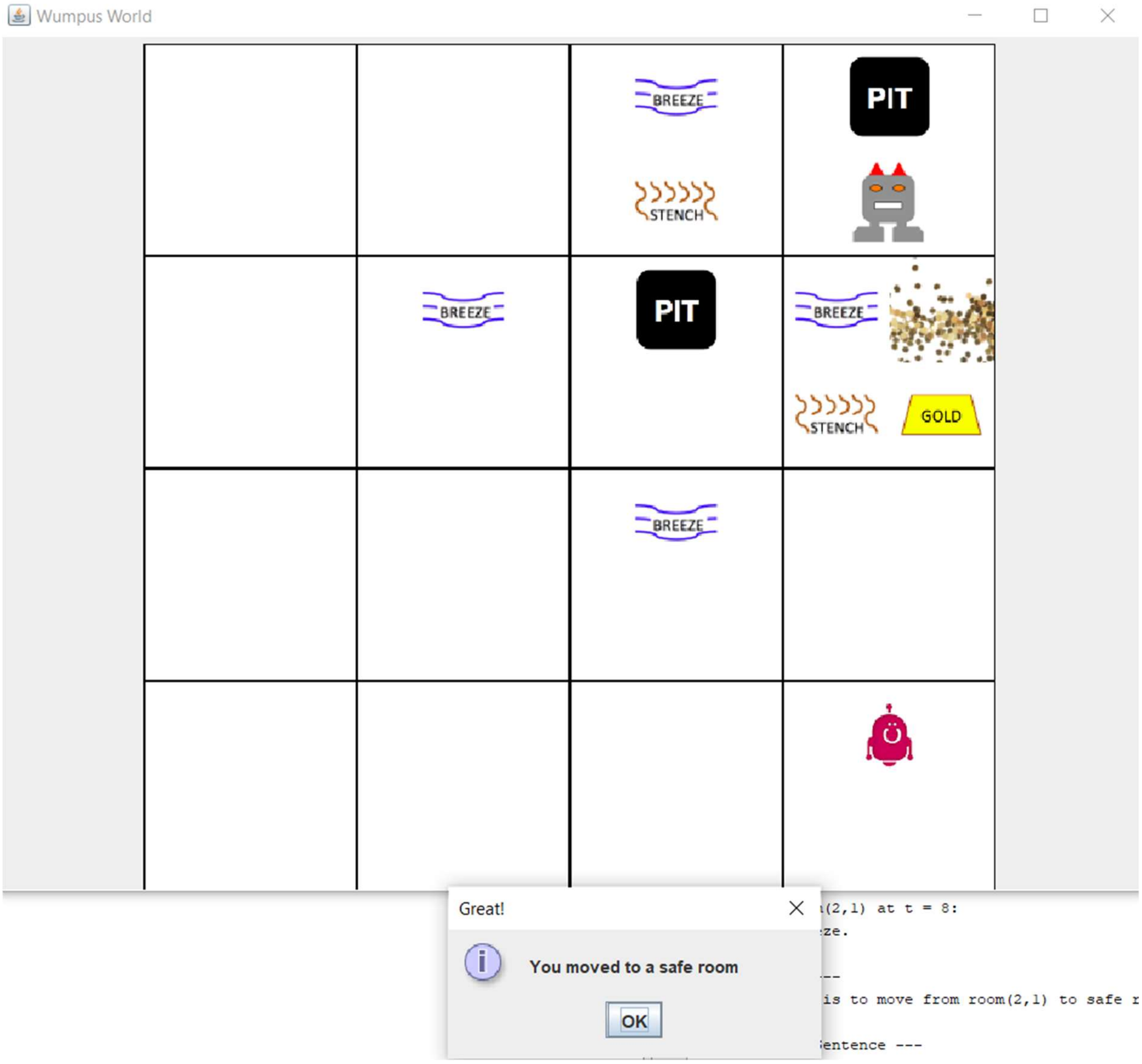
```
room(2,0) at t = 5:
nothing in room(2,0).

on ---
ion is to move from room(2,0) to safe r

on Sentence ---
```



Great!

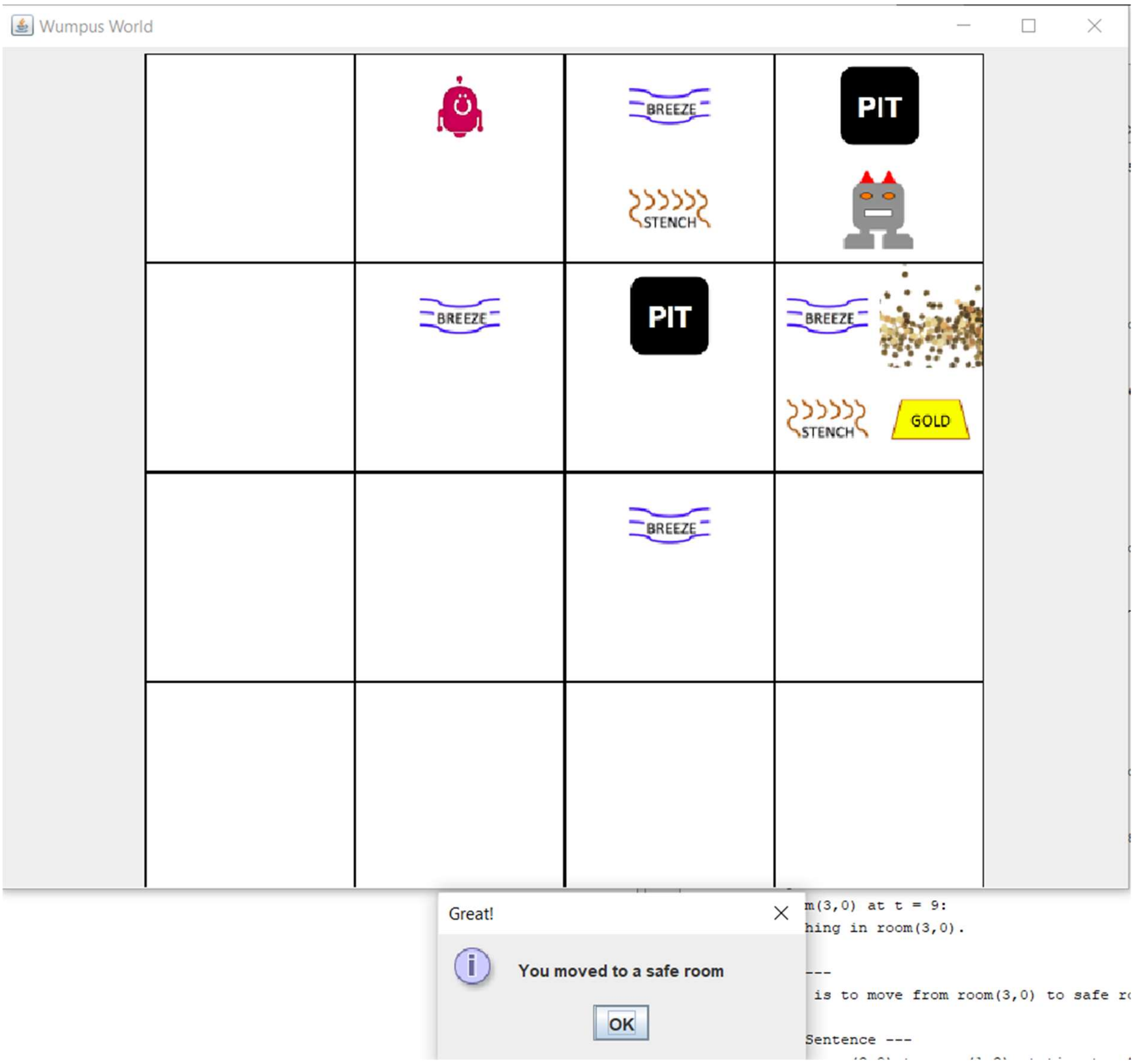
You moved to a safe room

```
sing room(0,3) at t = 6:
ense nothing in room(0,3).

 Action ---
t action is to move from room(0,3) to safe r

 Action Sentence ---
```



You moved to a safe room

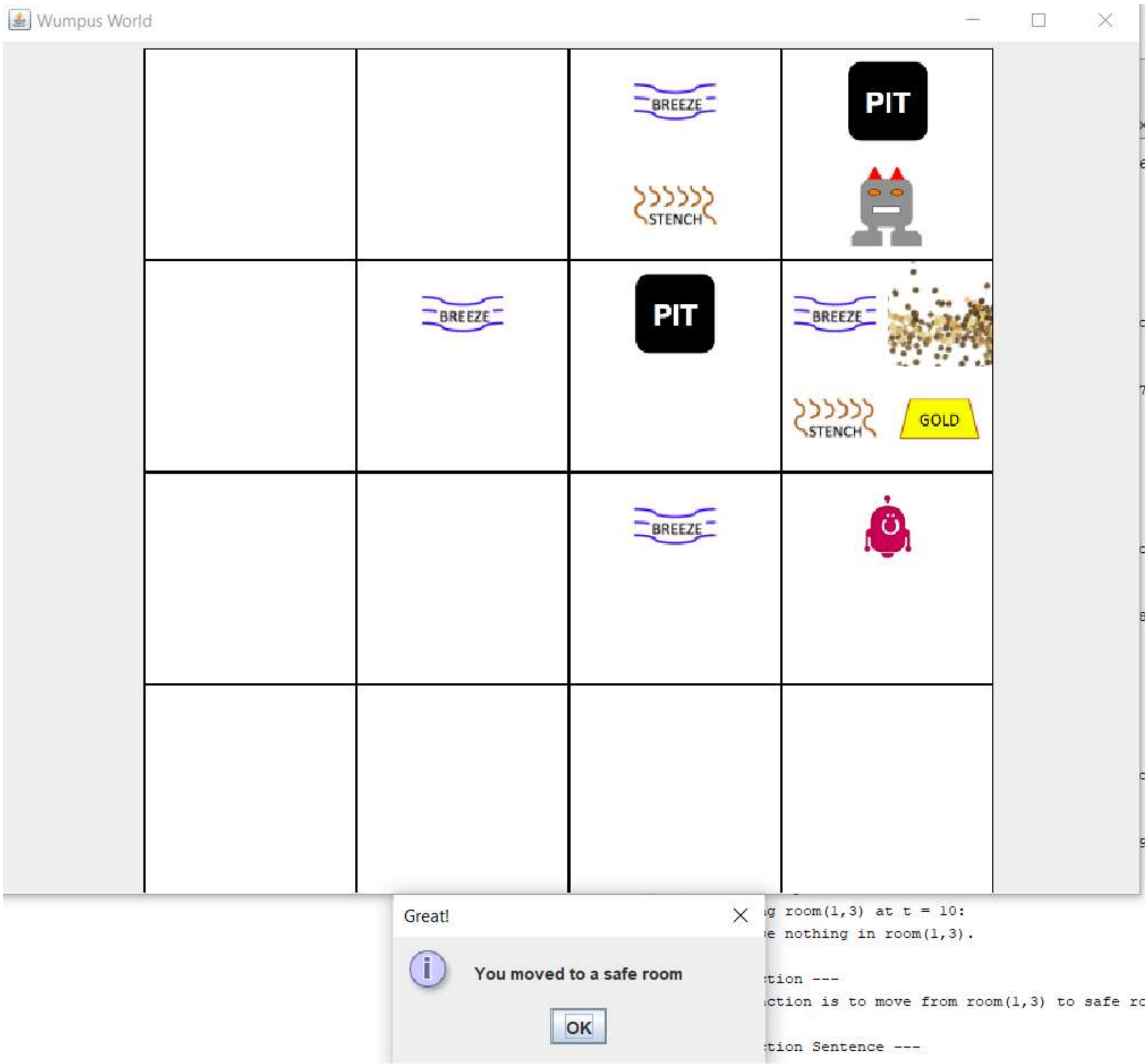
```
(2,1) at t = 8:
ze.

--
is to move from room(2,1) to safe r

entence ---
```



Great!

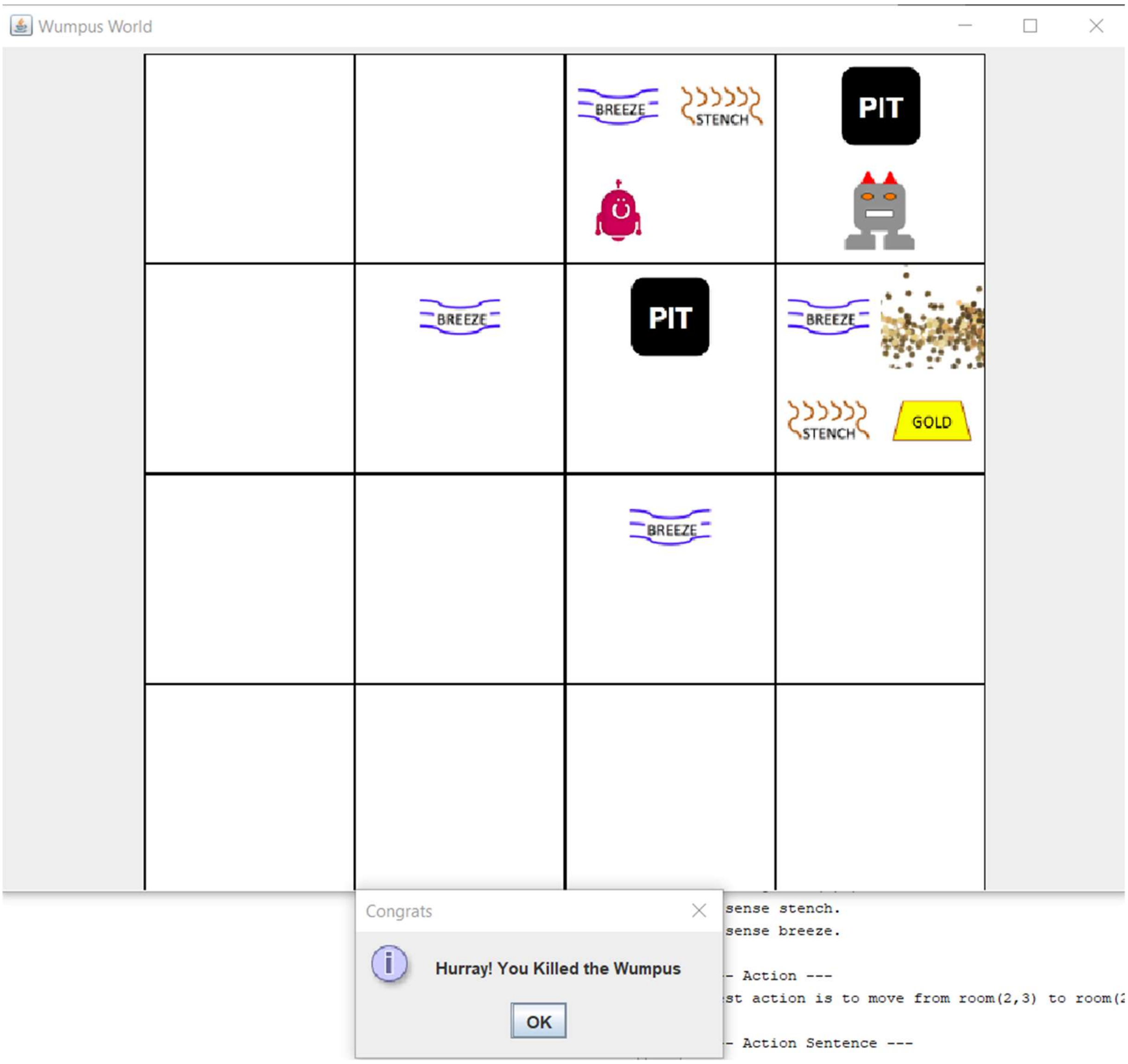
You moved to a safe room

```
m(3,0) at t = 9:
hing in room(3,0).

---
 is to move from room(3,0) to safe r

Sentence ---
```



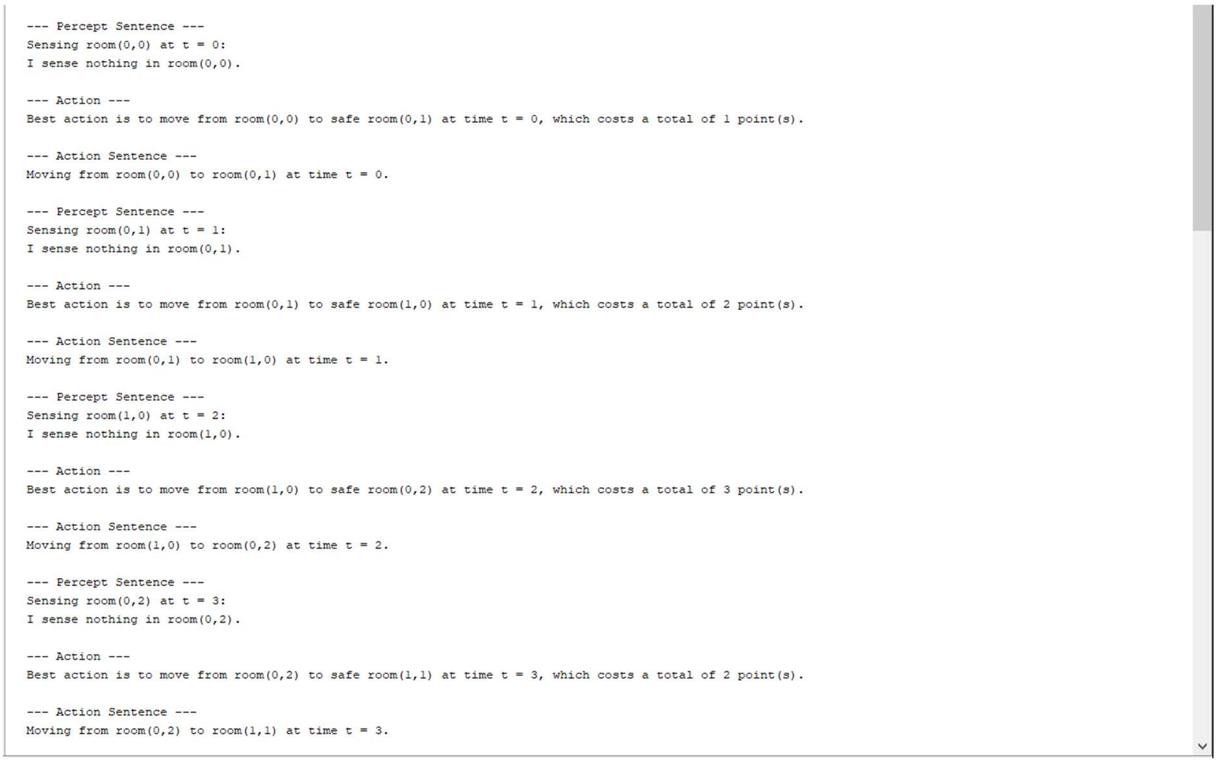
You moved to a safe room

```
g room(1,3) at t = 10:
e nothing in room(1,3).

tion ---
ction is to move from room(1,3) to safe rc

tion Sentence ---
```

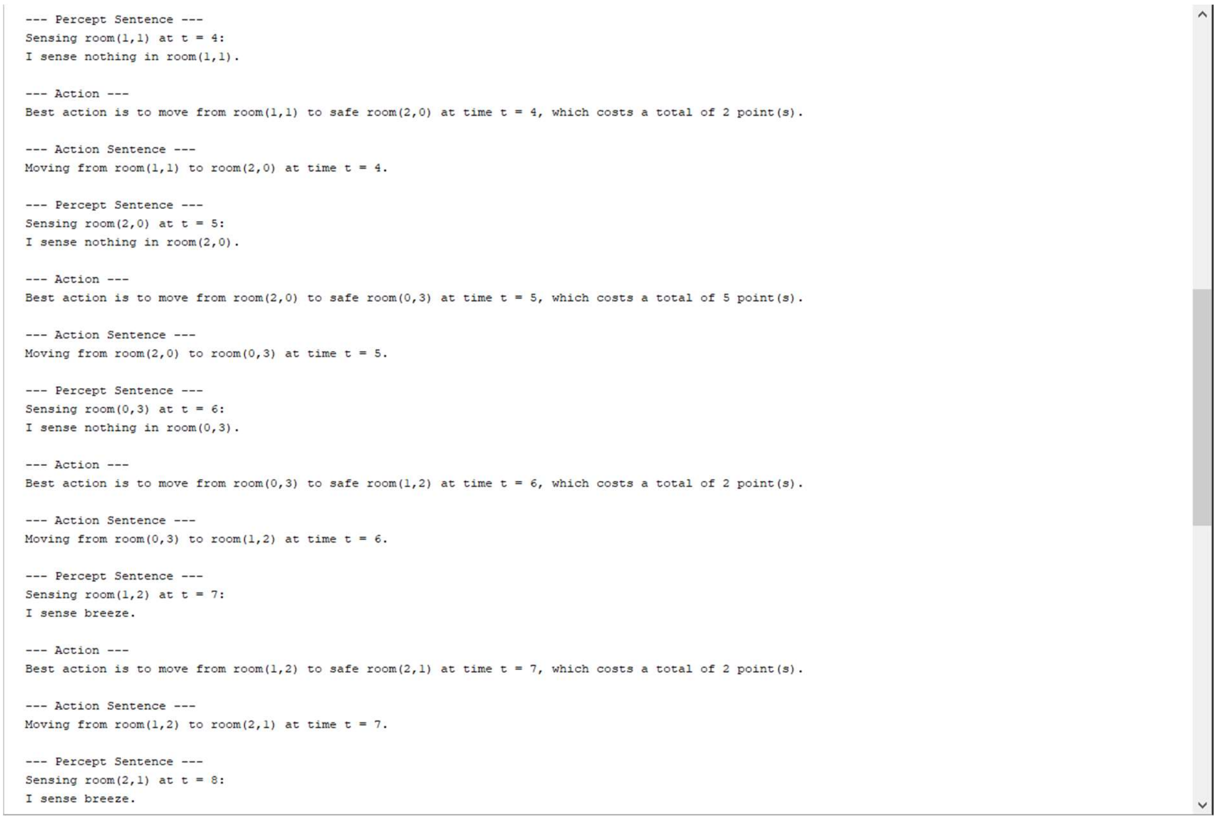


Hurray! You Killed the Wumpus

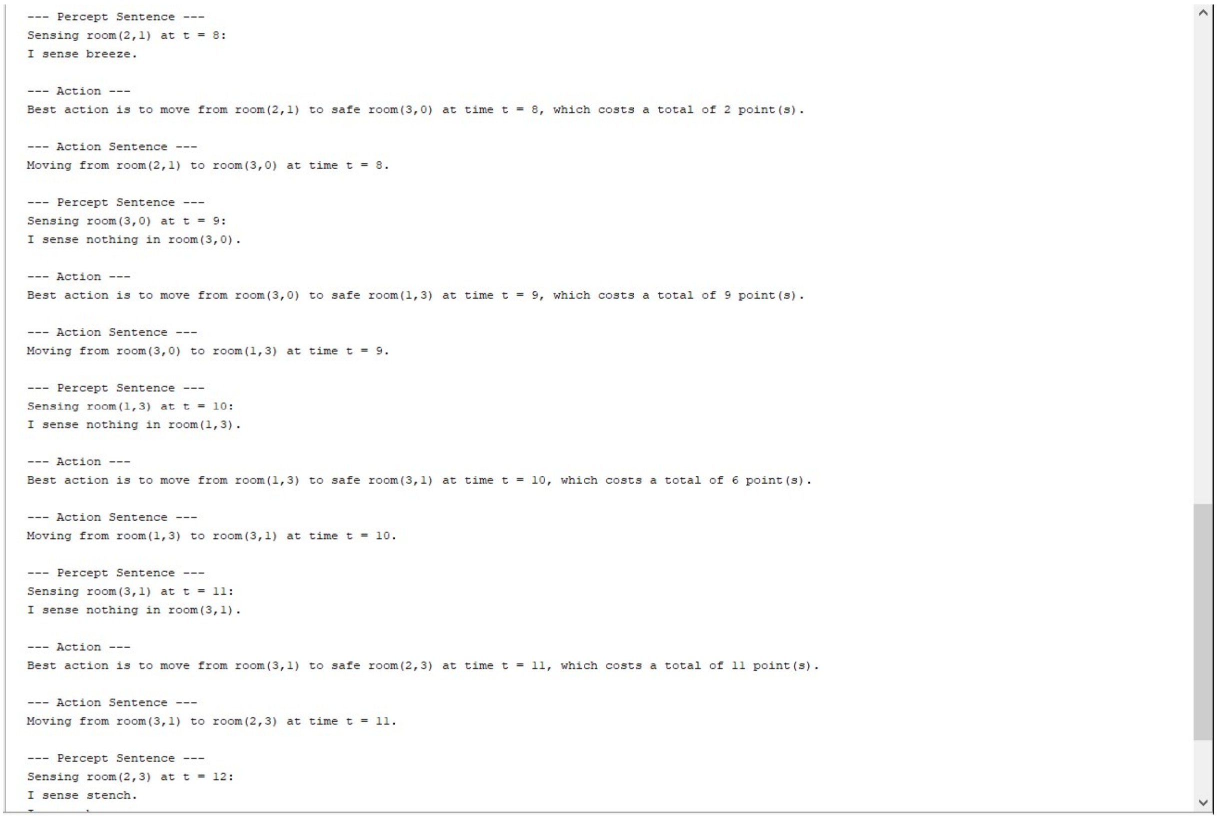
```
sense stench.
sense breeze.

- Action ---
st action is to move from room(2,3) to room(2

- Action Sentence ---
```



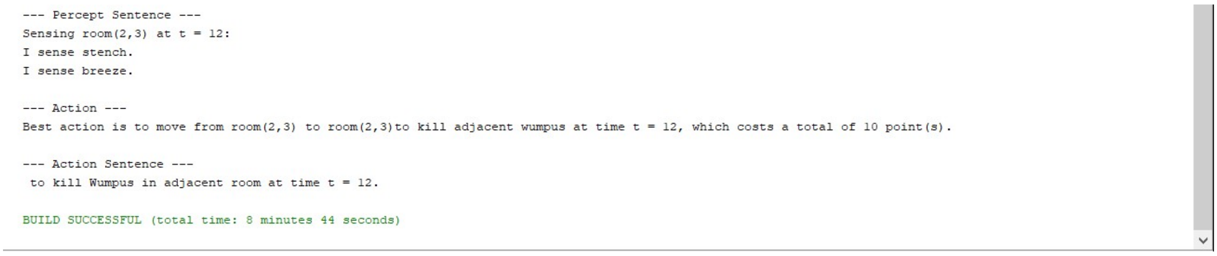
```
--- Percept Sentence ---
Sensing room(0,0) at t = 0:
I sense nothing in room(0,0).

--- Action ---
Best action is to move from room(0,0) to safe room(0,1) at time t = 0, which costs a total of 1 point(s).

--- Action Sentence ---
Moving from room(0,0) to room(0,1) at time t = 0.

--- Percept Sentence ---
Sensing room(0,1) at t = 1:
I sense nothing in room(0,1).

--- Action ---
Best action is to move from room(0,1) to safe room(1,0) at time t = 1, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(0,1) to room(1,0) at time t = 1.

--- Percept Sentence ---
Sensing room(1,0) at t = 2:
I sense nothing in room(1,0).

--- Action ---
Best action is to move from room(1,0) to safe room(0,2) at time t = 2, which costs a total of 3 point(s).

--- Action Sentence ---
Moving from room(1,0) to room(0,2) at time t = 2.

--- Percept Sentence ---
Sensing room(0,2) at t = 3:
I sense nothing in room(0,2).

--- Action ---
Best action is to move from room(0,2) to safe room(1,1) at time t = 3, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(0,2) to room(1,1) at time t = 3.
```



```
--- Percept Sentence ---
Sensing room(1,1) at t = 4:
I sense nothing in room(1,1).

--- Action ---
Best action is to move from room(1,1) to safe room(2,0) at time t = 4, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(1,1) to room(2,0) at time t = 4.

--- Percept Sentence ---
Sensing room(2,0) at t = 5:
I sense nothing in room(2,0).

--- Action ---
Best action is to move from room(2,0) to safe room(0,3) at time t = 5, which costs a total of 5 point(s).

--- Action Sentence ---
Moving from room(2,0) to room(0,3) at time t = 5.

--- Percept Sentence ---
Sensing room(0,3) at t = 6:
I sense nothing in room(0,3).

--- Action ---
Best action is to move from room(0,3) to safe room(1,2) at time t = 6, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(0,3) to room(1,2) at time t = 6.

--- Percept Sentence ---
Sensing room(1,2) at t = 7:
I sense breeze.

--- Action ---
Best action is to move from room(1,2) to safe room(2,1) at time t = 7, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(1,2) to room(2,1) at time t = 7.

--- Percept Sentence ---
Sensing room(2,1) at t = 8:
I sense breeze.
```

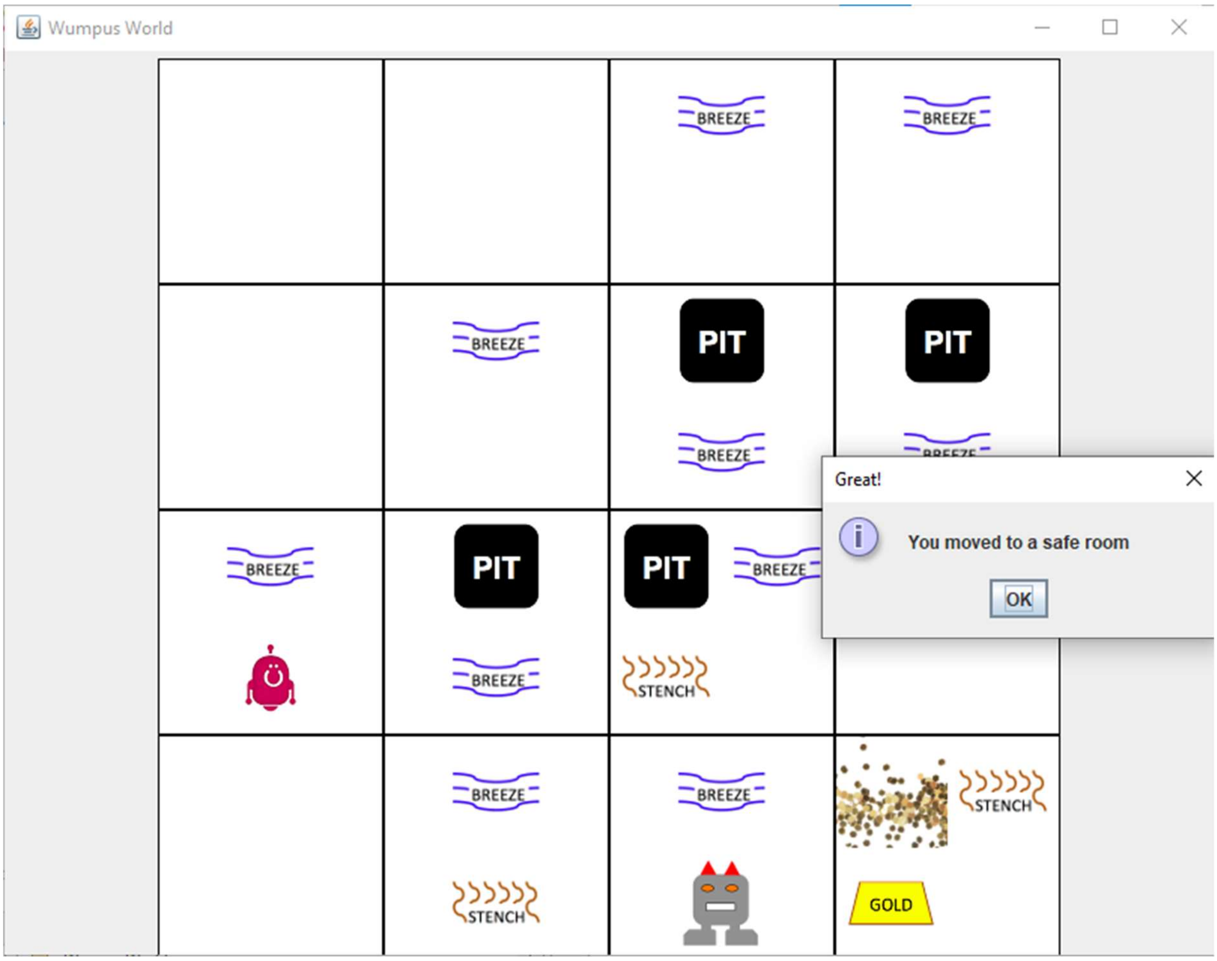


```
--- Percept Sentence ---
Sensing room(2,1) at t = 8:
I sense breeze.

--- Action ---
Best action is to move from room(2,1) to safe room(3,0) at time t = 8, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(2,1) to room(3,0) at time t = 8.

--- Percept Sentence ---
Sensing room(3,0) at t = 9:
I sense nothing in room(3,0).

--- Action ---
Best action is to move from room(3,0) to safe room(1,3) at time t = 9, which costs a total of 9 point(s).

--- Action Sentence ---
Moving from room(3,0) to room(1,3) at time t = 9.

--- Percept Sentence ---
Sensing room(1,3) at t = 10:
I sense nothing in room(1,3).

--- Action ---
Best action is to move from room(1,3) to safe room(3,1) at time t = 10, which costs a total of 6 point(s).

--- Action Sentence ---
Moving from room(1,3) to room(3,1) at time t = 10.

--- Percept Sentence ---
Sensing room(3,1) at t = 11:
I sense nothing in room(3,1).

--- Action ---
Best action is to move from room(3,1) to safe room(2,3) at time t = 11, which costs a total of 11 point(s).

--- Action Sentence ---
Moving from room(3,1) to room(2,3) at time t = 11.

--- Percept Sentence ---
Sensing room(2,3) at t = 12:
I sense stench.
```

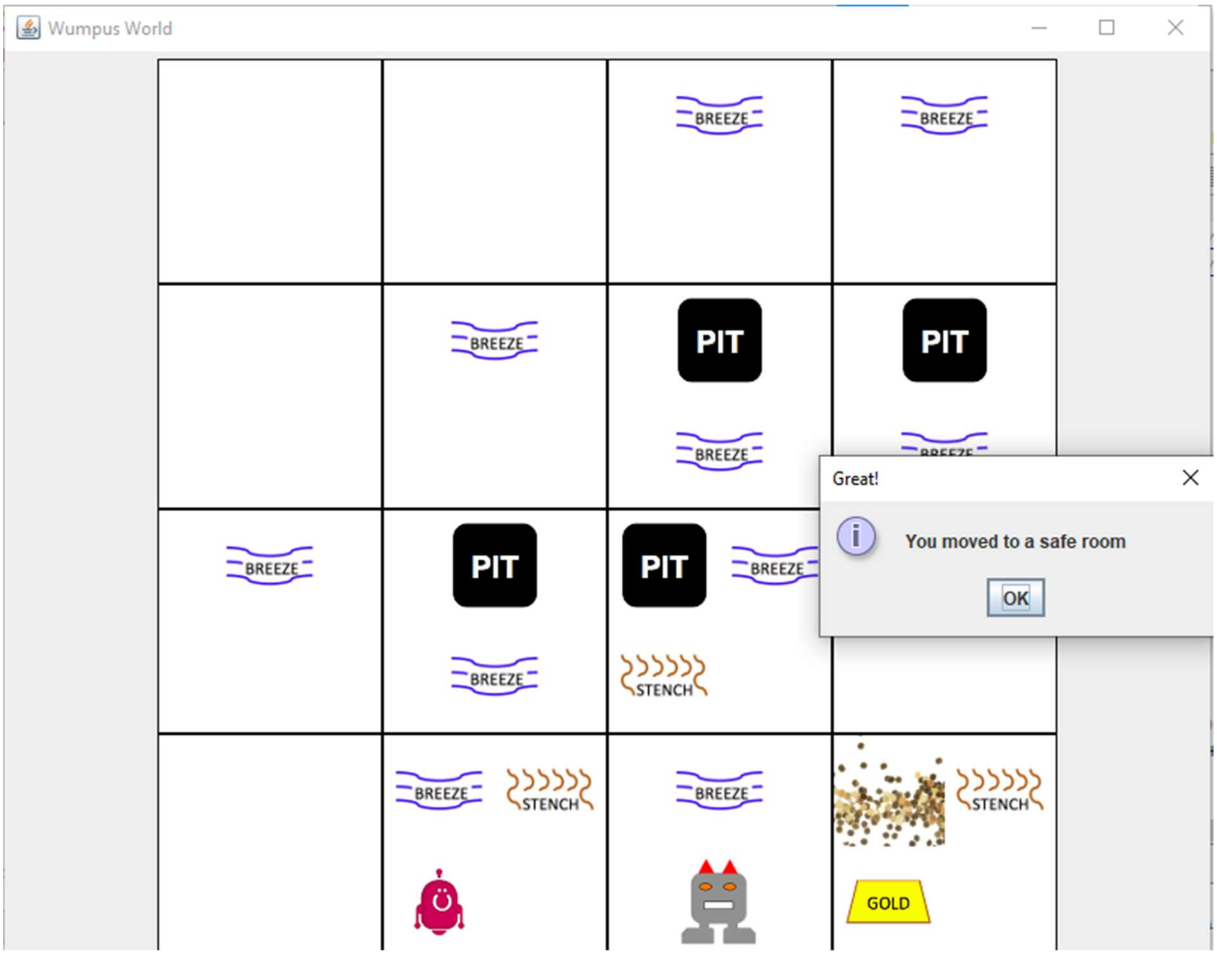


```
--- Percept Sentence ---
Sensing room(2,3) at t = 12:
I sense stench.
I sense breeze.

--- Action ---
Best action is to move from room(2,3) to room(2,3)to kill adjacent wumpus at time t = 12, which costs a total of 10 point(s).

--- Action Sentence ---
 to kill Wumpus in adjacent room at time t = 12.

BUILD SUCCESSFUL (total time: 8 minutes 44 seconds)
```

Experiment 2



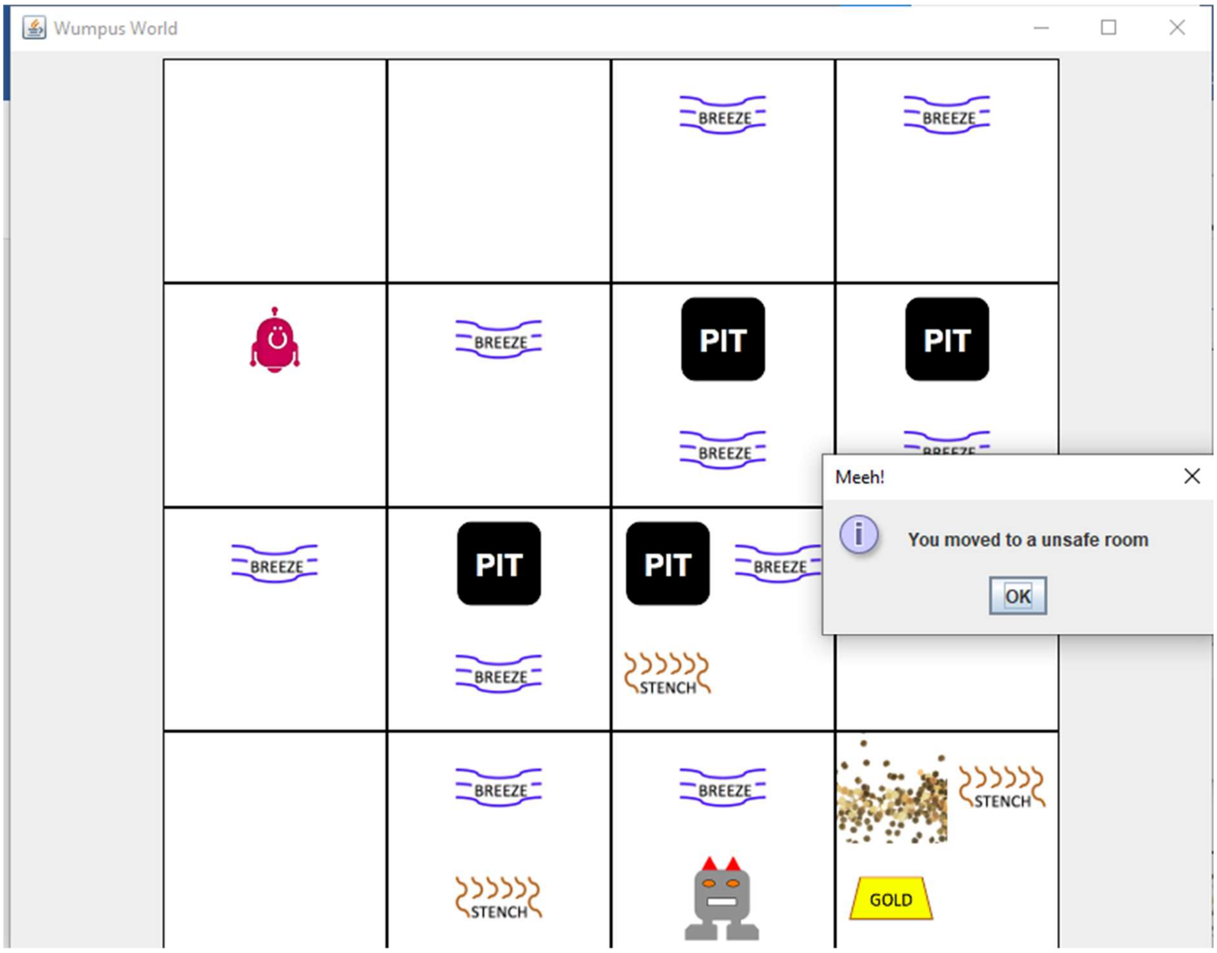
You moved to a safe room



Great!

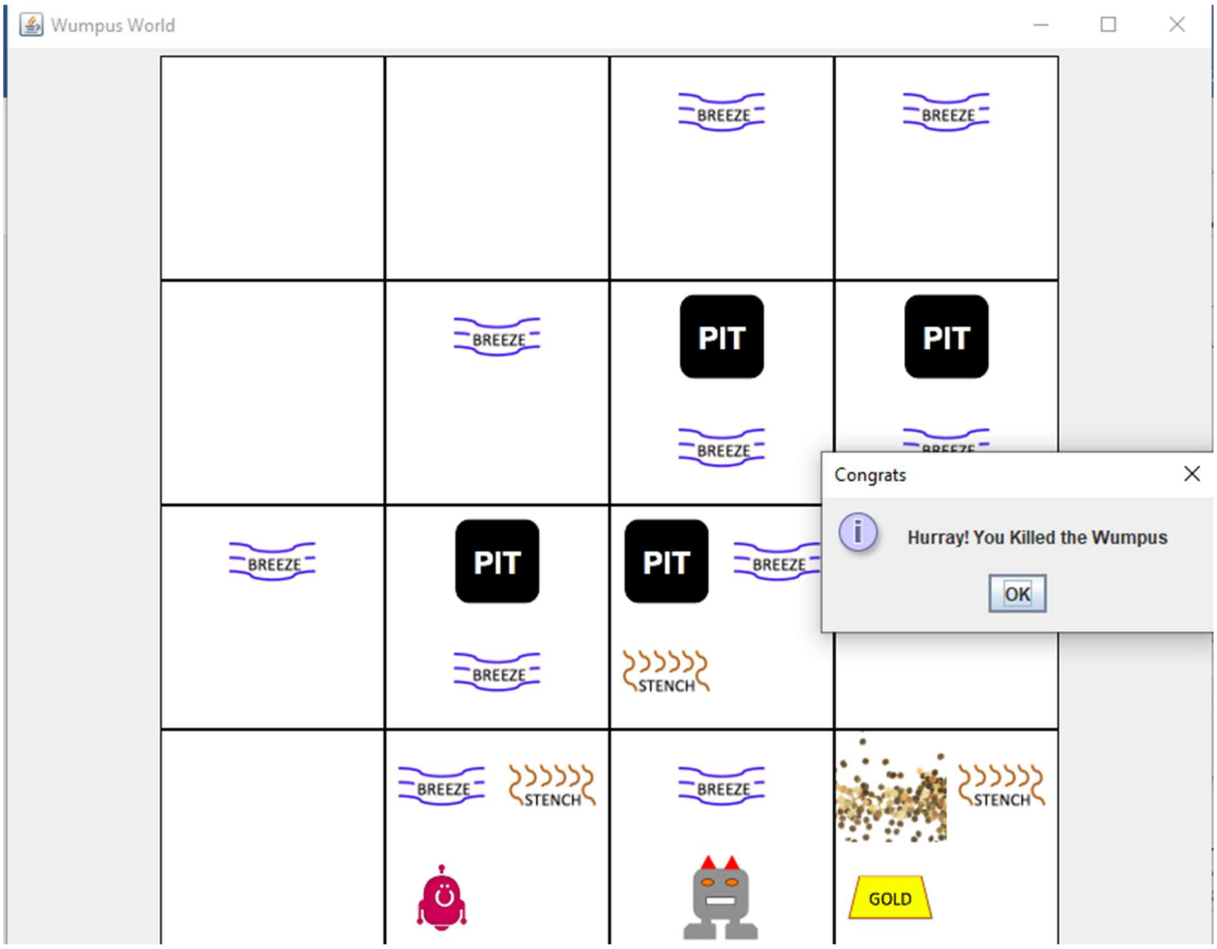
You moved to a safe room

OK



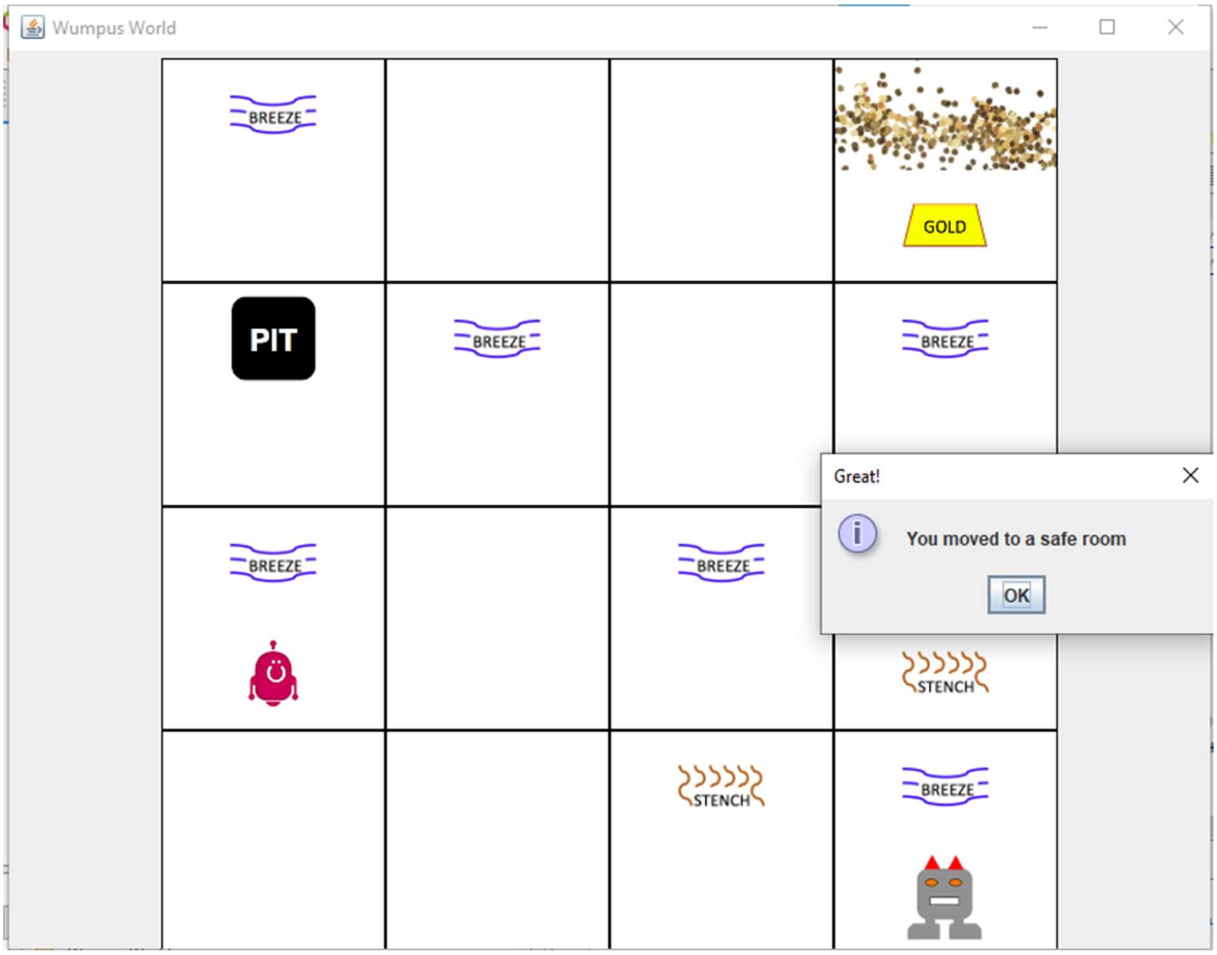
Meeh!

You moved to a unsafe room

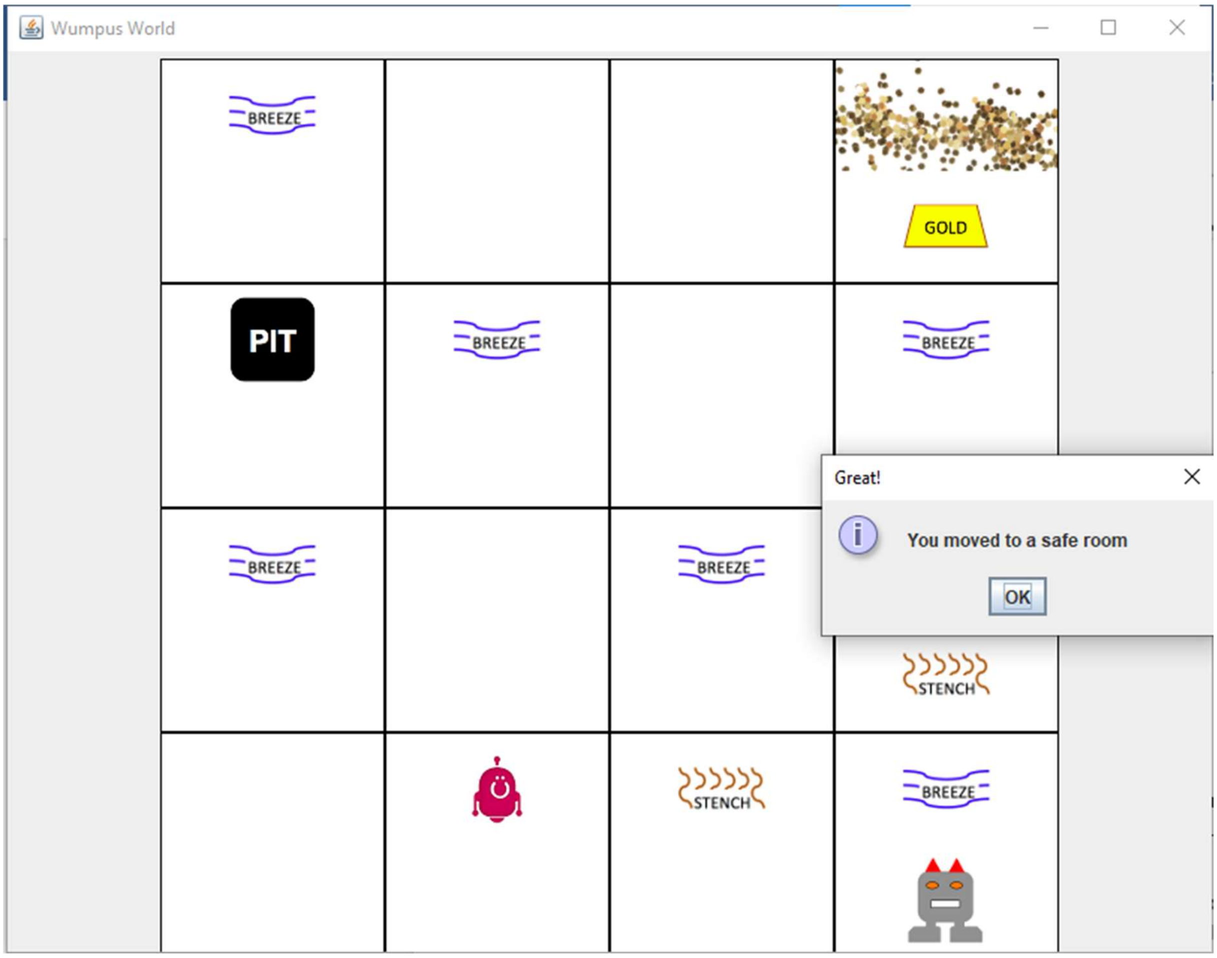


Hurray! You Killed the Wumpus

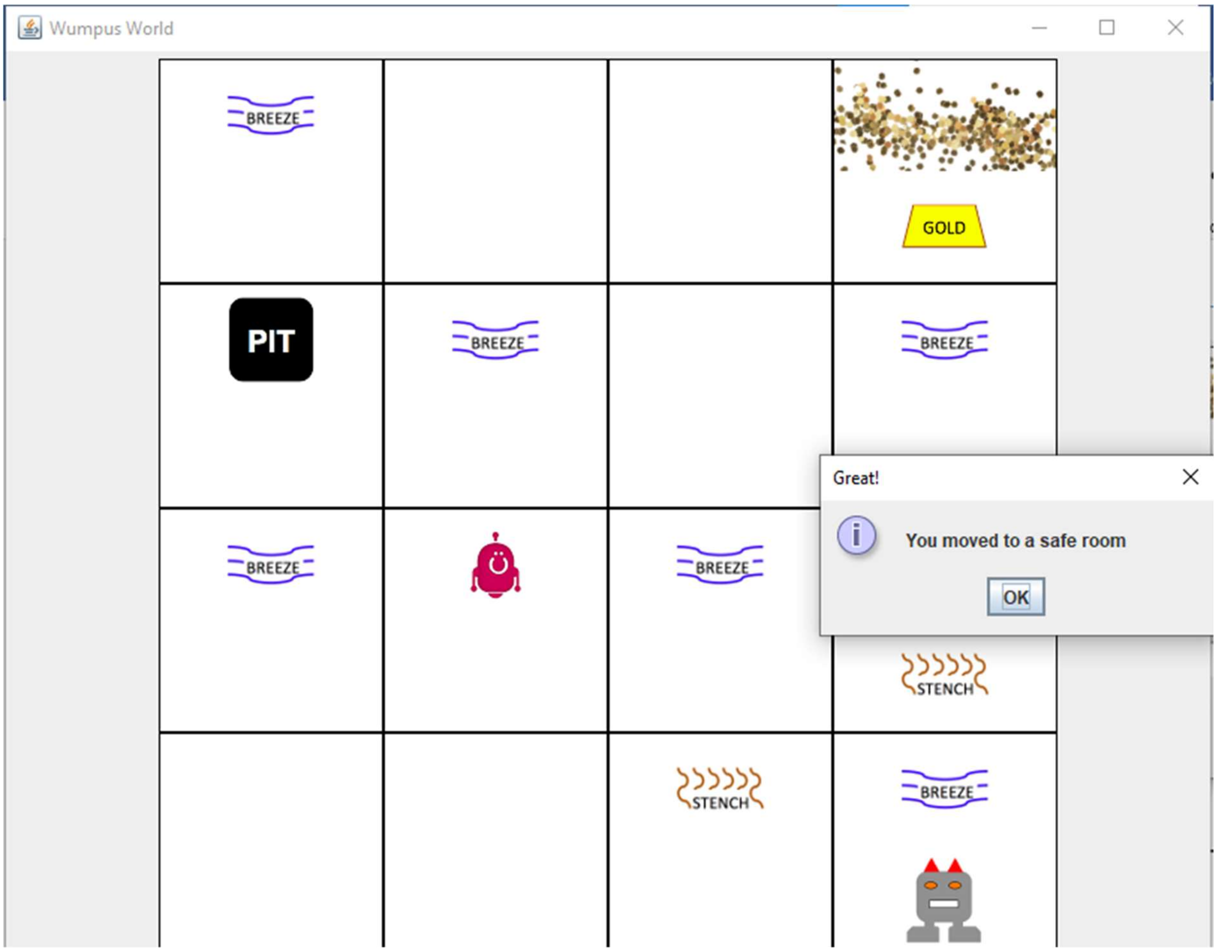
Experiment 3



You moved to a safe room

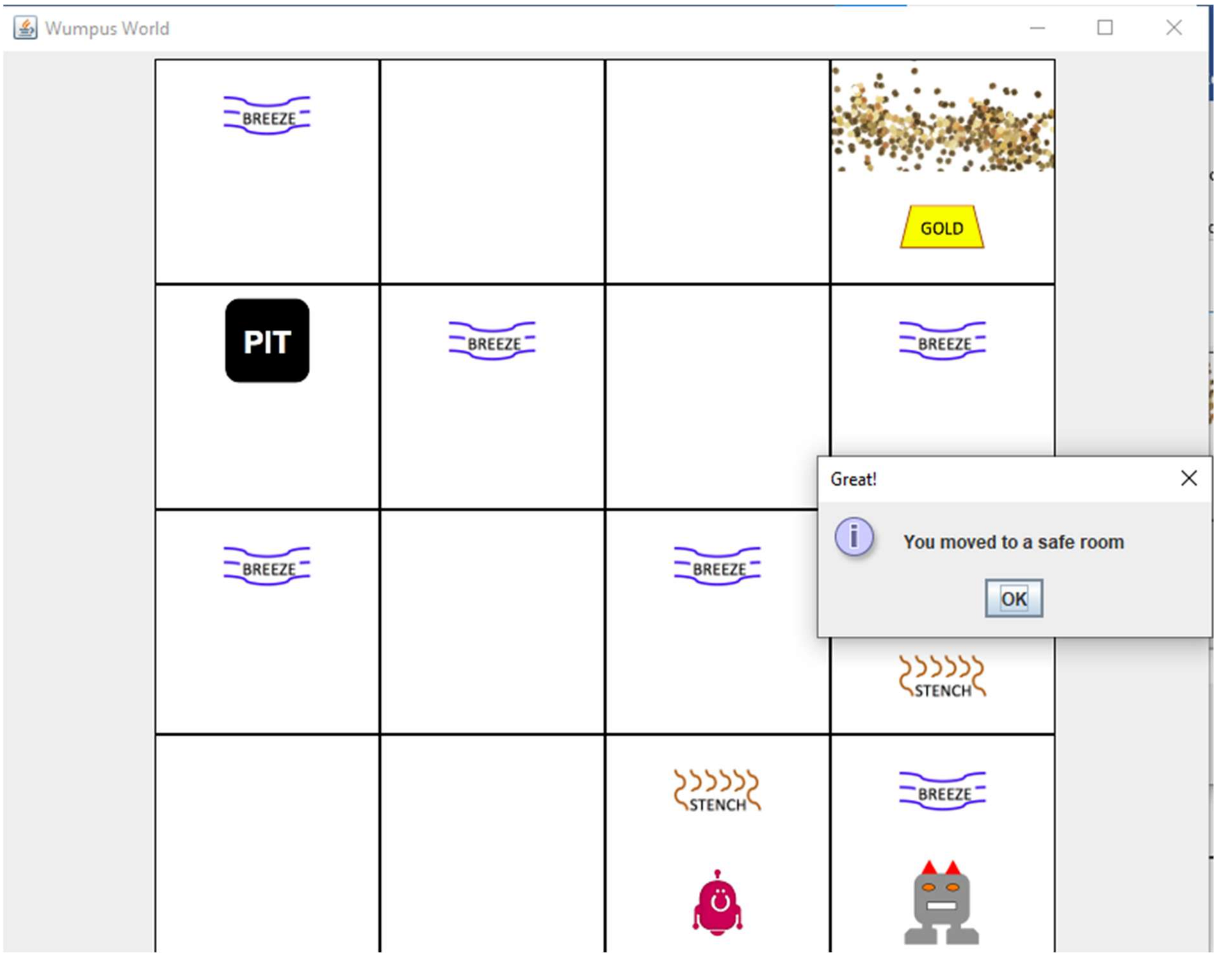


You moved to a safe room

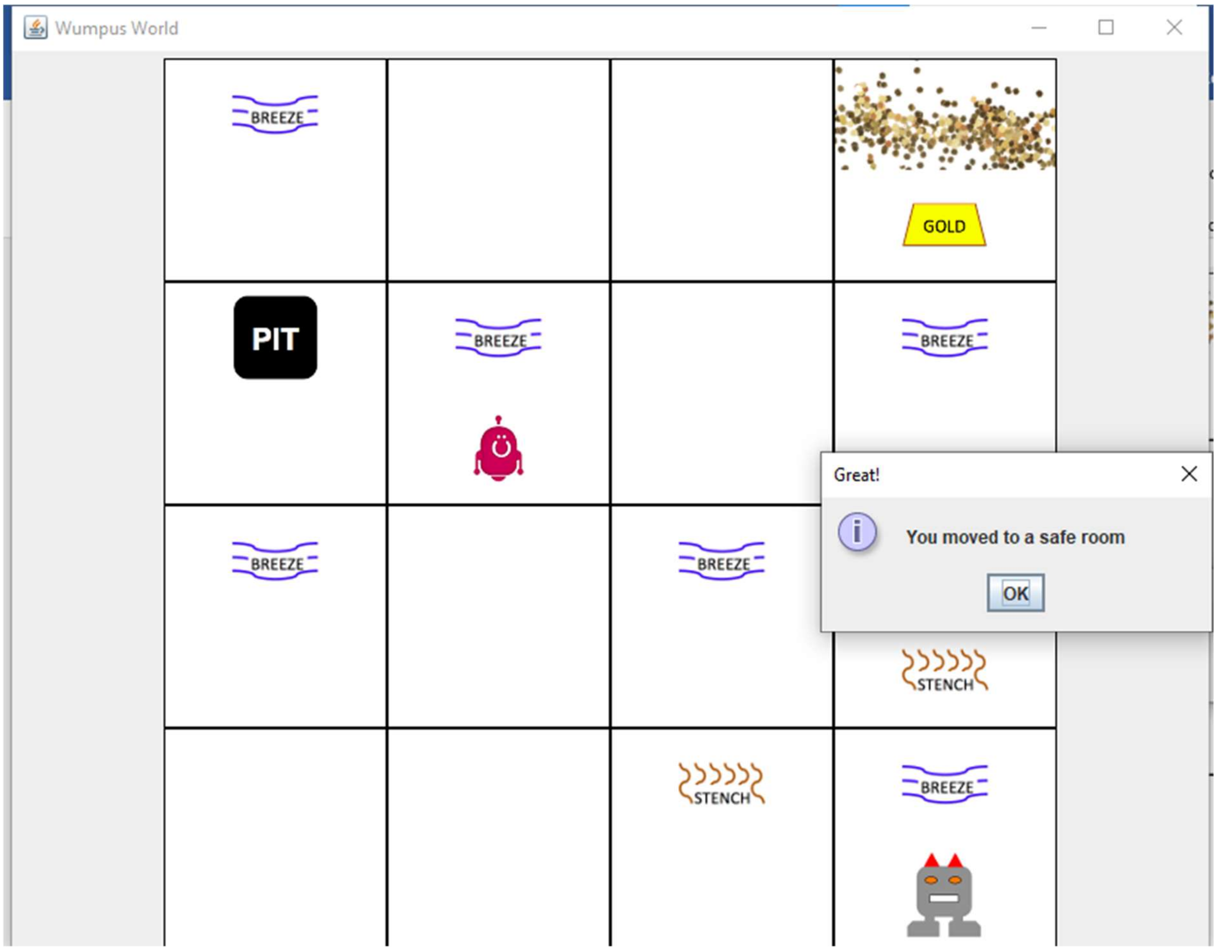


Great!

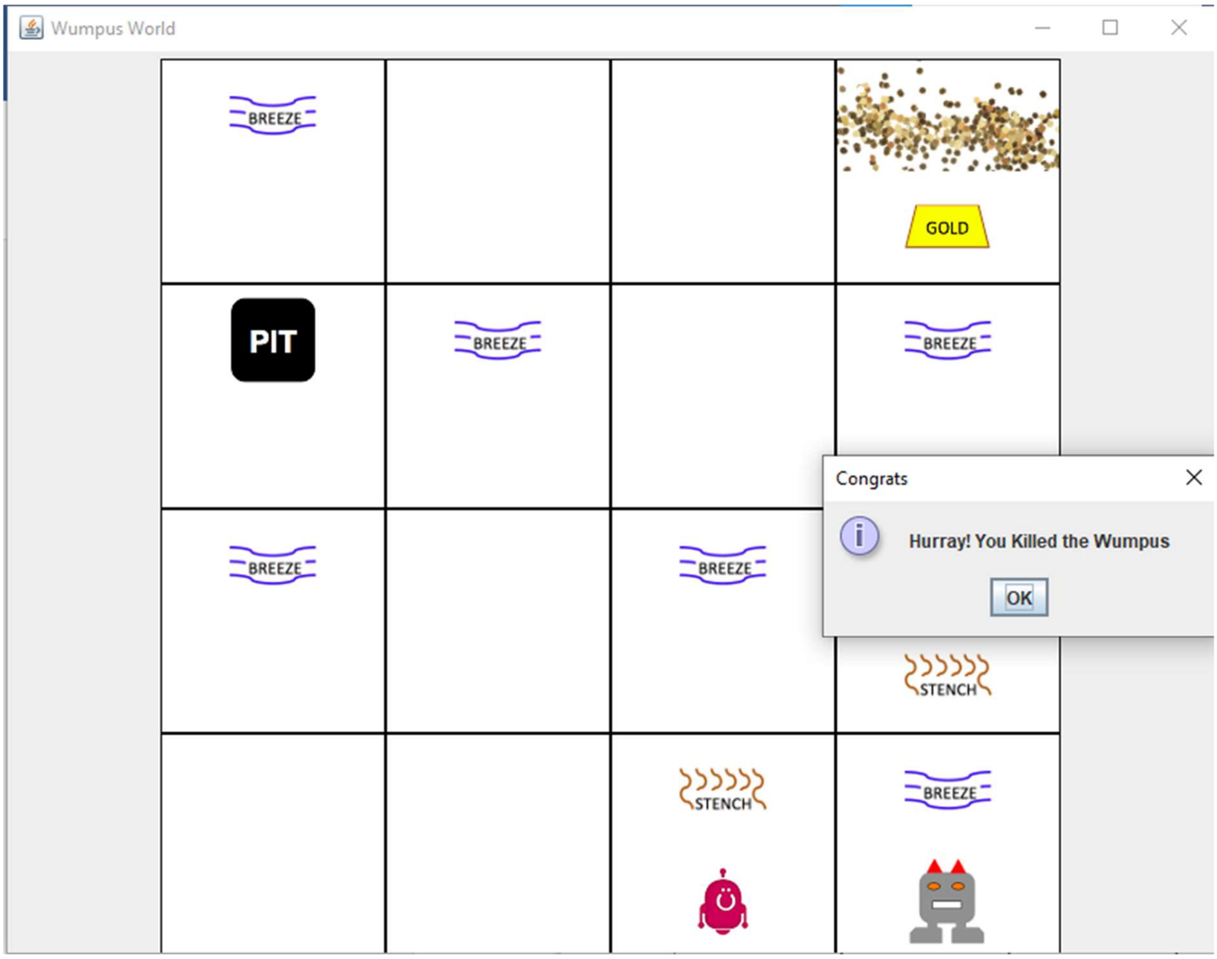
You moved to a safe room



You moved to a safe room

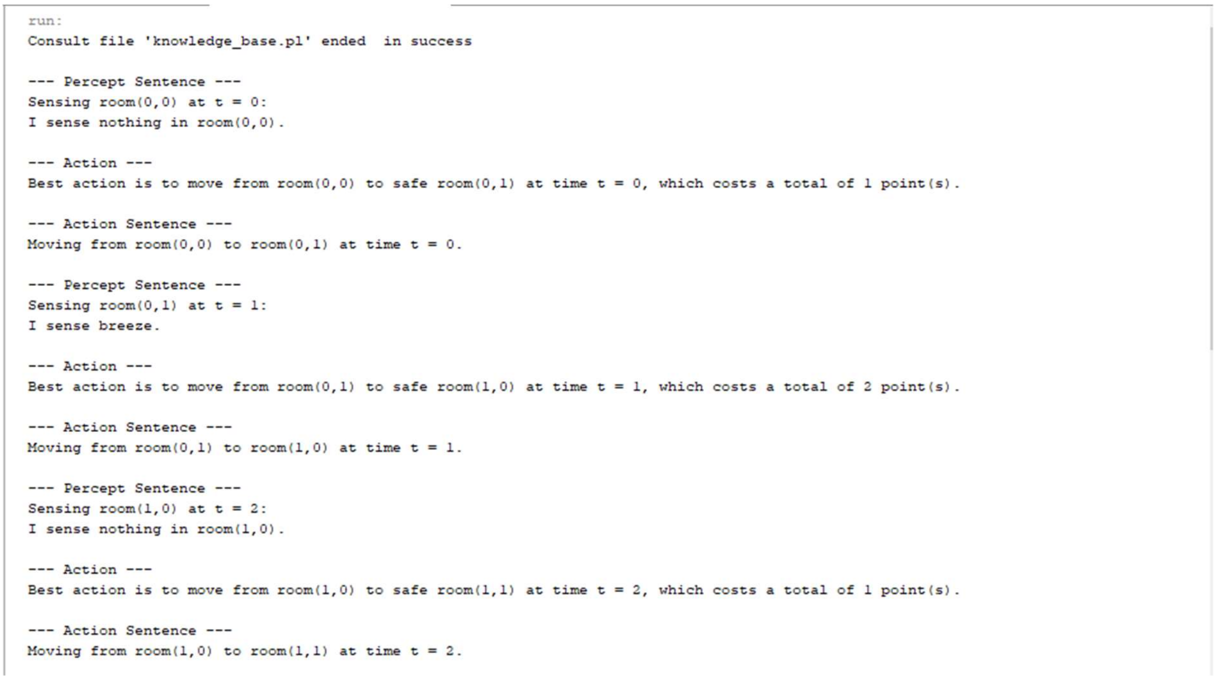


You moved to a safe room

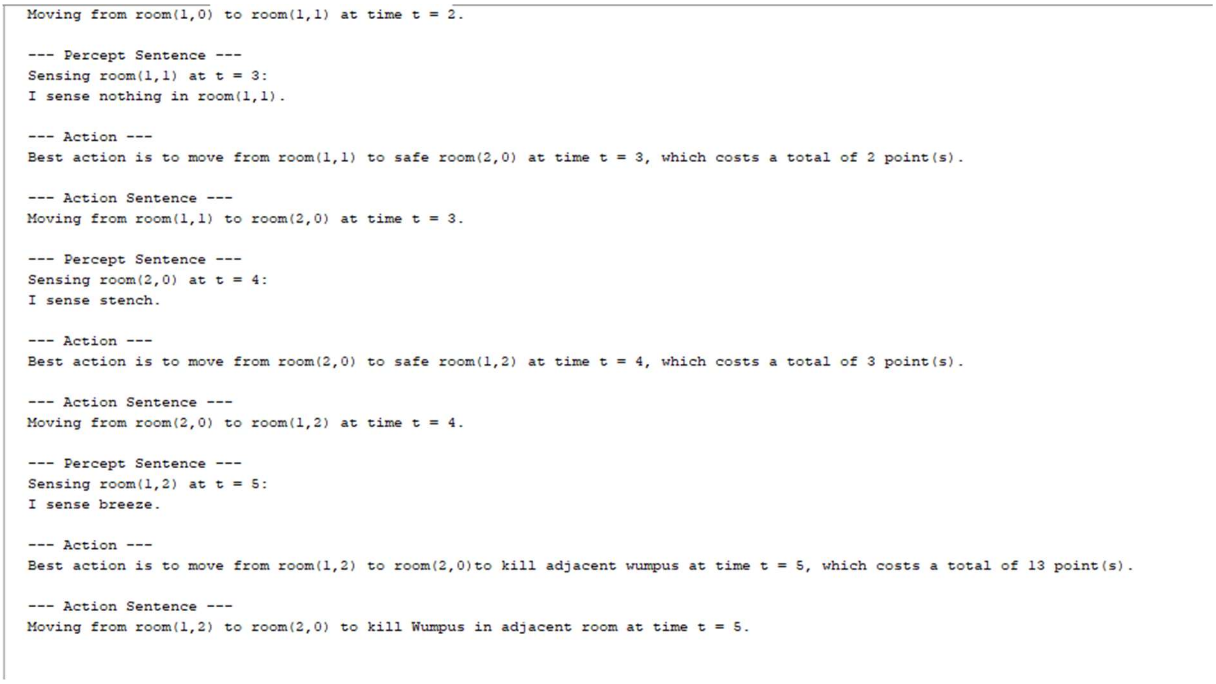


Congrats

Hurray! You Killed the Wumpus



```
run:
Consult file 'knowledge_base.pl' ended  in success

--- Percept Sentence ---
Sensing room(0,0) at t = 0:
I sense nothing in room(0,0).

--- Action ---
Best action is to move from room(0,0) to safe room(0,1) at time t = 0, which costs a total of 1 point(s).

--- Action Sentence ---
Moving from room(0,0) to room(0,1) at time t = 0.

--- Percept Sentence ---
Sensing room(0,1) at t = 1:
I sense breeze.

--- Action ---
Best action is to move from room(0,1) to safe room(1,0) at time t = 1, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(0,1) to room(1,0) at time t = 1.

--- Percept Sentence ---
Sensing room(1,0) at t = 2:
I sense nothing in room(1,0).

--- Action ---
Best action is to move from room(1,0) to safe room(1,1) at time t = 2, which costs a total of 1 point(s).

--- Action Sentence ---
Moving from room(1,0) to room(1,1) at time t = 2.
```



```
Moving from room(1,0) to room(1,1) at time t = 2.

--- Percept Sentence ---
Sensing room(1,1) at t = 3:
I sense nothing in room(1,1).

--- Action ---
Best action is to move from room(1,1) to safe room(2,0) at time t = 3, which costs a total of 2 point(s).

--- Action Sentence ---
Moving from room(1,1) to room(2,0) at time t = 3.

--- Percept Sentence ---
Sensing room(2,0) at t = 4:
I sense stench.

--- Action ---
Best action is to move from room(2,0) to safe room(1,2) at time t = 4, which costs a total of 3 point(s).

--- Action Sentence ---
Moving from room(2,0) to room(1,2) at time t = 4.

--- Percept Sentence ---
Sensing room(1,2) at t = 5:
I sense breeze.

--- Action ---
Best action is to move from room(1,2) to room(2,0)to kill adjacent wumpus at time t = 5, which costs a total of 13 point(s).

--- Action Sentence ---
Moving from room(1,2) to room(2,0) to kill Wumpus in adjacent room at time t = 5.
```

Limitations of the Solution

The first limitation that we have in our solution is the lack of randomness. Meaning, the agent is purely deterministic. In other words, if we run the code with the same starting configurations we are going to end up with the same result (if the agent loses in a certain configuration and we run the same configuration again, the agent is certainly losing again).

Another case in which the lack of randomness shows is the attempt to shoot. Meaning, if the agent does not know for certain that there is a Wumpus, it will never shoot randomly.

The future remedies that can be done to solve this issue is to define a random move if it got stuck. The concept is mainly similar to what we have already discussed in class (schotastic hill climbing). However, I am thinking about how we can increase the probability of getting the best random move. I think the latter is going to be challenging to code.

Note: if you would like to run the code with your own generated configurations, uncomment setRooms() in the constructor of WumpusPlanet class and update a CSV file according to the new configurations that you would like to be updated.