Caio Feiertag (1510590) - Lorenzo Saraiva (1110649) - Johann CORCUFF-REBECCHI (1720167)

Inteligência Artificial

Projeto 2

19/11/2017

INF1771

PUC
RIO

Index

I. Introduction

This project had for objective to implement a game, named “World of Wumpus” where a player try to discover as much gold as possible on a map, avoiding enemies and tricks. Using prolog we could dispose from a fact database, keeping the history of the records, that allow us to use logical operators in order to implement the artificial intelligence to guide the player inside the map.

II. Definition of the problem

Sensors allow the player to detect enemies, pits where it can fall and gold. The goal is to lead the player toward the map to make him discover all the gold without dying.

III. Methodology

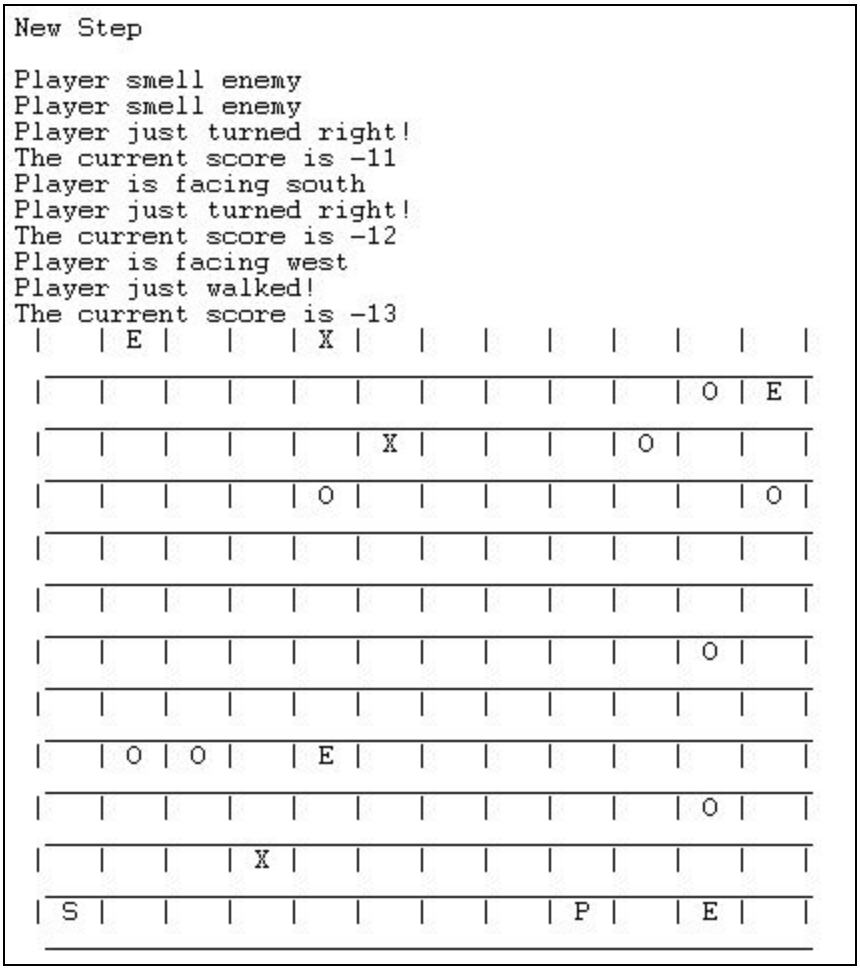
A. How to execute and understand the program

The game has been written exclusively in prolog. The graphic interface is using the native console interface of this language.

In order to execute the game you just have to open the **WumpusSafePlayer.pl** or **WumpusRiskyPlayer.pl** file with **Swi-prolog** and execute the following command on the terminal opened :

```
?- run(init).
```

You will see the console interface updating every second the game platform :



S represent the home of the player from where he is starting at the beginning and where he will come back at the end of its researches, P represent the player, E the enemies and O the pits.

At the end of the game, when the player will return in his home or if he dies, the loop will stop and you'll see your final score. If you want to relaunch the game with a different map, simply re-open the file and launch this same command.

There are 2 versions, because on the first one the player just avoid tricks, on the second one he can take some risks and shoot wumpus.

B. Prolog commands glossary

Our prolog file contains two type of facts : static facts and dynamic facts created at the initialization of the game and updated all along its execution for some of them. It contains also action and utility procedures.

Static facts :

```
amount_pit(8) // there are 8 pits in the game
amount_gold(3) // there are 3 gold treasures in the gale
worldSize(12) // the world is a board of 12 cases on each side
bulletDamage(20,50) // enemies can make either damage the player of 20 or
50 points
element(1,1) // the home of the player is situated at the X=1 and Y=1
coordinates on the board
```

Dynamic facts :

```
currentPos/2 // The actual coordinates of the player
points/1 // The score of the player
shots/1 // The number of shoot the player still has
currentLife/1 // The current life level of the player
initialPos/2 // The initial coordinate of the player
direction/1 // Its actual direction : north, east, west, south
visitedTiles/2 // All the visited square  by the player
enemy/4 // Coordinates, life level, and damage strength of enemies
pit/2 // Coordinates of the different pits
gold/2 // Coordinates of the different gold
element/2 // Coordinates of all the elements present on the board
safeTiles/2 // Coordinates of all the safe square the player encounter
possible_pit/2 // Possible coordinates of the pits the player encounter
possible_wumpus/2 // Possible coordinates of the wumpus the player
encounter
scrumble/2 //Coordinates of the bread crumbs the player drop on the board
to come back home
players_pit/2 // positions of the pits player identifies
players_wumpus/2 // positions of the wumpus player identifies
temp/2 //help to save the positions of bestPath method
```

Action procedures :

```
walk_forward :- // makes the player walk forward in its actual direction
turn_right:- // makes the player turn right from its actual direction
turn_left:- // makes the player turn left from its actual direction
grab_object:- // makes the player grab the gold treasure
climb_up:- // makes the player leave his house
apply_damage:- // makes the player receive the damages when it is walking
```

```
on a pit or wumpus
shoot:- // makes the player shoot in the direction he is
apply_damage_to_enemies_column(X, [H|T], Damage):- // makes the wumpus
receive the damages
apply_damage_to_enemies_line([H|T], Y, Damage):- // makes the wumpus
receive the damages
decision_maker :- // brain of the player to know what to do
movements:- // actions taken by player when he feels safe
go_back:- // action taken by the player when he feels something wrong
safe_surroundings:- // marks surroundings as safe
handle_breeze:- // action taken by the player when he feels a breeze
handle_smell:- // action taken by the player when he feels a wumpus
explore:- // action taken by the player when he decide to explore a new
square
find_direction(C,N):- // finding the best turn to face the desired
direction
gohome:- // makes the player go back home after the exploration
Collect_scrumble:- // collect bread crumbs in order to move on step forward
home
adjecent(X,Y,M,N) :- // Defining when the player is adjacent to pit or an
wall(X,Y) :- // sensor allowing the player to feel a wall
smell(X,Y) :- // sensor allowing the player to feel a wumpus
wind(X,Y) :- // sensor allowing the player to feel a pit
shine(X,Y) :- // sensor allowing the player to feel gold
Gohome:-  //  use it to go to the first position
identify_wumpus(X,Y):- // player identify where are the wumpus
identify_pit(X,Y):-     // player identify where are the pits
Going_home:- // Go home use it step by step
best_path(X,Y,A,B):- // calculate the best path beetween 2 positions
best_pathHome(X,Y):- // calculate the best path to the first position
Move_risk:- // attempt a risky move when smell a wumpus or a pit
Shoot_scream:- // method to shoot until the enemy dies
Checa_enemies:- // verifies if the wumpus died
Explore_unsafe:- // it is the same as explore but doesn’t need to go to a
safe position
```

Utility procedures :

```
/* Creating the world */
randomizeElements :-
```

```
generate_enemies(N):-
generate_golds(N) :-
generate_pits(N) :-

/* Initializing the world */
init:-

/* Running the game */
run(init):-
run(nextStep):-
nextStepCaller:-

/* Drawing the game */
draw_horizontal(N,M):-
draw_playersvision(N,M):-
```

B. Group work

We have been working together and interacting a lot in order to advance the project together and complete everyone's job.

Lorenzo Saraiva started to build the prolog file. He first understood the language and helped the other member to join in. He designed the architecture of facts and coded many action and utility procedures. He then worked on the report.

Caio Feiertag worked on the artificial brain of the player to make him research the gold treasures avoiding tricks and then come back home and the risky version. He then worked on the report.

Johann Corcuff-Rebechi, tried before the group entered in prolog, to design a graphical interface in Python and then in C#, but it was a failure. He then coded different action and utility procedures with a little focus on the Prolog console interface. He then worked on the report.

E. Problems encountered and solutions

We had different problems along the project. The first thing we did was to divide the project between the Prolog itself and the integration with C#. We faced problems in both understanding how to apply the logic from simple examples of Prolog in a more complex program and in integrating Prolog and C#. We decided to focus on the Prolog implementation and took some days just to study. With that, we managed to get a knowledge that was sufficient

to start the project, creating the first facts and functions, and decided that doing a graphic interface and the whole program fully in Prolog was viable, making the integration with another language unnecessary. After we finished the basic interface and movement, we continued to the creation of the decision_maker, the procedure that is the brain of the AI.

The first problem encountered during the phase was that the player would avoid all obstacles, but would easily get in a loop through the tiles he visited if there were dangers close by. To solve that, we create the function Explore, that makes the player go to a spot that he knows is safe, but hasn't visited yet. This improved significantly the result, but sometimes the player would have no unvisited tile and end up trapped, finishing the game. To solve that the function collect_scrumble was created. This function store the player's last option - it makes him go back from where he came from, and repeats that until he can again go to an unvisited safe location.

Our last important problem was to make the player go back to his home after getting all the gold. We also used the collect_scrumble function, so that the player only use perfectly safe positions to go back. We then made an optimization to let him go home quicker for a better score.

Then we created methods to identify the tricks on the map, but it doesn't improve our solution. Finally, the last approach was to create something to take risks, so he could avoid some tricks on the first positions. That way we created 2 versions, one that avoid all tricks and play safe, and another new one that can shoot and take risks.

IV. Results

The 2 versions of the player use two different strategies. One is taking risk and the other one avoid them entirely. After 10 different runs of each games, we could have the following results

WumpusSafePlayer.pl - Player do not take any risks

Score 1 : 2779 - 3 gold taken
Score 2 : 2809 - 3 gold taken
Score 3 : 837 - 1 gold taken
Score 4 : 2796 - 3 gold taken
Score 5 : 2819 - 3 gold taken
Score 6 : 2825 - 3 gold taken
Score 7 : -1 - Player felt a risk just next his home, so he went back
Score 8 : 2735- 3 gold taken
Score 9 : 2823 - 3 gold taken

Score 10 : -1 - Player felt a risk just next his home, so he went back

Average : 2042	Max : 2825	Min : -1

WumpusRiskyPlayer.pl - Player do take some risks

Score 1 : 2857 - 3 gold taken
Score 2 : 2651 - 3 gold taken - 1 wumpus killed, a second hurted
Score 3 : -1003 - Player took risk and felt in a pit
Score 4 : -1004 - Player took risk and felt in a pit
Score 5 : 2819 - 3 gold taken
Score 6 : -1010 - Player took risk and felt in a pit
Score 7 : -1013 - Player took risk and felt in a pit
Score 8 : 2693 - 3 gold taken - 1 wumpus hurted
Score 9 : -1003 - Player took risk and felt in a pit
Score 10 : 2773 - 3 gold taken - 1 wumpus killed

Average : 876	Max : 2857	Min : -1013

Globally the player not taking any risk is more efficient with an average two times superior to the one of the player taking risks. However the maximum score is reached by the player taking risks. At least our risky player kills wumpus and do not hesitate to go out from his home even if a trick is next to it.

V. Conclusion

Based on the results, we can conclude that the first version is better because the average is higher than the second version. We had a lot of trouble with the language and we think that's why the first version that doesn't take any risks is better.

One possible path would be to develop a player that measures risks and takes it only when this one is low or if he doesn't have any other options.