Programming Languages

Final Project

Prof. Benjamin Valdés

Eric Fernando Torres Rodríguez

A01700249

22/11/2021

Index: -

Introduction: -

Text base games trace as far back as teleprinters in the 1960s, when they were installed on early mainframe computers as an input-and-output form. These games shaped the interaction between users and the fictional environment that surrounded them.

The core functionality of these games consists in the input and output of data in form of text and simple decision making, in this document we will cover the development of a text base game with logic programing using Prolog and the applications that lead to use this language for the development of this project.

Solution: -

Based on my passion for videogames in all their presentations and as a tribute for the father of the now games that I enjoy, I created a text base video game using Logic Programming with Prolog, which interact with the user in an interactive word with items, puzzles, and enemies, all of these using prolog language commands.

The rules of this game are base in several predefine actions that as a user you can implement to interact with your environment and surroundings, each action can have or not consequences, lead to another place or even to death.

Prolog in the project:

Since text base games consist in simple instructions and in a finite number of results were most cases end up asking if something can be done or not, therefore prolog is an easy approach to develop a solution to an interactive and immersive experience because prolog is a logic language that is particularly suited to programs that involve symbolic or non-numeric computation.

Prolog consists of a series of rules and facts. A program is run by presenting some query and seeing if this can be proved against these known rules and facts.

Prolog is a language also uses two main functionalities that make them a great program for a text-base game: unification and backtracking, with the use of these we can unify facts in a knowledge base and create predicates and rules that can use the backtracking to create situation where if some criteria is not accomplish, we can have another result. For example, if a rule were a certain item is needed is accomplish, we can have access to a specific area of the game or not.

Implementation: -

1. Simple facts

Prolog facts make a statement over something, in the next figure we ca see for example that we declare that the first position of the user is the monastery or that the cult is alive. These simple facts help in the development of certain things that we need to be set in our world.

```
/*Start in the spawn*/
position(monastery).

/* This fact specifies that the cult is alive. */

alive(cult).
```

2. Fact with arguments to create connections

One of the core functionalities of our text base program are the connection between locations in our world, we use facts with arguments to create the imaginary roads that take the user to different locations. As an example, in the next image we can see that we initialize a “path” that declares that from the monastery you can go to the south and arrive to the forest.

```
/* Monastery*/
path(monastery,south,forest).
path(monastery,west,river).
path(monastery,east,town).
```

3. Rules and backtracking

Another pilar of the program and maybe the most important of all, are the rules, which allow us to make conditional statements about our world. Each rule can have several variations, called clauses. These clauses give us different choices about how to perform inference about our world.

In the next figure we can see a clear of example of the use of rules and backtracking, first we declare a rule that tell us that the path from the cave_entrance that goes to the east to arrive to the cave will be true only if the fact item_on(lantern, in_hand) is also true. Consequently, the second rule that only instructs to write a message and then fail so the rule does not return a true statement.

```
/* Cave entrance*/
path(cave_entrance,east,cave):-
  item_on(lantern, in_hand).

path(cave_entrance,east,cave):-
  write('It is to dark to move foward'), nl,
        !, fail.

path(cave_entrance,west,forest).
```

4. Unification

To retrieve something from our knowledge base, we make use of the variable unification.

```
/* This rule tells how to move in a given direction. */

move(Direction) :-
        position(Here),
        path(Here, Direction, There),
        retract(position(Here)),
        assert(position(There)),
        position(Place),
        scene(Place), !.
move(_) :-
        write('You can''t go that way.').
```

As we can see the rule move receives a variable called “Direction”, therefore we call the fact position with another variable that is “Here” that retrieves from the knowledge base the position where the user is. A query then is made were the path now consist in the actual position, to one direction and a variable that is going to be search in the knowledge base to know if there is any fact that accomplish the query. If the query results in true the rule continues, however if it fails the Direction will go into the other rule move printing a message that we can go in that direction.

5. Retract and Assert

In addition to what was explained in the last point, the functions retract and assert are use to manage facts in a dynamic way, retract takes a fact with the actual value that is saved and deletes it, then assert takes the same fact and saves a new variable, as we can see in the last image we retract the actual position where the user is and change it for the new position where the user moved.

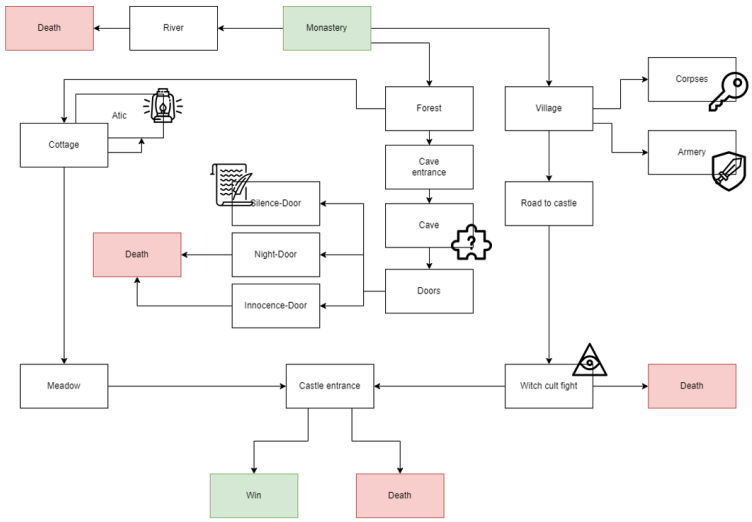
Game guide: -

Once the main functionality and application in prolog is set and understood we proceed to declare how to interact with the game and the world.

Commands

- **Start:** Command to start the game in the first location.
- **Instructions:** Deploy a menu with all the commands in this list
- **North, south, east, west, up, down:** Direction commands to move between places.
- **Take:** This command allows you to pick up items that are around places and can help or are necessary to advance in the history.
- **Attack:** As the name suggest this command makes you attack NPC's or enemies that you may find in the road.
- **Talk:** Use to talk with NPCs for information.
- **Look:** The user can use this command to look their surroundings in search for items.
- **Info:** Deploy info of the items collected.
- **Exit:** Exit the game.

Map



Items

- **Lantern**: Use to light up dark places.
- **Key:** Open buildings in the village.
- **Sword:** Use to attack enemies.
- **Shield:** Use to protect.
- **Riddle:** Riddle to pass the doors.
- **Letter:** Letter of a hierarchy of the witch cult.
- **Witch cult ropes:** Ropes of the witch cult.

Test: -

We run the program with the command **swipl –quiet adventure.pl**

```
PS D:\Documentos\Tec\7mo\Lenguajes\FP-rep\TextAdventure> swipl --quiet adventure.pl
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:115:
Warning:    Singleton variables: [X]
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:121:
Warning:    Singleton variables: [X]
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:124:
Warning:    Singleton variables: [X]
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:127:
Warning:    Singleton variables: [X]
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:130:
Warning:    Singleton variables: [X]
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:133:
Warning:    Singleton variables: [X]
Warning: d:/documentos/tec/7mo/lenguajes/fp-rep/textadventure/adventure.pl:136:
Warning:    Singleton variables: [X]
1 ?-
```

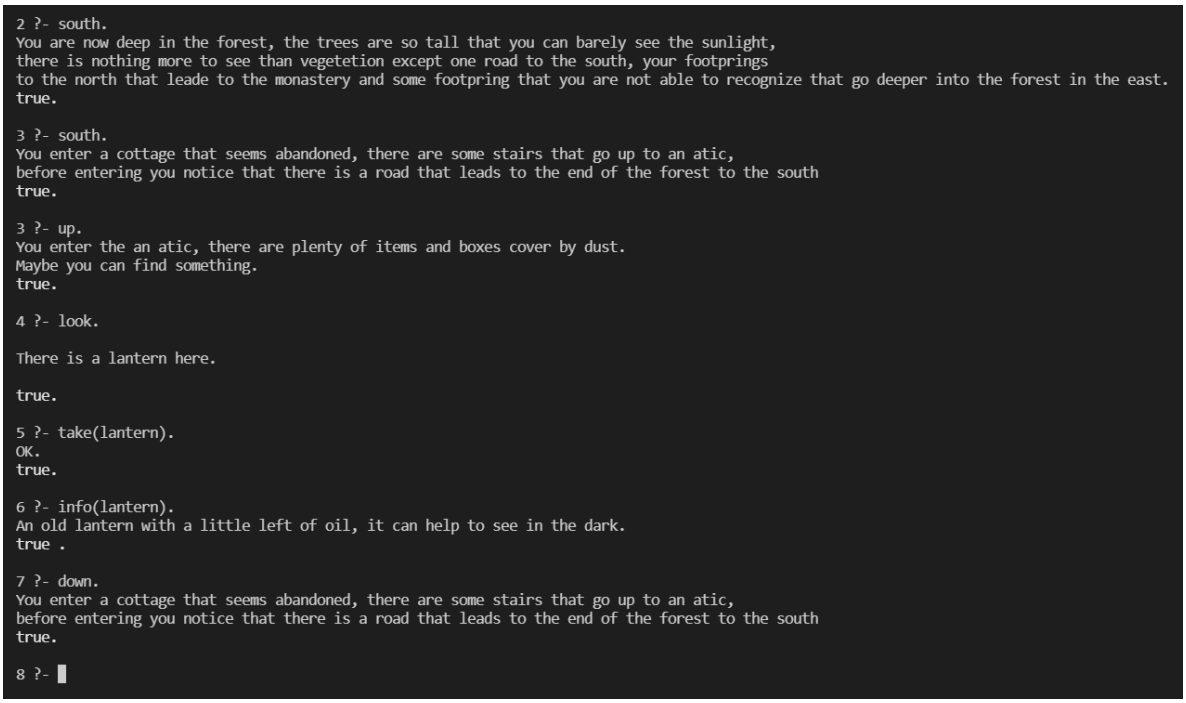
```
1 ?- start.
To enter this world you will have to write the next list of actions (Prolog sintax).
+ This are the actions that you can choose:
+ You will move through the world with the next direction comands          ---- north.  south.  east.  west.  up.  down.
+ You will be able to pick up objects that you find in teh world by typing ---- take(item).
+ Defend yourself from enemies if you have the something to do it.         ---- attack.
+ Talk with npcs around the world to acces places or know information.     ---- talk.
+ To describe again were you are.                                         ---- look.
+ Deploy information of the items.                                        ---- info(item).
+ Deploy this message again.                                              ---- instructions.
+ Exit the game and the program.                                          ---- exit.

You wake up surroended by smoke and fire, the monastery is destoyed, all you can see is your fellow monks laying dead on the ground.
The last thing you rememeber are person wereing a purple tunic with a icon on the center that represent only one thing. The cult of the witch
You need to advise the others about the danger that is coming to the kingdome, you need to go see the king who lives in the south and tell him what you saw
That is our only hope
You see the outside of the monastery it is destoyed to the goround it was a miracle that you suriveve.
There is a river to the West, a forest to the South and a road that lead to a village in the East
true.

2 ?-
```

```
2 ?- west.
You come across a long river, you can berealy see the other side of it, it is to deep and the current seems strong.
You can attempt to cross it straight by going to the south, or you can go back to the monastery by going east
true.

3 ?- south.
You are trap by the strong current, you were always an amazing heroic fighter but not a good swimmer, you drown in your own shame.
YOU DIED
PS D:\Documentos\Tec\7mo\Lenguajes\FP-rep\TextAdventure>
```



Set up instructions: -

1. **Download the project:**

   You can either clone the repository or download the adventure.pl file to access the program.
2. **Install SWI-Prolog:**

   Go to SWI Prolog and download the version of prolog that fills your computer needs.
3. **Run the code:**

   Open the directory where the file is located and run the command **swipl –quiet adventure.pl**

Conclusions: -

To summarize text base games are the base of interactive games that later became the video games that we nowadays have. The use of prolog as a tool to develop a text base game was a little challenging in the beginning nevertheless with the correct understanding of the paradigm and use of the language, I was able to deliver a project that reached my expectations.

If I were to repeat this project, I would develop more dynamics for the user to increase the immersive experience and accomplish a deeper narrative. Even so I am happy with the result of this project because videogame have always been an important part of my life, to be able to make my own and play as the creator of worlds is a totally different experience.

As I already said this project was an adventure, it had difficulties, puzzles to be solved, enemies to be defeated, but it also had NPCs and tools that help me to accomplish my mission. I hope to keep this project as a valuable experience and inspiration for new to come adventures.

Bibliography: -

Brna P (S.F) Prolog tutorial. Introduction to prolog. Retrieved from: http://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/