LOAL - Final Assignment: *Text Adventure*

KARP 21-04-19 V1, based on David Matuszek's great *UPenn*-courses

Objective

Use your *Prolog* knowledge to create a *Text Adventure* game.

Things To Learn

- Write *Prolog* facts and rules with lists and recursions.
- Use *cuts* to prevent backtracking.
- Modify the knowledge base at runtime using `assert` and `retract`.

Materials

- A *Prolog* interpreter. Recommended ones:
  - *GNU Prolog*
  - *SWI-Prolog*
- A text/*Markdown* editor capable of generating *.pdf*-documents.

Submission Guidelines

- A brief and **spoiler-free** description of your game as a neatly-formatted one-page *.pdf*-document. Write about your idea, the mechanics, include a *map* of the world if you like. Make the reader want to play your game!
- Your *Prolog* program as a `.pl`-file. Make sure to use meaningful names for variables, constants and predicates and include comments where necessary! **Make sure to implement a `start/0` procedure to give the player an overview of possible commands!**

Background

In the early days of computing - dark times, long before diverse colour palettes, *GUIs* and *WASD* controls and even longer before three-dimensional game graphics we're all used to now - early games were often developed using text-based interfaces. As you can infer from the name, both input and output of such games were represented using text: The player would type simple commands (e.g. `TAKE SWORD` or `GO NORTH`) to interact with his/her surroundings, while the game would answer by describing the world. This genre of so called *text adventures* can be seen as the precursor to modern *Point-And-Click-adventures*.

The first famous games of this genre include *Adventure* (also known as *ADVENT* and *Colossal Cave Adventure*) developed by William Crowther and Don Woods in 1975 and the *Zork* series published by *Infocom* in 1980. If you're curious: The first one is included in the `bsdgames` package for *Debian*-based systems, the latter one can easily be found online.

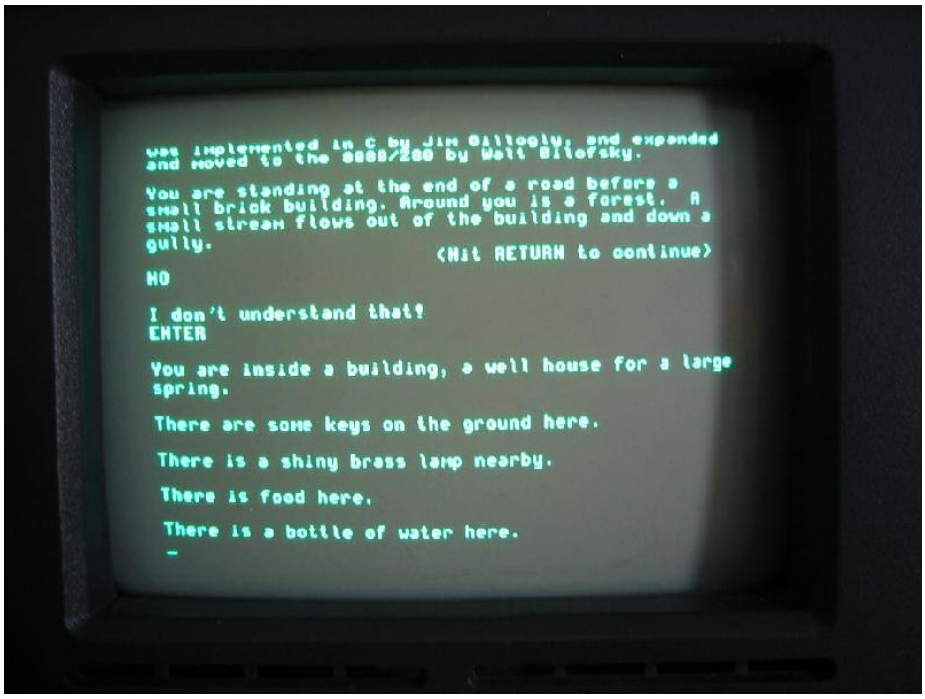

Figure 1: The game *ADVENT* on an *Osborne 1*, ca. 1982.

Tasks

Turns out that *Prolog* is really suitable for writing such a game. Your objective is to pick any theme you like - whether it's a wild treasure hunt on an alien planet or the quest of arriving in school on time is up to you - and create an adventure game based on your idea.

A key concept in adventure games are puzzles. Make sure to include at least one of the following in your game:

- *Locked doors*: In its most boring form, you must find a key and use it to unlock a door, thus giving you access to one or more additional rooms. With a little more imagination: You aren't admitted without a badge. You need to buy a ticket. You must give the troll a gold piece before you can cross the bridge. Waving the magic wand causes the rainbow bridge to appear. Et cetera. Any sort of locked door puzzle will do.
- *Hidden objects*: Boring form: You open a box and find something inside. More interesting: You break open a treasure chest. You use the combination to a safe. You peer into the crystal ball. You buy the candy bar from the vending machine. You disassemble the robot to get some part out of it.
- *Incomplete objects*: Your flashlight needs batteries. Your gun needs bullets. Your car needs gas. Your bicycle has a flat tire. You need a computer to get at the information on a floppy disk. You are a zombie and need a brain.
- *Limited resources*: You have a limited amount of time (to find the bomb before it goes off) or money (to buy the things you need), or food, drink, or sleep (so you don't collapse), or some other resource. Maybe you can find more resources in the game, maybe you can't. Depending on just what you decide to do, you may want to figure out how to do arithmetic in Prolog.

Make sure to include a `start/0` procedure to give the player an overview of possible commands and an `inventory/0` procedure to tell the player what he/she is holding.

Template

You can use the supplied file `adventure_template.pl` as starting point for writing your adventure. You can add to this file to create your own game - if some functionality is missing, include or fix it.

As inspiration I also included the simple examplary game *Spider* by David Matuszek, on whose great course this assignment is based on.

Grading

The absolute minimum for a positive grade is a working game with a reasonable extent of rooms, at least one puzzle and the documentation described in the *Submission Guidelines*. For a better grade include more puzzles or raise their

complexity - include advanced topics like *lists* or concepts we poorly covered such as *arithmetics*. Bonus points are also awarded for the idea and how fun your game is.

Please note, that due to the nature of the assignment, some of the grading may be subjective. If I like the game, the idea and most of all the effort, you may get bonus points. If your game is lacking interest and effort, you may get minus points.

For that reason:

- During development, you can contact me with your idea and your game mechanics to get an estimate of the grade you would receive.
- There will be a *public vote* where I'll let your colleagues try out your submissions. The winners will get bonus points.