Practical Assignment 1

Implementation of a Board Game in Prolog

Description

Objective: To implement a two-player board game in the Prolog language. A board game is characterized by the type of board and pieces, the rules for moving the pieces (possible moves) and the conditions for ending the game with defeat, victory or a draw. The game must allow three modes of use: Human/Human, Human/Computer and Computer/Computer. At least two game levels must be included for the computer. The implementation should also include a suitable user interface, in text mode.

Conditions for Assignment Completion

Group Formation: Groups of two (2) students enrolled in the same practical class. Exceptionally, and only if necessary, groups of three may be accepted.

Group and Topic Choice: The groups are chosen in the activity to be made available for this purpose on Moodle from 16:00 on September 29, 2023. In the first phase, one of the group members must choose the group/topic; in the second phase, the second member will join the group. Each topic (game) can only be chosen by a maximum of seven groups, in order to ensure that all topics are equally selected. The list of available games can be found at the end of this statement.

Deadlines: The assignment (source code and readme file) must be delivered by 8:00 AM on November 6, 2023, in the activity to be made available for this purpose on Moodle, with demonstrations carried out during the week of November 6-10, 2023. The demonstrations should be scheduled with the teacher of each practical class.

Assessment Weight: This assignment counts for 25% of the final grade. Evaluation focuses on implemented features, as well as quality of the code and respective comments, the readme file and participation in the assignment presentation. Grades may differ between group members.

Languages and Tools: The game must run under SICStus Prolog version 4.8, and should work on Windows and Linux. If any configuration is required (beyond the standard installation of the software), or a font other than the default font is used, this must be expressed in the README file, which must also include the steps required to configure and/or install the necessary components (on Windows and Linux). Inability to test the developed code will result in penalties in the evaluation. Ensure that you name the predicates as requested in the description below and that all code is properly commented.

Evaluation

Each group must submit a report in README format and the source code developed, as well as demonstrate the application. The submission should be made on the Moodle platform, in a ZIP file named

PFL_TP1_**TXX_#GROUP**.ZIP

Where **TXX** specifies the practical class (e.g. T08 for class 3LEIC08), and **#GROUP** is the group designation. Example: PFL_TP1_T08_Chess3.ZIP

The ZIP file should contain the **README** file and the PROLOG **source-code**.

The source code must be in a directory named ***src***, and must be **<u>properly commented</u>**. The main predicate, ***play/0***, must give access to the game menu, which allows configuring the game type (H/H, H/PC, PC/H, PC/PC), difficulty levels to be used by the artificial player(s), among other possible parameters, and start the game cycle.

The README file should be structured as follows:

- **Identification of the topic** (game) **and group** (group designation, student number and full name of each member of the group), as well as an indication of the contribution (in percentages, adding up to 100%) of each member of the group to the assignment;
- **Installation and Execution**: include all the necessary steps for the correct execution of the game in both Linux and Windows environments (in addition to the installation of SICStus Prolog 4.8).
- **Description of the game**: a brief description of the game and its rules (up to 350 words); you should also **include the links used to gather information (official game website, rule book, etc.)**
- **Game Logic:** Describe (merely copying the source code is not enough) the design and implementation of the game logic in Prolog. **The starting predicate must be *play/0***. This section should have information on the following topics (up to 2000 words in total):
    - **Internal Game State Representation**: describe how the game state is represented, including board (typically using list of lists with different atoms for the pieces), current player, and possibly captured and/or pieces yet to be played, or other information that may be required, depending on the game. It should include examples of the Prolog representation of initial, intermediate and final game states, and an indication of the meaning of each atom (i.e. how different pieces are represented).
    - **Game State Visualization:** description of the game state display predicate implementation. It should include information about the created menu system, as well as interaction with the user, including input validation. The display predicate should be called ***display_game(+GameState)***, receiving the current state of the game and the player who will make the next move. Appealing and intuitive visualizations will be valued. Flexible game state representations and visualization predicates will also be valued, for instance those that work with any board size, using an ***initial_state(+Size, -GameState)*** predicate that receives the board size as an argument and returns an initial game state.
    - **Move Validation and Execution:** describe how a play is validated and executed, obtaining a new game state. The predicate responsible for move validation and execution should be called ***move(+GameState, +Move, -NewGameState)***.
    - **List of Valid Moves:** describe how to obtain a list of possible moves. The predicate should be named ***valid_moves(+GameState, +Player, -ListOfMoves)***.
    - **End of Game:** verification of the end of the game, with identification of the winner. The predicate should be called ***game_over(+GameState, -Winner)***.
    - **Game State Evaluation:** describe how to evaluate the game state. The predicate should be called ***value(+GameState, +Player, -Value)***.
    - **Computer Plays:** describe how the computer chooses a move, depending on the level of difficulty. The predicate should be called ***choose_move(+GameState, +Player, +Level, -Move)***. Level 1 should return a valid random move. Level 2 should return the best play at the time (using a greedy algorithm), considering the evaluation of the game state, as described above.
- **Conclusions:** Conclusions about the work carried out, including limitations of the program (known issues), as well as possible improvements (roadmap) (up to 250 words);
- **Bibliography:** List of books, papers, web pages and other resources used during the development of the assignment.

You can also include one or more **images** illustrating the execution of the game, showing initial, intermediate and final game states (these game states can be hard-coded directly into the code file for this demonstration of the game state visualization, using predicates similar to the ***initial_state/2*** predicate).

Proposed Topics (Games)

The games to be implemented are board games for two players in which there is no influence of the luck factor in the course of the game. The games do not include dice, draws of any kind or initially hidden information.

Proposed games:

1. Agere - https://boardgamegeek.com/boardgame/397893/agere
2. Apart - https://kanare-abstract.com/en/pages/apart
3. Blinq - https://nestorgames.com/blinq_lona.html
4. Bounce - http://marksteeregames.com/Bounce_rules.pdf
5. Claustro - https://boardgamegeek.com/boardgame/391334/claustro
6. Clusterfuss - http://marksteeregames.com/Clusterfuss_rules.pdf
7. Crosscut - http://marksteeregames.com/Crosscut_rules.pdf
8. Differo - https://boardgamegeek.com/boardgame/375056/differo
9. Dropper - https://boardgamegeek.com/boardgame/384171/dropper
10. Dualma - https://boardgamegeek.com/boardgame/398462/dualma
11. Flügelrad - https://boardgamegeek.com/boardgame/400097/flugelrad
12. Isaac - https://www.iggamecenter.com/en/rules/isaac
13. Momentum - https://boardgamegeek.com/boardgame/73091/momentum
14. Murus Gallicus - https://www.iggamecenter.com/en/rules/murusgallicus
15. Period 5 - https://www.iggamecenter.com/en/rules/period5
16. Push Fight - http://www.boardspace.net/english/about_pushfight.html
17. Shakti - https://www.iggamecenter.com/en/rules/shakti
18. Six Making - http://www.boardspace.net/english/about_sixmaking.html
19. Splut - https://www.iggamecenter.com/en/rules/splut
20. Stacks - https://boardgamegeek.com/boardgame/359328/stacks
21. Stones & Rivers - https://boardgamegeek.com/boardgame/381420/stones-rivers
22. Tactigon - https://tactigongame.com/
23. Traxit - https://boardgamegeek.com/boardgame/392652/traxit
24. Trike - https://boardgamegeek.com/boardgame/307379/trike
25. Wald Meister - https://boardgamegeek.com/boardgame/371135/waldmeister