ProLog2425

Game and Group

The game, Sight, was developed by group Sight_6:

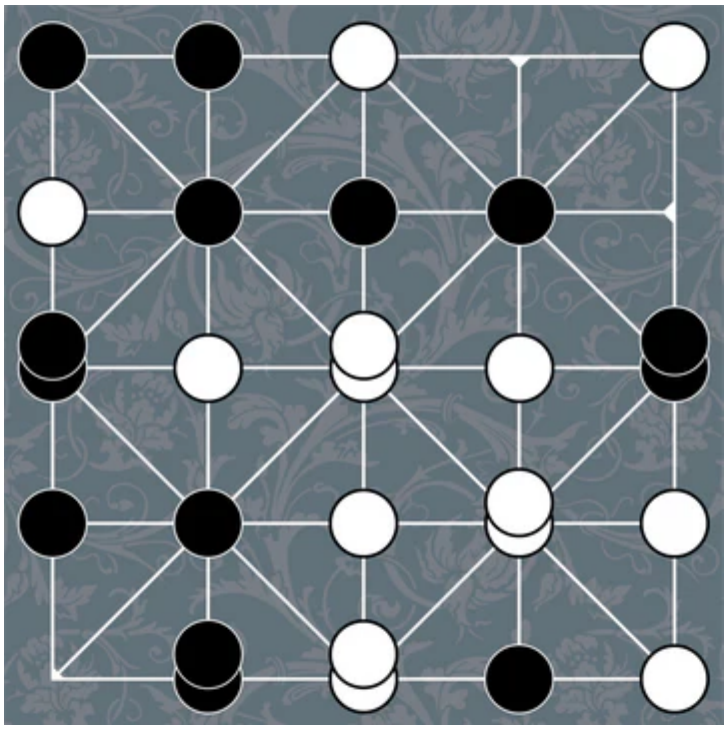
- Gabriela Dias Salazar Neto Silva
- David José Prata Barbedo Magalhães

Installation and Execution

To install this game, you first need to download and unzip the files in PFL_TP2_T04_Sight_6.ZIP. Go to the *src* directory and consult the ***game.pl*** file using the SICStus terminal. Finally, start the game by calling the predicate ***play/0***.

Description of the game

Sight is a game that can be played on the Alquerque board, with the "line of sight" mechanism. Depending on the position on the board, players use two types of actions: placement and movement of pieces, with the objective of eliminating their opponent's legal moves.



In this game, a "stack" is two or more pieces of the same color stacked together. A singleton piece is not a stack. Two pieces on the board are "visible" or "in line of sight" of each other if they are on the same grid line with no other pieces between them. Pieces adjacent to each other on the grid lines are also visible to each other. There are two types of moves in the game:

- When there is no stack of your color on the board: Place a piece of your color on the empty intersection. If there is a friendly piece in the line of sight of the placed piece, add a piece of your own color from your hand to all of them to make two-height stacks.
- When there are stacks of your color on the board: From one of your stacks, the topmost piece is moved to the adjacent empty intersection. If there are multiple stacks, the highest stack has priority. If there is no difference in height, the active player may choose. If there are pieces or stacks of the same color in the line of sight of the moved piece, add your pieces from your hand to all of them to add height. The piece or stack from which the move originated is unaffected.

The game rules are available here

Game Logic

Game Configuration Representation

The *game_loop* is divided into 5 elements:

Board: It's a matrix with dimensions set to 5x5. The initial board is empty, with 11 types of piece representation:

- *empty* = '| |'
- *black1* = '|B1|'
- *black2* = '|B2|'
- *black3* = '|B3|'
- *black4* = '|B4|'
- *black5* = '|B5|'
- *white1* = '|W1|'
- *white2* = '|W2|'
- *white3* = '|W3|'
- *white4* = '|W4|'
- *white5* = '|W5|' As the game progresses, the board is updated:

```
    1    2    3    4    5
  +----+----+----+----+----+
1 ||  |-|  |-|  |-|  |-|  |-|
  +----\----/----\----/----+
2 ||W1|-|  |-|  |-|  |-|  |-|
  +----/----\----/----\----+
3 ||  |-|  |-|  |-|  |-|  |-|
  +----\----/----\----/----+
4 ||  |-|B1|-|  |-|  |-|W1|-|
  +----/----\----/----\----+
5 ||  |-|  |-|  |-|  |-|  |-|
  +------------------------+
```

Player

PlayerNames: Registry containing the names assigned to both players

MoveHistory: Registry containing the move history in the game

TurnCount: Registry containing the curresnt turn count

Internal Game State Visualization

Before the game starts, one of the players will define the settings for the game:

- Game Mode: uman vs Human, Human vs Machine, Machine vs Machine
- Players names
- Difficulty assigned to the machine: Easy/Random or Hard/Scored
- Starting player

Move Representation

After setting the settings, the game enters the appropriate ***game_loop*** predicate. Here, ***valid_moves*** is called, which lists every possible move for the player given the current board. It then prompts the player to choose one of the moves, confirming that it is indeed valid.

```
player1, it is your turn.
Valid moves: [[1,2,1,1],[1,2,1,3],[1,2,2,1],[1,2,2,1],[1,2,2,2],[1,2,2,3],[1,2,2,3]]
Selected move: [1,2,1,3]
Applying stack move: [1,2] -> [1,3] for player1
Move applied successfully.
...
player2, it is your turn.
```

```
Valid moves: [[1,1],[2,1],[3,1],[4,1],[5,1],[2,2],[3,2],[4,2],[5,2],[1,3],[2,3],[3,3],[4,3],[5
Selected move: [5,2]
```

User Interaction

The menu appears upon running the ***play/0*** predicate:

```
| ?- play.
======================
         SIGHT
======================
1. Human vs. Human
2. Human vs. Machine
3. Machine vs. Machine
```

User interaction is done through the ***read*** function, always prompting the user to choose one of the options available

End of Game

The game ends when a player has no more valid moves available, giving the win to the other player. It's checked by the predicate ***game_over/2***.

Computer Plays

There are two level of difficulty in computer plays, random and scored.
Level 1: the bot randomly chooses move from the list of valid moves.
Level 2: the bot uses a scoring system we designed to determine the best possible move to the current board.

Conclusion

Implementing this game was done with a lot of difficulty due to the odd board representation through stacks and the presence of the sight lines. In the end, we managed to implement the game to a fully functional state with a clear and readable interface.