Programação Funcional e em Lógica
Turma 4 - Doblin_6

Alexandre Lopes (50%): teve mais foco no display, gerenciamento de turnos, validação de inputs e cálculo de pontos.

Rafael Campeão (50%): teve mais foco nos níveis das AI, nos movimentos e na sua validação.

Instalação e Execução

Não é necessária nenhuma instalação extra para a execução do nosso projeto, para além do **SICStus Prolog 4.9**. Para iniciar o jogo basta consultar o ficheiro game.pl no diretório src e invocar o predicado **play/0**.

Descrição do Jogo

O Doblin é um jogo multiplayer onde cada jogador possui uma grelha com o mesmo número de linhas e colunas, onde podem colocar o símbolo X ou o símbolo O. As grelhas estão interligadas: ao colocar um símbolo numa grelha, este irá aparecer nas coordenadas correspondentes da outra. Quando um jogador forma uma linha de 4 (horizontal, vertical ou diagonal) do seu símbolo ou um quadrado 2x2, na sua grelha, ele ganha um ponto. O jogo termina quando não houver mais espaços vazios nas duas grelhas. Ganha o jogador com menos pontos.
Site consultado: [Doblin | Board Game | BoardGameGeek]

Considerações Para Extensão do Jogo

Permitimos a escolha de uma grelha de 6x6 até 9x9.Na nossa implementação o jogo está limitado a 2 jogadores.O jogador 1 usa sempre o símbolo X e só tem uma interação direta com a primeira grelha, o jogador 2 usa sempre o símbolo O e só tem uma interação direta com a segunda grelha.

Lógica do Jogo

Representação da Configuração do Jogo

A configuração do jogo tem no total 5 argumentos:

- **Name1 e Name2:** nomes dos jogadores (podem ser humanos ou IAs).
- **Level1 e Level2:** níveis de dificuldade das IAs (apenas usados quando um ou mais jogadores são uma IAs).
- **Size:** tamanho das grelhas.

No predicado initial_state, o valor de Size é usado para criar as duas grelhas e os mappings, e o CurrentPlayer é inicializado com o nome do player1.

Representação do Estado Interno do Jogo

O nosso game state tem um total de 9 elementos:

- **Grid1**: A grelha usada pelo Player1 e onde os seus pontos serão contados. Esta grelha também é afetada pelas jogadas do Player2.
- **Grid2**: A grelha usada pelo Player2 e onde os seus pontos serão contados. Esta grelha também é afetada pelas jogadas do Player1.
- **CurrentPlayer**: O nome do jogador a quem pertence o turno atual.
- **Player1**: O nome do Player1.
- **Player2**: O nome do Player2.
- **RowMapping/ColMapping**: Listas usadas para traduzir as coordenadas da Grid1 para a Grid2.
- **AI1Level/AI2Level**: Níveis de dificuldade das IAs (inteiros) escolhidos pelos jogadores durante a configuração. Estes valores são definidos apenas caso uma IA seja escolhida para jogar.

As grelhas são representadas como uma lista de listas.

Game state inicial:

```
game_state([[_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ]],%grid1

        [[_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ],
        [_ ,_ ,_ ,_ ,_ ,_ ]],%grid2

        CPU1,% CurrentPlayer
        CPU1,CPU2,% Player1 e Player2
        [1,4,5,6,3,2],[3,6,1,2,4,5], % RowMapping e ColMapping
        1,1% AILevel1 e AILevel2)
```

Game state intermédio

```
game_state([[X ,_ ,O ,_ ,_ ,_ ],
        [O ,X ,O ,X ,O ,O ],
        [X ,O ,X ,_ ,X ,X ],
        [X ,_ ,X ,_ ,O ,X ],
        [X ,_ ,O ,X ,_ ,O ],
        [X ,_ ,O ,O ,O ,_ ]],
```

[[O ,_ ,X ,_ ,_ ,_],
[X ,X ,X ,_ ,_ ,O],
[O ,O ,X ,_ ,X ,_],
[O ,_ ,X ,_ ,O ,O],
[X ,X ,X ,O ,_ ,X],
[O ,O ,O ,X ,X ,O]],

CPU2,% CurrentPlayer
CPU1,CPU2,% Player1 e Player2
[1,4,5,6,3,2],[3,6,1,2,4,5],% RowMapping e ColMapping
1,1% AILevel1 e AILevel2)

Game state final

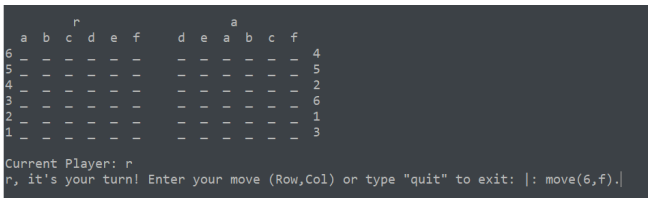
game_state([[X ,O ,O ,O ,X ,O],
[O ,X ,O ,X ,O ,O],
[X ,O ,X ,O ,X ,X],
[X ,O ,X ,X ,O ,X],
[X ,X ,O ,X ,X ,O],
[X ,X ,O ,O ,O ,O]],

[[O ,O ,X ,O ,O ,X],
[X ,X ,X ,O ,X ,O],
[O ,O ,X ,X ,X ,X],
[O ,O ,X ,X ,O ,O],
[X ,X ,X ,O ,O ,X],
[O ,O ,O ,X ,X ,O]],

CPU1,% CurrentPlayer
CPU1,CPU2,% Player1 e Player2
[1,4,5,6,3,2],[3,6,1,2,4,5],% RowMapping e ColMapping
1,1% AILevel1 e AILevel2)

Representação do Move

O predicado **move(Row,Col)** simboliza a posição de um símbolo numa grelha, sendo Row e Col as suas coordenadas. Estas podem ser transformadas pelo predicado **translate_coordinates,** que usa um **RowMapping** e um **ColMapping** para encontrar a sua posição na outra grelha. No predicado **move/3,** a Row e a Col são enviadas para o predicado **update_grid,** onde são colocadas na grid de um jogador e depois traduzidas pelo predicado **translate_coordinates** a seguir as coordenadas traduzidas são colocadas na grid do outro jogador. Se o player2 for humano é feita uma tradução extra antes das coordenadas serem enviadas para o **update_grid**, isto acontece para o Player2 poder usar as coordenadas que aparecem na sua grid.



Interação do Utilizador

No início do jogo, o utilizador deve escolher o tamanho das grelhas. Para tal, basta introduzir um número. A verificação garante que o valor esteja entre 6 e 9; caso contrário, o pedido será repetido até que o utilizador forneça uma entrada válida.

A seguir, é apresentado um menu com 5 opções. As primeiras 4 são os modos de jogo oferecidos, enquanto a quinta permite terminar a execução do programa. O utilizador escolhe uma opção escrevendo o número correspondente no terminal. Se a opção escolhida for a quinta, o programa termina; caso contrário, verifica-se se o número pertence à lista de valores válidos (de 1 a 4). Para finalizar a configuração, o utilizador deve inserir o nome dos jogadores e/ou o nível das IAs. Os nomes devem ter 16 caracteres ou menos e não podem ser "CPU", "CPU1" ou "CPU2", pois estes estão reservados para as IAs. Os níveis de AI disponíveis são: 1(random move) e 2(greedy move).

Com a configuração feita, o único input que o utilizador terá de fazer é o move, indicando onde deseja colocar o seu símbolo. O utilizador também pode escrever "quit" para terminar o programa. A verificação do move assegura que as coordenadas estão dentro dos limites da grelha do jogador e que a posição escolhida ainda não está ocupada.

Conclusões

O jogo Doblin foi implementado com sucesso em Prolog. Pode ser jogado nos modos: Human vs Human, Human vs Computer, Computer vs Human e Computer vs Computer, com a possibilidade de escolher o nível de dificuldade das IAs. Foi implementada uma série de verificações em todos os inputs e jogadas, garantindo a integridade do estado do jogo em todos os momentos.

Algumas melhorias que poderiam ser feitas são: deixar o jogador escolher o seu símbolo e fazer um nível de AI que tenha em consideração o ataque ao adversário e não simplesmente a sua defesa.

Nota: O nosso predicado value não recebe o GameState inteiro, apenas a grelha do jogador, o CurrentPlayer e o Player1 pois estes são os únicos elementos necessários para o predicado desempenhar a sua função.

Bibliografia

Para este projeto foram usados a documentação do SICStus-Prolog e o ChatGPT.

Ao ChatGPT foram pedidas ideias sobre como criar um predicado que encontre o index de um elemento de uma lista, e ajuda relativamente a fazer refactoring de funções de forma a não utilizar símbolos não declarativos como ->.