Eximo

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 04: Eximo
Henrique Manuel Martins Ferrolho - 201202772
João Filipe Figueiredo Pereira - 201104203

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

13 de Outubro de 2014

1 O Jogo Eximo

História

Eximo é um jogo de tabuleiro da família das Damas, concebido em 1 de Fevereiro de 2013.

Detalhes do Jogo

O jogo realiza-se num tabuleiro de dimensões 8x8, em que as casas têm todas cores semelhantes. Cada jogador começa com 16 peças colocadas em locais pré-definidos no respectivo lado do tabuleiro, como mostra a imagem abaixo.

Figura 1: Peça branca

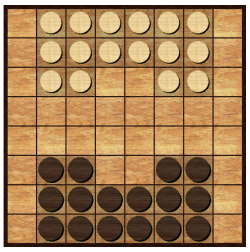

Figura 2: Tabuleiro

Figura 3: Peça preta

No jogo, as movimentações e as capturas podem ser ortogonais ou diagonais. Há apenas um tipo de peça: os homens. Os homens podem saltar sem efectuar captura. Quando um homem atinge a última linha, ocorre a libertação de outro homem.

Objectivo

O objectivo do jogo é, tal como nas Damas, **capturar todas as peças** do oponente, saltando sobre elas, ou **incapacitar o adversário** de realizar qualquer movimento.

Jogada

Em cada jogada, um jogador pode fazer uma de duas acções: **mover ou capturar**.

Movimento

Uma peça pode mover-se em três direcções: para a frente ou na diagonal (**norte, nordeste ou noroeste**). Numa jogada, o movimento nunca pode ser efectuado para a retaguarda.

Existem dois tipos de movimentos: **Normal e Salto**.

- **Movimento Normal**: uma peça move-se para uma **casa adjacente e vazia**.
- **Movimento em Salto**: **uma peça salta sobre uma peça aliada adjacente, se e só se a casa correspondente** (ao lado da peça aliada) **estiver vazia**, colocando assim a peça nessa casa. Se a mesma peça do jogador puder continuar a realizar o mesmo movimento de salto sobre outra peça amigável então terá de o fazer. **Durante um movimento de salto a peça não pode capturar peças inimigas**.

  Quando existe **mais do que uma forma de saltar**, o jogador **pode escolher a peça que irá usar para executar o salto**, bem como o tipo de salto ou sequência de saltos a fazer. Não é obrigatório que a sequência de saltos escolhida pelo jogador seja aquela que possui o maior número de saltos; porém,

após escolher uma sequência, o jogador deve executar todos os saltos possíveis.

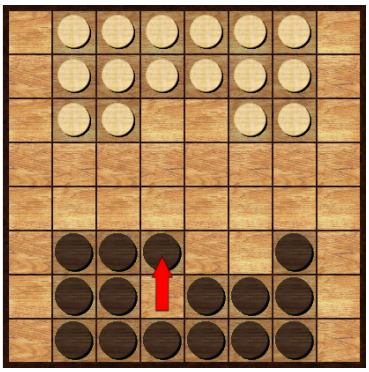

Figura 4: Movimento Normal

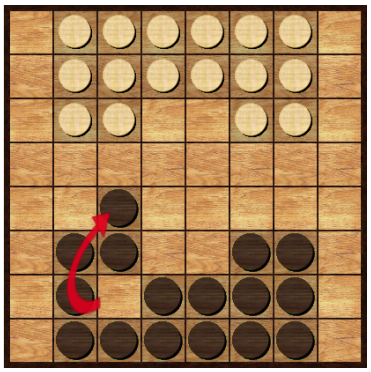

Figura 5: Movimento em Salto

Figura 6: Movimentos

Captura

Um **jogador pode capturar** em cinco direcções: **frente**, **diagonal para a frente**, **direita** ou **esquerda** (norte, nordeste, noroeste, este ou oeste).

- **Captura**: um jogador **salta sobre uma peça adjacente do adversário**, se a **próxima casa**, na mesma direcção, **estiver vazia**, colocando, assim, a peça sobre essa casa. A **peça do oponente é removida do tabuleiro**. Se a **peça** do mesmo jogador **puder continuar a capturar outras peças do adversário**, então **deve fazê-lo**. A **captura é obrigatória** e deve continuar enquanto for possível.

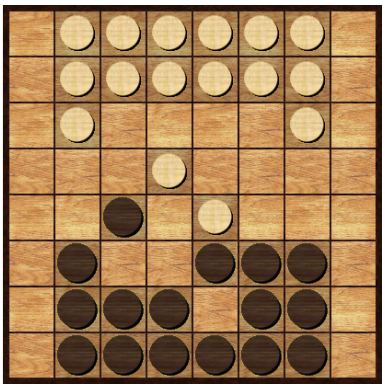

Figura 7: Estado anterior à captura

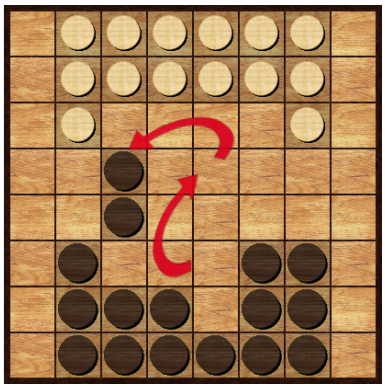

Figura 8: Estado posterior à captura

Tal como no **Movimento de Salto**, **o jogador escolhe** livremente **qual a sequência de saltos a efectuar**.

Última Linha

Quando uma peça atinge a extremidade do tabuleiro, essa peça é removida de imediato e o jogador recebe dois movimentos extra para efectuar nesse mesmo momento:

colocar duas peças novas numa casa vazia localizada nas duas primeiras linhas, à excepção das quatro casas laterais (duas do lado esquerdo, e duas do lado direito).

2 Representação do Estado do Jogo

Representação do estado inicial do tabuleiro:

```
[[empty, white, white, white, white, white, white, empty],
 [empty, white, white, white, white, white, white, empty],
 [empty, white, white, empty, empty, white, white, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, black, black, empty, empty, black, black, empty],
 [empty, black, black, black, black, black, black, empty],
 [empty, black, black, black, black, black, black, empty]]
```

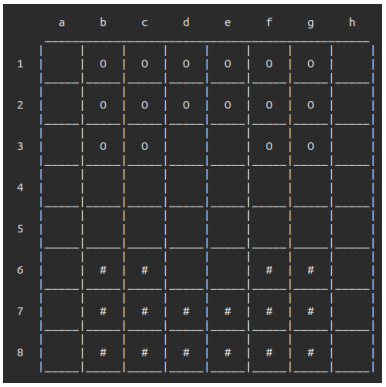

Figura 9: Estado inicial do tabuleiro visualizado na consola

Representação de um possível estado intermédio do tabuleiro:

```
[[empty, white, white, white, white, white, white, empty],
 [empty, white, white, empty, white, white, white, empty],
 [empty, white, white, white, empty, white, white, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, black, empty, empty, empty, empty, empty],
 [empty, black, black, black, empty, black, black, empty],
 [empty, black, empty, empty, black, black, black, empty],
 [empty, black, black, black, black, black, black, empty]]
```

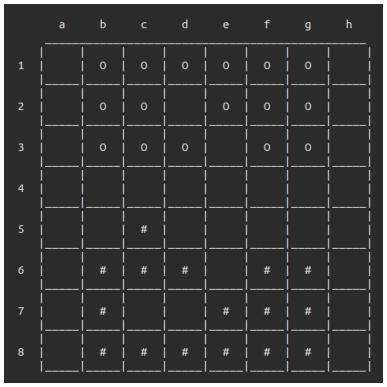

Figura 10: Estado intermédio do tabuleiro visualizado na consola

Representação de um possível estado final do tabuleiro:

```
[[empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, empty, white, empty, empty, empty, empty],
 [empty, empty, black, black, black, empty, empty, empty]]
```

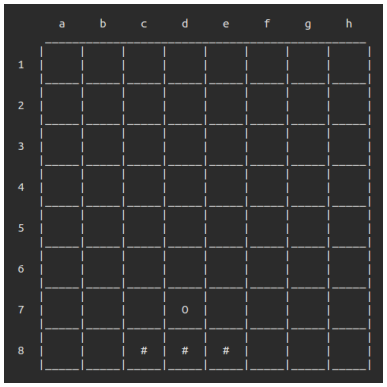

Figura 11: Estado final do tabuleiro visualizado na consola

3 Visualização do Tabuleiro

De seguida está apresentado o código que, em princípio, será usado para mostrar o tabuleiro na consola.

```
%=====================%
%= Board drawing =%
%=====================%
printColumnIdentifiers:-
        write('        a      b      c      d      e      f      g
h').

rowIdentifiersList([' 1 ', ' 2 ', ' 3 ', ' 4 ', ' 5 ', '
6 ', ' 7 ', ' 8 ']).

printInitialSeparator:-
        write('      -------------------------------------------- ').

createSeparatorN(0, _, []).
createSeparatorN(N, SS, [SS | Ls]):-
        N1 is N-1,
        createSeparatorN(N1, SS, Ls).

printBoardRowValues([]).
printBoardRowValues([Head | Tail]):-
        cell(Head, Piece),
        write('  '), write(Piece), write('  |'),
        printBoardRowValues(Tail).

printBoardRow([], []).
printBoardRow(Line, RowIdentifiersListHead):-
        length(Line, Length),
        createSeparatorN(Length, '-----|', Separator),
        createSeparatorN(Length, '     |', Separator2),
        write('      '), write('|'), printList(Separator2), nl,
        write(RowIdentifiersListHead), write('|'), printBoardRowValues(Line), nl,
        write('      '), write('|'), printList(Separator), nl.

printRemainingBoard([], []).
printRemainingBoard([Line | Tail],
        [RowIdentifiersListHead | RowIdentifiersListTail]):-
        printBoardRow(Line, RowIdentifiersListHead),
        printRemainingBoard(Tail, RowIdentifiersListTail).

printBoard([Line | Tail]):-
        printColumnIdentifiers, nl,
        printInitialSeparator, nl,
        rowIdentifiersList(RowIdentifiers),
        printRemainingBoard([Line | Tail], RowIdentifiers), nl.
```

A figura 9, 10 e 11 - apresentadas anteriormente - representam os outputs esperados para a consola do estado inicial, intermédio e final do tabuleiro, respectivamente, produzidos pelo código acima apresentado.

4 Movimentos

Cabeçalho do predicado de movimentação de uma peça:
movePiece(Row, Column, DestRow, DestColumn, Board)

Cabeçalho do predicado de captura de uma peça:
capturePiece(Row, Column, Board)