Ni-ju

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Niju_2:
João Pedro Furriel de Moura Pinheiro - up201104913
Ventura de Sousa Pereira - up201404690

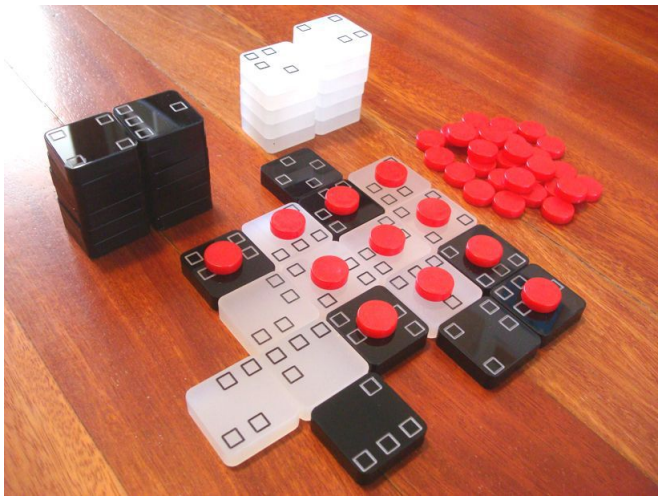
Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

15 de Outubro de 2017

1 O Jogo Ni-ju

Ni-ju (20, em japonês) é um jogo desenvolvido pelo designer e artista Néstor Romeral Andrés, em 2016, tendo sido publicado pela HenMar Games, nestorgames. O jogo é constituido por 40 peças, sendo estas divididas por cor - branca e preta, entre os jogadores, ficando, cada um, com 20 peças de padrões diferentes (daí o nome do jogo). Cada peça tem um padrão desenhado com pontos. Cada jogador deverá colocar uma peça na zona de jogo, seguido pelo seu adversário. Antes de jogar uma peça, o jogador pode rodar a mesma 90, 180 ou 270 graus para, desta forma, rodar o padrão que a peça representa. O objetivo do jogo é recriar o padrão descrito na peça, à volta da mesma. No final ganha quem conseguir recriar mais padrões, quando se esgotarem as peças. Nesta versão que desenvolveremos, o jogo termina quando todas as peças forem colocadas na mesa. Nesta versão,os discos de ajuda que são visíveis na imagem também não serão utilizados.

Regras: A única regra do jogo prende-se com a colocação das peças. Uma peça apenas poderá ser colocada numa posição adjacente (em cima, por baixo, à esquerda ou à direita) a outra peça que já esteja colocada na mesma.

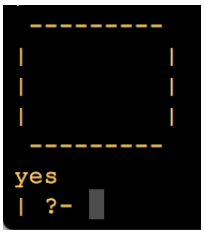


2 Representação do Estado do Jogo

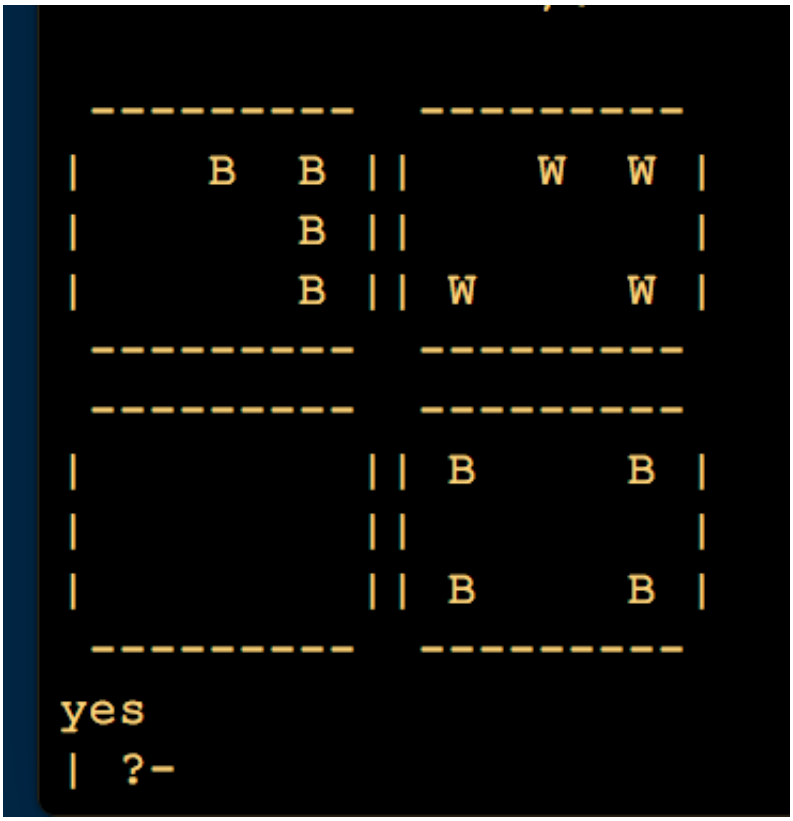
Inicialmente, o estado inicial do jogo será uma lista que represente um espaço vazio, visto que este é dinâmico. Modifica-se conforme a posição de cada peça jogada.

Estado Inicial:

[[[[-1, -1, -1], [-1, -1, -1], [-1, -1, -1]]]]

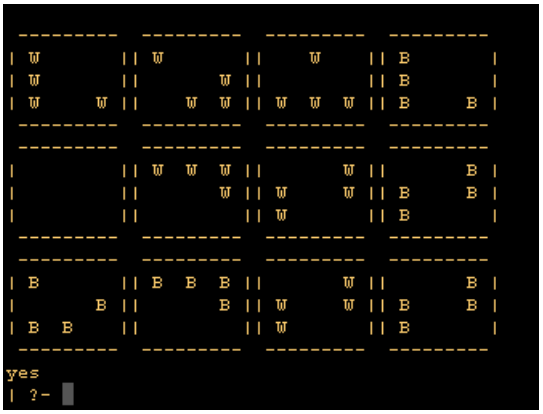


Num estado intermédio, o jogo terá algumas peças em jogo.
Possível representação:
[[
[[0, 1, 1], [0, 0, 1], [0, 0, 1]],
[[0, 2, 2], [0, 0, 0], [2, 0, 2]]
],
[
[[-1, -1, -1], [-1, -1, -1], [-1, -1, -1]],
[[1, 0, 1], [0, 0, 0], [1, 0, 1]]
]
]



O estado final é quando todas as peças já foram colocadas. Num hipotético jogo apenas com 11 peças, este seria um possível estado final.
Representação final:

```
[
[
[[2, 0, 0], [2, 0, 0], [2, 0, 2] ],
[[2, 0, 0], [0, 0, 2], [0, 2, 2] ],
[[0, 2, 0], [0, 0, 0], [2, 2, 2] ],
[[1, 0, 0], [1, 0, 0], [1, 0, 1] ]
],
[
[[-1, -1, -1], [-1, -1, -1], [-1, -1, -1] ],
[[2, 2, 2], [0, 0, 2], [0, 0, 0] ],
[[0, 0, 2], [2, 0, 2], [2, 0, 0] ],
[[0, 0, 1], [1, 0, 1], [1, 0, 0] ]
],
[
[[1, 0, 0], [0, 0, 1], [1, 1, 0] ],
[[1, 1, 1], [0, 0, 1], [0, 0, 0] ],
[[0, 0, 2], [2, 0, 2], [2, 0, 0] ],
[[0, 0, 1], [1, 0, 1], [1, 0, 0] ]
]
]
```



3 Visualização do Tabuleiro

O padrão de cada peça é representado por B's ou W's, dependendo se a peça é preta ou branca, respetivamente. Os espaços vazios no tabuleiro são representados por uma matriz, de 3x3, de espaços. Além disso, os espaços vazios da peça são, também, representados por espaços.

Predicados usados para a impressão do tabuleiro:

```
printElement(X) :- X =:= 1, write(' B ').
printElement(X) :- X =:= 2, write(' W ').
printElement(X) :- X =:= -1, write('   ').
printElement(X) :- X =:= 0, write('   ').
printPieceRowSeparation(BoardRow) :-
```

```
        BoardRow = [].

printPieceRowSeparation(BoardRow) :-

        BoardRow \= [],
        [\_ | Rest] = BoardRow,
        write(' ---------- '),
        printPieceRowSeparation(Rest).

printPieceRow([X1,X2,X3]) :-

        write('|'),
        printElement(X1),
        printElement(X2),
        printElement(X3),
        write('|').


printBoard(Board) :-

        Board = [].

printBoard(Board) :-

        [Row | Rest] = Board,
        printPieceRowSeparation(Row),
        nl,
        printPiecesRow1(Row),
        nl,
        printPiecesRow2(Row),
        nl,
        printPiecesRow3(Row),
        nl,
        printPieceRowSeparation(Row),
        nl,
        printBoard(Rest).

printPiecesRow1(BoardRow) :-

        BoardRow = [].

printPiecesRow1(BoardRow) :-

        [Piece | Rest] = BoardRow,
        [PieceRow1 | \_] = Piece,
        printPieceRow(PieceRow1),
        printPiecesRow1(Rest).


printPiecesRow2(BoardRow) :-
        BoardRow = [].
```

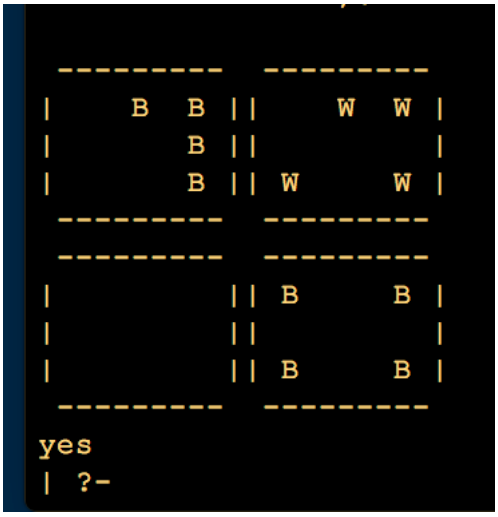
```
printPiecesRow2(BoardRow) :-
        [Piece | Rest] = BoardRow,
        [\_,PieceRow2,\_] = Piece,
        printPieceRow(PieceRow2),
        printPiecesRow2(Rest).


printPiecesRow3(BoardRow) :-

        BoardRow = [].

printPiecesRow3(BoardRow) :-

        [Piece | Rest] = BoardRow,
        [\_,\_,PieceRow3] = Piece,
        printPieceRow(PieceRow3),
        printPiecesRow3(Rest).
```



4 Movimentos

```
playPieceRight(CurrentBoard, Piece, Rotation, Row, Column,
    ResultingBoard).
playPieceDown(CurrentBoard, Piece, Rotation, Row, Column,
    ResultingBoard).
playPieceLeft(CurrentBoard, Piece, Rotation, Row, Column,
    ResultingBoard).
playPieceUp(CurrentBoard, Piece, Rotation, Row, Column,
    ResultingBoard).
```

Os parâmetros dos predicados das jogadas são:

- CurrentBoard: Estado atual do jogo.
- Piece: A peça a ser jogada.

- Rotation: A rotação da peça (0, 90, 180 ou 270 graus).
- Row: A linha onde será colocada a peça.
- Column: A coluna onde será colocada a peça.
- ResultingBoard: O estado do jogo após a colocação da peça.

Antes de efectivamente colocar a peça na mesa, serão desenvolvidos predicados para a validação da jogada, isto é, para garantir que a regra da colocação da peça seja cumprida, e apenas nesse caso, a peça seja colocada na mesa.