Jogo de Tabuleiro - Campo Bello

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

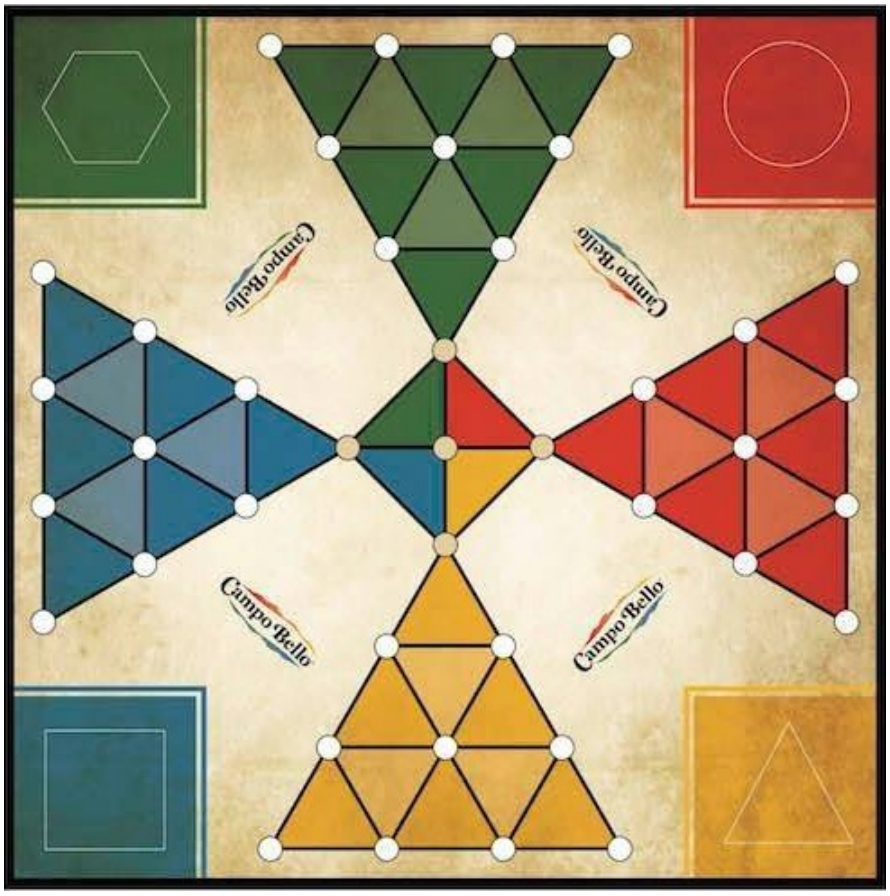
Grupo 04:
João Nuno Fonseca Seixas - 201505648
Renato Alexandre Sousa Campos - 201504942

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

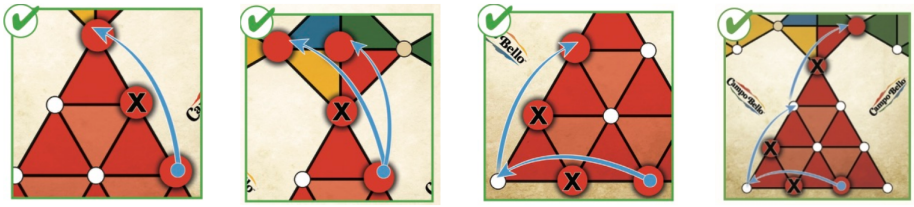
14 de Outubro de 2017

1 O Jogo Campo Bello

Campo Bello é um jogo que pode ser jogado de 2 a 4 jogadores. Tem inspiração no jogo clássico "Resta Um"ou "Peg Solitaire"em Inglês. É um jogo ainda recente, criado em 2017. Para ganhar, um jogador deve tentar remover todas as suas peças do tabuleiro antes dos adversários. O tabuleiro consiste em 4 triângulos que rodeiam um diamante central. Os triângulos correspodem às áreas iniciais de cada jogador. As peças são removidas ao saltar: saltar uma peça nossa causa a remoção da peça que foi saltada; saltar uma peça adversária permite-nos remover uma peça nossa do tabuleiro. Na variante de apenas 2 jogadores, que vai ser implementada, os jogadores ficam com triângulos opostos e jogam alternadamente. É também possível executar saltos duplos e triplos numa só jogada. No fim do jogo cada jogador pontua 1 ponto por cada uma das suas peças fora da área inicial e 3 pontos por cada uma das suas peças dentro da sua área inicial. O jogador com menos pontos ganha!



LEGAL MOVES



single jump | single jump | double jump | triple jump

ILLEGAL MOVES

end where you began | sharp corner jump | jump the same piece

http://www.campobellogame.com/

2 Representação do Estado do Jogo

O estado do jogo é representado por 2 listas que identificam em que posições do tabuleiro estão as peças de cada jogador (1 lista para cada jogador). As peças de cada jogador sáo designadas em Inglês por "movers".

```
blueMoversInitialPos([r1,r2,r3,r4,r5,r6,r7,r8,r9,b1,b2,b3,b4,b5,b6,b7,b8,b9]).
yellowMoversInitialPos([y1,y2,y3,y4,y5,y6,y7,y8,y9,g1,g2,g3,g4,g5,g6,g7,g8,g9]).

blueMoversMidGamePos([y1,r1,r5,r6,r7,r8,r9,b1,b5,b6,b7,mid]).
yellowMoversMidGamePos([r3,y2,y4,y5,y8,y9,g1,g4,g7,g8,g9]).

blueMoversFinalPos([]).
yellowMoversFinalPos([y2,y8,y9,g1,g9]).
```

Imagens do tabuleiro nos respetivos estados de jogo (inicial, intermédio e final):

```
| ?- yellowMoversInitialPos(Y),blueMoversInitialPos(B),displayBoard(Y,B).
   y y y y
b   y y y   b
 b   y y   b
b b   e   b b
 b  e e e  b
b b   e   b b
 b   y y   b
b   y y y   b
   y y y y
```

```
| ?- blueMoversMidGamePos(B),yellowMoversMidGamePos(Y),displayBoard(Y,B).
   y y y e
b   e y e   b
 e   e y   b
b b   e   e b
 e  e b e  e
e e   e   b b
 b   y e   b
e   e y y   b
   e e y y
```

```
| ?- blueMoversFinalPos(B),yellowMoversFinalPos(Y),displayBoard(Y,B).
   y e e e
e   e e e   e
 e   e y   e
e e   e   e e
 e  e e e  e
e e   e   e e
 e   y e   e
e   e e e   e
   e e y y
```

Com o decorrer do jogo as listas perdem elementos. O estado final do jogo é alcançado quando uma das listas estiver vazia ou quando não houver mais jogadas possíveis para os 2 jogadores.

3 Visualização do Tabuleiro

O predicado para visualização do tabuleiro é muito específico, visto que cada linha tem de ser impressa de maneira diferente.

```
| ?- yellowMoversPos(Y),blueMoversPos(B),displayBoard(Y,B).
   y y y y
b   y y y   b
 b   y y   b
b b   e   b b
 b  e e e  b
b b   e   b b
 b   y y   b
b   y y y   b
   y y y y
```

```
displaySingleP(P, PiecesY, _) :- member(P,PiecesY), write(y), write(' ').
displaySingleP(P, _, PiecesB) :- member(P,PiecesB), write(b), write(' ').
displaySingleP(_,_,_) :- write(e), write(' ').
displayPos([H|T],PiecesY,PiecesB) :- displaySingleP(H,PiecesY,PiecesB),
                                     displayPos(T,PiecesY,PiecesB).
displayPos([],_,_).

displayLine(1,PiecesY,PiecesB) :- write('   '),displayPos([g9,g8,g7,g6],PiecesY,PiecesB),nl.

displayLine(2,PiecesY,PiecesB) :- displaySingleP(b6,PiecesY,PiecesB), write('  '),
                                  displayPos([g5,g4,g3],PiecesY,PiecesB), write('  '),
                                  displaySingleP(r9,PiecesY,PiecesB),nl.

displayLine(3,PiecesY,PiecesB) :- write(' '), displaySingleP(b3,PiecesY,PiecesB), write('  '),
                                  displayPos([g2,g1],PiecesY,PiecesB), write('  '),
                                  displaySingleP(r5,PiecesY,PiecesB),nl.

displayLine(4,PiecesY,PiecesB) :- displayPos([b7,b1],PiecesY,PiecesB), write('  '),
                                  displaySingleP(g0,PiecesY,PiecesB), write('  '),
                                  displayPos([r2,r8],PiecesY,PiecesB), nl.

displayLine(5,PiecesY,PiecesB) :- write(' '), displaySingleP(b4,PiecesY,PiecesB),
                                  write(' '), displayPos([b0,mid,r0],PiecesY,PiecesB),
                                  write(' '),displaySingleP(r4,PiecesY,PiecesB),nl.

displayLine(6,PiecesY,PiecesB) :- displayPos([b8,b2],PiecesY,PiecesB), write('  '),
                                  displaySingleP(y0,PiecesY,PiecesB), write('  '),
                                  displayPos([r1,r7],PiecesY,PiecesB), nl.

displayLine(7,PiecesY,PiecesB) :- write(' '), displaySingleP(b5,PiecesY,PiecesB),
                                  write('  '), displayPos([g1,g2],PiecesY,PiecesB),
                                  write('  '), displaySingleP(r6,PiecesY,PiecesB),nl.

displayLine(8,PiecesY,PiecesB) :- displaySingleP(b9,PiecesY,PiecesB), write('  '),
                                  displayPos([y3,y4,y5],PiecesY,PiecesB),
                                  write('  '), displaySingleP(r6,PiecesY,PiecesB),nl.

displayLine(9,PiecesY,PiecesB) :- write('   '),displayPos([y6,y7,y8,y9],PiecesY,PiecesB),nl.
displayBoard(Y,B) :-
        displayLine(1,Y,B),
        displayLine(2,Y,B),
        displayLine(3,Y,B),
        displayLine(4,Y,B),
        displayLine(5,Y,B),
        displayLine(6,Y,B),
        displayLine(7,Y,B),
        displayLine(8,Y,B),
        displayLine(9,Y,B).
```

4 Movimentos

Elencar os movimentos (tipos de jogadas) possíveis e definir os cabeçalhos dos predicados que serão utilizados (ainda não precisam de estar implementados).