Campo Bello

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo *Campo Bello 3*:
José Pedro Dias de Almeida Machado - 201504779
Maria Francisca Azevedo Paupério - 201403785

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

15 de Outubro de 2017

1 O Jogo *Campo Bello*

Ao longo do relatório está descrita uma versão para 2 jogadores, pois só iremos desenvolver o jogo em *Prolog* para 2 jogadores, tendo em conta que a cada jogador são alocados dois triângulos opostos, para evitar que cada jogador faça duas jogadas consecutivas.

1.1 História

O *Campo Bello* é um jogo de tabuleiro baseado no jogo tradicional *Peg Solitaire*, mas que deve ser jogado por 2 a 4 jogadores, em simultâneo.

Figura 1: Tabuleiro de jogo.

1.2 Objetivo

No *Campo Bello* cada jogador tem de remover o maior número de peças suas, tentando terminar com um menor númedo peças do que o adversário.

1.3 Detalhes

O tabuleiro de jogo consiste em 4 triângulos de cores diferentes em torno dum diamante, como está ilustrado na figura 1. O jogo é inicializado com 9 peças por triângulo (figura 2). Cada peça tem a cor do triângulo respetivo (total de 36 peças).

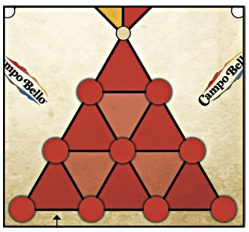

Figura 2: Detalhe da área de jogo vermelha ilustrando como são dispostas as 9 peças iniciais.

1.4 Procedimento

No início do jogo, cada jogador fica alocado a dois triângulos. Como foi dito anteriormente, vamos optar sempre por alocar dois triângulos opostos. Deste modo, cada jogador joga na cor que lhe corresponde nessa jogada (alternando entre as duas que lhe correspondem). Para mover uma peça, o jogador tem que, obrigatoriamente, eliminar outra, quer seja dele ou do adversário. Ao transpor uma peça sua, o jogador remove-a, se transpuser uma peça do adversário pode remover qualquer peça sua da cor que lhe corresponde nessa jogada. O jogo termina, quando nenhuma peça se puder mover, sendo que, para se mover, tem de saltar por cima de uma peça adjacente. Tendo em conta que cada peça vale 1 ponto dentro da sua área de jogo e 3 pontos na área de jogo do adversário, ganha o jogador que somar menos pontos.

1.5 Regras

A principal regra do jogo é mover uma peça se e só se alguma das posições adjacentes estiver ocupada e a posição depois dessa, no mesmo alinhamento, estiver livre. São permitidos saltos duplos e saltos triplos (figura 3). Não é permitido acabar a jogada na posição em que se iniciou (figura 4).

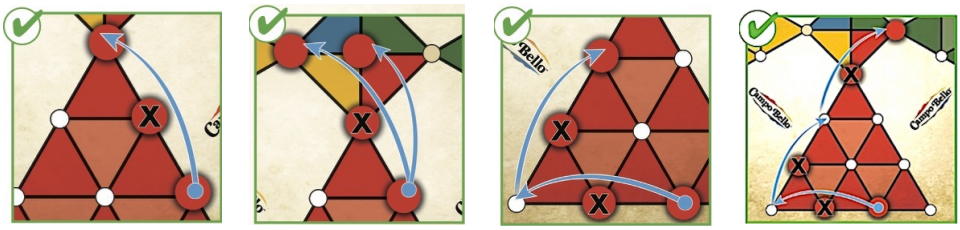

Figura 3: Jogadas permitidas.

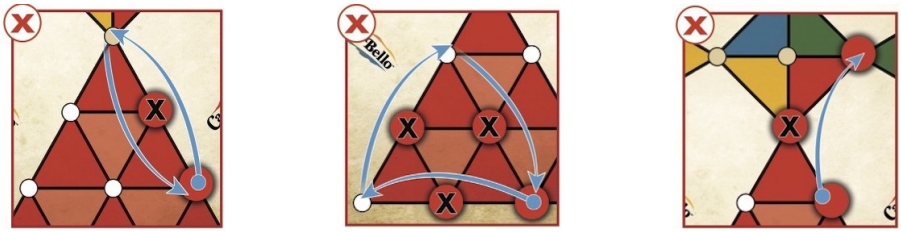

Figura 4: Jogadas proibidas.

2 Representação do Estado do Jogo

O jogo tem um formato de tabuleiro irregular e as cores, impossíveis de representar na consola, são um ponto muito importante no decorrer do jogo. Assim sendo, é difícil indexar cada posição através de linhas e colunas e distinguir a cor das peças. Deste modo, optamos por representar cada posição por um número e cada peça pela letra inicial da cor que tem seguida de um número.

```
%PIECES
%Blue pieces
symbol(p1,'B1').
symbol(p2,'B2').
%      ...
symbol(p9,'B9').

%Yellow pieces
symbol(p11,'Y1').
%      ...
symbol(p19,'Y9').

%Green pieces
symbol(p21,'G1').
%      ...
symbol(p29,'G9').

%Red pieces
symbol(p31,'R1').
%      ...
symbol(p39,'R9').
```

```
%POSITIONS
symbol(1, '01').
symbol(2, '02').
%      ...
symbol(38, '38').
symbol(39, '39').
symbol(40, '40').
symbol(50, '50'). %central position

%INITIALIZE BOARD
initialize_board(Board):-
  Board=[[p1,p2,p3,p4],
         [p5,p6,p7],
         [p8,p9],
         [p11,p21],
         [p15,10,p25],
         [p12,p18,p28,p22],
         [p16,20,50,30,p26],
         [p13,p19,p29,p23],
         [p17,40,p27],
         [p14,p24],
         [p38,p39],
         [p35,p36,p37],
         [p31,p32,p33,p34]].
```

O código acima é responsável por criar uma lista de listas. Cada lista representa uma linha onde é guardada a informação de cada posição do tabuleiro. O tabuleiro é inicializado com a formação inicial do jogo (9 peças por triângulo).

3 Visualização do Tabuleiro

As cores to tabuleiro estão representadas com a inicial do nome da cor em Inglês sendo que as peças de cada cor estão numeradas, as posições que não têm nenhuma peça contém um número que corresponde ao número da célula (de 1 a 40 e a posiçao central com o número 50).

A função `display_board` recebe o tabuleiro a representar, chamando a funçao que desenha a primeira linha e esta, por sua vez, chama a que desenha a segunda linha e assim sucessivamente. A cada linha tem que corresponder uma função diferente devido à forma irregular do tabuleiro, cada linha tem um número de elementos diferente e diferentes espaçamentos entre as posições.

Utilizamos como exemplo, o código abaixo, responsável por imprimir, na consola, a linha 7 do tabuleiro. O resultado da impressão de todas as linhas é demonstrado na figura 5.

```
display_line_7([S|E]):-
  write('|_________'),
  display_elems_line_7(S,E).
display_elems_line_7([S|E],T):-
  symbol(S,Val),
  write(Val),
  write('_________'),
  display_elems_line_7_1(E, T).
display_elems_line_7_1([S|E],T):-
  symbol(S,Val),
  write(Val),
  write('__'),
  display_elems_line_7_2(E, T).
display_elems_line_7_2([S|E],T):-
  symbol(S,Val),
  write(Val),
  write('__'),
  display_elems_line_7(E, T).
display_elems_line_7_2([], T):-
  write('________|'),
    nl,
    display_line_8(T).
```

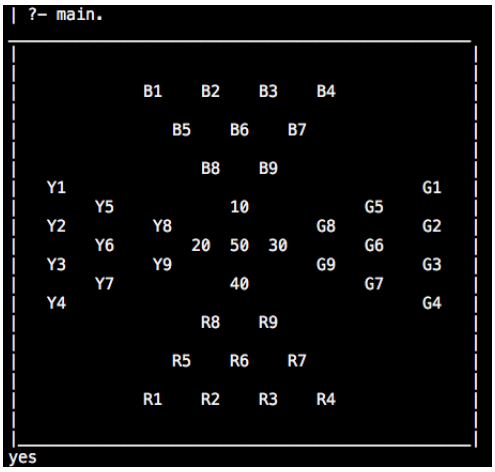

Figura 5: Visualização do tabuleiro com a representação inicial.

4 Movimentos

O jogador pode deslocar qualquer peça da cor que lhe compete jogar nessa jogada se a jogada for válida, ou seja, para uma posição que esteja livre e com a condição de transpor alguma peça, dele ou do adversário. Podendo eliminar mais do que uma peça por jogada desde que seja uma sequência de saltos válidos. A fim de validar a jogada, estamos a desenvolver três funções que estão na base da lógica dos movimentos:

- `move_piece(Piece, Position)` - move uma peça para uma determinada posição especificada se esta for válida e se apresenta uma peça sobre a qual jogar para a remover,para se não apresentará ao ultilizador uma mensagem de erro.

- `remove_piece(Piece)` - remove do tabuleiro uma determinada peça que seja necessária remover segundo as regras do jogo.

- `check_pos_adjacent(Position)` - verifica se existem peças nas posições adjacentes à posição da peça com a qual se pretende jogar.

5 Bibliografia

Todas as imagens e texto sobre o jogo presentes neste relatório foram retiradas do site oficial do jogo Campo Bello à excepção da figura 5 que é uma captura do tabuleiro impresso no terminal:

- http://www.campobellogame.com