Q!nto

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Q!nto_1:
João Miguel Fidalgo Esteves Nogueira - up201303882
Luís Miguel da Costa Oliveira - up201304515

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

6 de Outubro de 2015

1 Descrição detalhada do Jogo

1.1 História

"Q!nto" é um jogo de tabuleiro recente, criado no ano de 2014 no qual podem participar entre 2 e 4 jogadores. Foi criado por Gene Mackles e lançado pela PDG games

Este jogo faz parte de uma coleção de três jogos chamada Triple Play que é uma coleção de jogos de tabuleiro.

1.2 Conteúdo do Jogo

Estão incluídas no Jogo 60 cartas divididas como mostra a figura à direita.

Figura 1: Baralho

1.3 Regras de Jogo

Existem três versões para jogar o Q!nto, sendo elas o *Classic*, *Plus* e *Light*. Neste projeto usaremos apenas a versão Light que passamos a explicar a seguir:

Inicialmente cada jogador retira uma carta do baralho para escolher quem começa a jogar. O jogador cuja carta tem o naipe mais alto começa, sendo que a hierarquia é a definida na imagem acima (de cima para baixo). O jogo começa dividindo igualmente o baralho pelos jogadores presentes.

As cartas que cada jogador recebeu serão a pilha de cartas com que ele vai poder jogar. Estas ficarão viradas para baixo e cada jogador retira da pilha de cartas que lhe foi atribuída as primeiras 5.

Noção base: Uma linha são 2, 3, 4 ou 5 cartas numa linha ou coluna na qual as cartas têm a mesma côr ou cores diferentes e na qual as cartas têm a mesma forma ou formas diferentes. Uma linha com 5 cartas é um Q!nto. O primeiro jogador começa o jogo. A seguir joga o jogador à sua esquerda. Quando na vez de um jogador, este pode fazer uma de três coisas:

1. Adicionar 1, 2, 3, 4 ou 5 peças à grelha numa única linha reta e depois retirar do topo da sua pilha de cartas o número necesário para voltar a ficar com 5 na mão*. Se, numa das jogadas, o jogador completar um ou mais Q!ntos, este deve voltar a completar a sua mão* e deve jogar outra vez.

2. Trocar uma ou mais cartas da sua mão por wildcards já jogadas e/ou a carta Q!nto de forma a que as cartas trocadas continuem a funcionar naquela posição.

3. Passar a vez... e trocar algumas, todas ou nenhuma das susas cartas colocando-as no fundo da sua pilha e retirando do topo o número de cartas necessário para voltar a ficar com 5*.

*ou menos, se não houver cartas suficientes para perfazer 5.

O jogo termina quando houver 1 jogador que fica sem cartas, sendo este o vencedor.

Nota: Uma carta "wild" ou "Q!nto" pode representar cartas diferentes em diferentes direções. No caso de representar coisas diferentes e não ter na mão uma carta que também as consiga representar, então, essa carta é insubstituível.

2 Representação do estado do Jogo

O *Q!nto*, embora não seja um jogo de tabuleiro, basea-se na posição relativa das cartas na mesa. Por esse motivo optámos por guardar a informação do jogo numa lista de listas. Esta lista de listas será representada no ecrã tal como um tabuleiro.

2.1 Código da estrutura de dados da mesa

```
%%%%%%%%%%%%%%%%%%%%%%% Create empty table %%%%%%%%%%%%%%%%%%%%%%%%%

createEmptyLine(A, M, M).                  % Stops when the list reaches
createEmptyLine([A|B], N, M) :-            % the max number of elements
        N < M,                             %
        N1 is N + 1,                       % Recursively adds an emty elem
        A = ' ',                           % to the list (line)
        createEmptyLine(B, N1, M).

createEmptyBoard([BoardHead | BoardTail], M, M).
createEmptyBoard([BoardHead | BoardTail], N, M) :-
        N < M,              % Stops when the list reaches
        N1 is N + 1,        % the max number of elements
        createEmptyLine(BoardHead, 0, M),
        createEmptyBoard(BoardTail, N1, M).

% Recursively adds an empty elem to the list (list of lines)
```

O código acima é responsável por criar uma lista de listas que representam uma lista de linhas para guardar a informação do conteúdo da mesa. Como é suposto no início do jogo, a mesa é inicializada como vazia.

2.2 Código de representação do Ambiente de jogo

```
%#####################print all cells#####################%

printAllCells([A|[]], S, I) :-  % When the lists head is the
        I < S,                  % last line
        I1 is I + 1,            %
        startPrintLine(A, I1),  % Print the line
        nl,                     %
        startPrintDivisor(1, S),% prints line between lines
        nl.                     %

printAllCells([A|B], S, I) :-   % Normally recursively called
        I < S,                  % Iterator check
        I1 is I + 1,            % Iterator incrementation
        startPrintLine(A, I1),  % Prints index of the line
        nl,                     %
        startPrintDivisor(1, S),%
        nl,                     %
        printAllCells(B, S, I1).% Recursive Call

printAllCells(_,_,_).           %

%#####################printBoard#########################%

printBoard(A) :-                        % Initial Print
        createEmptyBoard(Board,0, A),   % Creates Empty Board
        printIdsOfColumns(A),           % Prints ID's
        nl,                             %
        printAllCells(Board, A, 0).     % Prints the board's content
```

O Código acima é responsável por inicializar a mesa e imprimi-la no ecrã.

No entanto, este não é todo o código responsável pela impressão da lista de listas. Os predicados em falta percorrem as linhas imprimindo cada elemento, tal como se certificam de que a formatação do texto é a mais correta possível por forma a garantir que o utilizador tem uma forma eficiente de perceber o estado atual do jogo.

♠ = 1 b = Black
! = 2 B = Blue
♦ = 3 G = Green
♥ = 4 R = Red
♣ = 5 Y = Yellow
W = Wildcard

Figura 2: Código das Cartas

2.3 Exemplos de Ambientes de jogo

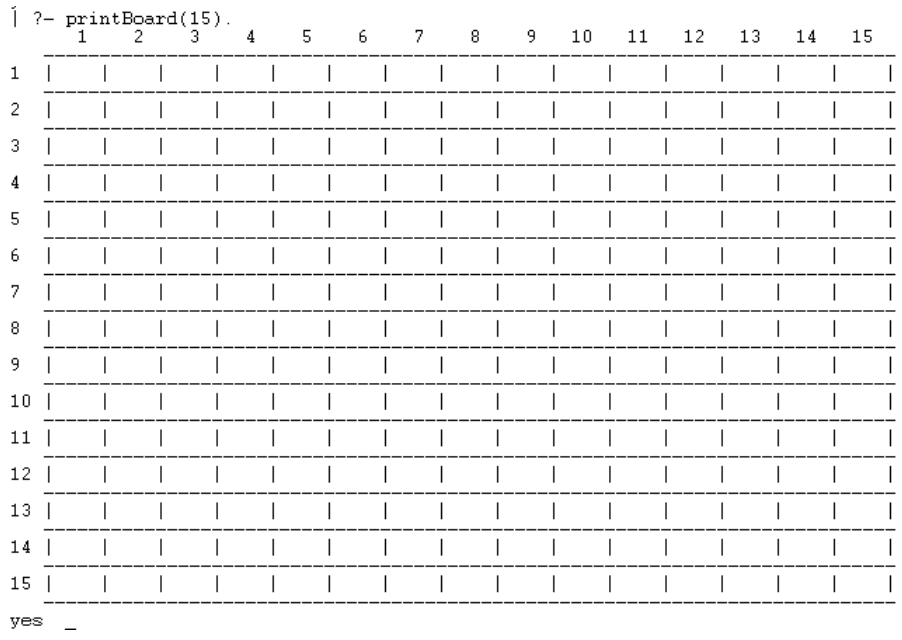

Figura 3: Exemplo de Ambiente Gráfico

Código de estados pré-definidos e respetiva apresentação.

```
createPresetA(L) :-
 L =
  [
   [empty, black_spades, empty, empty, empty],
    [empty, green_spades, green_clubs, green_diamonds, empty],
    [empty, blue_spades, empty, empty, empty],
    [empty, yellow_spades, empty, empty, empty],
    [red_exclamation, red_spades, red_wild, empty, empty]
  ].

 createPresetB(L) :-
  L =
  [
   [empty, empty, empty, green_spades, empty, empty],
    [empty, empty, empty, red_spades, empty, empty],
    [empty, empty, empty, yellow_wild, empty, empty],
    [empty, black_clubs, yellow_exclamation, wild_spades, green_hearts, blue_wild],
    [empty, yellow_hearts, green_exclamation, black_spades, blue_diamonds, red_clubs],
    [empty, blue_exclamation, red_exclamation, empty, empty, empty]
  ].
```

```
    1    2    3    4    5
  ------------------------
1 |    | 1b |    |    |    |
  ------------------------
2 |    | 1G | 5G | 3G |    |
  ------------------------
3 |    | 1B |    |    |    |
  ------------------------
4 |    | 1Y |    |    |    |
  ------------------------
5 | 2R | 1R | WR |    |    |
  ------------------------
```

Figura 4: Exemplo de Ambiente Gráfico - preset 1

```
    1    2    3    4    5    6
  ----------------------------
1 |    |    |    | 1G |    |    |
  ----------------------------
2 |    |    |    | 1R |    |    |
  ----------------------------
3 |    |    |    | WY |    |    |
  ----------------------------
4 |    | 5b | 2Y | 1W | 4G | WB |
  ----------------------------
5 |    | 4Y | 2G | 1b | 3B | 5R |
  ----------------------------
6 |    | 2B | 2R |    |    |    |
  ----------------------------
```

Figura 5: Exemplo de Ambiente Gráfico - preset 2