Shacru

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

PLOG_TP1_RI_Grupo_Shacru4:
Vitor Miguel Saraiva Esteves - 201303104

Faculdade de Engenharia da Universidade do Porto

16 de Outubro de 2016

1 O Jogo Shacru

1.1 Contextualização

Três Jogos: Shacru, Azacru & Pacru
Todos usam as mesmas peças, marcadores e tabuleiro mas são diferentes em estratégia, na dinâmica de jogo e no nível de complexidade. Os jogos não são meras variações mas assumem princípios semelhantes. É mais fácil aprender Azacru se já tiver jogado Shacru. É mais fácil aprender Pacru se já jogou Azacru. Neste caso iremos apenas focar-nos no **Shacru**:

1.2 Componentes do jogo

- 1 Tabuleiro de Pacru, que é dividido em nove sectores. Cada sector contém nove campos. Um campo tem a forma de uma estrela de oito pontas com um círculo interior;
- 16 peças de jogo, 4 de cada cor;
- 200 marcadores (pequenas colunas), 50 de cada cor.
- 1 caixa de cada cor para as respectivas peças e marcadores.

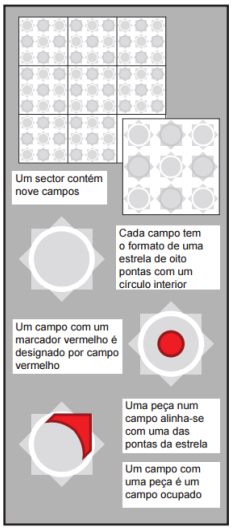

Figura 1: Tabuleiro de Pacru

1.3 Objetivo do jogo

O objetivo deste jogo é controlar o maior número de campos possível, utilizando os marcadores. Cada vez que move uma peça, muda o campo neutro onde a peça terminou o seu movimento, colocando aí um dos seus marcadores.O vencedor do jogo é o jogador com mais marcadores no tabuleiro. É ainda possível usar este número de marcadores de cada jogador como um valor que se acumula durante várias rodadas até que um dos jogadores atinja um valor máximo combinado previamente[1].

1.4 Regras

- Cada jogador é representado por uma cor e usa as pelas e marcadores da sua cor. A ordem das jogadas entre as cores é uma escolha dos participantes que devem decidir entre eles de alguma forma razoável;

[1] http://www.pacru.com/rulesPT.pdf

- Se há 3 ou 4 jogadores, cada jogador começa com três peças. Para dois jogadores, cada começa com quatro peças.Os jogadores, em cada turno, movem uma das suas peças. Só é possível passar o turno se não conseguir mover qualquer uma das suas peças;
- O jogador deve movimentar a peça numa de três direcções: movimentar em frente, ou num ângulo de quarenta e cinco graus (para a esquerda ou direita) em relação à posição actual da peça. A peça termina a jogada a apontar na direcção do movimento;

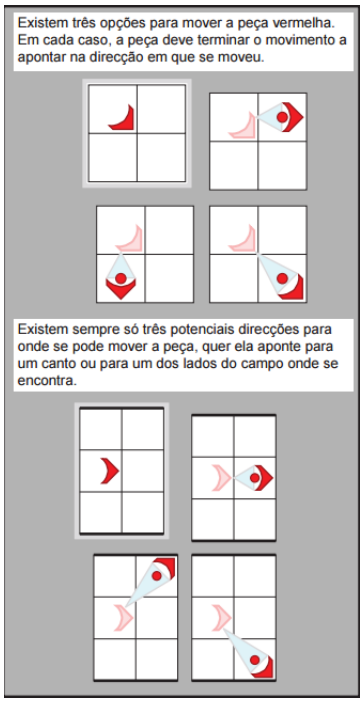

Figura 2: Turnos e Direcções

- Cada vez que move uma peça, muda o campo neutro onde a peça terminou o seu movimento, colocando aí um dos seus marcadores.Quando move uma peça por entre sectores, realiza um movimento normal como foi descrito acima;
- Não é permitido mover para um campo com um marcador da cor do adversário, nem que contenha outra peça.
- Cada marcador colocado corresponde a um ponto;
- O jogo termina se não houver jogadores capazes de mover peças, ou se os movimentos que sobram não sejam capazes de adquirir mais campos.
- O vencedor é aquele que tiver mais campos com marcadores, obtendo assim uma pontuação mais elevada.
- É possível que mesmo que tenha de passar até ao fim do jogo, ainda assim vença o jogo se os outros jogadores não conseguirem mais marcadores que os que você tem.

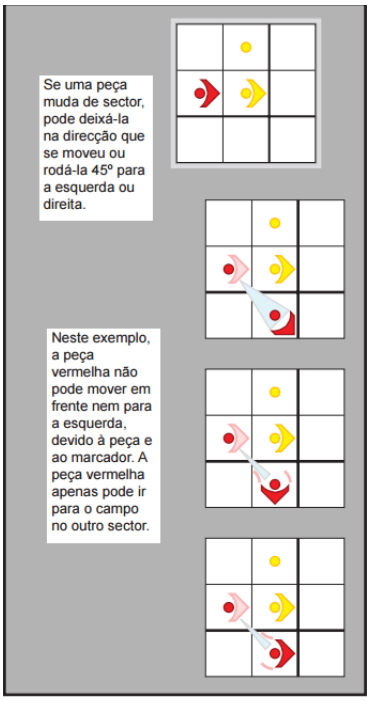

Figura 3: Movimentação e Orientação

2 Representação do Estado do Jogo

O tabuleiro será, a nível de programacão, representado por uma matriz ou, mais especificamente, uma lista de listas que representaram o estado do jogo. A nível de representação gráfica serão usados os seguintes componentes:

- **Representação gŕafica:**
  - espaços em branco = território não ocupado (livre)
  - 0 = marcador de ocupação (centro)
  - : = marcador do jogador 1
  - * = marcador do jogador 2
  - N/S/E/W/etc (Pontos Cardeais) - peças de direcção

A nível de representação lógica, a abordagem sintática escolhida teve em conta as posições possíveis para as várias peças de jogo e foi a seguinte:

- **Representação lógica - Jogador 1:**
  - -1 = livre (vazio)
  - 1 = marcador do jogador 1 (:)
  - 11 = peça do jogador 1 com direcção a NW
  - 12 = peça do jogador 1 com direcção a N
  - 13 = peça do jogador 1 com direcção a NE
  - 14 = peça do jogador 1 com direcção a W
  - 16 = peça do jogador 1 com direcção a E
  - 17 = peça do jogador 1 com direcção a SW
  - 18 = peça do jogador 1 com direcção a S
  - 19 = peça do jogador 1 com direcção a SE
- **Representação lógica - Jogador 2:**
  - -1 = livre (vazio)
  - 2 = marcador do jogador 2 (*)
  - 21 = peça do jogador 1 com direcção a NW
  - 22 = peça do jogador 1 com direcção a N
  - 23 = peça do jogador 1 com direcção a NE
  - 24 = peça do jogador 1 com direcção a W
  - 26 = peça do jogador 1 com direcção a E
  - 27 = peça do jogador 1 com direcção a SW
  - 28 = peça do jogador 1 com direcção a S
  - 29 = peça do jogador 1 com direcção a SE

2.1 Representação de um possível estado inicial do tabuleiro

```
printBoard([
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[1,18],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[2,17],[-1,-1],[-1,-1],
[-1,-1],[1,12],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[2,21],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[2,24],[-1,-1],[-1,-1],[-1,-1],[1,16],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[2,24],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[1,13],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1]
]).
```

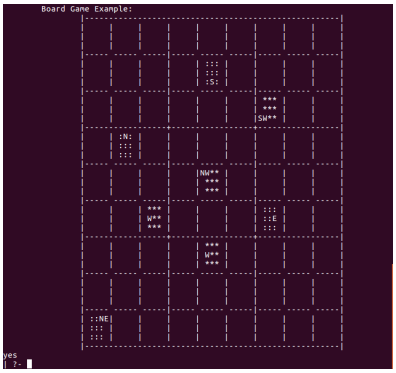

Figura 4: Exemplo de um estado inicial

2.2 Representação de um possível estado intermédio do tabuleiro

```
printBoard([
[0,1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[0,19],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,2],[0,29],[-1,-1],
[-1,-1],[1,18],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,2],[1,17],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,28],[-1,-1],[-1,-1],
[-1,-1],[2,21],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[1,12],[-1,-1],[-1,-1],[-1,-1],[2,24],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1]
]).
```

2.3 Representação de um estado final do tabuleiro

```
printBoard([
[1,11],[0,1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[0,1],[0,1],[0,1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[0,1],[0,1],[0,1],[0,1],[-1,-1],[-1,-1],[0,2],[0,2],[0,26],
[-1,-1],[0,1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,12],[1,12],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,2],[0,1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,1],[0,28],[0,1],[-1,-1],
[0,24],[0,2],[-1,-1],[-1,-1],[-1,-1],[0,1],[0,1],[0,1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,1],[-1,-1],[-1,-1],[0,2],[2,26],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,18],[-1,-1],[-1,-1],[-1,-1],[-1,-1]
]).
```

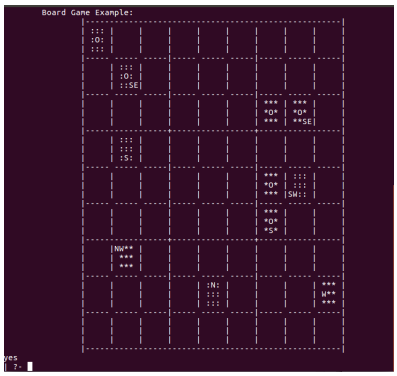

Figura 5: Exemplo de um estado intermédio

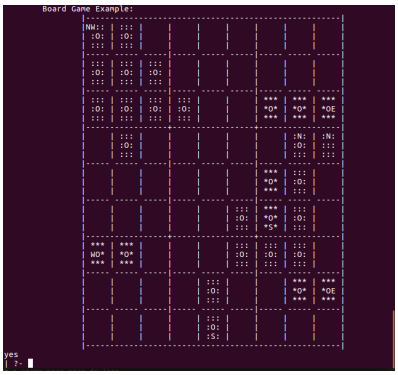

Figura 6: Exemplo de um estado final

3 Visualização do Tabuleiro

De modo a obter a interface gráfica mostrada previamente, foram desenvolvidos um conjunto de protótipos de predicados. Achei assim por bem utilizar os seguintes:

- Uma função inicial **printBoard(Board)** que recebe uma lista das peças (*board*) que constituem o tabuleiro e imprime-o;

Para auxiliar o predicado anterior desenvolveram-se ainda predicados auxiliares:

- **printBoardAux(Board, CurrentNumberHorizontal)**: chamada auxiliar de *printBoard* que permite desenhar o tabuleiro de forma dinâmica;
  - **printTopLine(NumberofDashes)**: imprime o topo do tabuleiro;
  - **printClearCell('_')**: imprime uma linha inteira (espaço vazio e barras laterais);
  - **NormalLineofTiles(NumberofCells)**, predicado auxiliar de printClearCell.Utilizado para chamar recursivamente printClearCell "NumberofCells"vezes;
  - **printFullCell(NumberofLines)**: utiliza os predicados anteriores para criar uma célula completa.
  - **printIntermediateLine(NumberofCells)**: predicado utilizado para imprimir linhas intermédias esteticamente delimitadoras.
  - **printFullTile(NumberofTiles)**: conjuga os predicados printFullCell e printIntermediateLine para obter uma tile constituida por uma célula e secções delimitadoras;
  - **printSectorLimit(NumberofSectors)**: predicado auxiliar de printFullSector, que delimita um sector;
  - **printFullSector**: utiliza printFullTile e o predicado auxiliar de limite, para criar um sector completo;

- **printBottomLine(NumberofDashes)**: imprime a linha final do tabuleiro;

- **convertCode(_)**: traduz os átomos em peças de jogo;
  - **translateCodeToChar(X, Y)**: tradução dos numéros X para os caractéres Y;
- **printExample(_)**: Imprime a informação relativa à representação básica do *Shacru*, aos seus elementos e ainda apresenta um exemplo da *Board* num determinado estado do jogo.

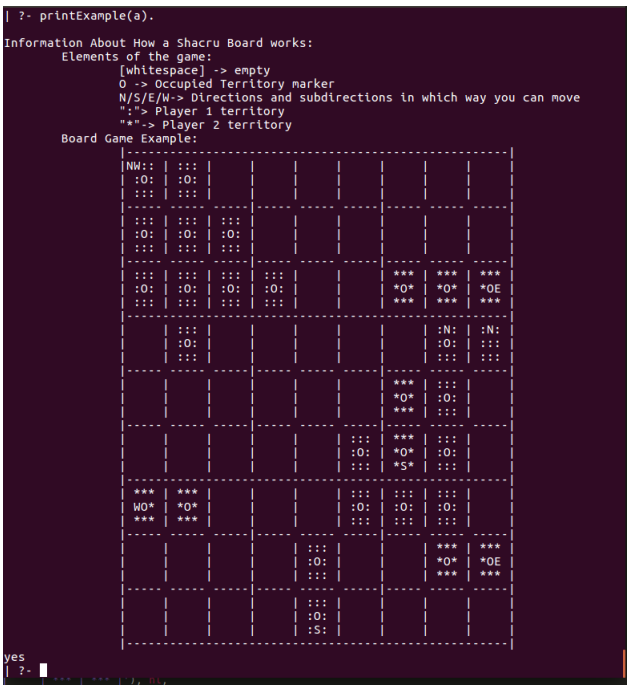

Figura 7: Informação geral apresentada

4 Movimentos

Os seguintes movimentos estão disponíveis no jogo *Shacru*:

- **choosePiece(+Board, +Player, +TileNumber).**
  - Selecciona qual das peças do tabuleiro o jogador deseja utilizar.
- **movePiece(+Board, +Player, +TileNumber, -NewBoard).**
  - Função utilizada para movimentação das peças.
- **validMoves(+Board, +Player, -PossibleMoves).**
  - Devolve as jogadas possíveis. PossibleMoves representa uma lista de coordenadas para movimentos possíveis.
- **value(+Board, +Player, +TileNumber, -Value).**
  - Avaliação do potencial de jogadas. Devolve o seu valor em *Value*.
- **gameOver(+Board, -Winner).**
  - Devolve o vencedor do jogo em *Winner*.
- **changeOrientation(+Board, +Player,+TileNumber, +PieceType, -NewBoard).**
  - Muda a direcção da peça escolhida.