**Monkey Queen**

**Relatório Intercalar**



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

**Monkey\_Queen\_1:**

Luís Miguel Gonçalves –

José Miguel Costa – 201402717

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

16 de Outubro de 2016

1. **O Jogo Monkey Queen**

Monkey Queen é um jogo de dois jogadores, jogado num tabuleiro 12x12 e foi concebido em 2011 por Mark Steere. Inicialmente o tabuleiro tem duas rainhas (uma pilha de 20 peças pretas e outra de brancas). Os dois jogadores fazem jogadas à vez que consistem em mexer a rainha, uma pilha por turno. O objetivo do jogo é matar a rainha inimiga ou deixar o adversário sem movimentos possíveis.

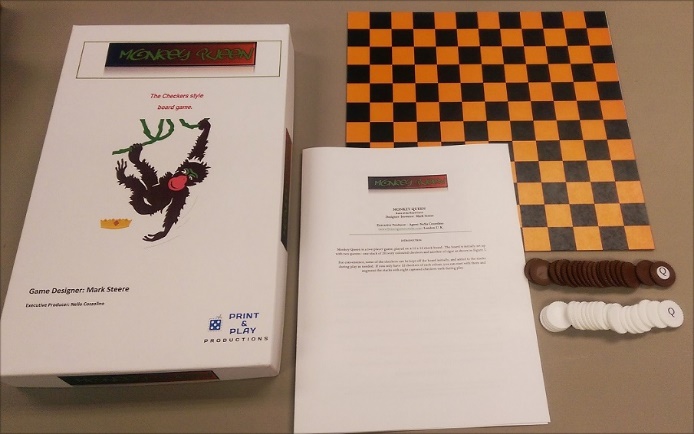


Image 1: Versão fisica do jogo

1. **Representação do Estado do Jogo**

Como referido acima o tabuleiro é quadrado, sendo que cada lado tem a largura de 12 células, assim sendo a abordagem que considerámos mais apropriada foi criar uma lista de 12 listas, em que cada lista representa uma linha do tabuleiro cada uma com 12 elementos. O tabuleiro é declarado da seguinte maneira, representando o seu estado inicial:

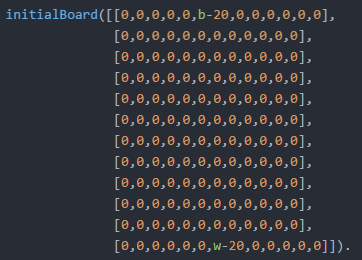
****

Image 2: Representação do tabuleiro inicial (prolog)

Cada posição da lista a **0** representa uma casa vazia no tabuleiro. As duas posições com o valor **20** representam as duas posições inicias das rainhas, sendo que 20 é o número de peças da pilha. Os caracteres **b** e **w** são indicativos da cor das peças, sendo preto e branco respetivamente.

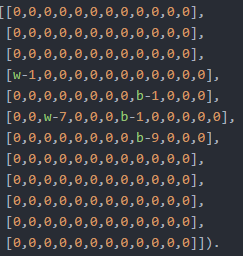
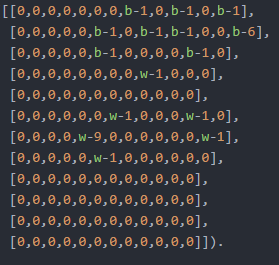
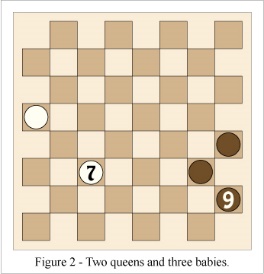
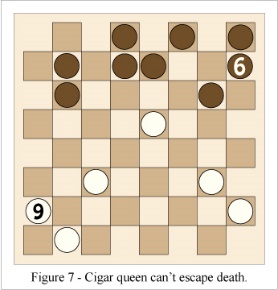


Image 3: Visualização Prolog e representação real (Posição Final)

Image 4:Visualização Prolog e representação real (Posição Intermédia)

A figura da esquerda representa um estado intermédio do tabuleiro. As posições onde **1** é o número na segunda parte do par, representam um bebé deixado pela rainha, e as outras duas posições com um número superior a 1 representam a rainha sendo o número a quantidade de peças na pilha.

A figura da direita representa um estado final do tabuleiro, onde o jogador que usa as peças pretas irá perder na próxima jogada, pois não pode fazer mais movimentos com a rainha sem esta ser capturada na jogada do oponente.

1. **Visualização do Tabuleiro**

Por forma a imprimir a lista foram declarados predicados em Prolog e chegámos a uma representação que achamos adequada e percetível para jogador:

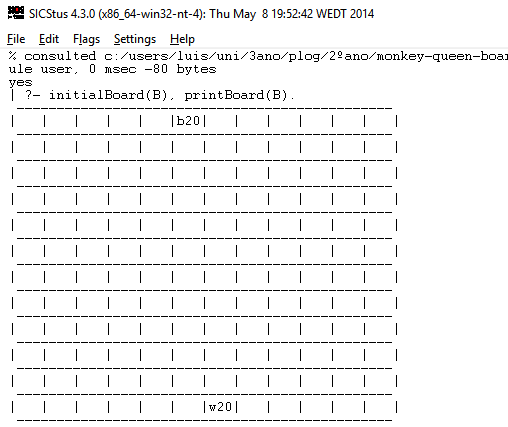


Image 5: Tabuleiro impresso em ASCII

Principais predicados para imprimir o tabuleiro:

* printBoard([]):-

printLine(x).

printBoard([H|T]):-

printLine(x),

printSpaces(H),

printBoard(T).

Em que a lista é passada como argumento ([H-T]) e a função printLine é responsável por desenhar as linhas horizontais do tabuleiro e printSpaces por desenhar as linhas verticais e por imprimir as peças no tabuleiro.

Por cada elemento da lista recebida por printSpaces é chamada a função translatePrint em que consoante o tamanho e cor da peça representa-a no tabuleiro com uma dimensão constante de 3 caracteres, como é possível visualizar abaixo:

* translatePrint(0):-

write(' ').

translatePrint(Colour-Char):-

Char < 10,

write(' '),

write(Char).

translatePrint(Colour-Char):-

Char >= 10,

write(Colour),

write(Char).

1. **Movimentos**

A rainha movimenta-se como uma rainha de xadrez (em qualquer direção o número de casas que quiser). Quando se movimenta sem capturar, a rainha deixa na sua posição anterior uma das suas peças reduzindo a altura da pilha em um. As peças que ficam para trás são os bebés. Quando se movimenta para capturar a rainha não perde peças e a captura funciona por substituição, como no xadrez. Uma rainha com uma pilha de tamanho dois não pode fazer movimentos que não sejam de captura.

Os bebés movimentam-se da mesma forma que a rainha para capturar, mas quando não capturam e se movimentam, têm, obrigatoriamente, de se aproximar da rainha inimiga.

Principais predicados para movimentar as peças:

* tryToMovePiece(BoardState, FX-FY, TX-TY, Board).
  + Irá receber toda a informação relativa ao movimento desejado pelo utilizador.
* validateFromPosition(BoardState, FX-FY).
  + Valida se a peça que o jogador pretende mover o pertence.
* validateToPosition(BoardState, TX-TY).
  + Valida se a posição final da peça é válida em termos de andar numa direção válida.
* validateMovePiece(BoardState, FX-FY, TX-TY, Board).
  + Valida se posição final da peça é válida em termos de respeitar as regras restantes do jogo.