

Mestrado Integrado em Engenharia Informática e Computação

Monkey Queen

Relatório Intercalar

Programação em Lógica

Grupo Monkey\_Queen\_2:

Ana Rita da Costa Torres, up201406093@fe.up.pt

Rui Pedro Correia Soares, up201404965@fe.up.pt

23 de novembro de 2016

Índice

Índice de Imagens

## 1.Jogo

### 1.Descrição do Jogo

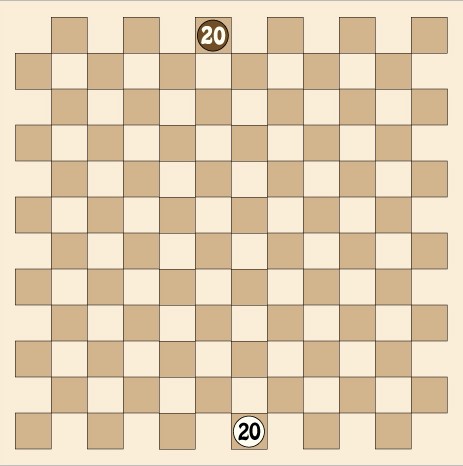
O jogo “Monkey Queen” foi concebido por Mark Steere em janeiro de 2011.

Este encontra-se disponível online nos seguintes sites:

o Ig Game Center ([http://www.iggamecenter.com/info/pt/main.html)](http://www.iggamecenter.com/info/pt/main.html) o Mind Sports ([http://www.mindsports.nl)](http://www.mindsports.nl/)

#### Regras do Jogo

O tabuleiro do jogo tem dimensões 12X12, cada jogador tem vinte peças em pilha, numa fase inicial, sendo um jogador da equipa *cigar* (peças castanhas)e outro da equipa *ivory* (peças brancas).



# Figura 1 - Tabuleiro no Estado Inicial

O jogador da equipa *ivory* é sempre o primeiro a iniciar o jogo.

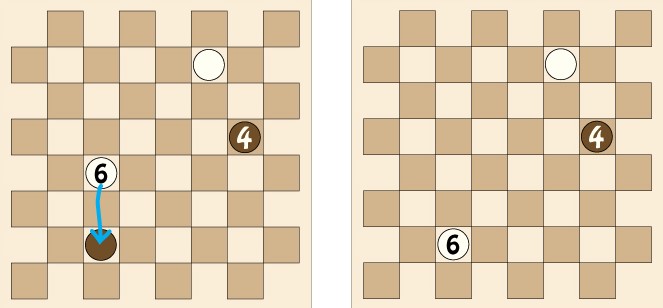
Existem dois tipos de peças:

* *Monkey Queen*: pilha de duas ou mais peças (só existe uma de cada jogador no tabuleiro);
* *Baby Monkey*: uma peça (*singleton*).

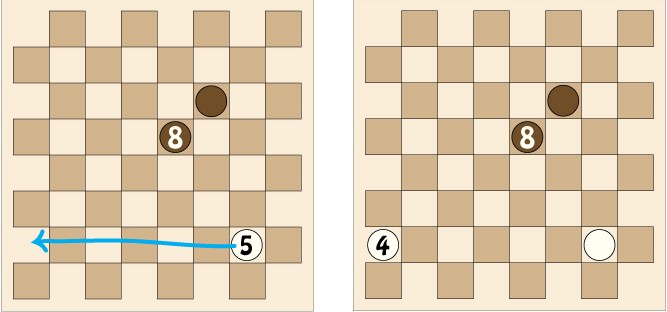
As peças movem-se em sequências de quadrados não ocupados em linha reta terminada por um quadrado ocupado pelo inimigo, podendo a orientação do movimento ser vertical, horizontal ou diagonal.

No caso da *Monkey Queen*, o movimento com captura pode ter dois desfechos, se o inimigo capturado for um *Baby Monkey*, esta ocupa o seu lugar, por outro lado se for a *Monkey Queen* adversária a equipa que realizou a jogada ganha.

Os movimentos sem captura só podem ser realizados pela *Monkey Queen*, se a sua pilha tiver um número de peças superior a dois e é obrigatório que ao efetuar o movimento esta peça deixe um *singleton* no seu quadrado de partida.

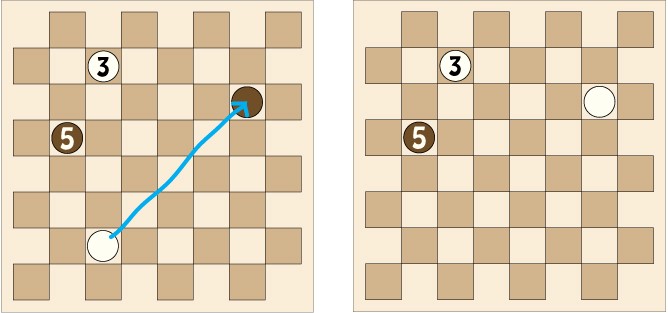


# Figura 2 - Ivory Queen captura Baby Cigar

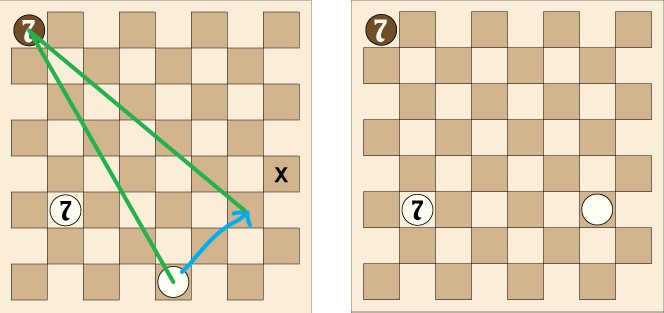


# Figura 3 - Ivory Queen executa movimento sem captura

O movimento de captura das peças *Baby Monkey* é semelhante ao movimento das *Queen Monkey*. Já o seu movimento sem captura tem outras condições, ao realizálo é imperativo que o *Baby Monkey* se coloque numa posição que se situe a uma menor distância da *Monkey Queen* adversário do que anteriormente.



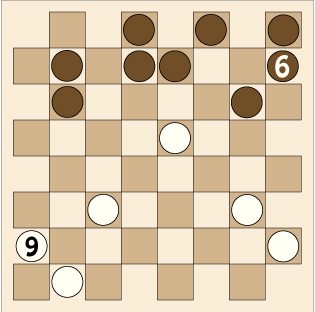
# Figura 4 - Baby Ivory captura Baby Cigar



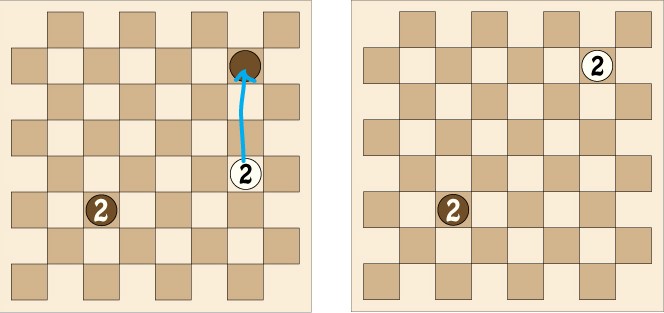
# Figura 5 - Baby Ivory executa movimento sem captura

Por fim, existem 2 maneiras de ganhar o jogo, sendo que o empate é um resultado impossível neste jogo:

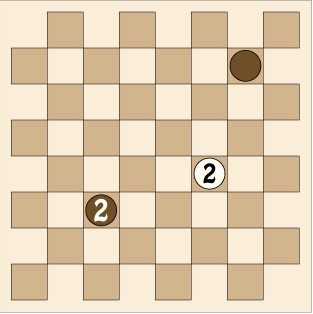
* Matar a *Monkey Queen* inimiga;
* Posicionar as peças de forma a que o oponente não seja capaz de realizar mais movimentos.



# Figura 6 - Cigar Queen não consegue evitar a captura



# Figura 7 - Baby Ivory só tem uma hipótese de movimento, que consequentemente conduz à sua captura



# Figura 8 - Ivory Queen não pode realizar qualquer movimento e, portanto, perde o jogo

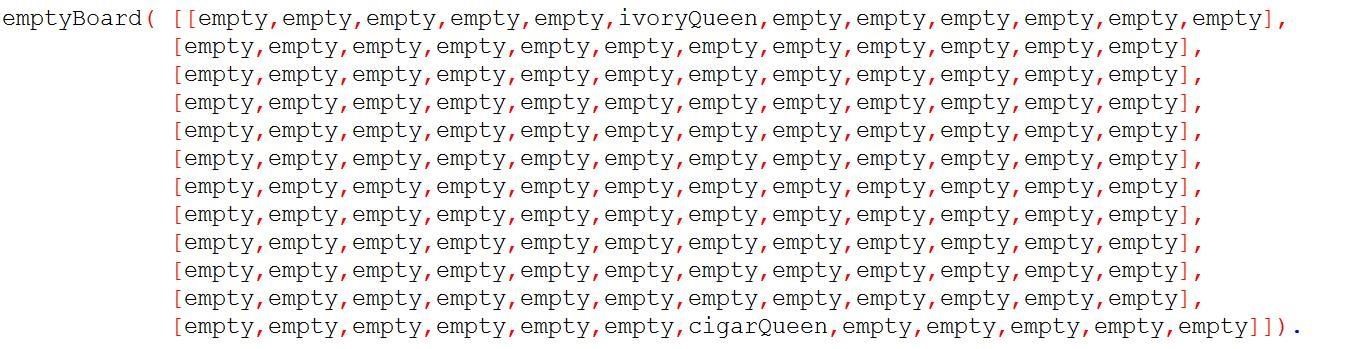
### 2.Implementação do Jogo

### 1.Representação do Tabuleiro

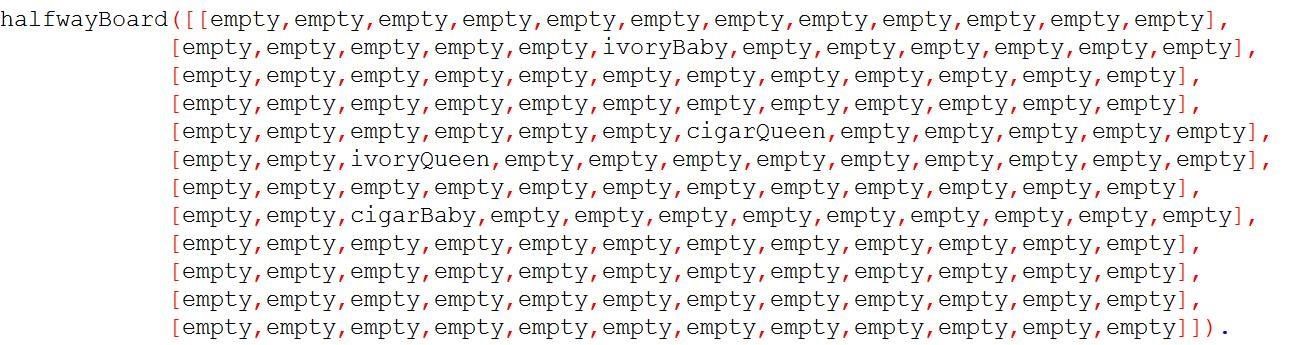
Para a implementação do jogo em Prolog, era necessário escolher que representação de dados usaríamos para o tabuleiro e peças. Como tal, escolhemos uma matriz (ie. lista de listas), de dimensões 12x12, cujos elementos são átomos que representam o estado em que se encontra a célula. De entre os átomos possíveis, temos a célula vazia (*empty*), a Monkey Queen de cada cor (*ivoryQueen, cigarQueen*), e os seus Monkey Babies (*ivoryBaby*, *cigarBaby*).

O tabuleiro começa, tal como referido nas regras, com cada jogador com 20 peças na sua pilha inicial.

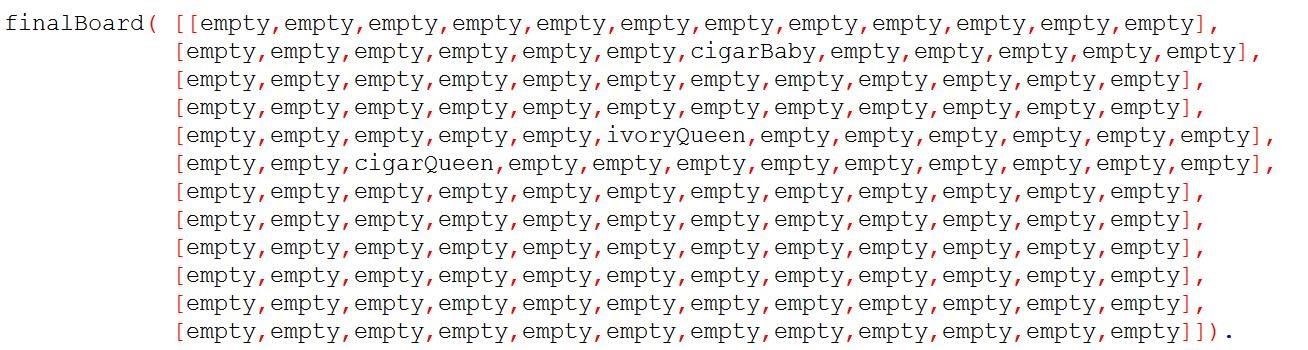
O jogador pode usar o predicado *getPiece(+X, +Y, +Board, -Symbol)* para obter a peça correspondente às coordernadas (X, Y) introduzidas. Para além disso pode usar o predicado (ainda não implementado) *setPiece(+X, +Y, +Symbol, +Board, -Board1)*.



# Figura 9 - Tabuleiro Vazio



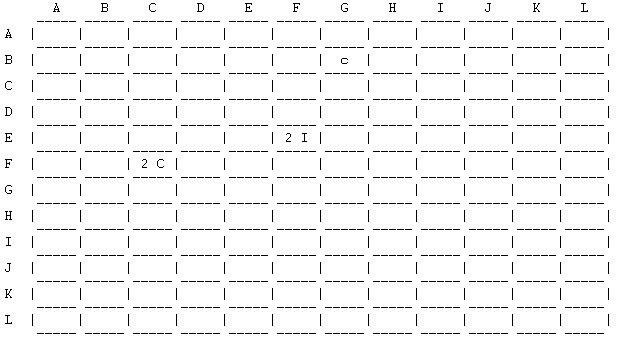
# Figura 10 - Tabuleiro Intermédio



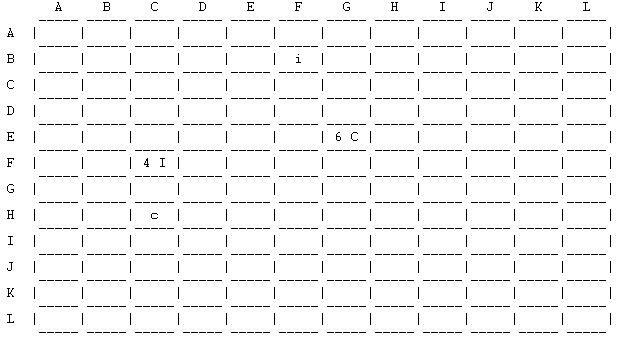
# Figura 11 - Tabuleiro Final

### 2.Visualização do Tabuleiro

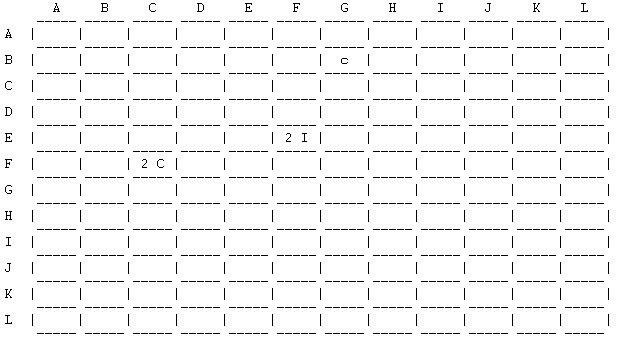
Para imprimir o tabuleiro de jogo para a consola, implementámos um predicado que percorre a representação do tabuleiro e imprime-o, recursivamente e fila a fila. predicado, que para além do estado tabuleiro necessita da quantidade de peças na Queen de cada jogador nesse estado é *printFancyBoard([+IvoryQueenStack, +CigarQueenStack, +Board]).*



# Figura 12 - Tabuleiro Inicial



# Figura 13 - Tabuleiro Intermédio



# Figura 14 - Tabuleiro Final

### 3.Movimentos

Para a verificação de movimentos, o predicado *checkMove(+Player, +X, +Y, +XTarget, +YTarget, +Board)* que, para um dado jogador, coordenadas atuais e coordenadas-alvo, verifica se há a jogada é possível. Para já verifica apenas jogadas que não resultem em captura.

Para executar jogadas, iremos usar o predicado *makeMove(+Player, +X, +Y, +Xtarget, +Ytarget, +Board)*, que primeiro verificará se a jogada é possível para o conjunto de coordenadas introduzido, e se for possível, faz a jogada de acordo com as regras.

## 2.Lógica de Jogo

1. Representação do estado de jogo

### 2. Visualização do tabuleiro

### 3. Lista de jogadas válida

### 4. Execução de jogadas

### 5. Avaliação do tabuleiro

### 6. Final do jogo

### 7. Jogada do computador

## 3.Interface com o Utilizador

## 4.Conclusões