Monkey Queen

Relatório Intercalar



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 4:

Marta Diogo Torgal Pinto - up201407727

Telmo João Vales Ferreira Barros - up201405840

Faculdade de Engenharia da Universidade do Porto

1. de Outubro de 2016

## 

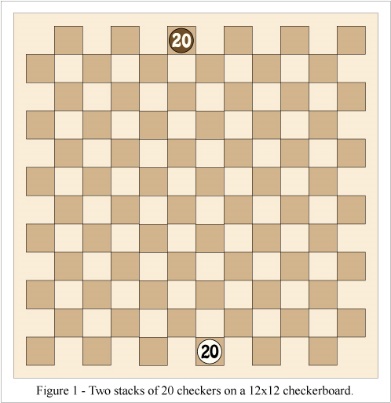
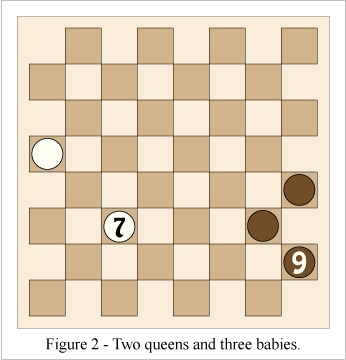
## Descrição do Jogo

Monkey Queen foi idealizado em Janeiro de 2011 por Mark Steere, sendo considerado por este um jogo abstrato, no qual dois jogadores competem num tabuleiro 12x12. O objetivo deste jogo é eliminar a rainha adversária ou deixar a mesma sem movimentos possíveis.

As jogadas são feitas alternadamente entre os jogadores, de cor Branco e Preto. O jogador 1 é o primeiro a jogar e joga com as peças brancas, o adversário joga com as peças pretas. No entanto, pode ser aplicada a *Pie Rule* após a primeira jogada, se o segundo jogador assim o entender, isto é, os jogadores trocam de posição e o jogador 2 passa a ser o jogador 1. O “novo” jogador 2 faz agora a sua “primeira” jogada no controlo das peças pretas.

O jogo começa com as duas rainhas nas colunas F e G de lados opostos do tabuleiro. Cada rainha é composta por uma pilha com 20 peças da sua cor.

No decorrer do jogo cada jogador terá exatamente uma rainha, que será a peça com pelo menos duas peças empilhadas, e poderá ter um ou mais bebés que são as peças unitárias da mesma cor que a sua.

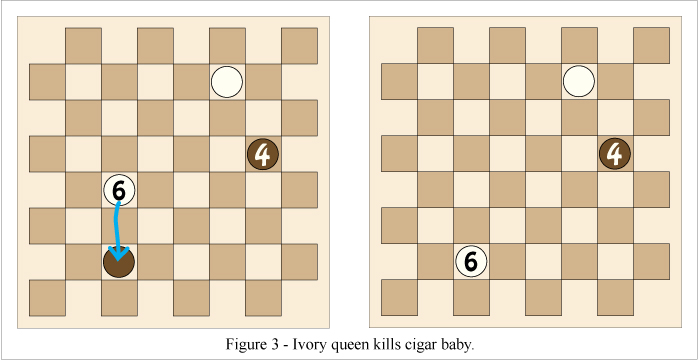


### Movimentos

Quanto à rainha, esta pode se mover no tabuleiro como num jogo de xadrez, isto é, pode mover a pilha toda que a constituí em qualquer direção (horizontal, vertical e diagonal) numa sequência de casas vazias.

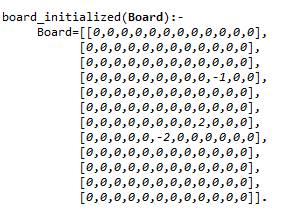
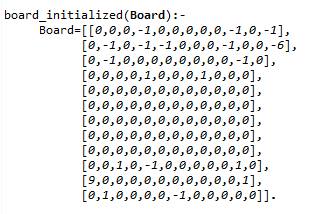
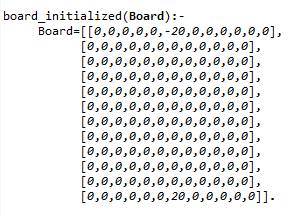
Se o movimento terminar numa casa ocupada por uma peça adversária, rainha ou bebé, a mesma é capturada e substituída pela rainha. Se o movimento da rainha não originar a captura de nenhuma peça adversária a mesma deixa na sua posição anterior um bebé, reduzindo a altura da sua pilha numa unidade.

Quanto aos bebés, estes podem-se mover exatamente da mesma forma da rainha para capturar a rainha ou bebés adversários.

Quando o seu movimento não originar a captura de nenhuma peça adversária, a sua distância em linha reta em relação à rainha adversária tem de ser encurtada.

Não é obrigatório proceder à captura de peças adversárias ainda que exista essa possibilidade.

## Representação do Estado do Jogo

Para representar o estado atual do jogo decidimos utilizar uma lista de listas que correspondem às diferentes linhas do tabuleiro. As peças e os espaços vazios são representados por números. Os números negativos representam as peças pretas e os positivos as peças brancas, o valor absoluto do número corresponde ao número de peças empilhadas (no caso das rainhas). Os espaços vazios são representados pelo número 0.

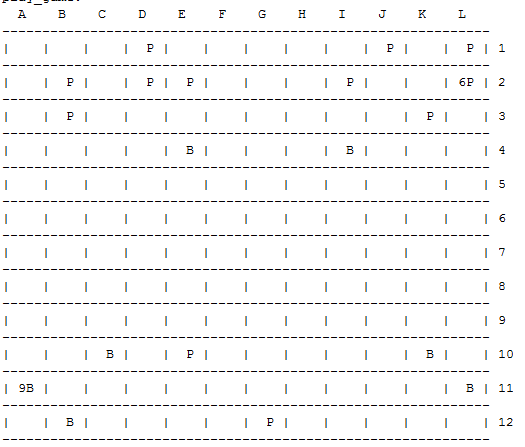
Estado Final

Estado Intermédio

Estado Inicial

## Visualização do Tabuleiro

O output do tabuleiro de jogo é o seguinte:



Para a sua visualização utilizamos os seguintes predicados:

display\_board([L1|Ls]):-

write(' A B C D E F G H I J K L'), nl,

write('-------------------------------------------------------------'), nl,

display\_lines([L1|Ls], *1*).

display\_lines([L1|Ls], Nlines):-

display\_line(L1), write('| '), write(Nlines), nl,

write('-------------------------------------------------------------'), nl,

Nextlines is Nlines + *1*,

display\_lines(Ls,Nextlines).

display\_lines([], Nlines).

display\_line([E|Es]):-

write('|'),

translate(E,T),

write(T),

display\_line(Es).

display\_line([]).

translate(*0*,' ')**.**

translate(*-20*,' 20P')**.**

translate(*-19*,' 19P')**.**

translate(*-18*,' 18P')**.**

translate(*-17*,' 17P')**.**

translate(*-16*,' 16P')**.**

translate(*-15*,' 15P')**.**

translate(*-14*,' 14P')**.**

translate(*-13*,' 13P')**.**

translate(*-12*,' 12P')**.**

translate(*-11*,' 11P')**.**

translate(*-10*,' 10P')**.**

translate(*-9*,' 9P ')**.**

translate(*-8*,' 8P ')**.**

translate(*-7*,' 7P ')**.**

translate(*-6*,' 6P ')**.**

translate(*-5*,' 5P ')**.**

translate(*-4*,' 4P ')**.**

translate(*-3*,' 3P ')**.**

translate(*-2*,' 2P ')**.**

translate(*-1*,' P ')**.**

translate(*20*,' 20B')**.**

translate(*19*,' 19B')**.**

translate(*18*,' 18B')**.**

translate(*17*,' 17B')**.**

translate(*16*,' 16B')**.**

translate(*15*,' 15B')**.**

translate(*14*,' 14B')**.**

translate(*13*,' 13B')**.**

translate(*12*,' 12B')**.**

translate(*11*,' 11B')**.**

translate(*10*,' 10B')**.**

translate(*9*,' 9B ')**.**

translate(*8*,' 8B ')**.**

translate(*7*,' 7B ')**.**

translate(*6*,' 6B ')**.**

translate(*5*,' 5B ')**.**

translate(*4*,' 4B ')**.**

translate(*3*,' 3B ')**.**

translate(*2*,' 2B ')**.**

translate(*1*,' B ')**.**

## Movimentos

Os seguintes cabeçalhos dos predicados representam os possíveis tipos de jogadas existentes e alguns predicados auxiliares para execução de tais movimentos:

*%Pedir ao utilizar a jogada a realizar, ou seja, posicao inicial da pec e posicao final*

*%X nome da coluna*

*%Y numero da linha*

move(Xinitial, Yinitial,Xfinal,Yfinal)**.**

*%verficar se a posicao é valida (se se encontra dentro do tabuleiro)*

legal\_pos(X,Y)**.**

*%verifica se os parametros introduzidos como posicoes finais nao saem do tabuleiro*

*%se a linha e a coluna finais forem diferentes das posicoes iniciais chama a funcao move\_diagonal*

*%se a linha final for igual a inicial chama a funcao move\_horizontal*

*%se a coluna final for igual a inicial chama a funcao move\_vertical*

legal\_move(Xinitial, Yinitial, Xfinal,Yfinal)**.**

*%verifica se a rainha se movimenta para uma posicao onde se encontra um baby ou rainha da equipa adversaria*

legal\_queen\_move(Xinitial,Yinitial,Xfinal,Yfinal)**.**

*%verfifca se as posicoes final sao posicoes validas em relacao a posicao inicial e guarda a nova posicao ou whatever*

diagonal\_move(Xinitial, Yinitial, Xfinal,Yfinal)**.**

*%guarda as novas posicoes apos fazer o movimento horizontal da peca*

horizontal\_move(Xinitial, Yinitial, Xfinal,Yfinal)**.**

*%guarda as novas posicoes apos fazer o movimento vertical da peca*

vertical\_move(Xinitial, Yinitial, Xfinal,Yfinal)**.**

*%pesquisa posicao rainha*

queen\_pos(Jog,Board)**.**

*%verifica se a peca ao movimentar-se se se aproxima da rainha*

queen\_aprox(Xqueen, Yqueen, Xinitial, Yinitial, Xfinal, Yfinal)**.**

*%verifica se existe um baby da equipa adversaria na posicao final*

*%se sim come esse baby e nao deixa um baby na posicao inicial*

*%se nao existir nenhuma peca na posicao final deixa um baby na posicao inicial ao deslocar se*

eat(Xinitial,Yinitial,Xfinal,Yfinal)**.**

*%deixa um baby na posição inicial da rainha após esta se movimentar*

dropBaby(X,Y)**.**

*%acaba o jogo?*

gameEnd(Rainha1, Rainha2)**.**

## Referências

Mark Steere. 2011. Monkey Queen. Acedido a 9 de Outubro de 2016. http://www.marksteeregames.com/Monkey\_Queen\_rules.html