Cannon

# Relatório Intercalar



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Cannon\_3:

Luís Alvela Duarte Mendes - 201605769 Miguel Dias de Carvalho – 201605757

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

21 Outubro de 2018

## História e Regras de Cannon:

História:

Cannon é um jogo tático de tabuleiro de origem asiática que simula uma batalha.

Consiste num jogo de captura de peças no mesmo grupo geral que o xadrez, jogando-se num tabuleiro quadrado de 10x10 com peças de duas cores diferentes (preto e vermelho) que representam os soldados de cada *phalanx* (cada exército ao controlo dos jogadores).

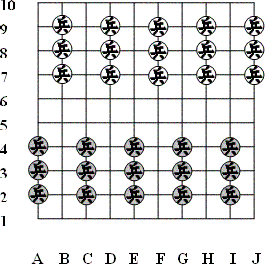
Adicionalmente, existe uma peça única para cada lado que simboliza a “cidade” de cada

jogador (ou seja, o objetivo de captura).

Regras:

Primeiramente, tanto o jogador vermelho como o preto devem de colocar os seus soldados em linha, desocupando a primeira e décima fila. Depois, colocam a peça da “cidade” algures na primeira linha relativa ao seu respetivo lado do tabuleiro sem contar com os cantos (neste caso, seria a linha 1 para o lado preto e 10 para o vermelho tal como mostra na figura 1).

*Nota:* É necessário alterar a figura, colocando um símbolo da “cidade” na linha do tabuleiro mesmo.



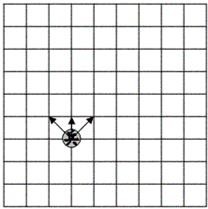
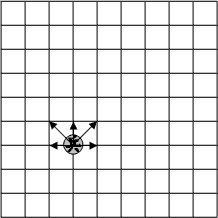
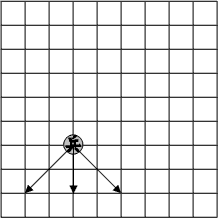
*Figura 1*

Em cada fase do jogo, cada interveniente tem duas opções:

1. Mover um soldado.
2. Usar um *cannon* (canhão), explicado na próxima secção. Passar a vez sem fazer uma destas duas coisas é **ilegal**.

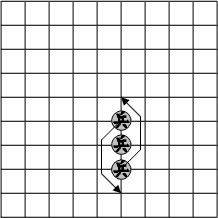
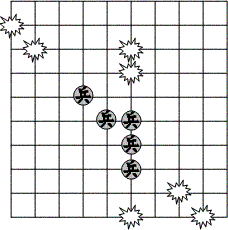
Conceitos:

* Movimento: um soldado poderá mover-se:
  + Para a frente, para *um* ponto adjacente a ele ortogonal ou diagonal. (figura 2)
  + Para capturar outra peça, para a direita, esquerda e para a frente. (figura 3)
  + Para recuar, *dois* pontos para um local ortogonal/diagonal se estiver adjacente a uma peça inimiga. (figura 4)

*Figura 2 Figura 3 Figura 4*

* Cannon: consiste numa linha de três soldados aliados quer ortogonal como diagonal que pode capturar peças a longa distância. O canhão só pode capturar se a posição imediatamente à sua frente não estiver ocupada, e, quando captura, pode fazê-lo a duas ou três posições de distância de cada extremo do canhão (figura 5).
  + O canhão captura sem se mover, podendo mover-se sem capturar peças, alterando a posição de cada extremo (figura 6).



*Figura 5 Figura 6*

Objetivo:

O objetivo consiste em capturar a peça “cidade” do outro jogador.

## Representação do Estado do Jogo

Para representar o jogo, mais especificamente, o tabuleiro, optamos por uma lista de listas de modo a simplificar o problema.

Representação inicial:

Ainda não se colocou a cidade de cada jogador, logo, o jogo ainda não começou, o tabuleiro encontra-se no seu estado inicial.

Nota: alinhar melhor as linhas dos excertos.

(excertos de código retirados diretamente do eclipse (SPIDER)):

initialBoard([

[emptyCell, emptyCell, emptyCell, emptyCell, emp-

tyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier],

[emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier],

[emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier, emptyCell, redSoldier],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[blackSoldier, emptyCell, blackSoldier, emp- tyCell, blackSoldier, emptyCell, blackSoldier, emptyCell, blackSol- dier, emptyCell],

[blackSoldier, emptyCell, blackSoldier, emp- tyCell, blackSoldier, emptyCell, blackSoldier, emptyCell, blackSol- dier, emptyCell],

[blackSoldier, emptyCell, blackSoldier, emp- tyCell, blackSoldier, emptyCell, blackSoldier, emptyCell, blackSol- dier, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell,emptyCell, emptyCell, emptyCell, emptyCell]

]).

Representação Intermédia:

Já se colocaram as cidades e um dos jogadores moveu-se, entretanto.

intermediumBoard([

[emptyCell, emptyCell, emptyCell, emptyCell, cityPiece, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[redSoldier, redSoldier, emptyCell, redSoldier, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, redSoldier, emptyCell, emptyCell, redSoldier, redSoldier, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, emptyCell, redSoldier, emptyCell, emptyCell, emptyCell , emptyCell, redSoldier, emptyCell, redSoldier],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, redSoldier, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, emptyCell, emptyCell, blackSoldier, blackSoldier, emptyCell, emptyCell, redSoldier, emptyCell, emptyCell],

[emptyCell, blackSoldier, blackSoldier, emp- tyCell, emptyCell, blackSoldier, emptyCell, redSoldier, emptyCell, emptyCell],

[emptyCell, emptyCell, blackSoldier, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, blackSoldier, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, cityPiece, emptyCell]

]).

Representação Final:

O jogador vermelho ganhou, capturando a cidade do jogador preto com um cannon.

finalBoard([

[emptyCell, emptyCell, emptyCell, emptyCell, cityPiece, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[redSoldier, redSoldier, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, redSoldier, emptyCell, emptyCell, redSoldier, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[blackSoldier, emptyCell, blackSoldier, blackSol- dier, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell],

[emptyCell, emptyCell, blackSoldier, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, redSoldier, emptyCell],

[emptyCell, emptyCell, blackSoldier, emptyCell, emptyCell, emptyCell, emptyCell, redSoldier, redSoldier, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, redSoldier, emptyCell],

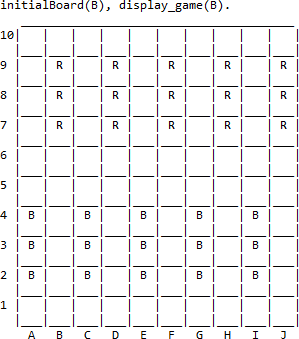
[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],

[emptyCell, emptyCell, emptyCell, emptyCell, emp- tyCell, emptyCell,emptyCell, emptyCell, cityPiece, emptyCell]

]).

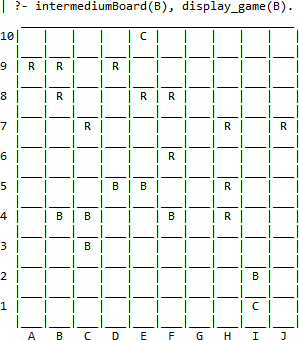
*Nota:* Alterar tabuleiros para ter um redCityPiece e um blackCityPiece para se poder distinguir melhor.

*Mais, o estado “final” não é assim tão final quanto isso, era melhor mostrar um soldado vermlho a pisar no local da cidade mesmo ou a cidade do outro jogador já ter desaparecido.*

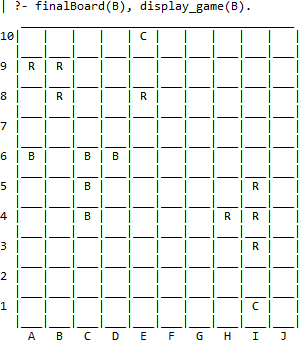
Desenho dos elementos: (screenshots do output do spider)

Tabuleiro Inicial:

Tabuleiro Intermédio:



Tabuleiro Final:



Código utilizado para o predicado de visualização:

**:-**include('boardLists.pl')**.**

*/\*Separador de cima\*/*

printGenericSeparator**:-**write(' '), nl**.**

*/\*Separador do meio\*/*

printMiddleSeparator**:-**write(' | | | | | | | | | | |'), nl**.**

*/\*Letras do fim\*/*

printLetters**:-** write(' A B C D E F G H I J '), nl**.**

*/\*Varios carateres e a sua correspondencia com elementos na lista interna\*/*

getSymbol(emptyCell, ' ')**.** getSymbol(redSoldier, 'R')**.** getSymbol(blackSoldier, 'B')**.** getSymbol(cityPiece, 'C')**.**

*/\*Imprime um daqueles carateres com barras a volta\*/*

printCell(**Char**)**:-** write('| '), write(Char), write(' ')**.**

*/\*Caso base: quando ja consumiu toda a linha (sublista da lista grande) so imprime uma barra final, um espaco e o separador\*/*

display\_board\_line([])**:-** write('| ') , nl, printMiddleSeparator**.**

*/\*Para cada linha, ou seja, sublista, vai tentando encontrar uma correspondencia en- tre o elemento a analisar nesta etapa, que vai sendo a cabeca, e o charater a impri- mir. Depois usa-se tail rescursion para ir passando ao proximo elemento\*/* display\_board\_line([**Head**|**Tail**])**:-** getSymbol(Head,**Char**), printCell(Char), dis- play\_board\_line(Tail)**.**

displayID(**ID**)**:-** ID \= '10', write(ID), write(' ')**.** displayID(**ID**)**:-** ID = '10', write(ID)**.**

*/\*Vai consumindo cada sublista da lista grande, cada linha [emptyCell, emp-*

*tyCell, ...] neste caso e um "Head", vai-se usando tail recursion para chegar as ou- tras linhas\*/*

display\_game\_aux([], [])**:-**printLetters, nl**.**

display\_game\_aux([**Head**|**Tail**], [**RHead**|**RTail**])**:-** displayID(RHead), display\_bo- ard\_line(Head), display\_game\_aux(Tail, RTail)**.**

display\_game(**Board**)**:-** printGenericSeparator, display\_game\_aux(Board, ['10', '9', '8',

'7', '6', '5', '4', '3', '2', '1'])**.**

*/\*display\_game(+Board,+Player)\*/*

# Referencias:

<https://boardgamegeek.com/boardgame/8553/cannon>

<http://www.boardspace.net/portuguese/about_cannon.html>