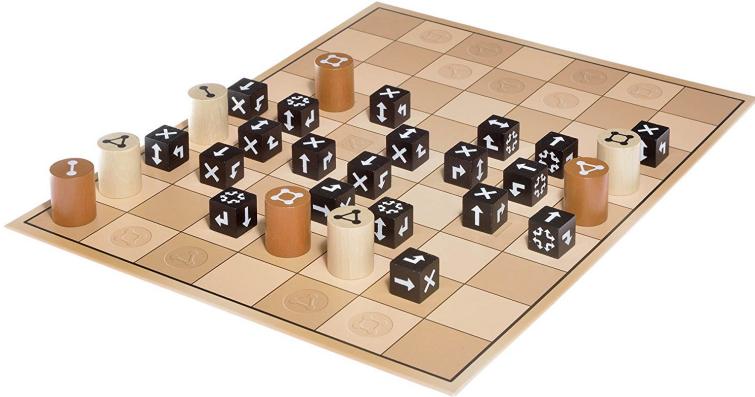


Programação em Lógica

Relatório Final

Barragoon 4



Leonardo Manuel Gomes Teixeira – up201502848
Maria Eduarda Santos Cunha – up201506524



1. Resumo

Este projeto é uma possível abordagem ao jogo Barragoon, descrito em detalhe na secção O Jogo Barragoon, em PROLOG.

Todo o processo de desenvolvimento verificou-se bastante difícil dada a novidade do paradigma da língua de programação em questão para nós, aliado ao facto de o próprio jogo ter uma lógica bastante complexa, pelo seu número elevado de regras e restrições.

Conseguimos implementar com sucesso todas as regras do jogo e completar os três modos de jogo – humano contra humano, humano contra computador e computador contra computador. No entanto, acabamos por não criar diferentes níveis de dificuldade.

Apesar do volume de regras, pensamos que a versão final é bastante *user friendly* e intuitiva de jogar.

De forma geral, o nosso feedback é bastante positivo. Este trabalho foi fundamental para a consolidação dos conteúdos lecionados nas aulas práticas e teóricas e sentimo-nos capazes de realizar os novos níveis ainda que não tenhamos tido tempo para tal.



Índice

1.	Resumo	2
2.	O Jogo Barragoon	4
2.1.	Tabuleiro e Peças.....	4
2.2.	Objetivo.....	4
2.3.	Movimentos	5
2.4.	Regras.....	5
3.	Lógica do Jogo.....	7
3.1.	Representação do Estado de Jogo	7
3.2.	Visualização do Tabuleiro.....	9
3.3.	Lista de Jogadas Válidas	11
3.4.	Execução de Jogadas.....	11
3.5.	Avaliação do Tabuleiro.....	12
3.6.	Final do Jogo	12
3.7.	Jogada do Computador.....	12
4.	Interface com o Utilizador.....	13
5.	Conclusões.....	15
6.	Bibliografia	15
7.	Anexos.....	16



2. O Jogo Barragoon

O Barragoon é um jogo de estratégia sem qualquer fator de aleatoriedade para 2 jogadores. Foi publicado pela primeira vez a 3 de março de 2014 pela companhia WiWa Spiele UG. As suas regras foram atualizadas pela última vez a 30 de março de 2016.

2.1. Tabuleiro e Peças

O jogo realiza-se num tabuleiro de 9x7 células e os jogadores jogam sempre à vez.

Existem 2 tipos principais de peças: as telhas de cada jogador, brancas ou castanhas, e os barragoons.

- As telhas dos jogadores possuem na sua face um símbolo com 2, 3 ou 4 círculos, relativo ao número de células que podem andar num só movimento (fig.1). Cada jogador começa com 7 telhas: 2 de 2 círculos, 3 de 3 círculos e 2 de 4 círculos.
- O barragoon é a peça central do jogo. É uma peça cúbica, em que cada uma das suas faces possui um símbolo que indica a permissão do jogador de mover a sua peça pela célula em que o barragoon se encontra (fig.2). O jogo começa com 8 barragoons, mas existem 32.



Figura 2: Telhas Brancas ou Castanhas



Figura 1: Faces de um Barragoon

2.2. Objetivo

Ambos os jogadores têm de recorrer às suas aptidões táticas para mover as suas telhas e dispor os barragoons de forma a que lhes seja permitido capturar todas as telhas do outro jogador ou, pelo menos, impedir o seu progresso.



O jogo acaba quando um dos jogadores já não consegue mover telhas, porque não possui nenhuma ou por as que tem se encontrarem limitadas por barragoons. O outro é o vencedor.

2.3. Movimentos

Existem 2 tipos de movimentos: full moves e short moves.

- Os full moves correspondem a percorrer x células, de acordo com o número de círculos na telha do jogador (2, 3 ou 4), respetivamente (fig.3);
- Os short moves correspondem a percorrer $x-1$ células, de acordo com o número de círculos na telha do jogador (2, 3 ou 4, logo movimentos de 1, 2 ou 3 células), respetivamente (fig.4).

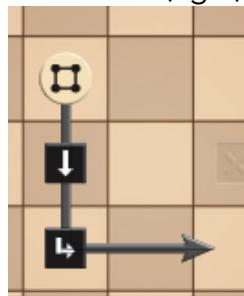


Figura 3: Full Move com Telha de 4 Círculos

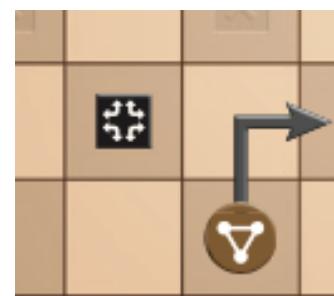


Figura 4: Short Move com Telha de 3 Círculos

2.4. Regras

- Uma peça é capturada se a peça do oponente terminar na mesma célula que ela (fig.5);
- Só é possível capturar uma peça durante um full move;
- As telhas com 2 círculos não podem capturar barragoons com o símbolo “todas as direções” virado para cima (fig.6);
- Se um barragoon for capturado, tem de voltar a ser colocado no tabuleiro, numa posição livre à escolha do jogador, com a face levantada para cima que ele preferir;
- Sempre que uma telha é capturada, são adicionados 2 barragoons novos ao tabuleiro, um por cada jogador, e coloca primeiro no tabuleiro o jogador cuja telha foi capturada;
- Nunca se pode mudar um barragoon de posição uma vez colocado;
- Quando se toca numa telha para a mover, não se pode trocar por outra ou voltar atrás;



- Durante um movimento, só se pode efetuar uma mudança de direção uma vez de 90° (fig.7);
- Os movimentos só podem ser verticais ou horizontais, nunca na diagonal.

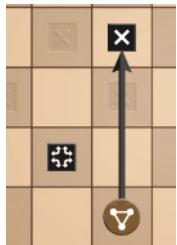


Figura 5: Captura de um Barragoon



Figura 6: Peça de 2 Círculos Não Captura Barragoon com Face “Todas as Direções” Voltada para Cima

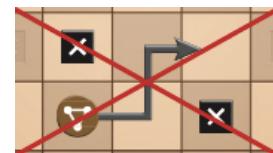


Figura 7: Movimento Impossível com 2 Mudanças de Direção

Ainda que só existam 32 barragoons e seria, por consequência, esperado que houvesse uma regra implementada para limitar a inserção de barragoons tendo em conta este valor, dado que sempre que um barragoon é comido é inserido o mesmo de volta e sempre que uma telha é comida são inseridos 2 e só há 7 telhas por jogador, concluímos que a situação em que se tenta inserir mais do que 32 nunca se vai verificar.

Por exemplo, tendo sido comidos 6 telhas de cada jogador, na situação final em que cada jogador possui 1 telha, quando um comer a do outro, o jogo acaba. E apenas nesse momento é que ocorreria a hipotética inserção do 33º.



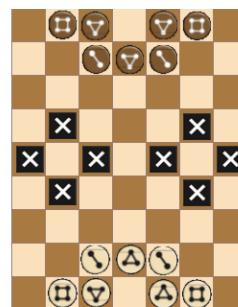
3. Lógica do Jogo

3.1. Representação do Estado de Jogo

Por questões de simplificação, os barragoons encontram-se aqui representados apenas por, por exemplo, barraX, em vez de bg-'barraX'. Na prática e em todos os fragmentos de código, referimo-nos a cada barragoon por um par, cujo primeiro elemento é bg e o segundo é a face voltada para cima.

Estado Inicial:

```
[[empty, b-4, b-3, empty, b-3, b-4, empty],
 [empty, empty, b-2, b-3, b-2, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [barraX, empty, barraX, empty, barraX, empty, barraX],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, w-2, w-3, w-2, empty, empty],
 [empty, w-4, w-3, empty, w-3, w-4, empty]]
```



		b4		b3				b3		b4	
			b2		b3		b2				
		x							x		
	x		x				x		x		
		x						x		x	
			w2		w3		w2				
		w4		w3				w3		w4	

Estados intermédios:

1.

```
[[empty, b-4, b-3, empty, b-3, b-4, empty],
 [empty, empty, b-2, b-3, b-2, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [barraX, empty, barraX, w-3, barraX, empty, barraX],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, w-2, empty, w-2, empty, empty],
 [empty, w-4, w-3, empty, w-3, w-4, empty]]
```

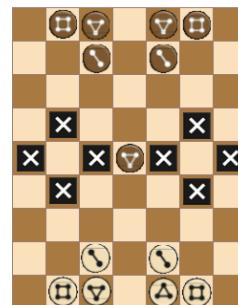


		b4		b3				b3		b4	
			b2		b3		b2				
	x		x		w3		x		x		
	x		x				x		x		
		x						x		x	
		w2				w2					
	w4		w3				w3		w4		



2.

```
[[empty, b-4, b-3, empty, b-3, b-4, empty],
 [empty, empty, b-2, empty, b-2, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [barraX, empty, barraX, b-3, barraX, empty, barraX],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, empty, w-2, empty, w-2, empty, empty],
 [empty, w-4, w-3, empty, w-3, w-4, empty]]
```



	b4	b3	b3	b4	
	b2	b2	b2		
	X	X	X	X	
X	X	X	b3	X	X
X	X	X	X	X	
	w2	w2	w2		
	w4	w3	w3	w4	

3.

```
[[empty, b-4, b-3, empty, b-3, b-4, empty],
 [empty, empty, b-2, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, barraX, empty, empty, b-2, barraX, empty],
 [barraX, empty, barraX, b-3, barraX, empty, barraX],
 [empty, barraX, empty, allDir, empty, barraX, empty],
 [empty, empty, empty, empty, right, w-2, empty],
 [empty, empty, w-2, empty, empty, empty, empty],
 [empty, w-4, w-3, empty, w-3, w-4, empty]]
```



	b4	b3	b3	b4	
	b2	b2	b2		
	X	X	X	X	
X	X	X	b3	X	X
X	X	X	*	X	
	w2	w2	w2		
	w4	w3	w3	w4	



4.

```
[[empty, b-4, b-3, empty, b-3, b-4, empty],
 [empty, empty, b-2, empty, empty, empty, empty],
 [empty, empty, empty, empty, empty, empty, empty],
 [empty, bg-'barraX', empty, empty, b-2, barraX, empty],
 [barraX, empty, barraX, empty, barraX, empty, barraX],
 [empty, b-3, empty, allDir, empty, barraX, empty],
 [empty, empty, empty, empty, right, empty, empty],
 [empty, empty, w-2, w-3, empty, empty, empty],
 [empty, w-4, w-3, empty, w-3, w-4, empty]]
```



I	b4	I	b3	I	I	b3	I	b4	I	
I	I	I	b2	I	I	I	I	I	I	
I	I	I	I	I	I	I	I	I	I	
I	I	X	I	I	I	b2	I	X	I	
I	X	I	I	X	I	I	X	I	I	
I	I	b3	I	I	*	I	I	X	I	
I	I	I	I	I	I	>	I	I	I	
I	I	I	w2	I	w3	I	I	I	I	
I	I	I	w2	I	I	I	I	I	I	
I	I	w4	I	w3	I	I	w3	I	w4	I

Estado final:

```
[[barraX, b-4, empty, b-2, b-3, b-4, barraX],
 [empty, barraX, empty, empty, empty, barraX, empty],
 [empty, barraX, empty, empty, empty, empty, empty],
 [barraX, w-2, barraX, empty, empty, empty, empty],
 [empty, barraX, empty, empty, empty, barraX, barraX],
 [empty, empty, empty, empty, b-2, empty, empty],
 [empty, empty, empty, empty, barraX, empty, empty],
 [empty, empty, barraX, b-3, barraX, empty, empty],
 [empty, barraX, w-3, barraX, w-3, barraX, empty]]
```



I	X	I	b4	I	I	b2	I	b3	I	b4	I	X
I	I	X	I	I	I	I	I	I	I	I	I	I
I	I	X	I	I	I	I	I	I	I	I	I	I
I	X	I	w2	I	X	I	I	I	I	I	I	I
I	I	X	I	I	I	I	I	X	I	X	I	I
I	I	I	I	I	I	I	I	X	I	I	I	I
I	I	I	I	I	X	I	b3	I	X	I	I	I
I	I	I	X	I	w3	I	X	I	w3	I	X	I

3.2. Visualização do Tabuleiro

```

1  initialBoard(
2    [[empty, b-4, b-3, empty, b-3, b-4, empty],
3     [empty, empty, b-2, b-3, b-2, empty, empty],
4     [empty, empty, empty, empty, empty, empty, empty],
5     [empty, bg-'barraX', empty, empty, empty, bg-'barraX', empty],
6     [bg-'barraX', empty, bg-'barraX', empty, bg-'barraX', empty, bg-'barraX'],
7     [empty, bg-'barraX', empty, empty, empty, bg-'barraX', empty],
8     [empty, empty, empty, empty, empty, empty, empty],
9     [empty, empty, w-2, w-3, w-2, empty, empty],
10    [empty, w-4, w-3, empty, w-3, w-4, empty]]).
11
12  displayGame(Game) :-
```



```

13  getBoard(Game, Board),
14  getCurrentPlayer(Game, Player),
15  clearScreen,
16  displayPlayerTurn(Player),
17  lettersAxis, nl,
18  horizontalBorder, nl,
19  numbersAxis(RowNumbers),
20  displayBoard(Board, RowNumbers), nl.
21  displayBoard([], []).
22
23  displayBoard([RowToDisplay|RemainingBoard],
24    [RowToDisplayNumber|RemainingRowNumbers]) :-
25    translate([RowToDisplayNumber]),
26    translate(RowToDisplay), border, nl,
27    horizontalBorder, nl,
28    displayBoard(RemainingBoard, RemainingRowNumbers).
29
30  % -- Board Translation --
31  translate([]).
32  translate(['empty'|R]) :- border, write(' _'), !, translate(R).
33  translate([w-2|R]) :- border, write(' w2 '), !, translate(R).
34  translate([w-3|R]) :- border, write(' w3 '), !, translate(R).
35  translate([w-4|R]) :- border, write(' w4 '), !, translate(R).
36  translate([b-2|R]) :- border, write(' b2 '), !, translate(R).
37  translate([b-3|R]) :- border, write(' b3 '), !, translate(R).
38  translate([b-4|R]) :- border, write(' b4 '), !, translate(R).
39  translate([bg-'barraX'|R]) :- border, write(' X '), !, translate(R).
40  translate([bg-'allDir'|R]) :- border, write(' + '), !, translate(R).
41  translate([bg-'oDirU'|R]) :- border, write(' V '), !, translate(R).
42  translate([bg-'oDirD'|R]) :- border, write(' A '), !, translate(R).
43  translate([bg-'oDirL'|R]) :- border, write(' <= '), !, translate(R).
44  translate([bg-'oDirR'|R]) :- border, write(' => '), !, translate(R).
45  translate([bg-'tDirH'|R]) :- border, write(' X '), !, translate(R).
46  translate([bg-'tDirV'|R]) :- border, write(' X '), !, translate(R).
47  translate([bg-'DtoR'|R]) :- border, write(' .> '), !, translate(R).
48  translate([bg-'DtoL'|R]) :- border, write(' <.'), !, translate(R).
49  translate([bg-'UtoR'|R]) :- border, write(' \">>'), !, translate(R).
50  translate([bg-'UtoL'|R]) :- border, write(' <\''), !, translate(R).
51  translate([bg-'LtoU'|R]) :- border, write(' -^'), !, translate(R).
52  translate([bg-'LtoD'|R]) :- border, write(' -v'), !, translate(R).
53  translate([bg-'RtoU'|R]) :- border, write(' ^-'), !, translate(R).
54  translate([bg-'RtoD'|R]) :- border, write(' v-'), !, translate(R).
55  translate(['um'|R]) :- write('1'), !, translate(R).
56  translate(['dois'|R]) :- write('2'), !, translate(R).
57  translate(['tres'|R]) :- write('3'), !, translate(R).
58  translate(['quatro'|R]) :- write('4'), !, translate(R).
59  translate(['cinco'|R]) :- write('5'), !, translate(R).
60  translate(['seis'|R]) :- write('6'), !, translate(R).
61  translate(['sete'|R]) :- write('7'), !, translate(R).
62  translate(['oito'|R]) :- write('8'), !, translate(R).
63  translate(['nove'|R]) :- write('9'), !, translate(R).
64
65  % -- Board Axis --
66  numbersAxis([um, dois, tres, quatro, cinco, seis, sete, oito, nove]).
67
68  lettersAxis :- write(' A B C D E F G').
69
70  % -- Board Borders --

```



```

71 horizontalBorder :- write('-----').
72 border :- write('|').

```

3.3. Lista de Jogadas Válidas

Obtenção de uma lista de jogadas possíveis.

3.4. Execução de Jogadas

O ciclo principal `playGame(+Game, -NewGame)` é caracterizado pela execução da jogada `playerTurn(+Game, -UpdatedGame)`, responsável por efetuar todos os pontos relativos à jogada de um jogador, `switchPlayer(+UpdatedGame, -NextPlayerGame)`, que trata de alternar os jogadores a cada jogada, e a chamada recursiva de `playGame`.

O predicado `playerTurn` é constituído por `playerMove(+Game, +RowSrc, +ColSrc, -Path)`, `validateMove(+Game, +RowSrc, +ColSrc, +Path, -PieceCaptured)`, `movePiece(+Game, +RowSrc, +ColSrc, +Path, -NewGame1)` e uma verificação de se alguma peça, telha ou barragoon, foi capturada.

Em `playerMove`, é pedido ao jogador que escolha a telha que pretende mover, através da inserção na consola da linha, representada por um número de 1 a 9, e da coluna, representada por uma letra de A a G (`chooseTile()`), é validado que essa telha existe e lhe pertence (`validateTile()`), pede-se o percurso que a peça vai efetuar, representado pelas teclas WASD (`choosePath()`) e, por fim, verifica-se se esse caminho não ultrapassa os limites do tabuleiro (`validatePath()`).

Em `validateMove`, trata-se de todas as questões relacionadas com o movimento da peça respeitar as regras do jogo. Verificando através de `getCell(+Board, +RowSrc, +ColSrc, -Piece)` qual é a telha que está a ser movida, com recurso a `isShortMove(+Piece, +Path)` e `isFullMove(+Piece, +Path)` determinamos se o percurso inserido pelo jogador é válido. Ainda, em `validateCrossMovements(+Game, +RowSrc, +ColSrc, +Path, +IsLongMove, +Piece, -PieceCaptured)` calcula-se se o movimento em questão obedece às regras de passagem em cima de barragoons e se não passa por cima de telhas se não for com o objetivo de as comer.

Em `movePiece`, é calculada a célula final em que a telha deve acabar através de `getDestCellFromPath(+RowSrc, +ColSrc, +Path, -RowDest, -ColDest)`, elimina-se o conteúdo dessa posição e procede-se à colocação da telha no novo destino com `moveFromSrcToDest(+GameBoard, +RowSrc, +ColSrc, +RowDest, +ColDest, -NewGameBoard)`.



3.5. Avaliação do Tabuleiro

Avaliação do estado do jogo, que permitirá comparar a aplicação das diversas jogadas disponíveis.

3.6. Final do Jogo

O jogo termina quando um dos jogadores já não tem telhas ou as que possui não se conseguem deslocar.

De forma a lidar com o primeiro ponto, utilizamos o predicado **countPlayerPieces(+Board, +CurrentPlayer, -CountPieces)** que verifica no tabuleiro o número de telhas do jogador cuja vez for de jogar. Se **CountPieces** retornar a 0, o jogo acaba e esse jogador perde.

Para verificar se, existindo peças, estas possuem movimentos possíveis, recorremos a **countMovesAvailable(+Game, +Row, +Column, -Count)** para contar os movimentos de uma peça. Se **Count** retornar a 0 para todas as peças em tabuleiro do jogador, o jogo acaba e esse jogador perde. Esta solução é um pouco *hardcoded*, pois recorremos a todos os percursos possíveis para cada telha para efetuar esta verificação.

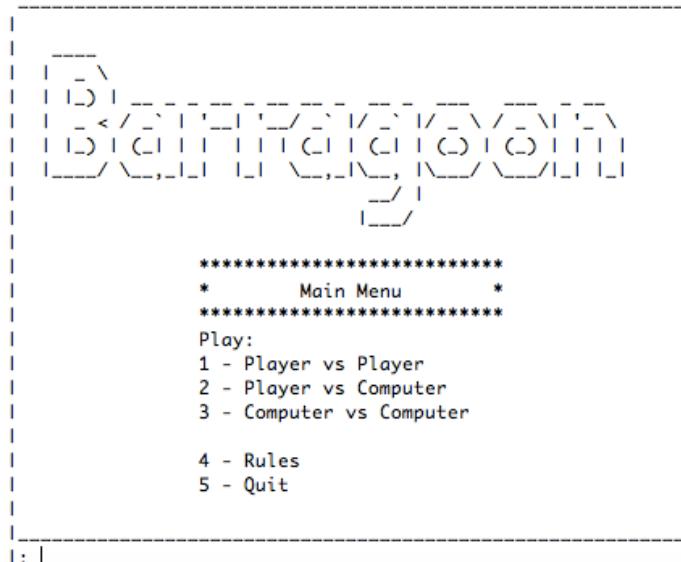
3.7. Jogada do Computador

Escolha da jogada a efetuar pelo computador, dependendo do nível de dificuldade.

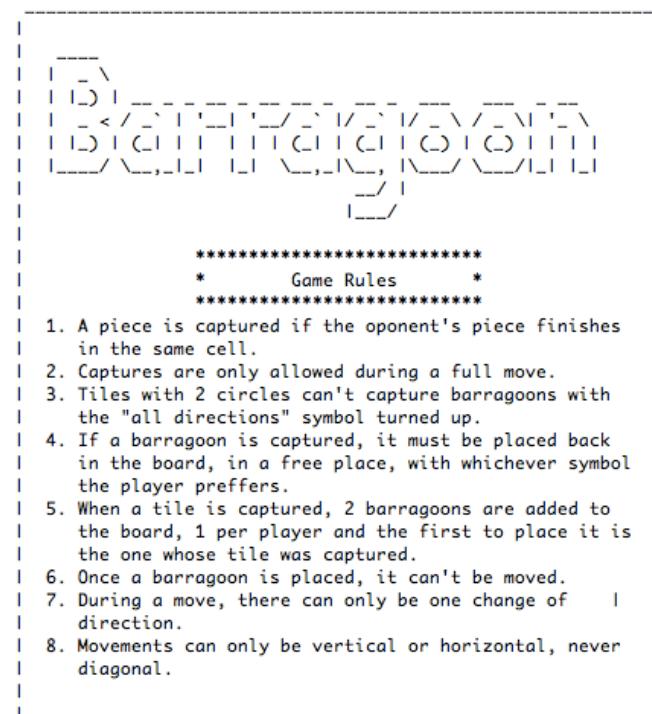


4. Interface com o Utilizador

Quando iniciado com o comando `start`, é apresentado o menu principal com as opções de jogar em modo humano contra humano, humano contra computador e computador contra computador, apresentar as regras ou sair do jogo.



Se for escolhida a opção 4 - Regras, é apresentado o seguinte menu que enumera as regras relevantes para o jogador.



Press any key to continue...



Iniciado o jogo em modo humano contra humano, as jogadas apresentam-se com um cabeçalho que indica a vez do jogador – White Player ou Black Player, o tabuleiro de jogo com eixos de números para as linhas e letras para as colunas e, por fim, perguntas relativas ao *input* do utilizador, como coordenadas da peça a mover e caminho que a peça deve seguir. Caso alguma destas informações seja inserida de forma errada, é sempre apresentada uma mensagem de erro e é pedido de novo ao jogador que insira a informação em questão até que a insira corretamente.

Se for o caso de inserir um barragoon no tabuleiro, é pedido ao respetivo jogador que insira a informação relativa a essa colocação (coordenadas de destino do barragoon e a face que se pretende que fique voltada para cima).

***** * White Player * *****						
A	B	C	D	E	F	G
1	b4 b3	b3 b4				
2		b2 b3 b2				
3						
4	X			X		
5	X	X	X	X	X	
6	X			X		
7						
8		w2 w3 w2				
9	w4 w3	w3 w4				

Which tile would you like to move?
Row: 9
Column: b

Please insert the path that you want that piece to follow:
(Use WASD - eg. wwwd + Enter).
l: wwd

***** * White Player * *****						
A	B	C	D	E	F	G
1				b3 b4		
2	b3	b2 b3 b2				
3		b4				
4	X				X	
5	w3	X	X	X	X	
6	X				X	
7		w4				
8		w2 w3 w2				
9		w3 w4				

Where do you wish to place your barragoon?
Row: 7
Column: a

Which barragoon do you wish to insert?

The options are:
1- X
2- + (All Directions)
3- => (One Direction)
4- <=> (Two Directions)
5- '>' (Left Turn)
6- '<' (Right Turn)
l: 4

In which direction you want to set the barragoon?

The options are:
1- - (Horizontal)
2- I (Vertical)
l:



5. Conclusões

Concluído o projeto, achamos importante ressaltar aquilo que poderia ter sido melhorado e não foi apenas pela ausência de tempo. Destaca-se o absentismo dos 2 níveis mínimos esperados para os modos de jogo humano contra computador e computador contra computador e a existência de soluções ditas *hardcoded* (por exemplo, o cálculo do fim do jogo através da verificação de todos os paths possíveis).

Ainda, pensamos que a nossa implementação é bastante intuitiva e de fácil compreensão, mas a sua complexidade poderia ter sido sensivelmente reduzida com alguma reutilização de código e *layering*.

Concluindo, consideramos que realizamos com sucesso aquilo a que nos propusemos e com o tempo que nos foi dado. Os conceitos das aulas foram, sem dúvida, consolidados e temos agora uma nova compreensão da utilidade desta língua de programação.

6. Bibliografia

- [1] https://www.youtube.com/watch?v=qG1i0_sn_FI
- [2] <https://boardgamegeek.com/boardgame/157779/barragoon>
- [3] http://www.barragoon.de/bsp/BARRAGOON_en.pdf
- [4] <https://stackoverflow.com>
- [5] <http://www.swi-prolog.org>



7. Anexos

main.pl

```

1  :- include('Utilities.pl').
2  :- include('Interface.pl').
3  :- include('Logic.pl').
4  :- use_module(library(system)).
5
6  %-----%
7  %---Barragoon -----%
8  %-----%
9  %---- escreva start -----%
10 %--- na consola para correr ---%
11 %-----%
12 %-----%
13
14 % --- START ---
15 start :-  
16 clearScreen,  
17 mainMenu.
```

Interface.pl

```

1  %-----%
2  %---Interface-----%
3  %-----%
4
5  %
6  % ----- MENUS -----
7  %
8
9  % -- Menus --
10 mainMenu :-  
11 displayMainMenu,  
12 getCharThenEnter(Option),  
13 (  
14 Option = '1' -> startGamePvP;  
15 Option = '2';  
16 Option = '3';  
17 Option = '4' -> displayRules;  
18 Option = '5';  
19  
20 clearScreen,  
21 write('ERROR : invalid input...'), spacing(1),  
22 mainMenu  
23 ).  
24
25 displayMainMenu :-  
26 upperFrame,  
27 titleFrame,  
28 write('|      *****|'),nl,  
29 write('|      * Main Menu *      |'),nl,  
30 write('|      *****|'),nl,  
31 write('|      Play:          |'),nl,  
32 write('|      1 - Player vs Player    |'),nl,  
33 write('|      2 - Player vs Computer   |'),nl,  
34 write('|      3 - Computer vs Computer |'),nl,
```





```

93
94    % -- Frames --
95    upperFrame :- 
96    write(' _____'),nl,
97    write('|           |'),nl.
98
99    lowerFrame :- 
100   write('|           |'),nl,
101   write('| _____|'),nl.
102
103   %
104   % ----- BOARD -----
105   %
106
107   % -- BOARD --
108   initialBoard( [[empty, b-4, b-3, empty, b-3, b-4, empty],
109   [empty, empty, b-2, b-3, b-2, empty, empty],
110   [empty, empty, empty, empty, empty, empty, empty],
111   [empty, bg-barraX, empty, empty, empty, bg-barraX, empty],
112   [bg-barraX, empty, bg-barraX, empty, bg-barraX, empty, bg-barraX],
113   [empty, bg-barraX, empty, empty, empty, bg-barraX, empty],
114   [empty, empty, empty, empty, empty, empty, empty],
115   [empty, empty, w-2, w-3, w-2, empty, empty],
116   [empty, w-4, w-3, empty, w-3, w-4, empty]]).
117
118   displayGame(Game) :-
119   getBoard(Game, Board),
120   getCurrentPlayer(Game, Player),
121
122   clearScreen,
123   displayPlayerTurn(Player),
124   lettersAxis,nl,
125   horizontalBorder, nl,
126   numbersAxis(RowNumbers),
127   displayBoard(Board, RowNumbers), nl.
128
129   displayBoard([], []).
130   displayBoard([RowToDisplay|RemainingBoard],
131   [RowToDisplayNumber|RemainingRowNumbers]) :-
132   translate([RowToDisplayNumber]),
133   translate(RowToDisplay),border, nl,
134   horizontalBorder, nl,
135   displayBoard(RemainingBoard, RemainingRowNumbers).
136
137   % -- Board Translation --
138   translate([]).
139   translate(['empty'|R]) :- border, write(' '), !, translate(R).
140   translate(['w-2'|R]) :- border, write(' w2 '), !, translate(R).
141   translate(['w-3'|R]) :- border, write(' w3 '), !, translate(R).
142   translate(['w-4'|R]) :- border, write(' w4 '), !, translate(R).
143   translate(['b-2'|R]) :- border, write(' b2 '), !, translate(R).
144   translate(['b-3'|R]) :- border, write(' b3 '), !, translate(R).
145   translate(['b-4'|R]) :- border, write(' b4 '), !, translate(R).
146   translate(['bg-'barraX|R]) :- border, write(' X '), !, translate(R).
147   translate(['bg-'allDir|R]) :- border, write(' + '), !, translate(R).
148   translate(['bg-'oDirU|R]) :- border, write(' A '), !, translate(R).
149   translate(['bg-'oDirD|R]) :- border, write(' V '), !, translate(R).
150   translate(['bg-'oDirL|R]) :- border, write(' <= '), !, translate(R).

```



```

151  translate([bg-'oDirR'|R]) :- border, write(' => '), !, translate(R).
152  translate([bg-'tDirH'|R]) :- border, write(' - '), !, translate(R).
153  translate([bg-'tDirV'|R]) :- border, write(' | '), !, translate(R).
154  translate([bg-'DtoR'|R]) :- border, write(' .> '), !, translate(R).
155  translate([bg-'DtoL'|R]) :- border, write(' < .'), !, translate(R).
156  translate([bg-'UtoR'|R]) :- border, write(' \\'> '), !, translate(R).
157  translate([bg-'UtoL'|R]) :- border, write(' <\'''), !, translate(R).
158  translate([bg-'LtoU'|R]) :- border, write(' -^ '), !, translate(R).
159  translate([bg-'LtoD'|R]) :- border, write(' -v '), !, translate(R).
160  translate([bg-'RtoU'|R]) :- border, write(' ^- '), !, translate(R).
161  translate([bg-'RtoD'|R]) :- border, write(' v- '), !, translate(R).
162  translate(['um'|R]) :- write('1'), !, translate(R).
163  translate(['dois'|R]) :- write('2'), !, translate(R).
164  translate(['tres'|R]) :- write('3'), !, translate(R).
165  translate(['quatro'|R]) :- write('4'), !, translate(R).
166  translate(['cinco'|R]) :- write('5'), !, translate(R).
167  translate(['seis'|R]) :- write('6'), !, translate(R).
168  translate(['sete'|R]) :- write('7'), !, translate(R).
169  translate(['oito'|R]) :- write('8'), !, translate(R).
170  translate(['nove'|R]) :- write('9'), !, translate(R).
171
172  % -- Board Axis --
173  numbersAxis([um, dois, tres, quatro, cinco, seis, sete, oito, nove]). 
174
175  lettersAxis :- write(' A B C D E F G').
176
177  % -- Board Borders --
178  horizontalBorder :- write(' -----').
179  border :- write('|').

```

Logic.pl

Utilities.pl