Breakthru

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Breakthru_1:
João David Gonçalves Baião - 201305195
Pedro Vieira de Castro - 201305337

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

7 de Outubro de 2014

1 O Jogo Breakthru

1.1 Introdução

O objectivo deste trabalho é realizar um programa em PROLOG onde será possível jogar um jogo de tabuleiro de forma virtual. Através deste trabalho procuramos adquirir competências na área da Programação em Lógica. O jogo escolhido para ser desenvolvido é o Breakthru.

O Breakthru é um jogo de mesa para duas pessoas. Foi criado por Alex Randolph. Foi comercialmente lançado por 3M Company na década de 60.

1.2 Regras do Jogo

O jogo tem como objetivo um dos jogadores conseguir fazer a *MotherShip* atravessar as defesas adversárias e chegar aos limites do tabuleiro. O outro jogador tem que impedir que isto seja possível de se concretizar. O jogador **amarelo** (o tiver que escoltar a *MotherShip)* tem ao seu dispor 12 embarcações para o ajudar enquanto o jogador **cinzento** (defensor) tem 20.

A cada jogada o jogador tem duas opções:

- Move horizontalmente ou verticalmente uma das suas embarcações duas posição ou duas das suas embarcações uma única vez cada uma. Caso o jogador mova a *MotherShip* fica impossibilitado de fazer qualquer outra movimentação.
- Destrói uma embarcação inimiga movendo uma das suas numa direção obliqua.

A posição inicial da *MotherShip* é no centro do tabuleiro. De seguida o jogador **amarelo** deverá escolher as posições iniciais das suas 12 embarcações dentro dos limites do quadrado branco (fig. 1). Posteriormente a esta disposição inicial do jogador **amarelo**, o jogador **cinzento** irá distribuir as suas 20 embarcações na zona exterior ao quadrado branco (fig. 1). Em alternativa os jogadores podem escolher uma distribuição *default* como está apresentado na fig. 1.

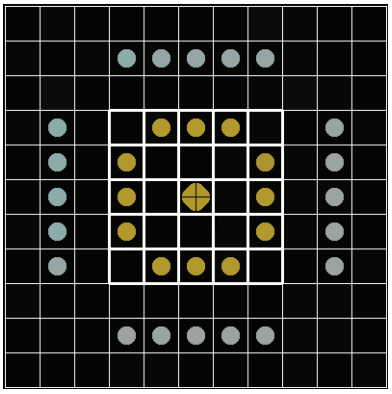


Figura 1

O Jogo acaba quando uma das seguintes condições é atingida:

- O jogador **amarelo** consegue fazer a sua *MotherShip* chegar ao limite do tabuleiro garantido a vitória para si.
- O jogador **cinzento** consegue destruir a *MotherShip*.

2 Representação do Estado do Jogo

Sendo o tabuleiro de 11x11 e as listas a melhor estrutura de dados a ser usada em Prolog escolhemos uma lista de listas para representar o tabuleiro. Cada elemento dessa lista será um espaço vazio ou uma peça.

Para distinguir as peças dos dois jogadores, estas têm valores diferentes, sendo que as peças do jogador cinzento correspondem a "1" no tabuleiro, as peças do jogador amarelo são "2" e a *MotherShip*, também esta do jogador amarelo corresponde ao "5". Os espaços vazios são "0". Portanto, a representação inicial do tabuleiro no caso de ter uma disposição default (Fig. 1) corresponderia à seguinte initial_board.

```
initial_board(
    [[0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,1,1,1,1,1,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0],
    [0,1,0,0,2,2,2,0,0,1,0],
    [0,1,0,2,0,0,0,2,0,1,0],
    [0,1,0,2,0,5,0,2,0,1,0],
    [0,1,0,2,0,0,0,2,0,1,0],
    [0,1,0,0,2,2,2,0,0,1,0],
    [0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,1,1,1,1,1,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0]]).
```

```
final_board(
    [[0,0,0,0,1,0,2,0,5,0,0],
    [0,0,0,1,0,0,0,0,0,0,0],
    [0,1,2,0,0,2,0,0,0,1,0],
    [0,0,1,0,0,0,0,0,1,0,0],
    [0,0,0,0,0,0,0,0,0,1,0],
    [0,0,0,0,0,0,0,0,0,0,0],
    [0,1,0,2,0,0,0,2,0,1,0],
    [0,1,0,0,1,1,2,0,0,0,0],
    [0,0,0,0,0,0,1,0,0,0,0],
    [0,0,0,0,0,0,0,1,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0]]).
```

O final_board é um exemplo de um possível final de jogo. Neste caso o jogador **amarelo** ganha visto que a *MotherShip* conseguiu atingir o limite superior do tabuleiro.

3 Visualização do Tabuleiro

Para mostrar o tabuleiro aos jogadores irá ser necessário haver uma impressão do tabuleiro na consola. O tabuleiro impresso na consola irá numerar as colunas e linhas para facilitar as jogadas dos jogadores. Irá também ser impresso uma bussola para ajudar os jogadores a tomarem a suas decisões (fig. 2).

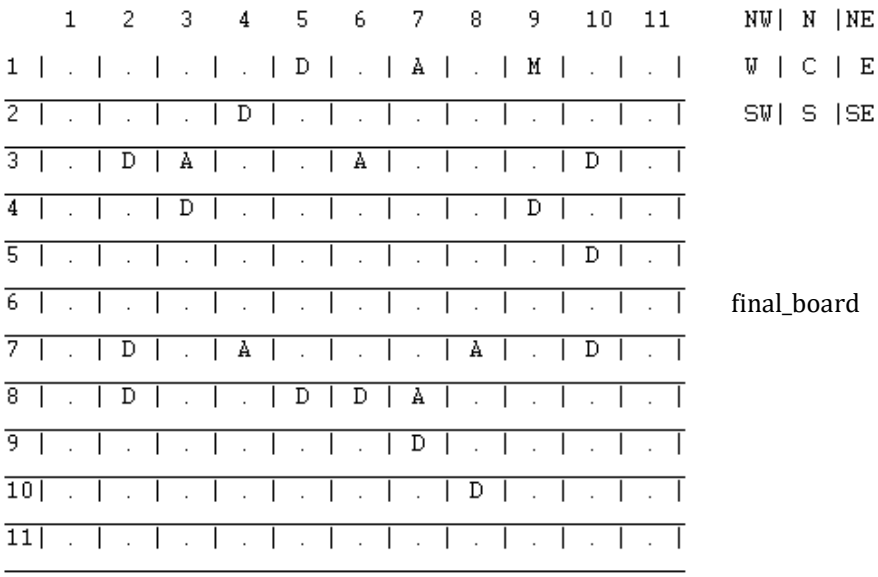
Quanto à simbologia usada a *MotherShip* irá ser simbolizada com um 'M', as embarcações do jogador **amarelo** com 'A' e as embarcações do jogador **cinzento** com 'D'.

```
NW| N |NE
W | C | E
SW| S |SE
```

Figura 2 - Bussola

```
    1   2   3   4   5   6   7   8   9   10  11
1 | . | . | . | . | . | . | . | . | . | . | . |
2 | . | . | . | D | D | D | D | D | . | . | . |
3 | . | . | . | . | . | . | . | . | . | . | . |
4 | . | D | . | . | A | A | A | . | . | D | . |
5 | . | D | . | A | . | . | . | A | . | D | . |
6 | . | D | . | A | . | M | . | A | . | D | . |
7 | . | D | . | A | . | . | . | A | . | D | . |
8 | . | D | . | . | A | A | A | . | . | D | . |
9 | . | . | . | . | . | . | . | . | . | . | . |
10| . | . | . | D | D | D | D | D | . | . | . |
11| . | . | . | . | . | . | . | . | . | . | . |
```

initial_board



Esta impressão vai ser possível correndo o seguinte código sobre as listas previamente apresentadas:

```
printTable(Table):-
    nl,
    printFirstLine(_),
    nl,nl,
    printLines(1,Table),
    nl.
printLines(_,[]).

printLines(1,[Lin|Resto]):-
    write(1), write(' |'),printLine(Lin), write('    W | C | E'),
    nl,
    printUnderscores(_),nl,
    printLines(2,Resto).

printLines(2,[Lin|Resto]):-
    write(2), write(' |'),printLine(Lin), write('    SW| S |SE'),
    nl,
    printUnderscores(_),nl,
    printLines(3,Resto).

printLines(N,[Lin|Resto]):- N < 10,N > 2,
    write(N), write(' |'),printLine(Lin), nl,
    printUnderscores(0),nl,
    N2 is N + 1,
    printLines(N2,Resto).
```

```
83  printLines(N,[Lin|Resto]):- N >= 10,
84      write(N), write('|'),printLine(Lin), nl,
85      N2 is N + 1,
86      printUnderscores(0),nl,
87      printLines(N2,Resto).
88
89  printLine([]).
90  printLine([El|Resto]):-
91      writePiece(El),
92      printLine(Resto).
93
94  printUnderscores(_):-write('_____________________________________________').
95
96  printFirstLine(_):-write('    1   2   3   4   5   6   7   8   9   10  11     NW| N |NE').
97
98  writePiece(0):-write(' . |').
99  writePiece(1):-write(' D |'). %%%% Defesa %%%%
100 writePiece(2):-write(' A |'). %%%% Ataque %%%%
101 writePiece(5):-write(' M |'). %%%% Objetivo %%%%
102
```

Parte 2

4 Movimentos

Para movimentar as peças vão ser pedidas aos jogadores as coordenadas originais da peça em questão e a direção a ser tomada pela peça.

MovePiece(Table,X,Y,Dir,Player,NumPlays)

(NumPlays: numero de jogadas possíveis, partindo do principio que uma moviementação obliqua/*MotherShip* vale 2 jogadas)

MovePiece vai ser responsável por confirmar as seguintes condições:

1. Confirmar que existe uma peça em (X,Y) e que essa peça pertence ao Jogador.
2. Confirmar que a direção é válida (N, S, E, W,NE,NW,SE,SW).
3. Confirmar que se na posição final não se encontra uma peça amigável.
4. Confirmar que o movimento não ultrapassa o número de Jogadas ainda possíveis.

Fontes: https://en.wikipedia.org/wiki/Breakthru_(board_game)