Crab Stack

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo:

António Maria Aires Pereira Teixeira de Melo	up201403053
Mónica Ariana Ribeiro Fernandes	up201404789

1. O Jogo Crab Stack

1.1. História

O jogo Crab Stack foi publicado em 2015. O jogo foi desenvolvido pelo designer Henri Kermarrec e pelo artista Stéphane Escapa. Publicado inicialmente pela Blue Orange Games e Foxgames (Poland).

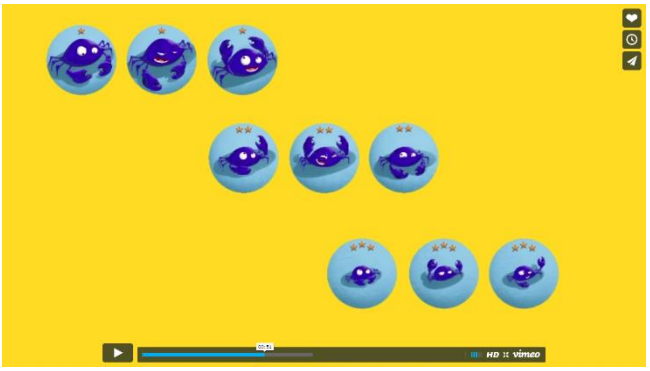
É um jogo de estratégia abstrata destinado a crianças com mais de 8 anos de idade, em que o tempo médio de jogo é de 20 minutos.

1.2. Regras do Jogo

O jogo consiste em imobilizar os caranguejos do adversário ao ficar sobre eles no tabuleiro.

O jogo pode ser jogado por 2,3 ou 4 pessoas. Consoante o número de pessoas o tabuleiro varia, isto é, para 2 pessoas só são utilizadas as rochas amarelas centrais, para 3 pessoas são juntam-se as 9 rochas pretas e para 4 juntam-se as restantes rochas (9 vermelhas) ao jogo.

Cada jogador é representado por uma cor.



Neste trabalho iremos utilizar o modo de jogo de 2 jogadores.

Existem 3 tipos de peças-caranguejo: pequenos, médios grandes e cada jogador recebe 9 peças, 3 de cada tipo.

Os caranguejos pequenos podem mover-se três casas, os médios duas e os grandes apenas uma.

Além disso os caranguejos ao moverem-se só podem ficar em casas em que o caranguejo que já la está seja do seu tamanho ou menor. Ou seja, o caranguejo pequeno só pode ficar em cima de caranguejos pequenos, o médio em cima de caranguejos médios e pequenos e o grande em cima de todos os outros caranguejos.

2. Representação do Estado do Jogo

O jogo é representado por uma lista de listas. O tabuleiro de jogo tem formato geométrico hexagonal o que tornou a sua representação mais difícil e por isso utilizamos espaços e barras de modo a tornar a sua forma um hexágono.

No entanto as casas interiores do tabuleiro são representadas como quadrados para tornar o desenvolvimento do jogo mais fácil e prático.

```
/* Empty Board */
empty_board([
           [a, a, a, a, b1, a, a, a, a, a, a, a, a, a, a, a, b2],
           [a, a, a, b1, a, a, e, e, e, a, a, b2],
           [a, a, b1, a, a, e, e, e, e, a, b2],
           [a, b3, a, a, e, e, c, e, e, a, b3],
           [a, a, b2, a, a, e, e, e, e, a, b1],
           [a, a, a, b2, a, a, e, e, e, a, a, b1],
           [a, a, a,  a, b2, a, a, a, a, a, a, a, a, a, a, a, b1]
          ]).
```

Os predicados para visualização do tabuleiro estão enunciados da seguinte forma:

```
display:-final_game_board(T), display_top, display_board(T).

display_board([]):-display_bottom.
display_row([]).


display_top:- write('      ___________'), nl.
display_bottom:- write('     -----------').

display_board([L1|Le]):-
        display_row(L1),
        nl,
        display_board(Le).

display_row([E|Ee]):-
        translate(E,V),
        write(V),
        display_row(Ee).
```

Optamos por alterar no próprio predicado ‘display’ a chamada do tabuleiro pretendido e não na consola do SICStus.

Utilizamos um predicado principal, o ‘display’ que chama os predicados ‘board’ (lista de listas), ‘display_top’ que representa a parte de cima do hexágono e ‘display_board’ que chama o tabuleiro e efetua a sua representação.

Utilizamos também algumas variáveis para criar o nosso tabuleiro que traduzimos posteriormente pelo predicado ‘translate’:

```
%a - empty space  ' '                    translate(a,  ' ').
%e - empty space initial                 translate(e,  'e  ').
%b1 - /                                  translate(b1, '/').
%b2 - \                                  translate(b2, '\\').
%b3 - |                                  translate(b3, '|').
%c - central hole '  '                   translate(c,  '  ').
%o - empty space after game start        translate(o,  'O  ').

%s1 - player1 piece small                translate(s1, '1s ').
%m1 - player1 piece medium               translate(m1, '1m ').
%l1 - player1 piece large                translate(l1, '1L ').

%s2 - player1 piece small                translate(s2, '2s ').
%m2 - player1 piece medium               translate(m2, '2m ').
%l2 - player1 piece large                translate(l2, '2L ').
```

3. Visualização no tabuleiro

O tabuleiro é representado por uma lista de listas.

Lista do tabuleiro vazio e respetiva visualização na consola (modo 2 jogadores):

```
    ___________
   /           \
  /  e  e  e    \
 /  e  e  e  e   \
|  e  e    e  e  |
 \  e  e  e  e   /
  \  e  e  e    /
   \           /
    -----------
```

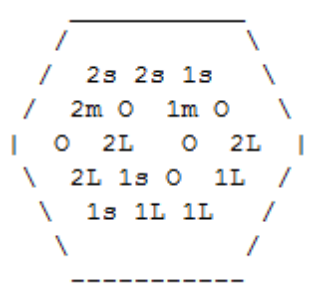
```
/* Empty Board */
empty_board([
          [a, a, a, a, b1, a, a, a, a, a, a, a, a, a, a, a, b2],
          [a, a, a, b1, a, a, e, e, e, a, a, b2],
          [a, a, b1, a, a, e, e, e, e, a, b2],
          [a, b3, a, a, e, e, c, e, e, a, b3],
          [a, a, b2, a, a, e, e, e, e, a, b1],
          [a, a, a, b2, a, a, e, e, e, a, a, b1],
          [a, a, a,  a, b2, a, a, a, a, a, a, a, a, a, a, b1]
         ]).
```

Lista do tabuleiro inicial de jogo (exemplo) e respetiva visualização na consola:

```
    ___________
   /           \
  /  1s 1s 1s   \
 /  1m 1m 1m 1L  \
|  1L 1L   2s 2s  |
 \  2s 2m 2m 2m  /
  \  2L 2L 2L   /
   \           /
    -----------
```

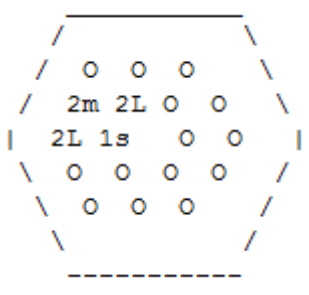
```
/* Initial Random Board */
initial_board([
          [a, a, a, a, b1, a, a, a, a, a, a, a, a, a, a, a, b2],
          [a, a, a,  b1, a, a, s1, s1, s1, a, a, b2],
          [a, a, b1, a, a, m1, m1, m1, l1, a,  b2],
          [a, b3, a, a, l1, l1, c, s2, s2, a, b3],
          [a, a, b2, a , a, s2, m2, m2, m2, a, b1],
          [a, a, a,  b2, a, a, l2, l2, l2, a, a, b1],
          [a, a, a, a, b2, a , a, a, a, a, a, a, a, a, a, b1]
         ]).
```

Lista do tabuleiro de jogo (exemplo do tabuleiro no decorrer do jogo) e respetiva visualização na consola:



```
/* Game Board */
game_board([
           [a, a, a, a, b1, a, a, a, a, a, a, a, a, a, a, a, b2],
           [a, a, a, b1, a, a, s2, s2, s1, a, a, b2],
           [a, a, b1, a, a, m2, o, m1, o, a, b2],
           [a, b3, a, a, o, l2, c, o, l2, a, b3],
           [a, a, b2, a, a, l2, s1, o, l1, a, b1],
           [a, a, a, b2, a, a, s1, l1, l1, a, a, b1],
           [a, a, a, a, b2, a, a, a, a, a, a, a, a, a, a, a, b1]
          ]).
```

Lista do tabuleiro no fim do jogo e respetiva visualização na consola:



```
/* Final Game Board */
final_game_board([
           [a, a, a, a, b1, a, a, a, a, a, a, a, a,  a, a, a, b2],
           [a, a, a, b1, a , a, o, o, o, a, a, b2],
           [a, a, b1, a , a, m2, l2, o, o, a, b2],
           [a, b3, a, a, l2, s1, c, o, o, a, b3],
           [a, a, b2, a, a, o, o, o, o, a, b1],
           [a, a, a, b2, a, a, o, o, o, a, a, b1],
           [a, a, a, a, b2, a, a, a, a, a, a, a, a, a, a, a, b1]
          ]).
```

Neste jogo o jogador 1 perdeu pois não consegue efetuar mais nenhum movimento. Pois todos os restantes caranguejos do jogo são de tamanho maior que o seu e por isso não pode mover-se para cima deles.

4. Movimentos

Alguns dos predicados que iremos usar no jogo:

- moveSmallCrab (**Board, player, X, Y**)

  Board – tabuleiro que está em jogo

  Player – player que efetua a jogada

  X – Coluna para a qual se vai mover se possível

  Y – Linha para a qual se vai mover se possível

- moveMediumCrab (**Board, player, X, Y**)
- moveLargeCrab (**Board, player, X, Y**)