Faculdade de Engenharia da Universidade do Porto

Green Skull

Programação Lógica

Relatório Intercalar

Tomás Gonçalves up201806763@fe.up.pt

Diogo Santos up201806878@fe.up.pt

1 - Descrição:

Introdução:

Green Skull é um jogo de tabuleiro disputado por dois jogadores. O material necessário para jogar é: um tabuleiro, peças redondas (8 verdes, 10 roxas e 10 brancas) e uma peça não redonda (geralmente em formato de crânio).

O tabuleiro é de formato triangular apresentando bordas de cores correspondentes aos três tipos de peças.

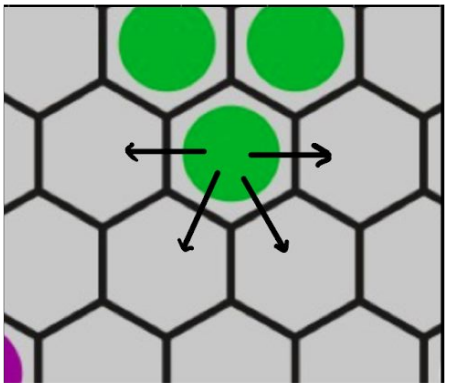
Os três tipos de peças representam diferentes criaturas mitológicas (embora a representação seja apenas abstrata). As peças verdes são chamadas de Zombies, as brancas de Orcs e as roxas de Goblins.

Instruções de jogo:

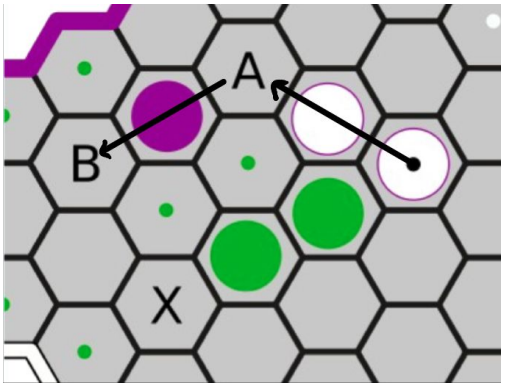
Ao iniciar o jogo, um jogador toma posse dos Goblins enquanto que o adversário controla os Orcs. O controlo das peças Zombie vai alternando entre os dois jogadores à medida que vão jogando (embora seja primeiro atribuído ao jogador que possui os goblins).

Ambos os jogadores podem efetuar um de dois movimentos possíveis:

-O primeiro é deslocar uma das suas peças para uma casa adjacente vazia.



-Pode também efetuar um ou mais saltos em linha reta sobre outra peça (incluindo peças do próprio jogador) caindo numa casa vazia. Estas peças pelas quais a peça vai passando por cima vão sendo removidas do tabuleiro. Se o jogador que possui o crânio optar por este segundo movimento, deve ceder o crânio ao seu adversário tendo agora ele posse dos Zombies.



O jogador que possui o crânio pode ainda mover uma das peças zombies após cada jogada com as suas respetivas peças.

O jogo termina quando **todas** as peças de um tipo forem comidas ou estiverem em contacto com a borda da mesma cor.

Pontuação:

Vence a espécie que obtiver mais pontos de acordo com a seguinte contagem:

- cada espécie recebe 2 pontos por cada peça que toque a borda da sua cor
- cada espécie recebe 1 ponto por cada peça capturada que não seja da sua cor

2 - Representação interna do Estado do jogo:

O tabuleiro triangular é representado através de uma lista de listas com a seguinte disposição:

```
initial([
        [' '],
        [' ', ' '],
        ['Z', ' ', 'Z'],
        ['Z', ' ', ' ', 'Z'],
        [' ', ' ', 'Z', ' ', ' '],
        [' ', ' ', 'Z', 'Z', ' ', ' '],
        ['O', ' ', ' ', 'Z', ' ', ' ', 'X'],
        ['O', 'O', ' ', ' ', ' ', ' ', 'X', 'X'],
        ['O', 'O', 'O', ' ', ' ', ' ', 'X', 'X', 'X'],
        ['O', 'O', 'O', 'O', ' ', ' ', 'X', 'X', 'X', 'X']]).
```

O tabuleiro é criado com as peças nas suas posições iniciais. As peças Zombie são representadas visualmente pela letra “Z”, Goblins por “O” e Orcs por “X”.

Os três tabuleiros seguintes representam estados de jogo intermédios. Podemos observar algumas peças movidas e outras retiradas do tabuleiro.

```
intermediate1([
    ['O'],
    [' ', 'X'],
    ['Z', 'X', 'O'],
    ['O', 'Z', ' ', ' '],
    [' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', 'O', ' ', ' ', ' '],
    [' ', 'X', ' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' ', 'O', ' ', ' '],
    [' ', 'X', ' ', ' ', 'Z', ' ', ' ', ' ', ' ', ' ']]).
```

```
intermediate2([
    ['O'],
    [' ', 'X'],
    ['Z', 'X', 'O'],
    ['O', 'Z', 'X', ' '],
    [' ', ' ', 'Z', 'Z', ' '],
    [' ', ' ', ' ', ' ', 'O', ' '],
    [' ', 'X', ' ', 'O', ' ', ' ', ' '],
    [' ', 'X', 'O', ' ', 'Z', ' ', ' ', ' '],
    ['X', ' ', 'Z', ' ', ' ', ' ', 'O', 'O', 'O'],
    [' ', 'X', ' ', ' ', 'Z', ' ', ' ', ' ', ' ', ' ']]).
```

```
intermediate3([
    [' '],
    ['Z', ''],
    [' ', ' ', 'O'],
    [' ', 'Z', 'X', ' '],
    [' ', ' ', 'Z', 'Z', ' '],
    [' ', ' ', ' ', ' ', 'O', ' '],
    [' ', '', ' ', 'O', ' ', ' ', 'X'],
    [' ', '', 'X', ' ', 'Z', ' ', ' ', ' '],
    ['X', 'O', '', ' ', ' ', ' ', 'O', 'O', 'O'],
    ['Z', 'X', ' ', ' ', ' ', ' ', 'O', ' ', ' ', 'X']]).
```

Esta última tabela mostra-nos o estado final do jogo. O jogo terminou pois todas as peças Zombie (“Z”) estão em contacto com a face do tabuleiro correspondente à sua cor (neste caso a base do triângulo).

```
final([
    ['O'],
    [' ', 'X'],
    [' ', 'X', 'O'],
    ['O', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', 'O', ' ', ' ', ' '],
    [' ', 'X', ' ', ' ', ' ', ' ', ' ', ' '],
    [' ', ' ', ' ', ' ', ' ', ' ', 'O', ' ', ' '],
    ['Z', 'X', ' ', ' ', 'Z', 'Z', ' ', ' ', ' ', ' ']]).
```

Guardamos a informação de qual o jogador que deve jogar através do facto que vamos alterando dinamicamente chamado player_turn e guarda uma string que indica qual é o jogador a jogar. Da mesma forma, representamos o detentor do crânio verde com o facto z_belongs_to que também contém uma string que nos diz qual dos dois jogadores controla os zombies nesse instante.

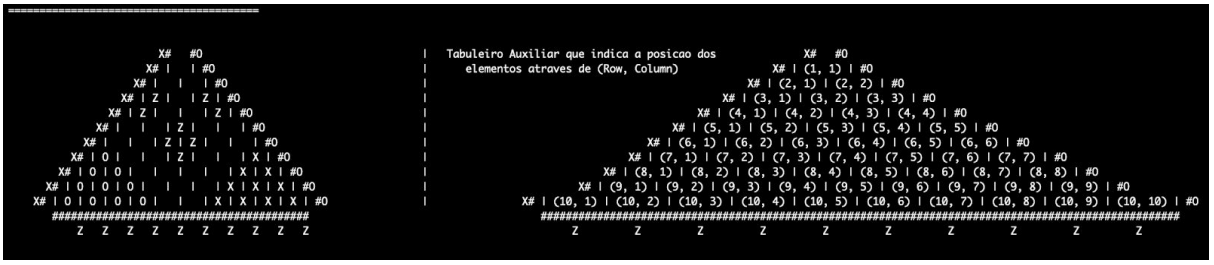
Em relação a peças eliminadas, temos também três factos que alternam dinamicamente: o_eliminated, z_eliminated e x_eliminated; estes factos contêm o número de peças eliminadas da sua espécie. Sempre que alguma destas peças é eliminada no respetivo facto, incrementamos o valor de peças eliminadas.

Em relação à representação das peças, são utilizadas strings como descrito anteriormente, cada uma situada na sua posição específica do tabuleiro.

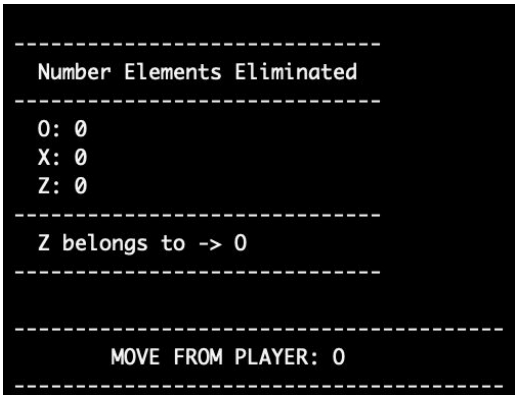
3 - Visualização do estado de jogo:

O jogo apresenta um segundo tabuleiro com índices para facilitar a escolha da peça a mover, tal como a casa para onde a mesma deve ser movida.

Na seguinte imagem é possível observar o estado inicial do jogo.



O jogo apresenta uma tabela indicativa do número de peças eliminadas de cada espécie. Indica também o jogador que possui a caveira assim como jogador que deve jogar no turno atual. Para efetuar um movimento, o jogador deve indicar as coordenadas da peça a mover, assim como as da casa para onde pretende mover a mesma.



Para podermos visualizar o tabuleiro temos de recorrer ao predicado “display_game/2” que recebe o tabuleiro e o jogador a jogar. Este predicado recorre a dois outros: “print_board/1” que recebe o tabuleiro e ao “print_score/0” que mostra as atuais pontuações. O predicado “print_board/1” recorre por sua vez a outro predicado “print_row1/1” que imprime a linha 1 este chama o “print_row2/1” que imprime a linha 2 e assim sucessivamente até à linha 10.

```
display_game(Board, _Player) :-
    print_board(Board),
    print_pontuations.


print_board(Board) :-
    format('\n\n=======================================\n', []),
    print_row1(Board).


print_score :-
    o_eliminated(O), z_eliminated(Z), x_eliminated(X), z_belongs_to(Player),
    format("-----------------------------\n", []),
    format("  Number Elements Eliminated \n", []),
    format("-----------------------------\n", []),
    format("  O: ~p                      \n", [O]),
    format("  X: ~p                      \n", [X]),
    format("  Z: ~p                      \n", [Z]),
    format("-----------------------------\n", []),
    format("  Z belongs to -> ~p                    \n", [Player]),
    format("-----------------------------\n\n", []).
```