Faculdade de Engenharia da Universidade do Porto
2021/22

Breakthrough Tanks

Programação Funcional e Lógica - Licenciatura em Engenharia Informática e Computação

Grupo 6 - Turma 6

Marta Cristina dos Santos Mariz	up201907020@edu.fe.up.pt
Miguel Norberto Costa Freitas	up201906159@edu.fe.up.pt

Identificação do trabalho e do grupo

Somos o grupo BreakthroughTanks_4.

Nome	up	Contribuição
Marta Cristina dos Santos Mariz	up201907020	50%
Miguel Norberto Costa Freitas	up201906159	50%

Instalação e execução

A instalação e execução são relativamente diretas. Qualquer sistema onde o SICStus esteja instalado corretamente é capaz de executar o programa.
Abre-se a aplicação SICStus, ou corre-se o comando SICStus no terminal e a seguir é necessário fazer consult da root do projeto do ficheiro breakthrough.pl

A seguir, para começar o jogo é apenas necessário correr o comando **play .** .

Descrição do jogo

O jogo consiste numa batalha entre dois exércitos de tanques. Ganha a equipa que chegar primeiro ao outro lado do tabuleiro, ou seja ultrapassou a frota adversária.
Movimento das peças é executado segundo o seguinte padrão:

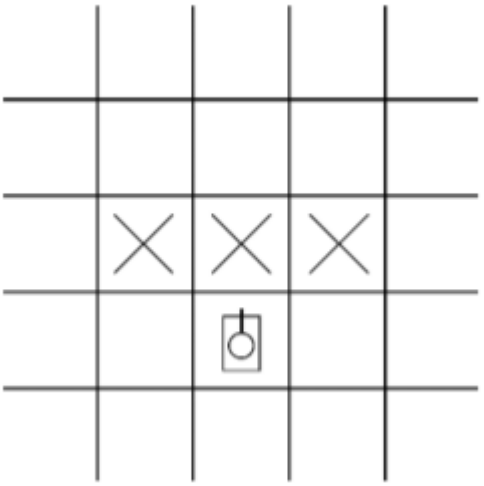

Figura 1: Representação dos movimentos possíveis das peças

Todas as peças, caso destruam uma peça inimiga, devem movimentar-se para a sua posição .

Existem 3 tipos de peças:

- **Tanque médio,** que pode destruir uma peça inimiga num espaço para a frente ou na diagonal para a frente, como está representado na figura.

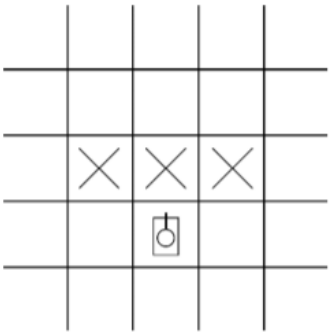

Figura 2: Representação da kill range possível do tanque médio

- **Tanque pesado,** que pode destruir uma peça localizada dois espaços na diagonal para a frente , e para a frente na mesma coluna. Se este se mover dois espaços destrói os tanques do adversário por onde passou.

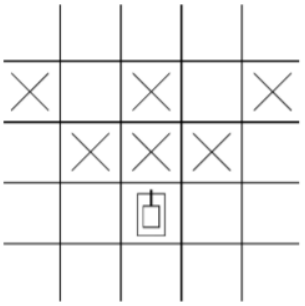

Figura 3: Representação da kill range possível do tanque pesado

- **Tanque destruidor,** que pode destruir um inimigo localizado um ou dois espaços para a frente.

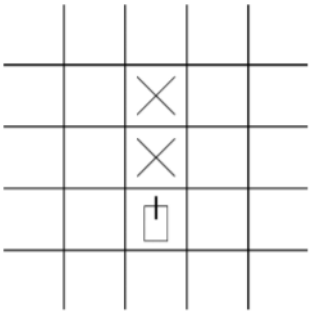

Figura 4: Representação da kill range possível do tanque destruidor

Quando deparado com a hipótese de “disparar” para uma casa ocupada por um oponente, o jogador pode decidir se pretende ou não realizar essa jogada. Se decidir realizar essa jogada, a peça do jogador destrói a peça do oponente e a peça move-se para essa posição. No entanto, não existe obrigatoriedade de capturar peças, como nas Damas.

Balas penetrantes

Não encontramos informação clara acerca de como implementar a destruição de uma peça inimiga que esteja entre a posição final e a posição atual de um tanque, pelo que se assume que se ela se encontra no caminho do “tiro” é também destruída.

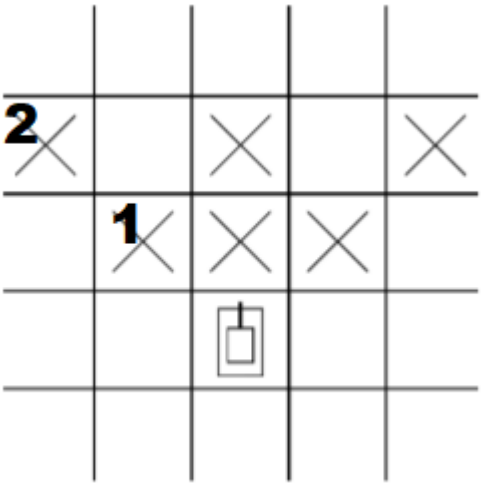

Figura 5: Exemplo do tiro penetrante

Por exemplo, no exemplo acima, se um inimigo se encontrar na posição 1 e o nosso tanque escolhe disparar para o quadrado 2, a peça inimiga situada em 1 será destruída igualmente.

Para recolher informações sobre as regras do jogo recorremos ao seguinte link: https://s3.amazonaws.com/geekdo-files.com/bgg283471?response-content-disposition=inline%3B%20filename%3D%22Breakthrough-Tanks-Rulebook.pdf%22&response-content-type=application%2Fpdf&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAJYFNCT7FKCE4O6TA%2F20220122%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220122T202720Z&X-Amz-SignedHeaders=host&X-Amz-Expires=120&X-Amz-Signature=1988bcb7e73813f04aa33823cbbc6a1f235883971404d9987a1d8c9cf2e57b3a

Lógica do jogo

O tabuleiro do jogo foi implementado como uma lista de listas, cada lista corresponde a uma célula do jogo que contém a linha, a coluna e o estado da célula (ocupada por peças branca ou pretas, ou vazia).

```
translate(0, '  .  ').
translate(1, 'B_MT ').
translate(2, 'B_HT ').
translate(3, 'B_TD ').
translate(4, 'W_MT ').
translate(5, 'W_HT ').
translate(6, 'W_TD ').
```

Figura 5: Representação de cada tipo de peça

O cumprimento das regras de movimento das variadas peças do jogo é determinado pelo predicado check_valid(+[CI-RI, CF-RF] ,+Player , +GameState,+Piece, -ListOfSpacesToChange), que verifica se a posição CF-RF (coluna_final, row_final) constitui uma posição válida para a determinada Piece deste Player que se encontra na posição atual de CI-RI(coluna inicial, row inicial) , retornando ainda uma lista que constitui um conjunto de posições pelas quais a peça passa no seu movimento, que é depois utilizada para tornar esses quadrados livres.

O final do jogo é determinado através do predicado game_over(GameState, Player, Winner) que verifica se o jogador das peças pretas chegou, com alguma peça, à linha 1, ou se o jogador das peças brancas chegou, com alguma peça, à linha 8.

4.1 Representação do Estado do Jogo

A representação do tabuleiro é feita através de uma lista de listas (células) (Fig. 6).

```
initialBoard([[1,3,3,2,2,3,3,1],
              [1,1,1,1,1,1,1,1],
              [0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0],
              [4,4,4,4,4,4,4,4],
              [4,6,6,5,5,6,6,4]]).
```

Figura 6: Representação do board inicial

4.2 Visualização do Estado do Jogo

A visualização do tabuleiro é feita através do predicado displayGame.

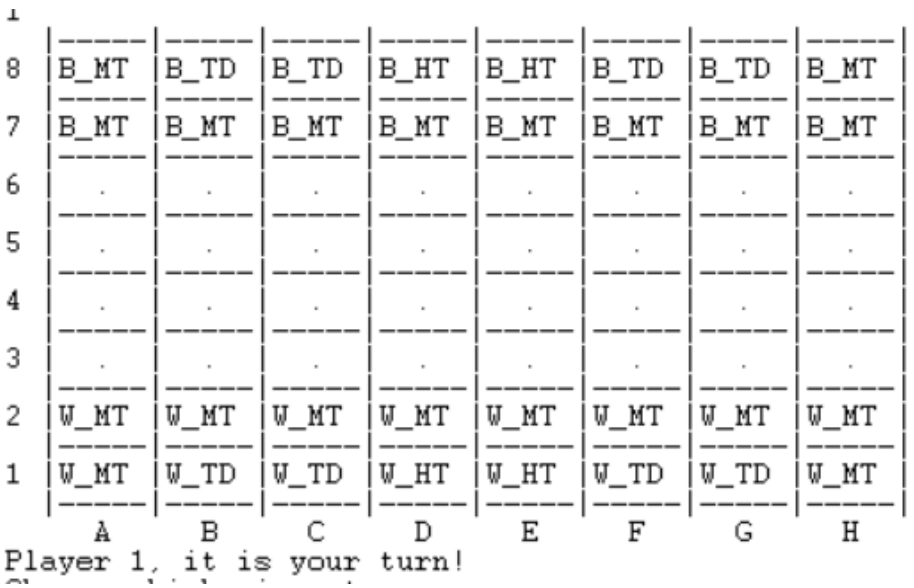

Figura 7: Exemplo de visualização do tabuleiro inicial

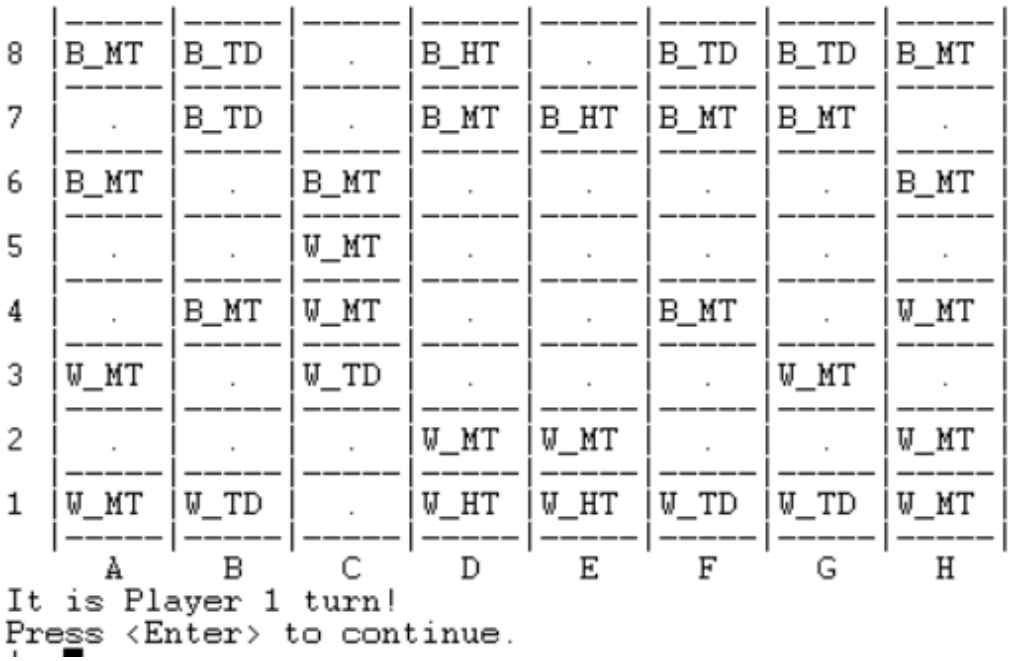

Figura 8: Exemplo de visualização de um estado de jogo intermédio do tabuleiro

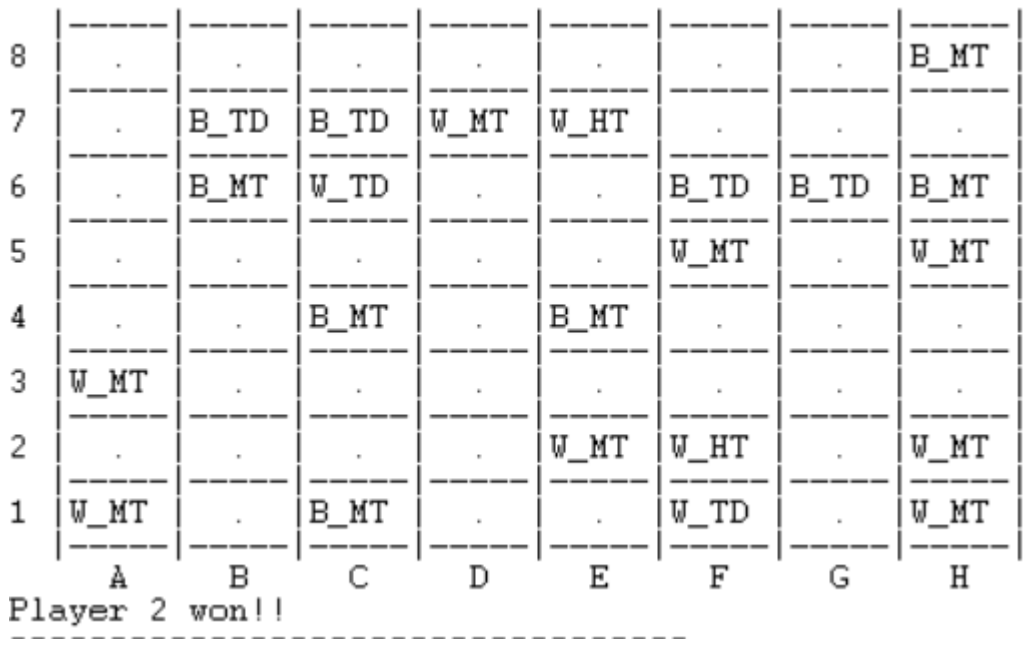

Figura 9: Exemplo de visualização de um estado de jogo final do tabuleiro

4.3 Validação de Jogadas

A validação de uma jogada é feita através do predicado check_valid(+[CI-RI, CF-RF] ,+Player , +GameState,+Piece, -ListOfSpacesToChange) que avalia como true se a jogada é válida para o jogador segundo as regras definidas acima.

4.4 Execução de Jogadas

A execução de uma jogada é feita através do predicado move([CI-RI, CF-RF], Player , GameState, UpdatedGameState) que cria um novo tabuleiro alterando o valor da célula de destino em CF-RF para o valor correspondente à peça localizada em CI-RI, para além de se encarregar ainda de apagar ainda todas as peças localizadas entre CI-RI e CF-RF, incluindo o próprio CI-RI (peças contidas no array ListOfSpacesToChange retornado da função check_valid).

4.5 Lista de Jogadas Válidas

A listagem de todas as jogadas válidas para uma peça é feita através do predicado valid_moves_piece(+GameState,+ Player,+ [CI-RI], -ListOfValidMoves) que devolve em ListOfValidMoves, para a peça localizada na posição CI-RI, todas as posições para as quais esta se pode mover. Através da aplicação do predicado findall/3, foi desenvolvida ainda uma função que retorna todas as jogadas válidas para todas as peças no tabuleiro de uma dado Player, o predicado valid_moves(+GameState, +Player, -ListOfMoves).

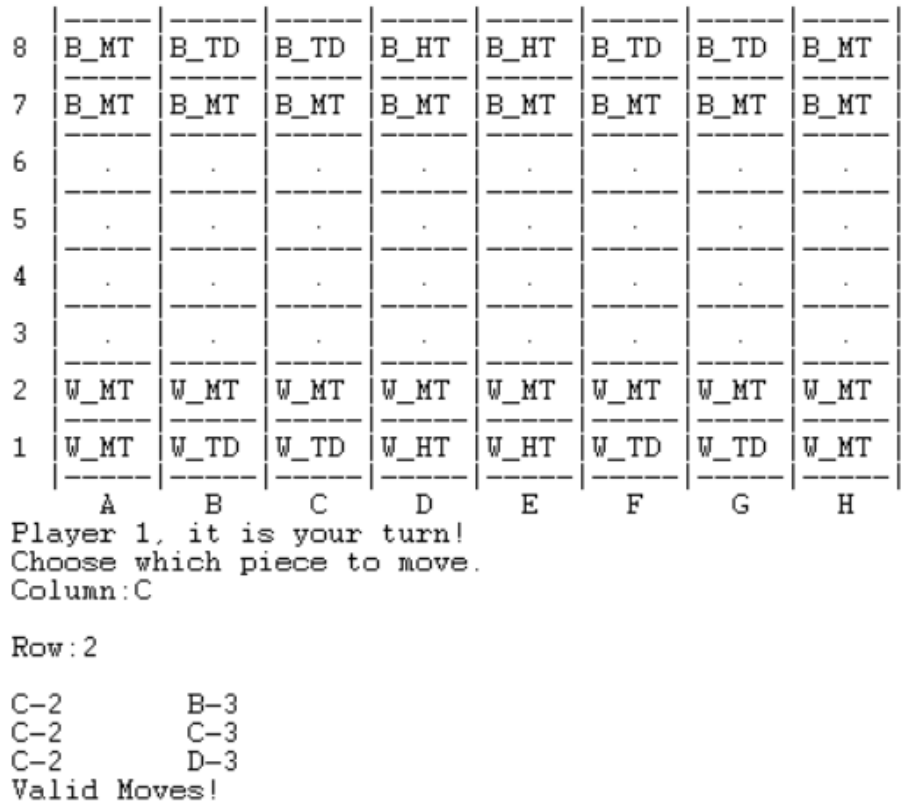

Figura 10: Exemplo das valid moves

4.6 Avaliação do Tabuleiro

O predicado value(+GameState, +Player, -Value) retorna uma avaliação do atual estado do tabuleiro para um dado Player. Este é calculado com a diferença do valor de peças de um player em relação ao outro, atribuindo os seguintes valores a cada peça (Heavy Tanks são avaliados como valendo 5 pontos, Medium Tanks valem 1 ponto e Tank Destroyers valem 2 pontos).

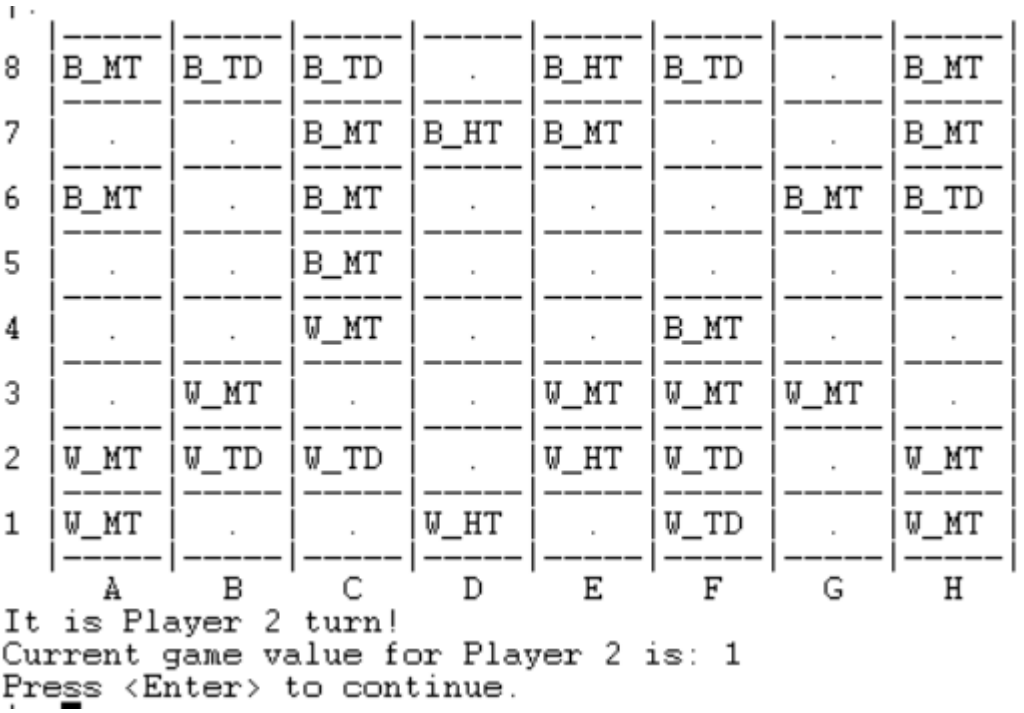

Figura 11: Exemplo da avaliação do tabuleiro

(O player 2 comeu um medium tank do player 1 e está portanto com +1 de valor em relação ao player 1)

4.7 Final do Jogo

O fim do jogo é determinado através do predicado game_over(+GameState, +Player, -Winner), que verifica se alguma peça dos jogadores alcançou a linha correspondente ao seu extremo oposto.

4.8 Cálculo da Jogada do Computador

Para o computador, as jogadas são decididas escolhendo uma jogada aleatória resultante do predicado valid_moves(+GameState, +Player, -ListOfMoves), que como descrito acima retorna uma lista com todas as jogadas possíveis para dado Player.

Conclusão

Em geral achamos que foi um ótimo projeto e fomos capazes de perceber melhor as capacidades e qualidades da linguagem de programação Prolog.

Não encontramos falhas ou issues na execução do projeto. Cada peça tem um movimento bem definido e todos os inputs do jogador são analisados. Para cada um verifica-se se corresponde a uma jogada válida, caso não corresponda, é repetido o pedido de input. Idealmente, teríamos implementado um segundo nível de dificuldade, avaliando o estado de jogo e escolhendo a melhor opção.

Outra possível melhoria seria implementar o que nas regras é descrito como “advanced mode”, que consiste em permitir ao jogador customizar o tabuleiro, nomeadamente escolher o tamanho e como organizar as diferentes peças dentro das respectivas duas filas de cada jogador.
Em suma, consideramos ter feito um bom trabalho e tivemos uma experiência positiva.

Bibliografia

- https://s3.amazonaws.com/geekdo-files.com/bgg283471?response-content-disposition=inline%3B%20filename%3D%22Breakthrough-Tanks-Rulebook.pdf%22&response-content-type=application%2Fpdf&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAJYFNCT7FKCE4O6TA%2F20220122%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220122T202720Z&X-Amz-SignedHeaders=host&X-Amz-Expires=120&X-Amz-Signature=1988bcb7e73813f04aa33823cbbc6a1f235883971404d9987a1d8c9cf2e57b3a