

*Fabrik*

***Relatório Final***

Programação em Lógica

Grupo: Fabrik\_2

*(12 de novembro de 2017)*

Bárbara Sofia Lopez de Carvalho Ferreira da Silva  **up201505628**@fe.up.pt

Julieta Pintado Jorge Frade **up201506530**@fe.up.pt

Resumo

Resumo sucinto do trabalho com 150 a 250 palavras (problema abordado,

objetivo, como foi o problema resolvido/abordado, principais resultados e conclusões).

Índice

|  |  |
| --- | --- |
| **Introdução - julieta** | 0 |
| **O Jogo: *Fabrik*** | 0 |
| **Lógica do Jogo** | 0 |
| Representação do Estado do Jogo | 0 |
| Visualização do Tabuleiro | 0 |
| Lista de Jogadas Válidas - julieta | 0 |
| Execução de Jogadas - sofia | 0 |
| Avaliação do Tabuleiro - julieta | 0 |
| Final do Jogo - sofia | 0 |
| Jogada do Computador - julieta | 0 |
| **Interface com o Utilizador - sofia** | 7 |
| **Conclusões - sofia** | 8 |
| **Bibliografia** | 0 |
| **Anexos** | 0 |

10-15 páginas

Introdução

Texto.

O Jogo: Fabrik

*Fabrik* é um jogo de tabuleiro criado em agosto de 2017. Consiste no conceito de duas figuras neutras, denominadas por “*worker*” ou “*arbeiter*”, estas são acessíveis aos dois jogadores, que em colaboração determinam os espaços em que os mesmos podem jogar, ou seja, onde podem deixar a sua peça em cada ronda.

A condição vencedora é um dos jogadores obter 5 das suas peças em linha, seja esta horizontal, vertical ou diagonal. Esta condição foi deliberadamente selecionada, pois é um dos conceitos mais utilizados em jogos clássicos e contemporâneos. Na verdade, as regras de colocação restrita no *Fabrik* ajudam a superar a vantagem do primeiro jogador, que existem em muitos outros jogos, como *Gomoku* e, assim, *Fabrik* está de certa forma relacionado com *Renju*.



**Figura 1**: tabuleiro do jogo

O material necessário para o jogo é um tabuleiro quadrado com 11x11 espaços, uma grande quantidade de peças brancas e pretas, e duas peças vermelhas, chamadas “*workers*”.

**Preparação**

Inicialmente, o tabuleiro está vazio. O jogador das peças pretas começa por colocar um dos *workers* em qualquer espaço. De seguida, o jogador das peças brancas coloca o outro *worker* num espaço ainda livre.

O jogador das peças pretas decide quem joga primeiro. Este deverá colocar uma peça da sua cor de acordo com as regras descritas mais abaixo. Após o jogo estar preparado, os jogadores deverão alternar entre si.

**Objetivo**

Os jogadores ganham assim que um deles conseguir obter uma linha de, pelo menos, 5 peças da sua cor seguidas, na ortogonal ou diagonal.

**Desenvolvimento**

Em cada ronda o jogador poderá mover um dos *workers* e colocá-lo num outro espaço vazio, este passo é opcional. Depois, deverá colocar uma das suas peças em qualquer linha de interseção de um dos *workers*, chamadas **linhas de vista**. Estas linhas radiam da posição do *worker* numa direção ortogonal e diagonal, enquanto existem espaços vazios. Assim que uma linha de vista alcançar uma peça, esta acaba nessa posição.

Em certos casos, é possível que os *workers* fiquem localizados na mesma linha ortogonal ou diagonal, assim, todos os espaços entre eles são considerados pontos de interseção, desde que estejam vazios.



**Figura 2**: os pontos de interseção das linhas de visão dos *workers* determinam onde as peças podem ser colocadas.

**Fim**

O jogador pede o jogo assim que não consiga colocar nenhum dos dois *workers* numa posição em que seja possível inserir uma peça nova.

Assim, ganha o jogo aquele que conseguir colocar, pelo menos, 5 peças da sua cor seguidas numa direção ortogonal ou diagonal.



**Figura 3**: fim do jogo em que o jogador com as peças pretas ganha.

**Source:** <https://spielstein.com/games/fabrik>

POR ISTO NA BIBIOSOJEO

Lógica do Jogo

Representação do Estado do Jogo

**Situação Inicial**

initialBoard([

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty]

]).

**Situação Intermédia**

midBoard([

[empty,empty,empty,empty,**white**,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,**red**,empty],

[empty,**white**,empty,empty,empty,empty,empty,empty,**white**,empty,empty],

[empty,empty,empty,empty,empty,**white**,empty,empty,**white**,empty,empty],

[empty,empty,empty,**black**,empty,**black**,**black**,**black**,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,**white**,empty,empty],

[empty,empty,empty,**black**,empty,**white**,empty,empty,empty,empty,empty],

[empty,empty,empty,**black**,empty,**red**,**black**,empty,empty,empty,empty],

[empty,**black**,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,**black**,empty,empty,empty,empty,empty,**white**,empty,empty]

]).

**Situação Final**

finalBoard([

[empty,empty,empty,empty,**white**,empty,empty,empty,empty,empty,empty],

[empty,empty,empty,empty,**white**,empty,empty,empty,empty,empty,empty],

[empty,**white**,empty,empty,empty,empty,empty,empty,**white**,empty,empty],

[empty,empty,empty,**black**,empty,**white**,empty,empty,**white**,empty,empty],

[empty,empty,empty,**black**,empty,**black**,**black**,**black**,empty,empty,empty],

[empty,empty,empty,**black**,empty,empty,empty,empty,**white**,empty,empty],

[empty,empty,empty,**black**,empty,**white**,empty,empty,empty,empty,empty],

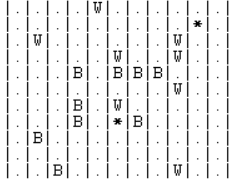
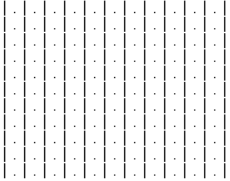
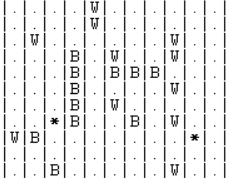
[empty,empty,**red**,**black**,empty,empty,**black**,empty,**white**,empty,empty],

[**white**,**black**,empty,empty,empty,empty,empty,empty,empty,**red**,empty],

[empty,empty,empty,empty,empty,empty,empty,empty,empty,empty,empty],

[empty,empty,**black**,empty,empty,empty,empty,empty,**white**,empty,empty]

]).



**Fig. 3** Situação Final vista na consola

**Fig. 2** Situação Intermédia vista na consola

**Fig. 1** Situação Inicial vista na consola

Visualização do Tabuleiro

Segue-se o código que será utilizado para mostrar o tabuleiro na consola:

*/\*pieces symbols\*/*

symbol(empty,S) :- S='.'.

symbol(black,S) :- S='B'.

symbol(white,S) :- S='W'.

symbol(red,S) :- S='\*'.

*/\*print board\*/*

printBoard([Head|Tail]) :-

write('|'),

printLine(Head),

nl,

printBoard(Tail).

printBoard([ ]).

*/\*print line \*/*

printLine([Head|Tail]) :-

symbol(Head,S),

write(S),

write('|'),

printLine(Tail).

printLine([ ]).

O output produzido está ilustrado nas imagens da página anterior.

Lista de Jogadas Válidas

Texto.

Execução de Jogadas

Texto.

Avaliação do Tabuleiro

Texto.

Final do Jogo

Após cada jogada é fundamental verificar o estado do jogo, pois, a qualquer momento, um dos jogadores pode ganhar ou ocorrer um empate, isto é, caso não exista mais nenhum espaço válido para colocar uma nova peça ou caso não exista um espaço livre no tabuleiro. De forma a poder verificar todos estes casos, foi implementado o predicado **checkGameState**, que recebe o tipo de peça do jogador que acabou de jogar e o tabuleiro atual. Este predicado vai chamar 6 outros predicados, e caso algum deles se verifique, o jogo acaba.

checkGameState(Player, Board) :-

((checkVictory(Player, 'Row', Board), write('You won!'));

(checkVictory(Player, 'Column', Board), write('You won!'));

(checkVictory(Player, 'DiagonalDown', Board), write('You won!'));

(checkVictory(Player, 'DiagonalUp', Board), write('You won!'));

(checkFullBoard(Board), write('Woops, no more space left! It is a draw!'));

(checkValidSpots(Board, 0, 0, Result), Result =:= 0, write('Woops, no more space left! It is a draw!'))).

Como o nome indica, o predicado **checkVictory** verifica se o jogador ganhou o jogo. Em particular, existem 4 condições vencedoras: o jogador ter 5 peças seguidas na mesma linha, coluna ou diagonal. Deste modo, cada predicado verifica se existe esse padrão no tabuleiro.

Quanto ao predicado **checkFullBoard**, este verifica se não existem mais espaços livres, empty, em todo o tabuleiro. Já o predicado **checkValidSpots** verifica se não existe nenhum espaço válido, ou seja, dentro das linhas de visão de cada worker, para o próximo jogador colocar a sua peça. Portanto, se algum destes dois predicados se satisfizer, o jogo acaba com empate.

Jogada do Computador

Texto.

Interface com o Utilizador

Texto.

Conclusões

Texto.

Bibliografia

Texto.

Anexos