Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Informática

Programação em Lógica

Pushee Pieces

Relatório Final

Autores:

Fernandes, Eduardo ei12130@fe.up.pt,
Coutinho, José Ricardo ei12161@fe.up.pt

11 de Novembro de 2014

Resumo

O problema abordado foi de desenvolver um jogo de tabuleiro, o Pushee Pieces, na linguagem Prolog utilizando o SICStus Prolog. O objetivo deste problema é de aprender a programar num novo paradigma que é a programação em lógica. O problema foi abordado por uma fase inicial de leitura da documentação sobre as regras de jogo e de experimentação do jogo. A fase seguinte foi a de discussão e descoberta das formas possíveis de resolver as jogadas, aliadas a um determinado estilo de representação do estado de jogo. Finalmente foram tomadas decisões sobre a estrutura a optar e métodos a utilizar para resolver certos problemas. O jogo foi concluído com enorme sucesso.

Conteúdo

Capítulo 1

Introdução

O seguinte documento contem o relatório final do primeiro trabalho prático, o jogo de tabuleiro *Pushee Pieces*, do grupo 1 para a disciplina de PLOG do curso MIEIC da FEUP.
No seu conteúdo temos a introdução, lógica do jogo, interface com o utilizador, extras implementados, conclusões, bibliografia, anexos e glossário.

O motor de jogo *Pushee Pieces* foi desenvolvido utilizando o SICStus Prolog. O terminal do SICStus é utilizado para visualizar o estado de jogo e interagir com o utilizador.

Capítulo 2

O Jogo *Pushee Pieces*

Publicado em 2014 por *Eric Daryl Stevenson*, o *Pushee Pieces* é um jogo para dois jogadores, jogado num tabuleiro quadrangular e quadriculado de tamanho mínimo de 6x6. Na figura 2.1 podemos ver uma foto do criador do jogo.

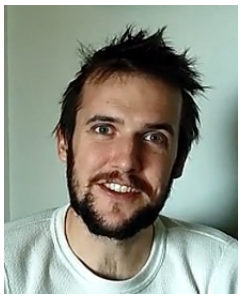

Figura 2.1: O criador do jogo, Eric Daryl Stevenson.

O **objetivo** do jogo é obter mais pontos que o outro jogador.

O **tabuleiro** é de tamanho arbitrário (mínimo 6x6) e com valores inteiros positivos aleatórios, com zero inclusive. Na figura 2.2 podemos ver exemplos de tabuleiros iniciais diferentes (sem peças).

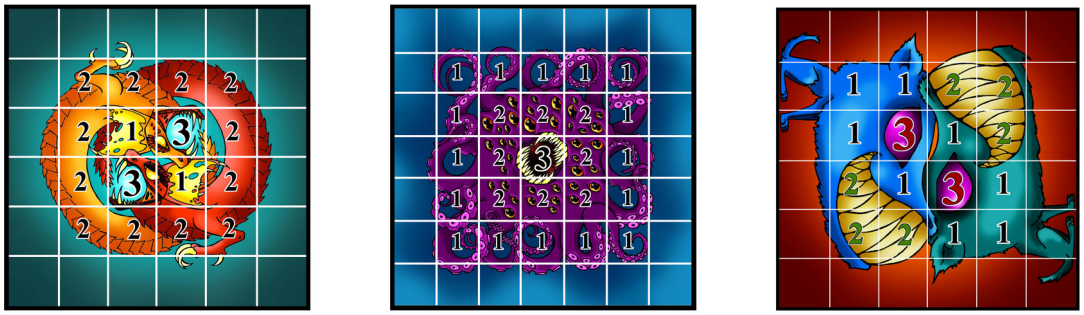

Figura 2.2: Três tabuleiros diferentes.

Cada **jogador** começa com 15 peças e têm alternadamente 15 turnos para colocar as suas peças, uma peça por turno, colocando assim todas as suas peças no tabuleiro.

As **peças** são retangulares e podem ser colocadas dentro de uma quadrícula, em uma de duas orientações possíveis, horizontal ou vertical. Na figura 2.3 podemos ver o aspeto das peças.

Cada **quadrícula** pode ou não ter um número associado, este número representa o valor dessa quadrícula. No caso de não existir número associado o valor da mesma é 0.

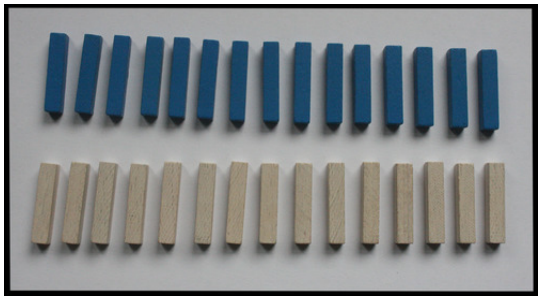

Figura 2.3: As peças do jogo.

A **orientação** da peça indica, respetivamente, que a peça pode empurrar na direção dos seus extremos, empurra na horizontal se colocada na horizontal e empurra na vertical se colocada na vertical. Na fig 2. podemos ver os diferentes tipos de orientação.

Figura 2.4: Diferentes tipos de orientação.

A **colocação** de uma peça implica **empurrar** em uma unidade, na direção respetiva, a peça ou conjunto contíguo de peças adjacentes. Ver exemplos na fig.2.5 e fig.2.6.

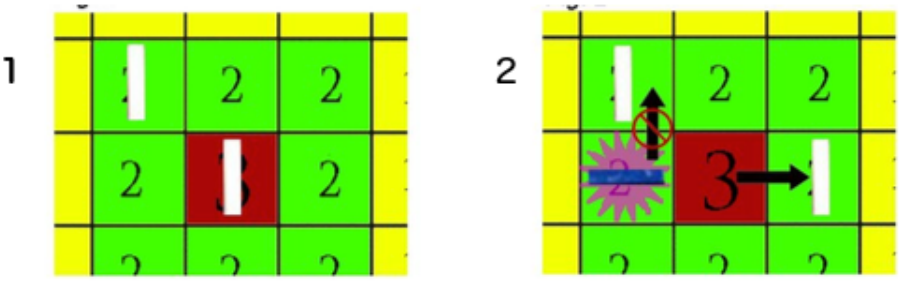

Figura 2.5: Colocação da peça e consequência.

As **peças empurradas** vão por sua vez repetir o processo de empurrar outras peças na sua respetiva orientação, até ao máximo de 3 cadeias de empurrões. No entanto, uma peça só pode ser empurrada apenas uma vez por turno. Ver exemplo na fig.2.7.

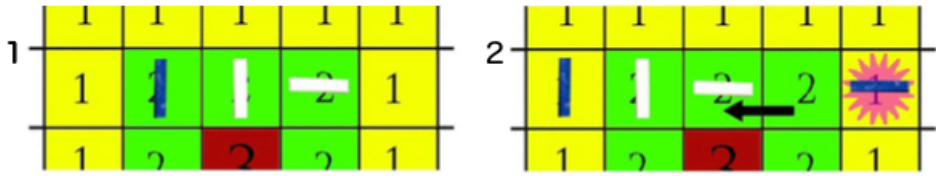

Figura 2.6: Empurrão de peças contiguas.

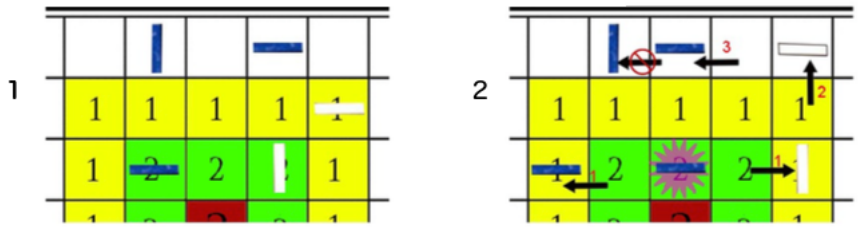

Figura 2.7: Empurrões em cadeia.

Se a **peça ou conjunto contíguo de peças adjacentes** estiverem contra uma extremidade do tabuleiro, na respetiva direção, então esta(s) não serão empurradas. Ver exemplo na fig.2.8.

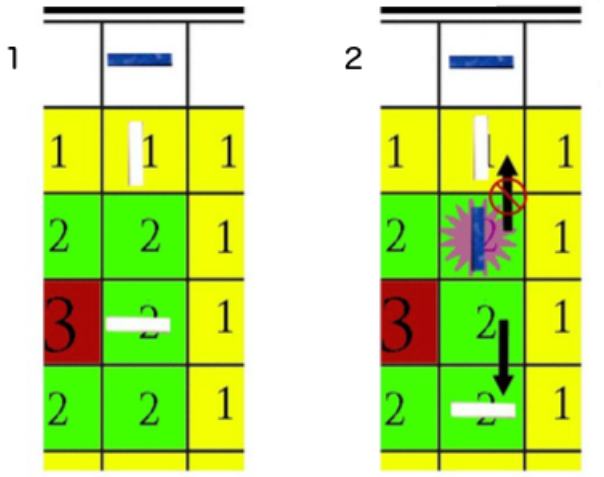

Figura 2.8: Limitações de empurrões.

Em cada **turno** pode haver no máximo três empurrões em cadeia, isto é, uma peça que foi empurrada vai consequentemente empurrar outras peças adjacentes, que ainda não foram afetadas nesse turno, e estas empurraram outras até atingir o número de empurrões em cadeia máximo ou até não ter mais peças para afetar.

O **fim de jogo** acontece quando o segundo jogador colocar a última peça e acontecerem as respetivas reações em cadeia. A pontuação de cada jogador é igual à soma do valores das quadrículas em que as peças desse jogador se encontram.

Capítulo 3

Lógica do Jogo

Resumidamente, o jogo apenas pede posições ao utilizador de onde colocar as peças. No entanto, as respetivas reações são resolvidas internamente pelo lógica do jogo evitando assim trabalho e potêncial batota do utilizador.

O jogo é iniciado com o predicado *play/3*, ver listing 7.1.

3.1 Representação do Estado do Jogo

O tabuleiro de jogo é guardado em duas matrizes separadas, a **matriz de peças** e **matriz de valores**, ambas de tamanho igual. O tamanho do tabuleiro do jogo é configurável no menu das configurações.

A **matriz de peças** é uma lista de listas que guarda as peças respetivas a cada quadrícula do tabuleiro. Esta matriz está em constante alteração (sempre que ocorre uma jogada ou uma chain).

Cada elemento do tabuleiro é uma lista e representa respetivamente uma linha da matriz.

Uma linha da matriz é composta por várias quadrículas e cada quadrícula é representada por três valores, o jogador, a orientação e o estado da peça, ver listing 3.1.

Listing 3.1: Exemplo de uma peça na matrix de peças.

```
1       [ player1 , horizontal , used ]
```

O **jogador** pode ser um de dois valores, *player1* ou *player2*.

A **orientação** pode ser um de dois valores, *vertical* ou *horizontal*.

O **estado da peça** pode ser um de três valores, *free* ou *used* ou *next*.

A **matriz de valores** é também uma lista de listas, mas apenas guarda a pontuação respetiva a cada quadrícula do tabuleiro. Após gerada, esta matriz permanecerá inalterável.

Cada quadrícula da matriz de valores é representada por um valor aleatório de 0 a 9.

O predicado *build/4* é responsável por gerar ambas as matrizes. Ver listing 7.2.

3.2 Visualização do Tabuleiro

O tabuleiro é visualizado no terminal, em modo de texto, a partir da consola do SICStus, ver exemplo fig.3.1.

A peça do jogador 1 é representada por 'X's e a peça do jogador 2 é representada por 'O's, ver exemplo fig.3.2.

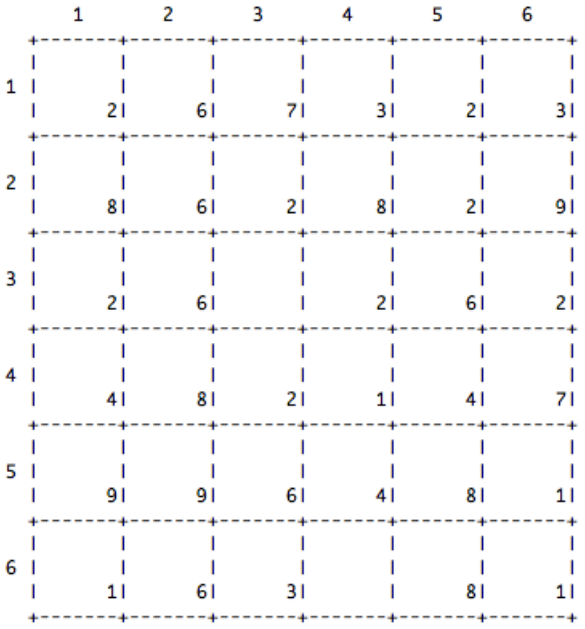

Figura 3.1: Exemplo de um tabuleiro vazio 6x6.

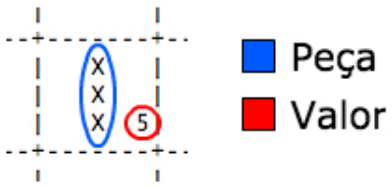

Figura 3.2: Legenda de uma quadrícula.

3.3 Lista de Jogadas Válidas

Não existe movimentação de peças pela parte do jogador. Existe no entanto movimentação de peças como consequência da colocação de uma peça. Por exemplo, o jogador coloca uma peça

e mediante a orientação da peça colocada esta pode ou não empurrar outras peças que estejam adjacentes.

O jogador tem de introduzir as coordenadas em que pretende colocar a peça e especificar a respetiva orientação. A colocação da peça é possível sempre que a quadrícula esteja vazia. Se não for possível, uma nova oportunidade de jogar será dada ao jogador. Após uma jogada com sucesso o tabuleiro resultante é novamente mostrado com as respetivas alterações.

O predicado *get_all/3* encontra todas as posições livres e possíveis de jogar se na variável *Status* estiver o valor *empty*, ver predicado no listing 7.4.

3.4 Execução de Jogadas

O predicado *turn/4* pergunta ao jogador onde quer jogar, interpreta a resposta e se for uma jogada válida então coloca a peça e executa as reações em cadeia, ver listing 7.5.

3.5 Avaliação do Tabuleiro

No fim de cada turno é verificado se existem colocações possíveis utilizando o predicado *is_any_moves_left/1*, ver listing 7.6.

Quanto à avaliação de reações em cadeia possíveis é também utilizado o predicado *get_all/3* mas com a variável *Status* com o valor *next*.

3.6 Final do Jogo

Se não houver mais peças ou se não houver mais jogadas possíveis então precede-se então à contagem dos pontos de cada jogador, ver predicado *get_score/6* no listing 7.7. São somados os pontos de cada quadrícula ocupada por esse jogador à pontuação total e quem tiver mais pontos ganha.

3.7 Jogada do Computador

Se for identificado que o jogador desse turno é um computador então é passado o nível de dificuldade para o predicado *placement_computer/5*, ver listing 7.8.

Capítulo 4

Interface com o Utilizador

O programa oferece uma interface de menus e sub-menus de fácil usabilidade, onde é mostrado ao utilizador as opções disponíveis e inquirindo a opção escolhida seguida da tecla *ENTER*.

O programa do jogo é iniciado com o predicado *init/0*. Ver fig.4.1.

```
| ?- init.
```

Figura 4.1: Predicado init/0 para iniciar o programa.

É apresentado um **texto inicial**, ver fig.4.2., com informação relativa ao trabalho. Basta carregar na tecla *ENTER* para passar ao **menu principal**.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                                          %%%%%%%%%%
%%%%%%%%%%                      Pushee Pieces                       %%%%%%%%%%
%%%%%%%%%%                                                          %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


   Desenvolvido por:
        ei12130 - Eduardo Fernandes
        ei12161 - José Ricardo Coutinho

   No ambito de:
        Programacao em Logica
        do Mestrado Integrado em Engenharia Informatica


                PRESSIONE QUALQUER TECLA PARA INICIAR

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:?:
```

Figura 4.2: Página de apresentação do trabalho.

No **menu principal**, ver fig.4.3, temos as opções:

- 1) Jogador VS Jogador
- 2) Jogador VS Computador
- 3) Computador VS Computador
- 4) Regras
- 5) Configurações
- 0) Sair

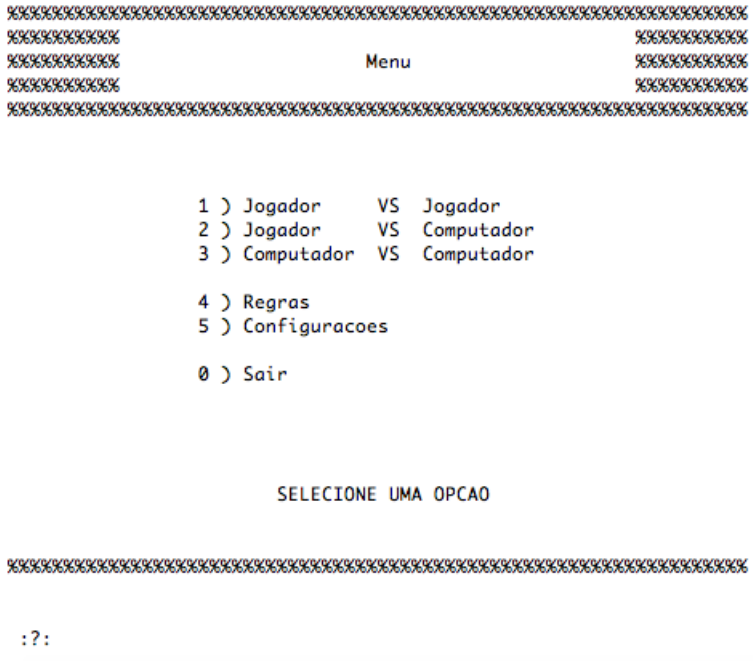

Figura 4.3: Menu principal.

Ao escolher a opção 1 o jogo começa imediatamente, mostrando o estado do tabuleiro e inquirindo ao utilizador onde quer colocar a peça, com a seguinte interface, ver fig.4.4.

Algumas jogadas depois, utilizador se introduzir "h 5 4", está a pedir para colocar a sua peça na vertical na posição X=5 e Y=4. Se o espaço estiver livre então é colocada a sua peça e mostrado o estado do tabuleiro, ver fig.4.5.

Imediatamente é mostrado o tabuleiro após todas as reações em cadeia, ver fig.4.6. Neste caso, existem cadeias possíveis portanto o tabuleiro é alterado.

Isto acontece alternadamente para o jogador 1 e 2 até ao fim do jogo onde é apresentado o resultado final e o vencedor, ver fig.4.7.

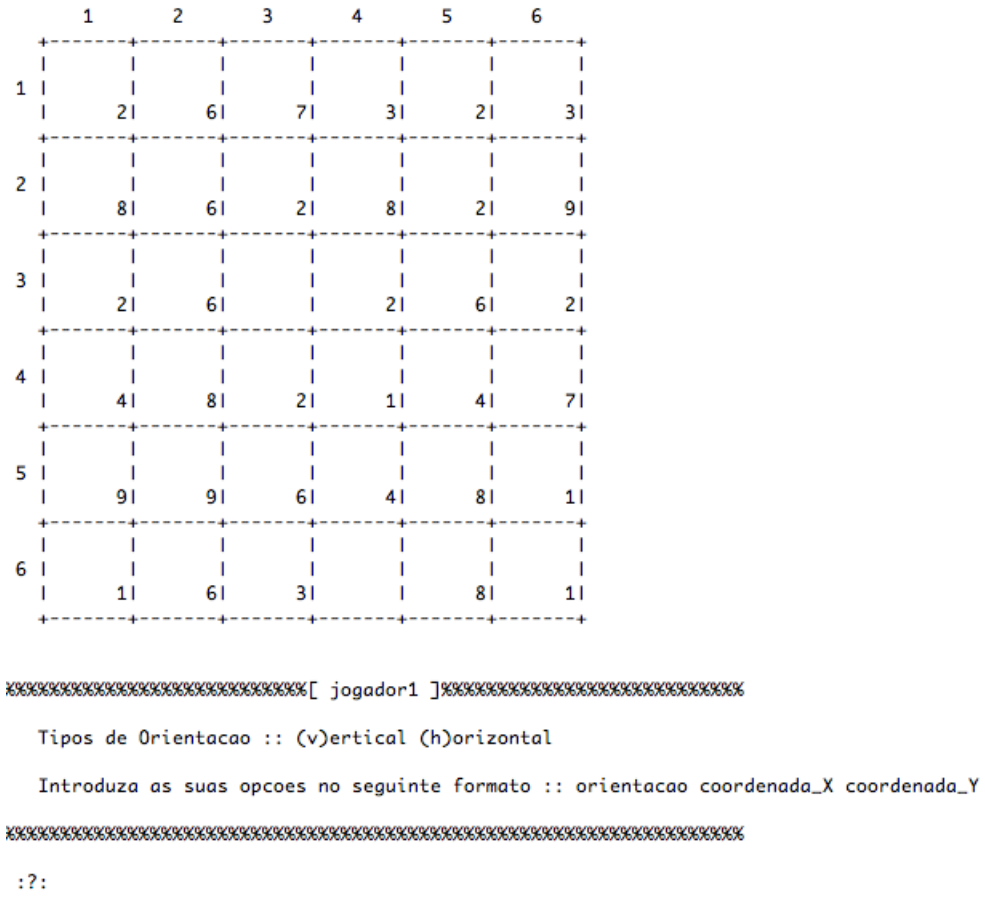

Figura 4.4: Pedido de jogada.

No menu principal, ao escolher a opção 2 ou 3 aparece um sub-menu que dá a escolher o modo de dificuldade para o computador, ver fig.4.8. Para mais detalhe ver o capítulo 5.

No **menu dificuldade** temos as opções:

- 1) Easy
- 2) Normal
- 3) Hard
- 4) Impossibru
- 0) Sair

Ainda no menu principal, ao escolher a opção 5 aparece um sub-menu que dá a escolher as configurações extra do jogo, ver fig.4.9. Para mais detalhe ver o capítulo 5.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Colocacao   %%%%%%%%%%%%%%%%%%%%%%%%%%%%

       1       2       3       4       5       6
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 1 |       |       |       |       |       |       |
   |      2|      6|      7|      3|      2|      3|
   +-------+-------+-------+-------+-------+-------+
   |       |   X   |       |       |       |       |
 2 |       |   X   |       |       |       |       |
   |      8|   X  6|      2|      8|      2|      9|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 3 |       |       | 00000 |       |       |       |
   |      2|      6|       |      2|      6|      2|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |   X   |       |       |
 4 |       |       |       |   X   | 00000 |       |
   |      4|      8|      2|   X  1|      4|      7|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 5 |       |       |       |       |       |       |
   |      9|      9|      6|      4|      8|      1|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 6 |       |       |       |       |       |       |
   |      1|      6|      3|       |      8|      1|
   +-------+-------+-------+-------+-------+-------+
```

Figura 4.5: Peça colocada no tabuleiro.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Cadeia   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

       1       2       3       4       5       6
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 1 |       |       |       |       |       |       |
   |      2|      6|      7|      3|      2|      3|
   +-------+-------+-------+-------+-------+-------+
   |   X   |       |       |       |       |       |
 2 |   X   |       | 00000 |       |       |       |
   |   X  8|      6|      2|      8|      2|      9|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 3 |       |       |       |       |       |       |
   |      2|      6|       |      2|      6|      2|
   +-------+-------+-------+-------+-------+-------+
   |       |       |   X   |       |       |       |
 4 |       |       |   X   |       | 00000 |       |
   |      4|      8|   X  2|      1|      4|      7|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 5 |       |       |       |       |       |       |
   |      9|      9|      6|      4|      8|      1|
   +-------+-------+-------+-------+-------+-------+
   |       |       |       |       |       |       |
 6 |       |       |       |       |       |       |
   |      1|      6|      3|       |      8|      1|
   +-------+-------+-------+-------+-------+-------+
```

Figura 4.6: Reações em cadeia do tabuleiro.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                              %%%%%%%%%%
%%%%%%%%%%                   Resultado                  %%%%%%%%%%
%%%%%%%%%%                                              %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


                  jogador1: 26 pontos
                  jogador2: 30 pontos

               O vencedor:  jogador2



yes
% source_info
| ?- |
```

Figura 4.7: Resultado e fim de jogo.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                              %%%%%%%%%%
%%%%%%%%%%                   Dificuldade                %%%%%%%%%%
%%%%%%%%%%                                              %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


                 1 ) Easy
                 2 ) Normal
                 3 ) Hard
                 4 ) Impossibru

                 0 ) Voltar ao menu principal



                        SELECIONE UMA OPCAO

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 :?: |
```

Figura 4.8: Menu de dificuldade para o computador.

No **menu configuraçoes** temos as opções:

- 1) Tamanho do tabuleiro
- 2) Colisão com parede
- 3) Rotação de peças
- 4) Número de peças
- 5) Número de cadeias
- 0) Voltar ao menu principal

Estas opções visam à sua frente o estado atual respetivo a cada propriedade.

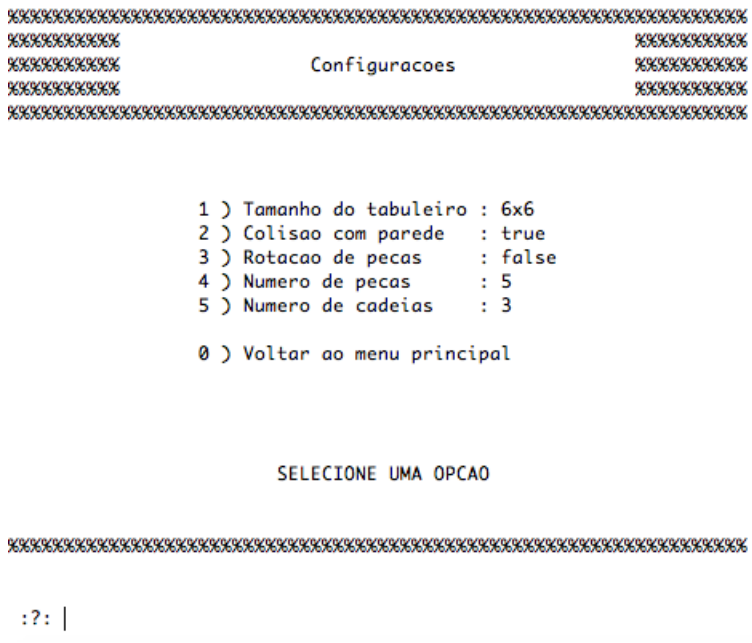

Figura 4.9: Menu configuração.

Capítulo 5

Extras Implementados

5.1 Níveis de Dificuldade

Como referido no capítulo 4 existem quatro modos de dificuldade para o computador. No entanto, cada modo não implementa heurísticas inteligentes para indicar a jogada, apenas implementa métodos diferentes e simples, de escolha de jogada e portanto o nome da dificuldade não deve ser interpretado à letra. Isto é, exceto o modo *Impossibru*.

No modo *Easy* o computador joga na primeira quadrícula disponível com uma orientação aleatória.

No modo *Normal* o computador joga em uma quadrícula disponível aleatória com uma orientação aleatória.

O modo *Hard* é igual ao modo *Normal*.

No modo *Impossibru* o computador faz batota e coloca peças em cima das do outro jogador(que não é permitido), assim sendo é impossível ganhar.

5.2 Tamanho do Tabuleiro

O tabuleiro pode ser gerado para qualquer tamanho, no entanto optamos por dar apenas quatro alternativas de forma a tornar o jogo mais fácil de jogar. Os tamanhos possíveis são: 6x6, 7x7, 8x8, 9x9. Ver figura 5.1.

5.3 Colisão com Parede

Esta configuração permite ligar ou desligar a funcionalidade das paredes do tabuleiro. Isto é, se as paredes estiverem desligadas então é possível empurrar peças para fora do tabuleiro.

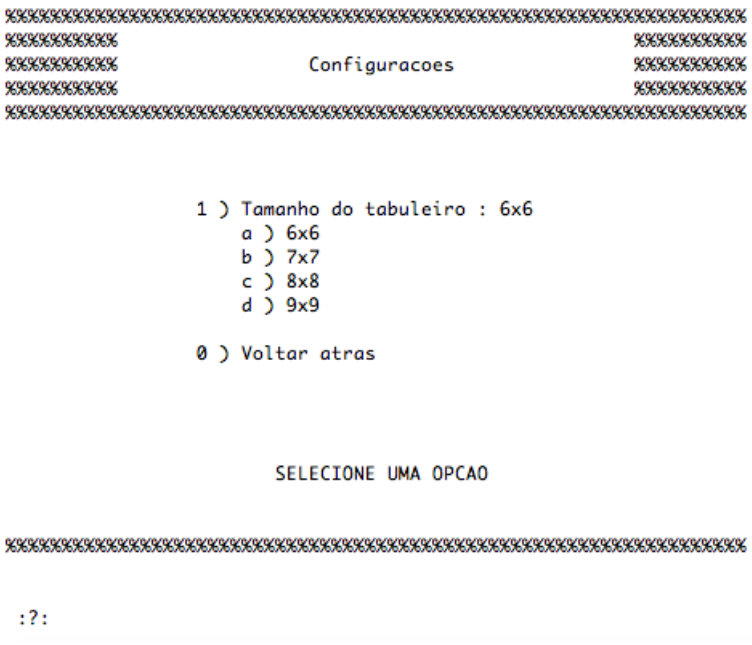

Figura 5.1: Menu configuração do tamanho do tabuleiro.

5.4 Rotação de Peças

Esta configuração permite ligar ou desligar a rotação de peças, e embora exista a configuração no menu, não foi implementada a sua funcionalidade por falta de tempo.

5.5 Número de Peças

Esta configuração permite escolher o número de peças pretendidas por jogador, afetando diretamente o número de turnos do jogo, podendo assim reduzir-se ou aumentar a duração do jogo. Ver fig.5.2.

5.6 Número de Reações em Cadeia

Esta configuração permite escolher o número de reações em cadeia (*chains*). Ver fig.5.3.

5.7 Melhor Usabilidade

Foram criados menus para facilitar a configuração e aspeto, mensagens de erro, uma secção de explicação das regras e navegação entre menus.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                               %%%%%%%%%%
%%%%%%%%%%                 Configuracoes                 %%%%%%%%%%
%%%%%%%%%%                                               %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


                 4 ) Numero de pecas : 5
                     a )  5
                     b ) 15
                     c ) 20
                     d ) 35

                 0 ) Voltar atras



                        SELECIONE UMA OPCAO

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


 :?: |
```

Figura 5.2: Menu configuração do número de peças.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                               %%%%%%%%%%
%%%%%%%%%%                 Configuracoes                 %%%%%%%%%%
%%%%%%%%%%                                               %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


                 5 ) Numero de cadeias : 3
                     a ) 0
                     b ) 3
                     c ) 6
                     d ) ilimitado

                 0 ) Voltar atras



                        SELECIONE UMA OPCAO

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


 :?:
```

Figura 5.3: Menu configuração do número de cadeias.

Capítulo 6

Conclusões

O uso de variáveis dinâmicas, *assert*'s e *retract*'s simplificam e separam a conduta de menus, e respetivas configurações do programa, de complicarem a funcionalidade e legibilidade dos predicados de jogo.

Este projeto poderia ser melhorado se auxiliado de uma melhor heurística para a dificuldade de computador e de uma interface gráfica para melhor visualização e usabilidade.

Capítulo 7

Anexos

Listing 7.1: Predicado responsável pelo jogo.

```
% play(+NTurns,+MatrixPieces,+MatrixValues)
play(N,MP,MV) :-
    N > 0, N1 is N-1,

    turn(player1,MV,MP,MP2), is_any_moves_left(MP2),
    turn(player2,MV,MP2,MP3), is_any_moves_left(MP3),

    play(N1,MP3,MV).

play(_,MP,MV) :- show_result(MP,MV).
```

Listing 7.2: Predicado que constroi as matrizes do tabuleiro.

```
% build_board(+Width,+Height,-MatrixPieces,-MatrixValues)
build(W,H,MatrixPieces,MatrixValues) :-
    build_board(W,H,build_element,MatrixPieces),
    build_board(W,H,build_value,MatrixValues).
```

Listing 7.3: Predicado imprime o tabuleiro.

```
% print_board(+MatrixPieces,+MatrixValues)
print_board(MP,MV) :- nl,
    head(MP,L), length(L,Width),
    write('    '), print_guide(1,Width),nl,
                  print_rows(1,Width,MP,MV),
    write('    '), print_border(Width), nl.
```

Listing 7.4: Predicado que obtém todas as jogadas possíveis.

```
% get_all(+Status,+Matrix,-List)
get_all(Status,M,L) :-
    setof( (X,Y), ( nth1(Y,M,Row), nth1(X,Row,P), member(Status,
    P) ), L).

get_all(_,_,[]).
```

Listing 7.5: Predicado que executa a jogada de um jogador.

```
% turn(+Player,+MatrixValues,+Matrix,-Matrix2)
turn(Player,MV,M,M4) :- % turn with computer playing
    is_computer(Player),
    msg_name(computer),
    flag_mode(Mode),
    placement_computer(Mode,Player,MV,M,M2), msg_place,
    print_board(M2,MV),

    flag_chain(N),
    chainreaction(N,M2,M3), msg_chain, print_board(M3,MV),
    clear(M3,M4).

turn(Player,MV,M,M4) :-
    placement(Player,M,M2), msg_place, print_board(M2,MV),

    flag_chain(N),
    chainreaction(N,M2,M3), msg_chain, print_board(M3,MV),
    clear(M3,M4).
```

Listing 7.6: Predicado que verifica se existem mais colocações disponiveis.

```
% is_any_moves_left(+Matrix)
is_any_moves_left(M) :-
    get_all(empty,M,L),
    \+ is_empty_list(L).
```

Listing 7.7: Predicado que calcula os pontos de cada jogador .

```
% get_score(+MatrixPieces, +MatrixValues,+P1Counter,+P2Counter,-
    P1Score,-P2Score)
get_score([LP|LPs],[LV|LVs],P1Incr,P2Incr,P1Score,P2Score) :-
    get_score_list(LP,LV,P1Incr,P2Incr,P1ListScore,P2ListScore),
    get_score(LPs,LVs,P1ListScore,P2ListScore,P1Score,P2Score).

get_score([],[],P1Score,P2Score,P1Score,P2Score).
```

Listing 7.8: Predicado que coloca peça do computador.

```
% placement_computer(+Mode,+Player,+MatrixValues,+Matrix,-
    MatrixOut)
placement_computer(easy,Player,_,M,M2) :-
    get_all(free,M,[(X,Y)|_]),
    random(1,3,Oi), orientation_from_number(Oi,O),
    P = [Player,O,next],
    set(P,X,Y,M,M2).

placement_computer(normal,Player,_,M,M2) :-
    get_all(free,M,L), length(L,Max), Max1 is Max+1,
    random(1,Max1,I), nth1(I,L,(X,Y)),
    random(1,3,Oi), orientation_from_number(Oi,O),
    P = [Player,O,next],
    set(P,X,Y,M,M2).

placement_computer(hard,Player,_,M,M2) :-
    get_all(free,M,L), length(L,Max), Max1 is Max+1,
    random(1,Max1,I), nth1(I,L,(X,Y)),
    random(1,3,Oi), orientation_from_number(Oi,O),
```

```
19      P = [Player,O,next],
20      set(P,X,Y,M,M2).
21
22  placement_computer(impossibru,Player,_,M,M2) :-
23      get_all(player1,M,L), length(L,Max), Max1 is Max+1,
24      random(1,Max1,I),       nth1(I,L,(X,Y)),
25      random(1,3,Oi),         orientation_from_number(Oi,O),
26      P = [Player,O,next],
27      set(P,X,Y,M,M2).
28
29  placement_computer(impossibru,Player,_,M,M2) :-
30      get_all(free,M,L), length(L,Max), Max1 is Max+1,
31      random(1,Max1,I),   nth1(I,L,(X,Y)),
32      random(1,3,Oi),     orientation_from_number(Oi,O),
33      P = [Player,O,next],
34      set(P,X,Y,M,M2).
```

Bibliografia

Eric Daryl Stevenson. Board game geek: Game information, a. URL http://boardgamegeek.com/boardgame/158636/pushee-pieces.

Eric Daryl Stevenson. Kickstarter: Project funding, b. URL https://www.kickstarter.com/projects/pocketvinyl/pushee-pieces-a-compact-9-microgame.

Eric Daryl Stevenson. Youtube: Gameplay video, c. URL https://www.youtube.com/watch?v=dwKSoeLow-0.

Glossário

FEUP Faculdade de Engenharia da Universidade do Porto. 3

MIEIC Mestrado em Integrado em Engenharia Informática. 3

PLOG disciplina de Programação em Lógica. 3

Prolog uma linguagem de programação em lógica. 3

SICStus um interpretador e compilador de Prolog. 3, 8