Programação em Lógica – 2019/2020

Squex_3

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

Índice

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

1.Introdução

Este projeto foi desenvolvido no âmbito da unidade curricular de Programação Lógica de 3º ano do curso Mestrado Integrado em Engenharia Informática e Computação, usando o Sistema de Desenvolvimento *SICStus Prolog*, tendo como tema o jogo de tabuleiro *Squex*.

O objetivo deste trabalho foi construir uma versão virtual do jogo em questão usando a linguagem *Prolog*.

2.0 Jogo *Squex*

Squex é um jogo de conexão abstrata para dois (2) jogadores.

Squex é jogado num tabuleiro 8x8 constituído por octógonos e quadrados, presentes entre os octógonos.

O objetivo de cada jogador é conectar os seus lados correspondentes do tabuleiro com uma linha contínua de peças.

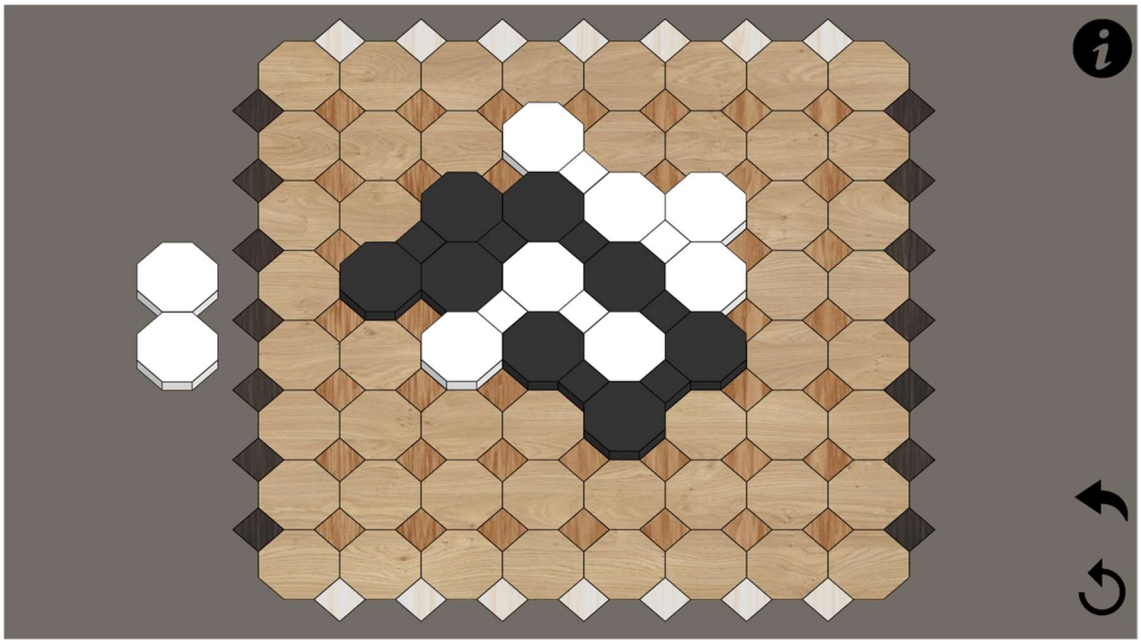


Fig.1 Tabuleiro de Squex nas jogadas iniciais

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

Regras:

1. Na sua vez, cada jogador pode colocar uma peça octogonal em qualquer espaço octogonal livre do tabuleiro.

2. Se uma peça é colocada na diagonal de uma peça da mesma cor, um quadrado dessa cor é colocado para conectá-las (é possível colocar até 4 quadrados colocando apenas uma peça).

3. Se uma peça for colocada na diagonal de uma peça da mesma cor e houver um quadrado da cor do oponente entre eles, o quadrado do oponente será substituído por um quadrado da cor do jogador. Os octógonos do oponente já não estão conectados pelo quadrado que lá estava. A isso chama-se “corte”.

4. Sempre que um jogador faz um "corte", o oponente pode jogar duas vezes seguidas (se um jogador fizer um "corte" na primeira jogada este perderá a sua segunda jogada e será a vez do oponente de jogar duas vezes consecutivas).

5. Um jogador só vence quando este conectar os seus dois lados do tabuleiro. (e for impossível para o seu oponente quebrar a ligação com um "corte" na jogada imediatamente seguinte. – TO DO!)

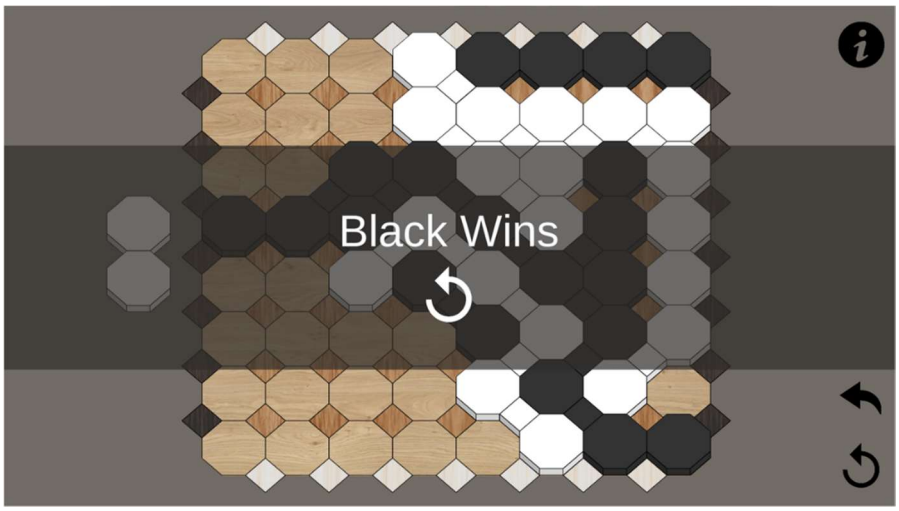


Fig.2 Tabuleiro final possível

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

3.Lógica do Jogo

3.1.Representação do Estado do Jogo

Situação Inicial

```
initialBoard([
    [corner,wcut,wcut,wcut,wcut,wcut,wcut,wcut,corner],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,blank],
    [corner,wcut,wcut,wcut,wcut,wcut,wcut,wcut,corner]
    ]).
```

Fig.3 Estado inicial visto em formato de código

Situação Intermédia

```
midBoard([
    [corner,wcut,wcut,wcut,wcut,wcut,wcut,wcut,corner],
    [black,black,black,black,black,black,black,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,white,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,white,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,white,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,white,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,blank,white,blank,blank,blank],
    [bcut,uncut,uncut,uncut,uncut,wcut,uncut,uncut,bcut],
    [blank,blank,blank,blank,blank,white,white,blank],
    [bcut,uncut,uncut,uncut,uncut,uncut,uncut,wcut,bcut],
    [blank,blank,blank,blank,blank,blank,blank,white],
    [corner,wcut,wcut,wcut,wcut,wcut,wcut,wcut,corner]
    ]).
```

Fig.4 Possível estado intermédio visto em formato de código

Situação Final

```
finalBoard([
    [blank,wcut,wcut,wcut,wcut,wcut,wcut,wcut,blank],
    [blank,blank,blank,white,black,black,black,black],
    [bcut,uncut,uncut,uncut,wcut,uncut,uncut,uncut,bcut],
    [blank,blank,blank,white,white,white,white,white],
    [bcut,uncut,uncut,uncut,wcut,wcut,wcut,wcut,bcut],
    [blank,blank,black,black,white,white,black,white],
    [bcut,uncut,bcut,bcut,bcut,wcut,uncut,uncut,bcut],
    [black,black,black,white,black,white,black,white],
    [bcut,uncut,uncut,wcut,wcut,bcut,bcut,uncut,bcut],
    [blank,blank,white,black,white,black,black,white],
    [bcut,uncut,uncut,uncut,bcut,wcut,bcut,uncut,bcut],
    [blank,blank,blank,blank,black,white,black,white],
    [bcut,uncut,uncut,uncut,uncut,wcut,bcut,wcut,bcut],
    [blank,blank,blank,blank,white,black,white,blank],
    [bcut,uncut,uncut,uncut,uncut,wcut,bcut,uncut,bcut],
    [blank,blank,blank,blank,blank,white,black,black],
    [blank,wcut,wcut,wcut,wcut,wcut,wcut,wcut,blank]
    ]).
```

Fig.5 Possível estado final visto em formato de código

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

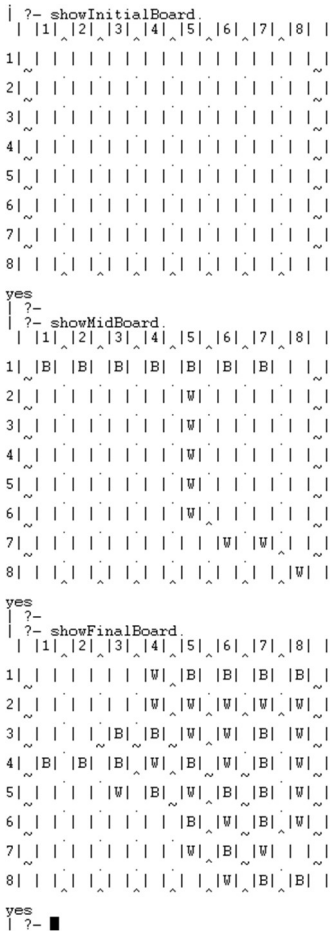


Fig.6 Estado inicial e possível estado intermédio e final vistos pela consola do programa *SICStus 4.5.1*

3.2.Visualização do Tabuleiro

De forma a apresentar o tabuleiro são usadas 3 funções: **displayCell** que escreve uma única célula no ecrã; **displayLine**, que percorre a lista passada como argumento e, para cada elemento da lista, chama a função displayCell; **displayBoard** que mostra o tabuleiro dado, percorrendo todas as listas encontradas dentro da lista inicial (lista de listas) e chamando a função displayLine.

3.2.1 Interface do Jogo

Para melhor visualização, criamos um menu, como se pode ver na figura abaixo apresentada, que indica o nome do jogo e as diversas opções de jogo, por ordem:

Pessoa->Pessoa;

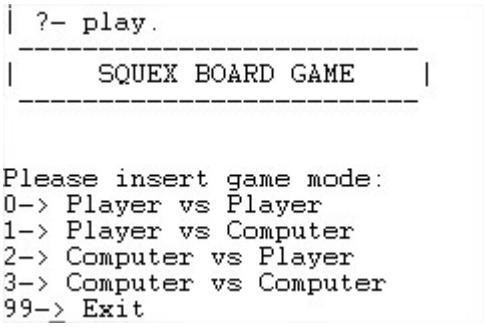
Pessoa-> Computador;

Computador->Pessoa;

Computador->Computador;

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

Sendo que os modos 2 e 3 apenas diferem na cor que cada jogador toma.



3.3.Lista de Jogadas Válidas

No nosso jogo, é possível colocar uma peça em qualquer posição do tabuleiro desde que este esteja vazio.

De modo a verificar todas as jogadas ainda possíveis usamos a função **valid_moves**, que percorre o tabuleiro todo e retorna uma lista com todas as jogadas válidas; retorna 'no' se o tamanho da lista de jogadas for 0.

3.4.Execução de Jogadas

Num jogo do tipo pessoa contra pessoa, o jogador apenas necessita de identificar o número da coluna e da linha em que pretende colocar a sua peça para que o jogo corra normalmente. Dentro da função **playPiece** são feitas as verificações necessárias para decidir quem será o próximo jogador, de acordo com as regras do jogo.

Inicialmente, antes, de o jogador ter a oportunidade de jogar, verifica-se se existem jogadas válidas com o auxílio da função **valid_moves** – se esta falhar o jogo termina.

Em segundo lugar, a função verifica se o local onde a peça vai ser colocada encontra-se livre ou não - **isCellEmpty**, em caso positivo a operação continua normalmente, em caso contrário é dada ao jogador uma outra oportunidade de jogar.

De seguida, o programa insere uma peça no tabuleiro usando **replaceBoardCell** de acordo com os parâmetros inseridos pelo jogador e da peça do mesmo, devolvendo o tabuleiro modificado. Após a peça ter sido colocada verificasse se algum quadrado deve ser colocado e se houve a ocorrência de um corte (**checkSquare**), tendo como valores de retorno um novo tabuleiro, se este foi modificado, e o valor de 'Cut' -

João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

identifica se houve um ‘cut’ ou não. Depois destas duas funções, faz-se a verificação de vitória do jogador.

De acordo com as regras do jogo, se o jogador foi cortado anteriormente pelo seu oponente e não o cortou na sua primeira jogada, este terá a oportunidade de jogar outra vez; se foi cortado e cortou o oponente, perderá a sua segunda jogada; caso não tenha sido cortado a jogada passará ao outro jogador.

É de notar que o número da linha inserida nas funções é o dobro do número da linha inserida pelo jogador.

isCellEmpty(+Board, +Row, +Column) - verifica se um elemento está vazio (“blank”) ou não.

replaceBoardCell(+Board, +Row, +Column, +Piece, -ModifiedBoard) - insere uma peça no tabuleiro e retorna o tabuleiro modificado.

checkSquare(+Board, +Row, +Column, +Piece, -Result, -Cut) - insere um quadrado no tabuleiro e retorna o tabuleiro modificado; Cut retorna como 1 caso haja um corte e retorna como 0 caso contrário.

3.5.Final do Jogo

Para que o jogo chegue a um fim é necessário que um dos jogadores (humano ou computador) consiga alcançar um estado de vitória ou, no caso de não haver mais nenhuma jogada possível e ninguém ter ganho, terem ambos alcançado um estado de empate.

Neste jogo, um estado de vitória é quando um jogador consegue ter um caminho que una os dois lados do tabuleiro. As únicas restrições para este caminho são:

Para as peças brancas, o sentido do caminho é vertical;

Para as peças pretas, o sentido do caminho é horizontal;

Para verificar se um jogador conseguiu ganhar é chamada uma das seguintes funções: checkWhiteVictory, no caso das peças brancas; checkBlackVictory, no caso das peças pretas.

Estas funções verificam que efetivamente o jogador possui um caminho no sentido pretendido, verificando os quadrados diagonais em todas as peças.

São parecidas, diferindo no sentido de propagação da função.

3.6.Avaliação do Tabuleiro

Para avaliação do tabuleiro, possuímos uma implementação informal.

Em cada jogada, calculamos, com as funções checkVictory (checkBlackVictory e checkWhiteVictory), se o jogador ganhou, sendo que, indiretamente, calculamos o valor do tabuleiro desse jogador nesse determinado momento, já que quanto maior o seu caminho, menos falta para ganhar e, consequentemente, mais valioso o seu tabuleiro.

3.7.Jogada do Computador

Dos 4 modos disponíveis, 3 deles são com o computador a "jogar". A implementação das jogadas do computador divide-se. Primeiramente é perguntada qual a "dificuldade" do computador, qual o seu nível e, dependendo da resposta, o computador irá comportar-se de maneiras distintas. No nível 1, apenas são gerados números pseudoaleatórios para as linhas e colunas, de maneira a colocar a peça do jogador no tabuleiro. No nível 2, existe uma verificação de qual a linha com mais peças da cor do jogador, ou seja, a linha mais perto da vitória e, caso haja espaço vazio nessa linha, a peça é colocada.

Criamos vários predicados para executar as jogadas dependendo do modo de execução e dificuldade do mesmo.

Para o modo 0, player vs player, e modos 1 e 2, player vs computer e computer vs player, em que existe jogada humana, criamos predicados que lêem o input do jogador para as linhas e colunas, respetivamente.

Nos restantes casos, modos 1, 2 e 3, em que o computador executa jogadas, foram criados predicados de decisão de jogada, "playBlackBot" e "playWhiteBot", que, em função do nível, calculam a melhor jogada ou apenas fazem um número pseudoaleatório.

```
playBlackBot(Board, TempR1, C1, Computer, Dificulty),
playWhiteBot(Board, TempR2, C2, Computer, Dificulty),
```


João Miguel dos Santos de Lima Monteiro – up201705580
Jorge David Monteiro Pacheco – up201705754

4.Conclusão

Na realização deste trabalho foi requerido muito esforço de ambos os elementos do grupo, no entanto promoveu bastante o nosso conhecimento da linguagem PROLOG, o nosso pensamento lógico e conhecimento na área da inteligência artificial.

Ao longo do desenvolvimento deste projeto, encontramos algumas dificuldades, nomeadamente nas condições de vitória e na inserção de um elemento no tabuleiro, porém todas estas foram eventualmente superadas.

Para sumarizar, o trabalho foi concluído com sucesso e mostrou-se um trabalho cativante principalmente na área do raciocínio lógico.

Bibliografia

As figuras 1 e 2 foram retiradas do seguinte URL:

https://boardgamegeek.com/image/4726662/squex