Relatório do projeto de Programação em Lógica

Trabalho 1

Quantum Leap_2

Universidade do Porto

Faculdade de Engenharia

FEUP

Turma 4 Grupo:

Joel Márcio Torres Carneiro 201100775

Miguel Geraldes Antunes Mendes 201105535

Resumo:

O problema abordado pelo grupo consiste na realização da lógica do jogo de tabuleiro Quantum Leap_2. Foi feito um relatório intercalar onde foram especificadas algumas das funções finais do jogo, como por exemplo: as regras, representações de estados de jogo, visualização do tabuleiro em modo de texto e movimentos.

O objetivo do trabalho é fazer o código em linguagem de programação lógica capaz da realização da lógica do jogo em questão. Assim, foram criadas funções e listas de forma a resolver o problema em mãos.

O jogo de tabuleiro pode ser jogado em modo jogador vs computador (com diferentes níveis de dificuldade) ou jogador vs jogador.

A aplicação desenvolvida é capaz de interagir com o utilizador de forma a que este navegue nas opções de jogo eficazmente. O utilizador é chamado a interagir para poder efetuar jogadas que são validadas pela aplicação.

Através deste projeto é possível aferir com eficácia as capacidades extensas de Prolog em operações complexas, principalmente em termos de utilização de listas, ou de forma mais profunda, listas de listas.

Introdução

O relatório foi desenvolvido no âmbito da unidade curricular de Programação Lógica, de forma a realizar os processos lógicos de um jogo.

O objetivo do trabalho é criar o código em linguagem de programação lógica que realize a lógica do jogo de tabuleiro Quantum Leap_2. É um jogo com características peculiares que lhe trazem uma jogabilidade diferente do comum.

Por ser um jogo tão interessante, intrigante e ao mesmo tempo fora do comum, despertou interesse ao grupo. É um jogo que põe em questão a capacidade de raciocínio e de adivinhar o que o adversário irá jogar. Desta forma escolheu-se o jogo Quantum Leap_2 como tema de trabalho.

Este relatório descreve as várias componentes do código que foi elaborado. Começou por se descrever o jogo e as suas regras. De seguida, mostrou-se a lógica utilizada no jogo, a representação do estado de jogo e a visualização, em modo de texto, do tabuleiro de jogo. O relatório continuou com a demonstração das jogadas válidas, com a execução de jogadas e com uma avaliação do tabuleiro. Na parte final do relatório é falado sobre o final do jogo, a jogada do computador e a interface do utilizador.

Breve descrição do jogo:

Quantum Leap é um jogo criado em 2013 pela Nestor Games.

O jogo toma lugar numa grelha hexagonal com cinco casas de cada lado. Um total de sessenta peças entram em jogo, sendo trinta peças pretas e trinta brancas. Inicialmente, todas as peças são colocadas de forma aleatória no tabuleiro, restando um único espaço livre. Jogando o jogador de peças brancas o primeiro turno, o outro jogador tem a oportunidade de trocar a posição de quaisquer duas peças que deseje.

Cada jogada é bastante peculiar, sendo que cada peça tem um potencial de salto correspondente ao número de peças aliadas que a rodeiam. Em cada jogada, esse potencial de salto é utilizado para eliminar uma peça adversária, tomando a posição desta última. Logo, numa jogada normal, hipotetizando que uma peça prestes a ser jogada se encontra rodeada de duas peças aliadas, pode mover-se através de 3 celas, sendo as duas primeiras as que efetivamente "saltou" com o seu potencial.

Este tipo de jogadas repete-se, sendo o vencedor o último jogador a conseguir efetuar uma jogada válida. Consequentemente haverá uma altura em que um jogador não conseguirá utilizar uma das suas peças em conjunto com o seu potencial para eliminar uma peça adversária, apercebendo-se que perdeu.

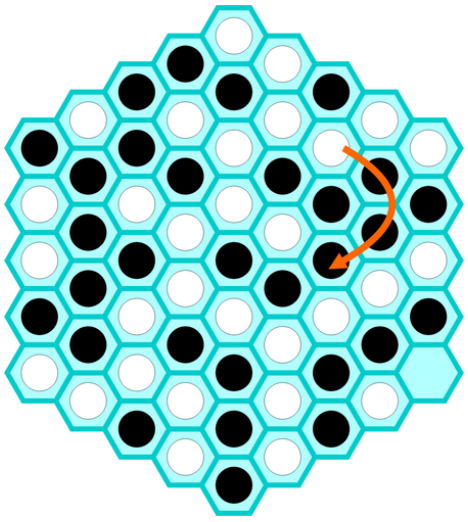


Figura de possível estado inicial e do tabuleiro, com possível jogada de uma peça branca.

Lógica de Jogo:

1.Representação do estado do jogo:

Tratando-se de uma grelha hexagonal, sugere-se um método alternativo para a representação do tabuleiro. Utilizando uma matriz normal com espaços tornados inválidos, podemos simular as posições e movimentos de peças de forma hexagonal, sem grandes esforços. Tome-se o exemplo da imagem fornecida anteriormente, tendo em conta que W = peças brancas, B = peças pretas e # denota um espaço inválido:

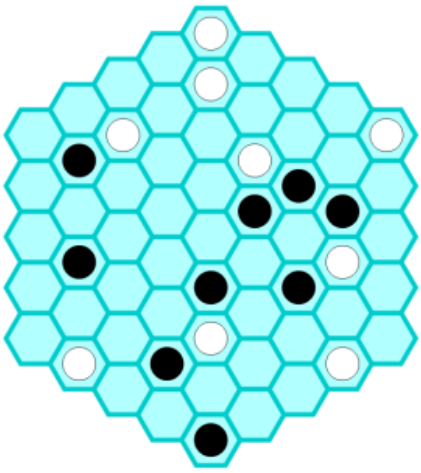
```
ExampleBoard(  [[W,W,B,W,W,#,#,#,#],
                [B,B,W,W,B,B,#,#,#],
                [B,W,W,B,B,B,W,#,#],
                [W,B,B,W,W,B,W,B,#],
                [B,B,W,W,B,B,W,B,_],
                [#,W,B,B,W,W,W,B,W],
                [#,#,W,B,W,B,B,W,B],
                [#,#,#,B,B,W,W,B,W],
                [#,#,#,#,W,W,B,W,B]]).
```

Vejamos agora uma situação em que estejamos a meio do jogo:

```
MiddleBoard(  [[W,_,_,_,W,#,#,#,#],
               [_,W,_,_,_,W,#,#,#],
               [_,_,_,W,B,B,_,#,#],
               [_,W,_,_,B,_,W,_,#],
               [_,B,_,_,_,_,B,W,_],
               [#,_,_,_,_,B,_,_,W],
               [#,#,_,B,_,_,W,_,_],
               [#,#,#,_,_,_,B,_,_],
               [#,#,#,#,_,W,_,_,B]]).
```

Uma situação possível de fim de jogo:

```
EndBoard([[W,_,_,_,W,#,#,#,#],
          [_,W,_,_,_,_,#,#,#],
          [_,_,_,W,B,B,_,#,#],
          [_,W,_,_,B,_,W,_,#],
          [_,B,_,_,_,_,B,_,_],
          [#,_,_,_,_,B,_,_,W],
          [#,#,_,B,_,_,W,_,_],
          [#,#,#,_,_,_,B,_,_],
          [#,#,#,#,_,W,_,_,B]]).
```

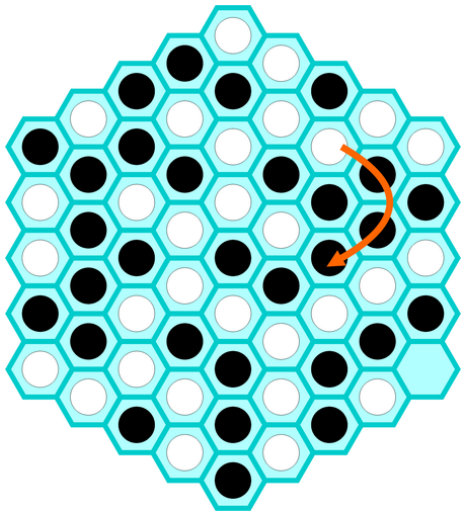


2.Visualização do tabuleiro em modo de texto:

Tomemos de novo este exemplo:

A representação sugerida para este exemplo de estado de jogo seria a seguinte:

```
    |W|W|B|W|W|
   |B|B|W|W|B|B|
  |B|W|W|B|B|B|W|
 |W|B|B|W|W|B|W|B|
|B|B|W|W|B|B|W|B|_|
 |W|B|B|W|W|W|B|W|
  |W|B|W|B|B|W|B|
   |B|B|W|W|B|W|
    |W|W|B|W|B|
```



Os predicados necessários para representar o tabuleiro ao utilizador aquando do correr do jogo seriam:

printBoard(X). // em que X será a estrutura atual do tabuleiro

Esta função printBoard executa o output do tabuleiro de forma "user-friendly".

printLine(X) // line do tabuleiro que está a ser atualmente analisada

Esta função printLine executa o output da linha especificada de forma "user-friendly".

3.Lista de jogadas válidas

Devido às regras inerentes ao jogo, o mais apropriado torna-se apenas pedir uma peça e o seu destino de movimento, para avaliar se tal movimento é válido.

Deste modo, os predicados necessários para realizar estes movimentos seriam:

```
getValidPlays(Board, X, Y, PlayList).
```

Onde "Board" denotaria o tabuleiro onde efetuar a operação, "X" e "Y" seriam a posição da peça a mover, PlayList a lista de jogadas possíveis.

4.Execução de jogadas

É necessário descobrir onde é possível colocar determinada peça, durante uma jogada. Para que a jogada seja feita, é gerado um novo tabuleiro no qual o local da peça escolhida a mover fica vazio e o destino, em vez da peça do adversário, tem a peça movida. Cada jogador só poderá mover as peças da sua cor.

Deste modo, os predicados necessários para realizar estes movimentos seriam:

```
movePieceTo(Board1, X1, Y1, X2, Y2, Board2).
```

Onde "Board1" denotaria o tabuleiro onde efetuar a operação, "X1" e "Y1" seriam a posição da peça a mover, "X2" e "Y2" seriam a posição de destino e "Board2" o tabuleiro actualizado.

5.Avaliação do Tabuleiro

Ao fim de cada jogada será feita uma avaliação sobre o estado de jogo do tabuleiro, de forma a serem avaliadas as possíveis jogadas a realizar. Se já não restar nenhuma jogada possível para o jogador, o jogo termina sendo a vitória do último jogador a fazer uma jogada válida.

A avaliação do estado do jogo permitirá comparar a aplicação das diversas jogadas disponíveis. Deste modo, os predicados necessários para realizar estes movimentos seriam:

```
getValidPlays(Board, X, Y, PlayList).
```

Onde "Board" denotaria o tabuleiro onde efetuar a operação, "X" e "Y" seriam a posição da peça a mover, PlayList a lista de jogadas possíveis.

6.Final do Jogo

O jogo termina quando já não é possível realizar jogadas válidas. Desta forma, o último jogador a efetuar um jogada válida é o vencedor.

Realiza-se uma avaliação ao tabuleiro de forma a avaliar se é possível mais alguma jogada por parte do jogador. Se não for possível o jogo termina com a vitória do adversário.

Desta forma, os predicados necessários para saber se o jogo terminou seriam:

```
getValidPlays(Board, X, Y, PlayList).
```

Onde "Board" denotaria o tabuleiro onde efetuar a operação, "X" e "Y" seriam a posição da peça a mover, PlayList a lista de jogadas possíveis.

Quando a função getValidPlays aplicada a todas as peças devolver sempre uma lista vazia o jogador do turno anterior vence pois não é possível realizar mais nenhuma jogada.

7.Jogada do Computador

Para que o computador possa efetuar uma jogada, essa jogada tem que ser válida. Desta forma, é necessária uma avaliação do tabuleiro para que se saiba a lista de jogadas válidas. Após essa avaliação, o computador irá realizar uma das jogadas consoante o nível de dificuldade escolhido pelo utilizador. Em modo fácil, da lista de jogadas válidas, o computador escolheria uma jogada aleatoriamente.

Desta forma, os predicados necessários para o computador efetuar uma jogada válida seriam:

```
getValidPlays(Board, X, Y, PlayList).
```

Após a obtenção da lista de jogadas válidas é chamada uma função que escolhe e executa uma jogada aleatoriamente, de entre as jogadas válidas. O predicado necessário para a escolha e execução aleatória será:

```
computerEasyPlay(Board1, PlayList, Board2).
```

Interface com o Utilizador

A aplicação inicia com a execução da função start(X). Após a sua execução na interface da aplicação é possível visualizar o estado atual do tabuleiro.

É também pedido ao utilizador que escolha a peça que quer movimentar e o local de destino. Para isso são pedidas as coordenadas da peça a mover e as coordenadas do destino. Feito o movimento de cada peça o tabuleiro é atualizado. Toda a mecânica de jogo é feita com "input's" por parte do utilizador e "output's" no ecrã do computador por parte da aplicação.

Conclusões

A realização deste trabalho permitiu aprofundar conhecimentos em relação à programação lógica. Assim, foi possível ficar mais familiarizado com a linguagem em questão, o que trará benefícios num futuro próximo.

Podemos concluir que foram alcançados todos os objetivos a que o grupo se propôs no início do seu desenvolvimento, realçando o facto de o jogo apresentar uma plataforma visual simples e eficaz.

Bibliografia

http://boardgamegeek.com/boardgame/140782/quantum-leap

Anexos