Ploy

Relatório Final



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Ploy\_1:

Miguel Lira Barbeitos Luís – up201405324

Miriam Cristiana Meireles Campos Gonçalves – up201403441

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

13 de Novembro de 2016

**Resumo**

No âmbito da unidade curricular de Programação em Lógica foi-nos proposto o desenvolvimento de um jogo de tabuleiro na linguagem PROLOG, da lista de projetos possíveis a desenvolver, escolheu-se o PLOY.

Uma vez que a linguagem PROLOG é bastante distinta das linguagens mais indutivas e complexas lecionadas nas outras unidades curriculares. Sendo esta puramente lógica, tornou-se um obstáculo a ultrapassar durante o desenvolvimento do projeto.

O desenvolvimento deste projeto foi possível devido ao empenho do grupo e trabalho em equipa. Foi necessário estudar o jogo para compreender as principais regras e principais objetivos, para que permitisse uma implementação simplificada e objetiva, também se procedeu a uma organização estrutural do tabuleiro, após aconselhado pelo professor das aulas práticas, permitindo-nos maior facilidade durante a implementação de funcionalidades. O maior desafio durante todo este processo de aprendizagem foi a adaptação a esta linguagem tão distinta e lógica, a representação mais humana possível deste jogo, tanto a nível de visualização como de execução, devido ao tabuleiro detalhado usado neste jogo e também às regras complexas e bastante pormenorizadas definidas.

Em conclusão, o grupo acredita que conseguiu chegar a um resultado com uma implementação simples e robusta. Também se conclui que durante toda a realização de código foi possível uma compreensão e enraizamento dos conteúdos lecionados ao longo das aulas teóricas e práticas, de uma forma mais lúdica, descontraída e apelativa.

**Conteúdo**

1. **Introdução** 3
2. **O Jogo Ploy** 3
3. **Lógica do Jogo** 5
   1. Representação do Estado de Jogo 5
   2. Visualização do Tabuleiro 6
   3. Lista de Jogadas Válidas 7
   4. Execução de Jogadas 8
   5. Avaliação do Tabuleiro
   6. Final do Jogo
   7. Jogada do Computador
4. **Interface com o Utilizador**
5. **Conclusões**
6. **Introdução**

Como referido anteriormente, foi proposto o desenvolvimento de um jogo de tabuleiro PLOY na unidade curricular de Programação em Lógica com o objetivo de que a implementação da parte lógica seja feita na linguagem de programação PROLOG e o desenvolvimento de uma representação deste jogo o mais humana possível. Sendo posteriormente possível o desenvolvimento gráfico nesta linguagem de modo a que permita uma comunicação e ligação entre a parte lógica e gráfica.

O desenvolvimento deste projeto tem como objetivo a consolidação e melhor compreensão de todos os conceitos e conteúdos lecionados durante as aulas, assim como uma familiarização com esta linguagem bastante distinta e singular em relação a outras mais populares e tradicionais.

O relatório encontra-se estruturado de maneira a que seja possível contextualizar este jogo, expondo, numa primeira parte, as suas regras, conceitos e objetivos. De seguida, abordar-se-á a parte lógica do jogo, os seus diferentes estados, a forma como a visualização do tabuleiro é feita, exemplificações e esclarecimentos relativamente a partes do código que permitem a execução de movimentos/rotações de peças. Numa última parte, será descrita a parte gráfica do jogo, ou seja, todas as funcionalidades permitidas na interface com o utilizador.

1. **O Jogo Ploy**

O Ploy é um jogo de tabuleiro, criado em 1970 pela empresa *3M Company*, sendo recomendado para jogadores com mais de 10 anos. Este jogo de estratégia e raciocínio é considerado uma das melhores variantes do xadrez, uma vez que este é constituído vários tipos de peças cada uma com as suas capacidades próprias de mobilidade. Cada partida reúne 2 ou 4 jogadores, sendo as regras ligeiramente diferentes conforme o número de jogadores. No entanto, neste projeto desenvolveu-se apenas partidas de 2 jogadores.

Existe dois tipos de jogadas:

* A **jogada de direção** ocorre quando um jogador muda a direção de uma das peças no tabuleiro. Um jogador não pode mudar a direção de mais do que uma peça por jogada. E não se pode usar uma jogada de direção na mesma jogada que uma jogada de movimento.
* Uma **jogada de movimento** é uma jogada em que uma peça se move na horizontal, vertical ou diagonal ao longo das linhas do tabuleiro e para um espaço livre. Esta jogada está limitada pelo número de indicadores da peça e assim como pelas direções do indicador. Se o espaço para o qual a peça quiser ser movida estiver ocupado por outra peça de outra cor o jogador pode capturá-la.

Uma **captura** ocorre quando um jogador move a sua peça para um espaço ocupado por uma peça de outra cor, ou seja, uma peça do seu adversário. Apenas uma peça pode ser capturada por jogada. Após a captura, a peça capturada é retirada do tabuleiro e o seu “captor” move-se para esse espaço.

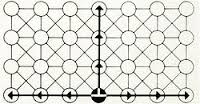


Figura 1: Esquema de movimento das peças conforme o seu formato

Cada peça tem um número de “indicadores direcionais” que determinam as direções em que a peça pode dirigir-se para em qualquer jogada.

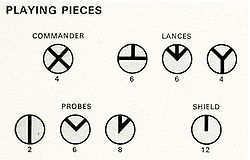


Figura 2: Formatos das peças, respetivas funções e número de peças por formato.

O **“*Shield*”** (ou **escudo**) tem apenas um indicador direcional e só se pode move apenas um espaço de cada vez. No entanto, estas peças são as únicas que podem fazer uma jogada de direção depois de uma jogada de movimento.

As **“*Probes*”** (ou **sondas**) têm dois indicadores direcionais e pode mover-se um ou dois espaços de cada vez.

As **“*Lances*”** (ou **lanças**) têm três indicadores direcionais e podem mover-se um, dois ou três espaços de cada vez.

O **“*Commander*”** (ou **comandante**) tem quatro indicadores direcionais, mas só pode mover-se apenas um espaço por jogada. Se um jogador ou a equipa conseguirem apanhar a peça comandante da equipa adversária o jogo acaba. Outra alternativa de o jogo terminar é capturar todas as peças do jogador oponente exceto o comandante.

O objetivo do jogo é capturar a peça comandante do jogador adversário. O tabuleiro deve ter a configuração representada na figura abaixo.

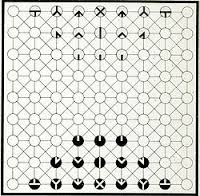


Figura 3: Configuração tabuleiro 2 jogadores

O jogador que tem as peças verde é quem executa a primeira jogada, para permitir simplificar o código o jogador que começa será sempre o jogador 1. Em cada jogada, o jogador pode fazer uma jogada de movimento ou direção, jogando alternadamente até o jogo acabar.

1. **A Lógica do Jogo**
   1. **Representação do Estado do Jogo**

Após diversas conversações com o professor das aulas práticas e estudos de como seria a maneira mais simples e eficaz de representar o tabuleiro, chegou-se à atual em que cada casa do tabuleiro é representado por uma matriz 3x3, sendo no tabuleiro estas casas representadas por uma lista de 9 elementos (exemplo: [l1,0,0,0,1,a,1,0,1,0]),estas casas situam-se sempre nas linhas e colunas pares do tabuleiro, ou seja todas as coordenadas pares do tabuleiro contêm casas vazias ou casa com uma peça de um dos jogadores, e sabemos que uma casa está vazia se a lista for [z,0,0,0,0,0,0,0,0,0]. Nestas listas de 9 elementos a nível lógico consideramos que o primeiro elemento é o contador de indicadores direcionais de uma peça para a lógica, os elementos a 1 ou a 2 dependo do jogador a quem pertence a peça serão os indicadores direcionais e o elemento no índice 5 é o número do jogador da peça que pode tomar o valor 1 ou 2.

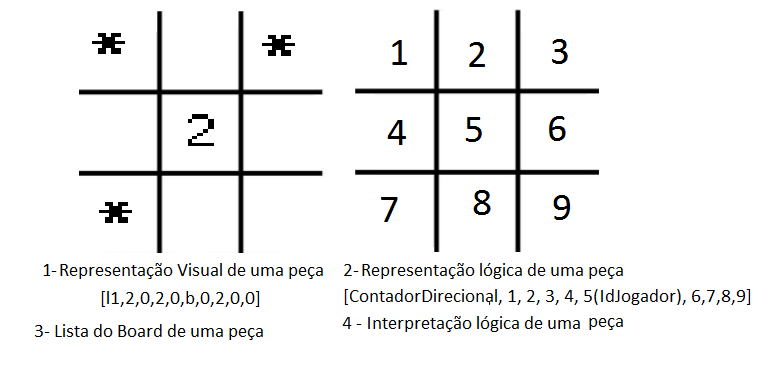


Figura 4: Conversão lógica de uma casa com peça do tabuleiro

O uso da representação das peças em matrizes de 3x3 é necessário para facilitar a rotação destas e a movimentação das peças na horizontal, vertical e diagonal tendo em conta os seus indicadores direcionais.

* 1. **Visualização do tabuleiro**

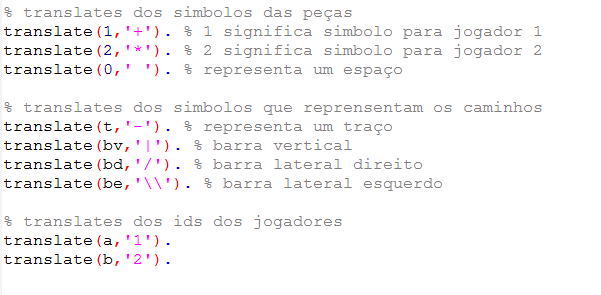
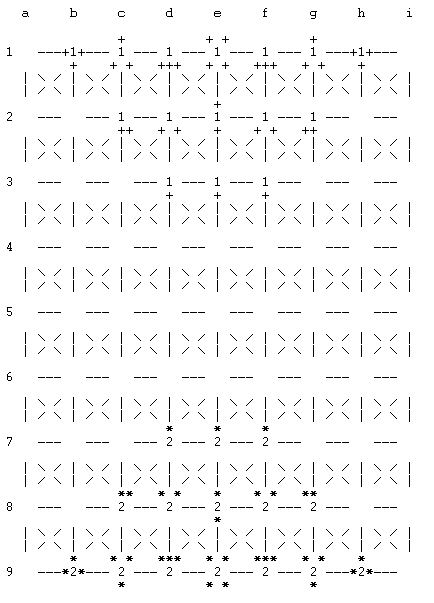
Para que a representação do tabuleiro com as respetivas peças dos respetivos jogadores seja visível e diferenciada foi criada uma estratégia em que as peças do jogador 1 são representadas pelo sinal ‘+’ e as do jogador 2 pelo símbolo ‘\*’. Para criar os possíveis caminhos existentes no tabuleiro usou-se os símbolos abaixo representados.

Figura 5 Predicados que permitem traduzir a linguagem do tabuleiro para os símbolos no display

Figura 6 Visualização do tabuleiro no estado inicial

O predicado principal que permite o display do tabuleiro é:

* **displayBoard(Board)** – chama os predicados auxiliares que fazem o display das letras das colunas e do tabuleiro e guarda em Board o tabuleiro.

Os predicados auxiliares:

* **display\_bords\_up** - faz display das letras das colunas.
* **display\_board([L1,L2|Ls], N)** – predicado recursivo que faz display das diferentes partes das casas (parte de cima, parte do meio e parte de baixo) e dos caminhos, e faz display dos números das linhas começando em N e incrementando até chegar ao fim do tabuleiro.
* **display\_board([L1|[]], N)** – predicado que faz display da última linha do tabuleiro.
* **display\_board([])** – condição de paragem do predicado recursivo.
* **display\_line([L1|Ls], 'Top')**  - faz display da parte de cima de uma casa.
* **display\_line([L1|Ls], 'Mid')** – faz display da parte do meio de uma casa.
* **display\_line([L1|Ls], 'Down')** – faz display da parte de baixo de uma casa.
* **display\_line([], \_Type) -** condição de paragem do predicado recursivo.
* **junta\_pecas([E1,E2,E3|\_Ls],Res)** – predicado que juntas as 3 partes de uma casa (‘Top’/cima,’Mid’/meio,’Down’/baixo) e faz o display delas.
* **display\_peca([E1|Es])** – predicado que converte as variáveis dos tabuleiros nos símbolos e faz o seu display.
* **display\_peca([]) –** condição de paragem do predicado recursivo.
  1. **Lista de Jogadas Válidas**

Uma vez que no Ploy existe dois tipos de jogadas: movimentos e rotações, torna-se necessário averiguar para cada tipo de jogada as jogadas válidas existentes.

Ao movimentar uma peça é pedido ao utilizador a introdução das coordenadas da peça que pretende mover, assim como, a direção para a qual quer mover e o número de casas que quer mover, tendo em conta estes fatores a lista de jogadas válidas é a seguinte:

* a orientação que o jogador escolheu corresponde a um dos indicadores direcionais nessa direção - valida\_orientacao(Orientacao) … preciso da função que vê se o bit orientação está a set
* a coordenada para a qual o utilizador quer mover a peça está vazia ou contém uma peça do jogador adversária - verifica\_casaJogador(Board, X, Y, Orientacao, Player, NovaCasa);

Na rotação, apenas se pretende rodar a peça atual, portanto não é necessário a criação de uma lista de jogadas válidas, uma vez que esta não implica alteração nas restantes casas do tabuleiro.

Colocar os prints das funções das listas

* 1. **Execução de jogadas**

Para ser possível executar uma jogada, tendo o Ploy, dois tipos de jogadas diferentes pede-se ao utilizador que introduza as coordenadas que pretende alterar (read\_coordenadas\_casa(X,Y)), de seguida que tipo de jogada deseja fazer (read\_tipo\_jogada(TipoJogada)), casa seja movimento pede-lhe o número de casas que se quer mover tendo em conta o número de indicadores direcionais na peça e , por fim, pede a direção para a qual quer mover a peça, após receber todos estes valores verifica-se se a peça tem um indicador direcional na direção desejada. Se tiver calcula-se as novas coordenadas no tabuleiro tendo em conta o número de casas que se vai mover, substitui-se nas novas coordenadas a peça e apaga-se nas coordenadas antigas a peça nessa casa. Caso contrário, volta a pedir-se uma nova direção ao utilizador até que introduza uma direção para a qual é possível mover a peça.

Se o utilizador pretender rodar a peça, então após colocar o movimento de rotação como o pretendido, este identifica se deseja rodar a peça para a esquerda ou para a direita

* a coordenada que o utilizador escolheu pertence ao tabuleiro e as suas coordenadas são pares, uma vez que no tabuleiro é nas coordenadas pares que se encontram as casas – valida\_coordenada(X,Y);
* a coordenada que o jogador escolheu pertence a este jogador – valida\_escolhaPeca(X,Y,Peca,Bit);
* o número de casas que o jogador quer mover a peça corresponde ao intervalo entre 1 e o contador dos indicadores direcionais dessa casa - valida\_NcasasUtilizador(NcasasPossiveis,NcasasEscolhidas);

