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 e 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 o trabalho em equipa. Foi necessário estudar o jogo para compreender as principais regras e principais objetivos, para que permitisse uma implementação simplificada e objetiva após feitas as esquematizações dos diferentes modos de jogo, 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 10
   6. Final do Jogo 10
   7. Jogada do Computador 10
4. **Interface com o Utilizador** 11
5. **Conclusões** 12

**Bibliografia** 13

**Anexos**  14

1. **Introdução**

Como referido anteriormente, foi proposto o desenvolvimento de um jogo de tabuleiro PLOY na unidade curricular de Programação em Lógica com o objetivo da implementação da parte lógica ser feita na linguagem de programação PROLOG e do 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 por 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 para 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.

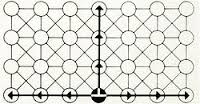


Figura 1: Esquema de movimento das peças conforme o seu formato

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

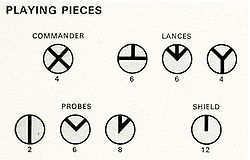


Figura 2: Formatos das peças, respetivas funções e número de peças por formato.

O **“*Shield*”** (ou **escudo**) tem apenas um indicador direcional e só se pode move apenas um espaço de cada vez.

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.

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

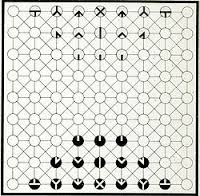


Figura 3: Configuração tabuleiro 2 jogadores

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 10 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 10 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 as orientações para onde a peça se pode mover e o elemento no índice 5 é o número do jogador da peça que pode tomar o valor 1 ou 2.

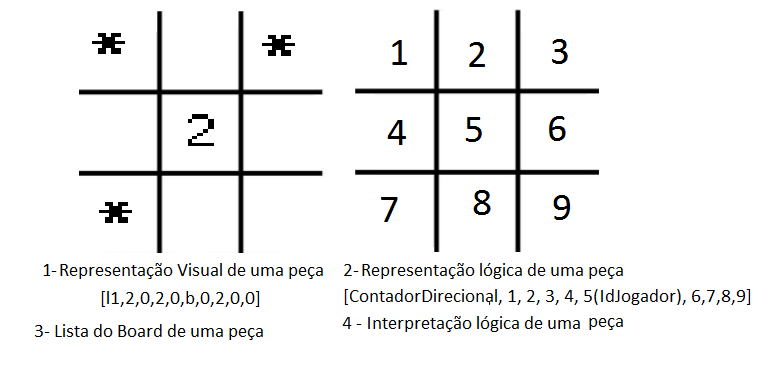


Figura 4: Conversão lógica de uma casa com peça do tabuleiro

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**

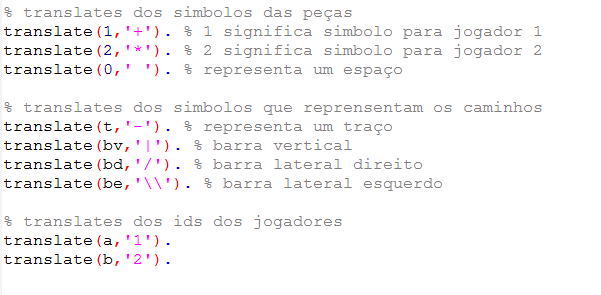
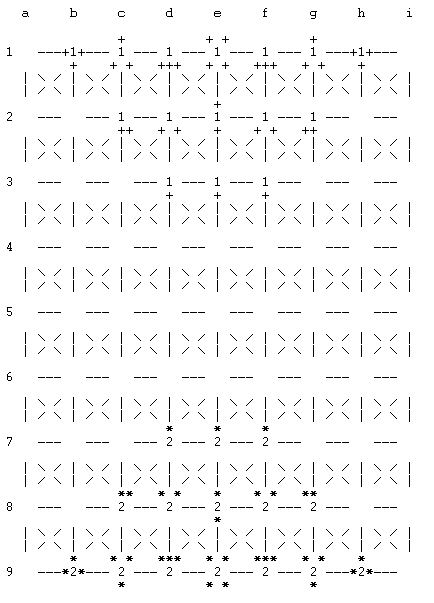
Para que a representação e visualização do tabuleiro com as respetivas peças dos respetivos jogadores seja visível e diferenciada foi criada uma estratégia em que as peças do jogador 1 são representadas pelo sinal ‘+’ e as do jogador 2 pelo símbolo ‘\*’. Para criar os possíveis caminhos existentes no 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 - **verificar\_bitOrientacao**(Board, X, Y, Orientacao, Jogador) (Fig.7).
* a coordenada para a qual o utilizador quer mover está dentro dos limites do tabuleiro e se essa casa está vazia ou contém uma peça do jogador adversária - **pode\_mover**(Board, Xantes, Yantes, NumeroCasas, Jogador, IdPeca, Orientacao) (Fig.8).

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.



Figura 7 Predicado verificar\_bitOrientacao

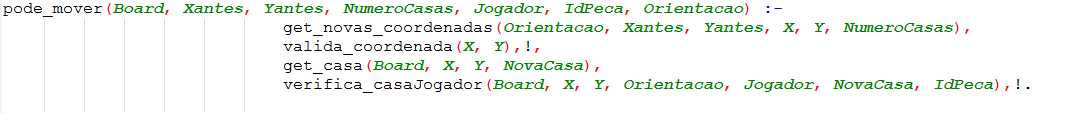


Figura 8 predicado pode\_mover

* 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 a casa para onde se pretende mover a casa tenha uma casa do mesmo jogador, então obriga-se o jogador a rodar a peça. 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.

Durante quer do processo de recolha de informação do jogador, quer de criação da informação do bot é necessário averiguar se estas informações estão corretas. Tal é feito recorrendo-se aos seguintes predicados que têm como objetivo verificar:

* que a coordenada que o jogador escolheu pertence a este jogador – **valida\_escolhaPeca**(X, Y, Player, Bit) (Fig.9);
* se o tipo de jogada que introduziu é válida - **valida\_Movimento**(TipoMove) (Fig.10);
* se o número de casas que deseja mover uma peça está dentro do intervalo de jogadas que é possível fazer com essa peça – **valida\_NcasasUtilizador**(NcasasPossiveis, NcasasEscolhidas) (Fig.11);
* se a orientação introduzida é válida e se essa peça tem um indicador direcional ativo nessa direção – **valida\_orientacaoPossivel**(Board, X, Y, Orientacao, Jogador) (Fig.12);
* Se o sentido para onde pretende rodar a peça é válido - **valida\_rotacao**(Sentido) (Fig.13).

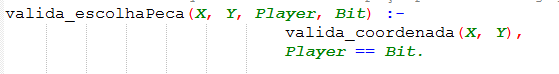


Figura 9 predicado valida\_escolhaPeca

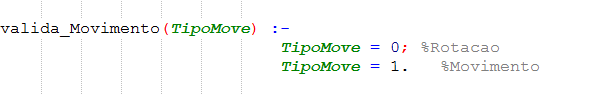


Figura 10 predicado valida\_Movimento

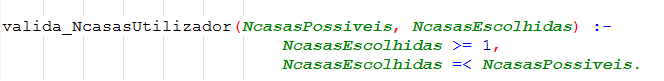


Figura 11 predicado valida\_NcasasUtilizador

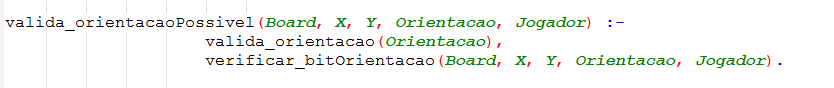


Figura 12 predicado valida\_orientacaoPossivel

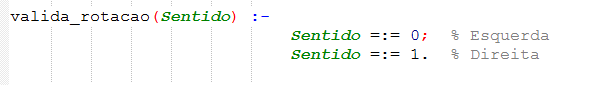


Figura 13 predicado valida\_rotacao

* 1. **Avaliação do tabuleiro**

Uma vez que a cada jogada apenas se verifica a peça que se quer rodar, ou a peça que se vai mover, apenas necessitamos de atualizar o tabuleiro eliminando a casa onde a peça estava para a casa onde a peça se moveu.

Não foi necessário, tendo em conta esta estratégia, a cada jogada fazer uma avaliação ao tabuleiro, mas sim uma atualização ao tabuleiro antigo.

* 1. **Final do Jogo**

O jogo finaliza quando se captura o comandante do jogador adversário, portanto, a cada jogada, se a peça do jogador se mover para uma casa do jogador adversário é necessário averiguar se esta peça adversário é o comandante, se for então a variável IdPeca, variável que nos permite saber se a peça capturada numa jogada foi o comandante, é atualizada para o valor 4, ou seja, o número de contador direcional dessa peça, esta atualização é feita no predicado **valida\_jogador**(Board, X, Y, Orientacao, Jogador, IdPeca) (Fig. 14) e a verificação de que o jogo acabou é feito com o predicado **fim\_deJogo(**IdPeca) (Fig. 15). Caso a peça adversário não seja o comandante, a peça se mova para uma casa vazia ou rode o IdPeca é atualizado com o valor 1.

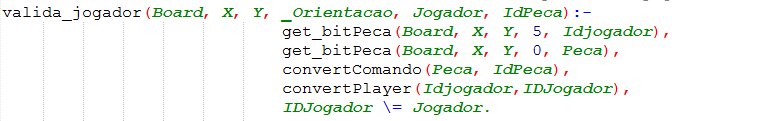


Figura 14 predicado valida\_jogador

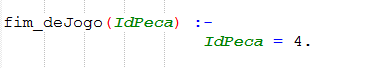


Figura 15 predicado fim\_deJogo

* 1. **Jogado do Computador**

Como foi pedido a criação de 3 modos de jogo Humano vs Humano, Humano vs Computador e Computador vs Computador, foi necessário proceder-se à criação de um jogador em que teria que gerar todas as variáveis necessárias de forma aleatória. Não sendo criado um nível de dificuldade maior, uma vez que não se conseguiu pensar, nem executar uma solução com um mínimo de inteligência possível para este trabalho.

1. **Interface com o Utilizador**

Foram criadas diversas funcionalidades que permitem uma fácil e simples interação do utilizador com a interface, nomeadamente um menu em que o utilizador escolhe o tipo de modo que quer jogar (Fig. 16).

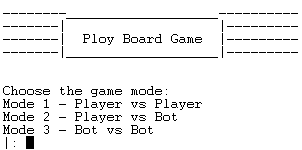


Figura 16 Menu inicial modo de jogo

Durante as jogadas do jogador humano é pedido a introduções de várias escolhas, nomeadamente, as coordenadas, o tipo de jogada e dependendo do tipo de jogada outras opções.

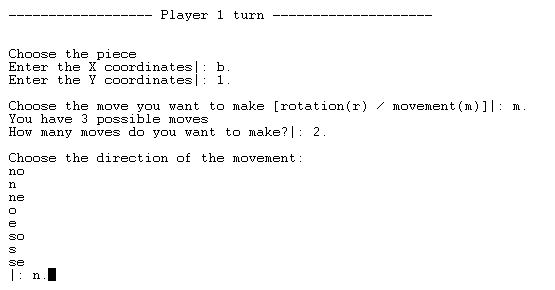


Figura 17 jogada movimento humano

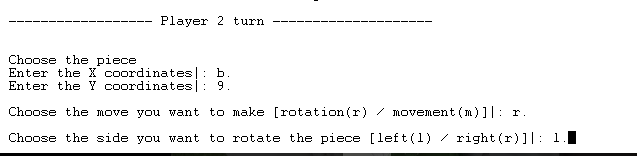


Figura 18 jogada rotação humano

No término do jogo é feito o display de uma mensagem que mostra o jogador que ganhou a partida (Fig. 19).

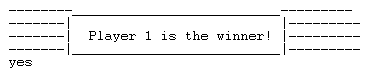


Figura 19 mensagem de vitória

1. **Conclusões**

Após várias horas de trabalho exaustivo para a implementação do Ploy, tendo em conta o resultado final, pode-se afirmar que os principais objetivos foram cumpridos e o resultado final é bastante positivo.

Todas as funcionalidades e regras do jogo tiveram que ser bastante planeadas, para que a implementação dos diferentes modos fosse executada de uma forma simples e direta. Em termos de código, foi necessário atribuir a cada elemento do grupo diferentes tarefas no decorrer da implementação do Ploy, no final, pode afirmar-se que o trabalho realizado por ambos os membros foi bastante equilibrado.

Considera-se que os verdadeiros entraves foi como implementar na linguagem as regras todas do PLOY e evitar ciclos infinitos em caso de peças bloqueadas por outras peças do mesmo jogador, estes obstáculos foram ultrapassados criando-se alternativas diferentes. Apesar, de se considerar o resultado final bastante satisfatório, pode-se acreditar que existe sempre espaço para melhorias, tanto a nível estrutural como tempo de execução de código. Claro que um maior conhecimento, assim como, um maior contacto desta linguagem de programação permitiria aprimorar o resultado final.

Em suma, esta linguagem permite um grande desenvolvimento do raciocínio lógico e a realização deste trabalho além de ter permitido isso, permitiu também consolidar todos os conceitos teóricos e práticas dados nas aulas, portanto, acredita-se que toda esta realização de código foi bastante benéfica para os alunos desta unidade curricular.

**Bibliografia**

* <https://en.wikipedia.org/wiki/Ploy_(board_game)>
* <http://www.thefullwiki.org/Ploy_(board_game)>
* Sterling, Leon; [The Art of Prolog](http://catalogo.up.pt/F/-?func=find-b&local_base=FEUP&find_code=SYS&request=000455890). ISBN: 0-262-69163-9
* Slides utilizados nas aulas

**Anexos**

O código fonte deste relatório encontra-se na pasta PLOG\_TP1\_FINAL\_PLOY1.zip anexado junto deste relatório.