**Eximo**

**Relatório Final**



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

**Grupo 04: Eximo**

Henrique Manuel Martins Ferrolho - 201202772 João Filipe Figueiredo Pereira - 201104203

Faculdade de Engenharia da Universidade do Porto

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

1. de Novembro de 2014

# Resumo

O relatório ainda se encontra por terminar.

# Introdução

No âmbito da unidade curricular de Programação em Lógica, do curso Mestrado Integrado em Engenharia Informática e Computação, foi-nos sugerido a elaboração de um jogo em PROLOG. Esse jogo foi selecionado pelo grupo dentro de um leque de várias opções que nos foram disponibilizados pelos docentes.

A escolha em torno de Eximo baseou-se no estilo que o prescrevia. A jogabilidade simples e a combinação de jogadas possíveis tornaram-no num bom partido e a motivação por parte do grupo para a usa implementação foi forte. Assim como nas Damas, é possível haver uma boa prática mental e estratégica com o desenrolar de uma partida entre dois elementos. O grupo tomou em consideração todos os pontos descritos em cima para a escolha de jogo final, que se viria a tornar o primeiro trabalho realizado em Programação em Lógica.

O objetivo deste primeiro trabalho foi a aplicação dos primeiros conceitos interiorizados nas aulas teóricas e desenvolvidos nas aulas práticas da cadeira. Este método de avaliação torna-se importante pois permite-nos avaliar os conhecimentos que adquirimos até então e saber se somos ou não capazes de, com uma linguagem de programação nova e um paradigma completamente diferente do que estamos habituados, produzir algo de útil para o quotidiano e futuro.

Este relatório encontra-se dividido em várias secções tais como:

* a história, regras e jogadas possíveis do jogo Eximo;
* a lógica implementada no jogo, descrição do projeto e da sua implementação em PROLOG;
* o modo de interação entre o utilizador e o programa, entre jogabilidade e visualização;
* uma conclusão final do grupo a este primeiro desafio com o ambiente PROLOG;
* uma lista de referências a livros, artigos, páginas web, etc.;
* uma séries de elementos anexados, como o código do projeto e outros.

# O Jogo Eximo

## História

Eximo é um jogo de tabuleiro da família das Damas, concebido em 1 de Fevereiro de 2013.

## Detalhes do Jogo

O jogo realiza-se num tabuleiro de dimensões 8x8, em que as casas têm todas cores semelhantes. Cada jogador começa com 16 peças colocadas em locais predefinidos no respetivo lado do tabuleiro, como mostra a imagem abaixo.

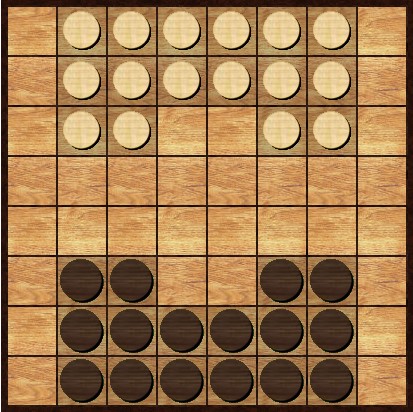




Figura 1 - Peça Preta

Figura 2 - Peça Branca

Figura 3 - Tabuleiro

No jogo, as movimentações e as capturas podem ser ortogonais ou diagonais. Há apenas um tipo de peça: os homens. Os homens podem saltar sem efetuar captura. Quando um homem atinge a última linha, ocorre a libertação de outro homem.

## Objetivo

O objetivo do jogo é, tal como nas Damas, **capturar todas as peças** do oponente, saltando sobre elas, ou **incapacitar o adversário** de realizar qualquer movimento.

## Jogada

Em cada jogada, um jogador pode fazer uma de duas ações: **mover ou capturar**.

## Movimento

Uma peça pode mover-se em três direções: para a frente ou na diagonal (**norte, nordeste ou noroeste**). Numa jogada, o movimento nunca pode ser efetuado para a retaguarda.

Existem dois tipos de movimentos: **Normal e Salto**.

**Movimento Normal**: uma peça move-se para uma **casa adjacente e vazia**.

**Movimento em Salto**: **uma peça salta sobre uma peça aliada adjacente, se e só se a casa correspondente** (ao lado da peça aliada) **estiver vazia**, colocando assim a peça nessa casa. Se a mesma peça do jogador puder continuar a realizar o mesmo movimento de salto sobre outra peça amigável então terá de o fazer. **Durante um movimento de salto a peça não pode capturar peças inimigas**.

Quando existe **mais do que uma forma de saltar**, o jogador **pode escolher a peça que irá usar para executar o salto**, bem como o tipo de salto ou sequência de saltos a fazer. Não é obrigatório que a sequência de saltos escolhida pelo jogador seja aquela que possui o maior número de saltos; porém, **após escolher uma sequência, o jogador deve executar todos os saltos possíveis**.

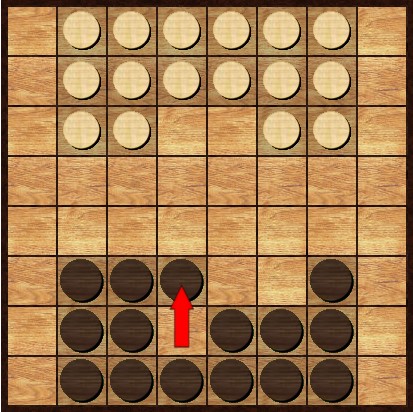
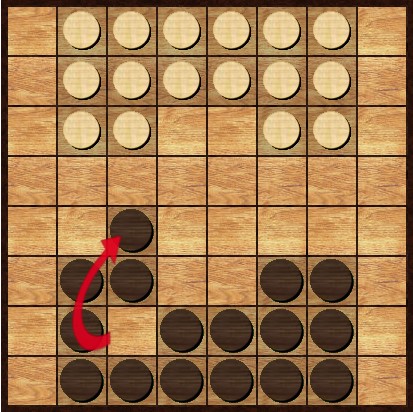


Figura 5 – Movimento em Salto

Figura 4 – Movimento Normal

Figura 6 - Movimentos

## Captura

Um **jogador pode capturar** em cinco direções: **frente**, **diagonal para a frente**, **direita** ou **esquerda** (norte, nordeste, noroeste, este ou oeste).

**Captura**: um jogador **salta sobre uma peça adjacente do adversário**, se a **próxima casa**, na mesma direção, **estiver vazia**, colocando, assim, a peça sobre essa casa. A **peça do oponente é removida do tabuleiro**. Se a **peça** do mesmo jogador **puder continuar a capturar outras peças do adversário**, então **deve fazê-lo**. A **captura é obrigatória** e deve continuar enquanto for possível.

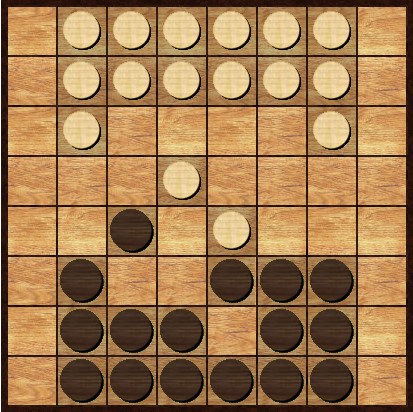
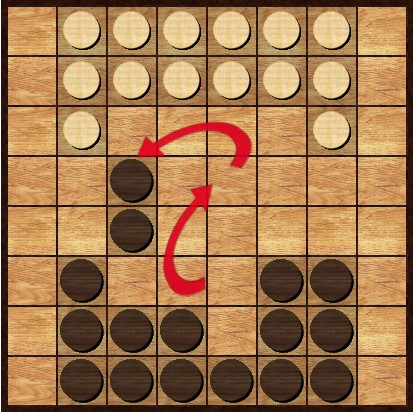


Figura 8 - Estado posterior à captura

Figura 7 - Estado anterior à captura

Tal como no **Movimento de Salto**, **o jogador escolhe** livremente **qual a sequência de saltos a efetuar**.

## Última Linha

Quando uma peça atinge a extremidade do tabuleiro, essa peça é removida de imediato e o jogador recebe dois movimentos-extra para efetuar nesse mesmo momento: colocar duas peças novas numa casa vazia localizada nas duas primeiras linhas, à exceção das quatro casas laterais (duas do lado esquerdo, e duas do lado direito).

# Lógica do Jogo

## Representação do Estado do Jogo

O estado do jogo é representado por uma lista de quatro elementos. Todo o código relativo à representação do estado do jogo encontra-se no ficheiro gameClass.pl.

O nosso objetivo ao armazenar toda a informação do estado do jogo numa lista era aproximar o desenvolvimento do projeto a um paradigma semelhante ao de programação orientada a objetos. No ficheiro gameClass.pl podem ser observados vários predicados que realçam este paradigma: os predicados *getters* e *setters*, que obtêm e modificam um elemento do estado do jogo, por exemplo.

O primeiro elemento da lista que compõe a *classe* do estado do jogo é uma lista de listas que representam o estado atual do tabuleiro de jogo, ou seja, o conteúdo de cada posição no tabuleiro. A disposição das peças no tabuleiro é, portanto, armazenada nesta *matriz*.

O segundo elemento da lista é outra lista de dois elementos - um par - cujo conteúdo é, respetivamente, o número de peças que o jogador branco e preto têm sobre o tabuleiro.

O terceiro elemento determina qual dos jogadores deve efetuar a jogada naquele estado. Para um estado de jogo em que seja a vez do jogador branco efetuar uma jogada, o terceiro elemento será portanto: *whitePlayer*; por sua vez, se fosse o jogador preto o próximo a jogar, o terceiro elemento seria *blackPlayer*.

Finalmente, o quarto elemento contém o modo de jogo. Existem três modos de jogo: humano contra humano, humano contra computador e computador contra computador. Os átomos que representam estes três modos de jogo diferentes são, respetivamente: ***pvp***, ***pvb*** e ***bvb***.

## Visualização do Tabuleiro

Os predicados responsáveis pela visualização do tabuleiro na linha de comandos encontram-se no final do ficheiro eximo.pl. Os predicados são quase todos recursivos e foram desenvolvidos por camadas.

Para imprimir o tabuleiro basta chamar o predicado ***printBoard(+Board)***, onde ***Board*** é um tabuleiro de um estado de jogo. Por sua vez, este predicado faz uso de outros predicados com funções cada vez mais específicas para imprimir o tabuleiro de uma forma simples, formatada e concisa sempre que necessário.

## Execução de Jogadas

Em cada jogada, é pedido ao jogador que tem a vez de jogar as coordenadas da peça no tabuleiro que deseja movimentar. Logo depois de as coordenadas serem validadas, o programa verifica que as coordenadas correspondem a uma peça do jogador que tem a vez de jogar e não a uma das peças do oponente. Esta verificação é feita com recurso ao predicado ***validateChosenPieceOwnership(+SrcRow, +SrcCol, +Board, +Player)*** e caso falhe, o programa retrocede e pede ao jogador para inserir outras coordenadas.

Uma vez validadas as coordenadas da peça a mover, é pedido ao jogador que insira as coordenadas do destino da peça que selecionou. Após essas coordenadas serem validadas, verifica-se se as coordenadas de origem e destino do movimento não são as mesmas com o predicado ***validateDifferentCoordinates(+SrcRow, +SrcCol, +DestRow, +DestCol)***. Caso as coordenadas de origem e destino sejam as mesmas, o programa retrocede até ao ponto inicial.