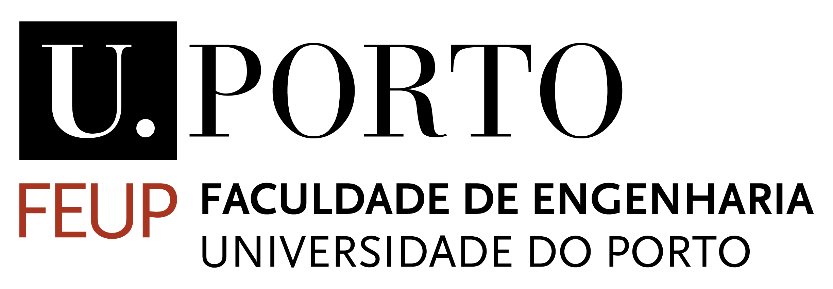
Martian Chess

Relatório Final



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo: Martian\_Chess\_1

Tiago José Grosso Pacheco – up201402722

Vasco Ferreira Ribeiro –up201402723

Faculdade de Engenharia da Universidade do Porto

13 de Novembro de 2016

**Resumo**

No âmbito da cadeira de Programação em Lógica, integrada no plano curricular do 3º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto, foi proposta a realização de um trabalho que consistiria na implementação em *SICStus Prolog* de um jogo de tabuleiro. Para a concretização dessa proposta, foi escolhido o jogo ***Martian Chess***.

Assim sendo, o objetivo em vista era criar o jogo ***Martian Chess*** de forma a que este pudesse ser jogado no modo “Jogador VS Jogador” e “Jogador VS Computador”, através de uma interface baseada em texto.

Fazendo uso das competências adquiridas através do trabalho efetuado nas aulas práticas e através de uma alguma pesquisa independente, começou-se o desenvolvimento deste projeto, cujo produto final se pretendia que fosse razoavelmente completo em termos de regras de jogo e interface com o utilizador.

Conteúdo

**Não foi encontrada nenhuma entrada de índice.**

Introdução

Tendo como proposta a realização de uma implementação em *SICStus Prolog* de um jogo de tabuleiro, optou-se pela tema ***Martian Chess*** por possuir regras que não são habituais no que toca a este tipo de jogos. Fala-se, por exemplo, da regra que diz respeito à posse das peças, posse essa que é determinada somente pela posição atual da peça. Esta regra leva a uma necessidade de pensamento estratégico diferente, pois obriga a que, frequentemente, se tenha de abdicar de uma peça para colher alguma vantagem. É, portanto, um jogo de cálculo de risco e de valor táctico de cada componente do tabuleiro.

Tinha-se, portanto, o objetivo final de construir o jogo ***Martian Chess*** em *Prolog.* Para tal, decidiu-se criar uma lista de listas para representar o tabuleiro de jogo e implementou-se uma interface baseada na indicação do número da linha e da coluna de cada peça que se deseja mover e da célula de destino de dita peça. Através de texto e de uma representação visual/textual do tabuleiro, dá-se aos jogadores todas as indicações necessárias, tanto a nível de instruções de jogo, como de progresso da partida e de resultado final.

Neste relatório, abordar-se-á os conceitos mais pertinentes relativamente à concretização deste projeto. Vai-se falar em que consiste o jogo ***Martian Chess***, quais as suas regras e qual a sua origem e história. Far-se-á, seguidamente, uma explicação da lógica do jogo em *Prolog*. Relativamente a esse tópico, vai ser feita referência aos vários aspetos do desenvolvimento deste trabalho, bem como às decisões tomadas para a sua realização. Ir-se-á discutir as várias fases e movimentos do jogo, bem como a sua implementação em *Prolog* e a forma como o utilizador interage com o programa. Por último, irão ser apresentadas as conclusões formuladas pela realização deste projeto.

O Jogo – ***Martian Chess***

O jogo de estratégia ***Martian Chess*** foi desenhado por Andrew Looney, um dos inventores das pirâmides de *Icehouse*. As primeiras versões das regras do jogo datam de 29 de janeiro de 1995, sendo, assim, considerado o segundo jogo de pirâmides mais antigo, logo a seguir ao famoso *Icehouse*.

***Martian Chess*** é um jogo de tabuleiro parecido com o Xadrez, uma vez que se joga num tabuleiro de Xadrez, temos vários tipos de peças e o objetivo é capturar as peças do adversário. No entanto, o controlo das peças é determinado pela localização, que determina quais as peças se pode mover. Cada jogador tem um quadrante de jogo, podendo apenas mover as peças presentes ao quadrante que lhe pertence e atacar as presentes nas outras zonas. O objetivo do jogo é fazer mais pontos (cada peça capturada dá pontos – Rainha: 3 pontos; Drone: 2 pontos; Pião: 1 ponto.

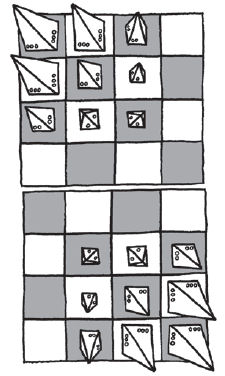
O jogo pode ser jogado por 2 ou 4 jogadores. Para efeitos deste trabalho, iremos abordar somente a variante de 2 jogadores.

Figura 1 - Posição Inicial do Jogo

O tabuleiro tem uma posição inicial de jogo, com as peças dispostas em cantos opostos e com os Piões dispostos na frente, seguidos dos Drones e posteriormente das Rainhas.

Cada Peça tem um tipo de movimento; contudo, saltar por cima de peças não é permitido:

* Rainha – Pode mover-se em todas as direções o número de casas que pretender;
* Drone – Pode mover-se 1 ou 2 casas na horizontal ou na vertical;
* Pião – Pode mover-se 1 casa na diagonal.

Figura 2 - Movimento Rainha



Figura 3 - Movimento Pião

Figura 4 - Movimento Drone

O jogo tem, no entanto, a possibilidade de, durante o jogo, fazer algumas promoções. Isto é, quando um jogador não possuir rainhas, ele poderá criar uma, movendo um Drone para uma casa onde esteja um Pião, fundindo-os. O mesmo pode acontecer juntando 2 Piões para dar origem a um Drone (quando não possuir nenhum Drone no seu quadrante).

O jogo termina quando um dos quadrantes estiver totalmente vazio.

Lógica de Jogo

As sub-secções seguintes referem-se à lógica de implementação do jogo ***Martian Chess*** em *Prolog* e às decisões tomadas pelos autores de forma a conseguirem traduzir o jogo de tabuleiro da melhor forma possível.

Representação do Estado do Jogo

A representação do jogo será feita com uma lista de listas, em que cada elemento de uma lista (um caracter) representa uma das posições do tabuleiro. A lista “Board” terá, assim, várias listas, cada uma delas representativa de uma das linhas do tabuleiro:

board([[r,r,d,v],

[r,d,p,v],

[d,p,p,v],

[v,v,v,v],

[v,v,v,v],

[v,p,p,d],

[v,p,d,r],

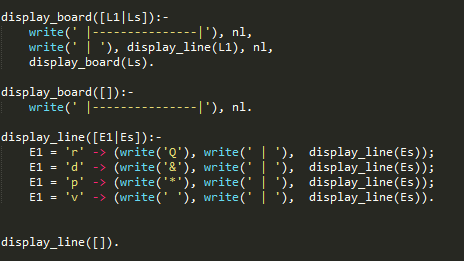
[v,d,r,r]

]).

Desta forma, assegura-se uma boa abstração do tabuleiro que permite trabalhar com listas para alterar (mover) os elementos.

Visualização do Tabuleiro

Para permitir a visualização do Tabuleiro, foram implementadas algumas funções em *Prolog*.

Primeiro chama-se a função *display\_board*. Esta função escreve, de forma recursiva, cada linha, invocando a função *display\_line* para cada lista dentro da lista *board* (lista de listas). Entre cada linha, vai-se colocando hífens de forma a tornar mais legível o tabuleiro.

A função *display\_line*, por sua vez, recebe uma lista (cada lista será uma das linha do board) e imprime todos os elementos, colocando hífens verticais entre cada um deles por uma questão de legibilidade e aparência.

Figura : Funções para escrever o tabuleiro

Como forma de facilitar a visualização de cada peça, optou-se por atribuir a cada uma delas um símbolo, sendo as Rainhas representadas por um “Q”, os Drones por um “&” e os Peões por um “ \* ”.

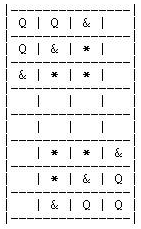


Figura 6: Visualização do Tabuleiro

Jogadas Válidas

Como já foi referido, no jogo ***Martian Chess***, cada peça tem um tipo de movimento que pode efetuar. Em primeiro lugar, nenhuma peça pode passar por cima de outra no seu movimento. Para tal, criou-se funções para verificar se o caminho de um movimento está desimpedido. **(ADICIONAR IMAGEM DAS FUNÇÕES QUANDO ESTIVEREM ACABADAS)**

Quanto às Rainhas, estas podem deslocar-se horizontal, vertical e diagonalmente, podendo o seu movimento pode ter qualquer comprimento. Os Drones, podem-se deslocar na vertical e horizontal uma ou duas casas. Por outro lado, os Peões apenas se podem mover na diagonal, e apenas numa só casa.

Execução das Jogadas

De forma a executar as jogadas pretendidas, começamos por chamar a função *askMove(X),* que pergunta ao utilizador qual a jogada pretendida pelo utilizador. De seguida, é chamada a função *movePiece(Board, InitLine, InitCol, DestLine, DestCol)*, que verifica a validade da jogada e efetua a mesma, em caso afirmativo.

Avaliação do Tabuleiro

A avaliação do Tabuleiro é feito de forma muito simples, no loop do jogo é verificado a função end\_game, caso dê true o loop termina e vai para o fim do jogo. Por outro lado, caso retorne falso então o loop continua e permite ao jogador fazer um novo movimento, chamando a função *askMove(X).*

Final do Jogo

O jogo termina quando um dos quadrantes dos jogadores se encontra sem peças. Assim, temos 2 funções que verifica se o primeiro e segundo quadrante se encontra com peças, em caso negativo o jogo termina, verifica a pontuação e o jogador vencedor é identificado.