**Nodes**

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Grupo Nodes\_02:

Nuno Miguel Freitas – up201404739

Daniel Garrido – up201403060

Faculdade de Engenharia da Universidade do Porto

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

16 de Outubro de 2016

# Resumo

Este projecto consiste num jogo de tabuleiro em que dois jogadores competem para rodear o *Node oponente* com as suas *units.* As *units* podem ser movidas se estiverem no caminho de linhas de transmissão, linhas rectas que saem dos *Nodes*, quer do jogador em questão quer do oponente. Um jogador pode mover quantas *units* quiser as vezes que quiser até mover o *Node* um espaço, aí o turno passa para o rival.

O objectivo final deste projecto foi criar um programa em Prolog que permitisse dois jogadores jogar ***Nodes*** um contra o outro, ou apenas um contra uma intelência artificial. O projecto foi concluido com sucesso, sendo que todas as directrizes para a concretização do trabalho foram seguidas e terminadas com êxito.

Durante o desenvolvimento do projecto deparámo-nos com um número de dificuldades incluindo desde a conquetenação de várias listas até à criação da inteligência artificial, estas derivadas principalmente da falta de experiência aquando o uso de Prolog. No entanto todas as barreiras ao sucesso foram superadas através de pesquisa e estudo.

Índice

[Resumo 1](#_Toc466842171)

[Introdução 3](#_Toc466842172)

[O Jogo 4](#_Toc466842173)

[Descrição 4](#_Toc466842174)

[Componentes 4](#_Toc466842175)

[Começar o jogo 4](#_Toc466842176)

[Regras 5](#_Toc466842177)

[Linhas de Comunicação 5](#_Toc466842178)

[*Relay* 5](#_Toc466842179)

[Interceção 5](#_Toc466842180)

[Movimentar as *Units* 5](#_Toc466842181)

[Salto 5](#_Toc466842182)

[Mover o *node* (acabar o turno) 6](#_Toc466842183)

[Ganhar 6](#_Toc466842184)

[Lógica do Jogo 7](#_Toc466842185)

[Visualização do Tabuleiro 7](#_Toc466842186)

[Lista de Jogadas Possíveis 7](#_Toc466842187)

[Execução de Jogadas 8](#_Toc466842188)

[Escolha da unidade e do movimento e verificação 9](#_Toc466842189)

[Avaliação do Tabuleiro 9](#_Toc466842190)

[Final do Jogo 9](#_Toc466842191)

[Interface 9](#_Toc466842192)

[Conclusões 10](#_Toc466842193)

# Introdução

Este trabalho foi feito no âmbito da cadeira Programação em Lógica do curso Mestrado Integrado em Engenharia Informática e de Computação e tem como objectivo final criar um programa que permita jogar o jogo de tabuleiro ***Nodes*** no modo de dois jogadores, em que esses dois jogadores jogam um contra o outro, ou de um jogador, em que este joga contra uma inteligência artificial.

Este relatório será estruturado a partir das directrizes dadas para a construção do relatório, como tal abordará os seguintes tópicos:

* **O Jogo** – Descrição sucinta do jogo, história e regras.
* **Lógica do jogo** – Descrição do desenvolvimento do projecto , separada de modo a abordar os vários tópicos criterizados pelo enunciado do projecto.
  + **Representação do estado do Jogo**
  + **Vizualização do Tabuleiro**
  + **Lista de Jogadas Válidas**
  + **Execuçãp de Jogadas**
  + **Avaliação do Tabuleiro**
  + **Final do Jogo**
  + **Jogada do computador**
* **Interface com o Utilizador** – Descrição do ódulo de interface com o utilizador
* **Conclusões**

# 

# O Jogo

## Descrição

Nodes é um jogo de tabuleiro com inspirações em clássicos como o xadrez e as damas. O jogador controla os movimentos das *units* para proteger o seu *node* das *units* do adversário.

As *units* podem mexer-se tantas vezes quantas o jogador quiser, desde que estas permaneçam ao alcance do sinal do *node*. Para superar e vencer o adversário é necessário gerir esta ligação entre as *units* e o *node* de ambos os jogadores. Ganha o jogador que conseguir bloquear todas as linhas de comunicação do oponente.

## Componentes

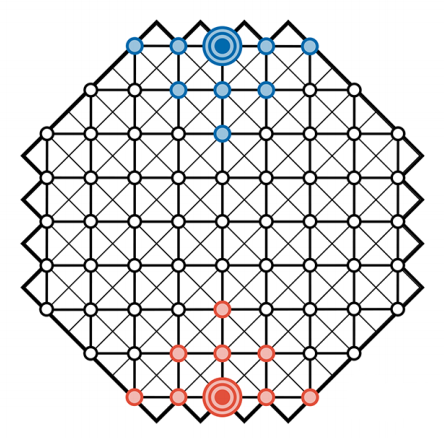
**Units**: Cada jogador começa e acaba o jogo com 8 *units*. Estas são as tropas do jogador. Necessitam do sinal do *node* para se poder movimentar. Conseguem bloquear as linhas de comunicação do oponente.

**Nodes**: Atuam como uma torre de comunicação. Transmitem sinais (na vertical, horizontal e nas diagonais) que habilitam as *units* a movimentarem-se. Defender o próprio *node* e atacar o do adversário forma o núcleo da estratégia ao longo de um jogo.

**Tabuleiro:** Os círculos brancos são os espaços onde todas as peças têm de estar no começo e termino de cada turno.

As linhas horizontais e verticais são as ruas. Estas trasmitem linhas de comunicação e permitem o movimento das peças entre espaços.

As linhas diagonais são as condutas. Estas apenas transmitem linhas de comunicação. Nenhuma peça pode movimentar-se por estas vias.



## Começar o jogo

Antes de o jogo começar, cada jogador deve dispor as peças no tabuleiro em forma de pirâmide, protegendo o *Node*. As ruas devem estar dispostas verticalmente e horizontalmente e as condutas diagonalmente. O jogador mais novo é o primeiro a jogar.

## Regras

### Linhas de Comunicação

As linhas de comunicação são produzidas pelos *nodes*. No total cada *node* transmite oito sinais, um em cada rua e conduta, que se estendem até ao fim do tabuleiro. Este sinal permite que as *units* de ambos os lados se movimentem, desde que estejam em linha com um dos sinais. O jogador deve preocupar-se em manter as suas *units* na linha de fogo do sinal, e as *units* do adversário fora.

Cada node transmite linhas de comunicação, mas estas não são exclusivas às *units* da cor do *node*. As *units* do adversário podem usar as linhas de comunicação do *node* do jogador e vice-versa.

### *Relay*

Se entre um *node* e uma *unit* que deseje mover estiver uma *unit* amigável, esta não bloqueia o sinal. Independentemente da quantidade de *units* amigáveis entre um *node* e a *unit* em questão, esta, a que se encontra mais distante, ainda recebe o sinal de comunicação, podendo mexer-se.

### Interceção

Se entre um *node* e uma *unit* que deseje mover estiver uma *unit* inimiga, esta bloqueia a linha de comunicação para todas as *units* amigáveis nessa direção.

### Movimentar as *Units*

Esta é a primeira fase de um turno. O jogador pode movimentar todas as units na ordem que desejar, as vezes que desejar. Atenção que apenas as *units* alcançadas por uma linha de comunicação têm a habilidade de se movimentarem ao longo das ruas. A partir do momento que uma *unit* perde contacto com uma linha de comunicação, esta não pode voltar a mexer-se nesse turno. As *units* apenas podem deslocar-se para espaços abertos, ou seja, espaços sem outras peças. Se a *unit* estiver rodeada, não pode mexer-se.

### Salto

Um salto ocorre quando uma *unit* “salta” uma unit adversária. Para tal poder acontecer é necessário verificarem-se duas condições. Em primeiro lugar, a *unit* tem de estar posicionada numa linha de comunicação ao longo de uma rua. Seguidamente, o local de destino da *unit* tem de estar alinhado com a mesma rua e ao longo da mesma linha de comunicação de quando começou o salto.

### Mover o *node* (acabar o turno)

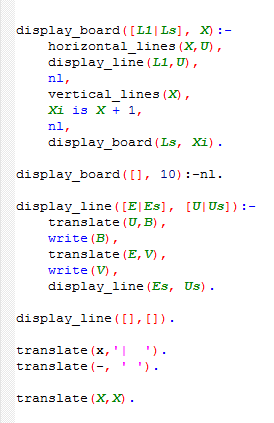
Quando o jogador já não desejar mexer mais as suas *units*, este pode acabar o turno mexendo o seu *node* uma vez numa direção à escolha. O *node* apenas se pode movimentar para um espaço vazio, ao longo de uma rua.

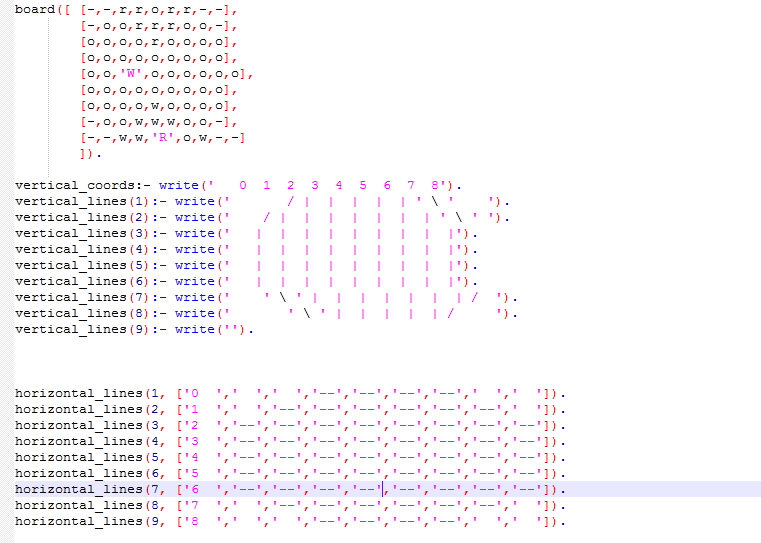
### Ganhar

O jogo acaba quando um dos jogadores não conseguir mexer o seu *node* no final do seu turno, ou seja, o node está rodeado por *units* adversarias ou por paredes do tabuleiro em todas as direções em que se pode movimentar (vertical e horizontal). Mesmo com o seu *node* já rodeado por *units* adversárias, é possível o jogador conseguir rodear o *node* do oponente. Neste caso o jogador que ficou rodeado primeiro perde, visto que não é capaz de terminar o seu turno.

# Lógica do Jogo

## Visualização do Tabuleiro

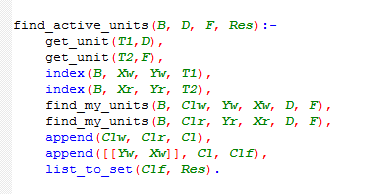
O tabuleiro é uma lista de listas de tamanho fixo, 9 colunas e 9 linhas, e é utilizado um predicado que recursivamente escreve o tabuleiro para vizualização. Para ajudar a vizualização do estado de jogo também são visiveis as linhas por onde as *units* podem mover. Isto pode ser tudo confirmado no código abaixo.



## Lista de Jogadas Possíveis

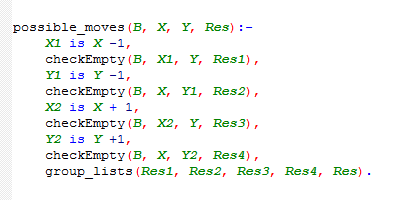
Devido à quantidade exagerada de jogadas que um jogador pode fazer num turno de Nodes decidimos dividir o turno de um jogador em 2 fases principais, escolher uma unidade válida e escolher as coordenadas da posição para onde movê-la, e mostrar as escolhas possíveis durante cada fase. Estas duas fases repetem-se até o jogador mover o *Node*.

Para encontrar a lista de unidades activas o programa executa o seguinte:

* A partir dos argumentos recebidos determina a posição do *Node* aliado e do oponente
* Determina todas as unidades activadas pelo *Node* do jogador e do oponente
* Junta essas duas listas e adiciona as coordenadas do node à lista

Para encontrar a lista de movimentos possíveis o programa efetua o seguinte:

* Verifica para os dois sentidos de cada direção vertical e horizontal se esses espaços estão vazios e cria listas com as coordenadas dos espaços vazios
* Agrupa as listas numa lista de listas



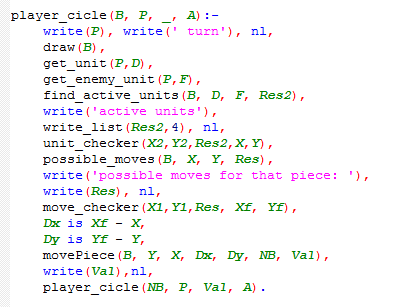
## 

## Execução de Jogadas

Em cada jogada é mostrado ao jogador uma lista de unidades que pode mover em forma de coordenadas . Após o jogador escolher uma unidade que se pode mexer é lhe mostrado uma lista de coordenadas para onde o jogador pode mexer essa unidade. O processo repete-se até o jogador escolher o Node e movê-lo, tornando-se o turno do oponente.

A jogada de um jogador é controlada por um ciclo que faz as seguintes tarefas:

* A partir do jogador descobre as unidades aliadas e inimigas
* Cria uma lista de unidades aliadas activadas, que se podem mexer e escreve essa lista para o jogador poder vê-la
* Pede ao jogador as coordenadas da unidade que quer mexer e verifica se essa consta na lista
* Cria uma lista de movimentos possíveis dependendo da unidade escolhida e escreve essa lista para o jogador poder vê-la
* Pede ao jogador as coordenadas para onde quer mover a unidade e verifica se essa posição consta na lista de espaços livres
* Move a peça
* Recomeça o ciclo



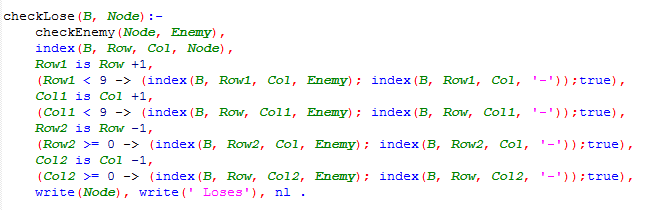
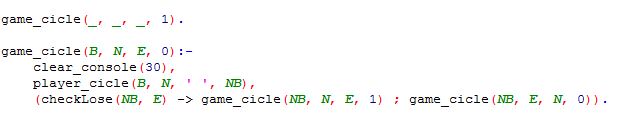
### Escolha da unidade e do movimento e verificação

Durante a jogada é mostrada aos jogadores uma lista de possiveis escolhas de unidade, e posteriormente de possivel movimento. Estas listas e a sua criação foram explicadas mais acima no tópico da lista de jogadas possíveis. Estas listas são também usadas para verificar se as escolhas do jogador são legais face às regras do jogo, se assim não for notifica o jogador e pede as coordenadas outra vez.

## Avaliação do Tabuleiro

A avaliação do tabuleiro ocorre durante a criação de listas de jogadas , não achá-mos que fosse necessário verificar mais o tabuleiro sem ser para a condição de vitória que vai ser discutido mais abaixo.

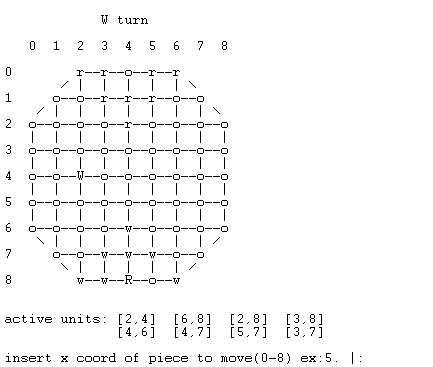
## Final do Jogo

 O programa verifica depois do final de um ciclo de jogo para um jogador se o *Node* do jogador oponente do está rodeado de units inimigas. Se assim for esse jogador perde.

# 

# Interface

Utilizando o comando “node.” O programa mostra um menu com o logo dos ***Nodes***, e pede para premir “Enter”. Depois de premir pode escolher o modo de jogo: “Contra A.I” ou “Contra outro Jogador”.

Durante o jogo é mostrado o tabuleiro no seu estado e são dadas as listas de unidades válidas para mover, dependendo da fase do turno do jogador. São pedidas as coordenadas para escolher as unidades ou para mover uma dependendo das listas fornecidas.

# 

# Conclusões

Este projecto realizado no âmbito da Unidade Curricular de Programação em Lógica foi concluido com sucesso, visto que todos os critérios para a criação do jogo ***Nodes*** foram desnvolvidos e implementados e o objectivo final de aprofundar e utilizar a linguagem Prolog foi também conseguido.

Entre as principais dificuldades encontrou-se entender a lógica da linguagem Prolog, á qual não estávamos acustomados, e a criação da inteligência artificial que devido à grqande quantidade de decisões que tem de fazer, não consegue fazê-las de uma maneira minimamente óptima para oferecer um desafio ao jogador.