

*Iris*

*Relatório Final*

Programação em Lógica - 3º ANO

Grupo: Iris\_2

João Carlos Sousa Barros 201105492

Luís Miguel Guimas Marques [201104354](mailto:up201104354@fe.up.pt)

Índice

[1. Introdução 3](#_Toc24918001)

[2. O jogo Iris 4](#_Toc24918002)

[2.1 História 4](#_Toc24918003)

[2.2 Regras 4](#_Toc24918004)

[2.2.1 Início 4](#_Toc24918005)

[2.2.2 Primeira jogada 4](#_Toc24918006)

[2.2.3 Após a primeira jogada 5](#_Toc24918007)

[2.2.3.1 Colocação das peças 5](#_Toc24918008)

[2.2.4 Tabuleiro cheio 6](#_Toc24918009)

[2.3 Pontuação 6](#_Toc24918010)

[2.3.1 Empates 7](#_Toc24918011)

[2.4 Curiosidades 8](#_Toc24918012)

[3 Lógica do Jogo 9](#_Toc24918013)

[3.1 Representação do Estado do Jogo 9](#_Toc24918014)

[3.2 Visualização do Tabuleiro 12](#_Toc24918015)

[3.3 Lista de Jogadas Válidas 13](#_Toc24918016)

[3.4 Execução de jogadas 14](#_Toc24918017)

[3.5 Final do Jogo 17](#_Toc24918018)

[3.6 Avaliação do tabuleiro 18](#_Toc24918019)

[3.7 Jogada do Computador 18](#_Toc24918020)

[4. Conclusões 19](#_Toc24918021)

[5. Bibliografia 20](#_Toc24918022)

# 1. Introdução

Para este trabalho foi-nos proposto desenvolver um jogo de tabuleiro utilizando como linguagem de programação Prolog.

O jogo que escolhemos foi o Iris, um jogo desenvolvido em 2019 por Craig Duncan. Este jogo foi desenhado para ser jogado por dois jogadores. Assim, desenvolvemos o nosso jogo de forma a ser possível jogá-lo de três formas distintas: Humano vs Humano, Humano vs Computador e Computador vs Computador.

Para o desenvolvimento do Iris, aplicamos conhecimentos adquiridos tanto nas aulas teóricas como nas práticas da cadeira de Programação em Lógica. Durante a execução do trabalho, fomo-nos apercebendo do quanto esta linguagem é útil para desenvolvimento de todos os programas onde a lógica pode ser aplicada.

Inicialmente tivemos alguma dificuldade na sua execução, visto ser uma linguagem com a qual estamos a ter contacto pela primeira vez, mas com o avançar do tempo e da prática fomos ficando cada vez mais familiarizados com a sua aplicação.

Como produto final, conseguimos um jogo que acaba por ser bastante divertido, mas acima de tudo, mais estratégico do que aparenta à primeira vista, onde durante a sua jogabilidade o utilizador acaba por tomar várias decisões que podem alterar o desfecho do jogo, tentando sempre pensar à frente do seu adversário.

# 2. O jogo Iris

## **2.1 História**

Tal como dito anteriormente, o jogo Íris foi criado por Craig Duncan em 2019.

É um jogo de estratégia abstrato jogado somente por duas pessoas num tabuleiro hexagonal com células hexagonais.

No perímetro do tabuleiro as células são coloridas e o interior são neutras/cinzentas.

O objetivo do jogo é formar um grupo de peças adjacentes entre si desde uma célula colorida a outra.

Sendo um jogo recente e com pouca popularidade não há muita informação disponível.

## **2.2 Regras**

### **2.2.1 Início**

O jogador 1 joga com peças pretas e o jogador 2 joga com peças brancas.

Inicialmente o tabuleiro está vazio.

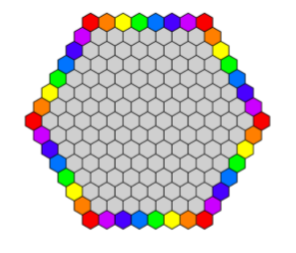


Figura 1: Tabuleiro

### **2.2.2 Primeira jogada**

No primeiro turno, o jogador que inicia só lhe é permitido colocar uma peça numa única célula neutra. Nos seguintes turnos, a começar no segundo turno pelo outro jogador, cada jogada é feita com duas peças.

### **2.2.3 Após a primeira jogada**

### **2.2.3.1 Colocação das peças**

A peça pode ser colocada numa célula colorida ou neutra.

Se a primeira peça for colocada numa célula colorida a segunda peça obrigatoriamente terá de ser colocada na célula oposta colorida, como ilustrado na figura 2.

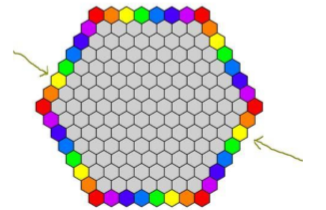


Figura 2

Se a primeira peça for jogada numa célula neutra, a segunda peça obrigatoriamente terá de ser jogada numa célula neutra não adjacente à primeira. Como é mostrado na figura 3, a preto é a primeira peça e os pontos vermelhos são células proibidas para a segunda peça.

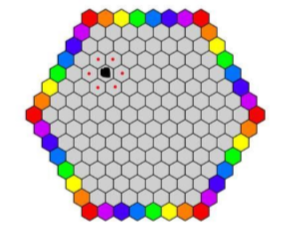


Figura 3

### **2.2.4 Tabuleiro cheio**

O jogo termina quando o tabuleiro fica cheio (não há mais hipóteses de jogada) ou quando ambos os jogadores passam. Com jogadores mais experimentes o tabuleiro não chega a ficar cheio uma vez que o jogador perdedor se apercebe que já não tem hipótese de ganhar e terminam ambos passando.

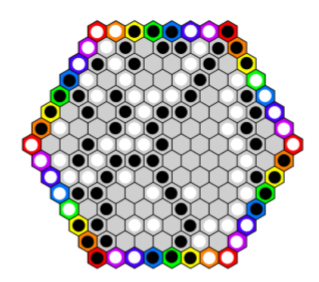


Figura 4: Tabuleiro cheio

## **2.3 Pontuação**

Terminando o jogo a contagem dos pontos é feita agrupando as peças adjacentes sendo que apenas as peças coloridas são contabilizadas.

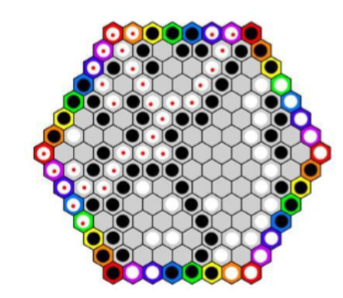


Figura 5: Exemplo de um grupo (pontos vermelhos)

A contagem de pontos é feita considerando a linha contínua que agrupa um maior número de peças coloridas. Caso ambos os jogadores tenham a maior linha com o mesmo número de peças coloridas verifica-se a linha seguinte até um jogador ter uma linha com mais peças coloridas que o adversário.

Na imagem em cima o jogador branco consegui fazer uma linha contínua agrupando 11 peças coloridas, como representado com os pontos vermelhos.

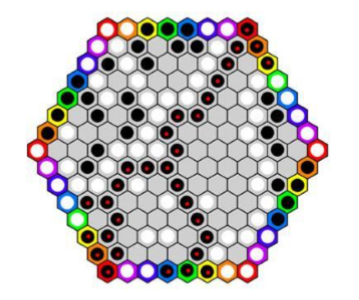


Figura 6: Grupo de peças do jogador preto

Neste caso o jogador preto conseguiu na sua maior linha agrupar 9 peças coloridas.

Neste exemplo o jogador branco ganha o jogo.

### **2.3.1 Empates**

Caso ambos os jogadores tenham o mesmo número de pontos na sua maior linha, o desempate é feito pela segunda maior linha e assim sucessivamente.

Suponhamos que o jogador branco tem grupos com a seguinte pontuação: [9, 7, 3, 3] e o jogador preto: [9, 7, 4]. Neste exemplo apesar de o jogador branco ter mais peças coloridas, como os dois primeiros grupos têm a mesma pontuação e o jogador preto tem mais peças no terceiro grupo, o jogador preto é o vencedor. ~ 1.4 Curiosidades Este jogo nunca termina em empate.

Como o tabuleiro é um hexágono o jogo vai sempre terminar com 7 grupos diferentes, o que implica que é impossível ambos os jogadores tenham o mesmo número de grupos. Este teorema é mais aprofundado pelo físico e criador de jogos Craige Schensted mais conhecido por Ea Ea.

Esta informação foi fornecida pela o próprio criador/designer do jogo, Craig Duncan. O qual fica aqui o nosso agradecimento especial.

## **2.4 Curiosidades**

Este jogo nunca termina em empate.

Como o tabuleiro é um hexágono o jogo vai sempre terminar com 7 grupos diferentes, o que implica que é impossível ambos os jogadores tenham o mesmo número de grupos. Este teorema é mais aprofundado pelo físico e criador de jogos Craige Schensted mais conhecido por Ea Ea.

Esta informação foi fornecida pela o próprio criador/designer do jogo, Craig Duncan. Ao qual fica aqui o nosso agradecimento especial.

# 3 Lógica do Jogo

## **3.1 Representação do Estado do Jogo**

emptyBoard([[null, null, null, red, orange, yellow, green, blue, red, null, null],

[null, null, blue, empty, empty, empty, empty, empty, orange, null, null],

[null, null, green, empty, empty, empty, empty, empty, empty, yellow, null],

[null, yellow, empty, empty, empty, empty, empty, empty, empty, green, null],

[null, orange, empty, empty, empty, empty, empty, empty, empty, empty, blue],

[red, empty, empty, empty, empty, empty, empty, empty, empty, empty, red],

[blue, empty, empty, empty, empty, empty, empty, empty, empty, orange, null],

[null, green, empty, empty, empty, empty, empty, empty, empty, yellow, null],

[null, yellow, empty, empty, empty, empty, empty, empty, green, null, null],

[null, null, orange, empty, empty, empty, empty, empty, blue, null, null],

[null, null, red, blue, green, yellow, orange, red, null, null, null]]).

midBoard([[null, null, null, red, orange, white, green, pink, red, null, null],

[null, null, pink, empty, empty, white, empty, empty, orange, null, null],

[null, null, black, empty, empty, empty, white, empty, empty, yellow, null],

[null, yellow, black, empty, empty, empty, white, empty, empty, green, null],

[null, orange, empty, black, empty, empty, empty, empty, empty, empty, pink],

[red, empty, empty, black, black, black, empty, white, empty, empty, red],

[pink, empty, empty, empty, empty, empty, white, empty, empty, orange, null],

[null, green, empty, empty, empty, empty, white, empty, empty, yellow, null],

[null, yellow, empty, empty, white, black, empty, black, black, null, null],

[null, null, orange, empty, empty, white, empty, empty, pink, null, null],

[null, null, red, pink, green, white, orange, red, null, null, null]]).

fullBoard([[null, null, null, black, black, white, black, white, white, null, null],

[null, null, white, empty, empty, white, empty, empty, black, null, null],

[null, null, black, empty, empty, empty, white, empty, empty, black, null],

[null, white, black, empty, empty, empty, white, empty, black, white, null],

[null, white, white, black, empty, empty, empty, black, empty, empty, black],

[ black, empty, white, black, black, black, black, white, black, empty, black],

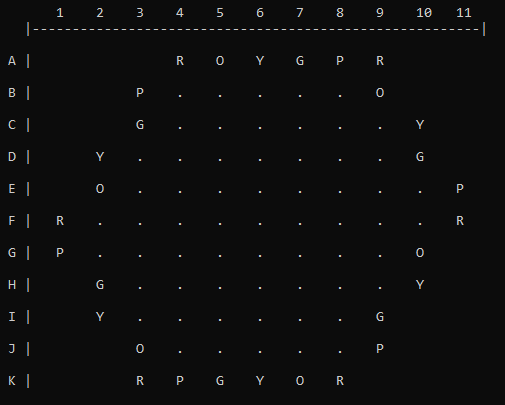
[ black, empty, white, black, black, empty, white, empty, empty, white, null],

[null, white, white, empty, white, empty, white, empty, empty, white, null],

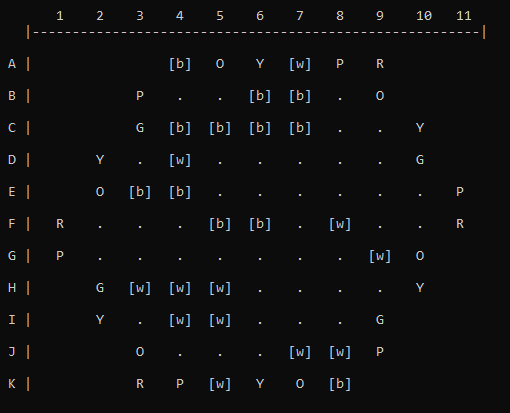
[null, black, black, white, white, black, white, black, black, null, null],

[null, null, black, empty, white, white, black, empty, white, null, null],

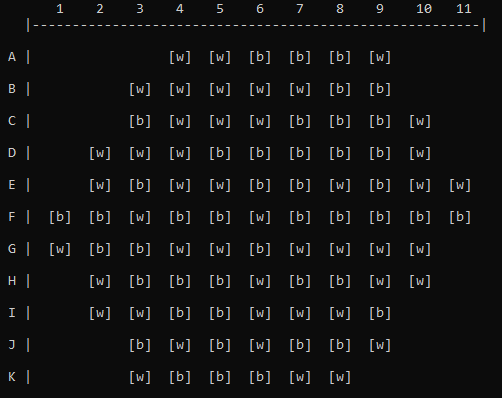
[null, null, white, white, black, white, black, black, null, null, null]]).



**Tabuleiro Vazio**

****

**Tabuleiro a meio do jogo**

****

**Tabuleiro cheio**

## **3.2 Visualização do Tabuleiro**

Abaixo temos o código utilizado para visualizar as imagens demonstradas na página anterior.

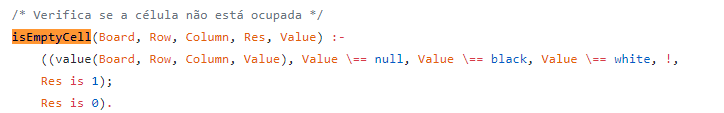


## **3.3 Lista de Jogadas Válidas**

No nosso jogo as jogadas são consideradas válidas quando são jogadas feitas uma casa onde não exista nenhuma peça de um jogador branco, nem nenhuma peça do jogador preto e seja uma casa dentro do tabuleiro. Para isso utilizamos o predicado checkMove que utilizamos do seguinte modo:



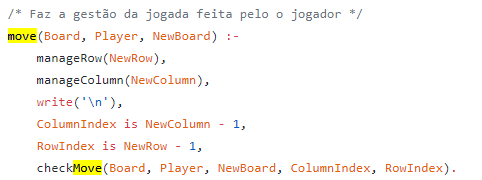
O predicado empty cell é o seguinte:



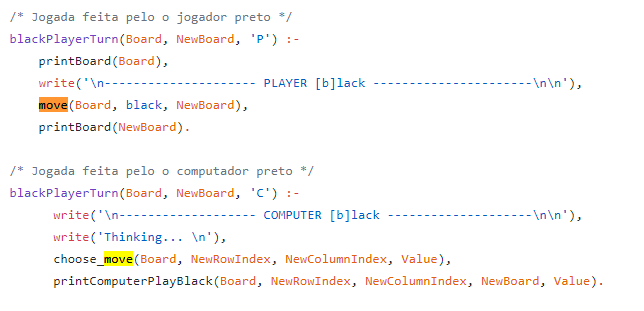
Ou seja, retorna 1 quando é uma célula não ocupada e válida ou 0 quando é uma célula já ocupada.

## **3.4 Execução de jogadas**

A execução de jogadas é feita através no nosso predicado move que funciona do seguinte modo:



O predicado move pode tanto ser chamado pelo blackPlayerTurn como pelo whitePlayerTurn, dependendo do jogador que está a jogar nesse turno. Cada um destes predicados é diferente, conforme esse jogador esteja a ser comandado por um humano ou pelo computador. Temos aqui um exemplo para o jogador preto, sendo que o do jogador branco é equivalente:

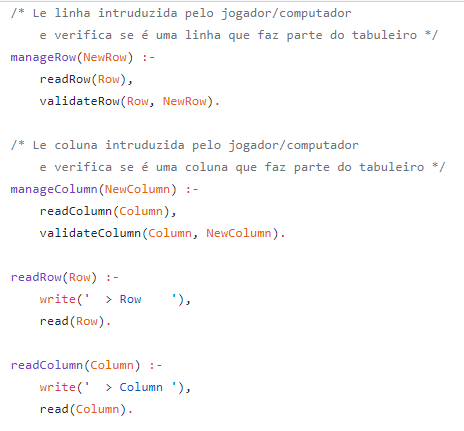


O blackPlayerTurn é chamado logo que é executado o predicado gameLoop, que por sua vez é executado logo depois de haver a primeira jogada do jogo, que é sempre executada pelo jogador branco.

O predicado gameLoop faz toda a gestão do jogo, desde a primeira jogada até que é detetado o fim do jogo (predicado game\_over) e foi implementado assim:

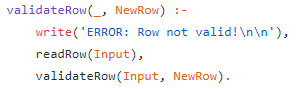


Os predicados manageRow e mandaColumn utilizam respetivamente os predicados readRow e readColumn, respetivamente, que vão ler os valores que o utilizador inserir no nosso jogo. O código utilizado foi o seguinte:

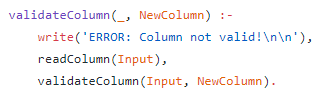


O predicado validateColumn e validateRow serve apenas para garantir que o utilizador introduz um input válido e retorna uma mensagem de erro caso seja inserido um caracter diferente daqueles que podem ser aceites. Podem ser aceites linhas de “a” até “k” e colunas de 1 até 11. Temos aqui um exemplo para uma linha e coluna aceites e a forma como imprimimos a mensagem de erro:





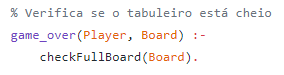


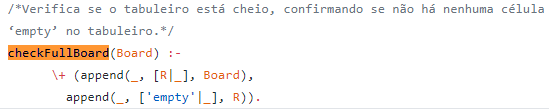


## **3.5 Final do Jogo**

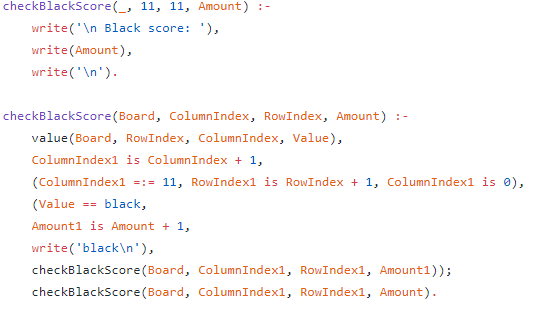
O jogo chega ao fim quando ambos os jogadores passam a sua vez ou quando o tabuleiro fica cheio. O código implementado para confirmar estas situações foi o predicado game\_over que verifica se o tabuleiro está cheio com o predicado checkFullBoard.

Estes dois predicados foram implementados do seguinte modo:





Assim, se o tabuleiro estiver cheio, o predicado game\_over assegura-se que o jogo chega ao fim e são determinadas as pontuações dos jogadores. As pontuações são contabilizadas com os predicados checkBlackScore e checkWhiteScore, que são idênticos. Aqui temos o exemplo do checkBlackScore:

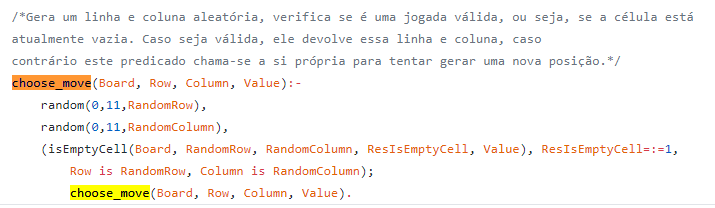


## **3.6 Avaliação do tabuleiro**

A avaliação do tabuleiro, tal como referido anteriormente é feita através do predicado gameLoop. É esse predicado que avalia se o jogo chegou ao fim, ou se é necessário gerar uma nova jogada.

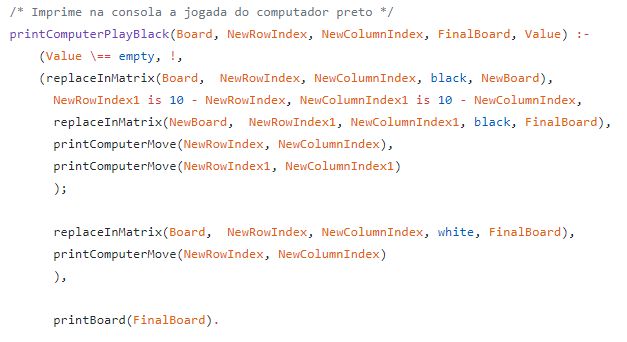
## **3.7 Jogada do Computador**

Para gerar jogadas feitas pelo computador, utilizamos os predicados choose\_move, implementado do seguinte modo:



Ou seja, são gerados números aleatórios entre 0 e 11 tanto para as linhas como para as colunas, e confirmado que esses valores se referem a uma célula vazia através do método isEmptyCell já descrito anteriormente. Caso a célula não seja vazia, choose\_move é chamado novamente de forma recursiva, até ser gerada uma célula vazia. Isto significa que o computador não faz qualquer tipo de planeamento das suas jogadas, o que corresponde ao nível mais básico de dificuldade. Infelizmente, não conseguimos implementar os vários níveis de dificuldade para o computador, pedido para este trabalho.

A impressão para a consola é feita através do método printComputerPlayBlack para o caso de o computador estar a jogar do lado preto ou printComputerPlayWhite para o caso de estar do lado branco. Ambos os métodos são equivalentes e abaixo temos o exemplo do computador a jogar do lado preto:



# 4. Conclusões

Com este trabalho aprendemos como funciona a implementação de um jogo aplicando conhecimentos adquiridos nas aulas de Programação em Lógica e de como a lógica pode ser tão útil para resolver problemas deste tipo, em que é necessário haver uma sequência de métodos para determinar o estado atual do jogo.

Durante a sua execução deparamo-nos com algumas dificuldades. No nosso caso, a maior dificuldade foi pensar como poderíamos estruturar o jogo de forma a funcionar de forma recursiva e de como poderíamos fazer o cálculo das pontuações dos jogadores, sendo que esta última não foi ultrapassada e não a conseguimos implementar no nosso trabalho.

Em geral, consideramos que o trabalho foi concluído com sucesso, mas poderíamos ter feito melhor, nomeadamente na aplicação de vários níveis de dificuldade/inteligência do computador e no sucesso na contabilização das pontuações dos jogadores.

# 5. Bibliografia

Para desenvolver este trabalho utilizamos a informação que tínhamos disponível sobre este jogo no link:

* <https://boardgamegeek.com/boardgame/286792/iris>

Através desse site e com alguma pesquisa conseguimos entrar em contacto com o criador do jogo, Craig Duncan, o que nos deu um esclarecimento adicional sobre como o jogo funcionava, sem o qual a nossa tarefa seria muito mais difícil. Desde já, deixamos aqui o nosso agradecimento.

* <https://faculty.ithaca.edu/cduncan/>

Para além disso utilizamos o manual do SICStus Prolog disponível no link:

* <https://sicstus.sics.se/sicstus/docs/latest4/pdf/sicstus.pdf>