**Faculdade de Engenharia da Universidade do Porto**

**Mestrado Integrado em Engenharia Informática e Computação**


# Relatório Intercalar

# "Froglet"

**Programação em Lógica (PLOG) – 2017/2018 – 1ºSemestre**

Turma 1, grupo 3

**Trabalho realizado por:**

| | | |
|---|---|---|
| Diogo Luís Rey Torres | 201506428 | up201506428@fe.up.pt |
| Francisco Teixeira Lopes | 201106912 | ei11056@fe.up.pt |

# 1. O Jogo

## *História*

Froglet é uma variante de um jogo de tabuleiro preservado pelo historiador de jogos de tabuleiro, Harold Murray. No seu livro, publicado em 1898, Murray descreve o jogo original e uma variante criada por si. A variante de Murray foi adaptada pela comunidade BrainKing para servir de versão online do jogo, esta é a variante denominada Froglet e a qual se vai implementar, as diferenças residem apenas no tamanho do tabuleiro e na distribuição de peças coloridas no tabuleiro.

## *Regras*

O jogo consiste num tabuleiro 12x12, que é preenchido aleatoriamente com peças em forma de sapo. Estas peças podem ser de 4 cores e a distribuição final tem de ser: 66 verdes, 51 amarelos, 21 vermelhos e 6 azuis. O objetivo é capturar o número máximo de pontos e cada cor tem um valor diferente, verde vale 1, amarelo vale 2, vermelho vale 3 e azul vale 4.

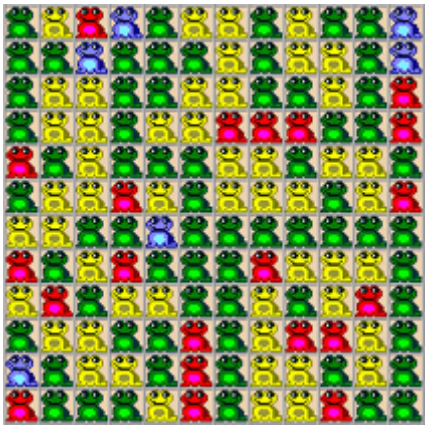

*Figura 1 - Exemplo disposição inicial*

O jogo começa com a remoção de qualquer sapo verde, após essa remoção, o jogo prossegue através de saltos. Para um salto ser válido, um jogador deve selecionar um sapo que tenha outro sapo diretamente adjacente, horizontalmente ou verticalmente, e que tenha um espaço vazio na direção do salto. Saltos múltiplos são permitidos mas não são obrigatórios, o jogador pode parar a qualquer salto sem ter de realizar os que se seguem. Porém, um jogador tem de efetuar no mínimo um salto.

*Figura 2 - Exemplo salto, a vermelho sapo selecionado*

*Figura 3 - Depois do salto*

O jogo termina quando não existirem mais saltos possíveis e o vencedor é quem tiver mais pontos.

Fontes:

https://en.wikipedia.org/wiki/Leap_Frog_(board_game)

https://brainking.com/en/GameRules?tp=54

# 2. Representação do estado do jogo

O tabuleiro de jogo é representado através de uma lista de listas, para representar o tabuleiro utilizam-se os símbolos GYRB, G – Green, Y – Yellow, R – Red e B – Blue.

Exemplo de estado inicial, intermédio e final em Prolog:

```
initial([
        ['Y','Y','Y','G','G','G','G','G','G','G','Y','Y'],
        ['G','R','R','R','R','Y','Y','G','G','Y','Y','Y'],
        ['G','G','Y','R','G','G','G','Y','G','G','Y','B'],
        ['R','G','Y','R','R','G','G','G','G','G','Y','Y'],
        ['G','G','Y','Y','R','G','G','G','G','G','Y','Y'],
        ['R','Y','Y','G','G','R','R','G','G','Y','Y','Y'],
        ['G','G','Y','Y','G','Y','G','G','Y','G','Y','Y'],
        ['G','Y','Y','B','G','Y','Y','G','G','Y','Y','Y'],
        ['R','Y','Y','G','G','G','R','B','Y','G','Y','Y'],
        ['R','B','Y','R','G','G','Y','R','B','Y','G','B'],
        ['G','G','G','R','G','R','G','R','Y','G','G','Y'],
        ['G','Y','G','G','G','R','G','Y','G','G','G','Y']]).


ongoing([
        ['Y','Y','Y','G','G','G','G','G','G','G','Y','Y'],
        ['G','R','R','R','R','Y','Y','G','G','Y','Y','Y'],
        ['G','G','Y',' ','G','G','G',' ',' ',' ',' ',' ',' '],
        ['R','G','Y',' ',' ',' ','G',' ',' ',' ','G',' ',' ','Y','Y'],
        ['G','G',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','Y','Y'],
        ['R','Y','Y',' ',' ','G',' ',' ',' ','G',' ',' ',' ','Y','Y'],
        ['G','G','Y','Y','G',' ',' ','G','G',' ',' ','G','Y','Y'],
        ['G','Y','Y','B','G','Y','Y','G','G','Y','Y','Y'],
        ['R','Y','Y','G','G','G','R','B','Y','G','Y','Y'],
        ['R','B','Y','R','G','G','Y','R','B','Y','G','B'],
        ['G','G','G','R','G','R','G','R','Y','G','G','Y'],
        ['G','Y','G','G','G','R','G','Y','G','G','G','Y']]).


final([
        ['Y',' ',' ',' ',' ',' ','G',' ',' ',' ',' ',' ','G',' '],
        [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
        [' ',' ',' ',' ',' ',' ','G',' ',' ',' ',' ',' ',' ',' '],
        [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
        [' ',' ',' ','R',' ',' ',' ',' ',' ',' ',' ','Y',' ','Y'],
        ['G',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
        [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','Y'],
        ['G',' ',' ',' ','G',' ',' ',' ',' ',' ',' ',' ',' ',' '],
        [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '],
        [' ','Y',' ',' ',' ','G',' ','Y',' ',' ',' ','B',' ','Y'],
        [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','G',' ',' ',' '],
        [' ',' ',' ',' ',' ',' ',' ',' ','B',' ','G',' ',' ',' ']]).
```

# 3. Visualização do tabuleiro em modo de texto

A representação interna do tabuleiro coincide com a representação em modo de texto pretendida. Porém, a representação em modo de texto é planeada de forma a ser mais agradável de visualizar que o formato denso de lista de listas.

O predicado de visualização é displayBoard(Board) e o output é o seguinte:

```
| ?- initial(X), displayBoard(X).
   A     B     C     D     E     F     G     H     I     J     K     L
|     |     |     |     |     |     |     |     |     |     |     |     |
-------------------------------------------------------------------------
|  Y  |  Y  |  Y  |  G  |  G  |  G  |  G  |  G  |  G  |  G  |  Y  |  Y  | 1
-------------------------------------------------------------------------
|  G  |  R  |  R  |  R  |  R  |  Y  |  Y  |  G  |  G  |  Y  |  Y  |  Y  | 2
-------------------------------------------------------------------------
|  G  |  G  |  Y  |  R  |  G  |  G  |  G  |  Y  |  G  |  G  |  Y  |  B  | 3
-------------------------------------------------------------------------
|  R  |  G  |  Y  |  R  |  R  |  G  |  G  |  G  |  G  |  G  |  Y  |  Y  | 4
-------------------------------------------------------------------------
|  G  |  G  |  Y  |  Y  |  R  |  G  |  G  |  G  |  G  |  G  |  Y  |  Y  | 5
-------------------------------------------------------------------------
|  R  |  Y  |  Y  |  G  |  G  |  R  |  R  |  G  |  G  |  Y  |  Y  |  Y  | 6
-------------------------------------------------------------------------
|  G  |  G  |  Y  |  Y  |  G  |  Y  |  G  |  G  |  Y  |  G  |  Y  |  Y  | 7
-------------------------------------------------------------------------
|  G  |  Y  |  Y  |  B  |  G  |  Y  |  Y  |  G  |  G  |  Y  |  Y  |  Y  | 8
-------------------------------------------------------------------------
|  R  |  Y  |  Y  |  G  |  G  |  G  |  R  |  B  |  Y  |  G  |  Y  |  Y  | 9
-------------------------------------------------------------------------
|  R  |  B  |  Y  |  R  |  G  |  G  |  Y  |  R  |  B  |  Y  |  G  |  B  | 10
-------------------------------------------------------------------------
|  G  |  G  |  G  |  R  |  G  |  R  |  G  |  R  |  Y  |  G  |  G  |  Y  | 11
-------------------------------------------------------------------------
|  G  |  Y  |  G  |  G  |  G  |  R  |  G  |  Y  |  G  |  G  |  G  |  Y  | 12
-------------------------------------------------------------------------
```

*Figura 4 - Output atual e pretendido*

# 4. Movimento

O jogo consiste apenas de saltos de uma coordenada para outra, por isso, precisa de poucos predicados só para movimento.

- chooseFrog(+Coords, +Board, -Success?) – permite ao jogador escolher uma coordenada no tabuleiro do sapo que quer usar para saltar. Success é output e indica se a o sapo é válido ou não.
- jump(+Coords, +Board, -NewBoard) – após escolher um sapo com sucesso, o jogador escolhe para onde saltar. NewBoard é output do tabuleiro modificado pela jogada.
- jumpAgain(-Decision) – pergunta ao jogador se pretende terminar a jogada ou fazer outro salto caso seja possível. Decision é output true / false.