**Faculdade de Engenharia da Universidade do Porto**

**Mestrado Integrado em Engenharia Informática e Computação**

**3º ano, 1º semestre**



Projeto 1 de Programação em Lógica:

Adaptoid

Relatório Intercalar

**16 de outubro de 2016**

**Grupo Adaptoid\_1:**

|  |  |
| --- | --- |
| Andreia Cristina de Almeida Rodrigues | up201404691@fe.up.pt |
| Gonçalo da Mota Laranjeira Torres Leão | up201406036@fe.up.pt |

**Índice**

[**O Jogo Adaptoid**](#_xx07k0wmuh35)

[**Informações gerais**](#_4aqrwdxw7cer)

[**Ações a realizar em cada turno**](#_juwl39ql977z)

[**Movimento e captura**](#_6a1u79807xy9)

[**Criação de um novo adaptoid**](#_91st0u6gl8v7)

[**Mutações dos adaptoids**](#_1fv1ia7thocq)

[**Alimentação dos adaptoids**](#_bple9hasdwjx)

[**Condições para terminar a partida**](#_rda4dhymig48)

[**Abordagem inicial à modelação do jogo em Prolog**](#_ure34j4g7qz7)

[**Representação do estado do jogo**](#_4xxcj1u8ul9)

[**Visualização do tabuleiro de jogo em modo de texto**](#_5nxphosqnmim)

[**Movimentos**](#_qwyyfbgba2ec)

[**Bibliografia**](#_fo921v6ldnbq)

# 

# **O Jogo Adaptoid**

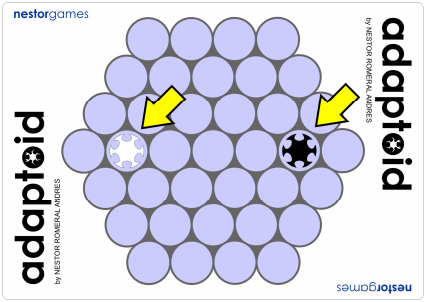
## Informações gerais

O jogo Adaptoid foi criado entre 2008 e 2009 por Néstor Romeral Andrés, um criador de jogos de tabuleiro espanhol [(«Néstor Romeral Andrés» 2016)](https://paperpile.com/c/lZgRxP/2tJ5). Trata-se de um jogo de tabuleiro para dois jogadores em que não existe a influência de nenhum fator sorte [(Andrés 2008; Sandler 2016; Valentine 2016)](https://paperpile.com/c/lZgRxP/8bhM+G8fs+bHVn).

O jogo é baseado em criaturas chamadas *adaptoids* que sofrem mutações sucessivas para se adaptarem melhor ao seu ambiente.

Cada jogador controla *adaptoids* de uma dada cor. Um jogador joga com peças brancas, o outro, com peças pretas. O jogo joga-se por turnos e ocorre num tabuleiro hexagonal de 37 casas (conferir a figura 1 abaixo para o esquema do tabuleiro).

No início da partida, cada jogador posiciona um *adaptoid* (sem pinças, nem pernas) no tabuleiro segundo a configuração apresentada na figura 1 abaixo. O jogador com as peças brancas é quem joga o primeiro turno.



*Figura 1: Configuração inicial do jogo  
(Fonte: Manual de instruções disponível em nestorgames.com)*

## Ações a realizar em cada turno

Em cada turno, o jogador ativo deve efetuar as seguintes ações, respeitando a ordem em que se apresentam:

* Se quiser e se for possível, pode mover um dos seus *adaptoids*. Este movimento pode gerar a captura de um *adaptoid*.
* Se quiser e se for possível, pode posicionar uma nova peça da sua cor no tabuleiro ou acrescentar a um dos seus *adaptoids* (pode ser ou não o que foi movido neste turno) uma pinça ou uma perna.

Ao terminar o turno, todos os *adaptoids* do adversário que não estão alimentados são capturados.

## Movimento e captura

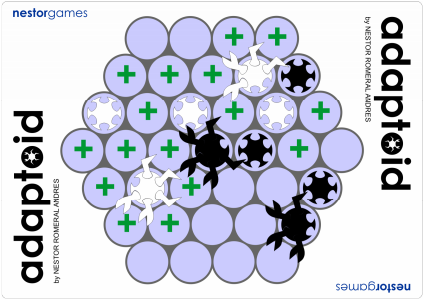
Em cada turno, um *adaptoid* pode-se mover para uma casa livre adjacente até tantas vezes quanto o número de pernas que possui. Assim, uma criatura sem pernas não pode efetuar nenhum movimento. A meio do movimento, não pode passar por casas ocupadas, mas pode terminar o movimento numa casa ocupada por um *adaptoid* inimigo.

Quando dois *adaptoids* inimigos se encontram na mesma casa, é capturado aquele que tiver o menor número de pinças. Em caso de empate, ambos as peças são capturadas.

Sempre que um *adaptoid* é capturado, o jogador adversário ganha um ponto.

## Criação de um novo *adaptoid*

Um novo *adaptoid* apenas pode ser posicionado no tabuleiro numa casa adjacente a outro *adaptoid* da mesma cor. Os novos *adaptoids* começam sempre sem pinças nem pernas.



*Figura 2: Posições (marcadas com uma cruz verde) onde é possível adicionar um “adaptoid” branco*

Cada jogador dispõe de apenas 12 peças do tipo *adaptoid*, pelo que só pode ter até esse número de criaturas no tabuleiro a qualquer momento. Quando um *adaptoid* é capturado, a peça correspondente pode ser novamente usada para criar um novo *adaptoid* mais à frente no jogo.

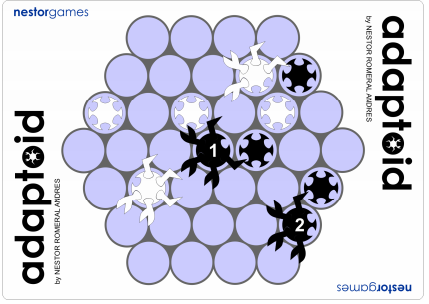
## Mutações dos *adaptoids*

Cada *adaptoid* pode ter até no máximo seis membros. Estes membros podem tanto ser pinças como pernas, e não há nenhuma imposição sobre a proporção entre estes dois tipos de membros. Assim, existem 28 estados em que um *adaptoid* se pode encontrar.

Tal para as peças do tipo *adaptoid*, cada jogador dispõe de apenas 12 pinças e 12 peças para poder usar nas suas peças. Quando um *adaptoid* é capturado, qualquer pinça ou perna que tinha volta para a posse do jogador correspondente, e poderá de novo ser usada.

## Alimentação dos *adaptoids*

Para se manterem vivos, os *adaptoids* têm de ser constantemente alimentados. Um *adaptoid* está alimentado se o número de casas livres adjacentes à sua posição é maior ou igual ao seu número de membros. Assim, quanto mais membros tiver uma criatura, mais são os seus requisitos para estar alimentado.



*Figura 3: Uma configuração possível do jogo para ilustrar o conceito de um “adaptoid” estar alimentado*

Considere-se a figura acima e que vai terminar o turno do jogador das peças brancas. O *adaptoid* preto marcado com um “1” tem 5 membros (3 pinças e 2 pernas) e tem apenas 4 casas livres à sua volta, pelo que é capturado. O mesmo ocorre com a criatura marcada com um “2” que tem 4 membros (3 pinças e 1 perna) mas apenas 3 casas livres à sua volta. As três restantes peças pretas nunca poderiam ser capturadas, independentemente do estado das suas casas vizinhas, pois não têm membros. Assim, no final do turno do jogador das peças brancas, este ganha 2 pontos, pois efetuou 2 capturas.

## Condições para terminar a partida

O objetivo do jogo é ganhar 5 pontos, o que é possível capturando 5 *adaptoids* do jogador adversário. O jogador que atingir este objetivo primeiro vence a partida. Além disso, um jogador perde, se, a um dado momento da partida, não tem nenhum *adaptoid* da sua cor no tabuleiro. Em caso de empate (se um *adaptoid* se mover para uma casa com uma peça inimiga com o mesmo número de pinças, e isto permitir a ambos jogadores atingir os 5 pontos), vence quem jogou mais recentemente.

# **Abordagem inicial à modelação do jogo em Prolog**

Nota: na escolha de nomes de predicados e de variáveis, e em comentários, para ser mais conciso, referimo-nos frequentemente a *adaptoids* como sendo *toids*.

## Representação do estado do jogo

Para representar o jogo, recorremos ao predicado game/5 que tem como campos, em ordem, o número do turno, o jogador ativo (wht - jogador branco, blk - jogador preto), os dados do jogador branco, os dados do jogador preto e o tabuleiro.

game(+turnNo,+activePlayer,+whtPlayerInfo,+blkPlayerInfo,+board).

Para representar um jogador, recorremos ao predicado player/4 que tem como campos, em ordem, a pontuação do jogador, e o seu número de peças de *toids*, pinças e pernas.

player(+points,+toids,+pincers,+legs).

Para o tabuleiro, usamos uma lista de listas em que cada elemento de uma sub-lista representa uma célula, que pode estar vazia (none) ou conter um *adaptoid* (cujos dados estão contidos num predicado *toid*). Quanto à indexação de elementos do tabuleiro, a primeira coordenada corresponde ao índice da linha (cuja numeração aumenta de cima para baixo), e a segunda corresponde ao índice da coluna (cuja numeração aumenta da esquerda para a direita). O elemento do canto superior esquerdo é [1,1] e não [0,0]. Dependendo da linha, dada a forma hexagonal do tabuleiro, o índice da coluna começa sempre em 1 mas o seu valor máximo varia entre 4 e 7.

Para representar um *adaptoid*, recorre-se ao predicado toid/3 que tem como campos, em ordem, a sua cor (wht ou blk), o seu número de pinças e o seu número de pernas.

toid(+color,+pincers,+legs).

Apresenta-se de seguida o código para visualização de três tabuleiros (estado inicial do jogo, estado intermédio, e final do jogo). Em cada um, o predicado display\_game recebe a estrutura game referida anteriormente. O código de display\_game será apresentado na secção seguinte.

A invocação de test(a) e test(b) deverá imprimir, respetivamente, um tabuleiro semelhante ao da figura 1 e ao da figura 2 (ou 3). A invocação de test(c) deverá imprimir um tabuleiro cujo estado foi obtido através do caso de test(b) em que se moveu a peça preta em [4,4] para [5,2], o que conduz a uma vitória do jogador preto.

% An initial game state

test(a) :-display\_game(game(1, wht, player(0, 11, 12, 12), player(0, 11, 12, 12),

[[none, none, none, none],

[none, none, none, none, none],

[none, none, none, none, none, none],

[none, toid(wht,0,0), none, none, none, toid(blk,0,0), none],

[none, none, none, none, none, none],

[none, none, none, none, none],

[none, none, none, none]])).

% A mid-game state

test(b) :-display\_game(game(20, blk, player(2, 7, 8, 7), player(4, 7, 6, 9),

[[none, none, none, none],

[none, none, none, toid(wht,2,2), toid(blk,0,0)],

[toid(wht,0,0), none, toid(wht,0,0), none, toid(wht,0,0), none],

[none, none, none, toid(blk,3,2), toid(blk,0,0), none, none],

[none, toid(wht,2,3), none, none, none, toid(blk,0,0)],

[none, none, none, none, toid(blk,3,1)],

[none, none, none, none]])).

% An end-game state

test(c) :-display\_game(game(21, wht, player(3, 8, 10, 10), player(5, 8, 9, 10),

[[none, none, none, none],

[none, none, none, toid(wht,2,2), toid(blk,0,0)],

[toid(wht,0,0), none, toid(wht,0,0), none, toid(wht,0,0), none],

[none, none, none, none, toid(blk,0,0), none, none],

[none, toid(blk,3,2), none, none, none, toid(blk,0,0)],

[none, none, none, none, none],

[none, none, none, none]])).

## Visualização do tabuleiro de jogo em modo de texto

Apresenta-se de seguida o código completo para visualização do tabuleiro (e do todo o estado do jogo) em modo de texto.

% Displays the game's state

display\_game(game(TurnNum, ActivePlayer, player(WhtPoints, WhtToids, WhtPincers, WhtLegs), player(BlkPoints, BlkToids, BlkPincers, BlkLegs), Board)):-

write('Turn No.'),write(TurnNum),nl,

display\_active\_player(ActivePlayer),nl,

write('White player: '), display\_points(WhtPoints), nl,

display\_inventory(WhtToids,WhtPincers,WhtLegs), nl,

write('Black player: '), display\_points(BlkPoints), nl,

display\_inventory(BlkToids,BlkPincers,BlkLegs), nl,

display\_board(Board).

% Print out a line to indicate the who's turn it is.

display\_active\_player(wht):-write('It\'s the white player\'s turn!').

display\_active\_player(blk):-write('It\'s the black player\'s turn!').

% Writes player's points

display\_points(P):- write(P), write(' points.').

% Writes player's inventory

display\_inventory(Toids, Pincers, Legs):- write('Inventory:'), nl,

write('[Toids: '), write(Toids), write(']'),

write('[Pincers: '), write(Pincers), write(']'),

write('[Legs: '), write(Legs), write(']').

% Displays the board

display\_board([L1|Ls]):- display\_cols\_idxs, nl, display\_lines([L1|Ls]).

% Displays the first four column indexes

display\_cols\_idxs:- write(' 1 2 3 4').

% Displays all the board's lines

display\_lines([]).

display\_lines([L1|Ls]):- length([L1|Ls], NLines), length(L1, NCols), LineNum is 8 - NLines,

write(' '), display\_topbot\_border(NCols), nl,

spaces\_before\_board(NCols,X), write(X), write(LineNum), display\_line(L1), nl,

write(' '), display\_topbot\_border(NCols), display\_col\_idx(LineNum), nl,

display\_lines(Ls).

% Displays the remaining three column indexes

display\_col\_idx(1):-write('5').

display\_col\_idx(2):-write('6').

display\_col\_idx(3):-write('7').

display\_col\_idx(LineNum):-LineNum > 3.

% Displays a single line

display\_line([]).

display\_line([C1|Cs]):-write('|'), display\_cell(C1), write('|'), display\_line(Cs).

% Displays a line of symbols for the top and bottom of each cell.

display\_topbot\_border(NCols):-spaces\_before\_board(NCols,X), write(X), display\_top\_bot\_cells(NCols).

display\_top\_bot\_cells(0).

display\_top\_bot\_cells(CellsLeft):-CellsLeft>0,write(' -- '), N is CellsLeft-1,display\_top\_bot\_cells(N).

% Determines how many ' ' to draw on each line to properly align the board.

spaces\_before\_board(4,' ').

spaces\_before\_board(5,' ').

spaces\_before\_board(6,' ').

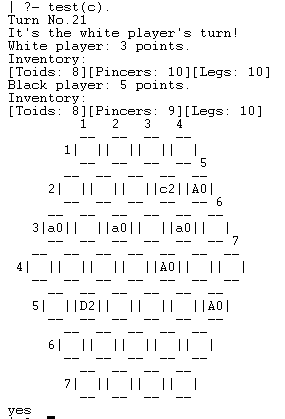
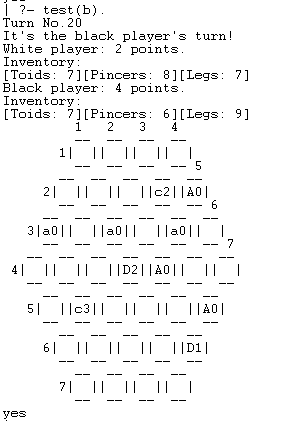
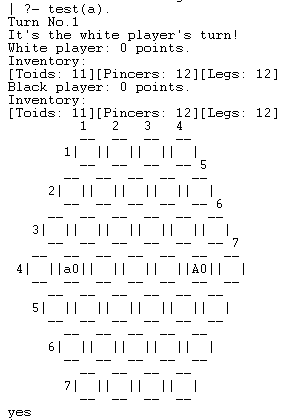
spaces\_before\_board(7,' ').

% Displays the content of a cell.

display\_cell(none):-write(' ').

display\_cell(toid(wht,P,L)):- char\_code('a',A), Code is A+P, char\_code(Letter,Code), write(Letter), write(L).

display\_cell(toid(blk,P,L)):- char\_code('A',A), Code is A+P, char\_code(Letter,Code), write(Letter), write(L).



*Figura 4: Visualização dos 3 tabuleiros, obtidos usando test(a), test(b) e test(c), respetivamente*

No nosso *interface* em modo de texto, um *adaptoid* é representado com dois caracteres. O primeiro caractere, uma letra do alfabeto, representa o número de pinças da criatura (A/a para 0 pinças, B/b para 1, C/c para 2, etc.). Uma letra minúscula é usado para peças brancas, e uma maiúscula, para peças pretas. O segundo caractere é um dígito que representa o número de pernas.

## Movimentos

Apresenta-se de seguida as cabeças dos predicados que implementam os tipos de jogadas possíveis.

%move\_toid(+game, +line, +column, +new\_line, +new\_column, -new\_game).

% Moves the toid on [line][column] to [new\_line][new\_column] if possible (checks if there's an adaptoid on [line][column] whose color matches the active player, if it has enough legs and free spaces to complete the movement).

%add\_toid(+game, +line, +column, -new\_game).

% Creates a new adaptoid which matches the active player's color and places it on on [line][column] if possible (checks for an adjacent toid of the same color, and if the player has an available toid piece).

%add\_pincer(+game, +line, column, -new\_game).

% Adds a pincer to a toid at [line][column] if possible (checks if there's a toid on [line][column] of the active player's color with less than 6 members and if the player has an available pincer piece).

%add\_leg(+game, +line, +column, -new\_game).

% Adds a leg to a toid at [line][column] if possible (checks if there's a toid on [line][column] of the active player's color with less than 6 members and if the player has an available leg piece).

# **Bibliografia**

[Andrés, Néstor Romeral. 2008. «Adaptoid». nestorgames.](http://paperpile.com/b/lZgRxP/8bhM) <http://nestorgames.com/rulebooks/ADAPTOID_EN.pdf>[.](http://paperpile.com/b/lZgRxP/8bhM)

[«Néstor Romeral Andrés». 2016. *boardgamegeek.com*. Acedido Outubro 7.](http://paperpile.com/b/lZgRxP/2tJ5) <https://boardgamegeek.com/boardgamedesigner/9393/nestor-romeral-andres>[.](http://paperpile.com/b/lZgRxP/2tJ5)

[Sandler, Arty. 2016. «Adaptoid». *www.iggamecenter.com*. Acedido Outubro 7.](http://paperpile.com/b/lZgRxP/G8fs) <http://www.iggamecenter.com/info/en/adaptoid.html>[.](http://paperpile.com/b/lZgRxP/G8fs)

[Valentine, Lee. 2016. «Adaptoid». *www.ogrecave.com*. Acedido Outubro 7.](http://paperpile.com/b/lZgRxP/bHVn) <http://www.ogrecave.com/reviews/adaptoid.shtml>[.](http://paperpile.com/b/lZgRxP/bHVn)