**Le Bloq**

# Relat´orio Final



## Mestrado Integrado em Engenharia Informa´tica e Computa¸ca˜o

Programa¸ca˜o em Lo´gica

**Grupo xx:**

### Eduardo Almeida - ei12018 Joa˜o Almeida - ei12053

Faculdade de Engenharia da Universidade do Porto Rua Roberto Frias, sn, 4200-465 Porto, Portugal

12 de Outubro de 2014

# Resumo

# Pretendia-se neste trabalho implementar um jogo de tabuleiro para dois jogadores.Foi desenvolvida a aplicação para jogar, usando Prolog como linguagem de implementação.

# O jogo permite três modos de utilização: Humano/Humano, Humano/Computador e Computador/Computador. Estão incluídos dois níveis de jogo possíveis: nível facil, representado por “1” no jogo e nível difícil, representado por “2”. O modo de utilização e seleção de nível podem ser escolhidos através de uma interface em modo de texto.

# De maneira divir as diferentes fases do trabalho e poder ter uma maior organização estrutural do projeto, o código foi dividido em diferentes ficheiros. O ficheiro “lists.pl” contém as funções auxiliares necessárias para a manipulação de listas (estrutura utilizada para representar o tabuleiro).O ficheiro “board.pl” é responsável pela criação e inicialização do tabuleiro de jogo. O principal ficheiro é o “lebloq.pl” pois é neste que se encontra toda a lógica e interface do jogo, por fim, e de forma a auxiliar a interface do jogo que é feita no ficheiro anteriormente referido, existe o “utilities.pl” que contém funções de verificação de “input”.

# Como resultado final temos uma aplicação que nos permite, através de uma interface em texto, jogar o jogo “Le Bloq”.

# Introdução

# O principal objetivo deste trabalho é a realização de um jogo de tabuleiro com uma interface em modo de texto, implementado em linguagem Prolog. A fonte de motivação para a realização deste projecto vem, essencialmente, da segunda parte deste projecto,isto é, podermos realizar uma interface gráfica para o jogo em 3D. Consideramos este trabalho como o ponto de partida, uma vez que é neste que se encontra lógica do jogo que posteriormente poderá ser jogada/manipulada através de uma interface gráfica.

# Este relatório está estruturado em 5 tópicos fulcrais: “O jogo” – responsável pela apresentação do jogo, isto é, as regras do jogo e o seu funcionamento; “Lógica do Jogo” – onde se encontra explicado a implementação da lógica do jogo (representação do estado do jogo, visualização do tabuleiro, execução de movimentos, verificação do cumprimento de regras, determinação do final do jogo e cálculo de jogadas a realizar pelo computador); “Interface com o Utilizador” –descreve o módulo de interface com o utilizador;

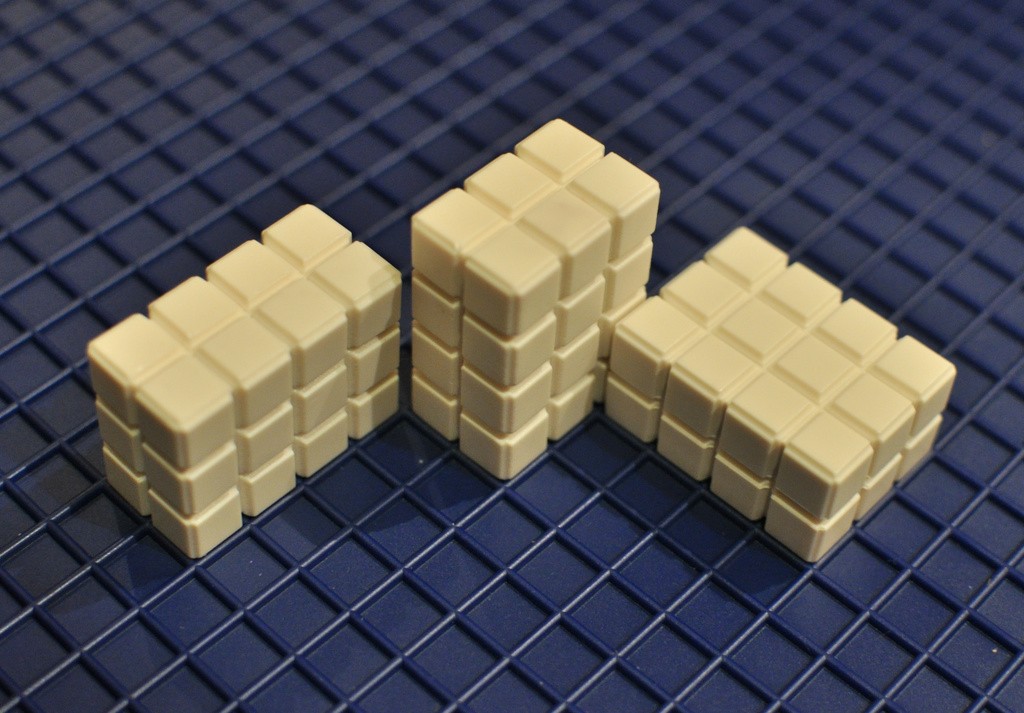
# “Conclusões” – quais as conclusões que se retirou após a realização deste projeto e o que poderá ser melhorado.

# No final do relatório existe uma secção de bibliografia onde são incluídas as referências bibliográficas que ajudaram no desenvolvimento do trabalho.

# O Jogo ”Le Bloq”

”Le Bloq”, ou ”1 Stein + Co.”, ´e um jogo de mesa abstrato para dois joga- dores (havendo adapta¸c˜oes para trˆes e quatro jogadores em simultˆaneo), jogado num campo dividido por quadrados.

Cada jogador, na sua vez, coloca uma pe¸ca de dimens˜oes 2 x 3 x 4 em campo (em qualquer alinhamento, ´a escolha do jogador).



O objetivo do jogo ´e conseguir isolar e bloquear espa¸cos em branco no campo. Apos o primeiro bloco ser jogado, todas as pe¸cas subsequentes tˆem de tocar em pelo menos uma pe¸ca colocada anteriormente, mas n˜ao podem tocar em nenhuma que tenha o mesmo alinhamento, a n˜ao ser que essa jogada permita que o jogador obtenha pontos (isolando um espa¸co em branco no campo, como

referido anteriormente).

O jogo termina quando n˜ao for poss´ıvel colocar mais nenhum bloco em campo, vencendo o jogador com maior pontua¸c˜ao.

Mais informa¸c˜oes sobre o jogo podem ser encontradas nas seguintes p´aginas da internet:

* + <http://boardgamegeek.com/boardgame/8965/1-stein-co>
  + <http://www.walmart.com/ip/Le-Bloq/11033949>
  + <http://www.barnesandnoble.com/p/le-bloq/1105892388>

# 3. Lógica do Jogo

# 3.1 Representação do Estado do Jogo

# 

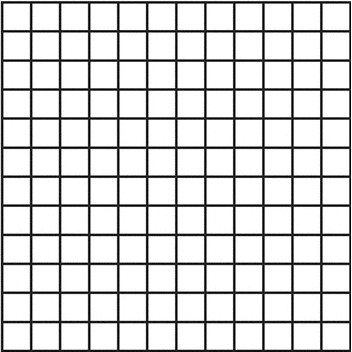
Em Prolog, o jogo ´e representado por uma lista bidimensional (uma lista de listas), em que a primeira lista-m˜ae representa as linha do campo de jogo e cada elemento da sub-lista uma coluna dessa linha.

Exemplo de um campo de jogo 3x3: [[L1C1, L1C2, L1C3], [L2C1, L2C2, L2C3], [L3C1, L3C2, L3C3]], em que ”L” significa linha e ”C” coluna.

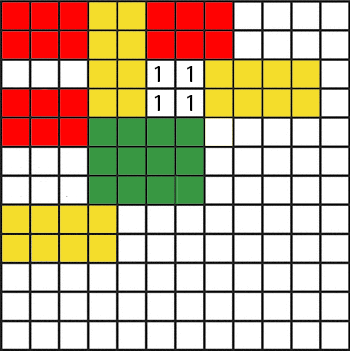
Tendo cada pe¸ca as dimens˜oes de 2x3x4, existem apenas **3** posic¸˜oes poss´ıveis de colocar uma dada peca: a ocupar 2x3 (6), 2x4 (8) ou 3x4 (12) blocos, podendo ser utilizadas estas designa¸c˜oes para guardar na lista de posic¸˜oes a maneira como uma dada quadr´ıcula foi ocupada.

Para os seguintes exemplos de posi¸c˜oes do jogo, ser˜ao utilizadas imagens representativas em que um bloco vermelho significa uma parte duma pe¸ca co- locada de modo a ocupar 6 blocos, um bloco amarelo 8 blocos e um bloco verde 12 blocos. No caso de um bloco conter um numero, esse refere-se a uma jogada que gerou pontos, representando o numero o identificador do jogador.

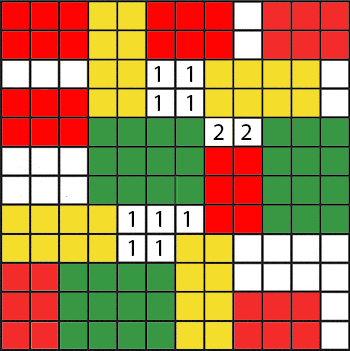
* + Estado do campo no inicio do jogo:



* + Estado do campo num tempo interm´edio:



* + Estado do campo no final do jogo:



# Visualiza¸c˜ao do Tabuleiro

* + printBoard([FirstRow | OtherRows]) – predicado responsável por imprimir no ecrã o tabuleiro do jogo. Recorre ao predicado printList.
  + printList([First|Others]) – predicado responsável por imprimir no ecrã uma lista de maneira “user-friendly”

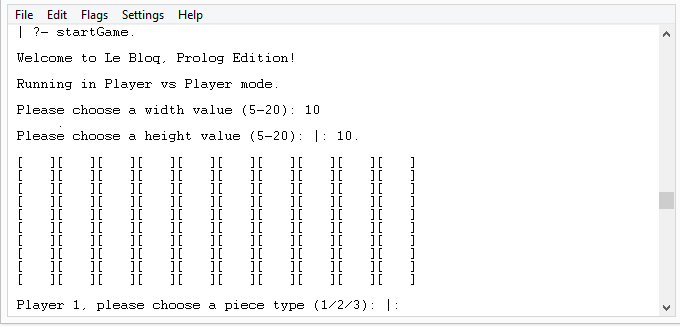


Figura 1 - Tabuleiro de jogo numa fase inícial.

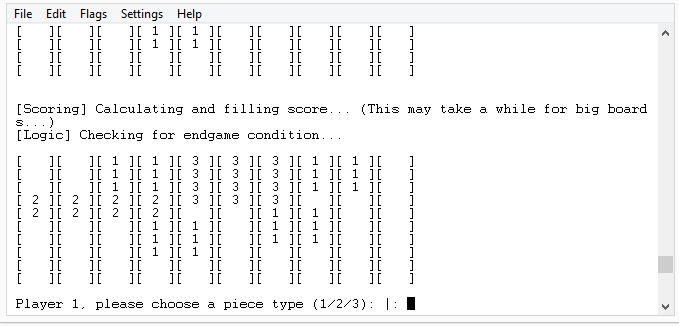


Figura 2- Tabuleiro de jogo numa fase avançada.

# 3.3 Lista de Jogadas Válidas

Existem um conjunto de regras lógicas que foram implementadas de maneira a que a jogabilidade do não fosse alterada. De maneira a cumprir essas mesma regras lógicas, foram implementados predicados responsáveis pela obtenção do conjunto de jogadas válidas num dado estado do jogo. Entre as quais destacam-se:

* getAvailableTurnAtIndex(Board, BoardSizeX, BoardSizeY, Index, RetX, RetY, RetType, RetOrientation) – predicado responsável pela obtenção de jogadas válidas num determinado “Index”. Recorre aos predicados auxiliares cAvailableTurn e itbAvailableTurn.

# 3.4 Execução de jogadas

# A verificação do cumprimento de regras está estritamente relacionada com o ponto anterior, uma vez que o programa compara a lista com o conjunto de jogadas válidas, ou seja, verifica se a jogada está dentro das regras de jogo. Mas além disso, na validação de jogadas são implementadas ainda mais regras lógicas, estas mais gerais. Os predicados responsáveis pela implementação dessas mesma regras e posterior execução da jogada, são, nomeadamente:

# validateTurn(Board, PieceType, PieceOrientation, PieceX, PieceY, NewBoard) – predicado responsável pela validação de uma jogada no tabuleiro. Recorre aos predicados pieceHasFreeSpace, pieceHasNoAdjacentSameBlock , pieceHasAdjacentBlock, fillBoardWithNewBlock.

# fillBoardWithNewBlock(Board, PieceType, PieceOrientation, PieceX, PieceY, NewBoard) – predicado responsável pela alteração do tabuleiro de jogo durante a jogada , ou seja, execução de uma nova jogada.

# 3.5 Avaliação do Tabuleiro

# 3.6 Final do Jogo

# É realizada uma verificação do estado do jogo sempre que estão a ser calculados os pontos após uma determinada jogada. O predicado responsável por essa verificação é o fillBoardWithScoring que invoca o checkForAvailableTurns. Este último, verifica qual o número de jogadas possíveis a realizar na próxima jogada. Se o valor retornado for 0, significa que o jogo chegou ao fim. Então, é invocado o predicado checkForWinner que retorna qual o vencedor do jogo.

# checkForAvailableTurns(Board, BoardSizeX, BoardSizeY) – predicado responsável por verificar se existem jogadas possíveis na próxima jogada.

# checkForWinner(\_, \_, BoardSizeY, \_, BoardSizeY, Player1Points, Player2Points, Winner) – predicado responsável analisar qual o jogador com mais pontos e indicar o vencedor.

# 3.6 Jogada do Computador

# A jogada do computador processa-se de maneira semelhante a uma jogada de computador, a diferença em questão está na maneira como é escolhida a peça e sua posição.

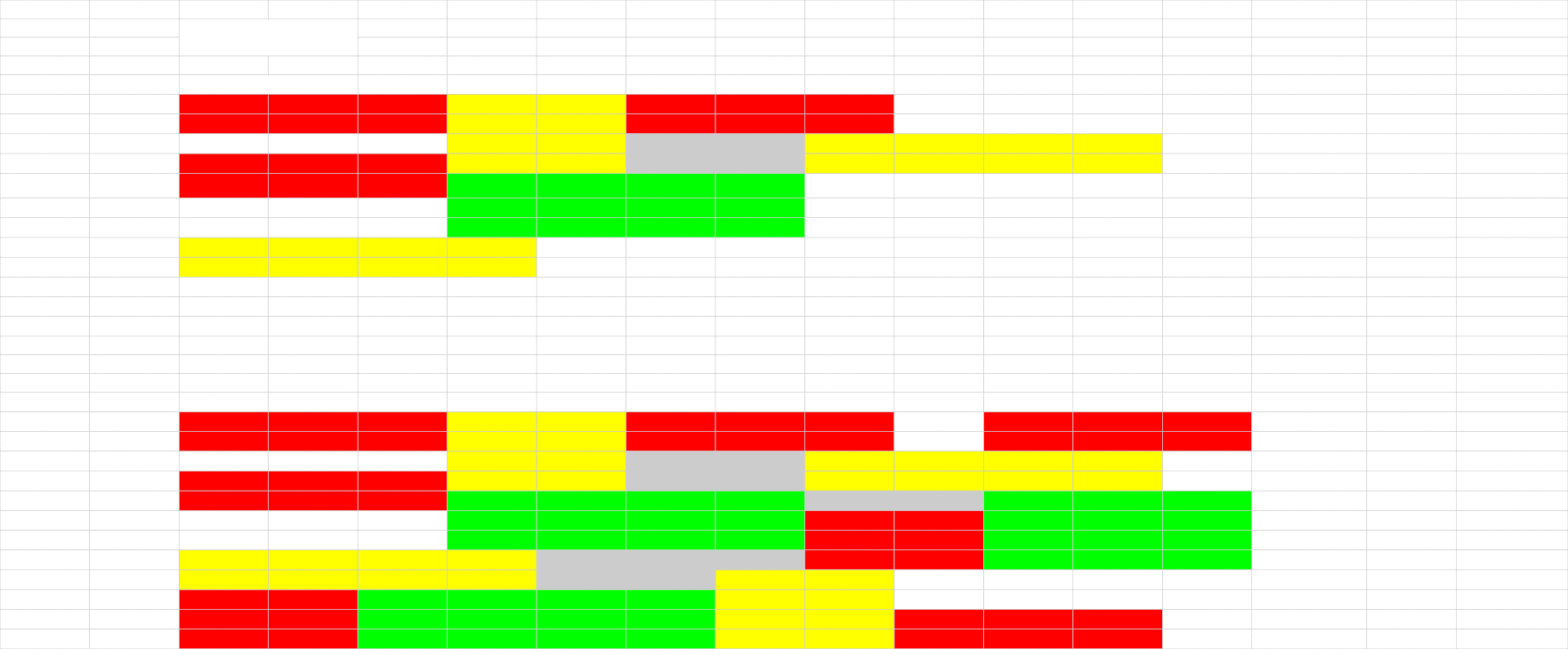
# Se o computador estiver no nível 1, é invocado o predicado getAnAvailableTurn que retorna uma jogada possível em determinado estado no jogo. Se o nível do computador estiver a 2, são verificadas o número de jogadas possíveis naqueles estado de jogo através do predicado countAvailableTurns, posteriormente é gerado um número aleatório entre 1 e o número de jogadas possíveis e posteriormente é selecionada a jogada X, em que X representa o número aleatório gerado. Esta seleção de jogada é feita através do predicado getAvailableTurnAtIndex. O predicado responsável por invocar todos os anteriores mencionados:

# playComputerino(Board, Difficulty, BoardSizeX, BoardSizeY, PlayCount, \_, NewBoard).

# 4 Jogada do Computador

Cada jogador, na sua vez, pode colocar um bloco 2x3, 2x4 ou 3x4, numa dada posi¸c˜ao.

* + play(BlockType, TopLeftCorner) - Predicado que permite ao jogador colo- car uma pe¸ca numa dada posi¸c˜ao (referida aqui pela posic¸˜ao da quadricula no canto superior esquerdo).
  + playAI() - Predicado que provoca uma jogada pela parte da inteligˆencia artificial.



Anexo 1

Estado Intermedio

Estado Final

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-2x4 | bloco-2x4 | bloco-2x3 | bloco-2x3 | bloco-2x3 | 0 | 0 | 0 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-2x4 | bloco-2x4 | bloco-2x3 | bloco-2x3 | bloco-2x3 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | bloco-2x4 | bloco-2x4 | 1 | 1 | bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-2x4 | bloco-2x4 | 1 | 1 | bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | 0 | 0 | 0 | 0 | 0 |
| bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |
| --- | --- |
| Átomos | Signficado |
| 0 | "Espaço livre |
| 1 | Ponto do Player 1 |
| 2 | Ponto do Player 2 |
| bloco-2x3 | Parte de bloco 2x3 |
| bloco-2x4 | Parte de bloco 2x4 |
| bloco-3x4 | Parte de bloco 2x4 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-2x4 | bloco-2x4 | bloco-2x3 | bloco-2x3 | bloco-2x3 | 0 | 0 | 0 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-2x4 | bloco-2x4 | bloco-2x3 | bloco-2x3 | bloco-2x3 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | bloco-2x4 | bloco-2x4 | 1 | 1 | bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-2x4 | bloco-2x4 | 1 | 1 | bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | 2 | 2 | bloco-3x4 | bloco-3x4 | bloco-3x4 |
| 0 | 0 | 0 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 |
| 0 | 0 | 0 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 |
| bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 1 | 1 | 1 | bloco-2x3 | 0 | bloco-3x4 | bloco-3x4 | bloco-3x4 |
| bloco-2x4 | bloco-2x4 | bloco-2x4 | bloco-2x4 | 1 | 1 | bloco-2x4 | bloco-2x4 | 0 | 0 | 0 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-2x4 | bloco-2x4 | 0 | 0 | 0 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-2x4 | bloco-2x4 | bloco-2x3 | bloco-2x3 | bloco-2x3 | 0 |
| bloco-2x3 | bloco-2x3 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-3x4 | bloco-2x4 | bloco-2x4 | bloco-2x3 | bloco-2x3 | bloco-2x3 | 0 |