MOD X

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Mod X_3:
António Manuel Vieira Ramadas - 201303568
Rui Miguel Teixeira Vilares - 201207046

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

9 de Outubro de 2015

1 O Jogo MOD X

1.1 Contextualização

O MOD X é um jogo de tabuleiro, nascido em 2015, recomendado para jogadores com mais de 15 anos de idade. É um jogo de estratégia, divertido e fácil de aprender. Cada partida reúne 2 a 4 jogadores e tem a duração prevista de 20 a 30 minutos.

1.2 Componentes do jogo

- 1 Tabuleiro 8x8;
- 56 peças de jogo em formato X (14 em cada cor);
- 72 marcadores de pontuação (18 em cada cor);
- 5 peças *Joker* em formato X (brancas);

Figura 1: Caixa do MOD X

1.3 Objetivo do jogo

O objetivo deste jogo é criar padrões com peças coloridas (vermelho, preto, amarelo e laranja). Pretende-se assim, formar o maior número de padrões possível, de forma a conseguir a melhor pontuação. O vencedor é o jogador com mais pontos. Evidentemente, é também suposto bloquear os adversários de modo que não consigam construir esses padrões[1].

1.4 Regras

- Define-se a ordem dos jogadores, cada um escolhe a cor das suas peças e define-se o limite de pontos[2];

[1] https://boardgamegeek.com/boardgame/131387/mod-x
[2] https://www.cryptozoic.com/games/mod-x

- Cada jogador inicia o jogo com 14 peças e 18 marcadores;
- Os *Jokers* são dispostos inicialmente de forma aleatória no tabuleiro;
- No seu turno, cada jogador coloca uma peça no tabuleiro, em qualquer posição livre;
- Os padrões utilizados para ganhar pontos são o "X", o "+" e o "cinco em linha";

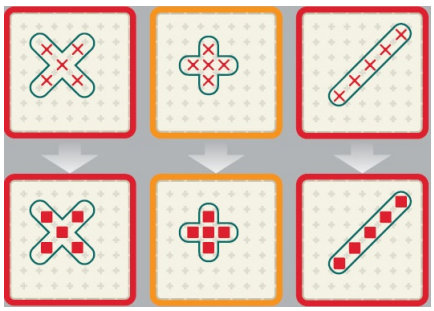
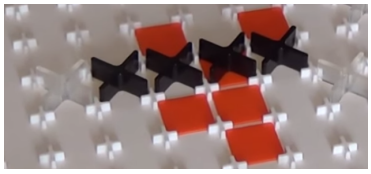
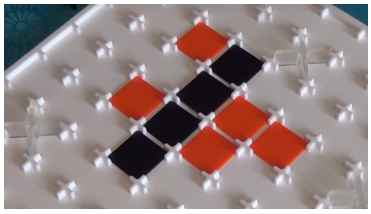


Figura 2: Padrões usados

- O jogador pode usar os *Jokers* para formar padrões, como se das suas próprias peças se tratassem;
- Quando um padrão é formado, retiram-se as peças de jogo e introduzem-se marcadores nessas posições. As peças de jogo podem agora voltar a ser utilizadas e as casas com marcadores também;
- Cada marcador colocado corresponde a um ponto;
- Caso tenha sido usado um *Joker* para formar um padrão, nessa posição não é introduzido um marcador. O *Joker* é agora colocado numa posição ao critério do jogador. Atenção, o *Joker* não pode ser usado para formar um novo padrão de imediato;



(a) Peças



(b) Marcadores

Figura 3: Substituição das peças por marcadores

- O primeiro jogador a atingir o número de pontos, determinado inicialmente, é o vencedor;
- O jogo pode também acabar quando um jogador já não dispõe de peças ou marcadores. Nesse caso, ganha o jogador com mais pontos até ao momento[3];

[3]https://youtu.be/pto-7O618rI

2 Representação do Estado do Jogo

Na representação exterior, ou seja, quando o tabuleiro é visualizado pelo utilizador, usamos as seguintes propriedades:

- Peças de jogo:
  - espaço em branco = livre (vazio)
  - 0 = *Joker* (transparente)
  - 1 = jogador 1
  - 2 = jogador 2
- Marcadores:
  - espaço em branco = vazio
  - * = marcador do jogador 1
  - : = marcador do jogador 2

Na representação interna, ou seja, nos dados armazenados pelo computador, usamos a seguinte sintaxe:

- Peças de jogo:
  - -1 = livre (vazio)
  - 0 = *joker* (transparente)
  - 1 = jogador 1
  - 2 = jogador 2
- Marcadores:
  - -1 = vazio
  - 11 = base do jogador 1
  - 22 = base do jogador 2

2.1 Representação do estado inicial do tabuleiro

$$[[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],$$
$$[-1,-1],[0,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],$$
$$[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],$$
$$[0,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,-1],[-1,-1],[-1,-1],$$
$$[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,-1],[-1,-1],$$
$$[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],$$
$$[-1,-1],[-1,-1],[0,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],$$
$$[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1]]$$

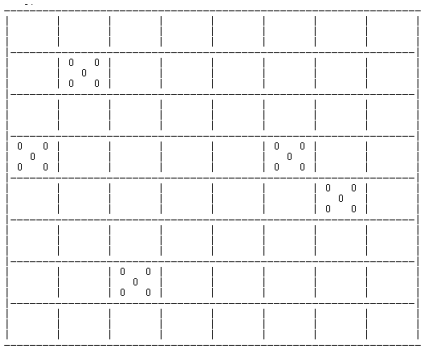


Figura 4: Estado inicial

2.2 Representação de um estado intermédio do tabuleiro

[[−1,−1],[2,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,11],[−1,−1],
[−1,−1],[0,−1],[−1,−1],[−1,−1],[−1,−1],[−1,11],[−1,11],[−1,11],
[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,11],[−1,−1],
[−1,−1],[−1,22],[−1,22],[−1,22],[−1,22],[0,−1],[−1,−1],[1,−1],
[0,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[0,−1],[−1,−1],
[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[1,−1],[−1,−1],[1,−1],
[2,−1],[2,−1],[0,−1],[2,−1],[1,−1],[−1,−1],[−1,−1],[−1,−1],
[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[0,−1]]

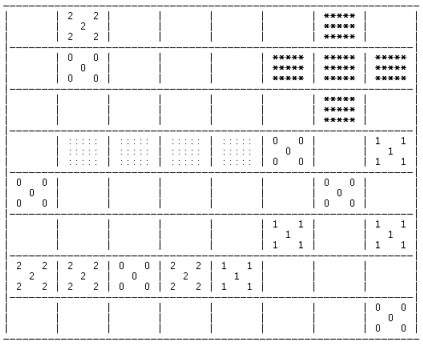


Figura 5: Estado intermédio

2.3 Representação de um estado final do tabuleiro

[[0,−1],[2,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,11],[0,−1],
[−1,−1],[0,−1],[−1,−1],[−1,−1],[2,−1],[2,11],[−1,11],[−1,11],
[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,11],[−1,−1],
[−1,−1],[−1,22],[1,22],[−1,22],[−1,22],[−1,−1],[−1,−1],[−1,11],
[0,−1],[−1,−1],[1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[−1,−1],
[−1,−1],[−1,−1],[−1,22],[0,−1],[−1,−1],[−1,11],[−1,−1],[−1,11],
[2,−1],[−1,22],[−1,−1],[−1,22],[1,−1],[−1,−1],[−1,−1],[−1,−1],
[−1,−1],[−1,−1],[−1,22],[−1,−1],[−1,−1],[−1,−1],[−1,−1],[0,−1]]

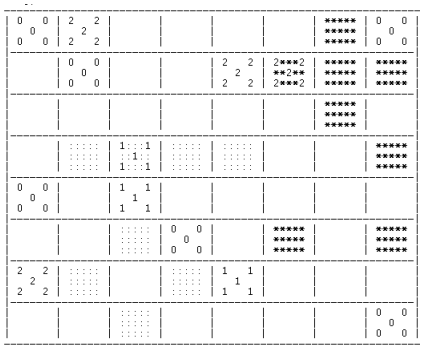


Figura 6: Estado final

3 Visualização do Tabuleiro

A visualização do tabuleiro em modo de texto é feita através da composição de caracteres ASCII, como visto anteriormente. Para este efeito, foram desenvolvidos um conjunto de predicados.

O predicado principal é:

- **printBoard(Board)**: recebe uma lista descritiva das peças que constituem o tabuleiro e imprime-o;

Os predicados auxiliares são:

- **printBoardAux(Board, CurrentNumberVertical)**: chamada auxiliar de *printBoard* que desenha *(Board.length mod 8)* vezes;
- **printHorizontalLine(NumberOfDashes)**: imprime o número de travessões horizontais pretendido;
- **printBlock(Line)**: imprime todos os blocos de uma determinada linha (horizontal);
  - **printLine1(' ', Tile)**: imprime a primeira linha do bloco;
  - **print1(' ', Tile)**: predicado auxiliar de *printLine1*, usada para a chamada recursiva;
  - **printLine2(' ', Tile)**: imprime a segunda linha do bloco;
  - **print2(' ', Tile)**: predicado auxiliar de *printLine2*, usada para a chamada recursiva;
  - **printLine3(' ', Tile)**: imprime a terceira linha do bloco;
  - **print3(' ', Tile)**: predicado auxiliar de *printLine3*, usada para a chamada recursiva;
  - **toPrintMiddle(N)**: verifica se $N > 0 \wedge N < 8$;
  - **printBeginning(_)**: imprime o travessão vertical da primeira linha do bloco;
  - **printMiddle(_)**: imprime o travessão vertical da segunda linha do bloco;
  - **printEnd(_)**: imprime o travessão vertical da terceira linha do bloco;
- **convertCode([Cross|Tile], X, Y)**: traduz o átomo em peça de jogo;
  - **convertCodeAux(Cross, [Tile|_], X, Y)**: predicado auxiliar de *convertCode*;
  - **translateCodeToChar(X, Y)**: traduz o código X para o código Y;
- **printInfo(_)**: apresenta informação à cerca da representação do jogo;

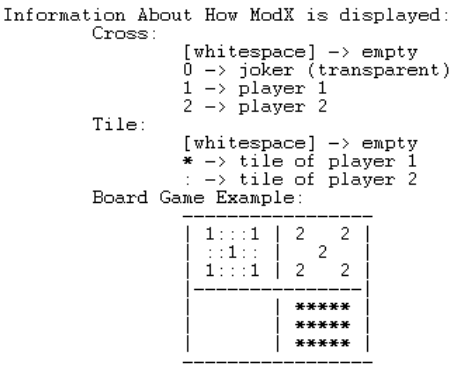


Figura 7: Informação geral apresentada

4 Movimentos

placeJoker(Board, Player, TileNumber).

Função usada para colocar o *Joker* numa posição específica. Falha se for escolhida numa posição incompatível.

validMoves(Board, Player, ListOfMoves).

Devolve as jogadas possíveis em *ListOfMoves*.

move(Board, Player, TileNumber, NewBoard).

Coloca uma nova peça de jogo no tabuleiro. Falha se a peça não poder ser adicionada naquela posição.

value(Board, Player, TileNumber, Value).

Avalia a jogada e devolve o seu valor em *Value*.

gameOver(Board, Winner).

Caso o jogo tenha acabado, devolve o vencedor em *Winner*.

choose_Move(Board, Level, Move).

Função em que consoante os diferentes níveis de dificuldade (*Level*) devolve jogadas possíveis.

putTiles(Board, Player, NewBoard).

Substitui as peças de jogo pelos marcadores do *Player*.