Moon Harvesters

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

PLOG_TP1_RI_Grupo_Moon_Harvesters_2:
Vitor Miguel Saraiva Esteves - 201303104
João Manuel Guedes Ferreira - 201303880

Faculdade de Engenharia da Universidade do Porto

15 de Outubro de 2017

1 O Jogo Moon Harvesters

1.1 Contextualização

Moon Harvesters é um jogo de tabuleiro desenhado para 2 ou 4 jogadores. O jogador assume o papel de chefe de uma companhia mineira lunar, que desenvolveu o seu próprio e patenteado sistema de extracção de Helium-3[1]. De modo a rentabilizar a operação, é necessário também enviar periodicamente robots lunares com o objetivo de identificar a posição de novos depósitos para extracção. **Moon Harvesters** é a reimplementação de outro jogo, do mesmo criador, designado **The Bitcoin Harvest**.

1.2 Componentes do jogo

- Tabuleiro com a superfície lunar (Grelha retangular 12*12 ou 12*16);
- 40 peças harvester (10 de cada um dos 4 tipos possíveis);
- 80 depósitos de Helium-3;
- 3 crateras de diferentes cores, que funcionarão como contadores de pontos;
- Caixa do Jogo e livro de instruções.

Figura 1: Tabuleiro de Moon Harvester

1.3 Objetivo do jogo

O objetivo deste jogo é capturar o maior número de depósitos de Helium-3 possível, sobrepondo os harvesters a estes. Os jogadores jogam por turnos e posicionam o respectivo harvester na posição mais favorável e dois novos depósitos de helium-3 em casas não ocupadas. Quando todos os jogadores passam o seu turno sucessivamente, o jogo acaba. No final do jogo, o vencendor do jogo é o jogador que capturou um maior número de Helium-3[2].

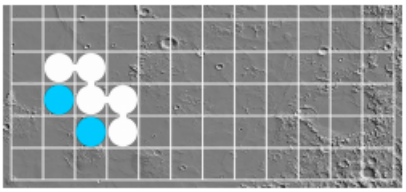
1.4 Regras

- Cada jogador escolhe uma cratera que representará a sua cor e em seguida, de maneira alternada, será feita a selecção do tipo de harvester a usar. A ordem de escolha das cores e do tipo de harvester fica ao critério dos participantes que devem decidir entre eles de alguma forma razoável;
- O jogador coloca o seu harvester no tabuleiro. Deve certificar-se que o mesmo não se sobrepõe a peças pré-existentes e que nenhuma das suas partes se encontra fora do tabuleiro. É ainda possível rodar a peça antes de a colocar[3].

[1] Helium-3 é um dos únicos dois isótopos estáveis de hélio com mais protões que neutrões.
[2] http://www.nestorgames.com/rulebooks/MOONHARVESTERS_EN.pdf
[3] Recomenda-se que se sobreponha ao maior numero possível de Helium-3

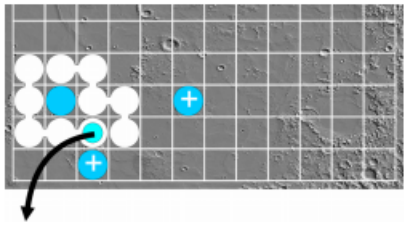
Figura 2: peças

- O jogador coleciona todos os depósitos de Helium-3 que o seu harvester tenha sobreposto.
- Cada depósito de Helium-3 é colocado na cratera e equivale a um ponto.
- Antes do turno seguinte, o jogador coloca 2 depósitos de Helium-3 em posições não ocupadas do campo.
- Caso o jogador não consiga efetuar uma jogada, deverá passar a sua vez.
- O jogo termina caso todos os jogadores passem a sua vez consecutivamente.
- Esvaziam-se as crateras e contam-se os pontos. O vencedor é o jogador que tiver capturado mais Helium-3. Em caso de empate, o jogador que tiver colocado o menor número de harvesters é o vencedor.



Example: *Player W places a harvester and then 2 deposits.*

Figura 3: Jogada inicial



Example: *Player L places a harvester atop a deposit to collect it. Then L places 2 deposits (marked with the '+' symbol)*

Figura 4: Jogada na qual é capturado um depósito de Hélio-3

2 Representação do Estado do Jogo

O tabuleiro será, a nível de programacão, representado por uma matriz ou, mais especificamente, uma lista de listas que representarão o estado do jogo. A nível de representação gráfica serão usados os seguintes componentes:

- **Representação gŕafica:**
  - espaços em branco = território não ocupado (livre)
  - 0 = marcador de depósito de Helium-3
  - 1 = um conjunto de números '1' será utilizado para indicar um harvester relativo ao jogador 1
  - 2 = um conjunto de números '2' será utilizado para indicar um harvester relativo ao jogador 2

A nível de representação lógica, a abordagem sintática escolhida teve em conta as posições possíveis para as várias peças de jogo e foi a seguinte:

- **Representação lógica - Jogador 1:**
  - -1 = livre (vazio)
  - 0 = marcador de depósito de Helium 3 (independente do jogador)
  - 1 = marcador de peça do jogador 1
  - 11 = harvester do jogador 1 com formato T
  - 12 = harvester do jogador 1 com formato W
  - 13 = harvester do jogador 1 com formato V
  - 14 = harvester do jogador 1 com formato U

- **Representação lógica - Jogador 2:**
  - -1 = livre (vazio)
  - 0 = marcador de depósito de Helium 3 (independente do jogador)
  - 2 = marcador de peça do jogador 2
  - 21 = harvester do jogador 2 com formato T
  - 22 = harvester do jogador 2 com formato W
  - 23 = harvester do jogador 2 com formato V
  - 24 = harvester do jogador 2 com formato U

2.1 Representação de um possível estado inicial do tabuleiro

```
printBoard([
[2,23],[2,23],[2,23],[1,11],[1,11],[1,11],[-1,-1],[-1,-1],[-1,-1],
[2,23],[-1,-1],[-1,-1],[-1,-1],[1,11],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[2,23],[-1,-1],[-1,-1],[-1,-1],[1,11],[-1,-1],[0,0],[-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,-1],[-1,-1],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[0,0],[-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,-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]
]).
```

```
Board Game Example:
|-------------------------------------------------------|
| 222 | 222 | 222 | 111 | 111 | 111 |     |     |     |
| 222 | 222 | 222 | 111 | 111 | 111 |     |     |     |
| 222 | 222 | 222 | 111 | 111 | 111 |     |     |     |
|----- ----- -----|----- ----- -----|----- ----- -----|
| 222 |     |     |     | 111 |     |     |     |     |
| 222 |     |     |     | 111 |     |     |     |     |
| 222 |     |     |     | 111 |     |     |     |     |
|----- ----- -----|----- ----- -----|----- ----- -----|
| 222 |     |     |     | 111 |     | 000 |     |     |
| 222 |     |     |     | 111 |     | 000 |     |     |
| 222 |     |     |     | 111 |     | 000 |     |     |
|-----------------+-----------------+-----------------|
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|----- ----- -----|----- ----- -----|----- ----- -----|
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|----- ----- -----|----- ----- -----|----- ----- -----|
|     |     |     |     |     |     | 000 |     |     |
|     |     |     |     |     |     | 000 |     |     |
|     |     |     |     |     |     | 000 |     |     |
|-----------------+-----------------+-----------------|
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|----- ----- -----|----- ----- -----|----- ----- -----|
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|----- ----- -----|----- ----- -----|----- ----- -----|
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|-------------------------------------------------------|
```

Figura 5: Exemplo de um estado inicial

2.2 Representação de um possível estado intermédio do tabuleiro

```
printBoard([
[2,23],[2,23],[2,23],[1,11],[1,11],[1,11],[-1,-1],[-1,-1],[-1,-1],
[2,23],[2,23],[2,23],[2,23],[1,11],[-1,-1],[-1,-1],[-1,-1],[-1,-1],
[2,23],[2,23],[0,0],[-1,-1],[1,11],[-1,-1],[0,0],[-1,-1],[-1,-1],
[-1,-1],[2,23],[-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],
[1,11],[1,11],[1,11],[1,11],[1,11],[1,11],[2,23],[2,23],[2,23],
[0,0],[1,11],[-1,-1],[-1,-1],[1,11],[-1,-1],[2,23],[-1,-1],[-1,-1],
[-1,-1],[1,11],[0,0],[-1,-1],[1,11],[-1,-1],[2,23],[-1,-1],[-1,-1],
[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1],[-1,-1]
]).
```

Board Game Example:

Player1: 0
Player2: 1

222	222	222	111	111	111			
222	222	222	111	111	111			
222	222	222	111	111	111			
222	222	222	222	111				
222	222	222	222	111				
222	222	222	222	111				
222	222	000		111		000		
222	222	000		111		000		
222	222	000		111		000		
000	222							
000	222							
000	222							
111	111	111	111	111	111	222	222	222
111	111	111	111	111	111	222	222	222
111	111	111	111	111	111	222	222	222
000	111			111		222		
000	111			111		222		
000	111			111		222		
	111	000		111		222		
	111	000		111		222		
	111	000		111		222		

Figura 6: Exemplo de um estado intermédio

2.3 Representação de um estado final do tabuleiro

```
printBoard([
[2,23],[2,23],[2,23],[1,11],[1,11],[1,11],[2,23],[2,23],[2,23],
[2,23],[2,23],[2,23],[2,23],[1,11],[1,11],[1,11],[1,11],[2,23],
[2,23],[2,23],[0,0],[1,11],[1,11],[-1,-1],[1,11],[1,11],[2,23],
[0,0],[2,23],[-1,-1],[1,11],[-1,-1],[-1,-1],[1,11],[1,11],[0,0],
[0,0],[0,0],[1,11],[1,11],[1,11],[0,0],[1,11],[1,11],[1,11],
[1,11],[1,11],[1,11],[1,11],[1,11,[1,11],[2,23],[2,23],[2,23],
[0,0],[1,11],[2,23],[0,0],[1,11],[2,23],[2,23],[1,11],[0,0],
[-1,-1],[1,11],[2,23],[-1,-1],[1,11],[2,23],[2,23],[1,11],[0,0],
[2,23],[2,23],[2,23],[2,23],[2,23],[2,23],[-1,-1],[1,11,[1,11]
]).
```

Board Game Example:

Player1: 1
Player2: 2

222	222	222	111	111	111	222	222	222
222	222	222	111	111	111	222	222	222
222	222	222	111	111	111	222	222	222
222	222	222	222	111	111	111	111	222
222	222	222	222	111	111	111	111	222
222	222	222	222	111	111	111	111	222
222	222	000	111	111		111	111	222
222	222	000	111	111		111	111	222
222	222	000	111	111		111	111	222
000	222		111			111	111	000
000	222		111			111	111	000
000	222		111			111	111	000
000	000	111	111	111	000	111	111	111
000	000	111	111	111	000	111	111	111
000	000	111	111	111	000	111	111	111
111	111	111	111	111	111	222	222	222
111	111	111	111	111	111	222	222	222
111	111	111	111	111	111	222	222	222
000	111	222	000	111	222	222	111	000
000	111	222	000	111	222	222	111	000
000	111	222	000	111	222	222	111	000
	111	222		111	222	222	111	000
	111	222		111	222	222	111	000
	111	222		111	222	222	111	000
222	222	222	222	222	222	111	111	111
222	222	222	222	222	222	111	111	111
222	222	222	222	222	222	111	111	111

Figura 7: Exemplo de um estado final

3 Visualização do Tabuleiro

De modo a obter a interface gráfica mostrada previamente, foram desenvolvidos um conjunto de protótipos de predicados. Achámos assim por bem utilizar os seguintes:

- Uma função inicial **printBoard(Board)** que recebe uma lista das peças (*board*) que constituem o tabuleiro e imprime-o;

Para auxiliar o predicado anterior desenvolveram-se ainda predicados auxiliares:

- **printBoardAux(Board, CurrentNumberHorizontal)**: chamada auxiliar de *printBoard* que permite desenhar o tabuleiro de forma dinâmica;
  - **printTopLine(NumberofDashes)**: imprime o topo do tabuleiro;
  - **printClearCell('_')**: imprime uma linha inteira (espaço vazio e barras laterais);
  - **NormalLineofTiles(NumberofCells)**, predicado auxiliar de printClearCell.Utilizado para chamar recursivamente printClearCell "NumberofCells"vezes;
  - **printFullCell(NumberofLines)**: utiliza os predicados anteriores para criar uma célula completa.
  - **printIntermediateLine(NumberofCells)**: predicado utilizado para imprimir linhas intermédias esteticamente delimitadoras.
  - **printFullTile(NumberofTiles)**: conjuga os predicados printFullCell e printIntermediateLine para obter uma tile constituida por uma célula e secções delimitadoras;
  - **printSectorLimit(NumberofSectors)**: predicado auxiliar de printFullSector, que delimita um sector;
  - **printFullSector**: utiliza printFullTile e o predicado auxiliar de limite, para criar um sector completo;
  - **printBottomLine(NumberofDashes)**: imprime a linha final do tabuleiro;
- **convertCode(_)**: traduz os átomos em peças de jogo;
  - **translateCodeToChar(X, Y)**: tradução dos numéros X para os caractéres Y;
- **printExample(_)**: Imprime a informação relativa à representação básica do *Moon Harvesters*, aos seus elementos e ainda apresenta um exemplo da *Board* num determinado estado do jogo.

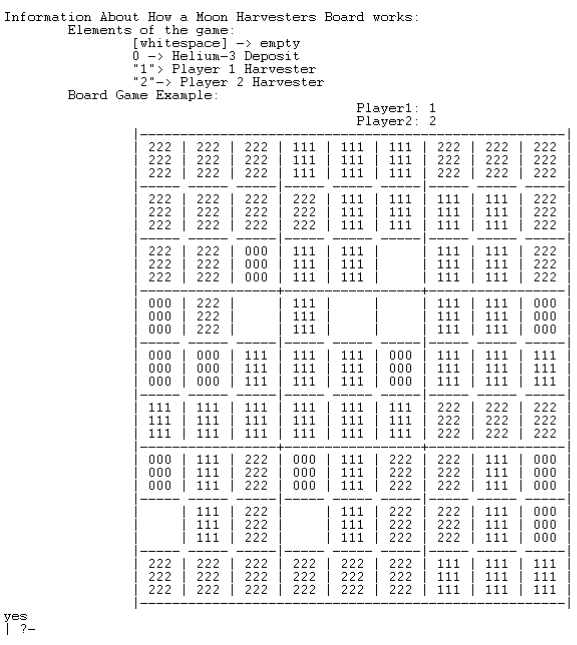


Figura 8: Informação geral apresentada

4 Movimentos

Os seguintes movimentos estão disponíveis no jogo *Moon Harvesters*:

- **choosePiece(+Player, -PieceType).**
  - Selecciona qual tipo de harvester o jogador deseja utilizar. É apenas chamado no inicio do jogo.
- **movePiece(+Board, +Player, +TileNumberInit, +TileNumberFinal, -NewBoard).**
  - Função utilizada para movimentação das peças. Utiliza *TileNumberInit* e *TileNumberFinal* como indicadores de posição inicial e final do harvester.
- **validMoves(+Board, +Player, -PossibleMoves).**
  - Devolve as jogadas possíveis. PossibleMoves representa uma lista de coordenadas para movimentos possíveis.
- **value(+Board, +Player, +TileNumberInit, +TileNumberFinal, -Value).**
  - Avaliação do potencial de jogadas. Devolve o seu valor em *Value*.
- **gameOver(+Board, -Winner).**
  - Devolve o vencedor do jogo em *Winner*.
- **rotatePiece(+Board, +Player,+TileNumberInit, +TileNumberFinal, +PieceType, -NewBoard).**
  - Muda a orientação do harvester escolhido.