Nodes

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Nodes_3:
Carolina Centeio Jorge - up201403090
Tiago Almeida - up201305665

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

16 de Outubro de 2016

1 Nodes: The Game

1.1 History

Nodes is a board game of abstract strategy, created by The Game Crafter and designed by RGBY Games. It was realeased on September 10, 2016 and it is still on its 1st edition. This game was made for casual gamers, in particular, for people who like chess due to their similarity. Nodes require 2 to 4 players over the age of 12.

1.2 Brief Description

1.2.1 Pieces

Nodes: Each player starts with **1** node. Nodes are like kings in chess: they can only move **one space in each turn**. In this game, they are also communication hubs. They emit signals in all eight directions (front, back, sides and diagonals), called **lines of communication**.

Units: Each player starts with **8** units. Unlike pawns in chess, units can move as many spaces as they want in each turn, **as long as they are along** a line of communication.

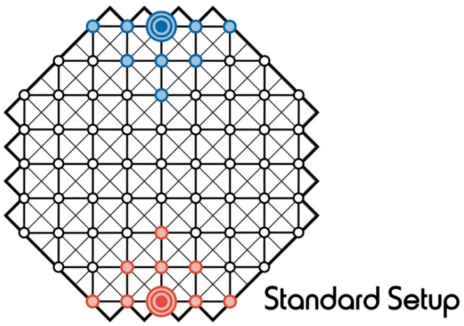

Fig. 1: Initial board set up

1.2.2 Board

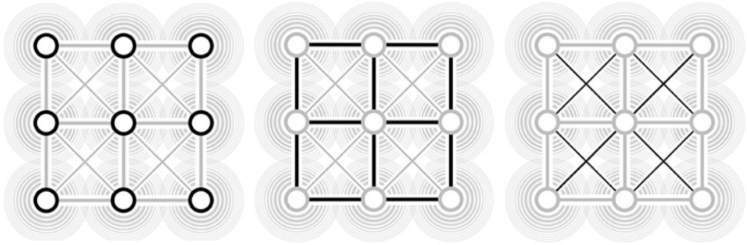

Fig. 2: Spaces, Roads and Conduits

Spaces: White circles where both units and nodes can finish their turn.
Roads: Boldest lines connecting spaces along which both pieces can draw ther paths.
Conduits: Thin crossing lines that cannot be part of a piece's path.

1.3 Rules

1.3.1 Lines of Communication

Lines of communication are the basis of the movement. Each node emits a line of communication along each road and conduit that surrounds it. All lines of communication are available in every player's turn, even if they are being emited by another player's node. Relay and Interception?

1.3.2 Unit Movements

Units can move through communication lines util the player decides to finish its turn. They cannot move through conduits. There is no limit on how many spaces a unit can move. Under some circumstances, it is possible for a unit to jump over another one. This can happen when when the spaces before and after the unit in the way are in the same line of communication: units cannot jump over nodes and can only jump one enemy unit at a time.

1.3.3 How to play

Each turn has three stages. The first one consists in visualizing the line of communication and determine which units can move. Then, the player moves each unit to the desired position (if possible).

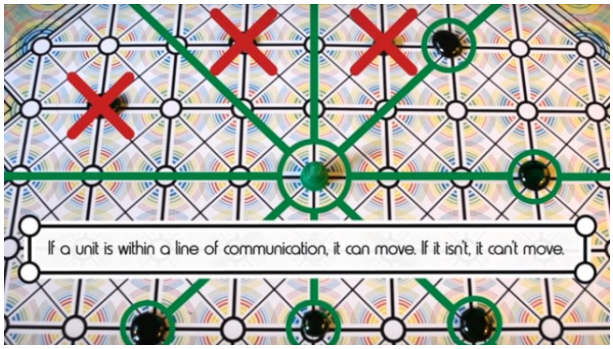

Fig. 3: In green, the communication Lines. Units with a cricle can move. Units with a cross cannot.

2 Representation of the Game State

The board is composed of a list of lists (representing each line) of lists (margin and content). This is a two player game representation.

It is an octogonal board and it uses units and nodes with the number of the player (ex: unit1 and node1 are an unit and a node of the player one. They are represented as 'x' and '1', respectively. For player 2, there are 'o' for units and '2' for the node).

This is the code that describes the different stages of the game (images follow):

Initial Formation:

```
board( [[[null,roof], [unit1,unit1,node1,unit1,unit1]],
        [[roof], [empty,empty,unit1,unit1,unit1,empty,empty]],
        [[], [empty,empty,empty,empty,unit1,empty,empty,empty,empty]],
        [[], [empty,empty,empty,empty,empty,empty,empty,empty,empty]],
        [[], [empty,empty,empty,empty,empty,empty,empty,empty,empty]],
        [[], [empty,empty,empty,empty,empty,empty,empty,empty,empty]],
        [[], [empty,empty,empty,empty,unit2,empty,empty,empty,empty]],
```

```
[[null], [empty,empty,unit2,unit2,unit2,empty,empty]],
[[null,null], [unit2,unit2,node2,unit2,unit2]]]).
```

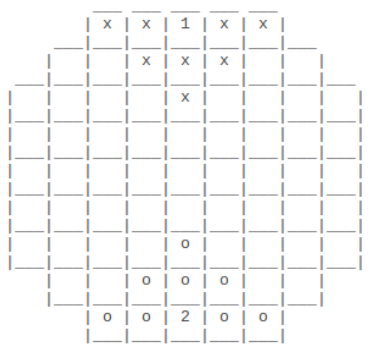

Fig. 4: Initial formation for 2 players.

Middle Stage:

```
board( [[[null,roof], [empty,empty,empty,empty,empty]],
       [[roof], [empty,empty,empty,empty,empty,empty,empty]],
       [[], [empty,empty,empty,empty,unit1,empty,empty,empty,empty]],
       [[], [empty,empty,empty,unit2,unit2,empty,empty,empty,empty]],
       [[], [empty,node2,unit2,unit2,unit1,unit2,unit2,empty,empty]],
       [[], [empty,unit2,unit1,unit1,empty,empty,empty,unit2,empty]],
       [[], [unit1,unit1,unit1,empty,node1,empty,empty,empty,empty]],
       [[null], [empty,unit1,empty,empty,empty,empty,empty]],
       [[null,null], [empty,empty,empty,empty,empty]]]).
```

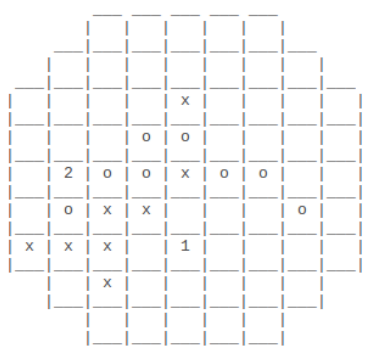

Fig. 5: An example of a middle game stage.

Final Stage:

```
board( [[[null,roof], [empty,empty,empty,empty,empty]],
       [[roof], [empty,empty,empty,empty,empty,empty,empty]],
       [[], [empty,empty,empty,empty,unit1,empty,empty,empty,empty]],
       [[], [empty,unit1,unit1,unit2,empty,empty,empty,empty,empty]],
       [[], [unit1,empty,node2,empty,empty,empty,empty,empty,empty]],
       [[], [unit1,unit2,unit1,empty,empty,empty,unit2,unit2,empty]],
```

```
[[], [empty, unit1, empty, empty, unit2, empty, empty, empty, empty]],
[[null], [empty, unit1, unit2, node1, unit2, empty, empty]],
[[null, null], [empty, empty, unit2, empty, empty]]]).
```

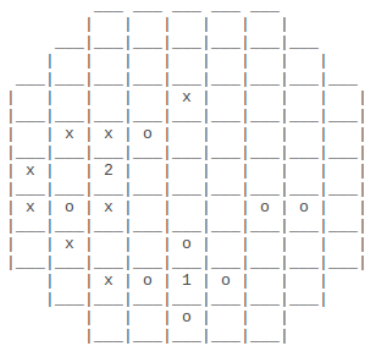

Fig. 6: Final stage: Node 1 surrounded.

3 Visualização do Tabuleiro

In this octogonal board, each square represents a **space**. In the original game, spaces are vertexes. For this reason, roads and conduits are not represented.

The board will be displayed with the **displayBoard(X)** predicate. This predicate draws the top of the board and calls the **displayBoardAux(X)** predicate which will display the lines. The **displayLine(X)** predicate draws the margins (**displayLineMargin(X)**), the content of the line (positions and its content: **displayLineContent(X)**) and the borders of the lines and board (**displayLineBottom(X)**, **displayLineMarginBottomLeft(X)** and **displayLineMarginBottomLeft(X)**).

```
translateContent(empty, '   ').
translateContent(unit1, ' x ').
translateContent(unit2, ' o ').
translateContent(node1, ' 1 ').
translateContent(node2, ' 2 ').
translateContent(roof, '   ').
translateContent(null, '   ').
translateBottom(X, '') :-
        member(X, [empty, unit1, unit2, node1, node2]).
translateBottomLeft(roof, ' ___').
translateBottomLeft(null, '    ').
translateBottomRight(roof, '___').
translateBottomRight(null, '    ').

displayLineMargin([]) :-
        write('').

displayLineMargin([LF1|LFs]) :-
        translateContent(LF1, V),
        write(V),
        displayLineMargin(LFs).

displayLineMarginBottomLeft([]):-
```

```
        write(' ').

displayLineMarginBottomLeft([LF1|LFs]) :-
        translateBottomLeft(LF1, V),
        write(V),
        displayLineMarginBottomLeft(LFs).

displayLineMarginBottomRight([]):-
        write(' ').

displayLineMarginBottomRight([LF1|LFs]) :-
        translateBottomRight(LF1, V),
        write(V),
        displayLineMarginBottomRight(LFs).

displayLineContent([]) :-
        write('|'),
        nl.

displayLineContent([LF1|LFs]) :-
        translateContent(LF1, V),
        write('|'),
        write(V),
        displayLineContent(LFs).

displayLineBottom([]) :-
        write('|').

displayLineBottom([LF1|LFs]) :-
        translateBottom(LF1, V),
        write(V),
        write('|___'),
        displayLineBottom(LFs).

displayLine([]) :-
        write(' ').

displayLine([E1|Es]):-
        displayLineMargin(E1),
        member(X, Es),
        displayLineContent(X),
        displayLineMarginBottomLeft(E1),
        member(X, Es),
        displayLineBottom(X),
        reverse(E1, Y),
        displayLineMarginBottomRight(Y),
        nl.

displayBoardAux([]) :-
        displayLine([]).

displayBoardAux([L1|Ls]):-
```

```
        displayLine(L1),
        displayBoardAux(Ls).

displayBoard(X) :-
        write('           ---  ---  ---  ---  ---'),
        nl,
        displayBoardAux(X).
```

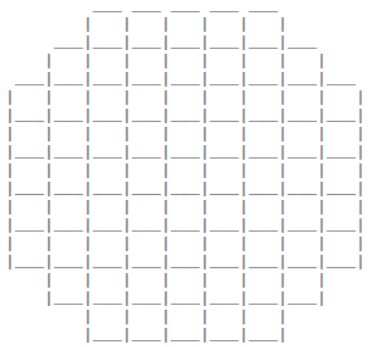

Fig. 7: Empty Board.

4 Movimentos

In each turn, players can move as many pieces as they want. For this to happen, there will be the following predicates:

```
up(+Board, +Player, -NewBoard).
down(+Board, +Player, -NewBoard).
right(+Board, +Player, -NewBoard).
left(+Board, +Player, -NewBoard).
```