Latice_3

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo L3:
José Pedro Soares João Pereira - up201304891
Pedro Romano Oliveira Silva Barbosa - up201306037

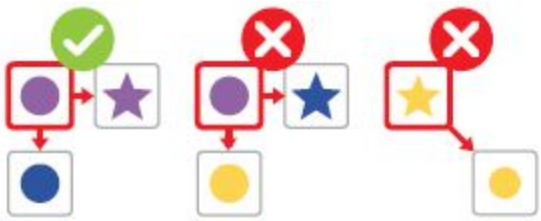
Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

05 de Outubro de 2015

1 O Jogo Latice

Latice, criado pela empresa Adacio, é um jogo de estratégia que pode ser jogado de 2 a 4 jogadores, com idade superior a 6 anos de idade, com um tempo previsível de jogo entre 10 a 30 minutos.

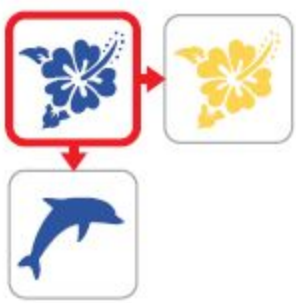
O objetivo de jogo, é ser o primeiro jogador a usar todas as suas peças, podendo apenas usa-las em posições adjacentes a outras peças do tabuleiro de jogo com a mesma cor ou forma da peça usada, podendo ganhar vantagem em relação a outros jogadores a partir de posições de jogo especiais e de peças singulares.



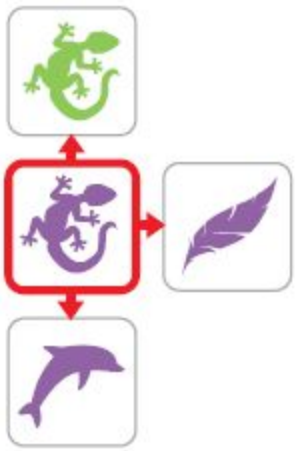
Cada jogo é constituido por um tabuleiro de jogo, 16 Sunstones e 16 Halfstones, e 84 peças jogáveis que serão divididas equitativamente pelo número de jogadores presentes.

As vantagens que cada jogador pode adquirir são personalizadas por dois tipos de pedras, Sunstones e Halfstones, cada Sunstone proporciona mais uma jogada ao jogador assim como duas Halfstones, caso sejam abdicadas por esses beneficios no inicio da jogada.

Uma Halfstone é ganha quando o jogador usa uma peça numa posição adjacente a outras duas peças com a mesma forma ou com a mesma cor da peça usada.

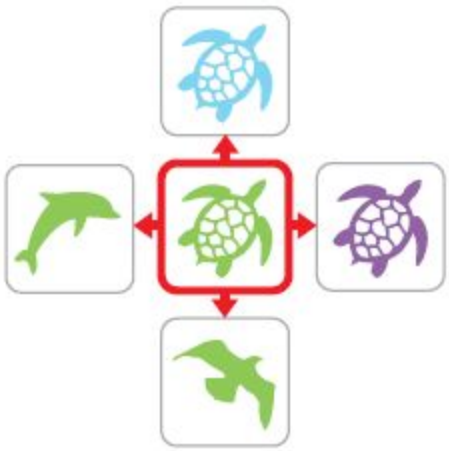


Cada Sunstone é ganha quando o jogador usa uma peça numa posição adjacente a outras três peças com a mesma forma ou com a mesma cor da peça usada.

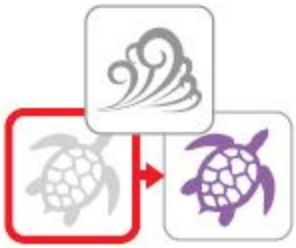


Outra opção para ganhar uma Sunstone, será quando o jogador utiliza uma das suas peças numa posição do tabuleiro com o simbolo de um Sol, *Sun Tile*:

Quando um jogador usa uma peça numa posição adjacente a outras quatro peças com a mesma forma ou com a mesma cor da peça usada, este jogador ganha duas Sunstones.



Entre as peças de jogo existe uma peça Wind que o jogador pode descartar em troca da possibilidade de mover uma das peças presentes no tabuleiro de jogo para uma das posições adjacentes dessa peça e ainda recebe uma jogada extra.



Uma peça Wind, também pode ser obtida por um jogador em troca de três Sunstones.

No inicio do jogo, o primeiro jogador coloca uma peça à sua escolha no centro do tabuleiro e completa a jogada retirando uma das peças que não estão em jogo.

Cada jogador tem 5 peças disponiveis para a jogada, e no fim desta é obrigado a retirar uma das peças que não estão em jogo com intuito de ter sempre as 5 peças disponiveis.

Um jogador pode abdicar da sua jogada para trocar algumas ou todas das suas peças por peças que não estavam em jogo.

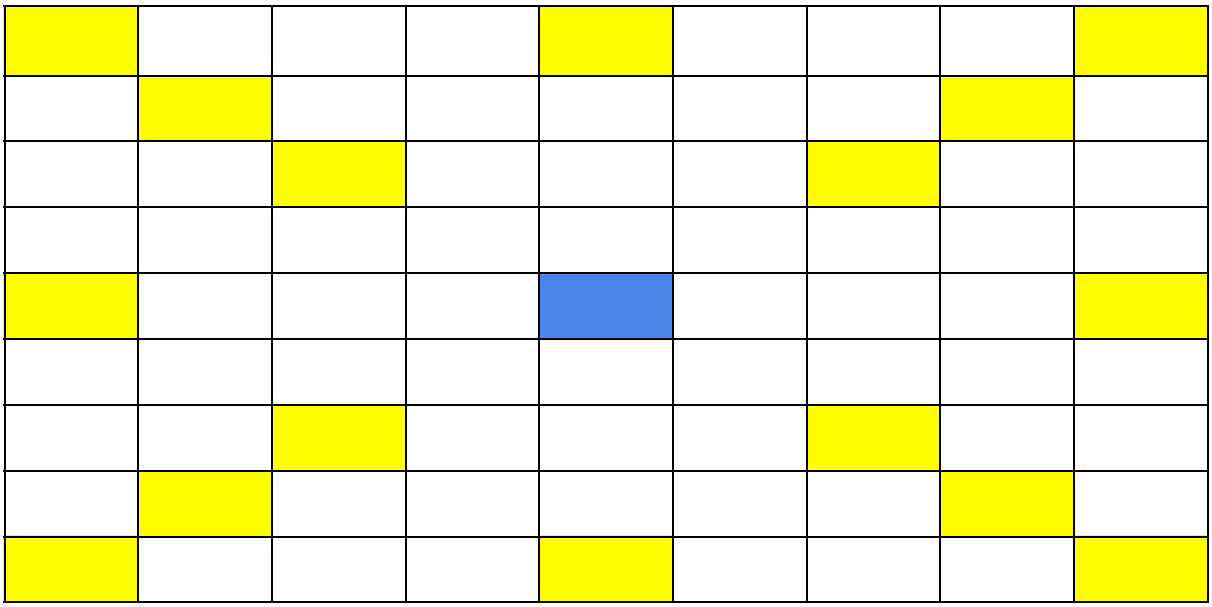
Caso um jogador não tenha uma peça possível de ser usada no tabuleiro de jogo, necessita de retirar peças de entre as peças que não estão em jogo, até que obtenha uma que possa ser jogada ou a ter não ter mais peças para retirar das que não estão em jogo.

Um jogador que não tenha peças para retirar de entre as que não estão em jogo, este é obrigado a passar a vez ao próximo jogador.

http://www.latice.com/latice/
http://www.boardgamegeek.com/boardgame/183959/latice

2 Representação do Estado do Jogo

O tabuleiro tem 81 posições divididas em secções quadradas, formando um quadrado de tabuleiro de jogo 9 posições por 9, como representado:



As posições a amarelo, são posições com o simbolo Sol, *Sun Tile*, que permitem ao jogador ganhar Sunstones. A posição a azul é a posição da primeira peça do jogo.

Para criar todas as posições do tabuleiro foi criada uma lista de listas com todas as posições possíveis do tabuleiro, designando por *f* uma posição normal de jogo, *s* uma posição especial de jogo, *Sun Tile*, e *m* a posição inicial da primeira peça de jogo:

```
tab([[s,f,f,f,s,f,f,f,s],
     [f,s,f,f,f,f,f,s,f],
     [f,f,s,f,f,f,s,f,f],
     [f,f,f,f,f,f,f,f,f],
     [s,f,f,f,m,f,f,f,s],
     [f,f,f,f,f,f,f,f,f],
     [f,f,s,f,f,f,s,f,f],
     [f,s,f,f,f,f,f,s,f],
     [s,f,f,f,s,f,f,f,s]]).
```

3 Visualização do Tabuleiro

Para se conseguir visualizar este tabuleiro em modo de texto foi criado um predicado, *viewTab/0* que recebe qualquer argumento e imprime o tabuleiro de jogo no ecrã:

```
| ?- viewTab.
```

E obtém-se o resultado:

```
| s | f | f | f | s | f | f | f | s |
| f | s | f | f | f | f | f | s | f |
| f | f | s | f | f | f | s | f | f |
| f | f | f | f | f | f | f | f | f |
| s | f | f | f | m | f | f | f | s |
| f | f | f | f | f | f | f | f | f |
| f | f | s | f | f | f | s | f | f |
| f | s | f | f | f | f | f | s | f |
| s | f | f | f | s | f | f | f | s |
```

Para obter este efeito, foi declarado um átomo *tab*, já mostrado acima, que contém as posições do tabuleiro de jogo, assim como o predicado *viewTab*, também já referido.

Adicionalmente foram criados quatro novos predicado:

- printList/1, recebe uma lista e imprime todos os seus elementos intercalados com " | " para obter um efeito visual mais facilmente perceptível
- printList/1, recebe uma lista vazia, imprime apenas um elemento final para que termine a recursividade do predicado anterior
- printLists/1, recebe uma lista de listas imprimindo cada uma dessas listas com o predicado acima referido
- printLists/1, recebe uma lista vazia, para que termine a recursividade do predicado anterior provocando o processamento da lista seguinte

A sua implementação encontra-se na seguinte imagem:

```
printLists([]).
printLists([H|T]) :-
    printList(H),
    printLists(T).

printList([]) :-
    write(' | '),
    nl.
printList([H|T]) :-
    write(' | '),
    write(H),
    printList(T).
```

4 Movimentos

As possíveis jogadas estão dependentes das peças já existentes no tabuleiro, seguindo as regras enumeradas no primeiro tópico, assim os possíveis movimentos apenas são realizados em posições adjacentes às peças que já se encontram no tabuleiro.

Os predicados que serão usados, são:

- useTile/2 :
    - recebe os argumentos: uma peça e uma posição;
    - objetivo: regista o uso de uma peça numa certa posição.
- sameColor/2 :
    - recebe os argumentos: duas peças;
    - objetivo: determina se as peças têm a mesma cor.
- sameFigure/2 :
    - recebe os argumentos: duas peças;
    - objetivo: determina se as peças têm a mesma forma.
- validPosition/2 :
    - recebe os argumentos: uma peça e uma posição;
    - objetivo: determina se uma peça pode ser usada numa certa posição
- checkForValidPosition/1:
    - recebe os argumentos: uma peça a ser usada;
    - objetivo: determina se uma peça pode ser usada em alguma posição do tabuleiro
- isSpecialPosition/1 :
    - recebe os argumentos: uma posição;
    - objetivo: determina se uma peça foi usada numa posição especial
- moveUp/1 :
    - recebe os argumentos: uma peça presente no tabuleiro do jogo;
    - objetivo: move a peça para cima, para a posição adjacente de onde se encontra
- moveDown/1 :
    - recebe os argumentos: uma peça presente no tabuleiro do jogo;
    - objetivo: move a peça para baixo, para a posição adjacente de onde se encontra
- moveLeft/1 :
    - recebe os argumentos: uma peça presente no tabuleiro do jogo;
    - objetivo: move a peça para a esquerda, para a posição adjacente de onde se encontra
- moveRight/1 :
    - recebe os argumentos: uma peça presente no tabuleiro do jogo;
    - objetivo: move a peça para a direita, para a posição adjacente de onde se encontra

Durante a elaboração do jogo, poderá vir a ser necessário a implementação de predicados adicionais para facilitar o desempenho das regras do jogo.