Class 2 T2_Three_Dragons_5

- Fábio Moreira (up201806296)
- Luís Tavares (up201809679)

The game: *Three Dragons*

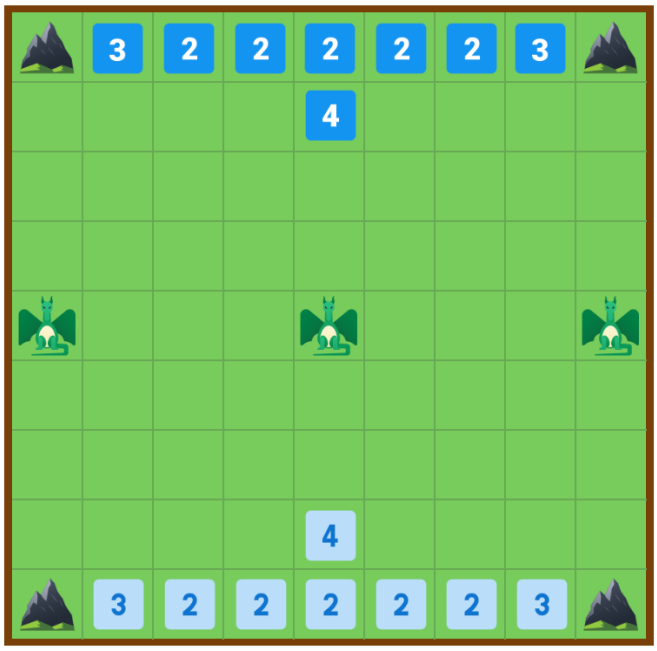
Three Dragons is a 2-player board game, created by Scott Allen Czysz, a board game designer. The game is inspired by ancient custodial capture board games, such as Tablut, with some differences:

1. The pieces have a related strength value. Only a stronger piece can capture a weaker piece;
2. There are three dragon caves in the board which allow players to get stronger pieces.

The latest version of the game is v0.40 released in 19th February 2020. The official Google Drive folder of the creator can be accessed here.

The board

The board consists a 9x9 square board with a mountain at each corner, three dragon caves in the middle row spaced with three squares, 8 white dices and 8 black dices (or any arbitrary color).



Gameplay

Gameflow

1. White move first.
2. At each turn the player must move a piece orthogonally any number of squares. Pieces may not overlap any mountain, dragon cave or game pieces.
3. A capture occurs when a player surrounds an opponent's piece on two opposite sides, or one player piece and a mountain or dragon cave. The enemy piece is removed.
4. The winner is declared when the opponent is reduced to only one piece.

Variants

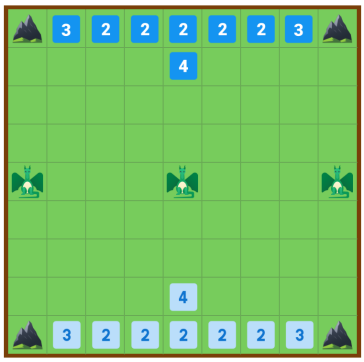
1. Capture by strength: occurs when the player move their piece to the side of a opponent's weaker piece. The opponent's piece is removed and the player's piece is weakened by 1 level.
2. Surrounding a dragon cave results in obtaining an extra playing piece (**summoning a dragon**). The side caves adds an extra 3 level piece to the player and the center cave an extra 5 level piece.

Gameplay notes

1. If a player moves their piece between two opponent pieces the piece **is not captured**.
2. It is only possible to obtain 3 dragons (1 for each cave).
3. The player can decide between a 'normal capture' or 'level capture'.
4. A player can capture multiple pieces with 'normal capture' but only one with 'level capture'.

Game representation

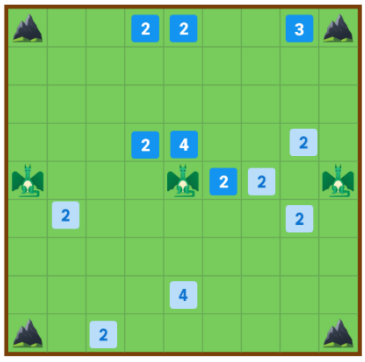
Initial State



```
initialBoard( [
    [mountain,black3,black2,black2,black2,black2,black2,black3,mountain],
    [empty,empty,empty,empty,black4,empty,empty,empty,empty],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [cave,empty,empty,empty,cave,empty,empty,empty,cave],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,empty,white4,empty,empty,empty,empty],
    [mountain,white3,white2,white2,white2,white2,white2,white3,mountain]
]).
```

	A	B	C	D	E	F	G	H	I
1	M	B3	B2	B2	B2	B2	B2	B3	M
2					B4				
3									
4									
5	C				C				C
6									
7									
8					W4				
9	M	W3	W2	W2	W2	W2	W2	W3	M

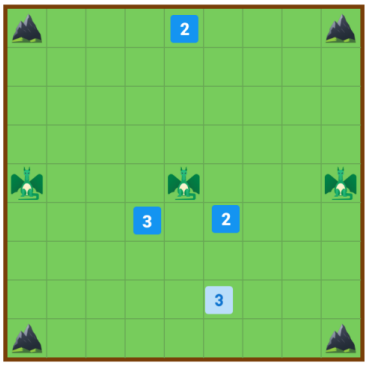
Middle State



```
middleBoard( [
    [mountain,empty,empty,black2,black2,empty,empty,black3,mountain],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,black2,black4,empty,empty,white2,empty],
    [cave,empty,empty,empty,cave,black2,white2,empty,cave],
    [empty,white2,empty,empty,empty,empty,empty,white2,empty],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,empty,white4,empty,empty,empty,empty],
    [mountain,empty,white2,empty,empty,empty,empty,empty,mountain]
]).
```

	A	B	C	D	E	F	G	H	I
1	M			B2	B2			B3	M
2									
3									
4				B2	B4			W2	
5	C				C	B2	W2		C
6		W2						W2	
7									
8					W4				
9	M		W2						M

Final State



```
finalBoard( [
    [mountain,empty,empty,empty,black2,empty,empty,empty,mountain],
    [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],
    [cave,empty,empty,empty,cave,empty,empty,empty,cave],
    [empty,empty,empty,black3,empty,black2,empty,empty,empty],
    [empty,empty,empty,empty,empty,empty,empty,empty,empty],
    [empty,empty,empty,empty,empty,white3,empty,empty,empty],
    [mountain,empty,empty,empty,empty,empty,empty,empty,mountain]
]).
```

	A	B	C	D	E	F	G	H	I
1	M				B2				M
2									
3									
4									
5	C				C				C
6				B3		B2			
7									
8						W3			
9	M								M

Board visualization

Since our board is represented as a list of lists of atoms, we use the predicate *translate* to print out the various elements (few examples provided below):

```
% :- translate(+Atom, -Translated).
translate(empty, S) :- S=' '.          %Empty cell
translate(black2,S) :- S=' B2 '.       %Black piece with relative strength of 2
translate(white3, S) :- S=' W3 '.      %White piece with relative strength of 3
translate(mountain, S) :- S=' M '.     %Mountain
translate(cave, S) :- S=' C '.         %Cave
```

In order to display the board, we use the predicate *displayBoard*, which will in turn, make calls to the predicates *printMatrix* and *printList*.

```
% :- displayBoard(+Board).
displayBoard(X) :-
    printColumnsRow,
    printSeparator,
    printMatrix(X,1).

% :- printMatrix(+Matrix, +N)
printMatrix([], _N).
printMatrix([Head | Tail], N) :-
    write(N),
    write('|'),
    printList(Head),
    printSeparator,
    N1 is N+1,
    printMatrix(Tail, N1).

% :- printList(+List).
printList([]) :- nl.
printList([Head|Tail]) :-
    translate(Head, X),
    write(X),
    write('|'),
    printList(Tail).
```