Mu Torere (Radiance) (3×3 Grid Version)

Board Numbering

```
0 | 1 | 2
--+---+--
3 | 4 | 5
--+---+--
6 | 7 | 8
```

- **Center (4)** = hub (*putahi*).
- **Outer (0,1,2,3,5,6,7,8)** = ring.

Setup

- Two players, each with **4 pieces** (● and ○).
- Pieces are placed **alternately around the ring**.
- Square **4** starts empty.

Example alternating setup:

```
● | ○ | ●
--+---+--
○ | ⊙ | ○
--+---+--
● | ○ | ●
```

Moves

1. Players alternate turns.
2. A piece may move into **square 4** if it is empty and adjacent.
3. A piece on the ring may move into an **empty ring square** *only if* it is directly **next to an opponent's piece**
   (adjacent horizontally, vertically, or diagonally).
4. A piece in **square 4** may move into any **adjacent empty ring square**.

Goal

- **Win by blocking your opponent** so they cannot make a legal move.

- There is no capturing — victory is achieved through movement and positioning.

In [1]:
```
from Game import *
```

```
Version:  0.3.14
```

- What is the state? (how are they represented?)
  - state = board, 0 empty, 1 player 1 piece, 2 player 2 piece
- What is a move?
  - [integer,integer] start,end

In [2]:
```
def initial_state():
    state=Board(3,3)
    for location in [0,2,6,8]:
        state[location]=1

    for location in [1,3,5,7]:
        state[location]=2

    state.pieces=["." , "●" , "○" ]
    return state
```

In [4]:
```
def show_state(state,player):
    print("Player",player)
    print(state)
```

```
 0 | 1 | 2
---+---+---
 3 | 4 | 5
---+---+---
 6 | 7 | 8
```

In [7]:
```
def adjacent(start,end):

    if start==0 and end in [1,3,4]:
        return True

    if start==1 and end in [0,2,4]:
        return True

    if start==2 and end in [1,5,4]:
        return True

    if start==3 and end in [0,6,4]:
        return True

    if start==4 and end in [0,1,2,3,5,6,7,8]:
        return True

    if start==5 and end in [2,8,4]:
        return True
    if start==6 and end in [3,7,4]:
        return True

    if start==7 and end in [6,8,4]:
        return True

    if start==8 and end in [5,7,4]:
        return True

    return False
```

In [8]:
```
def adjacent_squares(location):

    if location==0:
        return [1,3,4]

    if location==1:
        return [0,2,4]

    if location==2:
        return [1,5,4]

    if location==3:
        return [0,6,4]

    if location==4:
        return [0,1,2,3,5,6,7,8]

    if location==5:
        return [2,8,4]


    if location==6:
        return [3,7,4]


    if location==7:
        return [6,8,4]


    if location==8:
        return [5,7,4]


    raise ValueError("You can't get there from here")
```

In [10]:
```
def valid_moves(state,player):

    if player==1:
        other_player=2
    else:
        other_player=1

    moves=[]
    for start in range(9):
        if state[start]!=player:
            continue

        # this is a location with my piece

        for end in range(9):

            if state[end]==0 and end in adjacent_squares(start):

                # check for opponent in adjacent squares of the end

                if start!=4:
                    found=False
                    for check in adjacent_squares(end):
                        if state[check]==other_player:
                            found=True
                            break
                    if not found: # no opponent piece next to the end square
                        continue

                move=[start,end]
                moves.append(move)

        # todo: check if moving is next to opponent piece



    return moves
```

In [15]:
```
def update_state(state,player,move):
    start,end=move

    new_state=state
    new_state[start]=0
    new_state[end]=player

    return new_state
```

In [18]:
```
def win_status(state,player):
    # return None if the game is not over
    # return 'win' if player has won
    # return 'lose' if player has lost
    # return 'stalemate' if player has stalemate

    if player==1:
        other_player=2
    else:
        other_player=1

    if not valid_moves(state,other_player):
        return 'win'
```

Agents

In [19]:
```
def human_move(state,player):
    moves=valid_moves(state,player)
    print("Valid moves are: ",moves)
    move=None
    while move not in moves:
        move_input=input("Enter your move as start,end: ")
        start,end=move_input.split(",")
        move=[int(start),int(end)]

    return move

human_agent=Agent(human_move)
```

In [20]:
```
def random_move(state,player):
    return random.choice(valid_moves(state,player))
random_agent=Agent(random_move)
```

In [33]:
```
from Game.minimax import *
def minimax_move(state,player):
    values,actions = minimax_values(state,player,display=False,maxdepth=6)
    return top_choice(actions,values)
minimax_agent=Agent(minimax_move)
```

Running the Game

In [34]:
```
g=Game()
g.run(minimax_agent,random_agent)
```

```
====
Game  1
Player 1
 ●  ○  ●
 ○  .  ○
 ●  ○  ●

Player 1 moves [0, 4]
Player 2
 .  ○  ●
 ○  ●  ○
 ●  ○  ●

Player 2 moves [3, 0]
Player 1
 ○  ○  ●
 .  ●  ○
 ●  ○  ●

Player 1 moves [4, 3]
Player 2
 ○  ○  ●
 ●  .  ○
 ●  ○  ●

Player 2 moves [0, 4]
Player 1
 .  ○  ●
 ●  ○  ○
 ●  ○  ●

Player 1 moves [3, 0]
Player 2
 ●  ○  ●
 .  ○  ○
 ●  ○  ●

Player 2 moves [4, 3]
Player 1
 ●  ○  ●
 ○  .  ○
 ●  ○  ●

Player 1 moves [0, 4]
Player 2
 .  ○  ●
 ○  ●  ○
 ●  ○  ●

Player 2 moves [1, 0]
Player 1
 ○  .  ●
 ○  ●  ○
 ●  ○  ●

Player 1 moves [2, 1]
Player 2
 ○  ●  .
 ○  ●  ○
 ●  ○  ●

Player 2 moves [5, 2]
Player 1
 ○  ●  ○
 ○  ●  .
 ●  ○  ●

Player 1 moves [4, 5]
Player 2
 ○  ●  ○
 ○  .  ●
 ●  ○  ●

Player 2 moves [0, 4]
Player 1
 .  ●  ○
 ○  ○  ●
 ●  ○  ●

Player 1 moves [1, 0]
Player 2
 ●  .  ○
 ○  ○  ●
 ●  ○  ●

Player 2 moves [2, 1]
Player 1
 ●  ○  .
 ○  ○  ●
 ●  ○  ●

Player 1 moves [5, 2]
Player 2
 ●  ○  ●
 ○  ○  .
 ●  ○  ●

Player 2 moves [4, 5]
Player 1
 ●  ○  ●
 ○  .  ○
 ●  ○  ●

Player 1 moves [6, 4]
Player 2
 ●  ○  ●
 ○  ●  ○
 .  ○  ●

Player 2 moves [7, 6]
Player 1
 ●  ○  ●
 ○ ● ○
 ○ . ●

Player 1 moves [8, 7]
Player 2
 ● ○ ●
 ○ ● ○
 ○ ● .

Player 2 moves [5, 8]
Player 1
 ● ○ ●
 ○ ● .
 ○ ● ○

Player 1 moves [2, 5]
Player 2
 ● ○ .
 ○ ● ●
 ○ ● ○

Player 2 moves [1, 2]
Player 1
 ● . ○
 ○ ● ●
 ○ ● ○

Player 1 moves [0, 1]
Player 2
 . ● ○
 ○ ● ●
 ○ ● ○

Player 2 moves [3, 0]
Player 1
 ○ ● ○
 . ● ●
 ○ ● ○

Player 1 moves [4, 3]
Player 2
 ○ ● ○
 ● . ●
 ○ ● ○

Player 2 moves [0, 4]
Player 1
 . ● ○
 ● ○ ●
 ○ ● ○

Player 1 moves [1, 0]
Player 2
 ● . ○
 ● ○ ●
 ○ ● ○
```

```
Player 2 moves [4, 1]
Player 1
 ●  ○  ○
 ●  .  ●
 ○  ●  ○

Player 1 moves [7, 4]
Player 2
 ●  ○  ○
 ●  ●  ●
 ○  .  ○

Player 2 moves [8, 7]
Player 1
 ●  ○  ○
 ●  ●  ●
 ○  ○  .

Player 1 moves [5, 8]
Player 2
 ●  ○  ○
 ●  ●  .
 ○  ○  ●

Player 2 moves [2, 5]
Player 1
 ●  ○  .
 ●  ●  ○
 ○  ○  ●

Player 1 moves [4, 2]
Player 2
 ●  ○  ●
 ●  .  ○
 ○  ○  ●

Player 2 moves [1, 4]
Player 1
 ●  .  ●
 ●  ○  ○
 ○  ○  ●

Player 1 moves [0, 1]
Player 2
 .  ●  ●
 ●  ○  ○
 ○  ○  ●

Player 2 moves [4, 0]
Player 1
 ○  ●  ●
 ●  .  ○
 ○  ○  ●

Player 1 moves [3, 4]
Player 2
 ○  ●  ●
 .  ●  ○
 ○  ○  ●

Player 2 moves [6, 3]
Player 1
 ○  ●  ●
 ○  ●  ○
 .  ○  ●

Player 1 moves [4, 6]
Player 2
 ○  ●  ●
 ○  .  ○
 ●  ○  ●

Player 2 moves [0, 4]
Player 1
 .  ●  ●
 ○  ○  ○
 ●  ○  ●

Player 1 moves [1, 0]
Player 2
 ●  .  ●
 ○  ○  ○
 ●  ○  ●

Player 2 moves [4, 1]
Player 1
 ●  ○  ●
 ○  .  ○
 ●  ○  ●

Player 1 moves [2, 4]
Player 2
 ●  ○  .
 ○  ●  ○
 ●  ○  ●

Player 2 moves [1, 2]
Player 1
 ●  .  ○
 ○  ●  ○
 ●  ○  ●

Player 1 moves [4, 1]
Player 2
 ●  ●  ○
 ○  .  ○
 ●  ○  ●

Player 2 moves [7, 4]
Player 1
 ●  ●  ○
 ○  ○  ○
 ●  .  ●

Player 1 moves [6, 7]
Player 2
 ●  ●  ○
 ○  ○  ○
 .  ●  ●

Player 2 moves [3, 6]
Player 1
 ●  ●  ○
 .  ○  ○
 ○  ●  ●

Player 1 moves [0, 3]
Player 2
 .  ●  ○
 ●  ○  ○
 ○  ●  ●

Player 2 moves [4, 0]
Player 1
 ○  ●  ○
 ●  .  ○
 ○  ●  ●

Player 1 moves [3, 4]
Player 2
 ○  ●  ○
 .  ●  ○
 ○  ●  ●

Player 2 moves [0, 3]
Player 1
 .  ●  ○
 ○  ●  ○
 ○  ●  ●

Player 1 moves [1, 0]
Player 2
 ●  .  ○
 ○  ●  ○
 ○  ●  ●

Player 2 moves [2, 1]
Player 1
 ●  ○  .
 ○  ●  ○
 ○  ●  ●

Player 1 moves [4, 2]
Player 2
 ●  ○  ●
 ○  .  ○
 ○  ●  ●
```

```
Player 2 moves [3, 4]
Player 1
 ● ○ ●
 . ○ ○
 ○ ● ●

Player 1 moves [0, 3]
Player 2
 . ○ ●
 ● ○ ○
 ○ ● ●

Player 2 moves [4, 0]
Player 1
 ○ ○ ●
 ● . ○
 ○ ● ●

Player 1 moves [8, 4]
Player 2
 ○ ○ ●
 ● ● ○
 ○ ● .

Player 2 moves [5, 8]
Player 1
 ○ ○ ●
 ● ● .
 ○ ● ○

Player 1 moves [2, 5]
Player 2
 ○ ○ .
 ● ● ●
 ○ ● ○

Player 2 moves [1, 2]
Player 1
 ○ . ○
 ● ● ●
 ○ ● ○

Player 1 moves [4, 1]
Player 2
 ○ ● ○
 ● . ●
 ○ ● ○

Player 2 moves [6, 4]
Player 1
 ○ ● ○
 ● ○ ●
 . ● ○

Player 1 moves [7, 6]
Player 2
 ○  ●  ○
 ●  ○  ●
 ●  .  ○

Player 2 moves [8, 7]
Player 1
 ○  ●  ○
 ●  ○  ●
 ●  ○  .

Player 1 moves [5, 8]
Player 2
 ○  ●  ○
 ●  ○  .
 ●  ○  ●

Player 2 moves [2, 5]
Player 1
 ○  ●  .
 ●  ○  ○
 ●  ○  ●

Player 1 moves [1, 2]
Player 2
 ○  .  ●
 ●  ○  ○
 ●  ○  ●

Player 2 moves [4, 1]
Player 1
 ○  ○  ●
 ●  .  ○
 ●  ○  ●

Player 1 moves [3, 4]
Player 2
 ○  ○  ●
 .  ●  ○
 ●  ○  ●

Player 2 moves [0, 3]
Player 1
 .  ○  ●
 ○  ●  ○
 ●  ○  ●

Player 1 moves [4, 0]
Player 2
 ●  ○  ●
 ○  .  ○
 ●  ○  ●

Player 2 moves [3, 4]
Player 1
 ●  ○  ●
 .  ○  ○
 ●  ○  ●

Player 1 moves [6, 3]
Player 2
 ●  ○  ●
 ●  ○  ○
 .  ○  ●

Player 2 moves [4, 6]
Player 1
 ●  ○  ●
 ●  .  ○
 ○  ○  ●

Player 1 moves [8, 4]
Player 2
 ●  ○  ●
 ●  ●  ○
 ○  ○  .

Player 2 moves [5, 8]
Player 1
 ●  ○  ●
 ●  ●  .
 ○  ○  ○

Player 1 moves [2, 5]
Player 2
 ●  ○  .
 ●  ●  ●
 ○  ○  ○

Player 2 moves [1, 2]
Player 1
 ●  .  ○
 ●  ●  ●
 ○  ○  ○

Player 1 moves [0, 1]
Player 1
 .  ●  ○
 ●  ●  ●
 ○  ○  ○

Player  1 won.
```

```
Out[34]:  [1]
```

```
In [ ]:
```