Prácticas

Sistemas Inteligentes I

Sesión 3. Juegos

José A. Montenegro Montes
monte@lcc.uma.es

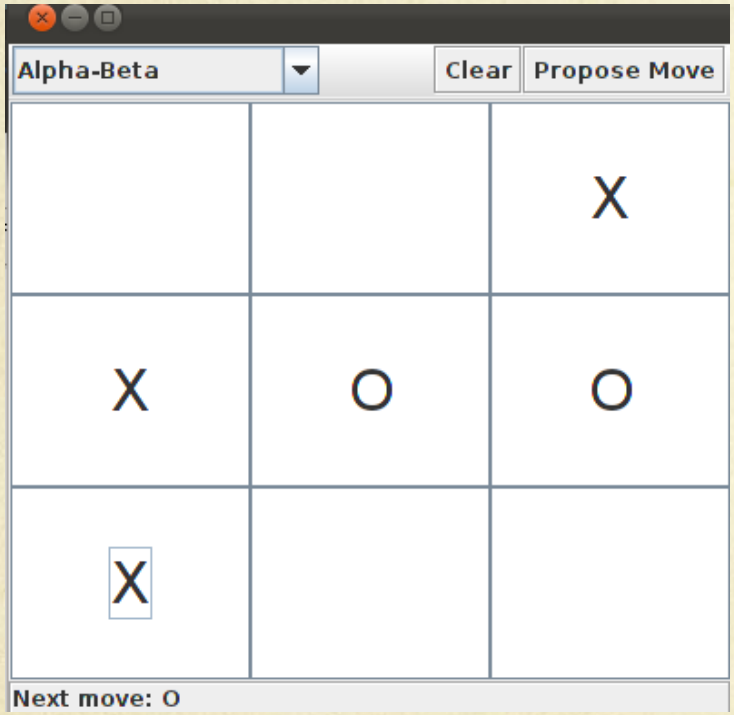
Resumen

- Ejercicios Tic-Tac-Toe (3 en raya)
- Práctica Entregar Nim

Ejercicio

Tic-Tac-Toe (3 en raya)

aima.gui.applications.search.games



TicTacToeApp

```
if (ae.getSource() == proposal)
    if (strategy.getSelectedIndex() == 0)
        game.makeMiniMaxMove();
    else  game.makeAlphaBetaMove();
```

aima.core.search.adversarial

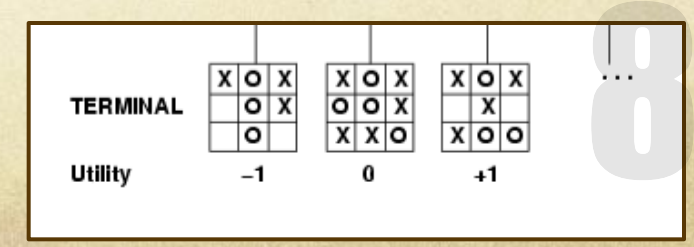
```
public void makeMiniMaxMove() {
        getMiniMaxValue(presentState);
        GameState nextState = (GameState) presentState.get("next");
        if (nextState == null)
                throw new RuntimeException("Mini Max Move failed");
        makeMove(presentState, nextState.get("moveMade"));
}

public void makeAlphaBetaMove() {
        getAlphaBetaValue(presentState);
        GameState nextState = (GameState) presentState.get("next");
        if (nextState == null)
                throw new RuntimeException("Alpha Beta Move failed");
        makeMove(presentState, nextState.get("moveMade"));
}
```

aima.core.search.adversarial

```
public abstract List<GameState> getSuccessorStates(GameState state);

public abstract GameState makeMove(GameState state, Object o);
public abstract int getMiniMaxValue(GameState state);
public abstract int getAlphaBetaValue(GameState state);

protected abstract int computeUtility(GameState state);
protected abstract boolean terminalTest(GameState state);
```

aima.core.environment.tictactoe

```
public int computeUtility(GameState state) {
        int utility =    computeUtility((TicTacToeBoard) state.get("board"),
                                 (getPlayerToMove(state)));
                return utility;
        }

private int computeUtility(TicTacToeBoard aBoard, String playerToMove) {
        int retVal = 0;
          if (aBoard.lineThroughBoard())
                if (playerToMove.equals("X"))  retVal = -1;
          else                                 retVal = 1;
        return retVal;
}
```



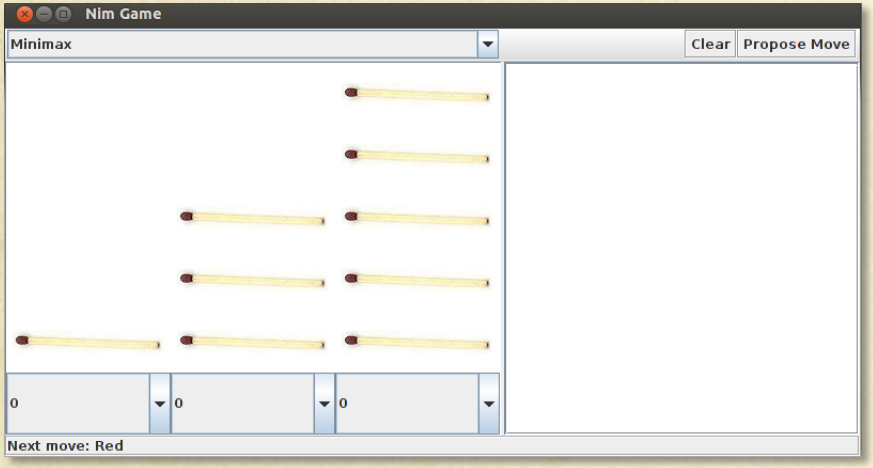
aima.core.environment.tictactoe

```
public boolean terminalTest(GameState state) {
        TicTacToeBoard board = (TicTacToeBoard) state.get("board");
        boolean line = board.lineThroughBoard();
        boolean filled = board.getNumberOfMarkedPositions() == 9;
        return (line || filled);
}
```

Práctica Entregar

Nim

Práctica Entregar Conecta4



NimApp

Implementar los métodos **isterminal** y **getUtility** de la clase Nim que permitan determinar la finalización del juego y la utilidad de un estado (para poder utilizar los algoritmos MINIMAX y la poda alfa-beta)

Práctica Entregar Nim

Utilizando los métodos definidos en NimBoard

```
public double getUtility (NimBoard state, NimPlayer player) {
                    // YOU MUST PROVIDE AN UTILITY FUNCTION HERE
        return 0.0;
}
```

```
protected boolean isTerminal(NimBoard state) {
        // YOU MUST PROVIDE A TERMINAL TEST HERE
        return true;
}
```

Sistemas Inteligentes

José A. Montenegro Montes

monte@lcc.uma.es