

# **B.M.S COLLEGE OF ENGINEERING**

**P.O. Box No.: 1908 Bull Temple Road,  
Bangalore - 560019**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**



**JAVA PROGRAMMING**

**19IS4PCJAV**

## **TIC TAC TOE**

Submitted by:

**1BM19IS112 - PRANAVA ADIGA**

**1BM19IS113 - PRANEETHA K**

**1BM19IS118 - PRATHVIRAJ PRABHU**

Submitted to:

**Mrs. Sindhu K**

**Assistant Professor**

# **B.M.S COLLEGE OF ENGINEERING**

**P.O. Box No.: 1908 Bull Temple Road,  
Bangalore - 560019**

## **DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**



## **CERTIFICATE**

Certified that the Project has been successfully presented at **B.M.S College Of Engineering** by **Pranava Adiga, Praneetha K, Prathviraj Prabhu** bearing **USN: 1BM19IS112, 1BM19IS113 and 1BM19IS118** in partial fulfillment of the requirements for the IV Semester degree in **Bachelor of Engineering in Information Science & Engineering** of **BMS COLLEGE OF ENGINEERING** Affiliated By **Visvesvaraya Technological University, Belgaum** as a part of the course **JAVA Programming (19IS4PCJAV)** during academic year 2020-2021.

**Faculty Name : Sindhu K**

**Designation : Assistant Professor**

**Department of ISE, BMSCE**

## TABLE OF CONTENTS

<b>SNO</b>	<b>CONTENT</b>	<b>PAGE NO</b>
1	ABSTRACT	3
2	PROBLEM STATEMENT	3
3	INTRODUCTION	3
4	OVERVIEW OF THE PROJECT	4
5	HIGH LEVEL DESIGN	4
6	TOOLS USED	5
7	IMPLEMENTATION	6
8	SNAPSHOT	14
9	REFERENCES	17

## **ABSTRACT**

The game of Tic-tac-toe is one of the most commonly known games. This game does not allow one to win all the time and a significant proportion of games played results in a draw. Thus, the best a player can hope is to not lose the game. This study is aimed at evolving a number of no - loss strategies using genetic algorithms and comparing them with existing methodologies. Moreover, an analysis of these solutions has given us insights about how to play the game to not lose it. Based on this experience, we have developed specialized efficient strategies having a high win-to-draw ratio. The study and its results are interesting and can be encouraging for the technique to be applied to other board games for finding efficient strategies.

## **PROBLEM STATEMENT**

By playing games, machine intelligence can be revealed. The problems are solved by forming a possible set of solutions based on the endgame condition, or searching for the set of solutions based on the current game condition. The machine cannot learn to play the games by itself. An evolutionary approach was employed to evolve and to learn for playing Tic-Tac-Toe without the need of a database. The Java Programming language and the branch of computer science that aims to create it. Tic-Tac-Toe is a pencil-and-paper game for two players, X and O, who take turns marking the spaces in a 3x3 grid. The X player usually goes first. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game.

## **INTRODUCTION**

Our Project name is Tic Tac Toe Mind Game. This game is very popular and is fairly simple by itself. It is actually a two player game. In this game, there is a board with  $n \times n$  squares. In our game, it is  $3 \times 3$  squares. The goal of Tic Tac Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally on a  $3 \times 3$  grid.

# OVERVIEW OF THE PROJECT

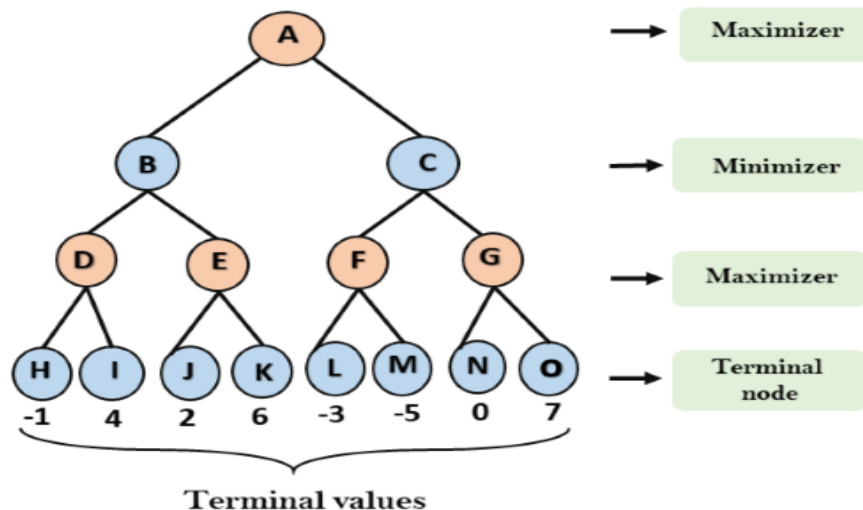
Tic-tac-toe is a two player game (one of them being your computer program). The two players take turns putting marks on a 3x3 board. The player who first gets 3 of his/her marks in a row (vertically, horizontally, or diagonally) *wins* the game, and the other loses the game.

Many people know how to win Tic Tac Toe and it's likely that your opponent does as well. For that reason, we'll teach you how to win a Tic Tac Toe game every time! The best Tic Tac Toe strategy is to place your Xs and Os in the center and in the corners of the 3x3 grid. From there, try to block your opponent from getting three Xs and Os in a row vertically, horizontally or diagonally.

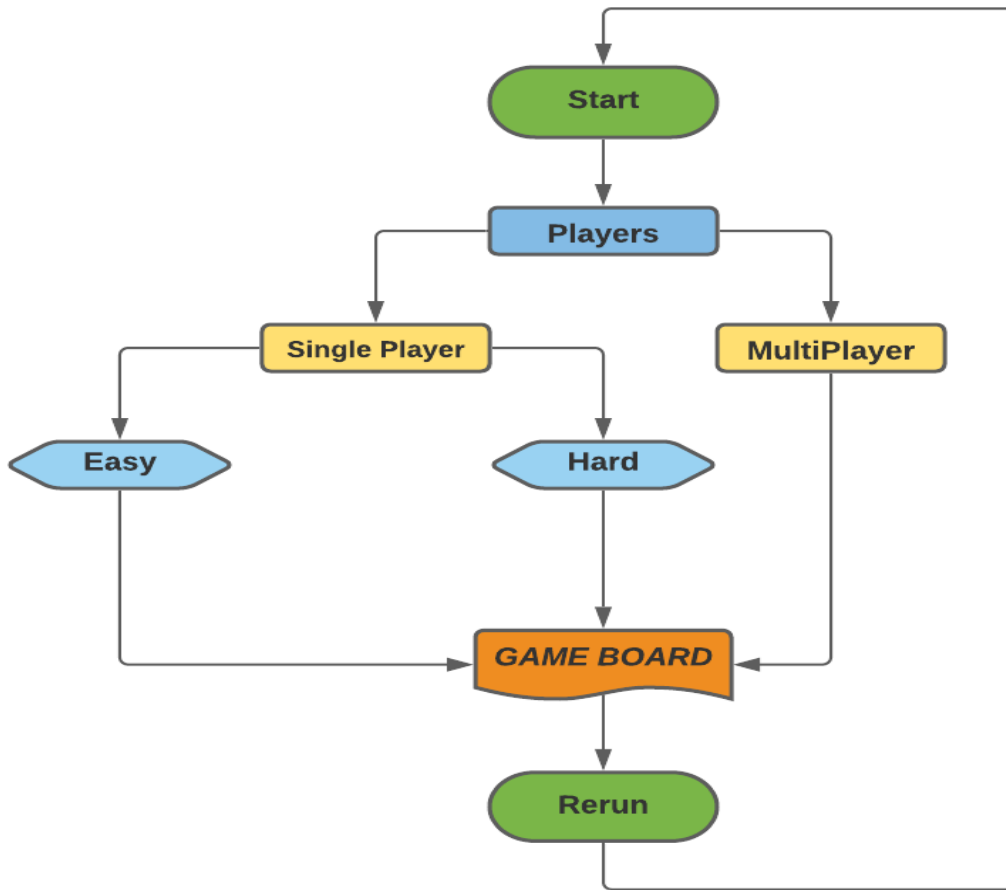
However, be careful - one wrong move will allow your opponent to win. If the game ends in a tie, don't be discouraged; most Tic Tac Toe games end this way, but a consistent strategy will ensure that you eventually win. Now that you know how to always win Tic Tac Toe, it's time to start playing a relaxing Tic Tac Toe game.

## HIGH LEVEL DESIGN

MINIMAX ALGORITHM LAYOUT :



## PROJECT DESIGN :



## TOOLS USED

- **FRAMEWORK - Swings**
- **Packages**
- **Exceptions**
- **Files**
- **Collections**
- **Inheritance**
- **Interface**
- **Strings**
- **Overriding**

# IMPLEMENTATION

## FRONT-END CODE :

```
public class frontEndUtil {
    int SIZE = 700;
    int BOARD_SIZE = 500;
    int CELL_SIZE = BOARD_SIZE / 3;

    Color BACKGROUND_COLOR = new Color(0x1c1c1c);
    Color CELL_COLOR = new Color(0x2a2a2a);
    Color O_COLOR = new Color(0xeded63);
    Color X_COLOR = new Color(0x63ebed);
    Color TEXT_COLOR = new Color(0xefefef);
    Color TEXT_COLOR_LITE = new Color(0x3e3e3e);
    Color WINNING_PAIRS = new Color(0x63ed9c);

    Font fontHelsky = FontRender.getHelskyStyle();
    Font fontHelsky2 = FontRender.getHelskyStyle2();
    Font fontNewsPaper = FontRender.getNewsPaperStyle();
    Font fontNewsPaper3 = FontRender.getNewsPaperStyle3();
    Font fontNewsPaper2 = FontRender.getNiconne();
    Font fontReMachine = FontRender.getRemachineStyle();

    Border borderCell =
    BorderFactory.createLineBorder(BACKGROUND_COLOR, 3);
}
```

## MINIMAX ALGORITHM :

```
public static MinMaxUtil miniMax(ArrayList<Character>
dummy, Boolean isMaximizer) {
    // noOfComparision += 1;

    // System.out.print(isMaximizer);        ////
    MinMaxUtil score = new MinMaxUtil(0, -1);
    if (checkWin(dummy, PLAYER).won) {
        score.setScore(-10);
        return score;
    } else if (checkWin(dummy, BOT).won) {
        score.setScore(20);
        return score;
    }

    ArrayList<Integer> emplList = emptyCells(dummy);
    if(emplList.size() == 0){
        return score;
    }

    int bestMove = -1;
    int bestScore = 0;

    if(isMaximizer){
        bestScore = Integer.MIN_VALUE;

        for (int i = 0; i < emplList.size(); i++) {
            dummy.set(emplList.get(i), BOT);
            MinMaxUtil sc = miniMax(dummy, false);
```



```

        if(sc.getScore() > bestScore){
            bestScore = sc.getScore();
            bestMove = empList.get(i);
        }
        dummy.set(empList.get(i), ' ');
    }
} else{
    bestScore = Integer.MAX_VALUE;

    for (int i = 0; i < empList.size(); i++) {
        dummy.set(empList.get(i), PLAYER);
        MinMaxUtil sc = miniMax(dummy, true);

        if(sc.getScore() < bestScore){
            bestScore = sc.getScore();
            bestMove = empList.get(i);
        }
        dummy.set(empList.get(i), ' ');
    }
}
// System.out.println();
return new MinMaxUtil(bestScore, bestMove);
}

```

#### TIC TAC TOE MAIN CODE :

```

tictactoe(Boolean SINGLEPLAYER, Boolean HARD) {
    this.SINGELPLAYER = SINGLEPLAYER;
    this.HARD = HARD;

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

```

```
frame.setLayout(null);
frame.setSize(SIZE, SIZE);
frame.setVisible(true);

frame.getContentPane().setBackground(BACKGROUND_COLOR);

HEADER.setBackground(BACKGROUND_COLOR);
HEADER.setBounds(0, 0, SIZE, 2 * (SIZE -
BOARD_SIZE) / 4);

TITLE.setAlignmentX(JLabel.CENTER_ALIGNMENT);
TITLE.setAlignmentY(JLabel.BOTTOM_ALIGNMENT);
TITLE.setForeground(TEXT_COLOR);

TITLE.setFont(fontHelsky);
HEADER.add("JLabel", TITLE);
frame.add(HEADER);

BODY.setBackground(BACKGROUND_COLOR);
BODY.setBounds(0, 2 * (SIZE - BOARD_SIZE) / 4, SIZE,
BOARD_SIZE);
BODY.setLayout(null);

BOARD.setBounds((SIZE - BOARD_SIZE) / 2, 0,
BOARD_SIZE, BOARD_SIZE);
BOARD.setBackground(BACKGROUND_COLOR);
BOARD.setLayout(new GridLayout(3, 3));

for (int i = 0; i < 9; i++) {
    origBoard.add(' ');
    buttons.add(new JButton());
    BOARD.add(buttons.get(i));
    buttons.get(i).setBorder(borderCell);
}
```

```

        buttons.get(i).setForeground(BACKGROUND_COLOR);
        buttons.get(i).setBackground(CELL_COLOR);
        buttons.get(i).setFocusable(false);
        buttons.get(i).setFont(fontNewsPaper);
        buttons.get(i).addActionListener(this);

    }
    BODY.add(BOARD);
    frame.add(BODY);

    JPanel FOOTER = new JPanel();
    FOOTER.setBackground(TEXT_COLOR);
    FOOTER.setBounds(0, SIZE - (SIZE - BOARD_SIZE),
SIZE, (SIZE - BOARD_SIZE) / 3);
    frame.add(FOOTER);
    //
    System.out.println(buttons[0].getText().equals(""));
    }

    public void displayWinningTripplets(int winningPairs[])
    {
        for (int j = 0; j < winningPairs.length; j++) {
            buttons.get(winningPairs[j]).setBackground(WINNING_PAIRS);
        }
    }

    public int bestSpot() {
        noOfComparision = 0;
        ArrayList<Character> object = new
ArrayList<Character>(origBoard);

        if (HARD) {

```

```

        return tictactoeAlgo.miniMax(object,
true).getIndex();
    } else {
        return tictactoeAlgo.EasyAlgo(object,
true).getIndex();
    }
}

public void declareWinner(String opString) {
    TITLE.setText(opString + " ");
    for (int i = 0; i < buttons.size(); i++) {
        buttons.get(i).setEnabled(false);
        buttons.get(i).setUI(new MetalButtonUI() {
            protected Color getDisabledTextColor() {
                return BACKGROUND_COLOR;
            }
        });
    }
    rerun.setBackground(X_COLOR);
    rerun.setFont(fontNewsPaper3);
    rerun.setFocusable(false);
    rerun.addActionListener(this);
    HEADDER.add(rerun);
}

public Boolean check(Character who) {
    wonUtil obj = tictactoeAlgo.checkWin(origBoard,
who);
    if (obj.won) {
        displayWinningTripplets(obj.winningTripplets);
        declareWinner(who + " won!!");
        return true;
    }
}

```

```

        if (tictactoeAlgo.checkTie(origBoard)) {
            declareWinner("Tie Game");
            return true;
        }
        return false;
    }

    public void playBot() {
        int pos = bestSpot();
        // System.out.println(pos);
        try {
            buttons.get(pos).setBackground(O_COLOR);

buttons.get(pos).setText(Character.toString(BOT));
            origBoard.set(pos, BOT);
        } catch (Exception e) {
            System.out.println(e);
        }
        XTurn = true;
        check(BOT);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        for (int i = 0; i < 9; i++) {
            if (e.getSource() == buttons.get(i)) {
                if (XTurn) {
                    if (buttons.get(i).getText() == "") {
                        //
buttons.get(i).setForeground(X_COLOR);

buttons.get(i).setBackground(X_COLOR);

```

```

buttons.get(i).setText(Character.toString(PPLAYER));
        origBoard.set(i, PPLAYER);
        XTurn = false;

        if (!check(PPLAYER) && SINGELPLAYER)
{
            playBot();
        }
    }
} else {
    if (buttons.get(i).getText() == "") {
        //
buttons.get(i).setForeground(O_COLOR);

buttons.get(i).setBackground(O_COLOR);

buttons.get(i).setText(Character.toString(BOT));
        origBoard.set(i, BOT);
        // System.out.println(origBoard);
        XTurn = true;
        check(BOT);
    }
}
}
if (e.getSource() == rerun) {
    frame.setVisible(false);
    players p = new players();
    p.showButtonDemo();
}
}
}

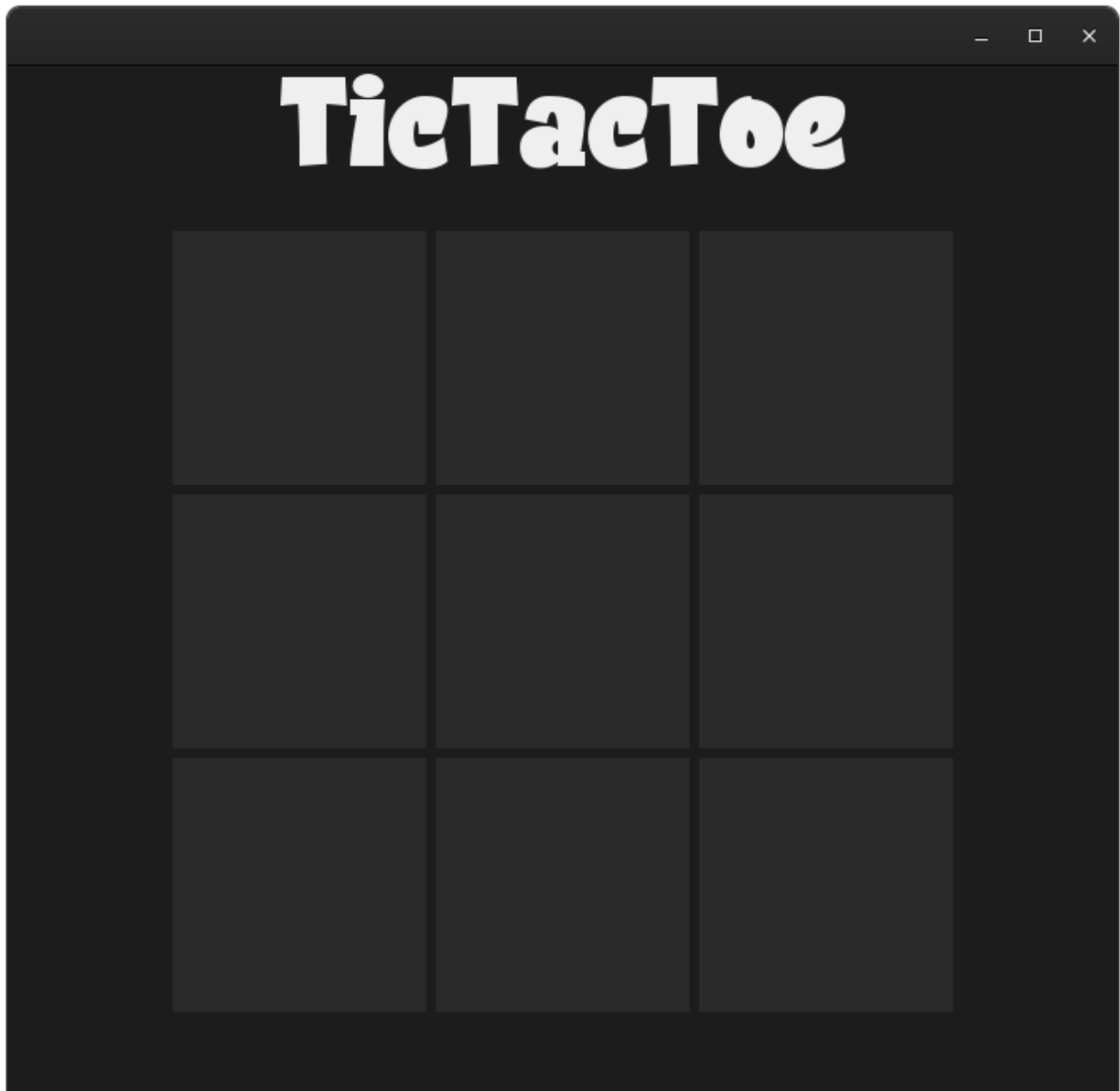
```

# SNAPSHOTS

## FRONT LOOK :

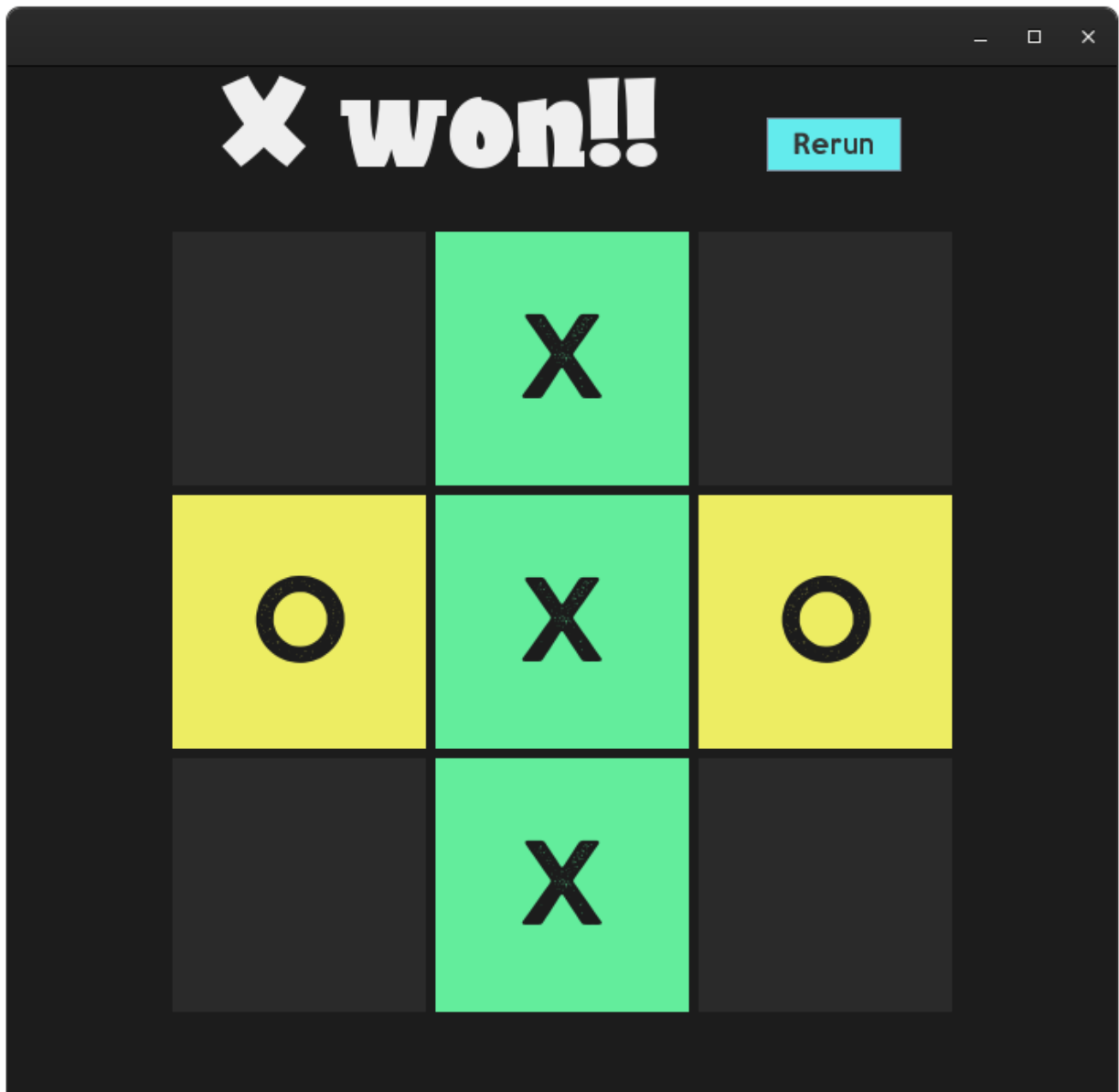


**GAME BOARD :**






**RESULT PAGE :**



## REFERENCES

- ❖ [YouTube Coding Challenge #149: Tic Tac Toe](#)
- ❖ [YouTube Coding Challenge 154: Tic Tac Toe AI with Minimax Algorithm](#)
- ❖ [YouTube Java tic tac toe game](#) 
- ❖ [YouTube Minimax Algorithm in Game Playing | Artificial Intelligence](#)
- ❖ [YouTube Introduction to Swing in Java | Free Java Course](#)
- ❖