C#: Tic-Tac-Toe

```
using System;
using System.Collections.Generic;
using static System.Console;


class Program
{
        static bool closeRequested = false;
        static bool matchOver = false;
        static bool playerTurn = true;
        static readonly Random random = new Random();
        static char[,] board;


        static void Main()
        {
                while (!closeRequested)
                {
                        board = new char[3, 3]
                        {
                                { ' ', ' ', ' ' },
                                { ' ', ' ', ' ' },
                                { ' ', ' ', ' ' },
                        };
                        matchOver = false;
                        while (!matchOver && !closeRequested)
                        {
                                if (playerTurn)
                                {
                                        #region Player Turn


                                        var (row, column) = (0, 0);
                                        bool moved = false;
```

```
                while (!moved && !matchOver && !closeRequested)
                {
                    Clear();
                    RenderBoard();
                    WriteLine();
                    WriteLine("Choose a valid position and
press enter.");
                    SetCursorPosition(column * 6 + 1, row * 4 +
1);
                    switch (ReadKey(true).Key)
                    {
                        case ConsoleKey.UpArrow:    row =
row <= 0 ? 2 : row - 1; break;
                        case ConsoleKey.DownArrow:  row
= row >= 2 ? 0 : row + 1; break;
                        case ConsoleKey.LeftArrow:
                        case ConsoleKey.RightArrow:
column = column <= 0 ? 2 : column - 1; break;
column = column >= 2 ? 0 : column + 1; break;
                        case ConsoleKey.Enter:
                            if (board[row, column] != '
')
                            {
                                break;
                            }
                            board[row, column] = 'X';
                            moved = true;
                            break;
                        case ConsoleKey.Escape:
                            Clear();
                            Write("Tic Tac Toe was
closed.");
                            closeRequested = true;
                            break;
                    }
                }

                #endregion
            }
            else
```

```
{
        #region Computer Move


        var possibleMoves = new List<(int X, int Y)>();
        for (int i = 0; i < 3; i++)
        {
                for (int j = 0; j < 3; j++)
                {
                        if (board[i, j] == ' ')
                        {
                                possibleMoves.Add((i, j));
                        }
                }
        }
        int index = random.Next(0, possibleMoves.Count);
        var (X, Y) = possibleMoves[index];
        board[X, Y] = 'O';


        #endregion
}
playerTurn = !playerTurn;


#region Check Board State


if (CheckForThree('X'))
{
        Clear();
        RenderBoard();
        WriteLine();
        Write("You Win.");
        matchOver = true;
}
else if (CheckForThree('O'))
{
```

```
                Clear();
                RenderBoard();
                WriteLine();
                Write("You Lose.");
                matchOver = true;
        }
        else if (CheckForFullBoard())
        {
                Clear();
                RenderBoard();
                WriteLine();
                Write("Draw.");
                matchOver = true;
        }

        #endregion
}

#region Play Again Check

if (!closeRequested)
{
        WriteLine();
        WriteLine("Play Again [enter], or quit [escape]?");
GetInput:
        switch (ReadKey(true).Key)
        {
                case ConsoleKey.Enter: break;
                case ConsoleKey.Escape:
                        closeRequested = true;
                        Clear();
                        break;
                default: goto GetInput;
        }
```

```
                }


                #endregion
            }
        }

        static bool CheckForThree(char c) =>
            board[0, 0] == c && board[1, 0] == c && board[2, 0] == c ||
            board[0, 1] == c && board[1, 1] == c && board[2, 1] == c ||
            board[0, 2] == c && board[1, 2] == c && board[2, 2] == c ||
            board[0, 0] == c && board[0, 1] == c && board[0, 2] == c ||
            board[1, 0] == c && board[1, 1] == c && board[1, 2] == c ||
            board[2, 0] == c && board[2, 1] == c && board[2, 2] == c ||
            board[0, 0] == c && board[1, 1] == c && board[2, 2] == c ||
            board[2, 0] == c && board[1, 1] == c && board[0, 2] == c;

        static bool CheckForFullBoard() =>
            board[0, 0] != ' ' && board[1, 0] != ' ' && board[2, 0] != ' ' &&
            board[0, 1] != ' ' && board[1, 1] != ' ' && board[2, 1] != ' ' &&
            board[0, 2] != ' ' && board[1, 2] != ' ' && board[2, 2] != ' ';

        static void RenderBoard()
        {
            WriteLine();
            WriteLine($" {board[0, 0]} ‖ {board[0, 1]} ‖ {board[0, 2]}");
            WriteLine("   ‖   ‖ ");
            WriteLine("═══╬═══╬═══");
            WriteLine("   ‖   ‖ ");
            WriteLine($" {board[1, 0]} ‖ {board[1, 1]} ‖ {board[1, 2]}");
            WriteLine("   ‖   ‖ ");
            WriteLine("═══╬═══╬═══");
            WriteLine("   ‖   ‖ ");
            WriteLine($" {board[2, 0]} ‖ {board[2, 1]} ‖ {board[2, 2]}");
        }
```

```
}
```

https://github.com/ZacharyPatten/dotnet-console-games/tree/master/Projects