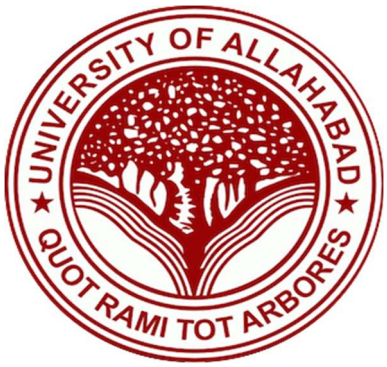
Tic Tac Toe Game



A project report
submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Science
in
Computer Science

By

AAFAQUE AHMAD ANSARI

B.Sc. III

Exam Roll No.: 2112001

Enrolment No.: U1810536

Under the Supervision of

Dr. ASHISH KHARE

Department of Electronics and Communication
J K Institute of Applied Physics & Technology
UNIVERSITY OF ALLAHABAD
Prayagraj – 211002, India

April, 2021

CANDIDATE'S DECLARATION

I, **Aafaque Ahmad Ansari,** hereby certify that the work, which is being presented in the report, entitled **Tic Tac Toe Game**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Science** and submitted to the institution is an authentic record of my own work carried out during the period *January-2021* to *April-2021* under the supervision of **Dr. Ashish Khare** at the Department of Electronics and Communication, University of Allahabad. The matter presented in this report has not been submitted elsewhere for the award of any other degree or diploma from any Institutions.

I declare that I have cited the reference about the text(s) /figure(s) from where they have been taken. I further declare that I have not willfully lifted up some other's work, para, text, data, results, etc. reported in the journals, books, magazines, reports, dissertations, theses, etc., or available at web-sites and included them in this report and cited as my own work.

Date: ……..

Signature of the Candidate

Aafaque Ahmad Ansari

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the **Mr. Aafaque Ahmad Ansari** has carried out this project/dissertation entitled **Tic Tac Toe Game** under my supervision.

Date: ……..

Signature of the Supervisor
Dr. Ashish Khare
(Supervisor)
Seal/Designation

ACKNOWLEDGEMENT

I would like to express my special thank of gratitude to my supervisor **Dr. Ashish Khare** and other teachers who gave me a golden opportunity to this wonderful project on the topic **"Tic Tac Toe Game"** which helped me in finalizing this project.

Secondly, I would like to thank to my parents, friends who helped me in completion in my project within a limited time frame.

Finally, I am grateful to almighty god for giving me the strength, patience and knowledge to complete my project.

Thank you
Aafaque Ahmad Ansari

ABSTRACT

The game TIC TAC TOE had an early variant which began in the first century in the Roman Empire. During that time, it was called Terni Lapilli where the players only had three pieces and had to move around the empty spaces to play. The actual game of TIC TAC TOE could be traced back to ancient Egypt. The game was also known during that time as "Three Men's Morris". The first reference to Noughts and crosses was made in 1864 in a British novel called can You Forgive Her. For many years the game was referred to as noughts and crosses but was changed in the 20th century by the United States to TIC TAC TOE.

The purpose of this documentation is to capture all the requirements by which the user can play a game of tic-tac-toe in Python as well as they can develop a logic that what is actually happening.

TABLE OF CONTENTS

Title Page No.

CANDIDATE'S DECLARATION ……...…2

CERTIFICATE FROM THE SUPERVISOR ……...3

ACKNOWLEDGEMENT …… 4

ABSTRACT.....5

LIST OF FIGURES 8

CHAPTER 1 INTRODUCTION

1.1 Background and Motivation 10
1.2 Objective.....10
1.3 Project Purpose……..……. 11
1.4 Tool Requirements……...11

CHAPTER 2 EXISTING WORKS

2.1 Introduction .....…….....12
2.1.1 Existing System……12
2.1.2 Proposed System……..12
2.2 Related Works 13

CHAPTER 3 SYSTEM, DESIGN AND ALGORITHMS

3.1 System Design …….....14
3.1.1 Input Design……..... 14
3.1.2 Output Design…… …… ……..……...…….…14
3.2 Methodology Used…….……15
3.3 Algorithms……..16
3.4 Implementation……..…..18

CHAPTER 4 TOOLS AND TECHNOLOGY

4.1 Introduction ……..20
4.1.1 Hardware Used…….....20
4.2.2 Software Used…….…20
4.2 Software Description…….....…….....21
4.2.1 Python 3.9.0 IDLE……21
4.2.2 Python 3……21

CHAPTER 5 RESULTS AND DISCUSSION

5.1 Introduction ……..22
5.2 Overview ……...……...23
5.2.1 Implementation…….23
5.2.2 Output……...23
5.3 Snapshots…….....24

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions.....…….28
6.3 Scope for Future Work..... …….28

REFERENCES ……...29

LIST OF FIGURES

Figure No. **Title** **Page No.**

1. Function to define computer move……..16
2. Function to define minimax algorithm…….17
3. Function to check game over state……...17
4. Function to define game board……18
5. Function to insert payer move……...18
6. Function to check for winning……..18
7. Function to check for draw……...18
8 Function to define player move……19
9. Print display to enter inputs……..19
10. Print result output of the game who wins…….……19

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

Tic-Tac-Toe also known as noughts and crosses is a paper and pencil game for two players, who take turns marking the spaces in a 3 x 3 grid traditionally. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game. It is a zero-sum of perfect information game. This means that it is deterministic, with fully observable environments in which two agents act alternately and the utility values at the end of the game are always equal and opposite. Because of the simplicity of tic-tac-toe, it is often used as pedagogical tool inartificial intelligence to deal with searching of game trees. The optimal move for this game can be gained by using minimax algorithm, where the opposition between the utility functions makes the situation adversarial, hence requiring adversarial search supported by minimax algorithm with alpha beta pruning concept in artificial intelligence.

1.2 OBJECTVE

One of the most universally played childhood games is TIC TAC TOE. An interactive TIC TAC TOE game is developed where two players will be able to play against each other in a suitable GUI by using proper mouse movements. This game will start by showing a simple display, prompt the user for a move and then prints the new board. The board looks like a large hash symbol (#) with nine slots that can each contain an X, and O, or a blank. There are two players, the X player and the O player. By default, player 1 (the O player) takes the initiative. The game will end in two situations: a win, when one player gets three in a row, horizontally, vertically or diagonally. A draw, when there are no remaining places to choose and neither of them has won.

Here are some rules for the game:

- The player with symbol 'O' goes first
- Players alternate placing X s and O s on the board until either:
  - One player has three in a row, horizontally, vertically or diagonally
  - All nine squares are filled.
- If all nine squares are filled and none of the players have three in a row, the game is a draw

1.3 PROJECT PURPOSE

1. To develop Artificial intelligence-based tic-tac-toe game for human Vs AI by implementing minimax algorithm with adversarial search concept.

2. To analyze the complexity of minimax algorithm through 3x3 tic tac toe game.

3. To study and implement alpha-beta pruning concept for improved speed of searching the optimal choice in tic-tac toe game.

4. To study optimizing methods for alpha-beta pruning using heuristic evaluation function.

1.4 TOOLS REQUIREMENT

1.4.1 HARDWARE REQUIREMENT

Minimum RAM: - IGB

Minimum Hard Disk: - 128GB

Processor: - Intel Pentium 4(1.50 GHz) or above

1.4.2 SOFTWARE REQUIREMENT

Operating System: - Support for both LINUX and WINDOWS users

Back End: - Python 3.9.0 Interpreter

Front End Language: - Python3

CHAPTER 2
EXISTING WORKS

2.1 INTRODUCTION

2.1.1 EXISTING SYSTEM

The Existing System is a simple game to play with paper and pencil between two people Here the whole process will be carried out in the hand-written format making nine square grids placing X's and O's and checking for the winner. This process will repeat every time. So, it will be a tedious job to draw a nine square grid every time with paper and pencil. The human effort is more here. Along with that the retrieval of the information is not easy as the records are maintained in the hand-written papers. This application requires correct feed on input into the respective field. Suppose the wrong inputs are entered then the whole process is to be done again So the users find it difficult to use.

2.1.2 PROPOSED SYSTEM

In this paper, we create a 3x3 tic-tac-toe game in Python. The system is designed so that two players can play a game of tic-tac-toe using Python IDLE software. The program will contain a display function and a select function to place the symbol as well as toggle between the symbols allowing each player a turn to play the game. The program will update after each player makes their move and will check for the conditions of the game as it goes on.

2.2 RELATED WORKS

We study several research papers on tic-tac-toe game and summarize the findings below.

Tic-Tac-Toe is a simple and yet an interesting board game. Researchers have used various approaches to study the Tic-Tac-Toe game.

Fok and Ong [3] and Grim et al. [4] have used artificial neural network based strategies to play it.

Citrenbaum [5] and Yakowitz [6] discuss games like Go-Moku, Hex and Bridg-It which share some similarities with Tic-Tac-Toe.

Many software implementations of Tic Tac game had been reported and recently it became available for smart phone such as the one for Apple iPhone[7], and other for Android environment[8].

Traditionally, the game of Tic-tac-toe is a pencil and paper game played by two people who take turn to place their pieces on a 3 times 3 grid with the objective of being the first player to fill a horizontal, vertical, or diagonal row with their pieces. What if instead of having one person playing against another, one person plays against a team of nine players, each of whom is responsible for one cell in the 3 times 3 grid? In this new way of playing the game, the team has to coordinate its players, who are acting independently based on their limited information.

Soedarmadji[13] present a solution that can be extended to the case where two such teams play against each other, and also to other board games. Essentially, the solution uses a decentralized decision making, which at first seems to complicate the solution.

Stephen Mann and Matthew Netsch[9] design of a parallel digital circuit that performs neural network (NN) calculations to evaluate Tic-Tac-Toe position was introduced. FPGA's are programmed to implement custom digital designs by physically mapping paths between the logic gates on each device. Using an FPGA allows the structure of the NN to be reprogrammed without any monetary cost. The author claims that NN implementation has better performance than traditional software implementations, because it takes advantage of the NN's inherent parallel structure.

Another NN application is implemented by Shahzeb Siddiqui et al [10] extend the game by adding two additional rows, two additional columns, and has been extended to the 3rd dimension. The paper calculate the optimum position using the idea of creating a neural network that uses backpropagation coupled with elements of a genetic algorithm to improve the likelihood that the most optimal solution is obtained and outline our methodology at the implementation level.

Pinaki Chakraborty[11] formally defines the Tic-Tac-Toe game and then develops artificial intelligence based strategies to play the same.

Leaw and Cheong[12] perform a minimalistic quantization of the classical game of tic-tac-toe, by allowing superpositions of classical moves.

An optically routed gate array (OPGA) is used by *Edward* [14] to implement a simple game of Tic-tact-toe to demonstrate the utility of electrooptical circuits which embody user input, display and logic functions in a single device. Progresses on routing techniques layout and logic simulation are also presented.

The hardware implementation of intelligent Tic-Tac toy is presented by Alauddin[15] .The implementation uses Graphical LCD (GLCD) touch screen and microcontroller. The microcontroller receives the player move from GLCD (displayed as X) and uses intelligent algorithm to analyze the move and choose the best counter move. The microcontroller displays the counter move on the screen as circle (O). The algorithm decides the winner when game is finished according to the Tic-Tac playing rule. The system is implemented using cheap available off the shelf electronic components and tested and proved to be working fast and efficiently

CHAPTER 3
SYSTEM, DESIGN AND ALGORITHMS

3.1 SYSTEM DESIGN

3.1.1 INPUT DESIGN

Input design is part of overall system design that requires special attention designing input data is to make the data entered easy and free from errors. The input forms are designed using the controls available in PYTHON INTERPRETER. Validation is made for each and every event that is happened. Help (how to play the game) information is provided for the users during when the players feel difficult.

Input design is the process of converting the user originated inputs to a computer-based format. A system user interacting through a workstation must be able to tell the system whether to accept the input to produce reports. The collection of input data is considered to be most expensive part of the system design. Since the input has to be planned in such a manner so as to get relevant information, extreme care is taken to obtain pertinent information.

3.1.2 OUTPUT DESIGN

Output design this application "TIC TAC TOE" generally refers to the results and information that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.
The output is designed in such a way that it is attractive, convenient and informative. Board is designed with various features, which make the console output more pleasing. As the outputs are the most important sources of information to the users, better design should improve the system's relationships with us and also will help in decision making. Form design elaborates the way output is presented and the layout available for capturing information.

One of the most important factors of the system is the output it produces. This system refers to the results and information generated. Basically, the output from a computer system is used to communicate the result of processing to the user.

3.2 METHODOLOGY USED

3.2.1 MINIMAX ALGORITHM

Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the opponent is also playing optimally. Its objective is to minimize the maximum loss. This algorithm is based on adversarial search technique. In a normal search problem, the optimal solution would be a sequence of actions leading to a goal state. Rather in adversarial search MAX finds the contingent strategy, which specifies the MAX's moves in the initial state, then MAX's moves in the states resulting from every possible response by MIN and continues till the termination condition comes alternately. Furthermore, given a choice, MAX prefers to move to a state of maximum value whereas MIN prefers a state of minimum value.

3.2.2 ALPHA-BETA PRUNING

The problem with minimax search is that the number of game states it has to examine is exponential in the depth of the tree. The minimax algorithm recursively calls itself until any one of the player wins or the board is full which takes a lot of computation time and makes it impossible to solve the 4X4 grid using standard minimax algorithm.
To solve this issue, we can use alpha-beta pruning algorithm which eliminates large parts of the tree from considerations by pruning the tree. When applied to a standard minimax tree, it returns the same move as minimax would, but it prunes away branches that cannot possibly influence the final decision. Basically, this algorithm applies the principle that there is no use expending search time to find out exactly how bad an alternative is if you have a better alternative. Alpha-beta pruning gets its name from the parameters that bound on the backed-up values that appears anywhere along the path:
α= the value of the best choices (i.e. highest value) so far at any choice point along the path for MAX

β= the value of the best choice (i.e. lowest value) so far at any choice point along the path for MIN\

3.3 ALGORITHMS

FINDING THE BEST MOVE:

We shall be introducing a new function called **compmove()**. This function evaluates all the available moves using **minimax()** and then returns the best move the maximizer can make. The pseudocode is as follows:

```
def compmove():
    bestscore = -800
    bestmove = 0
    for key in board.keys():
        if(board[key] == ' '):
            board[key] = bot
            score = minimax(board, 0, False)
            board[key] = ' '
            if(score > bestscore):
                bestscore = score
                bestmove = key

    insert(bot, bestmove)
    return
```

Fig.1

MINIMAX:

To check whether or not the current move is better than the best move we take the help of **minimax()** function which will consider all the possible ways the game can go and returns the best value for that move, assuming the opponent also plays optimally

The code for the maximizer and minimizer in the **minimax()** function is similar to **compmove()** , the only difference is, instead of returning a move, it will return a value. Here is the pseudocode:

```
def minimax(board, depth, ismax):
    if(checkmark(bot)):
        return 1
    elif(checkmark(player)):
        return -1
    elif(checkdraw()):
        return 0

    if(ismax):
        bestscore = -800
        for key in board.keys():
            if(board[key] == ' '):
                board[key] = bot
                score = minimax(board, depth + 1, False)
                board[key] = ' '
                if (score > bestscore):
                    bestscore = score
        return bestscore

    else:
         bestscore = 800
         for key in board.keys():
             if(board[key] == ' '):
                 board[key] = bot
                 score = minimax(board, depth + 1, True)
                 board[key] = ' '
                 if (score < bestscore):
                    bestscore = score
         return bestscore
```

Fig.2

CHECKING FOR GAMEOVER STATE :

To check whether the game is over and to make sure there are no moves left we use **emptyspace()** function. It is a simple straightforward function which checks whether a move is available or not and returns true or false respectively. Pseudocode is as follows:

```
def emptyspace(position):
    if board[position] == ' ':
        return True
    else:
        return False
```

Fig.3

3.4 IMPLEMENTATION

PRINT BOARD:

```
def printboard(board):
    print(board[1] + '|' + board[2] + '|' + board[3])
    print('-+-+-')
    print(board[4] + '|' + board[5] + '|' + board[6])
    print('-+-+-')
    print(board[7] + '|' + board[8] + '|' + board[9])
    print("\n")
```

Fig.4

INSERT MOVE:

```
def insert(letter, position):
    if emptyspace(position):
        board[position] = letter
        printboard(board)
```

Fig.5

CHECK FOR WINNING:

```
def checkwin():
    if(board[1] == board[2] and board[1] == board[3] and board[1] != ' '):
        return True
    elif(board[4] == board[5] and board[4] == board[6] and board[4] != ' '):
        return True
    elif(board[7] == board[8] and board[7] == board[9] and board[7] != ' '):
        return True
    elif(board[1] == board[4] and board[1] == board[7] and board[1] != ' '):
        return True
    elif(board[2] == board[5] and board[2] == board[8] and board[2] != ' '):
        return True
    elif(board[3] == board[6] and board[3] == board[9] and board[3] != ' '):
        return True
    elif(board[1] == board[5] and board[1] == board[9] and board[1] != ' '):
        return True
    elif(board[7] == board[5] and board[7] == board[3] and board[7] != ' '):
        return True
    else:
        return False
```

Fig.6

CHECK FOR DRAW:

```
def checkdraw():
    for key in board.keys():
        if(board[key] == ' '):
            return False
    return True
```

Fig.7

PLAYER MOVE:

```
def playermove():
    position = int(input("Enter the position: "))
    insert(player, position)
    return
```

Fig.8

DISPLAY:

```
printboard(board)
print("Positions are as Follows: ")
print("1, 2, 3 ")
print("4, 5, 6 ")
print("7, 8, 9 ")
print("\n")
print("player = X ")
print("bot = O ")
print("\n")
print("Wait for computer take Time ")
```

Fig.9

RESULT:

```
        if(checkdraw()):
            print("Draw!")
            exit()

        if checkwin():
            if letter == 'O':
                print("Computer Wins!")
                exit()

            else:
                print("Player Wins!")
                exit()
```

Fig.10

CHAPTER 4
TOOLS AND TECHNOLOGY USED

5.1 INTRODUCTION

5.1.1 HARDWARE USED

RAM: 4 GB

Hard Disk: 500 GB

Processor: AMD A6-7610 APU

5.1.2 SOFTWARE USED

Operating System: WINDOWS 10

Back End: Python 3.9.0 Interpreter

Front End Language: Python 3

5.2 SOFTWARE DESCRIPTION

5.2.1 PYTHON 3.9.0 IDLE

IDLE stands for Integrated Development and Learning Environment is an integrated development environment for python. It has been bundled with the default implementation of the language since 1.5.2b 1. it is packaged as an optional part of the python packaging with many Linux distribution. It is completely written in python3 and Tkinter GUI toolkit. IDLE is intended to be a simple IDE and suitable for beginners as well as advanced users. To that end it is cross platform and avoids feature clutter The features provided by the IDLE includes

- Python shell with syntax highlighting
- Integrated debugger with stepping, persistent breakpoints and call stack visibility
- Multi-window text editor with syntax highlighting, auto completion, smart indenting etc

5.2.2 PYTHON 3

Python is a general purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido Van Rossum during 1985-1990 Like Peri, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV show called "Monty Python's Flying Circus".

Python 3.0(also called "Python 3000 or Py3K") was released in December 3, 2008. The latest version of python accumulated new and redundant ways to program the same task, Python 3.x had an emphasis on removing duplicative constructs and modules, in keeping with "There should be one and preferable only one - obvious way to do it" Python's dynamic typing encourage the programmer to write a code that is clear well structured as well as easy to understand.

The features of dynamic typing are:

- Types are bound to values but not to variables
- Function and method lookup is done at runtime
- Values are inspect-able
- There is an interactive interpreter, more than one, in fact.
- You can list the methods supported by any given object

Because code is automatically compiled to byte code and executed Python is suitable for use as a scripting language Web application implementation language etc Because of its strong structuring constructs (nested code blocks, functions, classes modules and packages) and its consistent use of objects and OOP, Python enables you to write clear and logical code for small and large projects.

CHAPTER 5
RESULTS AND DISCUSSION

5.1 INTRODUCTION

The minimax algorithm performs a complete depth-first exploration of the game tree. It recursively calls itself until a winning state of any agent is found or the grid is full. If the maximum depth of the tree is m and there are b legal moves at each point, then the time complexity of the minimax algorithm is O (bm). The space complexity is O(bm) for generating all actions at once. For 3X3 board, the possible number of moves is (3 * 3)! = 9! If the board size is increased to 4X4 there is exponential growth and we experience an explosion of computational time. This means we now get 16! Possible states which is an incredibly large amount. So, we can estimate that computational time will be million times higher than 3X3 board. Hence, the time cost is totally impractical, making this algorithm infeasible in 4X4 board, but it serves as the basis for the mathematical analysis of games and for more practical algorithm.

To implement the minimax algorithm for 3x3 board we have applied the concept of alpha beta pruning for eliminating the solutions which would not make impact on the final decision. Considering above parameters the time complexity or alpha-beta pruning is O (bm/2) in best case i.e. depth is reduced to half in the best possible scenario. Even though there is such huge reduction in time, the time complexity is still very large considering our board. So, to decrease the computation time, the search can be limited to certain level i.e. reduce the depth of search tree. The maximum depth is taken as 9 in our system. But when reducing the depth of search tree, the output is not optimal as we have cut-off 9 levels of our search tree which is a very large number of nodes. Hence, we have used heuristic evaluation function that approximates the true utility of a state without doing a complete search. For this problem, the heuristic function used is given by:

$E(n) = M(n) – O(n)$

where, M(n) is total of possible winning path of AI

O(n) is total of possible winning path of human player

E(n) is total evaluation for state n

This heuristic gives positive value if AI i.e. maximizing agent has more chance or winning and negative value if the human player i.e. minimizing agent has more chance of winning.

5.3 OVERVIEW

5.3.1 IMPLEMENTATION

- After initializing this game, a 3x3 a hash shape square board grid will pop up in the screen of the user.
- The game will be played between Computer & the human player.
- Player is ‘X’ and Computer is ‘O’, It is interchangeable, Computer starts.
- The player will have to input a numerical character, from 1 to 9, to select a position for X or O into the space they For example: if they are playing with O and input 2, the O will go to the first row – the second column. If the player wants to place O in the third row – first column, then they have to enter 7. And, it is similar for the other positions.
- The game starts with on of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character (‘O’ or ‘X’).
- If the blank spaces in the grid are all filled, and there is no winner, then the game is said to be a draw.

5.3.2 OUTPUT

```
>>>
================ RESTART: C:\Users\HP\Desktop\tic tac toe.py ================
 | |
-+-+-
 | |
-+-+-
 | |


Positions are as Follows:
1, 2, 3
4, 5, 6
7, 8, 9


player = X
bot = O


Wait for computer take Time
O| |
-+-+-
 | |
-+-+-
 | |


Enter the position:
```

5.2 SNAPSHOTS

```
================ RESTART: C:\Users\HP\Desktop\tic tac toe.py ================
 | |
-+-+-
 | |
-+-+-
 | |


Positions are as Follows:
1, 2, 3
4, 5, 6
7, 8, 9


Wait for computer take Time
O| |
-+-+-
 | |
-+-+-
 | |


Enter the position: 3
O| |X
-+-+-
 | |
-+-+-
 | |


O|O|X
-+-+-
 | |
-+-+-
 | |


Enter the position: 5
O|O|X
-+-+-
 |X|
-+-+-
 | |


O|O|X
-+-+-
O|X|
-+-+-
 | |


Enter the position: 7
O|O|X
-+-+-
O|X|
-+-+-
X| |


Player Wins!
>>>
```

```
================ RESTART: C:\Users\HP\Desktop\tic tac toe.py ================
 | |
-+-+-
 | |
-+-+-
 | |


Positions are as Follows:
1, 2, 3
4, 5, 6
7, 8, 9


Wait for computer take Time
O| |
-+-+-
 | |
-+-+-
 | |


Enter the position: 5
O| |
-+-+-
 |X|
-+-+-
 | |


O|O|
-+-+-
 |X|
-+-+-
 | |



Enter the position: 3
O|O|X
-+-+-
 |X|
-+-+-
 | |


O|O|X
-+-+-
O|X|
-+-+-
 | |


Enter the position: 6
O|O|X
-+-+-
O|X|X
-+-+-
 | |


O|O|X
-+-+-
O|X|X
-+-+-
O| |


Computer Wins!
>>>
```

```
================ RESTART: C:\Users\HP\Desktop\tic tac toe.py ================
 | |
-+-+-
 | |
-+-+-
 | |


Positions are as Follows:
1, 2, 3
4, 5, 6
7, 8, 9


Wait for computer take Time
O| |
-+-+-
 | |
-+-+-
 | |


Enter the position: 4
O| |
-+-+-
X| |
-+-+-
 | |


O|O|
-+-+-
X| |
-+-+-
 | |
```

```
Enter the position: 3
O|O|X
-+-+-
X| |
-+-+-
 | |


O|O|X
-+-+-
X|O|
-+-+-
 | |


Enter the position: 9
O|O|X
-+-+-
X|O|
-+-+-
 | |X


O|O|X
-+-+-
X|O|O
-+-+-
 | |X


Enter the position: 8
O|O|X
-+-+-
X|O|O
-+-+-
 |X|X



O|O|X
-+-+-
X|O|O
-+-+-
O|X|X


Draw!
>>>
```

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

With the basis of minimax algorithm for mathematical analysis alongside speeding up the computation by alpha beta pruning concept and optimizing the utility function using heuristic function, the 3x3 tic tac toe game was developed. We explored that a 3x3 tic tac toe, an adversary search technique game in artificial intelligence, can be developed using these techniques. Increasing the size of this game would create a huge time complexity issue with the same algorithm and techniques, for which other logics must be further researched.

6.2 SCOPE AND FUTURE WORK

- We can use advanced algorithms like Minimax, it is a very powerful and universal algorithm that can be applied in a wide variety of applications.
- We can use more algorithms like Alpha-Beta Pruning using heuristic evaluation function to make our AI better.
- Thus, we can make an unbeatable Tic Tac Toe AI Game.
- Maybe we can make a Self Player Automatic Game in which we don't have to provide any input using concept of AI and Machine Learning.

REFERENCES

BOOK AND WEBSITES

- Automate the Boring Stuff with Python, AL SWEIGART, no starch press
- San Francisco.
- Article by AKSHAY L. ARADHYA on Minimax Algorithm in Game Theory.
- Alpha-Beta Pruning by CARL FELSTINER.
- ELLIS HOROWITZ AND SARTAJ SAHNI, Computer Algorithms, Computer Science Press, New York.
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
- https://tutorialspoint.dev/algorithm/game-theory/minimax-algorithm-in-game-theory-set-2-evaluation-function/

REFERENCES

[3] S. C. Fok and E. K. Ong. A High School Project on Artificial Intelligence in Robotics, Artificial Intelligence in Engineering, Vol. 10, No. 1, 1996, pp. 61-70.

[4] J. Grim, P. Somol and P. Pudil. Probabilistic Neural Network Playing and Learning Tic-Tac-Toe, Pattern Recognition Letters, Vol.26, No. 12, 2005, pp. 1866-1873.

[5] R. L. Citrenbaum. Strategic Pattern Generation: A Solution Technique for a Class of Games, Pattern Recognition, Vol. 4, No. 3, 1972, pp. 317-329.

[6] S. Yakowitz. A Statistical Foundation for Machine Learning, with Application to Go- Moku, Computers and Mathematics with Applications, Vol. 17, No. 7, 1989, pp. 1095- 1102

[7] iPhone Simple Tic-Tac-Toe Implementation, http://www.vworker.com/RentACoder/misc BidRequests/ ShowBidRequest.asp?lngBidRequestId=1622905

[8] Tic Tac for Android, http://www.androidappshome.com/tic-tac-toe-free-android-57.html.

[9] Stephen Mann and Matthew Netsch, "A parallel Embedded Neural Network for an Intelligent Turn-Based Engine", http://www.samduffysinger.pwp.blueyonder.co. uk/ Project%20Outline%20TicTacToe.pdf

[10] Shahzeb Siddiqui , Francis Mutuc and Nicholas Schmidt, " Designing a 5x5x5 Tic-Tac-Toe Game using a Neural Network with Backpropagation with a Twist", http://www.personal.psu.edu/fcm5007/eportfolio/AINeural Networks.pdf

[11] Pinaki Chakraborty, Artificial Intelligence Based Strategies to Play the Tic -Tac-Toe Game, Journal of Technology and Engineering Sciences, Vol 1, No. 1 January –June 2009
[12] J N Leaw and S A Cheong, Strategic insights from playing quantum tic-tac-toe, Journal of Physics A: Mathematical and Theoretical Volume 43 Number 45, 2010
[13] Soedarmadji, Decentralized Decision Making in the Game of Tic-tac-toe, IEEE Symposium on Computational Intelligence and Games, May 2006
[14] Edward P. Vogel, Tic tac toe game using an optically routed gate array, Proc. SPIE 2863, Current Developments in Optical Design and Engineering VI, 407, Nov 1, 1996
[15] Alauddin Al-Omary, Machine- Human Tic-Tac game based on Microcontroller Technology, International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 02– Issue 05, September 2013