

PROJECT REPORT
On
“TIC TAC TOE”

Submitted By
Pranav Lokhande

Guided By:-
Mr. Ratnesh K. Choudhary



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**S. B. JAIN INSTITUTE OF TECHNOLOGY
MANAGEMENT AND RESEARCH, NAGPUR.**

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

2021-2022

© S.B.J.I.T.M.R Nagpur 2022

**S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND
RESEARCH, NAGPUR**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SESSION 2021-2022

CERTIFICATE

This is to certify that the Project titled “**Tic tac toe**” is a bonafide work of **Pranav Anand Lokhande** carried out for the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in **Computer Science & Engineering**.

Mr. Ratnesh K. Choudhary

Assistant Professor

Mr. Animesh Tayal

Head of Department

INDEX

CERTIFICATE	I
INDEX	Ii
LIST OF FIGURES	Iii
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 METHODOLOGY	2
CHAPTER 3 TOOLS/PLATFORMS	3
CHAPTER 4 DESIGN & IMPLEMENTATION	4-8
4.1 ALGORITHM	
4.2 FLOWCHART	
4.3 SOURCE CODE	
CHAPTER 5 RESULT & DISCUSSION	9-10
5.1 OUTPUT	
5.2 DISCUSSION	
5.3 APPLICATION	
CHAPTER 6 CONCLUSION	11
REFERENCES	12

LIST OF FIGURE

FIG. NO.	TITLE OF FIGURE	PAGE NO.
1.1.1	Tic tac toe	1
4.1.1	layout	4
4.2.1	Tic tac toe game flowchart	5
5.1.1	Output Screenshot	11

CHAPTER 1

INTRODUCTION

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 in artificial 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.

X	O	X
	X	O
O	O	X

Fig. 1.1 Tic tac toe

CHAPTER 2

METHODOLOGY

- Calls `create_board()` to create a 9×9 board and initializes with 0.
- For each player (1 or 2), calls the `random_place()` function to randomly choose a location on board and mark that location with the player number, alternatively.
- Print the board after each move.
- Evaluate the board after each move to check whether a row or column or a diagonal has the same player number. If so, displays the winner name. If after 9 moves, there are no winner then displays -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.
- ***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 agent 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 appear

CHAPTER 3

TOOLS/PLATFORMS

Software Requirement

- a. **Language:** Python3
- b. **IDE:** VS Code
- c. **Libraries:** pygame, sys ,random.
- d. **Operating system :**Windows 10

1. Python3:

Python 3 is a new version of the Python programming language which was released in December 2008. This version was mainly released to fix problems that exist in Python 2. The nature of these changes is such that Python 3 was incompatible with Python. It is backward incompatible.

2. Visual Studio Code :

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

3. Libraries :

The pygame library is an open-source module for the Python programming language specifically intended to help you make games and other multimedia applications. Built on top of the highly portable SDL (Simple Direct Media Layer) development library, pygame can run across many platforms and operating systems.

4. Sys:

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.

5. Random :

To get access to the random module, we add `from random import *` to the top of our program (or type it into the python shell). Open the file `randOps.py` in vim, and run the program in a separate terminal. Note if you run the program again, you get different (random) results.

6. Operating system :

Windows 10 is a major release of Microsoft's Windows NT operating system. It is the direct successor to Windows 8.1, which was released nearly two years earlier. It was released to manufacturing on July 15, 2015, and later to retail on July 29, 2015

CHAPTER 4

DESIGN & IMPLEMENTATION

4.1 ALGORITHM

- In this project when it run the game is started.
- Its ask for who is first and who is second, and also ask for to select the symbol i.e [X/O].
- After selecting the player 1 and player 2 symbol. Game is started.
- At begging it's ask to enter the number in between 1-9.
- It's in the form of 1-9 box . as shown in figure

1	2	3
4	5	6
7	8	9

Fig. 4.1.1 layout

- If the no is in between 1-9 then it show in the box by respective symbol of player [X/O].
- After 9 step ,if the 3 consecutive same symbol comes, then the game will end ,and respective player win the game.
- If the 3 consecutive symbol is not came then the game will tie, and game will exit.
- If the number is not in between 1-9,then it's show enter the number between 1-9.
- If we repeat the number then it's show the number is already entered.

4.2 FLOWCHART

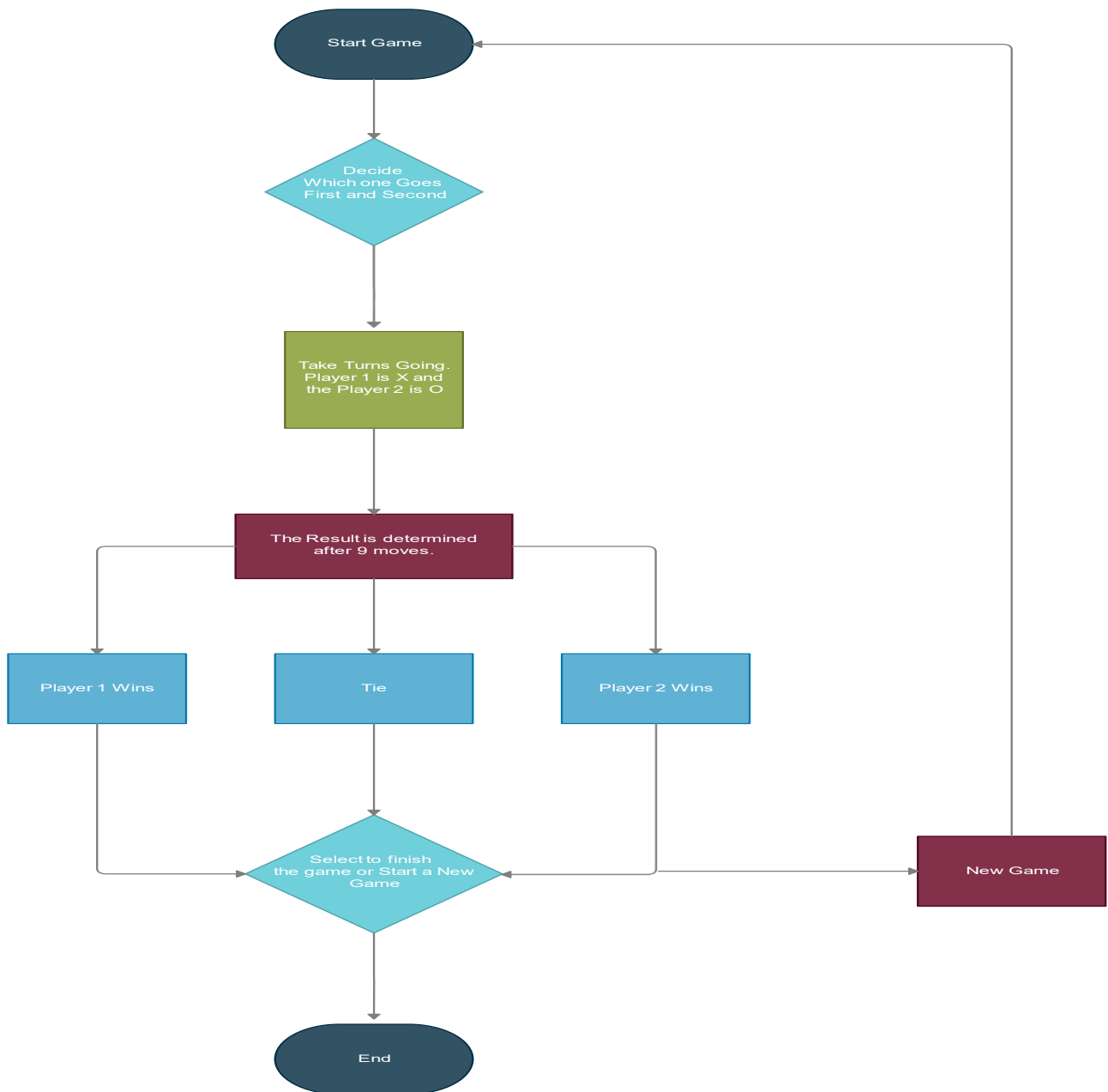


Fig. 4.2.1 Tic tac toe game flowchart

4.3 SOURCE CODE

```
board = [  
    ["-", "-", "-"],  
    ["-", "-", "-"],  
    ["-", "-", "-"]  
]  
  
user = True # when true it refers to x, otherwise o
```

```

turns = 0

def print_board(board):
    for row in board:
        for slot in row:
            print(f"{slot} ", end="")
        print()

def quit(user_input):
    if user_input.lower() == "q":
        print("Thanks for playing")
        return True
    else: return False

def check_input(user_input):
    # check if its a number
    if not isnum(user_input): return False
    user_input = int(user_input)
    # check if its 1 - 9
    if not bounds(user_input): return False
    return True

def isnum(user_input):
    if not user_input.isnumeric():
        print("This is not a valid number")
        return False
    else: return True

def bounds(user_input):
    if user_input > 9 or user_input < 1:
        print("This is number is out of bounds")
        return False
    else: return True

def istaken(coords, board):
    row = coords[0]
    col = coords[1]
    if board[row][col] != "-":
        print("This position is already taken.")
        return True
    else: return False

```

```

def coordinates(user_input):
    row = int(user_input / 3)
    col = user_input
    if col > 2: col = int(col % 3)
    return (row,col)

def add_to_board(coords, board, active_user):
    row = coords[0]
    col = coords[1]
    board[row][col] = active_user

def current_user(user):
    if user: return "x"
    else: return "o"

def iswin(user, board):
    if check_row(user, board): return True
    if check_col(user, board): return True
    if check_diag(user, board): return True
    return False

def check_row(user, board):
    for row in board:
        complete_row = True
        for slot in row:
            if slot != user:
                complete_row = False
                break
        if complete_row: return True
    return False

def check_col(user, board):
    for col in range(3):
        complete_col = True
        for row in range(3):
            if board[row][col] != user:
                complete_col = False
                break
        if complete_col: return True
    return False

```

```

def check_diag(user, board):
    #top left to bottom right
    if board[0][0] == user and board[1][1] == user and board[2][2] == user: return True
    elif board[0][2] == user and board[1][1] == user and board[2][0] == user: return True
    else: return False

while turns < 9:
    active_user = current_user(user)
    print_board(board)
    user_input = input("Please enter a position 1 through 9 or enter \"q\" to quit:")
    if quit(user_input): break
    if not check_input(user_input):
        print("Please try again.")
        continue
    user_input = int(user_input) - 1
    coords = coordinates(user_input)
    if istaken(coords, board):
        print("Please try again.")
        continue
    add_to_board(coords, board, active_user)
    if iswin(active_user, board):
        print(f"{active_user.upper()} won!")
        break
    turns += 1
    if turns == 9: print("Tie!")
    user = not user

```

CHAPTER 5

RESULT & DISCUSSION

5.1 OUTPUT

```
PS C:\Users\user\Downloads\rock> &
C:/Users/user/AppData/Local/Programs/Python/Python310/python.exe
c:/Users/user/Desktop/python/game.py

- - -
- - -
- - -
Please enter a position 1 through 9 or enter "q" to quit:5

- - -
- x -
- - -
Please enter a position 1 through 9 or enter "q" to quit:1

o - -
- x -
- - -
Please enter a position 1 through 9 or enter "q" to quit:9

o - -
- x -
- - x
Please enter a position 1 through 9 or enter "q" to quit:6

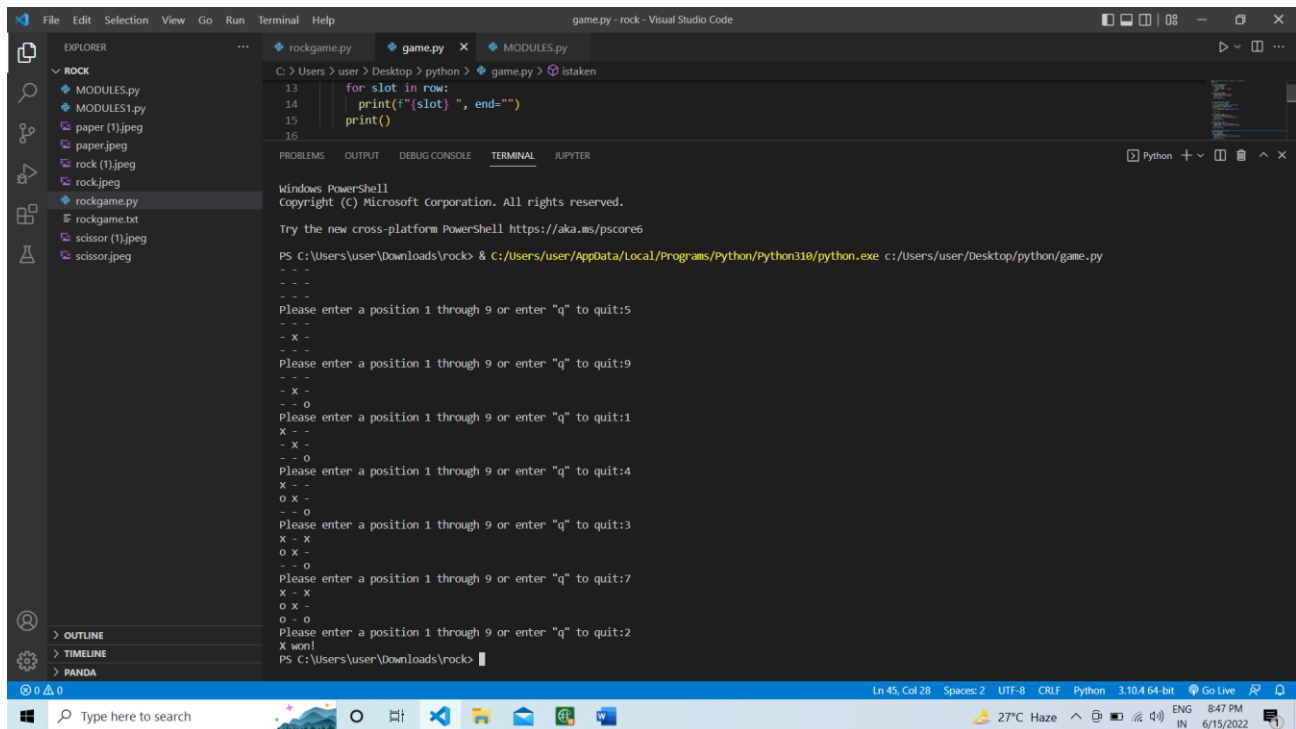
o - -
- x o
- - x
Please enter a position 1 through 9 or enter "q" to quit:7

o - -
- x o
x - x
Please enter a position 1 through 9 or enter "q" to quit:3

o - o
- x o
x - x
Please enter a position 1 through 9 or enter "q" to quit:8
```

X won!

PS C:\Users\user\Downloads\rock>



```
C:\Users\user\Desktop> python > game.py > istaken
13     for slot in row:
14         print(f"{slot} ", end="")
15     print()
16
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\user\Downloads\rock> & C:/Users/user/AppData/Local/Programs/Python/Python310/python.exe c:/Users/user/Desktop/python/game.py
- - -
Please enter a position 1 through 9 or enter "q" to quit:5
- - -
- X -
- - -
Please enter a position 1 through 9 or enter "q" to quit:9
- - -
- X -
- - -
Please enter a position 1 through 9 or enter "q" to quit:1
X - -
- X -
- - -
Please enter a position 1 through 9 or enter "q" to quit:4
X - -
O X -
- - -
Please enter a position 1 through 9 or enter "q" to quit:3
X - X
O X -
- - -
Please enter a position 1 through 9 or enter "q" to quit:7
X - X
O X -
O - O
Please enter a position 1 through 9 or enter "q" to quit:2
X won!
PS C:\Users\user\Downloads\rock>
```

Fig. 5.1.1 Output screenshot

5.2 DISCUSSION

1. Traditionally the first player plays with "X". So you can decide who wants to go with "X" and who wants to go with "O".
2. Only one player can play at a time.
3. If any of the players have filled a square then the other player and the same player cannot override that square.
4. There are only two conditions that may match will be draw or may win.
5. The player that succeeds in placing three respective marks (X or O) in a horizontal, vertical, or diagonal row wins the game.

Winning condition

Whoever places three respective marks (X or O) horizontally, vertically, or diagonally will be the winner.

5.3 APPLICATION

1. Tic tac toe is a software game which increase the used of brain ,by this brain thing more about strategies to win the game.
2. It help in increasing the focus by playing the game ,it is very easy to play .and we learn more thing in this game.

CHAPTER 6

CONCLUSION

It in project we created tic tac toe game which contain two player interface.

In this game we learn many new thing .its increase our knowledge by this we used may python module in this game . The Tic Tac Toe game is most familiar among all the age groups. Intelligence can be a property of any purpose-driven decision maker. This basic idea has been suggested many times. An algorithm of playing Tic Tac Toe has been presented and tested that works in efficient way. With the basis of minimax algorithm for mathematical analysis alongside speeding up t computation by alpha beta pruning concept and optimizing the utility function using heuristic function, the 4x4 tic tac toe game was developed. We explored that a 4x4 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

REFERENCE

- [1] K. Kask. [Online]. Available: <https://www.ics.uci.edu/~kkask/Fall-2016%20CS271/slides/04-games.pdf>. [Accessed 02 01 2020].
- [2] G. Surma. [Online]. Available: <https://towardsdatascience.com/tic-tac-toe-creating-unbeatable-ai-with-minimax-algorithm-8af9e52c1e7d>. [Accessed 20 12 2019].
- [3] P. G. ., P. S. P. Sunil Karamchandani, "A Simple Algorithm For Designing An Artificial Intelligence Based Tic Tac Toe Game".
- [4] 12 09 2019. [Online]. Available: <https://www.edureka.co/blog/alpha-beta-pruning-in-ai>. [Accessed 20 11 2019]
- [5] <https://geekflare.com/tic-tac-toe-python-code/>
- [6] <https://www.c-sharpcorner.com/UploadFile/75a48f/tic-tac-toe-game-in-pytho>

