

A
Project Report
On
“TIC TAC TOE GAME”

Of
Bachelor Of Engineering

In
Computer Science And Engineering(CSE)
Submitted By:

1. RITIKA GUPTA(CO21350)
2. YASHVI SHARMA(CO21370)

Under the supervision of
Dr. Sarabjeet Singh

ACKNOWLEDGEMENTS

We thanks Dr. Sarabjeet Singh (Asst. Professor) who have been the great inspiration and who have provided sufficient background knowledge and understanding of this subject. We feel privileged to extend our deep sense of gratitude to our parents for their support and encouragement. We came to know about so many new things I am really thankful to them. Secondly, I would like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Tic-Tac-Toe

Tic-tac-toe is not a very challenging game for human beings. If we're an enthusiast, we've probably move from the basic to some variant like three dimensional tic-tac-toe on a larger grid. If we sit down right now to play ordinary three-by-three tic-tac-toe with a friend, what will probably happen is that every game will come out a tie. Both i and my friend can probably play perfectly, never making a mistake that would allow your opponent to win. But can you describe how you know where to move each turn? Most of the time we probably aren't even aware of alternative possibilities; we just look at the board and instantly know where you want to move. That kind of instant knowledges great for human beings because it makes us a fast player. But it isn't much help in writing a computer program. For that, we have to know very explicitly what our strategy is.

THEORY OF GAME

Tic tac toe game has 16 cells for a 4x4 grid. The two players with their respective marks as 'X' and 'O' are required to place their marks in their turns one by one. Once the cell is occupied by a mark it cannot be used again. The game is won if the agent is able to make a row or column or a diagonal occupied completely with their respective marks. The game terminates once the winning situation is gained or the cells are fully occupied. Given a 4x4 grid, the agents has to find the optimal cell to fill with respective marks. A row is completely filled by a mark 'X' or 'O'. A diagonal is completely filled by a mark 'X' or 'O'. A column is completely filled by a mark 'X' or 'O'.

Source Code

```
//Tic Tac Toe Game in C++
//Importing the inbuild libraries in CPP
#include <iostream>#include <stdlib.h>

using namespace std;

//Array for the boardchar
board[3][3] = {{'1','2','3'},{'4','5','6'},{'7','8','9'}};

//Variable Declaration
int choice;

int row,column;

char turn = 'X';

bool draw = false;

//Function to show the current status of the gaming board
void display_board(){

//Rander Game Board LAYOUT
cout<<"PLAYER - 1 [X]t PLAYER - 2 [O]nn";

cout<<"tt  |  |  n";

cout<<"tt "<<board[0][0]<<" | "<<board[0][1]<<" | "<<board[0][2]<<" n";

cout<<"tt__|__n"; cout<<"tt  |  |  n";

cout<<"tt "<<board[1][0]<<" | "<<board[1][1]<<" | "<<board[1][2]<<" n";

cout<<"tt__|__n"; cout<<"tt  |  |  n";

cout<<"tt "<<board[2][0]<<" | "<<board[2][1]<<" | "<<board[2][2]<<" n";

cout<<"tt  |  |  n";}
```

```
//Function to get the player input and update the board
void player_turn(){
    if(turn == 'X'){
        cout<<"ntPlayer - 1 [X] turn : ";
    }
    else if(turn == 'O'){
        cout<<"ntPlayer - 2 [O] turn : ";
    }
}
//Taking input from user
//updating the board according to choice and reassigning the turn Start
cin>> choice;
//switch case to get which row and column will be update
switch(choice){
    case 1: row=0; column=0; break;
    case 2: row=0; column=1; break;
    case 3: row=0; column=2; break;
    case 4: row=1; column=0; break;
    case 5: row=1; column=1; break;
    case 6: row=1; column=2; break;
    case 7: row=2; column=0; break;
    case 8: row=2; column=1; break;
    case 9: row=2; column=2; break;
    default:
        cout<<"Invalid Move";
}
```

```
if(turn == 'X' && board[row][column] != 'X' && board[row][column] != 'O')
{
//updating the position for 'X' symbol if
//it is not already occupied
board[row][column] = 'X';
turn = 'O';
}
else if(turn == 'O' && board[row][column] != 'X' && board[row][column] != 'O'){
//updating the position for 'O' symbol if
//it is not already occupied
board[row][column] = 'O';
turn = 'X';
}
else {
//if input position already filled
cout<<"Box already filled!n Please choose another!!nn";
player_turn();
}
/* Ends */
display_board();
}
//Function to get the game status e.g. GAME WON, GAME DRAW GAME IN CONTINUE MODE
```

```
bool gameover(){
//checking the win for Simple Rows and Simple Column
for(int i=0; i<3; i++)
if(board[i][0] == board[i][1] && board[i][0] == board[i][2] || board[0][i] == board[1][i] && board[0][i] == board[2][i])
return false;
//checking the win for both diagonal
if(board[0][0] == board[1][1] && board[0][0] == board[2][2] || board[0][2] == board[1][1] && board[0][2] == board[2][0])
return false;
//Checking the game is in continue mode or not
for(int i=0; i<3; i++)
for(int j=0; j<3; j++)
if(board[i][j] != 'X' && board[i][j] != 'O')
return true;
//Checking the if game already draw
draw = true;
return false;}
```



```
//Program Main Method
int main()
{
cout<<"tttT I C K -- T A C -- T O E -- G A M Ettt";
cout<<"nttttFOR 2 PLAYERSnttt";
while(gameover()){
display_board();
player_turn();
gameover();
}
if(turn == 'X' && draw == false){
cout<<"nnCongratulations!Player with 'X' has won the game";
}
else if(turn == 'O' && draw == false){
cout<<"nnCongratulations!Player with 'O' has won the game";
}
else {
cout<<"nnGAME DRAW!!!nn";
}
}
```

