

NATIONAL COLLEGE OF ENGINEERING
(Affiliated to Tribhuvan University)
Talchhikhel, Lalitpur



Computer graphics

Project on:

“Tic Tac Toe”

Submitted By:

Aayush Shrestha
077BCT002
Group: A1

Submitted To:

Department of Electronics & Computer
Engineering

Table of Contents

Introduction:.....	3
Functions Used:	3
1. initgraph():	3
2. cleardevice():.....	3
3. line():.....	3
4. ismouseclick():.....	3
5. getmouseclick():.....	4
6. circle():.....	4
7. outtextxy():.....	4
8. setcolor():	4
9. setttextstyle():.....	4
10. delay():	4
11. getch():	4
12. closegraph():	4
Expected Output:	5
SourceCode:.....	5
Output:	11
Conclusion:	12

Introduction:

The project involves developing a Tic Tac Toe game in C++ using the **graphics.h** library. The game provides an interactive interface where two players can take turns using mouse clicks to place their 'X' or 'O' symbols on a 3x3 grid. The implementation includes basic graphics functions to draw the grid, symbols, and detect win conditions. Players can enter their names for a personalized experience, and the winning player's name is displayed upon victory. The project aims to create an engaging and visually appealing gaming environment, showcasing the integration of C++ programming with graphics functionalities. The game runs on older versions of Turbo C++ and offers an opportunity to explore graphics programming concepts while enjoying the classic Tic Tac Toe gameplay.

Objective:

The main objectives of this project are: -

- To learn and implement different features of graphics in C
- To interface the applications of graphics to the real world
- To familiarize with Graphics and its logical coding

Functions Used:

The functions used in this project are as follows:

1. **initgraph()**: This function initializes the graphics system and sets up the graphics window. It is the first function called when using the `'graphics.h'` library and prepares the environment for graphics-related operations.

2. **cleardevice()**: The `'cleardevice()'` function clears the entire graphics window by filling it with the background color. It is typically used before redrawing the contents of the window to prevent overlapping or artifacts.

3. **line()**: This function draws a straight line between two specified points on the graphics window. It takes the coordinates of the starting and ending points as its arguments and uses the current drawing color.

4. **ismouseclick()**: The `'ismouseclick()'` function checks whether a mouse click event has occurred. It is used to detect mouse interactions and is often paired with `'getmouseclick()'` to retrieve details about the mouse click.

5. `getmouseclick()`: When used with `'ismouseclick()'`, this function retrieves the details of a mouse click event, such as the button pressed (left, right, middle), and the coordinates (x and y) of the click on the graphics window.

6. `circle()`: The `'circle()'` function draws a circle on the graphics window. It requires the center coordinates and the radius as arguments. The current drawing color is used to render the circle.

7. `outtextxy()`: This function displays text on the graphics window at the specified coordinates (x, y). It allows developers to add textual information to the graphical user interface.

8. `setcolor()`: The `'setcolor()'` function sets the current drawing color. It allows developers to choose the color for lines, shapes, and text to be drawn on the graphics window.

9. `settextstyle()`: This function sets the font style and size for text to be displayed using `'outtextxy()'`. It allows developers to customize the appearance of the text.

10. `delay()`: The `'delay()'` function pauses the program execution for a specified number of milliseconds. It is commonly used to introduce delays between graphics-related operations, creating animations, or controlling game speed.

11. `getch()`: The `'getch()'` function waits for a key press from the user. It is often used to halt the program's execution until the user provides input, making it useful for handling user interactions.

12. `closegraph()`: This function closes the graphics window and releases any resources allocated by the `'initgraph()'` function. It ensures that the graphics environment is safely shut down before the program terminates.

Expected Output:

Tic Tac Toe		
	X	O
O	X	X

SourceCode:

```
#include <iostream>
#include <conio.h>
#include <graphics.h>
using namespace std;

const int WINDOW_SIZE = 300;
const int CELL_SIZE = WINDOW_SIZE / 3;

void drawBoard() {
    line(CELL_SIZE, 0, CELL_SIZE, WINDOW_SIZE);
    line(2 * CELL_SIZE, 0, 2 * CELL_SIZE, WINDOW_SIZE);
    line(0, CELL_SIZE, WINDOW_SIZE, CELL_SIZE);
    line(0, 2 * CELL_SIZE, WINDOW_SIZE, 2 * CELL_SIZE);
}

char checkWin(char board[3][3], int& x1, int& y1, int& x2, int& y2) {
```

```

// Check rows and columns
for (int i = 0; i < 3; ++i) {
    if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != ' ') {
        x1 = 0;
        y1 = i * CELL_SIZE + CELL_SIZE / 2;
        x2 = WINDOW_SIZE;
        y2 = i * CELL_SIZE + CELL_SIZE / 2;
        return board[i][0];
    }
    if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != ' ') {
        x1 = i * CELL_SIZE + CELL_SIZE / 2;
        y1 = 0;
        x2 = i * CELL_SIZE + CELL_SIZE / 2;
        y2 = WINDOW_SIZE;
        return board[0][i];
    }
}

```

```

// Check diagonals
if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != ' ') {
    x1 = 0;
    y1 = 0;
    x2 = WINDOW_SIZE;
    y2 = WINDOW_SIZE;
    return board[0][0];
}
if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != ' ') {
    x1 = WINDOW_SIZE;
    y1 = 0;
    x2 = 0;

```

```

        y2 = WINDOW_SIZE;
        return board[0][2];
    }

    // Check for draw
    bool draw = true;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (board[i][j] == ' ') {
                draw = false;
                break;
            }
        }
    }
    if (!draw)
        break;
}

if (draw)
    return 'D';

// Game still ongoing
return ' ';
}

void drawMove(int row, int col, char player) {
    int x = col * CELL_SIZE + CELL_SIZE / 2;
    int y = row * CELL_SIZE + CELL_SIZE / 2;

    if (player == 'X') {
        // Draw 'X'
        line(x - 25, y - 25, x + 25, y + 25);
    }
}

```

```

        line(x - 25, y + 25, x + 25, y - 25);
    } else if (player == 'O') {
        // Draw 'O'
        circle(x, y, 25);
    }
}

```

```

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    char board[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '}
    };

    char currentPlayer = 'X';
    char result;

    int x1, y1, x2, y2; // Coordinates for the line
    int lastRow, lastCol;

    while (true) {
        cleardevice();
        drawBoard();

        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j < 3; ++j) {
                if (board[i][j] != ' ') {
                    drawMove(i, j, board[i][j]);
                }
            }
        }
    }
}

```



```

    }
}
}

int mouseClicked = 0;
while (!mouseClick) {
    if (ismouseclick(WM_LBUTTONDOWN)) {
        int x, y;
        getmouseclick(WM_LBUTTONDOWN, x, y);
        int row = y / CELL_SIZE;
        int col = x / CELL_SIZE;

        if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == ' ') {
            board[row][col] = currentPlayer;
            mouseClicked = 1;
            currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
            lastRow = row;
            lastCol = col;
        }
    }
}

// Check for win or draw
result = checkWin(board, x1, y1, x2, y2);
if (result == 'X') {
    line(x1, y1, x2, y2); // Draw the winning line
    drawMove(lastRow, lastCol, board[lastRow][lastCol]);
    setcolor(RED);
    settxtstyle(SANS_SERIF_FONT, HORIZ_DIR, 3);
}

```

```

        outtextxy((WINDOW_SIZE-textwidth("Player 1 wins!"))/2, WINDOW_SIZE / 2 +
150,(char*)"Player 1 wins!");

            break;

    }

    else if(result == 'O') {

        line(x1, y1, x2, y2); // Draw the winning line

            drawMove(lastRow, lastCol, board[lastRow][lastCol]);

            setcolor(RED);

        settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 3);

        outtextxy((WINDOW_SIZE-textwidth("Player 2 wins!"))/2, WINDOW_SIZE / 2 +
150,(char*)"Player 2 wins!");

            break;

    }

    else if (result == 'D') {

        outtextxy((WINDOW_SIZE-textwidth("Its a Draw!"))/2, WINDOW_SIZE / 2 +
150,(char*)"Its a Draw!");

            break;

    }

    delay(200);

}

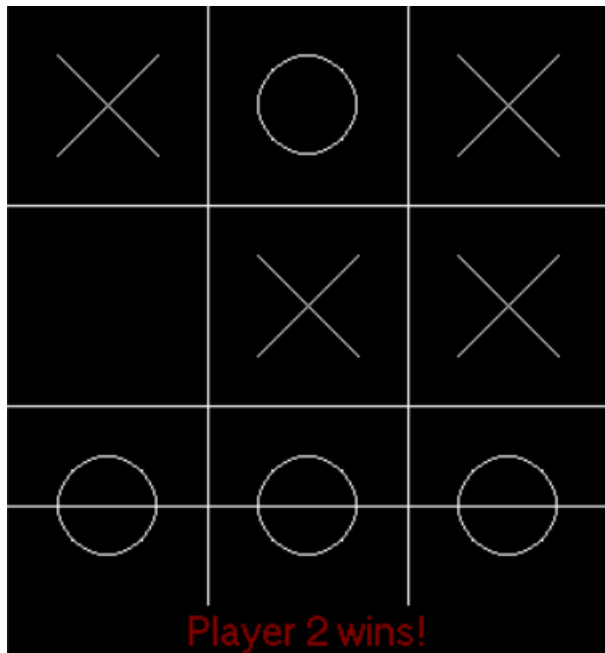
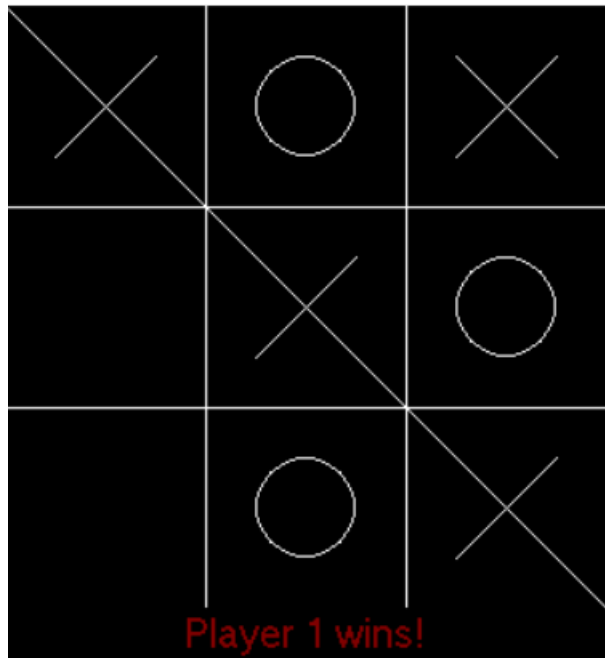
getch(); // Wait for a key press before closing the graphics window

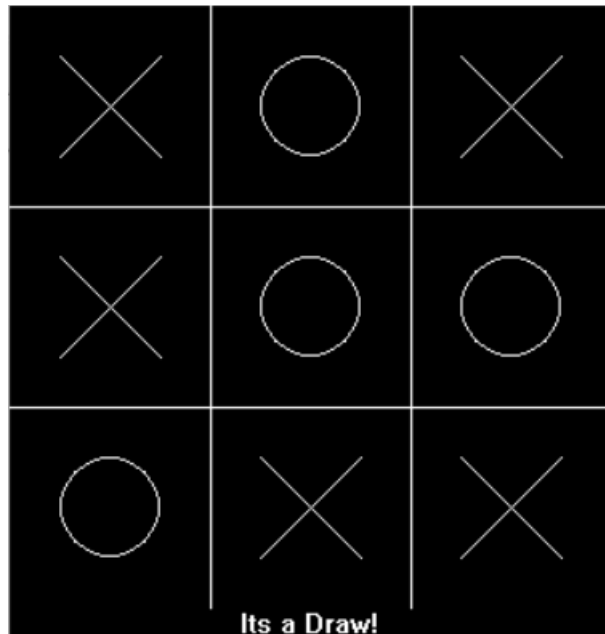
closegraph();

return 0;

```

Output:





Conclusion:

In this project, we developed a Tic Tac Toe game using the **graphics.h** library in C++. The game allows two players to take turns and place their 'X' or 'O' symbols on a 3x3 grid through mouse clicks. We implemented basic graphics functions to draw the grid, symbols, and detect win conditions. The game also allows players to enter their names, and upon a win, the winning player's name is displayed alongside the winning move. The use of **textwidth()** and **textheight()** functions improved text alignment. It provides a foundation for understanding graphics programming in C++ and can be extended with additional features in the future.