CV. Raman Global University Bhubaneshwar, Odisha ESTD-2020

REPORT

Tic-Tac-Toe Game in C++

Course Instructor:MD SIKANDAR

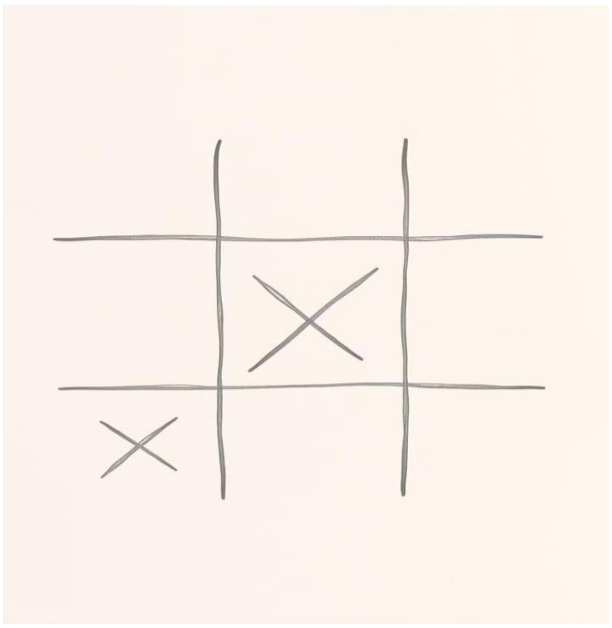
GROUP : 4

TEAM MEMBERS

SL.NO	REG.NO.	NAME
1	2201020519	BISHAL MONDAL
2	2201020750	NAFIS ABID
3	2201020745	ADNAN AFZAL SIDDIUI
4	2201020744	SUBHASIS DEY
5	2201020333	ROHIT GUPTA

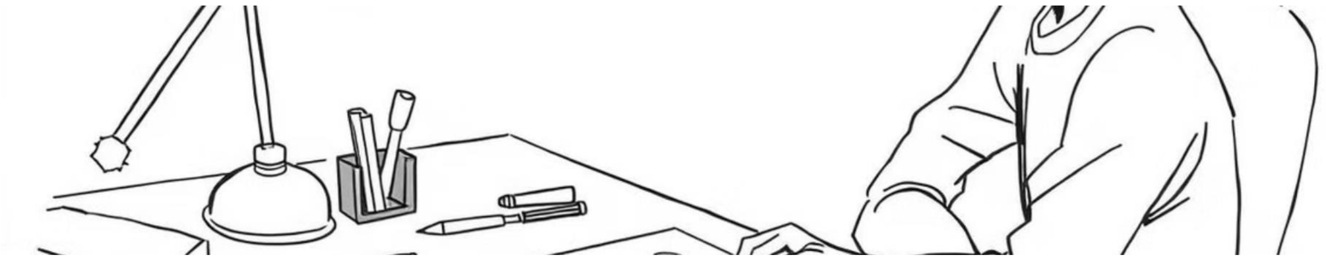
Introduction

Tic-Tac-Toe is a classic two-player strategy game that has been widely played for decades. The objective is for one player to place three of their marks (X or O) in a row—either horizontally, vertically, or diagonally—before their opponent does. This project implements the Tic-Tac-Toe game using C++, focusing on fundamental programming concepts such as input handling, game logic, and user interaction. The game is played on a 3x3 grid, and players take turns making their moves until a winner is determined or the game ends in a draw.



Project Objectives

- Develop a simple yet functional two-player Tic-Tac-Toe game.
- Implement a structured and modular approach to game design.
- Ensure proper input validation to prevent invalid moves and maintain fairness.
- Efficiently determine the game's outcome (win, draw, or continue).
- Provide a user-friendly interface for easy gameplay.
- Explore alternative solutions and improvements for enhanced functionality.



Technology Stack Used

- **Programming Language: C++**
  - Chosen for its efficiency and strong support for object-oriented programming.
  - Provides robust memory management and control over system resources.
- **Development Environment:** Code::Blocks, Visual Studio Code, or any C++ IDE
  - These IDEs offer syntax highlighting, debugging tools, and project management features.
- **Standard Libraries Used:**
  - **iostream:** Handles input and output operations for console-based interaction.
  - **cstdlib:** Used for system functions like clearing the screen.
  - **vector:** Provides dynamic array functionality for managing game data efficiently.
- **GUI Implementation:** SFML (Simple and Fast Multimedia Library)
  - Used for rendering graphical elements and handling user input.
  - Enables a modern, interactive Tic-Tac-Toe experience.
- **Compilation Tool:**
  - **GCC (GNU Compiler Collection):** A widely used compiler that ensures compatibility across different systems.
  - **MSVC (Microsoft Visual C++):** Offers optimized performance for Windows-based applications.

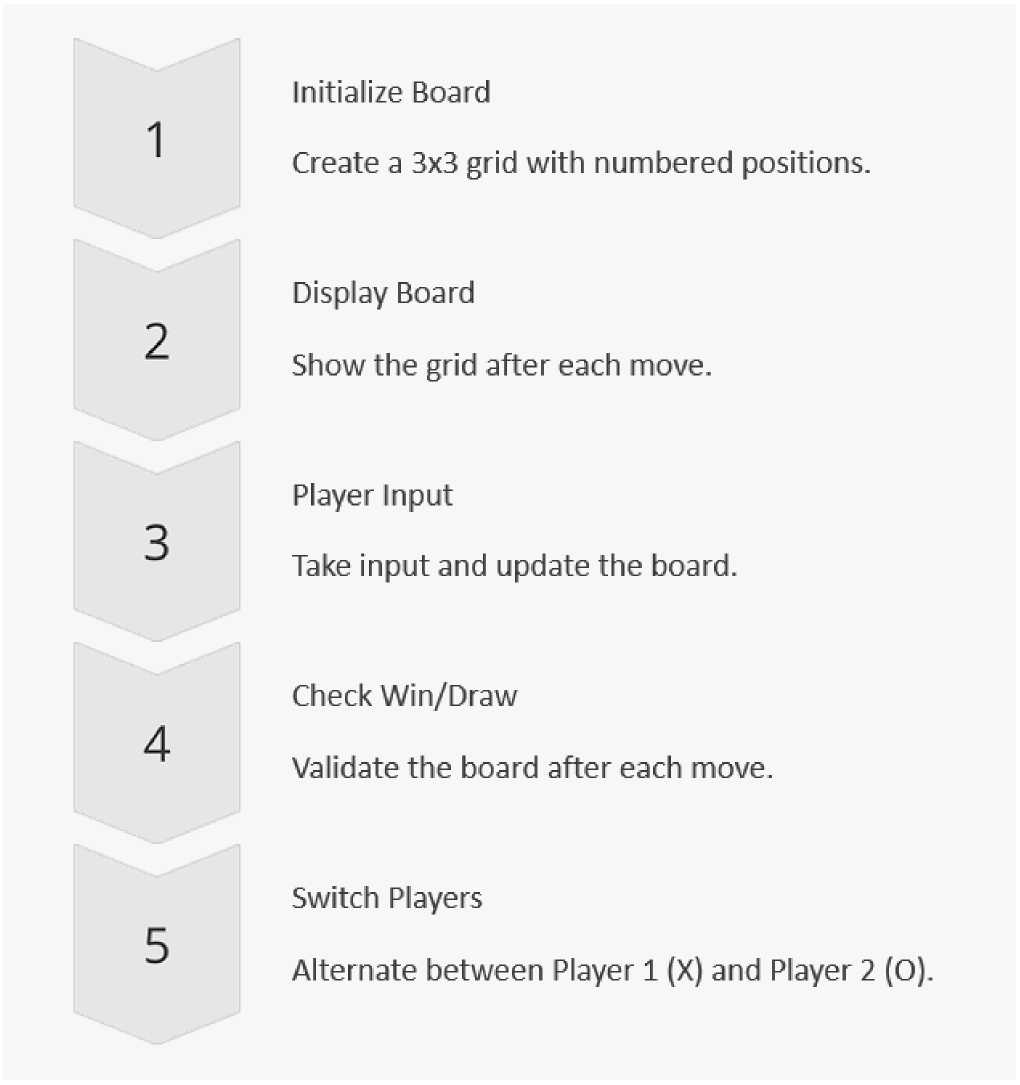
Key Features

1. **Two-Player Mode:** Enables two players to take turns making their moves.
2. **Input Validation:** Ensures that players cannot overwrite existing moves, preventing errors and maintaining fairness.
3. **Game Board Updates:** Dynamically updates and displays the game board after each move, allowing players to see the current state of the game.
4. **Win/Draw Checks:** Determines when a player has won or if the game has ended in a draw, ensuring the game follows its rules correctly.
5. **User-Friendly Interface:** A graphical user interface (GUI) implementation using

SFML for enhanced visual experience.

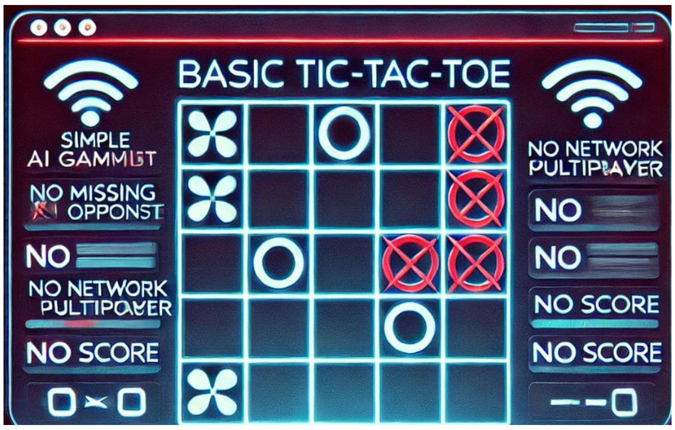
Implementation Steps

- **Initialize Board:** A 3x3 grid is created and assigned numbered positions for user-friendly input.
- **Display Board:** The game board is displayed after each move to reflect the current state.
- **Player Input Handling:** Players take turns entering their moves, and the board is updated accordingly.
- **Check Win/Draw Conditions:** The program checks after each move if a player has won or if the game has ended in a draw.
- **Switch Players:** Alternates between Player 1 (X) and Player 2 (O) after each valid move to ensure fair gameplay.
- **Loop Until Game Ends:** The game continues until a winning condition is met or the grid is full, resulting in a draw.


1
Initialize Board
Create a 3x3 grid with numbered positions.
2
Display Board
Show the grid after each move.
3
Player Input
Take input and update the board.
4
Check Win/Draw
Validate the board after each move.
5
Switch Players
Alternate between Player 1 (X) and Player 2 (O).

Drawbacks of the Current Implementation

- **Limited AI:** The game currently lacks an AI opponent, restricting it to two-player mode.
- **No Network Multiplayer:** Only supports local play; online multiplayer functionality is absent.
- **Basic User Interface:** While SFML enhances the visuals, additional effects and animations can further improve the experience.
- **No Score Tracking:** The game does not maintain a scoreboard to track multiple rounds or player performance.



Alternative Solutions & Future Enhancements

- **Implementing AI Opponent:** Using the Minimax algorithm, an AI can be added to allow single-player mode.
- **Online Multiplayer:** Integrating socket programming to enable remote play over a network.
- **Expanding Game Modes:** Implementing a 4x4 or 5x5 board variation to increase difficulty.
- **Enhanced UI/UX:** Adding animations, sounds, and interactive elements to improve the game experience.

Graphical Implementation Using SFML

```
#include <SFML/Graphics.hpp>
#include <optional>
#include <iostream>

using namespace sf;
using namespace std;

const int SIZE = 3;
const int CELL_SIZE = 200;
char board[SIZE][SIZE] = {{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}};
char current_marker = 'X';
int current_player = 1;

int checkWin() {
    for (int i = 0; i < SIZE; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != ' ')
            return current_player;
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != ' ')
            return current_player;
    }
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != ' ')
        return current_player;
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != ' ')
        return current_player;
    return 0;
}

bool placeMarker(int row, int col) {
    if (board[row][col] != ' ') return false;
    board[row][col] = current_marker;
    return true;
}

void swapPlayer() {
    current_marker = (current_marker == 'X') ? 'O' : 'X';
    current_player = (current_player == 1) ? 2 : 1;
}

int main() {
    RenderWindow window(VideoMode(Vector2u(600, 600)), "Tic-Tac-Toe");
    Font font;
    if (!font.openFromFile("arial.ttf")) {
        cerr << "Failed to load font!" << endl;
        return -1;
    }

    Text text(font);
    text.setString(" ");
    text.setCharacterSize(50);
    text.setFillColor(Color::White);

    int winner = 0, moves = 0;
    window.display();
}
```

Conclusion

This project successfully demonstrates fundamental programming concepts, including user input handling, game logic, and modular design. The implementation of SFML significantly enhances the graphical user experience. However, adding AI, online multiplayer, and improved UI/UX can make the game even more engaging and feature-rich. Future improvements can transform this project into a more interactive and enjoyable gaming experience.