TicTacX: A Command-Line Tic-Tac-Toe Game Implementation in C

Avishek Dutta (2524093042)
Md Ishrak Mashroor (2524709042)
Ashab Mahmud Raseen (2524767042)
Sneha Nandy (2524508042)

CSE115.4 | Group 2

Summer 2025

Abstract

This paper summarises development of a C-based Tic-Tac-Toe game TicTacX developed in CSE115. It has a working multiplayer component to be used with an ANSI and Unicode enhanced command line interface and capable input validation and cross-platform support. The main highlights evidence modular design, data structures, and the development of UI. Additional Single-player mode updates will include AI.

1 Introduction

Noughts and Crosses or Tic-Tac-Toe is one of the simplest strategic games of computation game theory. A basic strategic game, which can be used to illustrate fundamental programming concepts, algorithms, and UI design. TicTacX represents some good C programming techniques including memory management, modularity and cross-platform applications as it aims at the ultimate design of a full, modular and extendable game system. It is currently implemented to match human-versus-human gaming, but it is also intended to support computer-based opponents through AI in future, a feature that makes it appropriate both in education and leisure.

2 Project Objectives

The three key objectives of the TicTacX project are to have a fully playable Tic-Tac-Toe game that has a user-friendly command line interface as well as a validated input and along with the cross-platform capabilities within Windows, Linux platforms, and macOS. There is a modular design that can be used to add improvements such as AI opponents in the future. Secondary goals entail the presentation of software engineering concepts by way of clean codes, documentation, and best practices. The project uses CSE115 concepts that provide practical skills in algorithms, data structures, and interfaces development.

3 System Architecture and Design

3.1 Modular Structure

TicTacX provides a modular design with 3 big components, namely main controller which deals with user interaction, game logic which is the core of the game rules and the board management which takes care of display and state management. This design is in line with software engineering design principles, where coupling is low, and cohesion high, so that development, testing can be done independently and this design can be extended easily in the future.

3.2 Data Structures

The game state is as 3×3 character array where the cell contains either X, or O or blank. This permits quick accessibility and appraisal of the board. The system keeps work of gameplay phases such as input, validation and win checking through the use of a finite state machine which makes it predictable and simpler to debug.

4 Implementation Progress

4.1 Completed Features

The present operating TicTacX has a working two-player multiplayer two-people-per-system mode, a more capable CLI with ANSI escape codes, supporting colored indicators of the players and Unicode-based representations of the boards and clearing the screen. The input validation deals with such malfunctions as wrong coordinates, out-of-range moves, and taken cells, displaying clear error messages and allowing to attempt again. Windows, Linux, and macOS are cross-platform compatible through OS-detection preprocessors to provide adequate UTF-8 and console settings.

4.2 User Interface Design

The CLI system of coordinating game mode, instructions, project details and pausing to exit includes the menu mode that is developed to be easily accessed and understandable to the user who interacts with it.

There are clean grid lines in unicode characters betwixt which red X and blue O symbols are portrayed to make it clearer. The labels to the coordinates and the format simplify the perception and minimize the confusion.

The system incorporates good input validation to deal with bad input in coordinate format, range and occupied cell, to allow the game to carry out accurate game play.

5 Algorithm Implementation

5.1 Game Logic Engine

The game logic checks all 8 possible winning combinations (rows, columns, diagonals) in constant time to detect wins. The system manages turns, validates moves, and detects draws when the board is full with no winner, ensuring full rule enforcement.

5.2 Planned AI Implementation

AI support will be implemented soon starting with an easy difficulty level with randomly selecting moves. The system makes the game action fast and unpredictable, which is beneficial in case of a beginner. Additional features of advanced AI can be the minimax with alpha-beta pruning, Monte Carlo Tree Search, the difficulty can be set by the use of controlled randomness, which provides challenges to players of all abilities.

6 Performance Analysis

6.1 Computational Complexity

Its performance is efficient in current implementation with constant-time validation of moves and checking of wins because of the fixed individual square dimensions since the size of the board is 3×3. The use of memory is low with the board taking up 9 bytes. It is flexible, and the system does not crash even after hours of using the game.

6.2 Cross-Platform Performance

According to performance tests, there is consistency in the behavior and reaction time in Windows, Linux, and macOS. The implementation across platforms can provide adequate support and display formatting of UTF-8 with little overhead, and visual constancy and trustworthiness across terminal circumstances.

7 Testing and Validation

The functionality and input validation of the system have been well tested and the UI consistency tested on all the options of game modes and outcome. By testing cross-platform, Unicode is correctly displayed, color settings, and input are formatted on Windows, Linux, and macOS. User acceptance testing was done to enhance the usability and error messages on the basis of the feedback.

8 Conclusion and Future Work

TicTacX is an example of a fully functioning multiplayer game leveraging all of the fundamental concepts of programming, thus reaching all of its objectives; that of being entertaining, stable, and pedagogically rich. It is modular and has all of its documentation covered so that it can have future additions such as AI opponents or new game modes. The project forms a practical experience in game development, algorithms and software engineering, since its code is clean and sufficiently tested, which makes it educationally applicable and suitable for other projects.