Final Report: Tic-Tac-Toe (XO) Bot

Abstract

This report outlines the development of a Tic-Tac-Toe (XO) Bot Game in C, covering both a terminal-based version with data structures (2d Arrays, Hash-Maps, Trees) and a graphical/web version using Raylib and WebAssembly. The document details the problem statement, implementation, data structures used, challenges faced, and key learnings.

Problem Description

The goal was to create an AI-driven Tic-Tac-Toe game where the user can play against a bot with three difficulty levels (easy, medium, hard). Additionally, a GUI version playable in a web browser was required.

Implementation Details

- Console Version: Moves tracked in a hashmap (uthash.h); full game history stored in a binary tree. User inputs 1-based coordinates; bot move generation varies by difficulty.

- GUI/Web Version: Built with Raylib for desktop; compiled to WebAssembly with Emscripten for online play. Includes HTML/JS/WASM files and a Makefile for build automation.

Data Structures Used

- 2D Arrays: To Create the board to put the moves on.

- Hash-Map (uthash.h): To store and access user and bot moves by move number.

- Binary Tree: To store sequential board states for backtracking and history display.

- Raylib Structures: For GUI rendering and event handling in the graphical version.

Challenges Faced

- Downloading and integrating new libraries (uthash.h, Raylib functions).

- Understanding and setting up WebAssembly compilation and runtime (Emscripten).

- Designing the hard-level bot by evaluating permutations and winning strategies.

- Uploading and configuring the project on GitHub Pages.

- Configuring VS Code, environment variables, and switching from Cygwin64 to MinGW-w64 for compatibility.

- Debugging compilation and runtime errors in both console and GUI/Web modes.

Conclusion and Learnings

Through iterative cycles of failure and learning, the project provided deep insights into advanced C data structures, cross-platform GUI development with Raylib, and WebAssembly deployment. Key takeaways include effective problem decomposition, build automation, and the importance of environment configuration for successful cross-platform software development.