Faculty of Engineering

Department of Electronics and Electrical Communications

ELC2080 Project

Tic Tac Toe game

Performance Measurement and Optimization

Under the supervision of:

Dr. Omar Ahmed Nasr

Contents

1.Introduction

1.1 Purpose

This document outlines the procedures, metrics, and results used to evaluate the performance of the Tic Tac Toe game. It also documents optimization strategies applied to ensure the system meets its performance-related requirements.

1.2 Background

Traditionally, Tic Tac Toe has been played on paper, but the increasing demand for digital and easily accessible entertainment highlights the need for a modernized version. This project addresses that demand by delivering a contemporary, feature-rich application that transforms the classic game into an engaging digital experience.

1.3 Target audience

The Tic Tac Toe desktop application is designed to appeal to a wide and diverse audience, including users of different age groups and technical proficiencies:

- **Families:** the game's simplicity and intuitive interface make it ideal for family-friendly usage, including children.
- **Friends**: it creates an environment full of cheer and happiness while playing. So, it is good to be a fun game that is played with friends.
- **Strategic thinkers**: players interested in challenging the AI to test and improve their tactical decision-making skills.

By catering to this diverse user base, the game aims to deliver an enjoyable experience that balances accessibility with strategic depth.

1.4 Scope

The Tic Tac Toe Game is a C++ application with a graphical user interface (GUI) that allows users to play either against another human or an AI opponent. It includes features such as user authentication, session management, personalized game history, and replay capability. The AI has different levels of difficulty. The system will be tested using Google Test and development will follow version control and CI/CD practices using GitHub and GitHub Actions.

1.5 Definitions, Acronyms, and Abbreviations

- **AI:** Artificial Intelligence.
- **GUI:** Graphical User Interface.
- **CI/CD:** Continuous Integration and Continuous Deployment.
- **SRS:** Software Requirements Specification.
- **Qt:** Cross-platform application development framework for GUI.
- **SQL:** Database engine.
- **UX:** User Experience.
- **UI:** User Interface.
- **OS:** Operating System.

1.6 Overview

This project aims to make the user experiences the enduring strategy and challenge of Tic Tac Toe in a modern and user-friendly format. This application offers a captivating and convenient way to test your skills, challenge others or AI opponent and experience the enduring fun of Tic Tac Toe.

The rest of this document provides a structured evaluation of the efficiency and responsiveness of the Advanced Tic Tac Toe Game. It defines measurable performance goals, outlines the testing environment and metrics used, presents the results of empirical measurements, and describes the optimizations implemented to meet the defined targets.

2.Metrics and Measurement Methods

2.1 Performance Metrics Measurement

The initial step involves establishing a performance baseline by identifying and measuring key metrics. These metrics serve as benchmarks for evaluating the effectiveness of optimization efforts. Here's a breakdown of this crucial phase:

- **Identifying Key Metrics**
  - **CPU Utilization:** This metric measures the percentage of processing power utilized by the game at any given moment. High CPU usage can lead to lag and stuttering during gameplay.
  - **Memory Consumption:** This metric tracks the amount of RAM occupied by the game. Excessive memory usage can impact overall system performance.
  - **Response Time:** This metric measures the time it takes for the game to respond to user actions, such as clicking on a board cell. Slow response times can hinder user enjoyment**.**

- **Data Collection**

  Continuously monitoring the selected metrics throughout different stages of gameplay helps us understand resource usage patterns and pinpoint potential bottlenecks for further analysis and optimization.

2.2 Measurements methods can be used

- **Manual instrumentation:** Timestamp logs before and after target function calls.
- **Profilers:** Qt Creator's built-in profiler for method-level call analysis.
- **Valgrind:** For tracking memory allocation and leak detection.
- **System Monitors:** htop (Linux) or Task Manager (Windows) for CPU/RAM during runtime.

2.3 Performance Objectives

Objective	Target threshold
GUI Input Response Time	≤ 100 ms Ensures quick visual feedback to user interactions like clicking or selecting moves.
AI Decision Time (Normal)	≤ 500 ms Allows the AI to feel responsive and smooth in medium difficulty mode.
AI Decision Time (Hard)	≤ 1000 ms Acceptable upper bound for harder AI computations while preserving user engagement.
Memory Usage During Game Session	≤ 50 MB Ensures the application remains lightweight and suitable for low-resource systems.
CPU Usage During Game Session	≤ 25% Prevents overloading the system and ensures background processes are unaffected.

3.Code Optimization Techniques

Once performance bottlenecks are identified through careful measurement, we can implement targeted code optimizations to address them. This phase involves a combination of algorithmic improvements and resource management strategies:

3.1 Algorithm Optimization

- **Minimax Algorithm**

  The Minimax algorithm, utilized by the AI opponent, systematically explores all potential game states to identify the most optimal move. To enhance its efficiency, this algorithm can be optimized through two primary methods: first, by implementing memorization techniques to cache previously computed game states, thereby preventing redundant calculations; and second, by employing heuristics to prioritize more promising moves, which effectively reduces the search space and can lead to faster decision-making, albeit sometimes at the cost of absolute optimality.

- **Alpha-Beta Pruning**

  This powerful technique eliminates branches in the Minimax decision tree that are demonstrably suboptimal, significantly reducing the search space and accelerating AI move selection.

3.2 Resource Management

- **Memory Management**: To prevent memory leaks, which occur when allocated memory is not properly released, it is crucial to implement best practices. This includes using appropriate data structures and strictly adhering to proper memory allocation and deallocation routines, ensuring efficient and responsible memory utilization throughout the application.
- **Data Structure Selection:** The choice of data structures significantly influences the game's performance in terms of storing and manipulating data. By carefully selecting efficient data structures, such as hash tables for rapid lookups or arrays for sequential access, developers can minimize processing overhead and achieve faster data access times.

4.Detailed Analysis

This section provides a detailed analysis of the expected performance characteristics of the Professional Tic-Tac-Toe application, focusing on memory usage, CPU utilization, and response time.

4.1 Memory Usage and CPU utilization

Event	Memory Usage	CPU Utilization
Just started	20.2MB	0%
After opened	21MB	0%
While playing	49MB	0.5%
Changing the frame (login, choosing mode, history, etc.)	37.5MB	0.2%
After saving the game	42MB	0%

4.2 Response time

Event	Response Time
UI Element Click	0.027ms
AI Decision (Easy)	0.025ms
AI Decision (Normal)	0.044ms
AI Decision (Hard)	0.72ms