Performance Measurement and Optimization

Project: Advanced Tic Tac Toe Game

1. Introduction

1.1 Purpose

The purpose of this document is to outline the strategy for measuring and optimizing the performance of the Advanced Tic Tac Toe Game. It includes performance objectives, tools, techniques, and a plan for continuous performance improvement.

1.2 Scope

This document applies to the entire Advanced Tic Tac Toe Game application, focusing on critical components such as the AI opponent, user interface responsiveness, and data handling.

2. Performance Objectives

2.1 Response Time

- **Objective:** inputs are processed until the user enter its data and clicks ok button
- **Components Affected:** User Interface, Game Logic

2.2 AI Move Calculation Time

- **Objective:** Ensure AI opponent moves are calculated within 1 second.
- **Components Affected:** AI Algorithm (Minimax with alpha-beta pruning)

2.3 Data Handling

- **Objective:** Ensure database operations (e.g., saving game history) are performed
- **Components Affected:** SQLite Database

3. Performance Measurement

3.1 Tools and Techniques

3.1.1 code examples to calculate the response time

```
#include <QElapsedTimer>

QElapsedTimer timer;
timer.start();
// Your code here
qint64 elapsed = timer.elapsed();
qDebug() << "Elapsed time: " << elapsed << " ms";
```

3.1.2 code examples to calculate AI Move Calculation Time

```
QElapsedTimer timer;
timer.start();
// AI move calculation code here
qint64 elapsed = timer.elapsed();
qDebug() << "AI move calculation time: " << elapsed << " ms";
```

3.1.3 code examples to calculate Database Operation Time

```
QElapsedTimer timer;
timer.start();
// Database operation code here
qint64 elapsed = timer.elapsed();
qDebug() << "Database operation time: " << elapsed << " ms";
```

3.2 Metrics

3.2.1 User Interface Response Time

- Measure time taken from user input (e.g., button click) to the corresponding action being completed.

3.2.2 AI Move Calculation Time

- Measure time taken by the AI to decide and make a move.

3.2.3 Database Operation Time

- Measure time taken for database read/write operations.

4. Performance Optimization

4.1 Code Optimization

4.1.1 Algorithm Optimization

- **Minimax with Alpha-Beta Pruning:** Ensure optimal implementation to reduce unnecessary calculations.

4.1.2 Data Structures

- Use efficient data structures to manage game state and history.

4.2 Resource Management

4.2.1 Memory Management

- Ensure proper allocation and deallocation of memory.
- Use smart pointers to manage dynamic memory.

4.2.2 CPU Utilization

- Optimize loops and recursive functions.
- Minimize the use of heavy computations in the main UI thread.

4.3 Database Optimization

4.3.1 Indexing

- Create indexes on frequently queried columns.

4.3.2 Query Optimization

- Optimize SQL queries to reduce execution time.

5. Performance Testing

5.1 Testing Environment

- Define a controlled environment to conduct performance tests (e.g., specific hardware and software configurations).

5.2 Test Scenarios

5.2.1 User Interface Response

- Test various user interactions and measure response times.

5.2.2 AI Move Calculation

- Test AI performance with different game states and measure calculation times.

5.2.3 Database Operations

- Test database read/write operations with various data volumes and measure execution times.

5.3 Automated Testing

- Implement automated performance tests using tools like Google Test or Qt Test.

6. Continuous Performance Improvement

6.1 Monitoring

- Implement real-time monitoring of performance metrics in the production environment.

6.2 Feedback Loop

- Regularly collect and analyze performance data.
- Implement improvements based on the analysis.

6.3 Regular Audits

- Conduct regular performance audits to identify potential bottlenecks and areas for optimization.

7. Appendices

7.1 Appendix A: References

- Valgrind Documentation
- gprof Documentation
- Qt Creator Profiler Documentation

7.2 Appendix B: Glossary

- **Profiling:** The process of measuring the space (memory) and time complexity of a program.
- **Indexing:** A database optimization technique to speed up the retrieval of records.