ENGG1410: Introductory Programming for Engineers

Mini Project #1: Tic-Tac-Toe Game Simulation with Enhanced AI

Mohamad Abou El Nasr

School of Engineering, University of Guelph

October 28, 2024

Start Date

Week #8, 2024

Duration

2 Weeks

Due Date

Week #9, 2024.
Demo in lab and report upload in DropBox Midnight Sunday Nov. 10, 2024. (Check DropBox for exact dates and details).

Objectives

In this mini-project, you will build a simulation of the **Tic-Tac-Toe** game in C. You will implement two-player mode and player vs Computer (AI mode). The AI will have basic strategic capabilities, trying to win if there is a chance and preventing you from winning again if possible. The program will allow customization of grid size and will include score tracking across multiple rounds. This project serves as an educational tool to reinforce students' understanding of arrays, loops, functions, and basic AI logic in C programming.

The main objectives of this project are:

- Practice procedural and data abstraction in C.
- Work with arrays and functions.
- Implement AI strategies for decision-making.

Introduction

The **Tic-Tac-Toe** game is a simple yet classic example of a two-player game. It is played on a 3x3 grid, but this project allows for customizable grid sizes up to 10x10. Players alternate turns, placing their mark (X or O) in an empty cell. The player who first forms a horizontal, vertical, or diagonal line of their marks wins. If the grid is filled without any player forming a line, the game ends in a draw.

In this project, we will enhance the traditional game by allowing players to play against an AI opponent. The AI can be programmed with basic strategies like blocking and winning moves.

Main Topic of Mini Project #2

You will implement a C program that simulates the **Tic-Tac-Toe** game, which includes the following features:

- A 2D array represents the game board, supporting customizable grid sizes from 3x3 up to 10x10.
- Two game modes: player vs player and player vs AI.
- Enhanced AI that blocks the opponent's winning moves and tries to win whenever possible.
- Detection of win and draw conditions.
- Score tracking across multiple rounds.

Requirements

Your program must implement the following functionality:

1. Two game modes: player vs player, and player vs AI.
2. AI must use basic strategies for blocking and winning moves. (Not randomly filling an empty place)
3. The grid size should be customizable from 3x3 to 10x10.
4. Score tracking must be included across multiple rounds.

Design Overview

The program is structured around the following components:

- **Board Representation**: A 2D array is used to represent the game board, where each cell contains either 'X', 'O', or an empty space.
- **Game Modes**: Players can choose to play against another player or an AI.
- **User Interaction**: The program takes user input for moves and displays the current state of the board after each turn.
- **Win/Draw Detection**: Functions are implemented to check for win or draw conditions after each move.
- **Score Tracking**: Scores are updated and displayed after each round.

Implementation Details

The program consists of several functions that handle various aspects of the game. Each function is described in detail below.

Main Function

The `main()` function is the entry point of the program and handles the following:

- Initializes the game by setting up the board and selecting the game mode.
- Manages the main game loop, alternating turns between players or AI.
- Calls the appropriate functions to handle player moves, check for win/draw conditions, and update the score.
- Prompts the user for a rematch at the end of each round.

Board Initialization

The `initializeBoard(char board[MAX_SIZE][MAX_SIZE], int size)` function sets up the board by filling it with empty spaces. This function ensures that the board is ready for a new round.

Display Board

The `printBoard(char board[MAX_SIZE][MAX_SIZE], int size)` function prints the current state of the board in a grid format. It provides a clear visualization of the game's progress, helping players make informed moves.

Player Move

The `playerMove(char board[MAX_SIZE][MAX_SIZE], int size, char player)` function prompts the user for input and validates the move. It ensures that the input is within the bounds of the grid and that the selected cell is empty.

AI Move

The `aiMove(char board[MAX_SIZE][MAX_SIZE], int size)` function generates a random valid move for the AI player. It iterates until an empty cell is found, simulating a basic AI opponent.

Win Detection

The `checkWin(char board[MAX_SIZE][MAX_SIZE], int size)` function checks for a winning condition by evaluating rows, columns, and diagonals. It returns 1 if a win is detected and 0 otherwise.

Draw Detection

The `checkDraw(char board[MAX_SIZE][MAX_SIZE], int size)` function checks if the board is completely filled without a winner, indicating a draw.

Score Update

The `updateScore(char winner)` function increments the score of the winning player prints the current game score and keeps track of the scores across rounds.

Pseudocode

```
1. Initialize game board
2. Select game mode
3. Loop until the game is over:
   a. Display the 2D game board
   b. Player makes a move (or AI in AI mode)
   c. Check if the current player has won
   d. If not, check for a draw
   e. Switch players and continue
4. End the game and display the result
5. Ask if players want a rematch
```

Code

Below is the start code for the Tic-Tac-Toe game,

```
// Include necessary headers
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_SIZE 10  // Maximum grid size
```

Deliverables

- **Demo**: Demonstrate your program progress to the Teaching Assistant during lab hours. Be prepared to explain the code and answer any questions about the design and implementation.
- **Demo**: In coming week, you will need to demo the operation of the game with at least three Functions need to be operating error free (displaying the board, checking winning or checking draw, implementing the two player play mode , implementing AI move and score update)
- **Final Report**: The report must follow the format described in the next section.
- **Zipped Project File**: The project file should include the source code, README file, and input data (if any).

Report Format

Your report should include the following sections:

1. **Title Page**: Include the course number, your group number, and the date.
2. **Problem Statement**: Briefly describe the problem your program solves.
3. **Assumptions and Constraints**: List any constraints or assumptions you made.
4. **Solution Description**:
   - Flowchart or block diagram of the system.
   - Pseudocode for your program.
   - Justification of design decisions and system overview.
5. **Error Analysis**:
   - Ensure there are no syntax, semantic, or logical errors.
   - Describe any problems encountered and how you resolved them.

Pseudocode

Pseudocode

```
1. Initialize game board
2. Select game mode
3. Loop until the game is over:
   a. Player makes a move (or AI in AI mode)
   b. Check if the current player has won
   c. If not, check for a draw
   d. Switch players and continue
4. End the game and display the result
5. Ask if players want a rematch
```

AI mode

The AI logic is designed to block the opponent's winning moves and win when possible. The AI's decision-making process is:

- **Win Check**: If the AI can win in the next move, it takes that move.
- **Block Check**: If the opponent is about to win, the AI blocks the move.
- **Random Move**: If no immediate win or block is available, the AI makes a random move.