LUDO like UCSC

Seniru Senaweera - 2023/CS/187

September 1, 2024

Abstract

This report discusses the design and implementation of the board and piece structures in LUDO like UCSC. The structures were chosen to efficiently represent the game's components and support various game mechanics. An analysis of the program's efficiency is also presented.

1 Introduction

This project involves the development of a LUDO-CS game where players move pieces on a board with various mechanics, including capturing, direction-based movement, and special conditions like powers. The core structures are designed to represent the board, the pieces, and the players.

2 Structures

2.1 Code for Structures

The key data structures used in the program are shown below.

Listing 1: Structure to represent each square on the board

```
typedef struct Square
{
    int square_number;
    int pieceCount;
    struct Square *prev;
    struct Square *next;
} Square;
```

Listing 2: Structure to represent a piece (pawn) on the board

```
typedef struct
{
    int isInPlay;                // -1: In base, 1: Active on the board, 2
```

```
    int direction;                  // 0: Clockwise, 1: Anticlockwise
    Square *current_square;         // Pointer to the current square the piec
    int isCapture;
    int approachCellPassed;
    int homeStraight;
    int isSick;                     // 1: sick, 2: energized, -1: else
    int isSickNumber;
    int mysteryCell;
    int block;
    int brief;                      // Cannot move = 0
} Piece;
```

Listing 3: Structure to represent a player and their pieces

```
typedef struct Player
{
    char colour;
    Piece pieces[TOTAL_PIECES]; // Array of pieces for the player
    int approachCell;           // The approach cell number for the playe
    int hasWon;                 // place of the player (1: 1st place, 2:
} Player;
```

2.2 Justification for Structures

The chosen structures are justified based on the following considerations:

- **Square Structure:** The 'Square' structure uses a doubly linked list to represent the board, allowing for efficient traversal in both directions. This is essential for move both clockwise and anticlockwise.
- **Piece Structure:** The 'Piece' structure includes fields to manage various game states such as whether the piece is in play, its current direction, and special conditions like sickness. This design allows for comprehensive state management of each piece.
- **Player Structure:** The 'Player' structure groups a player's pieces and tracks their progress in the game. The use of an array for the pieces enables quick access to any piece.hasWon variable describes place of the final result.

3 Efficiency Analysis

3.1 Justification for Efficiency

The design choices ensure that the program runs efficiently in terms of both time and space:

- The linked list implementation of the board allows for flexible game mechanics (e.g., reversing direction) without compromising performance.
- The compact representation of pieces supports quick updates and state checks, which is crucial during gameplay where multiple pieces may move or interact simultaneously.
- Overall, the structures are optimized for the specific needs of the game, balancing complexity with ease of implementation and maintainability.

4 Conclusion

This report presented the core structures used in the game, providing justification for their design based on the specific requirements of the game.