LUDO-UCSC:REPORT

Warusha Shamalka

September 1, 2024

1 Introduction

This is the latex report for my ludo project.This report will include what are the structures that I used to represent the board and pieces and justifications for them.

2 Content

1. What are the Structure I have used and Justification.
2. Summary of My Functions.

3 What are the structures I have used and Justification

I have used two structures.One for players and another one for pieces.

1. Player Structure:

   I have included four members under this structure.Then main usecase of this structure is to arrange the player order according to the first player.

   First member of this structure is player name.I wanted to store every players' name.So i can easily recognize the player whenever I want.Since C language does not support for strings i used a char array to store player name.

   The second member of this structure is base piece count.I needed this to calculate how many pieces actually on the base.And also I use this to print the number of pieces in the base and number of pieces in the board after each round.I did not invent a new member to represent the number of pieces on the board.I have calculated it using the base piece count.

   My third member of this structure is initial Roll.I used this to store the players' initial dice value and to choose who has the highest dice value.

   The last member of this structure is caputure flag.I needed this member to give another turn to the same player when a capture occured.This is bool type and it will become true when a capture occured.Then I give another turn to the same player using this member.Next section I will discuss about my second structure.

2. Piece Structure: The main usecase of this structure is to handle single piece.Every piece has some special properties to handle and using a this structre for a single piece is the best way to handle it. I used a 2D array of structures to access each piece more easily.The reason is from this 2D array i can represent players from the rows and their pieces from the columns.So I can easily access any number by just changing the row number and column number.

   First member is playerName; This stores the name of the player who owns the piece. It's essential for identifying which player the piece belongs to, especially in a multiplayer environment where multiple players have pieces on the board.This will be used to differentiate between pieces of different players, allowing the game to attribute moves and actions to the correct player.

   Second member is pieceName;This gives a unique identifier to each piece.Since a player might have multiple pieces on the board, each needs a way to be uniquely identified.In the game, this will help track the movement of specific pieces and apply rules or effects to the correct piece when a move is made.

   Third member is capture;This indicates whether the piece has captured another piece, which is a key event.It will trigger game events or effects,such as sending the captured piece back to the start, and also award the capturing player with additional turn.I used a bool type here and by doing that i prevented some unneccessary calculations.Otherwise I might need to get capture count for give access to the homestraight for a piece.

   Fourth member is state;The state represents the current status of the piece (e.g., in base, on board, at home) The game logic will check this member to determine what actions are possible for the piece (e.g., whether it can move or if it's already finished).

   Fifth member is location;This tracks the current location of the piece on the board, which is critical for determining where the piece can move next.This is used to calculate the next position of the piece when the dice is rolled and determine interactions with other pieces or board elements.

Sixth member is homeStraightLocation;Once a piece enters the home straight,this variable tracks its specific location within that.This will help to manage the rules and moves when the piece is on its final approach to reaching home.

Seventh member is direction;This indicates the direction the piece is moving.It's necessary to manage special cases,like when a piece might be moving backward due to a game effect.This will be referenced when moving the piece to ensure it follows the correct path according to the rules of the game.

4 Summary of my Functions

4.1 Global Variables

In my ludo project I have used diceValue and maxIndex as global variables.The reason to use diceValue as a global variable is to use it in the pieceCapture function without passing as an argument.And I thought it would be helpful for future improvments as well.I always looked to reduce the number of global variables as much as possible.

My second global variable is maxIndex and it has initialized to minus one.maxIndex refers to the index number of the player who has the highest value.The reason is to make it as a global variable is to use it again in arrangePlayer function.Since I am calling firstPlayer and unique recursively making it as a global variable would brings more clarity.

4.2 About Functions

1. firstPlayer function:

   Rolls the dice for all players to determine who goes first.Calls unique to ensure that the initial roll values are unique.I used a for loop to assign the initial roll for every player and all the variable types are shorts.

2. unique function:

   Checks if there's a tie in the highest dice rolls. If so, it calls firstPlayer() again to reroll.

3. arrangePlayers function:

   The function first prints the initial roll values for each player.It then identifies the player with the highest roll value (maxIndex) and announces that they will begin the game.The players' order is rearranged based on the highest roll value using a temporary array. The new order ensures that the player with the highest roll starts first.The new order of players for the round is printed.Finally, the function calls pieceReset() to reset the game pieces, likely preparing them for the start of the game.Since its using a temporary array to rearrange players the execution time will be inceresed.

4. pieceReset function:

   This resets the state of each piece for all players.This includes setting the starting positions and resetting attributes like direction, capture state, and location.

5. pieceCapture and pieceCaptureReset function:

   Handles the logic for when one player's piece captures another player's piece. This includes resetting the captured piece's state and position.There are some logics that are same for both clockwise and counter clockwise pieces,so by including those things into one function i was able to reduce the redundancy and and improve the efficientcy,

6. pieceDirection function:

   Determines the movement direction of a piece (clockwise or counterclockwise).

7. getPieceFromBase function:

   Moves a piece from the base to the board, starting its journey on the path.The function exits early by returning false as soon as it finds a piece in the base and moves it to the board. This prevents unnecessary iteration through the remaining pieces once the required operation is completed.when a piece is found early in the loop.The function efficiently handles the transition of a piece's state from BASE to BOARD in a single step.The direction of the piece is assigned immediately after the state change using the pieceDirection function. Since the function directly uses the returned value from pieceDirection() to set the piece's direction, this avoids redundancy.

8. homeStraight function:

   Manages the logic for pieces approaching their home stretch and checks if they reach home.In here again i used early returns to exit as soon as a piece is moved or an action is taken. This approach avoids unnecessary processing of the remaining pieces once a valid move has been made.The function handles both clockwise and counter clockwise directions efficiently by splitting the logic based on the piece's direction. This avoids complexity and improved readability.And alsoI used simple arithmetic to update the piece's position to improve the efficientcy.

9. gameRounds function:

   The nextRound variable is used to control the loop, allowing for an immediate exit when a player wins.And this function selectively processes each piece based on its state.It optimizes piece movement by breaking out of loops once a valid move is made (break statements). This prevents further unnecessary checks after a piece has been successfully moved, reducing the number of iterations.I only used shorts to reduce the wastage of memory.