Checkers Game Data Model Concept

Pawprint: dltdc
dltdc@mail.missouri.edu

Description: create a data model concept for the game of Checkers (Draughts).

Requirements:

Our ultimate goal is to develop a Checkers (Draughts) game. The app will allow both human and computer players in any combination: human-human, computer-human, computer- computer. The app is to do the following:

- Collect and display the player names.
- Determine who moves first and gets assigned the black (dark) pieces.
- Display a board with pieces the user can interact with if one or both players are human.
- Prevent illegal moves.
- Identify when there is a winner, loser, or if a draw has occurred.
- The data model is a representation of the Checkers game as data and methods without any reference to UI elements. In coming up with the data model you need to consider the best data representation for the state of the game (the pieces on the board) so that it is easy to:

    determine what moves are possible/allowable.

    determine if a move is illegal.

    determine a winner, loser, or a draw condition.

    implement the algorithms (outside of the model) for the computer (AI) to determine moves.

implement a UI representation of the game and game play. The data model is to include not only the data to represent the pieces on the board but all of the data needed for playing the game. The data model is to include methods for changing/managing the state of the model. These methods are to be used to change and determine the state of the game.

Solution:

A MVC model can be built for checker game. By extracting entities and functions of Requirements, I find that the Player can be assigned as models, as well as the board, the checker board can be assigned as view, and the CheckerGame can be assigned as controller.

At first we initial the checker game, arrange the board state according to the game type (e.g, 10X10 or 8X8), and show the players' name on each side. So, the controller will have a reference to each player. This reference will be given after all players join the game, it will be in the construct method of controller. The viewer elements are binding with the users' name and color.

According to the color we assigned to the player, we can determine who moves first.

When user click one man or king on the board, the controller will read the current game situation, and calculate the possible movement the chessman have. Then, viewer will display these possible movement.

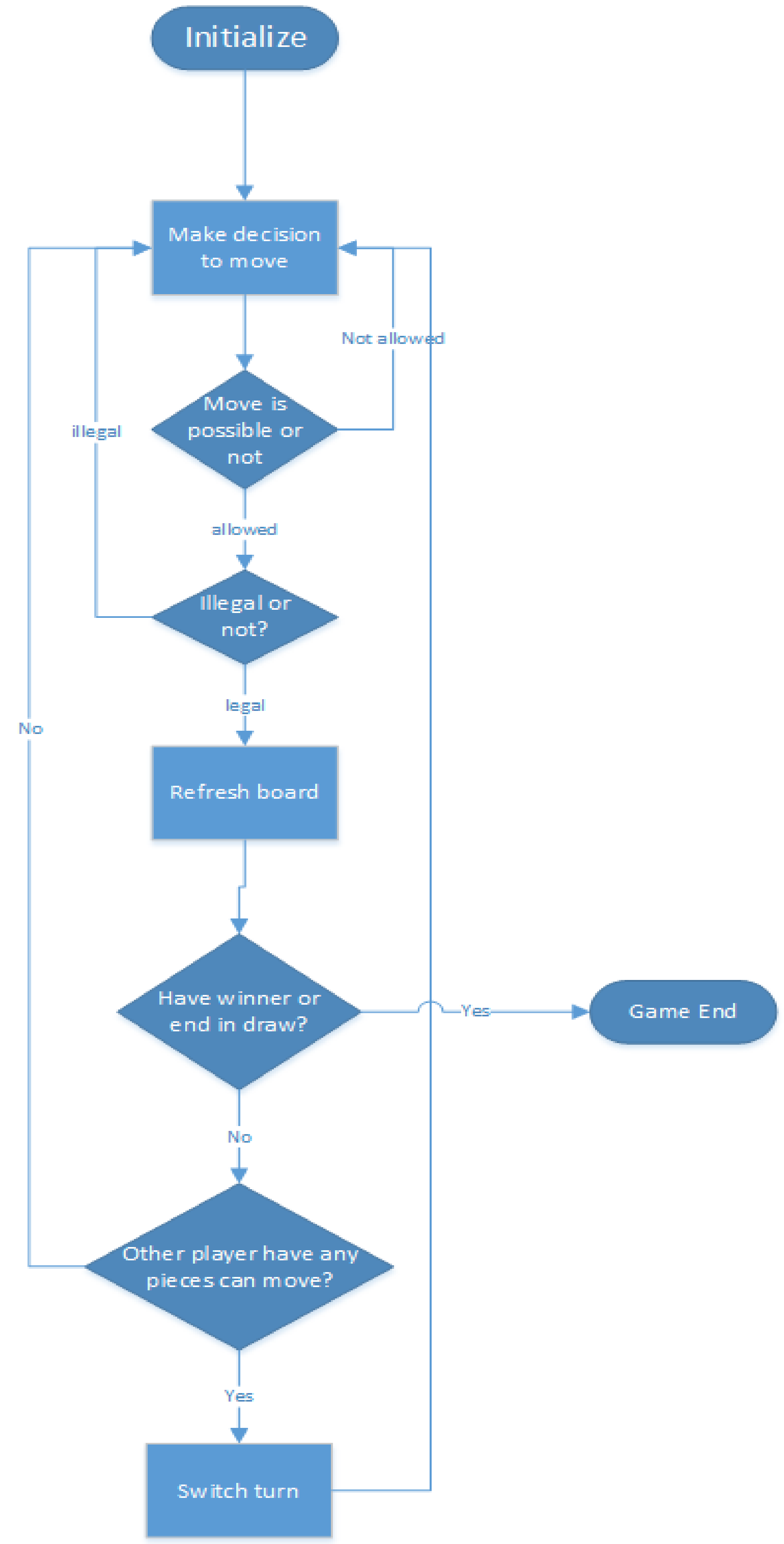
After player make decision, they click the movement and the controller will check whether this step is illegal or not. If this step is legal, then the controller will calculate the number of pieces and check whether the player is the winner or not. If it ends in a draw, the game also pop up the result and terminated. If the winner or loser not given at this step, the board will refresh all the pieces, including the pieces which have been ate by chessman's movement.

Pseudo code:

```
Class player{

        Name:str

        Color:int

        checkerboard: Checkerboard

        isAI: boolean

        Move(checkerboard, position)

}

Class piece{

        PositionX:int

        PositionY:int

        Color:int

        isKing:Boolean

        setKing()

}

Class board{

        Board:array[]

        lastBoard:array[]

        haveWinner()

}
```

```
Class CheckerGameController {

          Board:board

          CheckerGameController(player1, player2, gametype)

          Initialize()

          Move(color, position, moveto)

          isMoveLegal(board, position, moveto)

          isFinnshed(board)

}
```

The flow chart of it is:


Initialize
Make decision to move
Not allowed
Move is possible or not
illegal
allowed
Illegal or not?
legal
No
Refresh board
Have winner or end in draw?
Yes
Game End
No
Other player have any pieces can move?
Yes
Switch turn