Checkers Game Data Model Concept

Yiqing Wang

yw283

Electrical Engineering & Computer Science

Sample Layout:

The sample layout for the Checkers is shown as follows [1]. I place the (0,0) light color block at the coordinate system's origin.

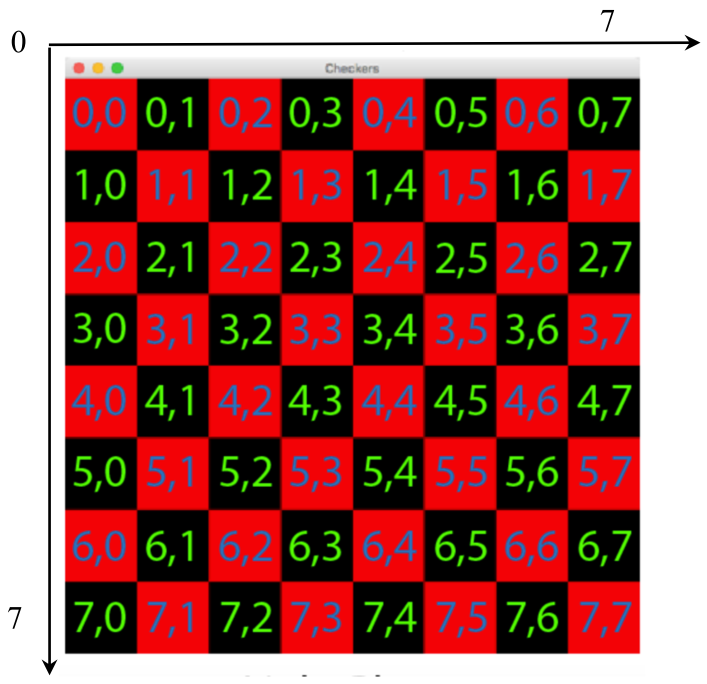

Figure 1. Sample layout for Checkers Game.

General Rules:

1. The player with the black checkers always goes first.
2. Only the dark squares of the checkered board are used.
3. A piece may move only diagonally into an unoccupied square.
4. A player wins by capturing all of the opposing player's pieces or by leaving the opposing player with no legal moves.
5. The game ends in a draw, if neither side can force a win.

Light Player

A **move** is considered to be legal if the row and column can satisfy the following rules:

1. Current Row – Previous Row = +1
2. Current Column – Previous Column = +1/-1
3. 0 <= Current & Previous Row <= Total Rows
4. 0 <= Current & Previous Column <= Total Columns

A **jump** is considered to be legal if it can satisfy the following rules:

1. Light Check is located in the diagonal space nearest to a dark checker.
2. A space on the other side of dark checker has to be empty.
3. Current Row – Previous Row = +2
4. Current Column – Previous Column = 2 * (Opponent's Checker Column - Previous Column)

Dark Player

A **move** is considered to be legal if the row and column can satisfy the following rules:

1. Current Row – Previous Row = -1
2. Current Column – Previous Column = +1/-1
3. 0 <= Current & Previous Row <= Total Rows
4. 0 <= Current & Previous Column <= Total Columns

A **jump** is considered to be legal if it can satisfy the following rules:

1. Dark Check is located in the diagonal space nearest to a light checker.
2. A space on the other side of light checker has to be empty.
3. Current Row – Previous Row = -2
4. Current Column – Previous Column = 2 * (Opponent's Checker Column - Previous Column)

Kings:

1. A light checker becomes a king once its row is equal to 0.
2. A dark checker becomes a king once its row is equal to Total Rows – 1.
3. King can move forward and backward.
4. King can move forward and backward on the same turn if it captures checkers.

Data Representation:

PlayerType: Player

LightPlayer: Player

DarkPlayer: Player

CheckersNum: int

TurnNum: int

PreviousPosition: (int, int)

CurrentPosition: (int, int)

AvailablePosition: (int, int)

CheckersType: boolean (kings or men)

JumpAction: jump

MoveAction: move

GameResult: int (win or lose or draw)

Methods:

setPlayerType (Player)

getPlayerType (): Player

setLightPlayer (Player)

setDarkPlayer (Player)

getCheckersNum (): int

setTurnNum (int)

getTurnNum (): int

set PreviousPosition (int, int)

getPreviousPosition (): (int, int)

setCurrentPosition (int, int)

getCurrentPosition (): (int, int)

getAvailablePosition (): (int, int)

setCheckersType (boolean)

getCheckersType (): Boolean

isJumpAction (): boolean

isMoveAction (): boolean

makeJumpAction (jump)

makeMoveAction (move)

setGameResult (int)

getGameResult (): int