BCIT
Comp 3951
Technical Programming Option
Option Head Mirela Gutica
Winter 2021

Mark: _____ /100

ChessGame
Project Specifications
[by CodePlay]

Eddie Xu
Jeffery M Joseph
Samuel Park

BCIT - ChessGame Project Specifications

Description:

Genre
The game belongs to the abstract strategy genre, characterized by a focus on strategic skill and tactical prowess, devoid of reliance on a simulation of real-world events.

Historical Context
Chess boasts a venerable lineage, tracing its origins back to 6th-century India, from whence it proliferated across Persia, the Islamic world, and subsequently, Europe. The modern iteration of chess, governed by a codified set of rules established in the 19th century, represents the culmination of the game's evolutionary trajectory.

Objective
The paramount objective in chess is to orchestrate a scenario wherein the adversary's king is under inescapable threat of capture—known as checkmate. This condition signifies the opponent's king is placed under direct threat, with no lawful maneuvers available to extricate it from peril.

Available Actions
Within the framework of your chess game, participants are empowered to execute a variety of strategic actions, as delineated by the canonical rules of chess:

Each piece is endowed with unique movement capabilities:
Pawns advance one square forward, or an option to move two squares. Their capture is executed diagonally.
Rooks traverse any number of squares along either a row or column.
Knights employ an L-shaped trajectory: two squares in one direction followed by a perpendicular shift of one square.
Bishops navigate any number of squares diagonally.
Queens amalgamate the movements of rooks and bishops, commanding any number of squares along a row, column, or diagonal.
Kings move a single square in any direction.
Capturing opponent pieces is achieved by relocating one's piece to a square occupied by an opposing piece.

Conclusion Criteria
The game reaches its denouement upon delivering a checkmate to the opponent's king, heralding a victory. Resignation by a player also serves as a valid conclusion, signaling an acknowledgment of an inevitable defeat.

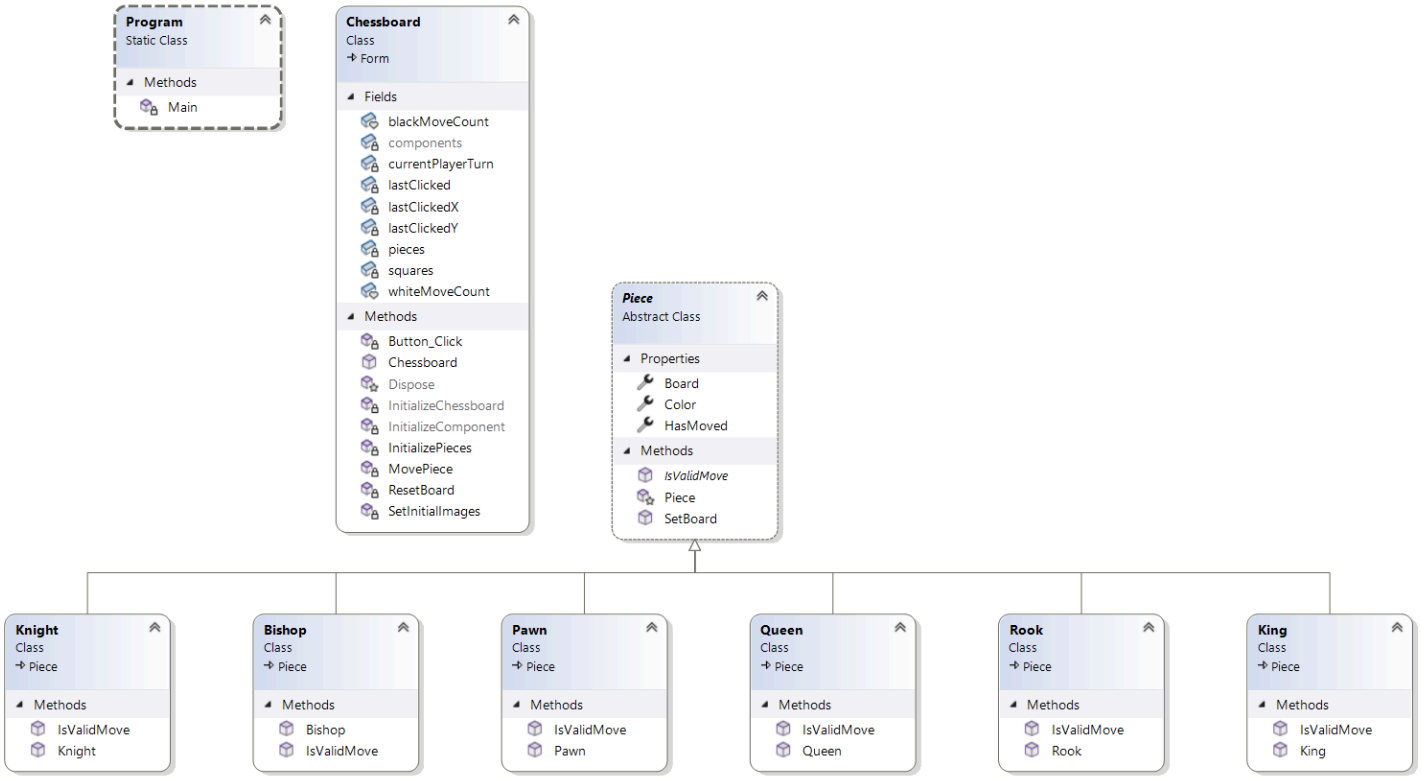
Overview
The Chess Game Project ingeniously encapsulates the quintessence of chess within a digital milieu, facilitating a two-player interactive experience steeped in strategic and tactical deliberation. Through a meticulously crafted graphical interface, the application simulates the chessboard environment, enabling intuitive player interaction through mouse-based inputs. This digital rendition meticulously adheres to the established rules governing piece movements, turn sequencing, and the critical criteria for checkmate, thereby ensuring an authentic chess-playing experience that resonates with both novice and seasoned players alike.

Interface

Original chess game:

Pieces:

Class diagram



Game Engine:

```
Game Initialization ->
function InitializeGame
  InitializeChessboard()
  InitializePieces()
  SetInitialImages()
  currentPlayerTurn = "white"
  whiteMoveCount = 0
  blackMoveCount = 0
end function


Initialize Chessboard ->
function InitializeChessboard
  for each square in chessboard
    Set square color based on position (i + j) % 2
    Attach click event handler Button_Click to square
  end for
end function
function InitializePieces
  Place all pieces in their initial positions
  for each piece position
    Create Piece instance (Pawn, Rook, Knight, Bishop, Queen, King) with appropriate color
    Assign piece to its starting square
    Set background image of square based on piece type and color
  end for
end function


Initialize Pieces ->
function InitializePieces
  Place all pieces in their initial positions
  for each piece position
    Create Piece instance (Pawn, Rook, Knight, Bishop, Queen, King) with appropriate color
    Assign piece to its starting square
    Set background image of square based on piece type and color
  end for
end function



Player Interaction ->
function Button_Click(sender, eventArgs)
  Identify clicked square and retrieve its location
  If first click and square contains piece of current player's color
    Highlight square
    Store reference to clicked square and piece
  ElseIf second click and valid move
    Move piece to target square
```

```
    Update board and GUI accordingly
    Switch currentPlayerTurn
    Check for checkmate or draw conditions
    If game over
      Display winner or draw message
      ResetBoard() or prompt for new game
    Else
      Continue game
    End If
  End If
end function


Move Piece ->
function MovePiece(targetX, targetY)
  Validate move based on piece rules
  If move is valid
    Update piece position in internal board representation
    If capture occurs, remove captured piece
    Update GUI to reflect new piece positions
    Increment move count for current player
    Switch currentPlayerTurn
  End If
end function


End Game ->
function CheckEndGameConditions
  If checkmate
    Display winner
    ResetBoard() or prompt for new game
  ElseIf draw (stalemate, insufficient material, threefold repetition, fifty-move rule)
    Display draw message
    ResetBoard() or prompt for new game
  End If
end function


Game Rest ->
function ResetBoard
  Clear board both visually and internal representation
  InitializePieces()
  currentPlayerTurn = "white"
  Reset move counts
end function
```

Testing plan:

Properties: appearance and general issues:

Properties	Expected Results	Result
GameField: Appearance, reset, draw, pieces presence, fullness	Upon game startup, the game field should display the board and pieces in their initial positions. It should reset after each game ends.	Under implementation
Block: Each block types movement	The game should generate the 7 shapes of blocks The blocks should rotate and move in the game area The blocks should not overlap on collision The blacks should stop at the margin of the game area	Completed
Player	The correct player's turn should be displayed. Moves should only be allowed for the current player. The end of the game should be detected properly with the correct win message.	Under Implementation

Testing plan based on task analysis

Scenarios:

- User launches the game

Case	Expected Results	Result
Bishop-piece initialised	Bishop successfully initialised.	Valid
Is Bishop move valid?	Bishop move is valid.	Valid
King-piece initialised?	King successfully initialised..	Valid
Is King move valid?	King move is valid.	Valid
Knight-piece initialised?	Knight successfully initialised.	Valid
Is Knight move valid?	Knight move is valid.	Valid

Pawn-piece initialised?	Pawn successfully initialised.	Valid
Is Pawn move valid?	Pawn move is valid.	Invalid
Queen-piece initialised?	Queen successfully initialised.	Valid
Is Queen move valid?	Queen move is valid.	Valid
Rook-piece initialised?	Rook successfully initialised.	Valid
Is Rook move valid?	Rook move is valid.	Valid