King Me

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 board.board Class Reference

Public Member Functions

- def __init__ (self, height, width, firstPlayer)

  Initialize the board with the default state.
- def resetBoard (self, height, width, firstPlayer)

  Used to reset the board to the original state.
- def updateBoard (self)

  Updates the board with the current state after a move is made by the user or AI.

Public Attributes

- **width**
- **height**
- **blacklist**
- **whitelist**
- **boardState**
- **gameWon**
- **turn**
- **maxDepth**

Static Public Attributes

- int **RED** = 1
- int **WHITE** = 0
- int **NOTDONE** = -1

3.1.1 Constructor & Destructor Documentation

3.1.1.1 __init__()

```
def board.board.__init__ (
              self,
              height,
              width,
              firstPlayer )
```

Initialize the board with the default state.

Parameters

height	The height of the board
width	The width of the board
firstPlayer	The current turn (who goes first)

```
    Constructs a board, right now maxDepth is statically assigned
```

3.1.2 Member Function Documentation

3.1.2.1 resetBoard()

```
def board.board.resetBoard (
              self,
              height,
              width,
              firstPlayer )
```

Used to reset the board to the original state.

Parameters

height	The height of the board
width	The width of the board
firstPlayer	The current turn (who goes first)

3.1.2.2 updateBoard()

```
def board.board.updateBoard (
              self )
```

Updates the board with the current state after a move is made by the user or AI.

```
    Updates the array containing the board to reflect the current state of the pieces on the
    board
```

The documentation for this class was generated from the following file:

- board.py

3.2 GUI.GUI Class Reference

Public Member Functions

- def __init__ (self)

  The init method loads the graphics, sets the dimensions of the board and calls make_display()
- def make_display (self)

  make_display() creates the screen, sets the caption and adds the buttons to the screen
- def display_board (self, board_state)

  Given the state of a board, displays the board on the screen.
- def update_message (self, message)

  Sets the message to be displayed.
- def display_piece (self, colour, row, col)

  Displays a piece of the colour given, in the row and collumn given.
- def calc_pos (self, row, col)

  Calculates the position on the screen of the top left corner of the square given.
- def get_clicked_object (self, pos)

  get_clicked_object() is passed the position of a mouseclick and returns what was clicked on
- def get_square_clicked (self, pos)

  get_square_clicked() is called when the user clicks on the board.

Public Attributes

- **board_img**
- **red_piece**
- **white_piece**
- **new_game_button**
- **tutorial_button**
- **board_height**
- **board_width**
- **num_cols**
- **num_rows**
- **screen**
- **message**

3.2.1 Member Function Documentation

3.2.1.1 calc_pos()

```
def GUI.GUI.calc_pos (
              self,
              row,
              col )
```

Calculates the position on the screen of the top left corner of the square given.

This function will be used by display piece to determine where on the screen to place the image

Parameters

row	The row number of the square
col	The collumn number of the square

Returns

(x,y) the coordinates of the top left corner of the square on the screen

3.2.1.2 display_board()

```
def GUI.GUI.display_board (
              self,
              board_state )
```

Given the state of a board, displays the board on the screen.

Loops through the board_state and calls display_piece to display the pieces

Parameters

board_state	Two dimensional array representing the state of the board

3.2.1.3 display_piece()

```
def GUI.GUI.display_piece (
              self,
              colour,
              row,
              col )
```

Displays a piece of the colour given, in the row and collumn given.

Parameters

colour	The colour of the piece to be displayed
row	The row to display the piece
col	The collumn to display the piece

3.2.1.4 get_clicked_object()

```
def GUI.GUI.get_clicked_object (
              self,
              pos )
```

get_clicked_object() is passed the position of a mouseclick and returns what was clicked on

Parameters

pos	The tuple representing the mouseclick location on the screen

Returns

A string indicating what was clicked

3.2.1.5 get_square_clicked()

```
def GUI.GUI.get_square_clicked (
              self,
              pos )
```

get_square_clicked() is called when the user clicks on the board.

When the user clicks on the board the tuple containing the row and collumn of the corresponding square clicked on is returned

Parameters

pos	The position of the mouseclick

Returns

(col,row) The row and collumn corresponding to the square the User clicked on

3.2.1.6 update_message()

```
def GUI.GUI.update_message (
              self,
              message )
```

Sets the message to be displayed.

Parameters

message	The message to be displayed

The documentation for this class was generated from the following file:

- GUI.py

3.3 menu.menu Class Reference

Public Member Functions

- def tutorial (self)

    Class to display the message box (utilizing Tinker library) for the Tutorial once called upon.
- def newgame (self, board, width, height, firstPlayer)

    Resets the game by reseting the pieces back to original spots.

3.3.1 Member Function Documentation

3.3.1.1 newgame()

```
def menu.menu.newgame (
              self,
              board,
              width,
              height,
              firstPlayer )
```

Resets the game by reseting the pieces back to original spots.

Parameters

board	The game board to be reset
width	The width of the game board
height	The height of the game baord
firstPlayer	The ID of the player

The documentation for this class was generated from the following file:

- menu.py

3.4 pieces.pieces Class Reference

Public Member Functions

- def __init__ (self, row, col, color)

    Class to represent a checkers piece.
- def make_king (self)

    Represent a checkers king piece.
- def move (self, row, col)

    Move the checkers piece to the desired location.

Public Attributes

- **row**
- **col**
- **color**
- **king**

3.4.1 Constructor & Destructor Documentation

3.4.1.1 __init__()

```
def pieces.pieces.__init__ (
              self,
              row,
              col,
              color )
```

Class to represent a checkers piece.

Parameters

row	The row location of the piece.
col	The column location of the piece.
color	The color of the piece.

3.4.2 Member Function Documentation

3.4.2.1 move()

```
def pieces.pieces.move (
              self,
              row,
              col )
```

Move the checkers piece to the desired location.

Parameters

row	The row location of the move.
col	The column location of the move.

The documentation for this class was generated from the following file:

- pieces.py

Chapter 4

File Documentation

4.1 board.py File Reference

Class to represent the board and handle the state of the board of the checkers game.

Classes

- class board.board

4.1.1 Detailed Description

Class to represent the board and handle the state of the board of the checkers game.

Author

Carson Wilcox, Ardhendu, Dylan, Thaneegan

Date

March 16th 2021

4.2 gameLogic.py File Reference

Provides logic for game (including making moves)

Functions

- def gameLogic.iterWhiteMoves (board)

  Iterates through the white pieces array to generate moves for white pieces.
- def gameLogic.iterBlackMoves (board)

  Iterates through the black pieces array to generate moves for black pieces.
- def gameLogic.iterWhitePiece (board, piece)

  Checks for possible moves for the selected piece.
- def gameLogic.iterBlackPiece (board, piece)

  Checks for possible moves for the selected piece.
- def gameLogic.iterBoth (board, piece, moves)

  Checks for possible moves for the selected piece.
- def gameLogic.moveSilentBlack (board, moveFrom, moveTo, winLoss)

  Moves a black piece, calls the updateboard method, determines if black piece player has won or loss (sets the turn to white after)
- def gameLogic.moveSilentWhite (board, moveFrom, moveTo, winLoss)

  Moves a white piece, calls the updateboard method, determines if white piece player has won or loss (sets the turn to black after)
- def gameLogic.moveBlack (board, moveFrom, moveTo, winLoss)

  Moves a black piece, calls the updateboard method, determines if black piece player has won or loss (sets the turn to white after)
- def gameLogic.moveWhite (board, moveFrom, moveTo, winLoss)

  Moves a white piece, calls the updateboard method, determines if white piece player has won or loss (sets the turn to black after)

4.2.1 Detailed Description

Provides logic for game (including making moves)

Author

Carson Wilcox, Thaneegan, Dylan, Ardhendu

Date

3/17/2021

4.2.2 Function Documentation

4.2.2.1 iterBlackMoves()

```
def gameLogic.iterBlackMoves (
              board )
```

Iterates through the black pieces array to generate moves for black pieces.

Parameters

board	The state of the current board

Returns

Returns the location of the possible move of the selected piece

```
Main Generator for black moves
```

4.2.2.2 iterBlackPiece()

```
def gameLogic.iterBlackPiece (
              board,
              piece )
```

Checks for possible moves for the selected piece.

Parameters

board	The state of the current board
piece	The piece to be moved

Returns

Returns the location of the valid move of the selected piece

```
Generates possible moves for a black piece
```

4.2.2.3 iterBoth()

```
def gameLogic.iterBoth (
              board,
              piece,
              moves )
```

Checks for possible moves for the selected piece.

Parameters

board	The state of the current board
piece	The piece to be moved
moves	The set of possible moves

Returns

Returns the location of the valid moves of the selected piece

```
    Handles the actual generation of moves for either black or white pieces
```

4.2.2.4 iterWhiteMoves()

```
def gameLogic.iterWhiteMoves (
              board )
```

Iterates through the white pieces array to generate moves for white pieces.

Parameters

board	The state of the current board

Returns

Returns the location of the possible move of the selected piece

```
    Main generator for white moves
```

4.2.2.5 iterWhitePiece()

```
def gameLogic.iterWhitePiece (
              board,
              piece )
```

Checks for possible moves for the selected piece.

Parameters

board	The state of the current board
piece	The piece to be moved

Returns

Returns the location of the possible move of the selected piece

```
    Generates possible moves for a white piece
```

4.2.2.6 moveBlack()

```
def gameLogic.moveBlack (
              board,
              moveFrom,
              moveTo,
              winLoss )
```

Moves a black piece, calls the updateboard method, determines if black piece player has won or loss (sets the turn to white after)

Parameters

board	The state of the current board
moveFrom	The location of the piece to be moved
moveTo	The location of where the piece has to be moved to
winLoss	The state of the game

```
    Move a black piece from one spot to another. \n winLoss is passed as either 0(white)
    or 1(black) if the move is a jump
```

4.2.2.7 moveSilentBlack()

```
def gameLogic.moveSilentBlack (
              board,
              moveFrom,
              moveTo,
              winLoss )
```

Moves a black piece, calls the updateboard method, determines if black piece player has won or loss (sets the turn to white after)

Parameters

board	The state of the current board
moveFrom	The location of the piece to be moved
moveTo	The location of where the piece has to be moved to

```
    Move black piece without printing
```

4.2.2.8 moveSilentWhite()

```
def gameLogic.moveSilentWhite (
              board,
              moveFrom,
              moveTo,
              winLoss )
```

Moves a white piece, calls the updateboard method, determines if white piece player has won or loss (sets the turn to black after)

Parameters

board	The state of the current board
moveFrom	The location of the piece to be moved
moveTo	The location of where the piece has to be moved to

```
    Move white piece without printing
```

4.2.2.9 moveWhite()

```
def gameLogic.moveWhite (
              board,
              moveFrom,
              moveTo,
              winLoss )
```

Moves a white piece, calls the updateboard method, determines if white piece player has won or loss (sets the turn to black after)

Parameters

board	The state of the current board
moveFrom	The location of the piece to be moved
moveTo	The location of where the piece has to be moved to
winLoss	The state of the game

```
    Move a white piece from one spot to another. \n winLoss is passed as either 0(white)
    or 1(black) if the move is a jump
```

4.3 GUI.py File Reference

Following module handles the graphical user interface for the checkers game.

Classes

- class GUI.GUI

Variables

- tuple **GUI.screen_dimensions** = (1060, 720)

4.3.1 Detailed Description

Following module handles the graphical user interface for the checkers game.

Author

Dylan, Thaneegan, Ardhendu

Date

March 3 2021

4.4 menu.py File Reference

This class represents the Menu for the game. The menu contains the functionality for Tutorial and New Game.

Classes

- class menu.menu

4.4.1 Detailed Description

This class represents the Menu for the game. The menu contains the functionality for Tutorial and New Game.

Author

Ardhendu, Dylan, Thaneegan

Date

March 15th 2021

4.5 minmax.py File Reference

Provides the minmax functionality for AI as well as static evaluation.

Functions

- def minmax.is_won (board)

  Function to check if game is won.
- def minmax.minMax2 (board)

  Main minimax algorithm function for AI.
- def minmax.maxMove2 (maxBoard, currentDepth)

  Function to calculate best move for AI.
- def minmax.minMove2 (minBoard, currentDepth)

  Function to calculate and predict best move from perspective of the player.
- def minmax.maxMinBoard (board, currentDepth, bestMove)

  Function to calculate and predict best move from perspective of the player.
- def minmax.staticEval2 (evalBoard)

  Function to evaluate the board state and which player is favoured to win to assess AI move.

4.5.1 Detailed Description

Provides the minmax functionality for AI as well as static evaluation.

Author

Carson Wilcox, Thaneegan, Dylan, Ardhendu

Date

3/17/2021

4.5.2 Function Documentation

4.5.2.1 is_won()

```
def minmax.is_won (
              board )
```

Function to check if game is won.

Function takes in board object and returns bool

Parameters

board	board object cotaining the state

4.5.2.2 maxMinBoard()

```
def minmax.maxMinBoard (
              board,
              currentDepth,
              bestMove )
```

Function to calculate and predict best move from perspective of the player.

Function takes in board object as well as currentdepth to return best move for the AI

Parameters

maxBoard	board object with final state after completeing all the possible moves
currentDepth	depth for the AI to predict best player move

4.5.2.3 maxMove2()

```
def minmax.maxMove2 (
              maxBoard,
              currentDepth )
```

Function to calculate best move for AI.

Function takes in board object as well as currentdepth to return best move for the AI

Parameters

maxBoard	board object with final state after completeing all the possible moves
currentDepth	depth for the AI to predict best move

4.5.2.4 minMax2()

```
def minmax.minMax2 (
              board )
```

Main minimax algorithm function for AI.

Function takes in board object and returns most appropirate move for the AI

Parameters

board	board object cotaining the state

4.5.2.5 minMove2()

```
def minmax.minMove2 (
              minBoard,
              currentDepth )
```

Function to calculate and predict best move from perspective of the player.

Function takes in board object as well as currentdepth to return best move for the AI

Parameters

maxBoard	board object with final state after completeing all the possible moves
currentDepth	depth for the AI to predict best player move

4.5.2.6 staticEval2()

```
def minmax.staticEval2 (
              evalBoard )
```

Function to evaluate the board state and which player is favoured to win to assess AI move.

Parameters

evalBoard	board that needs to be evaluated

4.6 pieces.py File Reference

Class to represent a checkers piece.

Classes

- class pieces.pieces

4.6.1 Detailed Description

Class to represent a checkers piece.

Author

Ardhendu, Dylan, Thaneegan

Date

March 15th 2021

Index