Stage 1 Software - Checkmate

[Proposal 2](#_Toc515601762)

[Overview 2](#_Toc515601763)

[Basic Gameplay 2](#_Toc515601764)

[Time Controls 2](#_Toc515601765)

[Number of players 2](#_Toc515601766)

[Ending the Game 3](#_Toc515601767)

[Check and Checkmate 3](#_Toc515601768)

[Win on Time 3](#_Toc515601769)

[Resignation 3](#_Toc515601770)

[Stalemate 3](#_Toc515601771)

[Three-fold Repetition 3](#_Toc515601772)

[Fifty-move Rule 3](#_Toc515601773)

[Insufficient Material 3](#_Toc515601774)

[Pieces and their Moves 4](#_Toc515601775)

[Pawn 4](#_Toc515601776)

[Knight 4](#_Toc515601777)

[Bishop 4](#_Toc515601778)

[Rook 5](#_Toc515601779)

[Queen 5](#_Toc515601780)

[King 5](#_Toc515601781)

[Special Moves and Situations 6](#_Toc515601782)

[Castling 6](#_Toc515601783)

[En passant 6](#_Toc515601784)

[Promotion 6](#_Toc515601785)

[Screen designs 7](#_Toc515601786)

[Structure Chart 9](#_Toc515601787)

[Subs and Functions 11](#_Toc515601788)

# 

# Proposal

## Overview

Chess is a game that has been played for centuries by millions of people all over the world. It is a game of strategy, in which two teams play against each other using six unique pieces on an eight by eight grid board. The game is believed to have originated in India, and has spread throughout the world since then. The objective of chess is to ‘Checkmate’ the other opponent. This is done by trapping the opposing king in a place where it cannot move. Below is an overview of the possible moves of all the pieces.

## Basic Gameplay

Each side starts of with their pieces in order on both sides of the board. When the game starts, the white side moves any one of their pieces first. Then the black side moves, and the moves alternate until the game ends. Each piece can move in different ways, shown in detail below in the ‘Pieces and their Moves’ section, and has the ability to capture the other sides pieces.

## Time Controls

Many chess games are played with time controls. Time controls mean limited amounts of time for each player. The time controls can vary for each game. A very quick game might be played with a one-minute time control. This means that every time it is a player’s turn, the player’s clock ticks down however long the player takes to decide his move. While one player’s clock is counting down, the other player’s clock is paused.

There is also something known as ‘increment’. This means that every time a player makes a move in a game, however much the increment is set at is added on to his time. For example, in a game with one-minute time controls with a twelve second increment, every time a player makes a move 12 seconds is added onto his time. Often time controls are expressed as <total-time> + <increment>. For example, a five minute game with a three second increment would be expressed at 5 + 3.

## Number of players

Chess is game played with two sides. This means it is a two player, but not necessarily between two humans, as computer AI’s can play chess. All chess tournaments are played against different matches between two humans.

# 

## Ending the Game

### Check and Checkmate

When a King is being attacked by another piece, or in danger of being captured the next move, the King is in ‘check’. The King has to get out of check. This can be in three ways - moving out of check, capturing the piece holding the king in check, or blocking the check with another piece. In check cannot be blocked in the case of a knight holding the king in check. When neither of these three moves are possible, the King is in checkmate and the opponent wins.

### Win on Time

Some chess games are played on time control, which means each player only has a limited amount of time to play. Once this allotted time runs out for a player, the other opponent wins.

### Resignation

A player may resign a game, conceding the game to the other opponent.

### Stalemate

The player whose turn it is has no legal move and is not in check.

### Three-fold Repetition

If three occurrences of a position occur either player can claim a draw, but this is not compulsory.

### Fifty-move Rule

If during the past 50 moves no pawn has moved or piece been captured, either player can claim a draw.

### Insufficient Material

If neither player has enough pieces to checkmate the opponent then the game is a draw on insufficient material.

# 

## Pieces and their Moves

### Pawn

|  |  |  |
| --- | --- | --- |
|  |  | The Pawn can move forward two places at its starting rank, but can otherwise only move forward one place. Pawns can only capture pieces diagonally. Pawns have two special moves - En passant captures and Promotions. These are discussed in more detail in the ‘Special Moves and Situations’ section. |

### Knight

|  |  |  |
| --- | --- | --- |
|  |  | The Knight is the only piece that can jump over other pieces. It moves in an L-shape, two squares horizontally in any direction and then one square vertically in any direction, or two squares vertically in any direction and then one square horizontally in any direction. It captures pieces by landing on them. |

### Bishop

|  |  |  |
| --- | --- | --- |
|  |  | The Bishop can move any amount of pieces diagonally, but cannot move over other players. It captures pieces by landing on them. |

### Rook

|  |  |  |
| --- | --- | --- |
|  |  | The Rook can move any number of pieces horizontally and vertically, but cannot move over other players. It captures pieces by landing on them. Along with the King, the Rook can perform a special move called ‘Castling’, discussed in more detail in the ‘Special Moves and Situations’ section below. |

### Queen

|  |  |  |
| --- | --- | --- |
|  |  | The Queen can do every move the bishop and the rook can, meaning it can move vertically and horizontally any number of moves. It cannot jump over any pieces, and captures pieces by landing on them. |

### King

|  |  |  |
| --- | --- | --- |
|  |  | The King is the most important piece, and can move one square in any direction. Along with the Rook, the King can perform a special move called ‘Castling’, discussed in more detail in the ‘Special Moves and Situations’ section below. |

# 

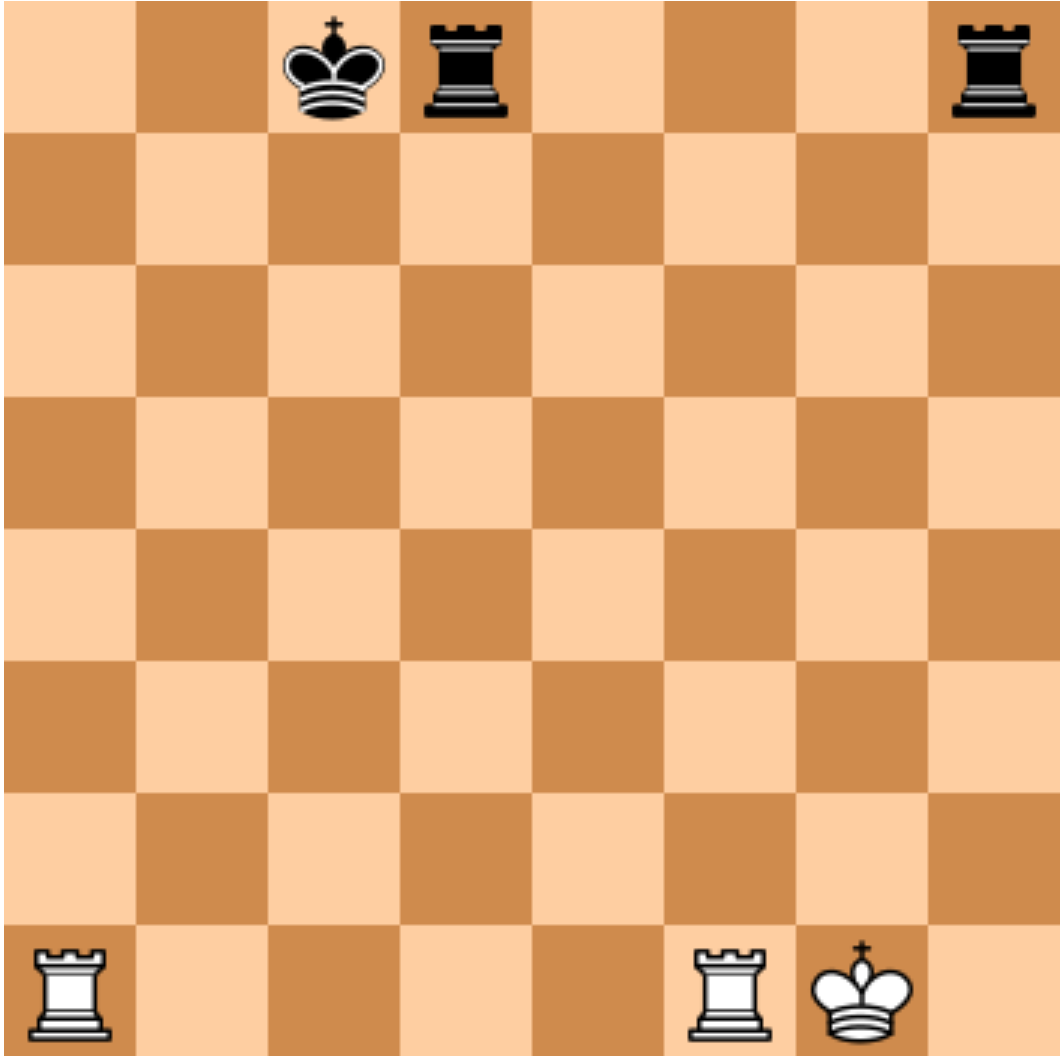
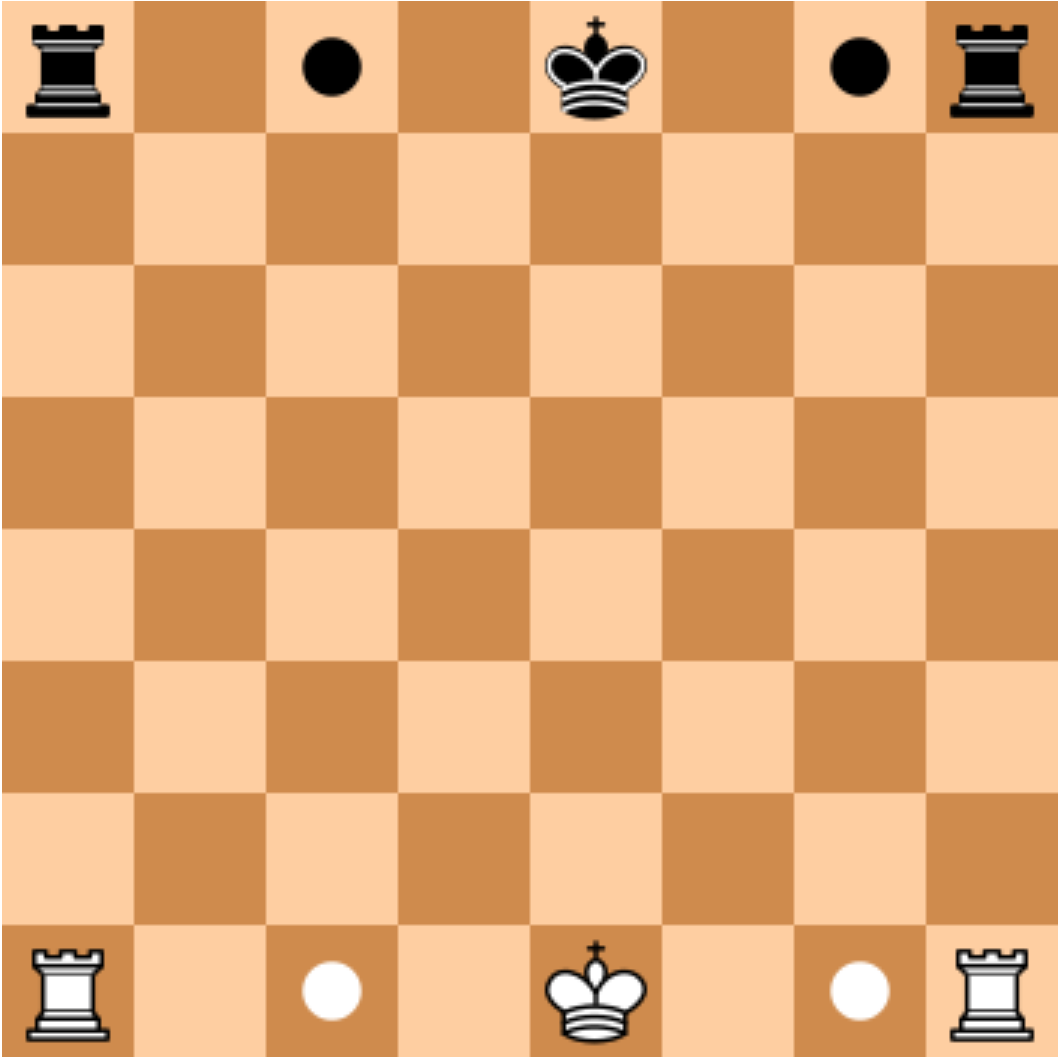
## Special Moves and Situations

Chess has a variety of special moves and situations which vary the gameplay significantly.

### Castling

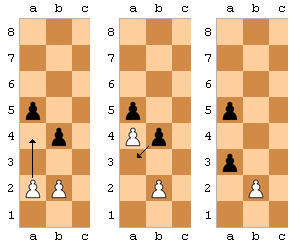
Once a game, the King can ‘castle’ with the Rook. This involves the King moving two places to the involved Rook, and the Rook then moving to the other side of the King.

Kings can castle on either side. The side with the least distance to the rook is called the Kingside and the other side is called Queenside. Below are diagrams of castling on both sides, before and after.



### En passant

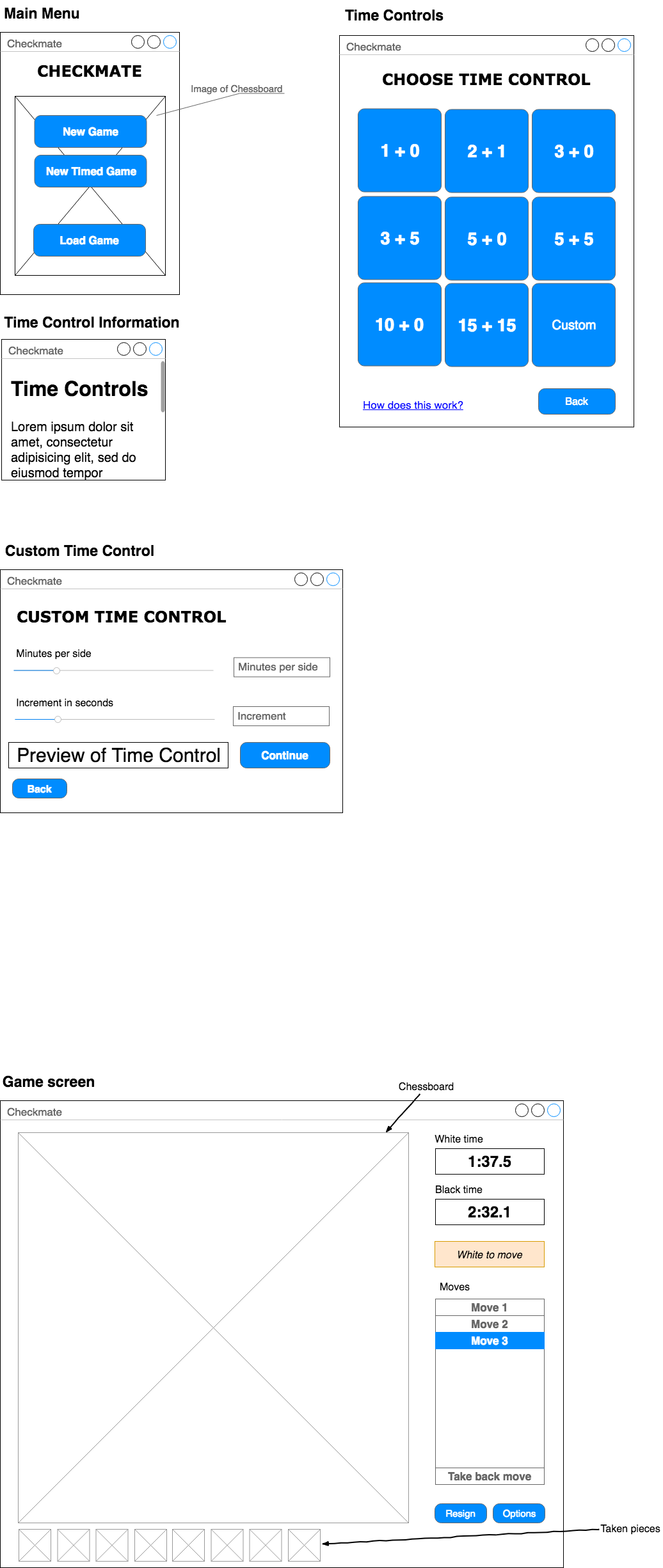
When a pawn moves two squares on its first move and there is an opponent's pawn horizontally adjacent to it, the opponent’s pawn can capture the first pawn *En passant*. This means that the opponent’s pawn can diagonally move behind the first pawn and capture it. However, this move can only be done when the first pawn advances two squares on its first move, and if *En passant* is not done on the very first move it is possible, the right to do so is forfeited. Below is a step by step diagram of a black Pawn executing *En passant*.

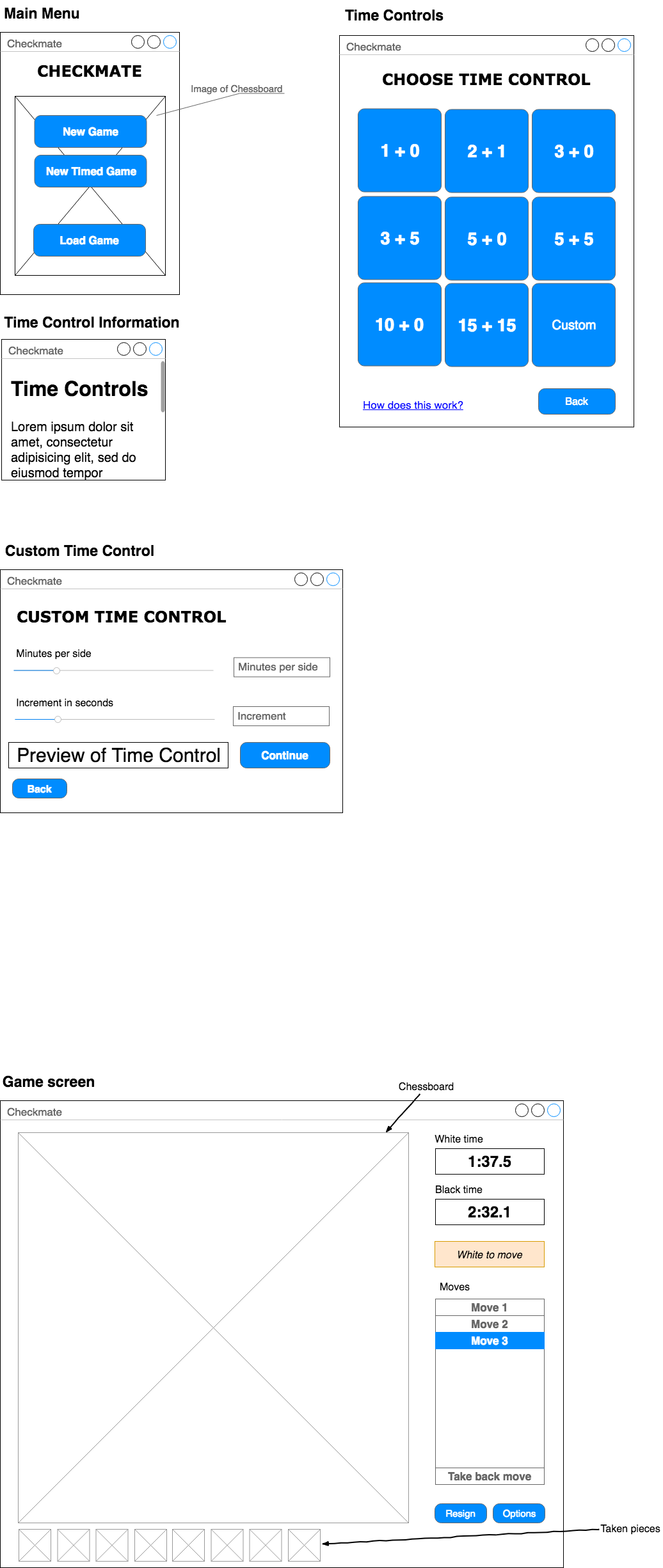


### Promotion

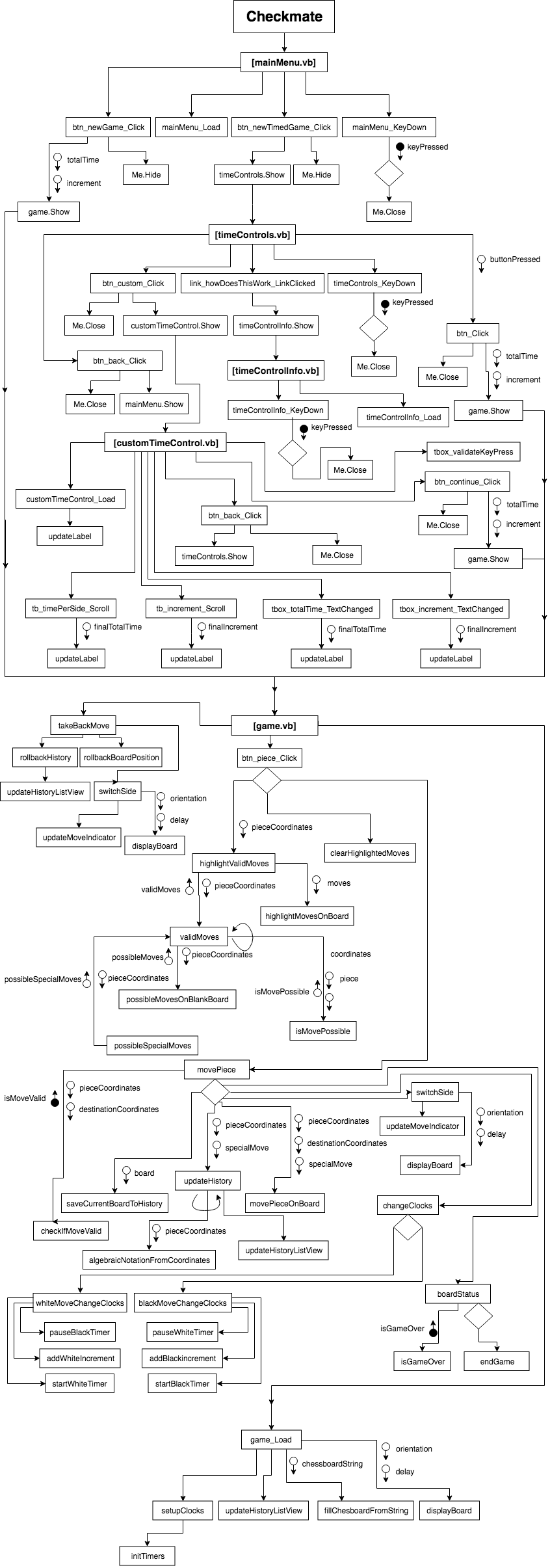
If a pawn moves to the back rank of the board, it can get Promoted. This means that the Pawn can be exchanged for any other piece of the same color. Usually the Pawn is traded for a Queen, but it can be traded for any piece.

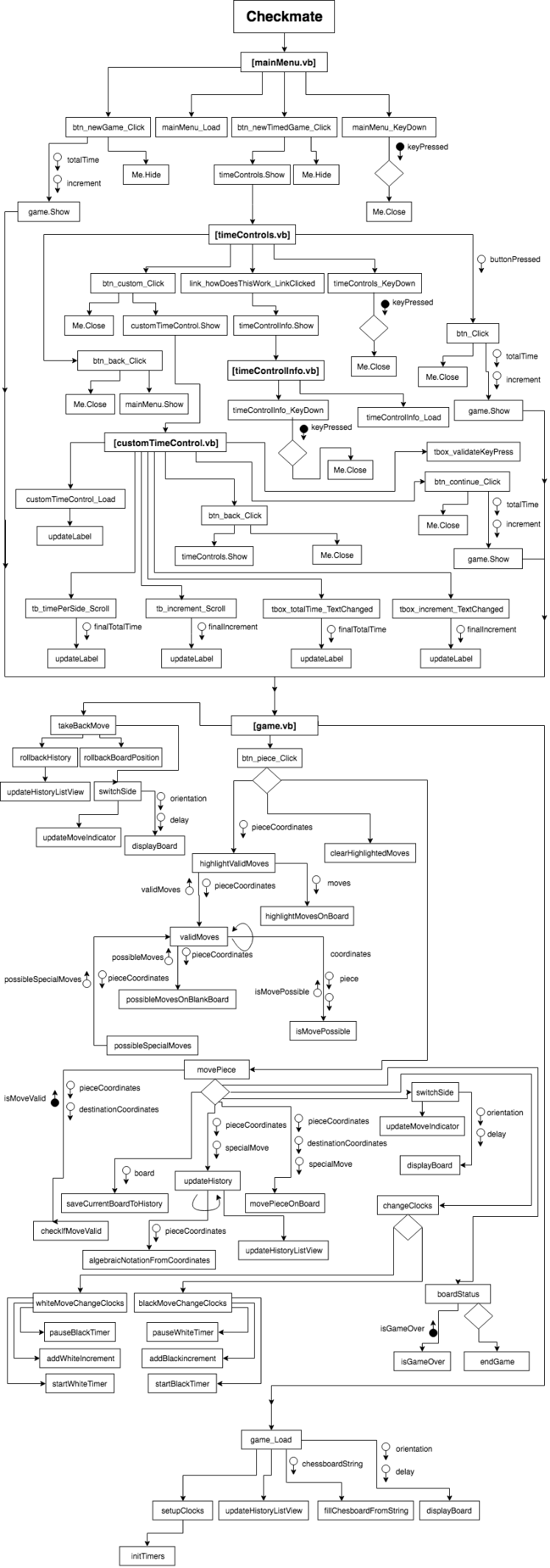
# Screen designs

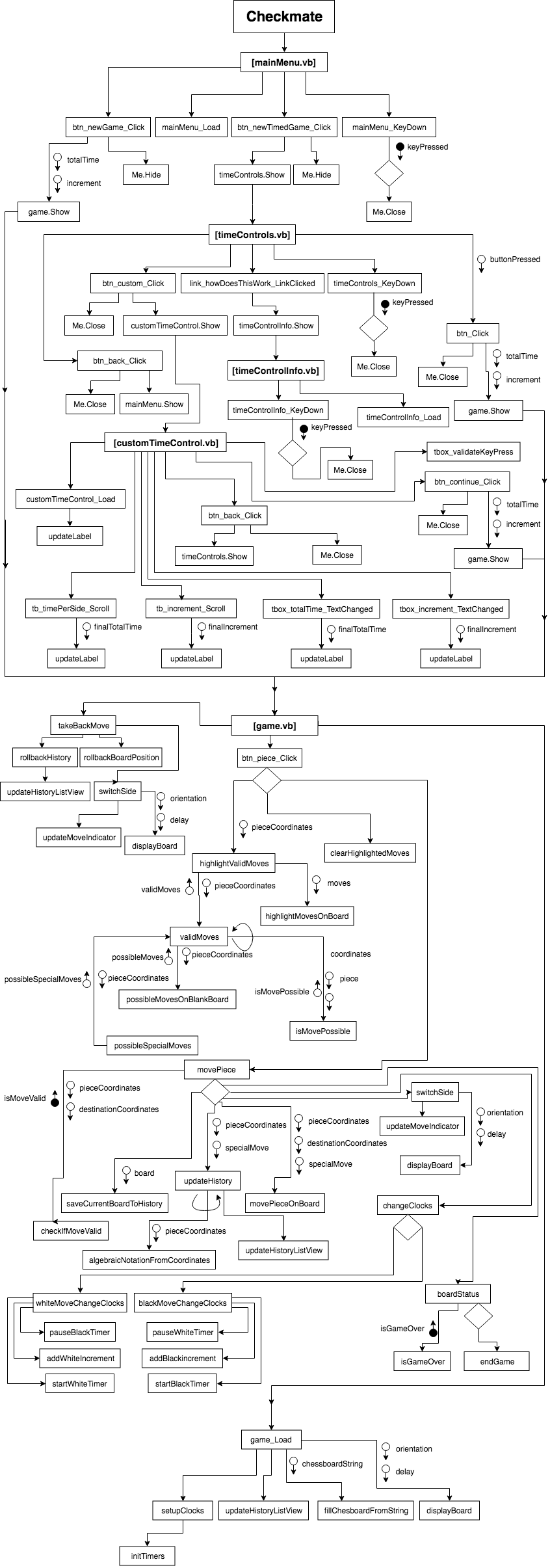




# Structure Chart







# Subs and Functions

|  |  |  |
| --- | --- | --- |
| Name of function | Description | Data passed in and returned |
| **Main menu** | | |
| btn\_newGame\_Click | Hides mainMenu form and shows game form. | None |
| btn\_newTimedGame\_Click | Hides mainMenu form and shows timeControl form. | None |
| mainMenu\_KeyDown | Closes window if escape pressed. | The key that was pressed |
| **Time Controls** | | |
| btn\_custom\_Click | Hides timeControls form and shows customTimeControl form. | None |
| link\_howDoesThisWork\_LinkClicked | Hides timeControls form and shows timeControlInfo form. | None |
| timeControls\_KeyDown | Closes window if escape pressed. | The key that was pressed |
| btn\_Click | Updates minutesPerSide and increment values depending on button clicked. Handles all time control buttons except for custom button. Then hides timeControls form and shows game form. | The Time Control values will be returned.These are returned in a string, in the form “[timecontrol],[increment’]” |
| btn\_back\_Click | Closes Time Control Info and opens Main menu | None |
| **Time Control Info** | | |
| timeControlInfo\_KeyDown | Closes window if escape pressed. | The key that got pressed |
| **Custom Time Control** | | |
| updateLabel | Updates Preview label showing user-made custom Time Control | Uses global Time |
| tb\_timePerSide\_Scroll | Updates time per side value and then calls updateLabel. | None |
| tb\_increment\_Scroll | Updates increment value and then calls updateLabel. | None |
| tbox\_totalTime\_TextChanged | Updates time per side value and then calls updateLabel. | None |
| tbox\_increment\_TextChanged | Updates increment value and then calls updateLabel. | None |
| btn\_continue\_Click | Closes Custom Time Control form and opens game form. | The Time Control values will be returned.These are returned in a string, in the form “[timecontrol],[increment’]” |
| btn\_back\_Click | Closes Custom Time Control form and opens Time Control form. | None |
| **Game** | | |
| btn\_piece\_Click | Called when user clicks on chess piece. Does one of three things - either highlights the valid moves, removes the highlight of the valid moves or if the valid moves are already highlighted, move the piece to the clicked square. | None |
| highlightValidMoves | Calls the validMoves function to get a list of valid moves, and passes that to the highlightMovesOnBoard function to highlight them. | Takes the coordinates of the piece that was clicked as a string in the form ‘x y’, where x and y are coordinates,  and doesn’t return anything. |
| validMoves | Calls possibleMovesOnBlankBoard to get all possible moves for a given piece on a blank board. These moves are then passed along to the isMovePossible function to see if they are valid. Then the possibleSpecialMoves function is called to see if any special moves can be executed (e.g. castling, pawn promotions etc.) All the possible special moves and the valid moves are returned. | Takes coordinates of piece as a string in the form ‘x y’, where x and y are coordinates, and returns valid moves as a string, in the form “x y,x y” where x and y are coordinate pairs. |
| possibleMovesOnBlankBoard | Returns possible moves on blank board for any piece in any position (on a blank board.) Function consists of switch case that does different things based on the piece type. | Takes coordinates of piece as a string in the form ‘x y’, where x and y are coordinates, and returns possible moves on blank board as a string, in the form “x y,x y” where x and y are coordinate pairs. |
| isMovePossible | Checks if there are any pieces obstructing move or anything else that could prevent it from happening and returns the result. | Takes coordinate of piece as a string in the form ‘x y’, where x and y are coordinates, and coordinate of location to be tested. Return a boolean, true if move possible, false if not. |
| possibleSpecialMoves | Function is in the form of a switch statement that looks for possible special moves depending on the piece type. These moves include castling, pawn promotions, en passants and many others. Returns coordinates of possible special moves. | Takes coordinates of piece as a string in the form ‘x y’, where x and y are coordinates, and returns possible special moves as a string, in the form “x y,x y”, where x and y are coordinate pairs. |
| clearHighlightedMoves | Sub that clears highlighted moves from the board | None |
| movePiece | Sub that moves a piece and adds the moved piece to history. First checks if the piece is valid (isMoveValid). Then it saves the current position of the move to the board history, using saveCurrentBoardToHistory. After that, it calls updateHistory to update the history on the side of the game form. Then, it moves the piece using movePieceOnBoard, switches sides using switchSide and changes clocks using changeClocks. After the piece is moved the board status is checked using boardStatus to check whether the game has finished. | None |
| isMoveValid | Calls validMoves and checks whether move is a valid move. | Takes the piece coordinates as a string in the form ‘x y’, where x and y are coordinates, and the destination coordinates and returns a boolean, True if valid, False if not. |
| saveCurrentBoardToHistory | Converts the current board from a 2d array to 64 character string, with different characters representing different pieces. Saves this string in an array. This is used to take back moves later if needed. | None |
| updateHistory | This functions creates an algebraic notation move given a starting piece and a destination square. It takes care of the many different cases of notation possible (castling, checkmate, etc.) If the move is known to be a special move, this can be specified in the ‘special’ parameter as a number, with different numbers representing different special moves.  It uses the function algebraicNotationFromCoordinates to convert from x and y coordinates to chess algebraic notation. After this, it calls updateHistoryListView to visually update the history. | Takes pieceCoordinates as a string in the form ‘x y’, where x and y are coordinates, and special, as an integer, and doesn’t return anything. |
| algebraicNotationFromCoordinates | Converts from x and y coordinates to algebraic notation. | Takes coordinates as a string in the form ‘x y’, where x and y are coordinates, and returns a string in the form ‘x y’, where x is the rank and y is the file. |
| updateHistoryListView | Updates history ListView | None |
| movePieceOnBoard | Takes the piece coordinates, destination square and a parameter ‘special’ (indicating if special move) and moves piece on chessboard. This is done by modifying the 2d array of the chessboard. | Takes piece coordinates as a string in the form ‘x y’, where x and y are coordinates, and destination coordinates as a string in the form ‘x y’.  Also takes the parameter ‘special’ as an integer and doesn’t return anything. |
| switchSide | Calls updatesMoveIndicator and switches board view using displayBoard. | None |
| updateMoveIndicator | Updates the visual move indicator. | None |
| displayBoard | Displays the current 2d array of the board visually. | None |
| changeClocks | Either calls whiteMoveChangeClocks or blackMoveChangeClocks depending on who’s turn it is. The latter is called if it is black’s move, and the first when it is white’s move. | None |
| whiteMoveChangeClocks | Calls pauseBlackTimer, addWhiteIncrement, and startWhiteTimer. | None |
| blackMoveChangeClocks | Calls pauseWhiteTimer, addBlackncrement, and startBlackTimer. | None |
| pauseBlackTimer | Pauses black timer. | None |
| pauseWhiteTimer | Pauses white timer. | None |
| addBlackIncrement | Adds increment value to black’s time. | None |
| addWhiteIncrement | Adds increment value to white’s time. | None |
| startBlackTimer | Starts black’s timer. | None |
| startWhiteTimer | Starts white’s timer. | None |
| boardStatus | Calls isGameOver to check if game is over. If it is, then calls endGame to end the game. | None |
| isGameOver | Checks status of the game to see if it is drawn or a side has lost. Returns integer that indicates status of game, where different numbers indicate different results. | None |
| endGame | Ends the game and indicates visually through change in window title ([black|white] wins!) and a messagebox. | None |
| takeBackMove | Calls rollbackHistory and rollbackBoardPosition to take back last move. Then switchSide is called to switch the side (as one move was taken back the side will be changed.) | None |
| rollbackHistory | Deletes last move of history of moves and calls updateHistoryListView to display the new history. | None |
| rollbackBoardPosition | Uses previously stored board positions to rollback board position one move. | None |
| game\_load | Game initialisation function. Calls setupClocks, updateHistoryListView, fillChessboardFromString and displayBoard. | None |
| setupClocks | Calls initTimers for black and white timers | None |
| initTimers | Gives black and white timer objects handler functions. | None |