**Software Design  
Document**

for

Chess Game

Version 1.0 approved

Prepared by Snake Army (Group 1)

Burmeister Uruchurtu, Alec

Blanco, Bruno

Lockman, Cory

Lemmon, Dylan

Currie Buckner, Ty Baron

Texas State University

<date created>

# Contents

[1 Introduction 1](#_Toc117484244)

[1.1 Purpose 1](#_Toc117484245)

[1.2 System Overview 1](#_Toc117484246)

[1.3 Definitions, Acronyms and Abbreviations 1](#_Toc117484247)

[1.4 Supporting Materials 2](#_Toc117484248)

[1.5 Document Overview 2](#_Toc117484249)

[2 Architecture 2](#_Toc117484250)

[2.1 Overview 2](#_Toc117484251)

[2.2 Component 1..n 2](#_Toc117484252)

[3 High-Level Design 3](#_Toc117484253)

[4 Low-Level Design 4](#_Toc117484254)

[4.1 Modules Overview 4](#_Toc117484257)

[4.2 Module Specifications 4](#_Toc117484258)

[4.2.1 Module X1 4](#_Toc117484259)

[4.2.2 Module X2 5](#_Toc117484260)

[Appendix A – Group Log 6](#_Toc117484261)

# Introduction

* 1. **Purpose**

This program is intended to mimic a game of chess, following standard chess rules including piece movement.

## 1.2 System Overview

Each piece will be its own class, inheriting its functions from a ‘piece’ class. The board will be managed by a database which will be updated by a manager class. The board will update and redraw based on changes to the database.

## 1.3 Definitions, Acronyms and Abbreviations

* Castling – Castling consists of moving the King two squares towards a Rook on the player's first rank, then moving the Rook to the square over which the King crossed. Castling may only be done if the King and Rook involved have not moved this game, the squares between the King and the Rook involved are unoccupied, the King is not in check, and the King does not cross over or end on a square in which it would be in check.
* Check – A state in which the King will be taken in the opponent’s next move. The side in check may not end their move with their King still in check.
* Mate – End of the game. The King is mated when check state cannot be removed via legal moves.
* Bishop – Each player starts the game with two Bishops. White having them start on squares c1 and f1, and black having them start on squares c8 and f8. The Bishop can only move diagonally and will always stay in squares of the same color it started the game in (a Bishop starting on a white square will never move into a black square).
* King – The King is the most important piece in the game, a mated King ends the game with the mated King’s side losing. Each player only has one King, white having it start on e1 and black having it start on e8. The King can only move one space horizontally, diagonally, or vertically. The King cannot move into a position that would cause it to be in check.
* Knight – Each player starts the game with two Knights. White having them start on squares b1 and g1, and black having them start on squares b8 and g8. The Knight moves by first moving 1 square horizontally or vertically, then 1 square diagonally in an outward direction. The Knight may also jump over pieces of either color to reach its destination.
* Pawn – The lowest value piece in a chess game that can move one square forward (or two on its first move) and can only capture another piece when moving one square diagonally. Each player starts with eight Pawns on the second rank of the board from each players perspective. If a Pawn reaches the opponent's end of the board it can be promoted to any other piece.
* Queen – Each player starts the game with only one Queen. White having it start on square d1 and black having it start on square d8. The Queen may move horizontally, vertically, and diagonally any number of squares but may not jump over pieces.
* Rook – Each player starts the game with two Rooks. White having them start on squares a1 and h1, and black having them start on squares a8 and h8. The Rook can only move horizontally or vertically across the chess board.
  + This portion will be updated as necessary as the document grows.

## 1.4 Supporting Materials

1. Bodlaender, Hans. “The Rules of Chess.” The Chess Variant Pages, [www.chessvariants.com/d.chess/chess.html](http://www.chessvariants.com/d.chess/chess.html)
2. “Chessboard.” Wikipedia, Wikimedia Foundation, 24 Jan. 2018, en.wikipedia.org/wiki/Chessboard.

This portion will be updated as necessary as the document grows.

## Document Overview

# Architecture

<The architecture provides the top level design view of a system and provides a basis for more detailed design work. This is the section where you should include your High-Level design Component Diagram.

**2.1 Class Diagram**

Figure 2.1 shows the class diagram for the chess software. The diagram is represented by the Unified Modeling Language.

* 1. **Templates**
     1. **Function Template**

Function details will be formatted in the following manner:

**Purpose:** Description how the function is used in the software.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:** Description of the procedure of the function.

**2.3 ChessBoardGame class**

This class is the main class in the program. It shall create objects of the Board class and Pieces interface to create the necessary components for the GUI and general movement of the game. Its methods shall initialize the game, track user choices on whether to continue gameplay, verify legal chess moves and verify whether a winner has been decided after each turn.

**2.3.1 Initialize**

**Purpose:** The purpose of this function is to initialize the game. Its methods will create the game board GUI.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.3.2 endGame**

**Purpose:** The purpose of this function is to create the framework that will end the game upon having a winner decided.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.3.3 giveUp**

**Purpose:** The purpose of this function is to provide the framework for the user to quit the game before a winner is decided.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.3.4 verifyWin**

**Purpose:** The purpose of this function is check if a winner has been decided after each player’s turn.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.3.5 verifyMove**

**Purpose:** The purpose of this function is to check whether a player’s chosen move is legal.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.4 Board Class**

This class provides the necessary structures to create a game board. It does so by creating a 32 x 32 array(or database) and calls for the creation of the 32 necessary game pieces.

**2.4.1 Initialize**

**Purpose:** The purpose of this function is to create the chess board and populated it with the 32 necessary pieces.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.5 Pieces Interface**

This interface provides the general framework to create 2 teams of the necessary chess pieces. It has a member called move that will allow each piece implementing the interface to describe their own movement laws.

**2.5.1 Move**

**Purpose:** The purpose of this function is to be a general function for movement, described by the individual pieces implementing this interface.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.6 Pawn Class**

This class provides the framework to give the pawn movement across the board.

**2.6.1 Move**

**Purpose:** The purpose of this function is to create the pawns ability to move generally.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.6.1 DoubleMove**

**Purpose:** The purpose of this function is to create the pawn’s ability to double move.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.7 Knight Class**

This class provides the framework to give the knight movement across the board.

**2.7.1 Move**

**Purpose:** The purpose of this function is to create the knight’s ability to move generally.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.8 Queen Class**

This class provides the framework to give the queen movement across the board.

**2.8.1 Move**

**Purpose:** The purpose of this function is to create the queen’s ability to move generally.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.9 King Class**

This class provides the framework to give the king movement across the board.

**2.9.1 Move**

**Purpose:** The purpose of this function is to create the king’s ability to move generally.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.9.2 Castle**

**Purpose:** The purpose of this function is to create the king’s special ability to castle.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.10 Rook Class**

This class provides the framework to give the rook movement across the board.

**2.10.1 Move**

**Purpose:** The purpose of this function is to create the rook’s ability to move generally.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

**2.11 Bishop Class**

This class provides the framework to give the bishop movement across the board.

**2.11.1 Move**

**Purpose:** The purpose of this function is to create the bishop’s ability to move generally.

**Inputs:** Description of the parameters of the function.

**Outputs:** Description of the return values of the function.

**Calls:** Functions called by this function.

**Called:** Functions this function calls.

**Procedure:**

# Overview

<This section provides a high level overview of the structural and functional decomposition of the system. Focus on how and why the system was decomposed in a particular way rather than on details of the particular components. Include information on the major responsibilities and roles that the system (or portions of it) must play.

# Component 1..n

<Describe an element (subsystem, component, etc...) from architecture in further detail. When appropriate, include information on how the element is further broken down and the interactions and relationships between these subcomponents.

# High-Level Design

<This section describes in further detail elements discussed in the Architecture. Normally this section would be split into separate documents for different areas of the design.

High-level designs are most effective if they attempt to model groups of system elements from a number of different views.

## View / Model Component 1..n

<Provide a description and diagrams of a system component or set of components that describes a clearly defined view or model of the entire system or a subset of the system.