Software Design Document

for

Chess Chaos

Version 1.0 approved

Prepared by Drew Grubb

Texas State University

Committed on March 13, 2018

Table of Contents

1. Introduction

1.1 Purpose

The software (Chess Chaos) described and modeled in this Software Design Description shall allow a user to play a game of chess. Chess Chaos will have a variety of modes and game types: Player vs. Player, Player vs. AI, AI vs. AI, Timed Mode, Casual Mode, Random Mode.

Users interacting with the software will be able to select game modes and preferences with relative ease in the integrated menu system and play chess without running into game breaking bugs.

1.2 System Overview

Chess Chaos will be programmed using the Java Development Kit (JDK) to run via the Java Virtual Machine (JVM). The structure of the system shall use a Game Loop, where all the components of the game are updated before rendering each component to the screen sequentially.

The Game Loop works using the following steps:

Update:

- Menu State:
  - Check Input on button locations
  - If input exists, perform button specific action
- Play State:
  - Check current game state for Check or Checkmate
  - If in Checkmate, end game.
  - If in Check, update possible moves.
  - Check input on board and menu button locations
  - If input exists on board, update or select pieces based on current game state.
  - If input exists on button locations, perform button specific actions.
  - If board updates, update possible moves of each piece on updated board
- Replay State:
  - Change each move in the Queue at a specific interval until the end is reached.

Render:

- Menu State:
  - Render background image
  - Render buttons by location

- Animate buttons if they are being hovered by the mouse

- Play State & Replay State:
  - Render background image
  - Render board spaces
  - Render pieces
  - Render buttons by location
  - Animate buttons if they are being hovered by the mouse (PlayState only)

1.3 Definitions, Acronyms and Abbreviations

Listed by relevance with a secondary listing alphabetically

Chess Terms and Pieces:

Bishop: A piece that can move diagonally across any number of squares until it is stopped by the edge of the board or the presence of another piece. Can capture the first enemy piece it runs into.

Capture: The act of removing an enemy piece from the board by occupying its space with your own Piece.

Castle: A special move where, if the King has not been moved and either of the corresponding team's Rooks have not been moved, the King may move two squares towards a Rook and the Rook may move to the other side of the King.

Check: To make a move that places your opponent's King in the Line of Sight of a piece, forcing them to move reactionarily.

Checkmate: To make a move that places the enemy King in Check where they cannot make a move to get themselves out of check, therefore ending the game in a victory.

En Passant: A special move where, if an enemy Pawn has just performed a double jump, a friendly pawn next to that pawn can perform a Capture in basic pawn style, sliding forward one space and horizontally one space.

King: A piece that can move one space in any direction or Castle if the state of the board allows it. The goal of the game is to force your opponent's King into Checkmate. Can capture an enemy piece provided doing so would not result in a Check.

Knight: A piece that makes its moves in an L shape (one space vertically & two spaces horizontally or visa versa). Can jump over pieces and directly capture pieces.

Line of Sight: If a move from a piece would knock out another piece, it is commonly said that the second piece is in the Line of Sight of the first piece.

Pawn: The lowest rank piece that can move forward two spaces on its first move and only one space every move after that. Capture is allowed only if the enemy piece is one space "in front of" and one space horizontally the pawn.

Promotion: The act of moving your pawn all the way to the opposing border throughout the course of the game, which allows you to switch out your pawn with a piece of any type, regardless of whether or not it is a duplicate.

Queen: The strongest piece in Chess, the Queen can move with the combined motion of the Rook and the Bishop.

Rook: A piece that can move any number of spaces horizontally or vertically until it is stopped by a border or piece. Can capture the first enemy it encounters in any direction.

Stalemate: An end game situation in which a player can make no legal moves but is not in check. Results in a draw.

Interface and Integration

AI: (Artificial Intelligence) A software developed replication of human intelligence, allowing the software to operate without direct, manual control.

API: (Application Programming Interface) An interface that provides development tools and communicates with software components with regards to building a specific program.

GUI: (Graphical User Interface) An interface that allows users to interact with software with textless, visual tools, such as windows, scroll bars, and buttons.

IDE: (Integrated Development Environment) A component of software development that allows programmers and users to analyze the software in question to looks for defects and attempt to debug the software.

Java: A programming language that relies heavily on object-oriented design and a general-purpose implementation of various software components.

JDK: (Java Development Kit) An implementation of the Java Runtime Environment specifically based on developer necessity.

JRE: (Java Runtime Environment) An application that allows users to develop and run coded software that relies on the Java programming language.

UML: (Unified Modeling Language) A language used to develop diagrams, serialize objects across multiple language databases, and visualize the design of software.

1.4 Supporting Materials

GitHub Repository for this Project
https://github.com/drewgrubb0/Chess_Chaos

Contains:
- Software Requirements Specification
- Software Design Document (This document)
- Source code and associated project files
- Test Plan
- Any licensing information (MIT License)

Personal References:

My LinkedIn (Hire me)
https://www.linkedin.com/in/drew-grubb/

1.5 Document Overview

This document discusses the architecture and design aspects of Chess Chaos.

2. Architecture

Chess Chaos uses a state design pattern, switching between a Menu State and a Play State based on the state of the system.

The Menu State is used to set the settings of the game and initialize it before switching to the Play State. The Menu State will have multiple buttons, and each will initiate a specific task.

The Play State is how the user will play the game of chess. It is responsible for updating the Game object which updates the board, timers, and pieces. The PlayState also updates any in game buttons.

The Replay State takes a previously saved game and replays it, allowing the user to study saved games.

For more information on the Game Loop, read System Overview 1.2.

2.1 Overview

Due to the large amount of entities that must be updated within the game at any given time (Up to 32 in Normal Chess), as well as pieces being dependent on one another, the Game Loop update method is the best choice for this design.

Pieces are updated at the beginning of the game, and at the end of each subsequent turn. This is to ensure that the moving of pieces does not overlap the possible moves of other pieces.

Pieces are rendered after all updating so that the visual display does not negatively affect the player's perception of the current board state (For example, if a player moves a piece the loop render is immediately updated instead of on the next frame).

2.2 Architecture Component 1: Game Loop

Main function initializes the game loop and keeps it constantly running.

Separates functional components into updating and rendering

Main also initializes the Display class and the States

For more information on the Game Loop, read System Overview 1.2.

2.3 Architecture Component 2: Menu State

There are several different menu sets used to set up the game.

The sets are as follows:

Initial Screen: [Play Chess | Play a Replay | Quickplay]

Board Screen: [Standard Chess | Random Chess | Etc]

Players Screen: [Player | Random AI | Hard AI | Etc]

Settings Screen: [Casual | Timed Mode]

2.4 Architecture Component 3: Play State & Replay State

These states will handle the interaction between the user and the board, as well as interactions between the user and the buttons.

There are specific button sets for each state:

Play State:

Clicking on a piece "selects" it.

Clicking on a tile moves a selected piece to that tile (if allowed), and switches the turn

Pause/Resume button

Reset button

Undo button

Quit to Menu

Replay State

Pause/Resume button

Change move delay

Undo/Redo move

Quit to Menu

2.5 Architecture Component 4: Pieces / Board

Each piece in Chess Chaos will be created with the common superclass Piece, which will consider each piece's unique move set and image file. Some pieces also have different move sets if they have not already moved, such as the Pawn or the King.

Moves are kept within a custom MoveSet.

Move contains an origin Position (int x, int y) and a destination position.

The Board class handles the movement of pieces within the 2D piece matrix and also keeps track of the state of the game.

3. High-Level Design

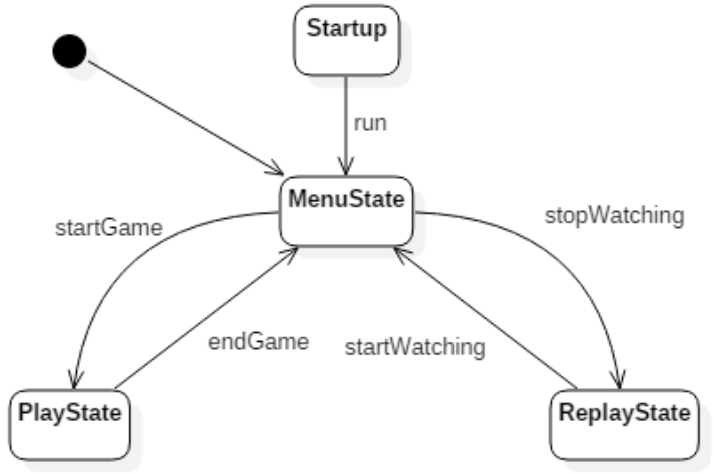
Class diagrams have had getters and setters omitted.

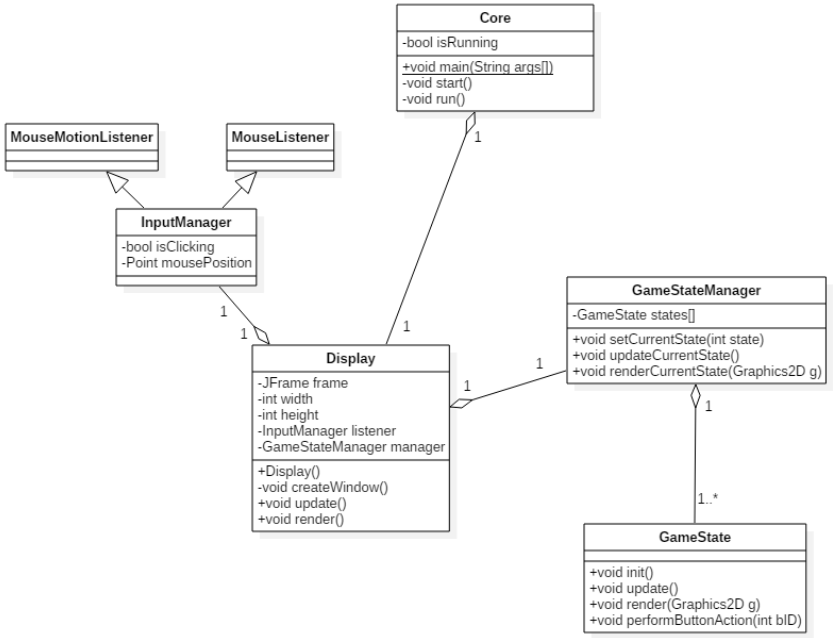
Built in Java class diagrams are empty

Class diagrams expanded upon in another diagram are empty

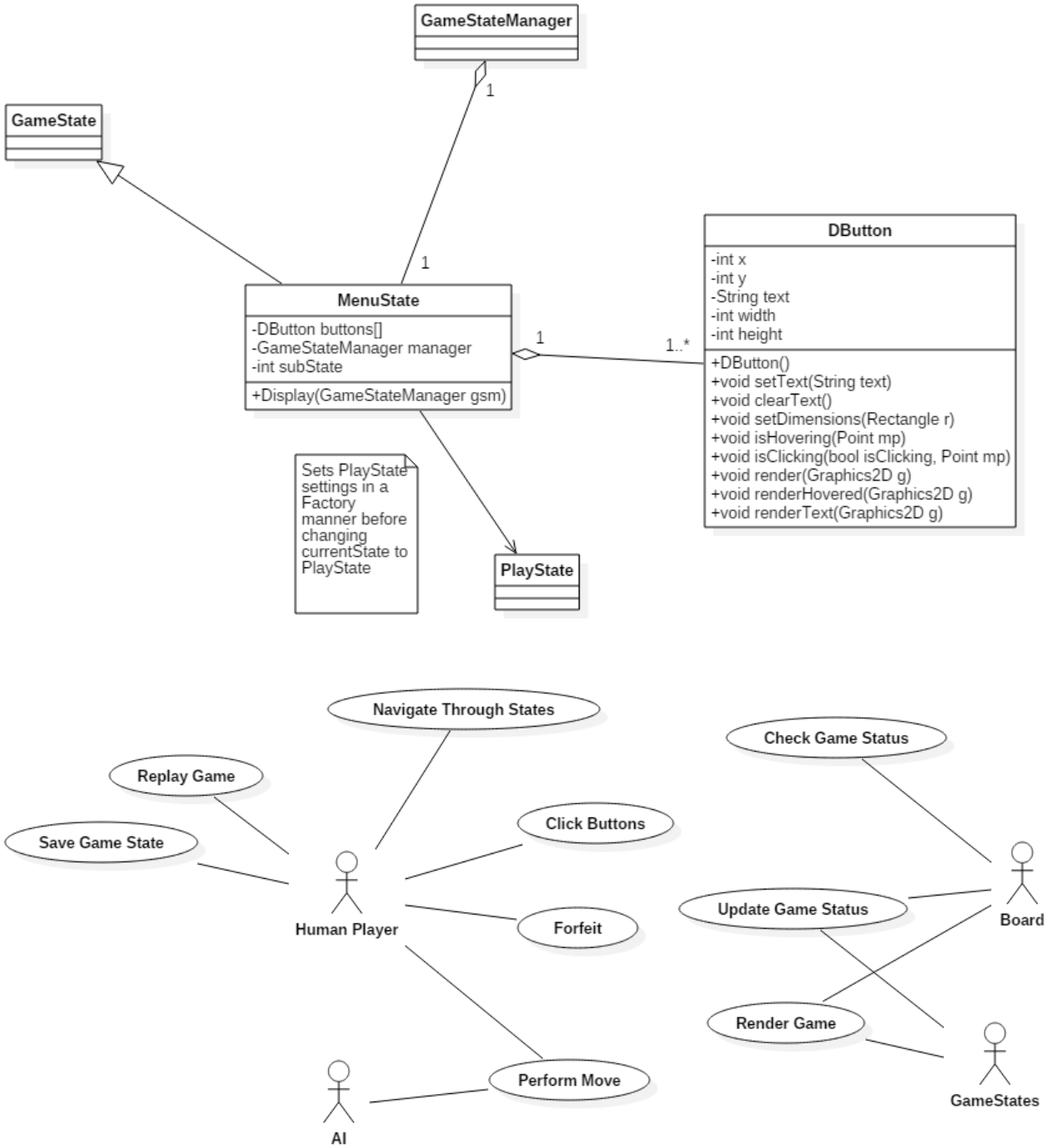
Final fields are omitted

3.1 Architecture Component 1: Game Loop Model



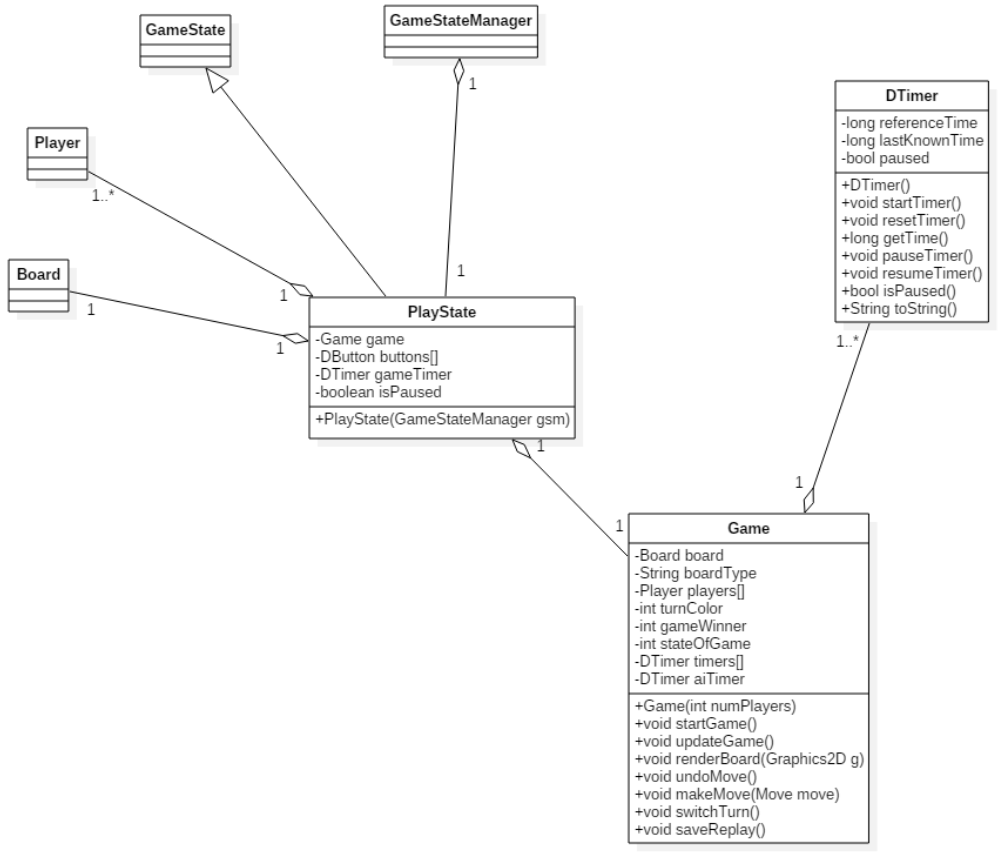


3.2 Architecture Component 2: Menu Model



Use case is general and implemented in this document as a concept only due to the nature of the standalone project having a single user.

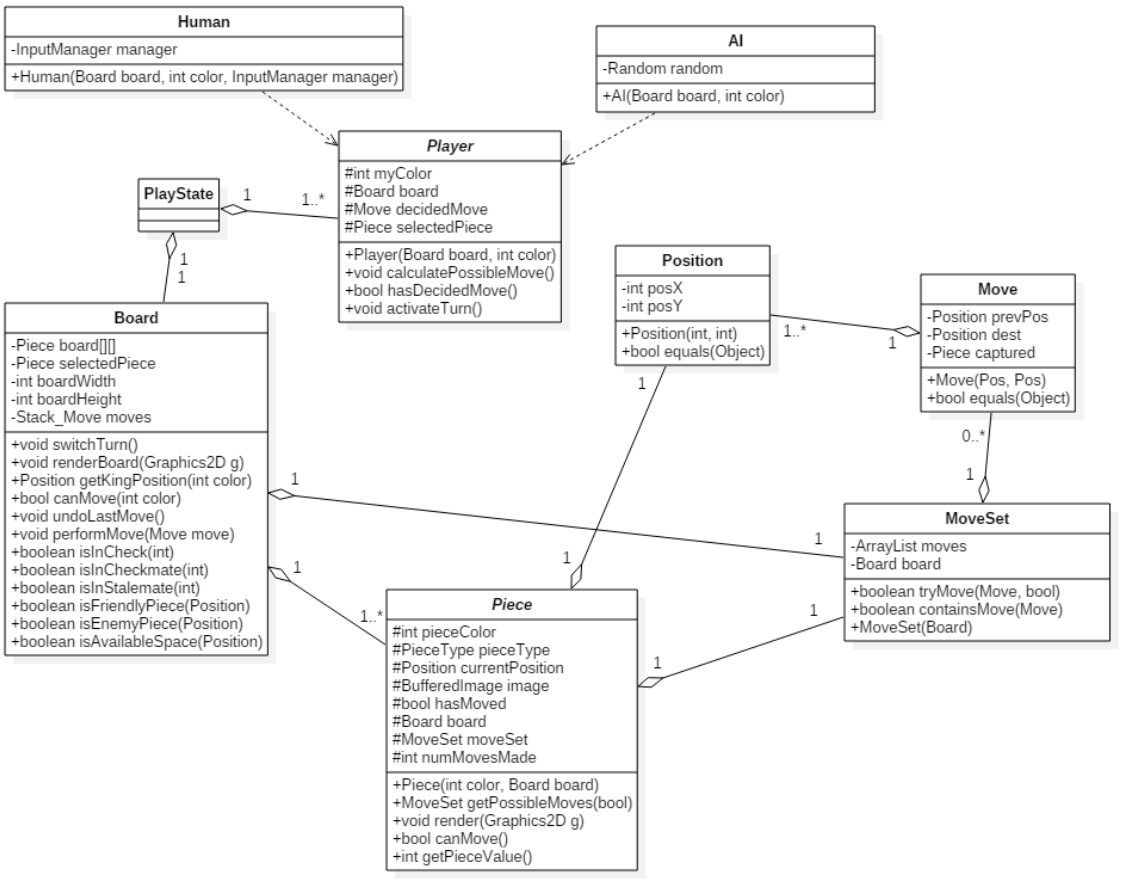
3.3 Architecture Component 3: Play State / Replay Model



PlayState and ReplayState will differ in the implementation of DButtons.

PlayState also has more fields, as ReplayState will not keep track of timers or specific players; only moves and the board.

3.4 Architecture Component 4: Piece / Board Model



Although Chess Chaos will not have network multiplayer, the Human class will be implemented with networking in mind, having an InputManager passed to it instead of using InputManager in a static context.