

CS 320 Course Project Final Report

for

Chess

Prepared by

Group Name: Chess

|  |  |  |
| --- | --- | --- |
| Steven Call | 11451145 | [steven.call@wsu.edu](mailto:steven.call@wsu.edu) |
| Daniel Maida | 11441203 | [daniel.maida@wsu.edu](mailto:daniel.maida@wsu.edu) |
| Nazar Stelmakh | 11441299 | [nazar.stelmakh@wsu.edu](mailto:nazar.stelmakh@wsu.edu) |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
|  |  |
| Date: | 9 December 2016 |
|  |  |
|  |  |
|  |  |

Contents

Contents ii

1 Introduction 1

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

2 Design 2

2.1 System Modeling 2

2.2 Interface Design 2

3 Implementation 3

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

4 Testing 4

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

5 Analysis 5

6 Conclusion 6

Appendix A - Group Log 7

# 

# Introduction

This Chess interface will be a self contained system that is being developed for a 300 level software engineering project. This system will be developed with the intent of allowing users to select from multiple game options, as well as select who their opponent will be. This system will have multiple modes of gameplay allowing different features to be enable or disabled based upon preferance. No outside devices will be needed to play or interact with this system, outside of the normall components of a computer.

## Project Overview

For this project we created a Chess implementation to simulate this classic board game. The goal of our Chess simulation was to provide a clean user inferface that engaged the players in an epic battle between man vs. machine.

The main features of our Chess simulations are the following. When first starting the simulation the player has the option of player vs. player or player vs. computer game modes. The user also has the option to cycle through two default themes that change the color of the chess tiles, and also changes the background image behind the chess board. During game play, the user has the option to undo the move until the first move of the game. The user also has the option to reset any time during game play.

## Definitions, Acronyms and Abbreviations

IEEE – Institute of Electrical and Electrons Engineers.

AI – Artificial Inteligence

JDK – Java Development Kit

Java – Programming language

## References and Acknowledgments

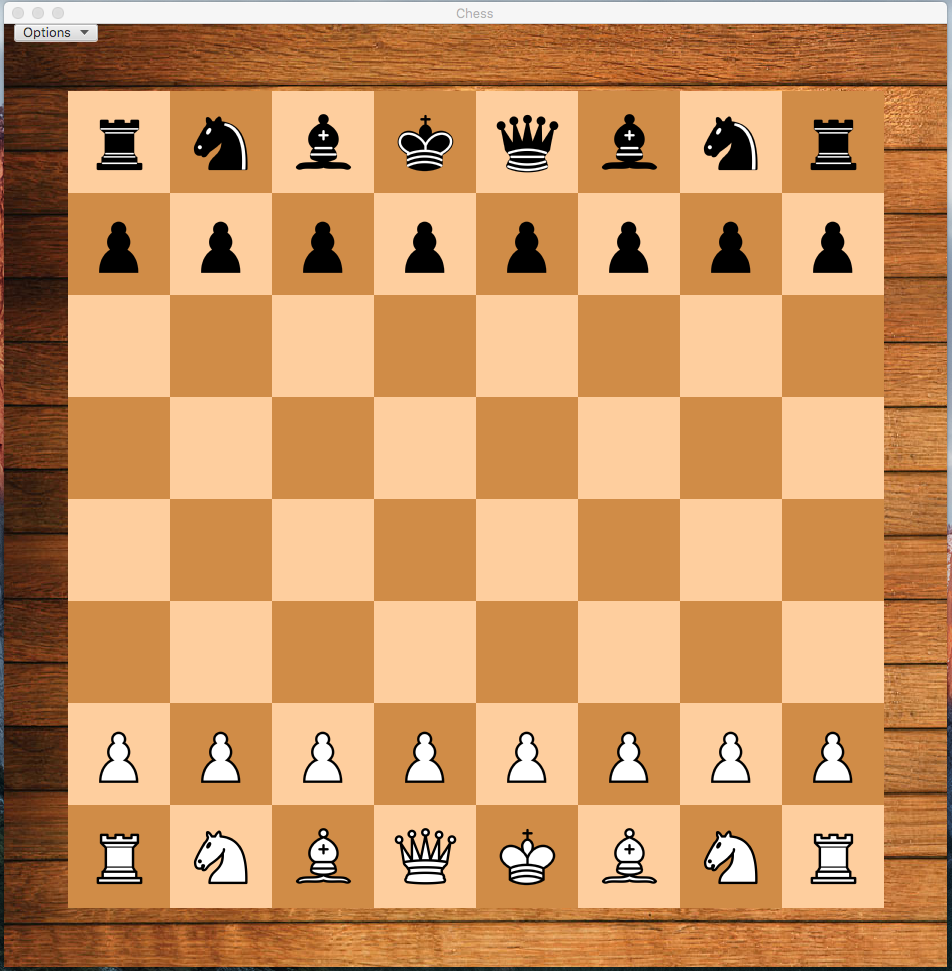
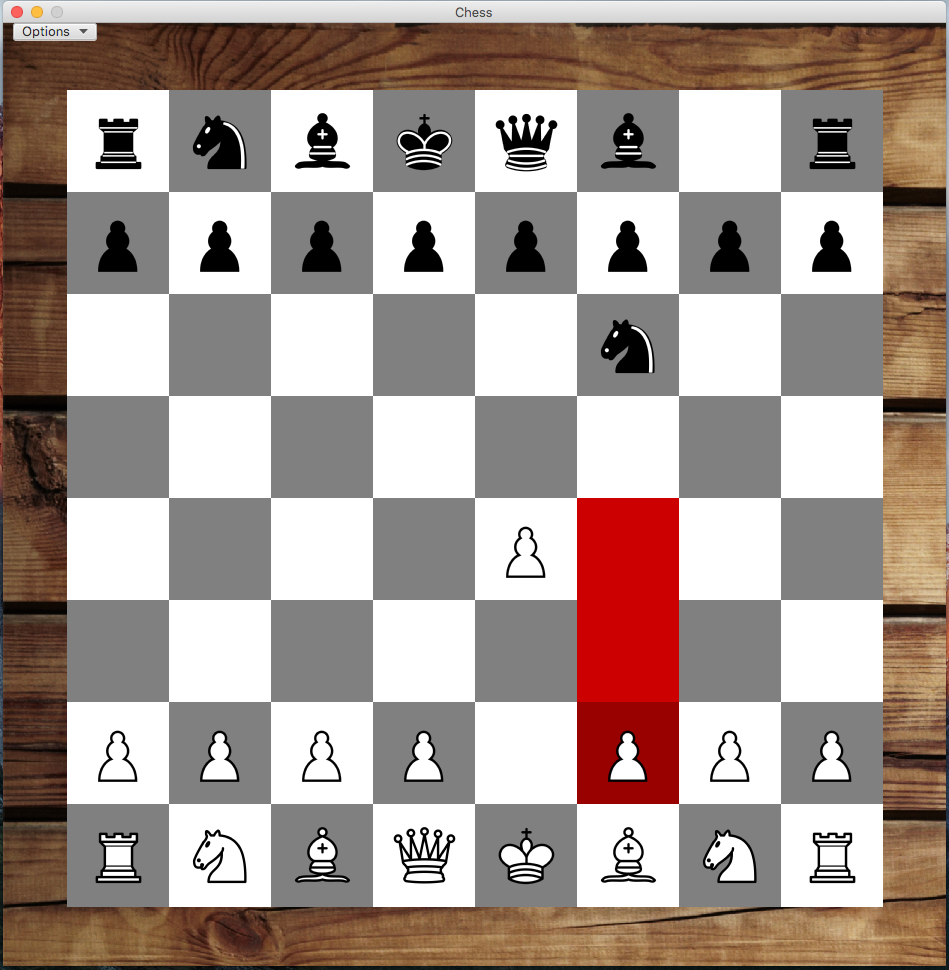
No outside works are referenced in this document.

# Design

## System Modeling

Our implementation strictly follows the design document (deliverable 2).

## users_b:nazar7700:Desktop:2.pngInterface Design



# Implementation

## Development Environment

The programing language that was used for this was Java. For the development enviroment we used Jetbrain’s IntelliJ IDE for the development of our software. We also used Gluan’s Scene Builder for creating the GUI. In additions, we used the application framework JavaFX for the creation of our application.

## Task Distribution

The main task for each group member was to implement one of the chess Piece classes. This task was equally distributed to Steven, Nazar and Daniel.

The second tasks that were assignmed to each group was the main graphical interface, the implementation of Chess rules, and finally the chess artificial intelligence. Daniel’s assigned task was the implementation of the GUI. Nazar’s assigned task was the implentation of chess arificial intelligence. Steven’s assigned task was to implement the rules of a Chess game.

## Challenges

One of the biggest changelles we encountered was the King piece class. It was unexpectedly more complicated than originally anticipated. Another challenged that we encountered was the chess artificial intelligence. It was extremely difficult to find to work on it, and there was not enough time allotted in order to completely finished a full implementation of an AI.

# Testing

## Testing Plan

**The Testing Process:** We will be creating a test checklist for our inspection group to follow while testing our program and variables. There will be two types of testing that will occur, static testing being done by the inspection group and Dynamic where test code is created to test specific situations among our code. The inspection group will be given the game and will run it trying to test the implementation and functionality of the game design as well as the options buttons within our program.

**Requirements traceability:** The requirements that our program is needing to meet will be based on the expectations of the game rules and standards in a common game of chess. We will have basic requirements to test such as realizing a winner, realizing when a player is in check, what moves can get them out of check, not playing pieces out of bounds (off the board), etc.

**Tested items:** The game will be tested by running the program and have the inspection group play the game. During the game, they will be trying to find exceptions and bugs that allow them to break the rules or where improper moves will break the game. Statically this will be the testing to ensure that our game design and functionality is working correctly.

**Testing schedule:** We will give the game to the inspection group and give them a few days to look over the code and mess with it to find the issues. The group will follow our checklist and keep track of issues they had so that we can record their testing results.

**Test recording procedures:** Our inspection group will be recording the results of their tests in a word document along, like that of a checklist. If issue are found the group will record the issue that they found as well as explain the scenario in which it happened and what they were trying to do while it happened. When we are running our coded test cases we will have the output of each test print whether it was successful and where the test failed in your code, this will be done using JUNIT testing.

**Hardware and software requirements:** Our program has been initialized and programmed on different computers such as mac OS X, and Windows 10. Our code is required to be accessible and deliverable to any interface or system and be functional. Our code is also required to be very user friendly to the point that the user does not need to know what is going on behind the scenes and can just use the program and not have issues with player restrictions. These will be tested within our inspection group for static testing and we will create test programs for Dynamic Testing.

**Constraints:** With this project the only constraint on who can test this program is that they have a basic understanding of the game of chess. Available moves are provided to the user but they will need to know how to win and some of the basic rules of the game to test.

## Tests for Functional Requirements

## Unit Testing:

All unit tests were written in Java’s JUnit testing framework.

test\_Initialization():

The purpose of this test is test that the Controller and the game board are properly initialized.

Testing initialization.....pass

Result;

test\_PieceInitialization()

The purpose of this test case is to test the initialization of every chess piece in the game board. And that every abstract piece is correctly placed on the game board.

Result: Testing Piece initialization.....pass

test\_Bishop()

The purpose of this test is to test the Bishop’s allowable moves. Does the Bishop piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing bishop allowable moves.....pass

test\_King()

The purpose of this test is to test the King’s allowable moves. Does the King piece correcly move within the game board, and is it correcly placed in the new game board position.

Result: Testing King allowable moves.....pass

test\_Knight()

The purpose of this test is to test the Knight’s allowable moves. Does the Knight piece correcly move within the game board, and is it correcly placed in the new game board position.

Result: Testing Knight allowable moves.....pass

test\_Pawn()

The purpose of this test is to test the Pawn’s allowable moves. Does the Bishop piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing Pawn allowable moves.....pass

test\_Queen()

The purpose of this test is to test the Queen’s allowable moves. Does the Queen piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing Queen allowable moves.....pass

test\_Rook()

The purpose of this test is to test the Rook’s allowable moves. Does the Rook piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing Rook allowable moves.....pass

## Interface:

test\_Interface()

The purpose of this test case is to test the initialization of the game board and the controller.

test\_Buttons()

The purpose of this test case is to initialize the controller and test the initialization of the buttons in the interface.

## Tests for Non-functional Requirements

REQ 1.0 - The A.I. engine should process a move in a reasonable amount of time:

We will try to advance the skills of our AI before the end of the term.

REQ 2.0 – Every move made by the user will be processed instantaneously:

With the current program, the user can make a move using the GUI and have it instantaneously processed and carried out. There is no lag time between the user’s interaction and the actual event happening on the board.

REQ 3.0 – The game should load in less than three seconds:

Our program currently loads and does not have a stall in the process while trying to open, there is no lag time which allows for the game to load in under 3 seconds. This accomplishes this requirement even though we have extensive code it still loads instantaneously.

REQ 4.0 – Changing themes will be instantaneous:

Our button that changes themes is under the options menu, when clicked this will change the theme of the board to the second option. This action happens instantaneously when the button is clicked, there is no lag time for this command.

## Hardware and Software Requirements

**Hardware and software requirements:** Our program has been initialized and programmed on different computers such as mac OS X, and Windows 10. Our code is required to be accessible and deliverable to any interface or system and be functional. Our code is also required to be very user friendly to the point that the user does not need to know what is going on behind the scenes and can just use the program and not have issues with player restrictions. These will be tested within our inspection group for static testing and we will create test programs for Dynamic Testing.

# Analysis

**Effort Analysis:**

Through out this project we have had many difficulties that we had to work through and overcome. While completing this project our group organized times to meet and push forward with a group effort on our program. Each member would be assigned different tasks to accomplish during the time period that we were working together.

Each individual put in roughly about 32-40 hours of work on our project to accomplish our end result. One of our largest struggles was finding hours that the group could meet and work on the project together inbetween work and school schedules.

The largest milestone that we came across that took the most effort was finishing the code for the King and the gameplay of being in check and possible moves. This was the most difficult because we had to use every other pieces move availability to check to see if the king could make a “safe” move to get out of check or prevent itself from going into check.

# Conclusion

In conclusion, while working through our various struggles with these project our group has furthered our knowledge in the area of group dynamics. There are definitely pro’s and con’s of working in a group on a large time consuming project, pro’s being more ideas and man power to progress the project while the con’s are scheduling conflicts.

We have better understood the design and development that is needed to target a specific customer based audience and meet the needs and requirments desired. When working through our project we could have made the game very easy for ourselves (the programers) to use and understand but we needed to make the gui and program easy for customers, with no understanding of the code behind the game, to use and understand. We were able to further our understanding of gui designs and what can be used as a very user friendly design. We have better understood and how to use object oriented design to make our program and project cleaner and more appealing to the general audience that will be based on our customers.

While going through and creating our program we started to see general patterns and processes that we could make reusable. For example when coding the specific pieces we realized that making the code more generalized we could use the code from one piece such as the bishop and implement it in the queen who also has the same move properties. By reusing this code we were able to expedite the process of finishing the code and move on to more important user interface issues.

Appendix A - Group Log

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Meeting/Collaborative  Work Time (hrs) | Individual Work Time (hrs) | Total Time (hrs) |
| Daniel Maida | 32 | 6 | 38 |
| Nazar Stelamkh | 32 | 6 | 38 |
| Steven Call | 32 | 6 | 38 |