# Software Requirements Specification

# for

# Chess Battle

Version 1.0 approved

Prepared by:

XYZ

September 2, 2016

Table of Contents

Table of Contents [ii](#__RefHeading___Toc441230970)

Revision History [ii](#__RefHeading___Toc441230971)

1. Introduction [1](#__RefHeading___Toc441230972)

1.1 Purpose [1](#__RefHeading___Toc441230973)

1.2 Document Conventions [2](#__RefHeading___Toc441230974)

1.3 Intended Audience and Reading Suggestions 2

1.4 Product Scope 2

1.5 References 3

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Functions [3](#__RefHeading___Toc441230980)

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies [5](#__RefHeading___Toc441230985)

3. External Interface Requirements 5

3.1 User Interfaces [5](#__RefHeading___Toc441230987)

3.2 Hardware Interfaces 6

3.3 Software Interfaces 6

3.4 Communications Interfaces 6

4. System Features [6](#__RefHeading___Toc441230991)

4.1 System Feature 1 7

4.2 Use case for chess battle (and so on) [8](#__RefHeading___Toc441230993)

5. Other Nonfunctional Requirements 9

5.1 Performance Requirements 9

5.2 Software Quality Attributes 9

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  | 0.0 |
|  |  |  |  |

# 1. Introduction

## Purpose

This document specifies all the requirements for the Chess game software. These requirements relate to the functionality, constraints, performance, attribute and the system interface. In this form are analyzed the basic characteristics and the features of the program and other necessary information on how those are implemented.

The Chess Battle is a desktop based application used to play chess game. First goal is to allow two users or players to play the game interactively and the second goal will be that the program should be working and allow the users to play the game.

## Document Conventions

**GUI** - Graphical User Interface

**SRS -** Software Requirement Specification

**Bishop -** One of two pieces of the same color that may be moved any number squares diagonally, as long as no other piece blocks its way. One piece always remains on White squares and the other always on Black.

**Castling** - To move the king two squares horizontally and bring the appropriate rook to the square the king has passed over.

**Check** - To make a move that puts the opponents King under direct attack.

**Checkmate** - A situation in which an opponent’s king is in check and it cannot avoid being captured. This then brings the game to a victorious result.

**Chess Board** - A board you need to play Chess. Have 64 black and white square.

**Chess** - A game played by 2 people on a chessboard with 16 pieces each.

**En Passant** - A method by which a pawn that is moved two squares can be captured by an opponent's pawn commanding the square that was passed

**King** - The main piece of the game, checkmating this piece is the object of the game. It can move 1 space in any direction.

**Knight** - This piece can move 1 space vertically and 2 spaces horizontally or 2 spaces vertically and 1 space horizontally. This piece looks like a horse. This piece can also jump over other pieces.

**Pawn** - One of eight men of one color and of the lowest value usually moved one square at a time vertically and capturing diagonally.

**Player or user** - A user or a player will be the person that is playing the chess game.

**Queen** - This piece can move in any number of spaces in any direction as long as no other piece is in its way.

**Rook** - One of two pieces of the same color that may be moved any number squares horizontally or vertically, as long as no other piece blocks its way.

**Stalemate** - A situation in which a player’s king is not in check, but that player can make no move. This then results is a stalemate, which is a draw.

## Intended Audience and Reading Suggestions

This Software Requirements document is intended for:

* Developers who can review project’s capabilities and more easily understand where their efforts should be targeted to improve or add more features to it (design and code the application it sets the guidelines for future development).
* Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.
* End users of this application who wish to read about what this project can do. The end user can also go through the user manual so that can get an idea how to play the game.

## Product Scope

Chess Battle is a cross-platform application that allows two players to engage in a real-time game of chess. The primary goal is to provide a chess game that is intuitive and entertaining for players of all skill levels. The game also provides some useful features, such as the ability to save a game and return to it later. In general, it focuses on providing a simple, streamlined playing experience.

## References

More about Chess Battle can be found:

[chessprogramming.wikispaces.com](https://chessprogramming.wikispaces.com/)

[chessprogramming.net](http://www.chessprogramming.net/)

In this website you can find out more about the project and discuss any questions in the forums. You can go back and look at chess programming , code and problems that have been solved. There you can also find information about the developers as well as the project’s main characteristics such as programming language and algorithms.

# Overall Description

## Product Perspective

The software provides a platform to the beginners as well as the pundits of chess to play and enjoy the game at one platform. There is no shortage of computer-based chess games. Chess Battle is not significantly different from these other games.

The game allows two players to play a game of chess. It provides a simple GUI. It also allows players to save and load games. Players may undo the most recent move as well. The program contains one main component which consists from the board of the game and the pawns which are used through the game.

## Product Functions

1. Allows two people to play a game of chess over a platform with GUI interface that makes easy for the user to interact with the game.

2. Allows a player to save their current game or load a previous one

3. Allows a player to undo their last move

4. The software keeps a record of the pieces each player killed.

5. Chess Battle also restricts the moves not allowed in the chess, for the beginners it helps to learn chess.

## User Classes and Characteristics

The user of Chess Battle needs only to be familiar with the basic actions and moves of mouse and keyboard. The user of Chess Battle need experience and be able to play chess at least a basic level. Although the game uses simple text symbols anyone that knows the rules of the game and can understand the symbols on the pawns can use it without any problem. Furthermore, user needs to be very familiar and the comprehended chess rules.

## Operating Environment

This program will operate in the following operating environment for the client and the server GUI:

• Apple Mac OS X (Universal)

• Linux/Unix (Source code)

• Windows (95/98/2000/XP/Vista/7)

## Design and Implementation Constraints

This program is created using C++ programming language and uses the GUI header files for the User interface. So a minimum PC having at least 64mb of RAM and CPU over 400MHz is required to run the program with good speed. Also the program uses at least 15 megabytes of hard disk space to store the program libraries. An installer is used to unpack them all in an install wizard. For language support only from the basic English language pack that can be enabled within the program.

## User Documentation

Here are the links and there is also a user manual that can help the user to play the chess game on Chess Battle:

1. There is a user manual that guides the beginner to play the game of chess and makes familiar with the rules of chess.
2. Online Documentation :

* www-math.bgsu.edu/~zirbel/**chess**/Basic**ChessRules**.**pdf**
* https://www.fide.com/FIDE/handbook/LawsOf**Chess**.**pdf**

## Assumptions and Dependencies

Chess Battle is not platform dependence and can be installed in any operating system capable to run C++ environment.

**Rank:**

Before moving and capturing pieces, it is important to understand the pawns rank.

The pieces are ranked with each piece able to capture an equal or weaker piece. The rank

form the strongest to the weakest is:

General > Advisor > Elephant > Chariot > Horse > Soldier

The Soldier and the Cannon has some special abilities:

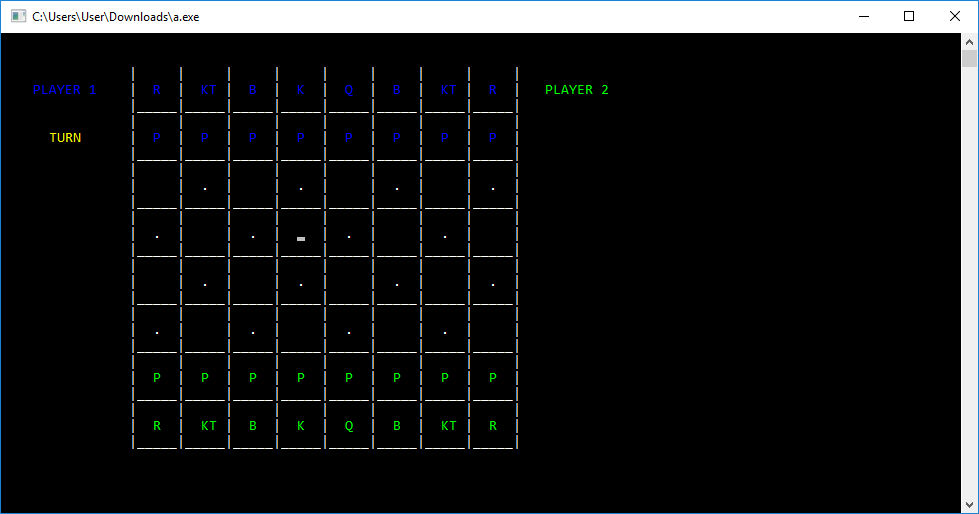
The soldier can capture the General, but the General cannot capture Soldiers. This is significant, since are five Soldiers and only one General. The General can capture any piece but a pawn, and pawns can only capture the king and other pawns. This makes the General more vulnerable than the Advisor (the next lower pieces).

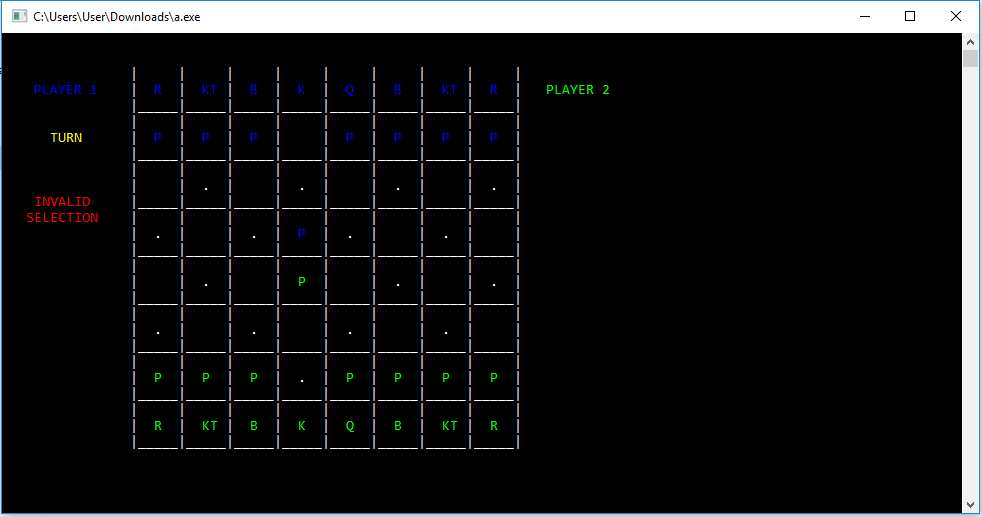
Cannons are the other exception to the rank rule: they can capture any piece including the General , and can be taken by any piece except the Soldier.

# External Interface Requirements

## User Interfaces

Chess Battle provides a simple GUI that the user operates with a mouse. The client is stand-alone. It only requires the C++ environment and, of course two players to play the game on software. Game data will be stored on the system.





## Hardware Interfaces

Chess Battle will run on any computer with the following:

• The C++ Runtime Environment

• A keyboard and mouse

## Software Interfaces

**Client**: The Chess Battle software has an interface that translates the user’s actions with the mouse into data the server will be able to understand and relay to the other user’s computer, in order to actually perform the move on both computers.

## Communications Interfaces

To play Chess Battle there is no requirement of any type of communication. The software is an desktop based application that is independent of the network and protocols.

# System Features

Generally there are no safety or performance requirements for the system. Because of its low system requirements it can be used almost at any computer if it uses the required Software. And because it is open source any user can see its code and realize that there is nothing dangerous for his operating system.

## System Feature 1

4.1.1 Description and Priority

**Moving:**

Any piece can move up, down, left or right if there is no other piece in that place and as long as it stays in the board.

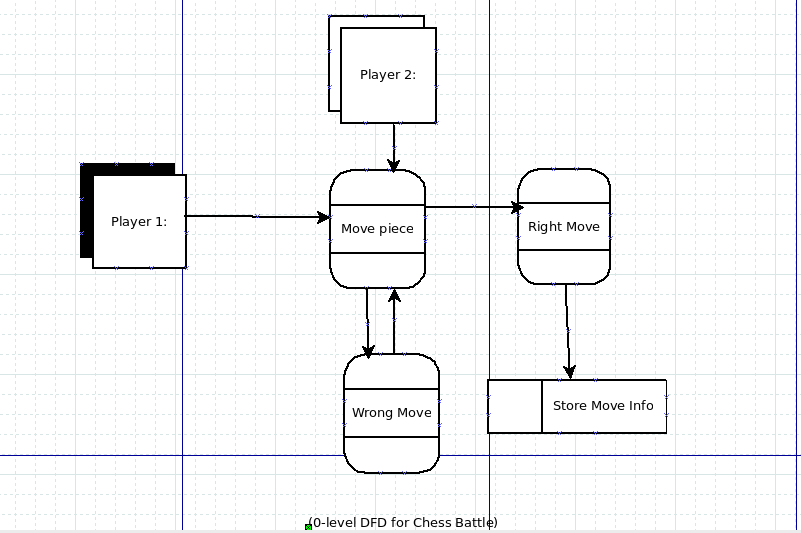
**Capturing, Attacking:**

With the exception of the cannon, a piece captures by moning onto a piece it can capture. The captured piece is then removed from the board and replaced by the capturing piece. You may only capture pour opponent’s face-up pieces. The cannon Although it moves normally (orthogonally one space at a time), it captures by jumping over a single piece (of either color, or even face down) and landing on its victim. The jumped piece may be surrounded on either or both sides by empty squares. This is the only way canons can capture.

**Game Over:**

The game is over when a player cannot make a legal move (cannot move, capture or turn up a piece). In that instance. The other player wins. A player may also choose to resign, or both players may mutually agree to draw.

**4.1.2 Simple DFD for chess Battle**



**4.2 Use case for chess Battle**

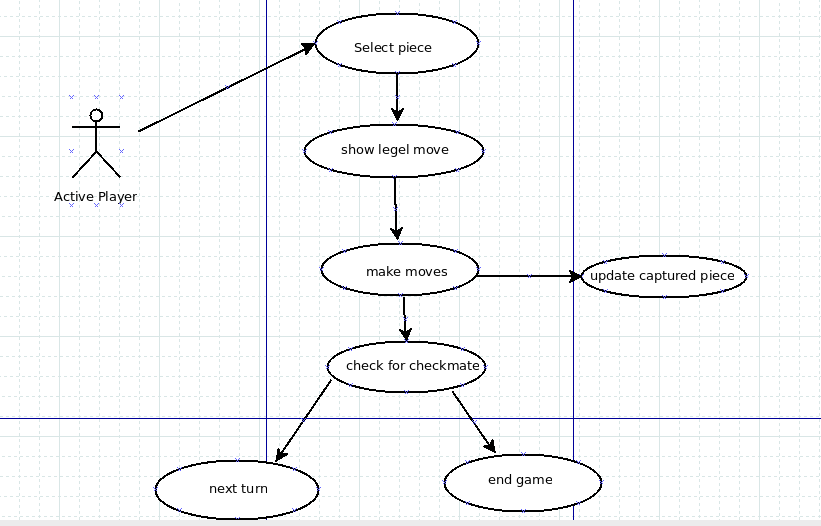
**Move**

Precondition: During the Play Game state

**Main Flow:**

The active player chose a piece to select it. The game displays the positions it can move to. The player selects the new destination by chosing. The piece is moved there if it is a valid move. Their opponent becomes the active player.

**Alternate Flow:**

The active player may decide to select a different piece by chosing one of their own. If there are no valid moves and the active player is not in check the game ends as a stalemate. If there are no valid moves and the active player is in check the inactive player wins.

**Redo**

Precondition: During play game state, and an undo hasn't been requested this turn.

Main Flow: The inactive player may request to undo their prior move (i). The active player is asked to Accept

(ii). If accepted the last move is reversed and the inactive player becomes the active player(iii).

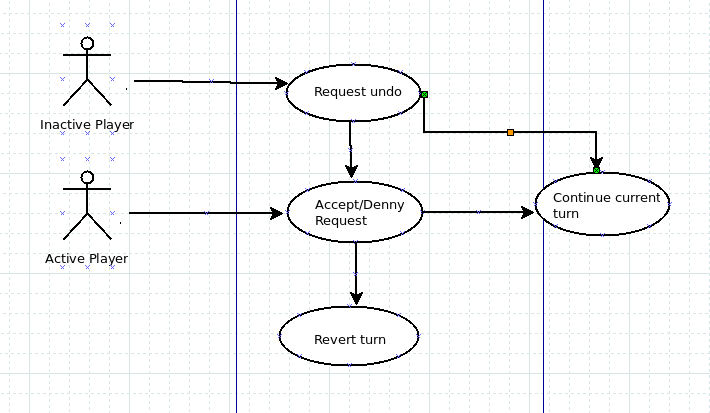
**Sub Flows:**

I. The inactive player requests a redo. II. The active player accepts.

III. The prior turn is reverted and the inactive player becomes the active player.

Alternative Flows:

The active player does not accept the undo, play is resumed as usual.



# Other Nonfunctional Requirements

## Performance Requirements

Generally there are no safety or performance requirements for the system. Because of its low system requirements it can be used almost at any computer if it uses the required Software. And because it is open source any user can see its code and realize that there is nothing dangerous for his operating system.

## Software Quality Attributes

1. Basic Chess Battle is a small and light project that after it is installed it can be easily used from any user.

2. Basic Chess Battle is a project that once it is uninstalled from a computer, leaves no trace. So there is no way any data can be found later.

3. Basic Blind Chess is developed under the C++ programming and can be shared easily.