e-Chess

Submitted by: Taimoor Abdullah Khan

Supervised by: Dr. Muddassar Azam Sindhu

Department of Computer Science

Quaid-i-Azam University

Islamabad

Session (2013-2017)

Acknowledgement

First of all I would like to express my gratitude to Allah Almighty, whose blessings I have seen during my time at Quaid-e-Azam University in the form of achievements, success and learning. I thank my parents for being concerned about my education all my life and for supporting it.

This project is the last step of my BS program. It has been full four years. I have come to meet good friends and have had the opportunity to learn from great teachers. Among the teachers, I would like to specially thank Dr. Mudassar Azam Sindhu for his supervision over this project and teaching us our subject in a manner that made complex topics easy to understand, Dr. Ghazanfar Farooq for his persistence in teaching his subjects in detail and with a steady pace so that no one in class gets left behind, he also took good care of us, specially gave me opportunity to work with senior students which honed my skills further and a special thanks to Dr. Rabbi Ayaz for letting me sit in his ETSD class, I am able to do so much more that I credit to that class which I would not have been able to do otherwise. Lastly all the teachers who have taught us in our four years Dr. Afzal Bhatti, Dr. Fayyaz Azam, Madam Tehsin Zahra, Dr. Farid Khan, Dr. Shamoona F. Qazi, Sir Abdul Sattar, Sir Akbar Saeed, Dr. Jaffri Hussain, Dr. Inayatullah, Dr. Mubashir Mushtaq, Madam Urooj, Madam Sana Jaffry, Dr. Zahid Rasheed, Madam Tasmia, Dr. Khalid Saleem, Madam Memoona Afsheen, Dr. Umer, Dr. Shoaib Karim, Sir S. M. Naqi.

I am also greatly thankful to friends Imran, Waqas and Rashid for keeping me company and sane during this time. I would also like to thank all my class fellows specially Abdul Rehman, Adeel Javed and Mahnoor Hamid for their efforts as friends in our learning and education as well as to make me a better human being.

Thank you!

Taimoor Abdullah Khan

Abstract

Being a video game lover, I wished to have a final year project that would let me hone my skills in game development and allow me learn and implement AI. In today's time Unity3d is popular for game development. So I opted to learn it and use it for my project. The challenge was that unity3d uses 3d objects in a 3d space and you cannot control them from inside some code like a main class, rather you have to attach scripts or code to these 3d objects and control their behavior by using them. The real complexity of this project is programming the rules of the game. A single unit can move in different directions according to the type of unit, its movement can be blocked by other units in its path, or if there is a check on a king then all these cannot move unless they move to remove the check. I had to write code once to make an instance of this work, then upon the next rule, all of that would have to be modified or rewritten to incorporate the next rule. Similar is the AI, it requires a separate class of a virtual board which is separate than the one shown to the player. Instances of this board is created multiple times when calculates its move and finally the AI plays a valid move. All in all, it was a wonderful practice to be able to implement such a project by myself alone.

Contents

List of Tables

List of Figures

Chapter 1

Software Project Management Plan

1.1 Introduction SPMP

This document specifies the project management plan for the e-Chess game software. It relates to the process of Software development cycle and deliverables such as Software Requirement Specification, Software Design Description and Software Test Documentation.

1.2 Project Overview

Chess is a two-player strategy board game played on a chessboard, a checkered gameboard with 64 squares arranged in an eight-by-eight grid. Chess is played by millions of people worldwide. This project plan is for making the game software e-Chess, a computer version of the real Chess game.

1.3 Project Deliverables

Project deliverables are:

- Software Project Management Plan (SPMP)
- Software requirement Specifications (SRS)
- Software Design Description (SDD)
- Software Test Documentation (STD).

1.4 Project Organization

1.4.1 Software Process Model

Waterfall process model is used for e-Chess. As there is no risk and no change in requirements, a straight forward model is within understanding to be good for use.

1.4.2 Roles and Responsibilities

The roles of requirement gathering, analysis, design, implementation, testing are all to be done by Taimoor Abdullah Khan.

1.4.3 Tools and Techniques

Following are the tools used for this project.

Table 1.1 – Tools and Techniques

Tools and techniques	
MS Word	Used for documentation.
StarUML	Used for making Use case diagram, System Sequence diagrams, Domain Class diagrams.
Unity 3D 5	Game development IDE which focuses primarily on 3D. It is free to use.
Project Libre	Used to make time table for project plan distributing tasks over the working months.
Alpha-Beta Pruning	Algorithm for AI that will be used for the computer player of the game.

1.5 Project Management Plan

1.5.1 Analysis

Description

The chess game is analyzed, basic actions are viewed and how the game works and from that use cases are derived and functional and non-functional requirements are extracted.

Deliverables and Milestones

SPMP

SRS

Resources Needed

A computer, internet connection, MS Word for documentation.

Dependencies and Constraints

Listed in the Figure 1.1

1.5.2 Design

Description

After the analysis is done, the design part is done, from the use cases, use case diagram is made, then Domain Model, System Sequence Diagram, Class Diagram.

Deliverables and Milestones

SDD

STD

Resources Needed

A computer, internet connection, MS Word for documentation, StarUML for Class diagram, System Sequence Diagram, Use case Diagram.

Dependencies and Constraints

Listed in the Figure 1.1

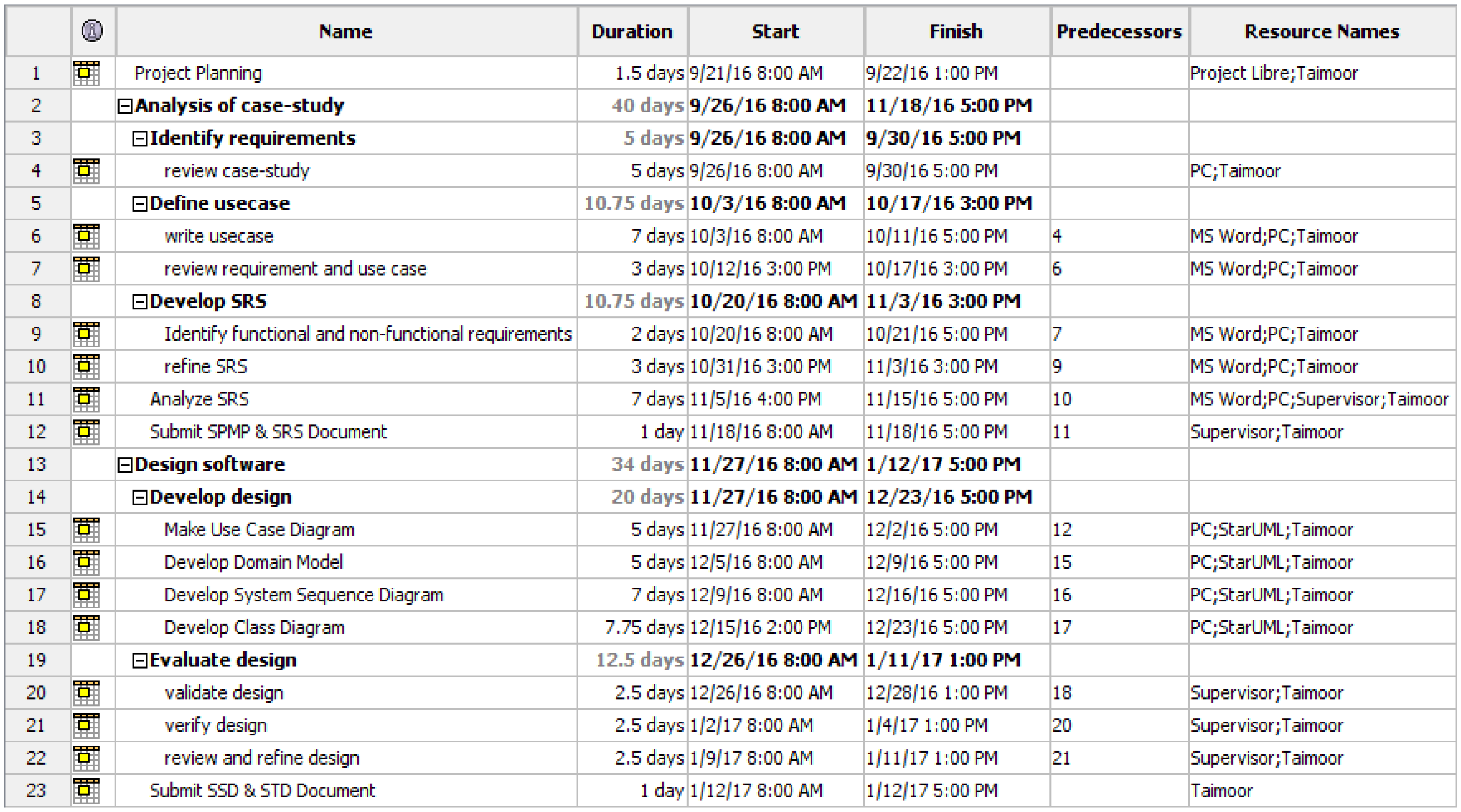
1.5.3 Assignments

All tasks are assigned to Taimoor Abdullah Khan while Supervisor will advise him where needed.

1.5.4 Timetable

Figure 1.1 shows the timetable.

Figure 1.1 Plan Time Table



	ⓘ	Name	Duration	Start	Finish	Predecessors	Resource Names
1	▦	Project Planning	1.5 days	9/21/16 8:00 AM	9/22/16 1:00 PM		Project Libre;Taimoor
2		⊟**Analysis of case-study**	40 days	**9/26/16 8:00 AM**	**11/18/16 5:00 PM**		
3		⊟**Identify requirements**	5 days	**9/26/16 8:00 AM**	**9/30/16 5:00 PM**		
4	▦	review case-study	5 days	9/26/16 8:00 AM	9/30/16 5:00 PM		PC;Taimoor
5		⊟**Define usecase**	10.75 days	**10/3/16 8:00 AM**	**10/17/16 3:00 PM**		
6	▦	write usecase	7 days	10/3/16 8:00 AM	10/11/16 5:00 PM	4	MS Word;PC;Taimoor
7	▦	review requirement and use case	3 days	10/12/16 3:00 PM	10/17/16 3:00 PM	6	MS Word;PC;Taimoor
8		⊟**Develop SRS**	10.75 days	**10/20/16 8:00 AM**	**11/3/16 3:00 PM**		
9	▦	Identify functional and non-functional requirements	2 days	10/20/16 8:00 AM	10/21/16 5:00 PM	7	MS Word;PC;Taimoor
10	▦	refine SRS	3 days	10/31/16 3:00 PM	11/3/16 3:00 PM	9	MS Word;PC;Taimoor
11	▦	Analyze SRS	7 days	11/5/16 4:00 PM	11/15/16 5:00 PM	10	MS Word;PC;Supervisor;Taimoor
12	▦	Submit SPMP & SRS Document	1 day	11/18/16 8:00 AM	11/18/16 5:00 PM	11	Supervisor;Taimoor
13		⊟**Design software**	34 days	**11/27/16 8:00 AM**	**1/12/17 5:00 PM**		
14		⊟**Develop design**	20 days	**11/27/16 8:00 AM**	**12/23/16 5:00 PM**		
15	▦	Make Use Case Diagram	5 days	11/27/16 8:00 AM	12/2/16 5:00 PM	12	PC;StarUML;Taimoor
16	▦	Develop Domain Model	5 days	12/5/16 8:00 AM	12/9/16 5:00 PM	15	PC;StarUML;Taimoor
17	▦	Develop System Sequence Diagram	7 days	12/9/16 8:00 AM	12/16/16 5:00 PM	16	PC;StarUML;Taimoor
18	▦	Develop Class Diagram	7.75 days	12/15/16 2:00 PM	12/23/16 5:00 PM	17	PC;StarUML;Taimoor
19		⊟**Evaluate design**	12.5 days	**12/26/16 8:00 AM**	**1/11/17 1:00 PM**		
20	▦	validate design	2.5 days	12/26/16 8:00 AM	12/28/16 1:00 PM	18	Supervisor;Taimoor
21	▦	verify design	2.5 days	1/2/17 8:00 AM	1/4/17 1:00 PM	20	Supervisor;Taimoor
22	▦	review and refine design	2.5 days	1/9/17 8:00 AM	1/11/17 1:00 PM	21	Supervisor;Taimoor
23	▦	Submit SSD & STD Document	1 day	1/12/17 8:00 AM	1/12/17 5:00 PM		Taimoor

Chapter 2

Software Requirements Specification

2.1 Introduction SRS

2.1.1 Purpose

This document specifies the requirement for e-Chess game software. These requirements relate to functionality, constraints, performance, attributes and system interface.

2.1.2 Scope

This document describes the software requirements of the e-Chess game software. This document will be used by the supervisor, developer and Examiners.

2.2 Definitions

2.2.1 Chess

A game played by two people on a Chess Board with 16 pieces each player with the goal to Checkmate the other player's King

2.2.2 Chess Board

It is a square board. The game of Chess is played upon it. It is divided into 8x8 grid or 64 tiles with alternating black and white colors.

2.2.3 Rank

These are the rows of the chess board tiles. These are 1, 2, 3, 4, 5, 6, 7 and 8 with 1 starting from White's side of the board. These are used to denoted the location of a piece.

2.2.4 File

These are the columns of the chess board tiles. These are namely a, b, c, d, e, f, g, h. These are used to denoted the location of a piece.

2.2.5 Player

The one of two players who participate in the game. One player commands white colored units while the other commands black colored units.

2.2.6 Checkmate

A situation in the game, where the king is in Check and cannot avoid being captured. This results in victory for the player who Checkmates the opponent's King.

2.2.7 Check

A situation where the King is under line of attack. In such a situation the only allowable moves for the Checked side are which saves the King from being Checked.

2.2.8 Stalemate

A situation where the player's King is not in check but the player cannot make any moves. This means that the game is a draw.

2.2.9 Unit

Refers to any type of chess game unit that the player uses in the game to play.

2.2.10 King

The most important unit in the game. The goal of the game is to Checkmate the king to achieve victory. This unit is not powerful at all, so all the other units protect it. It can move one step in any direction.

2.2.11 Queen

A unit that is able to move horizontally, vertically or diagonally by any number of steps as long as its path is not blocked.

2.2.12 Bishop

One of the two units of the same color that is able to move diagonally by any number of steps as long as its path is not blocked.

2.2.13 Knight

One of the two units of the same color that is able to move in a sequence of 2 steps vertically and 1 step horizontally or 1 step vertically and 2 steps horizontally. It can jump over other units

2.2.14 Rook

One of the two units of the same color that is able to move vertically or horizontally by any number of steps as long as its path is not blocked.

2.2.15 Pawn

One of the eight units of the same color that is able to move only one step but with exception of two steps from its starting location. It only moves in the direction of enemy's starting side of the board. It cannot move if there is a unit blocking its path. It can only capture an enemy unit one step diagonally front-left or one step diagonally front-right of it.

2.2.16 Castling

A move involving the King and Rook. The king moves two steps towards the rook and one of the rook moves to the spot that the king crossed. But provided that the king and the said rook haven't been moved in the game session before and there are no other units blocking their path and destination.

2.2.17 En Passant

A move by which one pawn that moved two steps from its starting position, can be captured by the other player's pawn at the location where the pawn passed.

2.2.18 Product Overview

Chess is a two-player strategy board game played on a chessboard, a checkered gameboard with 64 squares arranged in an eight-by-eight grid. Chess is played by millions of people worldwide, both amateurs and professionals. [**1**]

Chess is believed to have originated in India, sometime before the 7th century, being derived from the Indian game of Chaturanga [**2**]. Chaturanga is also the likely ancestor of the Eastern strategy games Xiangqi [**3**], Janggi [**4**] and Shogi [**5**]. The pieces took on their current powers in Spain in the late 15th century; the rules were finally standardized in the 19th century. [**1**]

Until recently, chess was a recognized sport of the International Olympic Committee; some national sporting bodies such as the Spanish Consejo Superior de Deportes also recognize chess as a sport. Chess was included in the 2006 and 2010 Asian Games. [**1**]

The rest of this document describes the system requirement for the e-Chess game software.

2.3 Specific Requirements

2.3.1 External Interface Requirements

This game Software is an application for Windows.

2.3.2 User Interfaces

It includes an interface similar to a chessboard, the user can tap or click (depending on the platform) on a unit to highlight and then tap or click on a tile to move that unit there. The game menus will be simple with not much functionality other than starting the game or resetting the game. e-Chess requires memory and data storage as it needs to be installed as an application to be used. It is not portable and needs to be installed first.

2.3.3 Hardware Interface

The game is for Windows, the hardware must meet the criteria of having 100 MB memory space, 100 MB storage space, mouse and keyboard for Windows.

2.3.4 Software Interface

Software Interface for player AI functionality that uses on Alpha-Beta Pruning. Each movement and type of unit capture can be assigned a rank, e.g. movement as 1, pawn as 2, king as 5, bishop, rook and knight as 3 and queen as 4. For one team these values are positive and for the other team these are negative. With these ranked, the algorithm can then iterate the possibilities and prune them to see the most beneficial move it can play.

2.3.5 Communications Protocols

No communication protocols are required because the game software is standalone.

2.3.6 Software Product Features

Here is the list of the major functions the game must perform or must let the player/user perform.

- Start the game
- Select a unit
- Move unit according to their allowed movement
- Capture enemy units
- Do check on enemy king
- Do checkmate on enemy king
- Detect Stalemate and result in a draw
- Detect other draw condition and result in a draw
- Record move timers
- Record number of moves total in game
- Make one player victorious when he does a checkmate

2.4 Use Cases

2.4.1 Start Game

Table 2.1 – UC1 Start Game

UC-1 name: Start Game	
Primary actor	User.
Stakeholder & Interests	User will be able to start the chess game.
Pre-condition	1. Game software should be running.
Post-condition	Game has started successfully and white Player can play the first move.
Main Success Scenario	1. User selects the Start Game button. 2. System loads Game. 3. Game starts.
Alternate Flows	1a. User presses Start Game, system crash. 2a. System cannot load Game into memory.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.4.2 Move Unit

Table 2.2 – UC2 Move Unit

UC-2 name: <u>Move Unit</u>	
Primary actor	User and Computer Player.
Stakeholder & Interests	User or Computer will be able to move a chess unit to a valid specified location.
Pre-condition	1. Game match should be started.
Post-condition	Unit moves to the location.
Main Success Scenario	1. User or Computer selects one of his Unit. 2. User or Computer selects the location. 3. Game checks validity. 4. Unit moves to the location.
Alternate Flows	1a. User selects opponents unit, nothing happens. 1b. User doesn’t select any unit, nothing happens. 3a. Location isn’t valid, nothing happens.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.4.3 Capture Unit

Table 2.3 – UC3 Capture Unit

UC-3 name: <u>Capture Unit</u>	
Primary actor	User and Computer Player.
Stakeholder & Interests	User or Computer will be able to capture an enemy unit with one of his own.
Pre-condition	1. Game match should be started.
Post-condition	Unit captures the enemy unit.
Main Success Scenario	1. User or Computer selects one of his Unit. 2. User or Computer selects enemy unit. 3. Game checks if capture is possible. 4. Enemy unit is captured and User's unit moves to enemy unit location.
Alternate Flows	1a. User selects opponents unit, nothing happens. 1b. User doesn't select any unit, nothing happens. 2a. User selects his own unit, redo from first step again. 3a. Capture isn't possible, nothing happens.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.4.4 Surrender

Table 2.4 – UC4 Surrender

UC-4 name: <u>Surrender</u>	
Primary actor	User.
Stakeholder & Interests	User will be able to Surrender his king, therefore forfeiting the game match.
Pre-condition	1. Game match should be started.
Post-condition	Player surrenders successfully.
Main Success Scenario	1. User clicks the surrender button. 2. Surrender is successful.
Alternate Flows	1a. User doesn't click the surrender button, nothing happens.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.4.5 Checkmate

Table 2.5 – UC5 Checkmate

UC-5 name: <u>Checkmate</u>	
Primary actor	User and Computer Player.
Stakeholder & Interests	User or Computer will be able checkmate the enemy King.
Pre-condition	1. Game match should be started.
Post-condition	Enemy King is checkmated.
Main Success Scenario	1. User or Computer selects the Unit. 2. User or Computer selects the location or enemy unit. 3. Game checks validity of move. 4. Unit moves to the location or captures enemy at that location. 5. Game checks if enemy King is under check and cannot escape or can be save by another unit. 6. Checkmate is successful.
Alternate Flows	1a. User selects opponents unit, nothing happens. 1b. User doesn't select any unit. 3a. Location isn't valid, nothing happens. 3b. Cannot reach enemy unit. 5a. King is not under check, cannot checkmate. 5b. King can escape or can be saved by another unit.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.4.6 Play with Human

Table 2.6 – UC6 Play with Human

UC-6 name: Play with Human	
Primary actor	User.
Stakeholder & Interests	User will be able to start the chess game against human player.
Pre-condition	1. Game software should be running.
Post-condition	Game has started successfully and white Player can play the first move.
Main Success Scenario	1. User selects the Start Game button. 2. System loads Game. 3. User selects human opponent 4. Game starts.
Alternate Flows	1a. User presses Start Game, system crash. 2a. System cannot load Game into memory.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.4.7 Play with Computer

Table 2.7 – UC7 Play with Computer

UC-7 name: Play with Computer	
Primary actor	User.
Stakeholder & Interests	User will be able to start the chess game against Computer player.
Pre-condition	1. Game software should be running.
Post-condition	Game has started successfully and white Player can play the first move.
Main Success Scenario	1. User selects the Start Game button. 2. System loads Game. 3. User selects computer opponent 4. Game starts.
Alternate Flows	1a. User presses Start Game, system crash. 2a. System cannot load Game into memory.
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

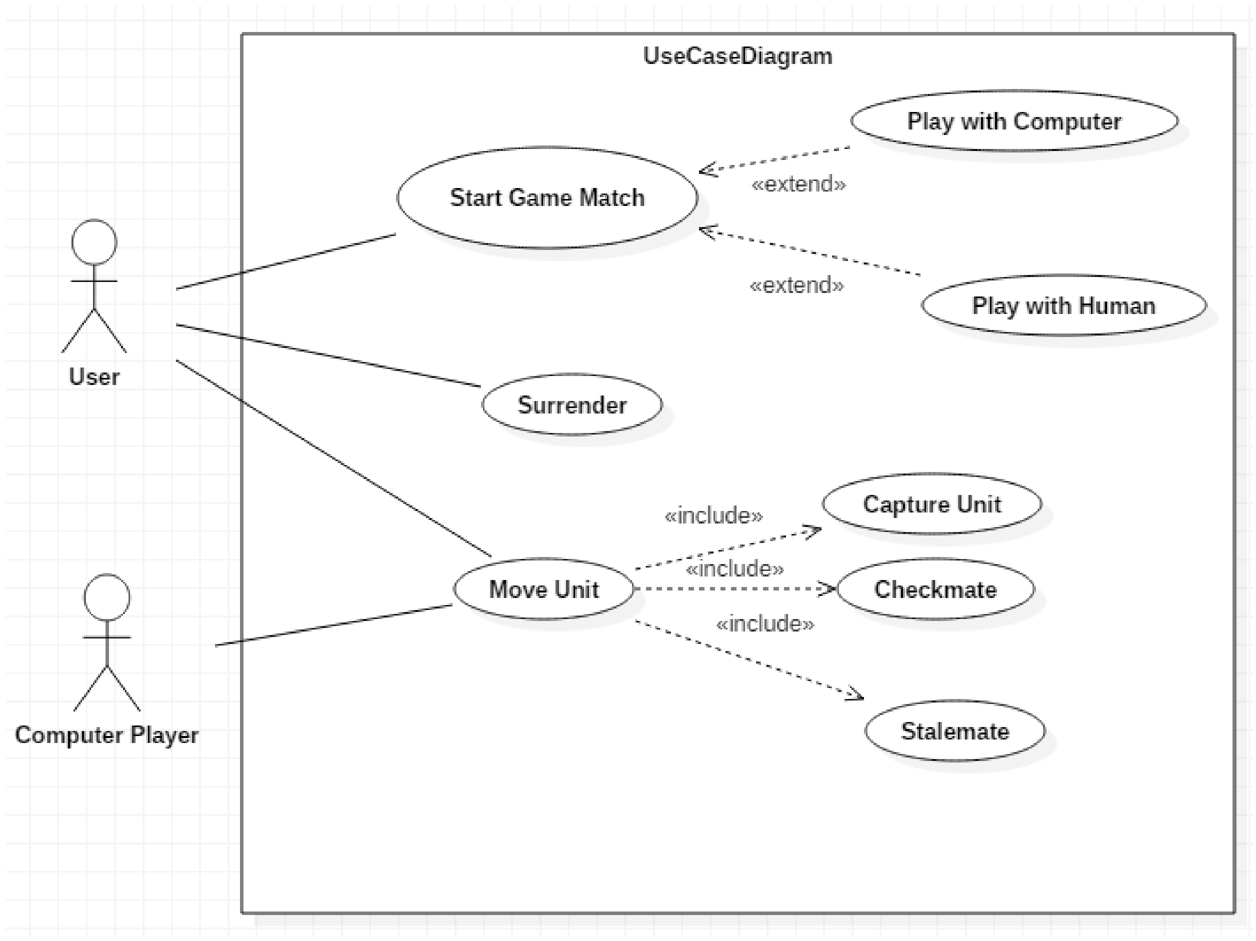
2.4.8 Stalemate

Table 2.8 – UC8 Stalelmate

UC-8 name: <u>Stalemate</u>	
Primary actor	User and Computer Player.
Stakeholder & Interests	User or Computer will be able stalemate the game resulting in draw.
Pre-condition	1. Game match should be started.
Post-condition	Stalemate condition is reached.
Main Success Scenario	1. User or Computer selects the Unit. 2. User or Computer selects the location or enemy unit. 3. Game checks validity of move, no further moves can be played. 4. Stalemate happens.
Alternate Flows	1a. User selects opponents unit, nothing happens. 1b. User doesn’t select any unit..
Special Requirements	None.
Technology	Keyboard and mouse. LCD display for output.
Frequency	Multiple times.

2.5 Use case diagram

Figure 2.1 – Use Case Diagram



2.6 Software System Attributes

Software system attributes define overall factors that affect run-time behavior, software design, and user experience. To develop high quality game, software system attributes are the benchmarks that describe system's intended behavior within the environment for which it was built. Here is the list of some software system attributes.

2.6.1 Reliability

The system should give running time of over 10 hours continuous and never crash or hang, other than as the result of an operating system error.

2.6.2 Availability

Once the game is installed, game should not take more than 10 seconds to reach the main menu upon launching. Load time for game session start is almost instantaneous.

2.6.3 Security

There are no such security constraints.

2.6.4 Maintainability

- Files will be well commented.
- Authorship will be mentioned.
- Object oriented design will be used.
- Camel Case naming convention will be used.
- Updates to the software will also be documented and kept in a separate files as logs.

2.6.5 Portability

The system should be portable to windows 8 and later versions. No other specific portability requirements have been identified.

2.6.6 Performance

Following are the performance requirements of this game

- **Frame rate**: e-Chess shall execute with frame rate of at least 30 frames per second.
- **Processing Power:** e-Chess will only require a 20 MHz processor to run smoothly.
- **Executable Size:** The size of e-Chess executable will be less than 100 Megabytes.
- **In-game Load Times:** e-Chess's load time to start a match will be near instantaneous.
- **Launch Time:** The time to reach main menu on launching the game will not exceed 10 seconds.

Chapter 3

Software Design Description

3.1 Design

3.1.1 Introduction

Then purpose of this chapter is to provide assistance in implementation of the Software e-Chess. It provides information regarding the classes, components, flow of activities and as well as a narration of the screen displays. It lists diagrams such as usecase diagram, domain model, sequence diagrams, class diagram, activity diagram, component diagram.

3.1.2 Design Overview

The e-Chess uses object oriented programming approach. Each chess piece is a unit of either white or black color. The player has the choice to play against another human player or play against a computer player. The computer player is driven by an algorithm which incorporated Artificial intelligence. Alpha-Beta pruning[6] is used as the algorithm for computer player.

3.1.3 Pseudocode:

```
function alphabeta(node, depth, α, β, maximizingPlayer)
    if (depth = 0 or node is a terminal node) return the heuristic value of node
    if maximizingPlayer
        v := -∞
        for each child of node
            v := max(v, alphabeta(child, depth – 1, α, β, FALSE))
            α := max(α, v)
            if β ≤ α
                break (* β cut-off *)
        return v
    else
        v := ∞
        for each child of node
            v := min(v, alphabeta(child, depth – 1, α, β, TRUE))
            β := min(β, v)
            if β ≤ α
                break (* α cut-off *)
        return v
(* Initial call *): alphabeta(origin, depth, -∞, +∞, TRUE)
```

3.1.4 Requirements Traceability Matrix

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or development team and their traceability in a single document delivered at the conclusion of the life-cycle.

In other words, it is a document that maps and traces user requirement with test cases. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should miss while testing.

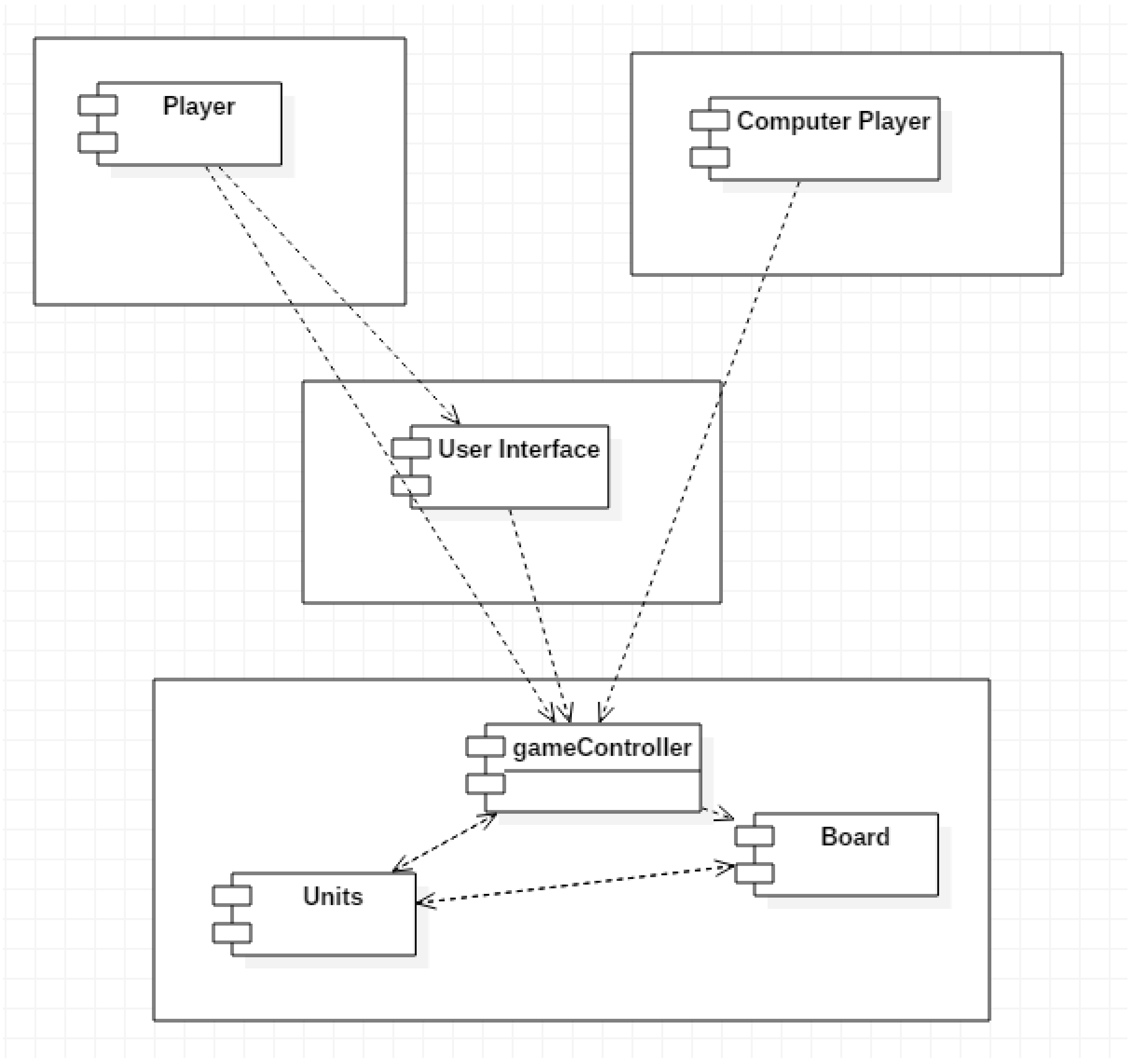
Table 3.1 – Requirements Traceability Matrix

Requirements Traceability Matrix				
Project Name:	e-Chess			
Use Cases	Domain Model	Sequence Diagram	Class Diagram	Test Cases
UC1	Figure 7	Figure 8	Figure 14	TC-1
UC2	Figure 7	Figure 10	Figure 14	TC-2
UC3	Figure 7	Figure 10	Figure 14	TC-3
UC4	Figure 7	Figure 11	Figure 14	TC-4
UC5	Figure 7	Figure 13	Figure 14	TC-5
UC6	Figure 7	Figure 8	Figure 14	TC-7
UC7	Figure 7	Figure 9	Figure 14	TC-8
UC8	Figure 7	Figure 12	Figure 14	TC-6

3.1.5 System Architectural Design

The major system architecture is described with the diagram. Major functionality is separated in rectangles and components reside in them. Direction of arrows shows the interaction. Interaction can exist two ways between components

Figure 3.1 – System Architecture Diagram



3.1.6 System Architecture Tier

This software is 2-tier architecture. It does not incorporate a database as the game does not require any long term storage facility for its functionality.

3.1.7 System Interface Description

The hardware interface includes a computer while the software interface includes Windows operating system for windows. As the game is made on Unity3D software, it will be easily built into an executable without errors.

3.2 Detailed Description of Components

The following are the components:

3.2.1 Units

Units are the chess pieces; there are 16 on each side. On one side there are 8 Pawns, 2 rooks, 2 knights, 2 bishops, one queen and one king. Each unit is designated with these types by an attribute which distinguishes them.

3.2.2 Player

The player component is to provide an interface for the human player. The player's actions are simulated through this component.

3.2.3 Computer Player

The computer player is an artificial player that the player can decide to compete against. This component does not require visual interfaces, so it can directly communicate with the game controller. It uses Alpha-Beta pruning algorithm to look ahead at possible moves the human player can attempt. Then by associating values with benefit or fitness for its own victory it decides the best possible move it should play.

3.2.4 Game Controller

It is the brain of the game, it will make the calls to check whether moves played by the player are valid or not, whether they lead to a checkmate, capture or stalemate. It connects most of the components together and controls the flow of the game.

3.2.5 Board

This component maintains the collection of units and their current locations. It is responsible for calculating a checkmate and stalemate because it knows their locations. The game controller calls its functionalities after each player turn.

3.2.6 User Interface

The user interface takes input from the user and displays the game to them. It shows all the simulation in a visual form. This will be done by Unity3D itself.

3.3 User Interface Design

The characteristics of the interfaces and how the user interacts with the software are discussed here.

3.3.1 Description of User Interface

The user can interact by clicking with the mouse on windows operating system. At start the user will select which type of opponent he wants to play against; a human or computer opponent. After that the game will start. The player will click a unit and select a valid tile on the board to move the unit, and then the other player's turn will come. If the opponent is a computer then the move will be played automatically and the player's turn will come again. But if the opponent is a human then the mouse control will be controlling the other color team's units, when the opponent player plays his move then the control will revert to the first player.

3.3.2 Screen Images

Following are the screen images of the game.

Figure 3.2 – Screen Image 1

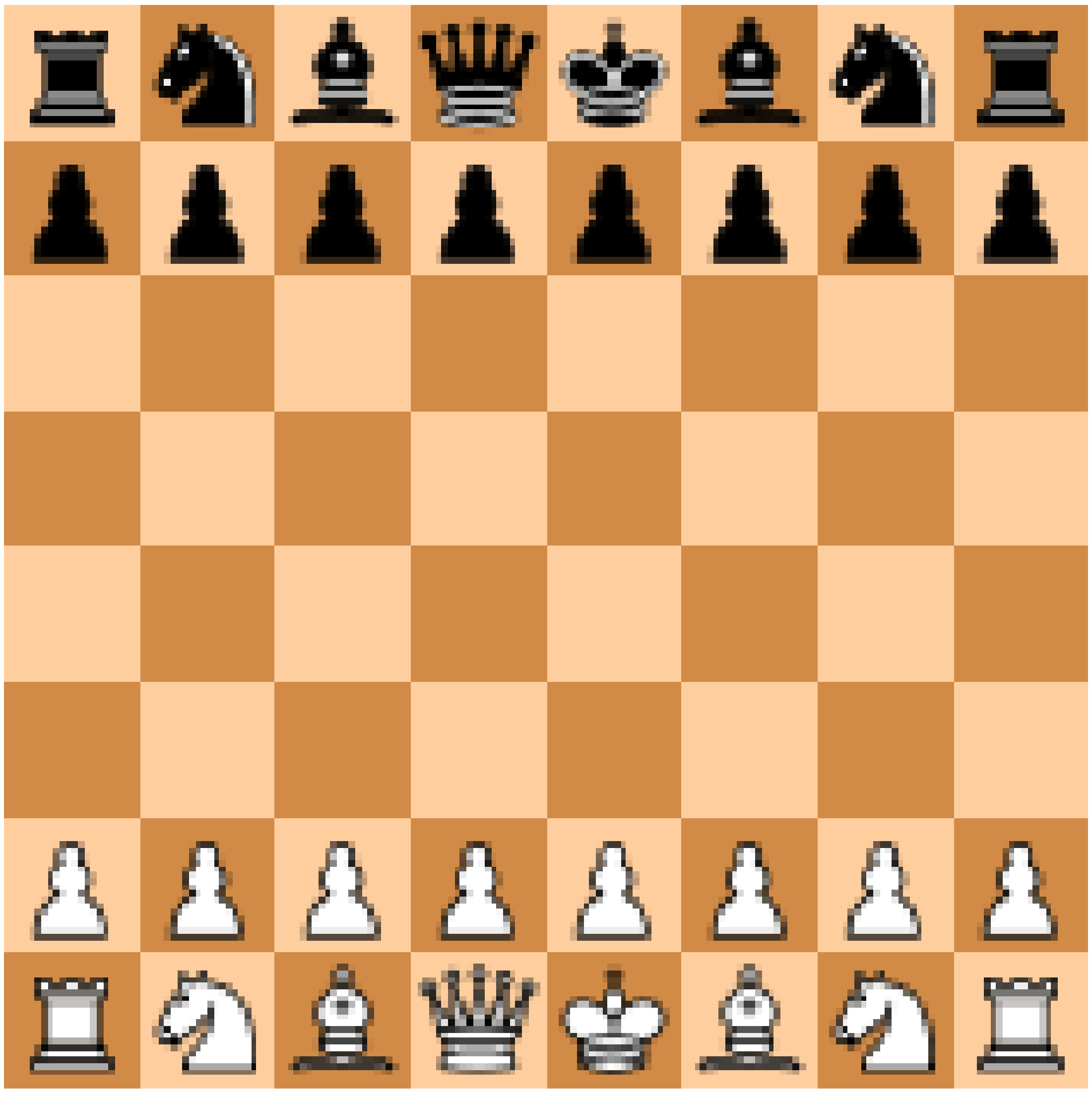


Figure 3.3 – Screen Image 2

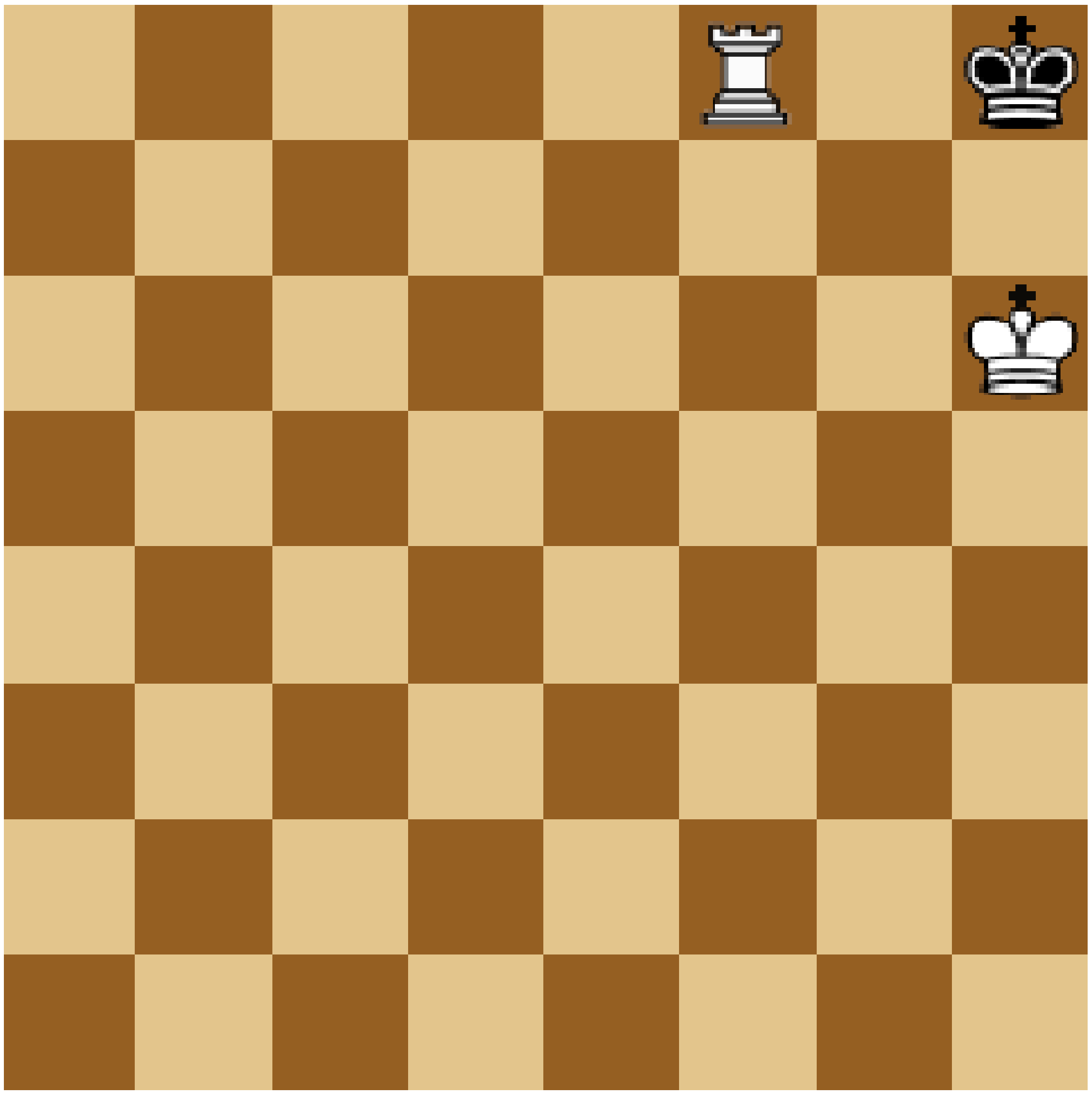
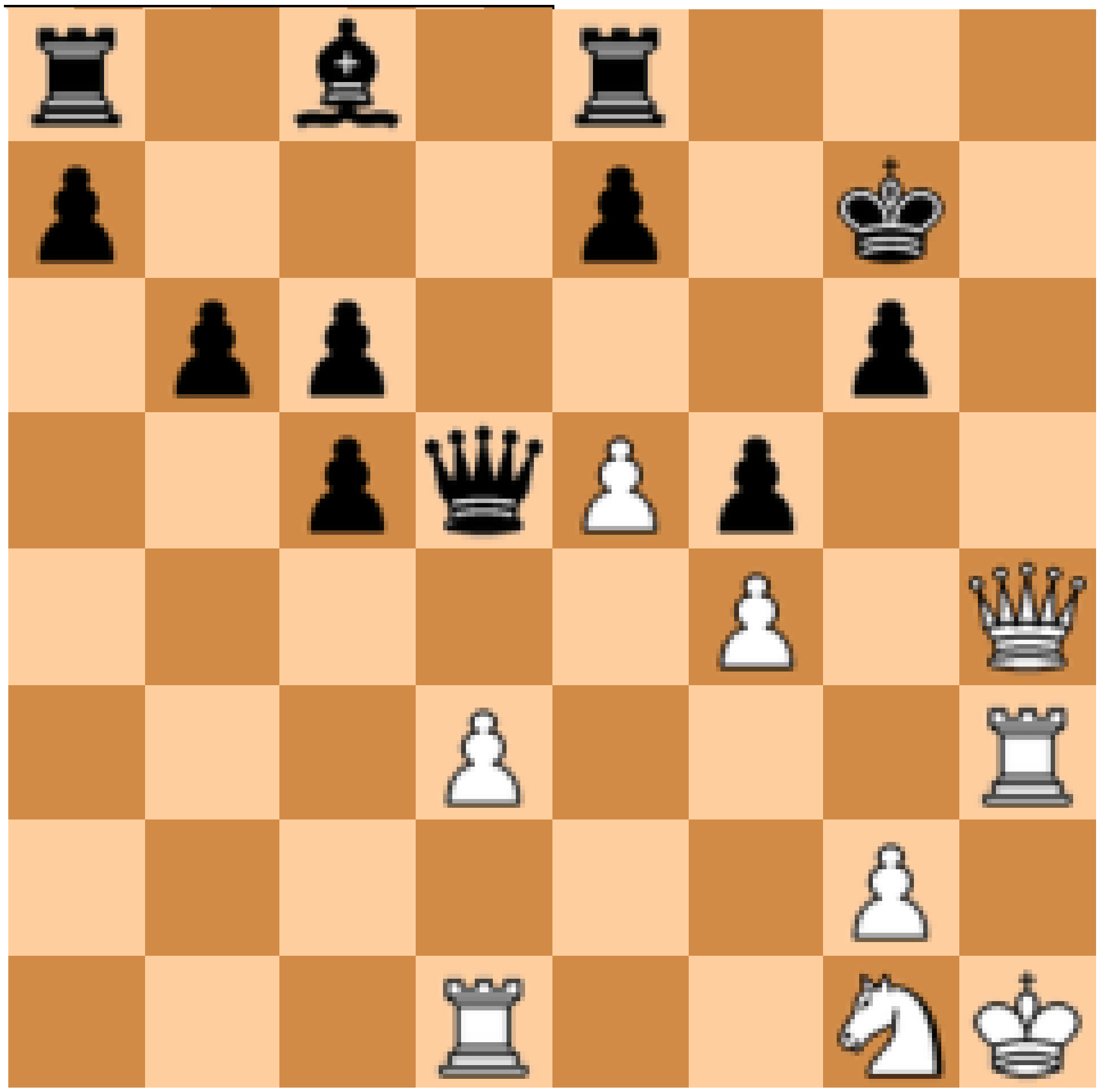
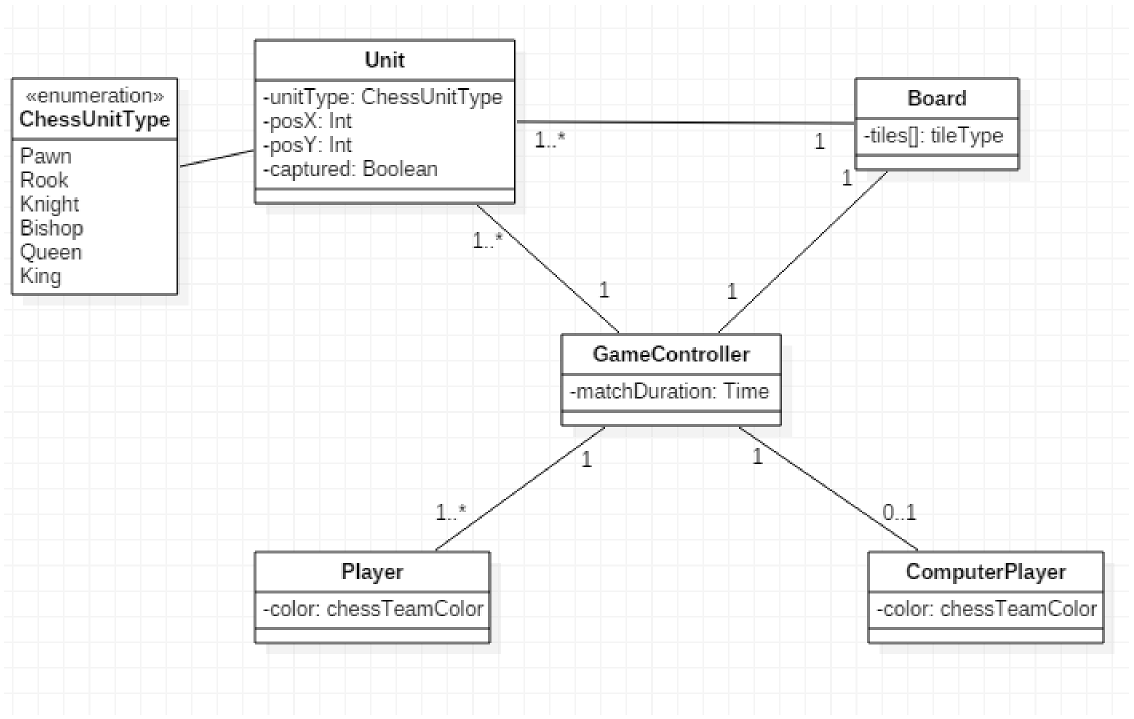


Figure 3.4 – Screen Image 3



3.4 Domain Model

Figure 3.5 – Domain Model



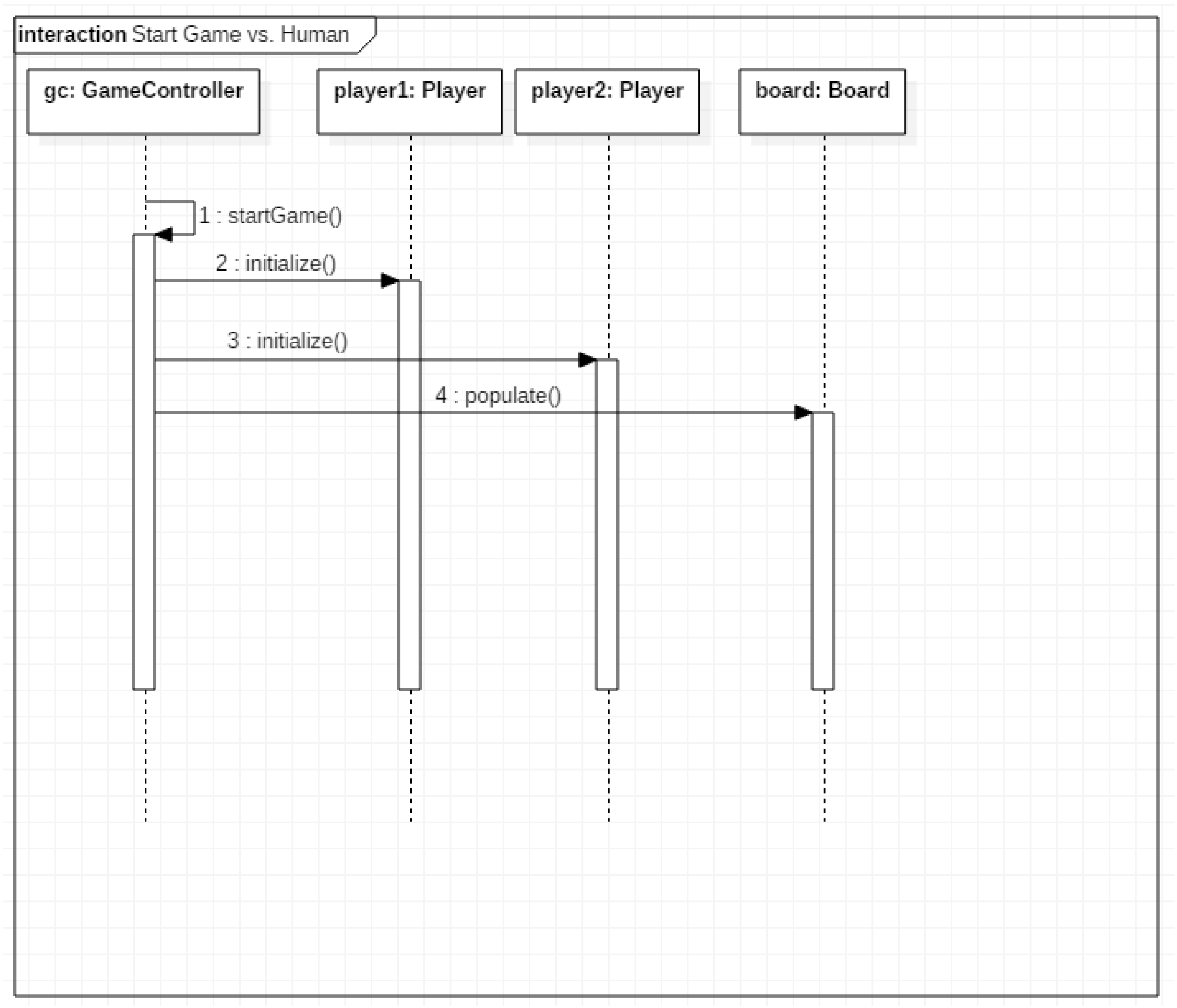
3.5 Sequence Diagram

Sequence diagrams display the sequence of interactions between different objects. A sequence per each usecase is made.

3.5.1 Start Game vs. Human

Player starts the game and selects human opponent. The GameController class initializes the game setting and populates the board.

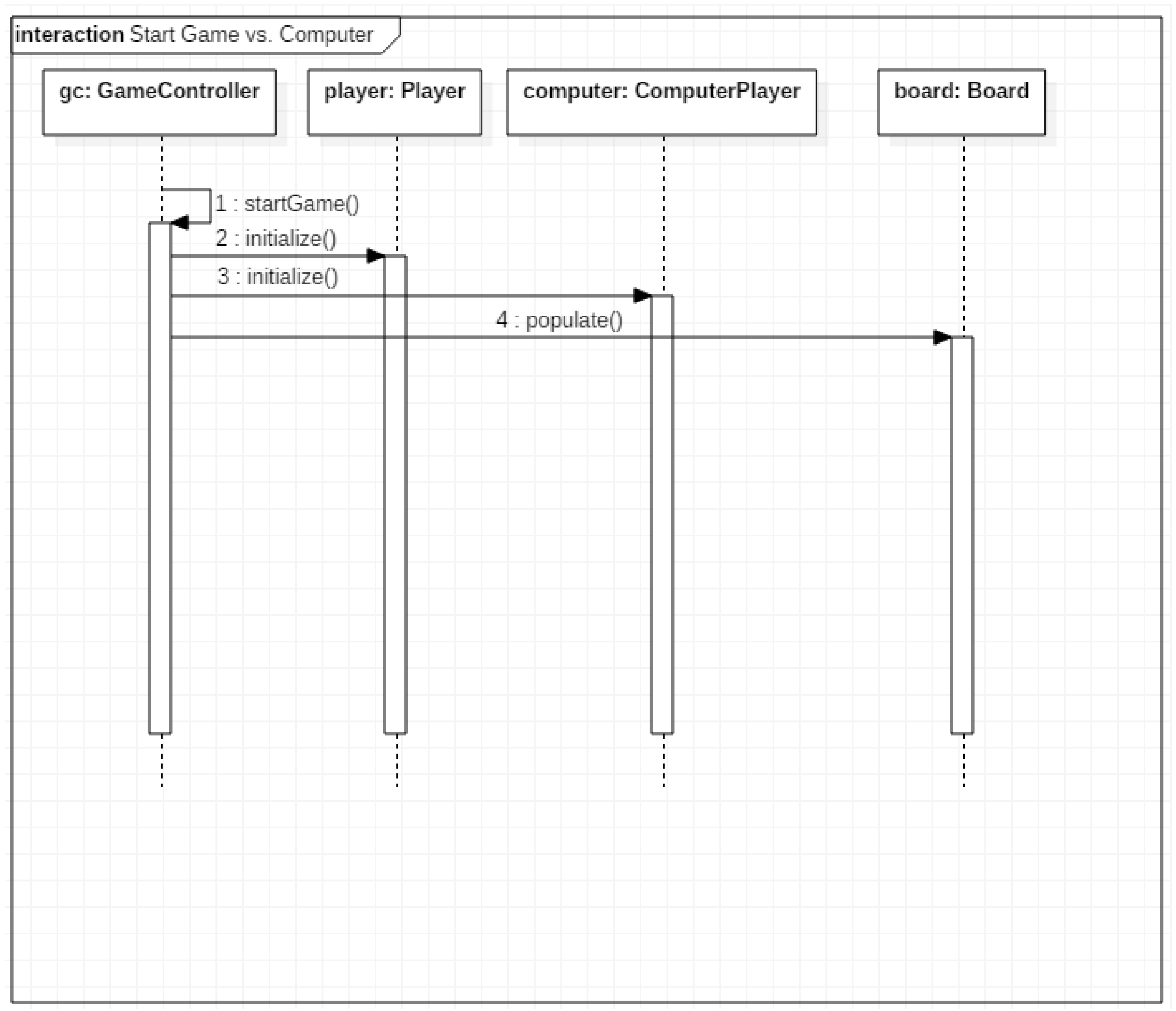
Figure 3.6 – Sequence Diagram: Start Game vs Human



3.5.2 Start game vs. Computer Player

Player starts the game and selects Computer Opponent

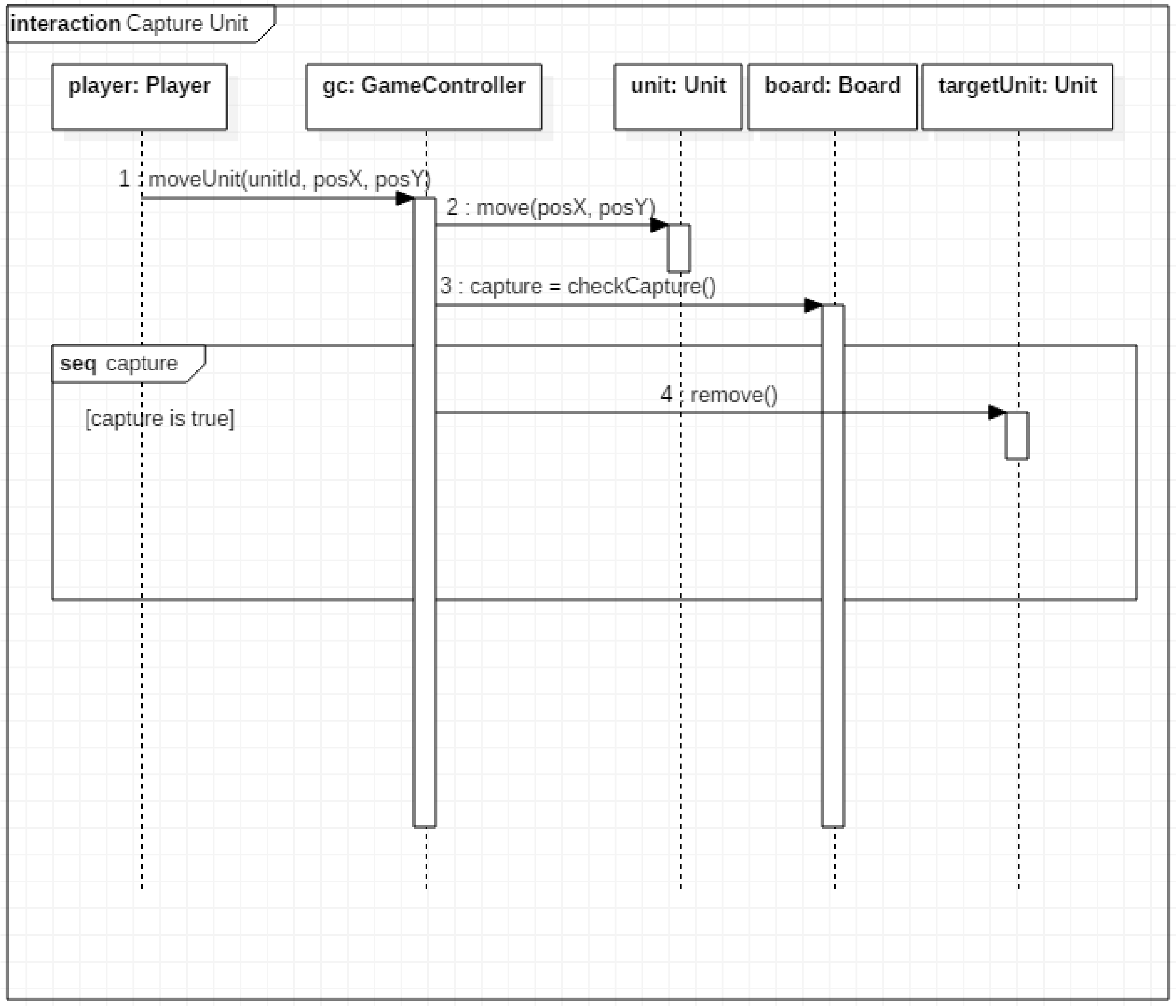
Figure 3.7 – Sequence Diagram: Start Game vs Computer Player



3.5.3 Capture Unit

The player or Computer moves a unit and captures the enemy unit.

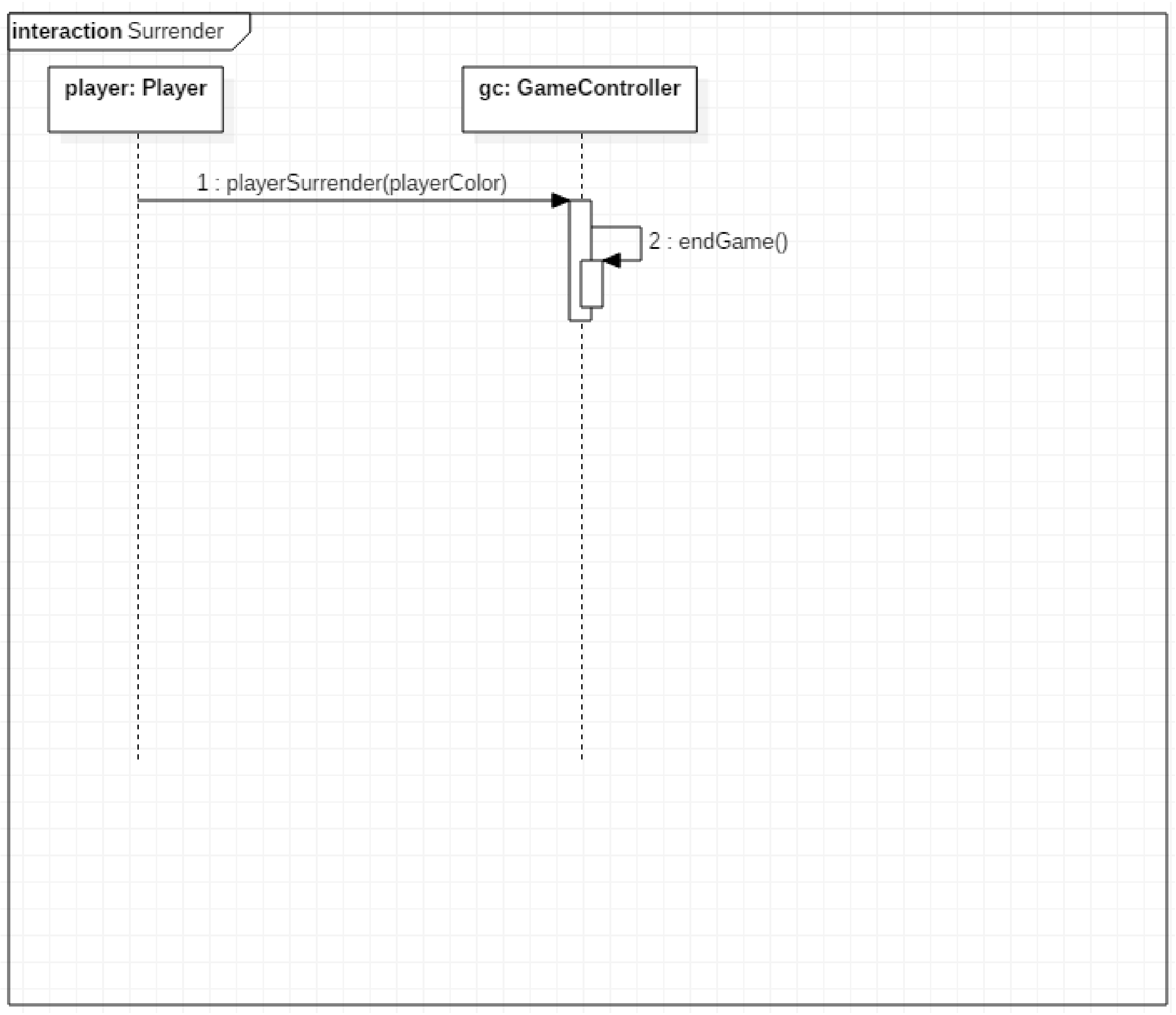
Figure 3.8 – Sequence Diagram: Capture Unit



3.5.4 Surrender

Player surrenders from the game.

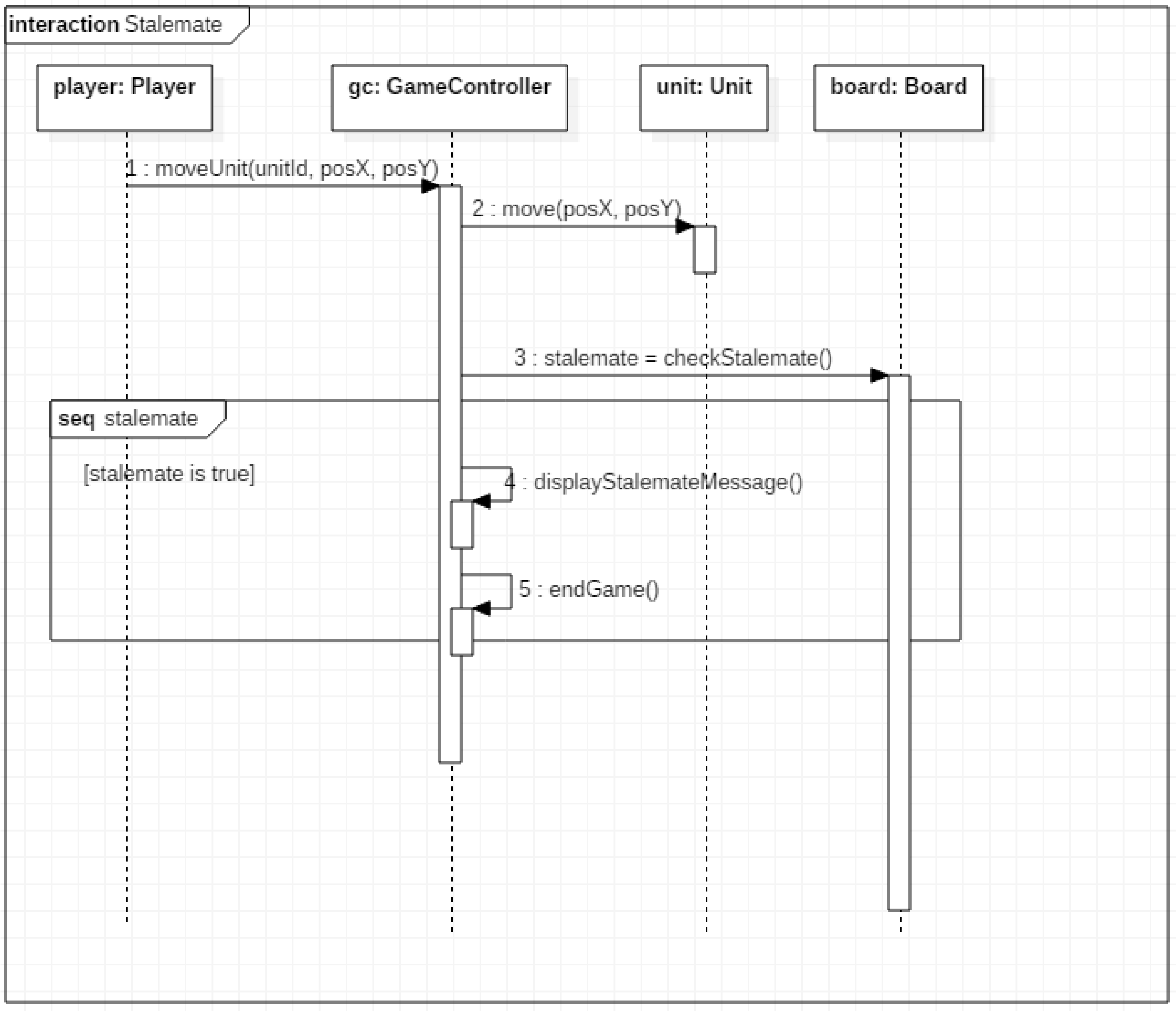
Figure 3.9 – Sequence Diagram: Surrender



3.5.5 Stalemate

The game moves to a position where no further moves can be played and no player resulted in victory.

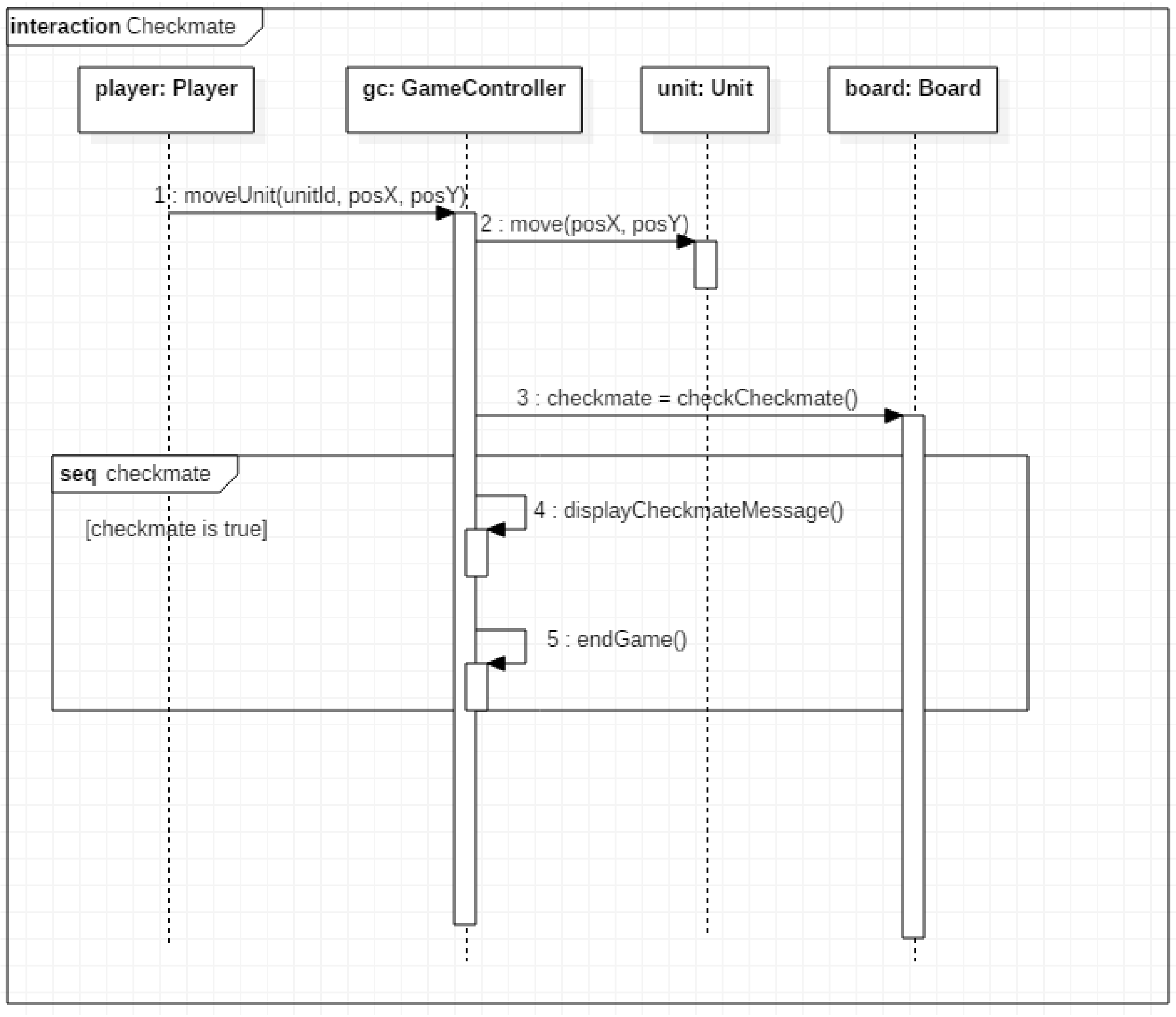
Figure 3.10 – Sequence Diagram: Stalemate



3.5.6 Checkmate

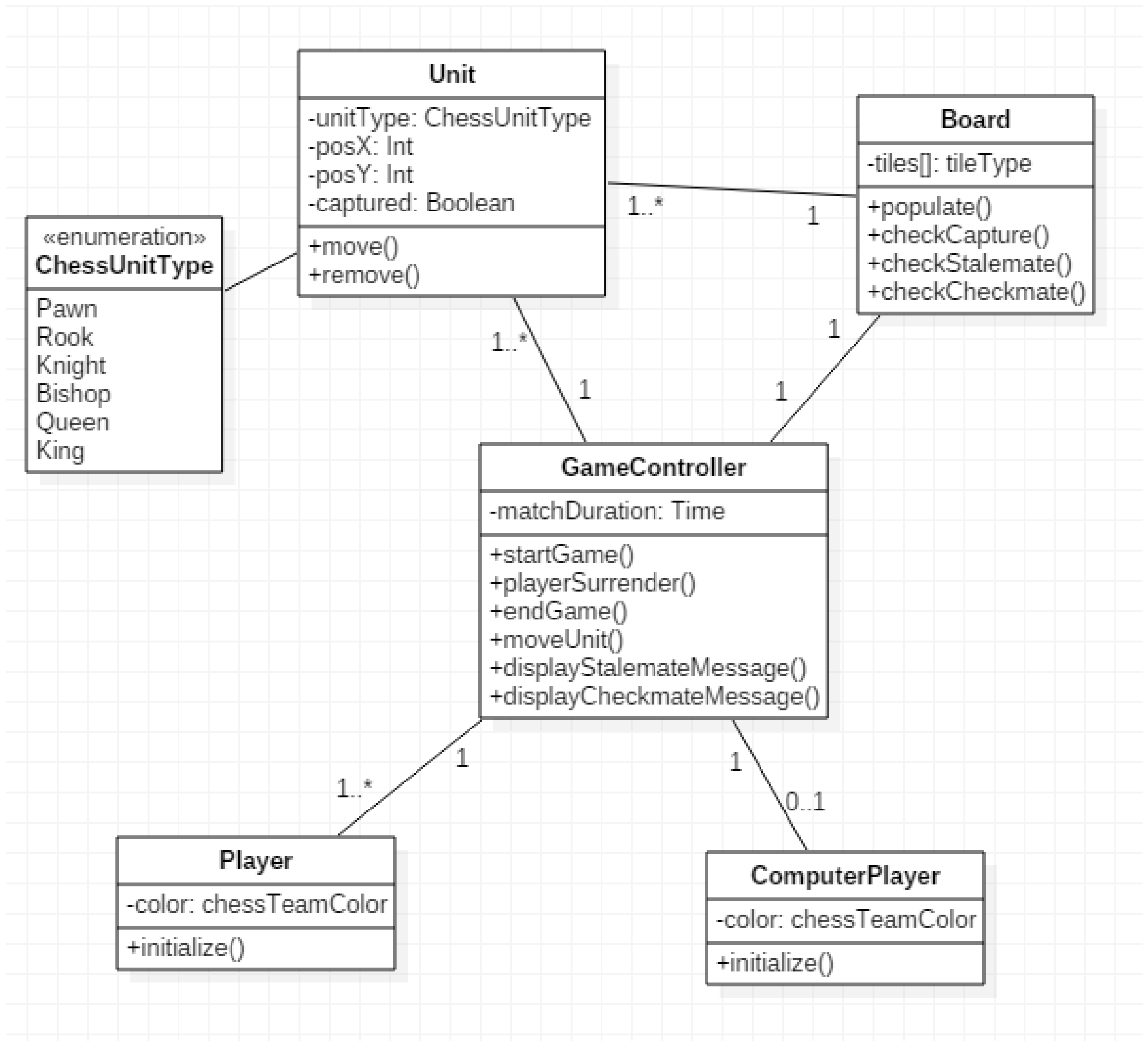
The game moves into a position where one team's king is in check and cannot evade it.

Figure 3.11 – Sequence Diagram: Checkmate



3.6 Class Diagram

Figure 3.12 – Class Diagram



Chapter 4

Software Test Documentation

4.1 Testing Introduction

The purpose of this chapter is testing of the software. All components and all use cases are tested to see if they execute successfully according to the required functionality. It is very crucial to the success of a software that it works correctly as intended. Bugs and errors are very dangerous as they can break the whole software and cause it to be unusable.

4.1.1 Test Approach

UAT or user acceptance testing is used the method used, it is also called beta testing. The solution which works for the users is verified. That means that we don’t check whether our software will have memory leaks or if it crashes, but rather we check correct working for the user of the software.

4.2 Test Plan

4.2.1 Features to be tested

1- Start game
2- Choose opponent
3- Move unit
4- Capture unit
5- Surrender
6- Stalemate
7- Checkmate

4.2.2 Features not to be tested

1- Memory consumed
2- Frame rate of the game
3- Memory leaks

4.3 Test Cases

Table 4.1 –Test Case 1

Test Case 1:	
Purpose	Start game
Setup	1. Launch game.
Instructions	1. Select opponent
Expected Result	Game starts successfully.
Observed Result	Game starts successfully on testing.
Frequency	Pass: 1 Fail: 1
Verdict	Passed

Table 4.2 –Test Case 2

Test Case 2:	
Purpose	Move Unit
Setup	1. Game is started.
Instructions	1. Select a unit 2. Select a valid tile
Expected Result	Selected unit moves to that tile.
Observed Result	Unit moves successfully.
Frequency	Pass: 2 Fail: 4
Verdict	Passed

Table 4.3 –Test Case 3

Test Case 3:	
Purpose	Capture Unit
Setup	1. Game is started.
Instructions	1. Move a unit to enemy unit location.
Expected Result	Enemy unit is captured and removed from the board.
Observed Result	Unit is captured and removed from the board successfully.
Frequency	Pass: 1 Fail: 3
Verdict	Passed

Table 4.4 –Test Case 4

Test Case 4:	
Purpose	Surrender
Setup	1. Game is started.
Instructions	1. Select option to surrender
Expected Result	Game match is surrendered.
Observed Result	Game match is surrendered successfully.
Frequency	Pass: 1 Fail: 1
Verdict	Passed

Table 4.5 –Test Case 5

Test Case 5:	
Purpose	Checkmate
Setup	1. Game is started. 2. Game is played to a limit where enemy can be checkmated in next move.
Instructions	1. Move unit to checkmate enemy king.
Expected Result	Game match is won.
Observed Result	The king is placed in a checkmate and the game is won successfully.
Frequency	Pass: 1 Fail: 2
Verdict	Passed

Table 4.6 –Test Case 6

Test Case 6:	
Purpose	Stalemate
Setup	1. Game is started. 2. Game is played to a limit where game can be turned into a stalemate in next move.
Instructions	1. Move unit to cause stalemate.
Expected Result	Game match is finished with a draw.
Observed Result	Game is concluded to a stalemate successfully.
Frequency	Pass: 1 Fail: 2
Verdict	Passed

Table 4.7 –Test Case 7

Test Case 7:	
Purpose	Play with Human
Setup	1. Game is launched.
Instructions	1. Select Human opponent.
Expected Result	Game starts against human opponent.
Observed Result	Game starts against a human opponent successfully.
Frequency	Pass: 1 Fail: 1
Verdict	Passed

Table 4.8 –Test Case 8

Test Case 8:	
Purpose	Play with Computer
Setup	1. Game is launched.
Instructions	1. Select Computer opponent
Expected Result	Game starts against computer opponent.
Observed Result	Game starts against computer opponent successfully.
Frequency	Pass: 1 Fail: 1
Verdict	Passed

Chapter 5

Implementation Details

5.1 Implementation Introduction

This chapter provides information about the implementation of e-Chess. Implementation is the formation of technical specifications and algorithms into a working physical form as a program or as software. After the implementation the game is a usable 3D replica of the real world chess game.

5.2 Framework

The game is developed using in Unity3d version 5.5. It works with 3D objects placed inside a 3D environment. Uses technologies of lighting, material and camera control. For the programmer's own functionality to be incorporated, scripts can be added to each game object and those scripts are written in one of a few languages. e-Chess uses C# for its scripting.

5.3 Operation System

This game is made for Windows Operating System. But if required in the future, it can be easily ported to other platforms as well because unity3d is flexible in building the project into a number of operating systems.

5.4 Programming Language

C# is used to program this game. Since it is made in unity3d, it can be made in C# and javascript. Since C# is strict about datatypes and supports object orientation more clearly, it is a better choice for a game that involves a lot of checks after each player move. It is also fairly easier to debug.

5.5 Screenshots

Figure 5.1 – Screenshot 1

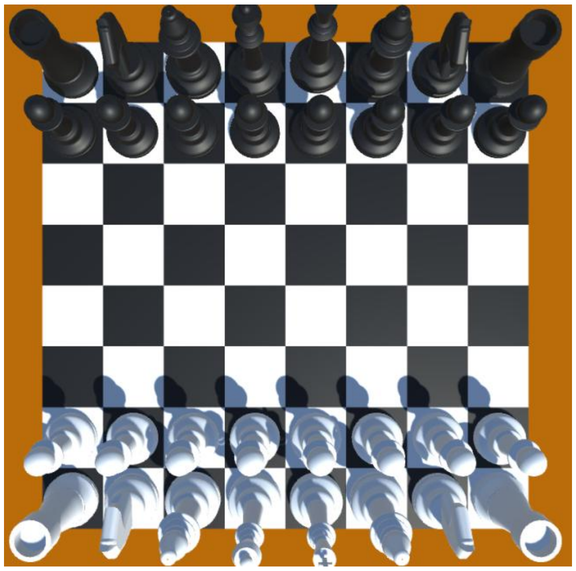


Figure 5.2 – Screenshot 2

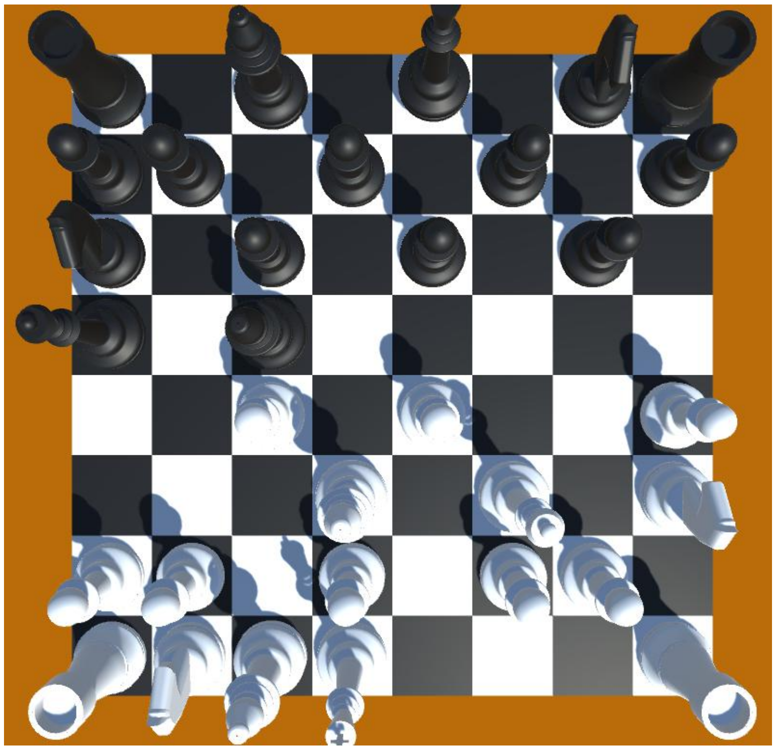


Figure 5.3 – Screenshot 3

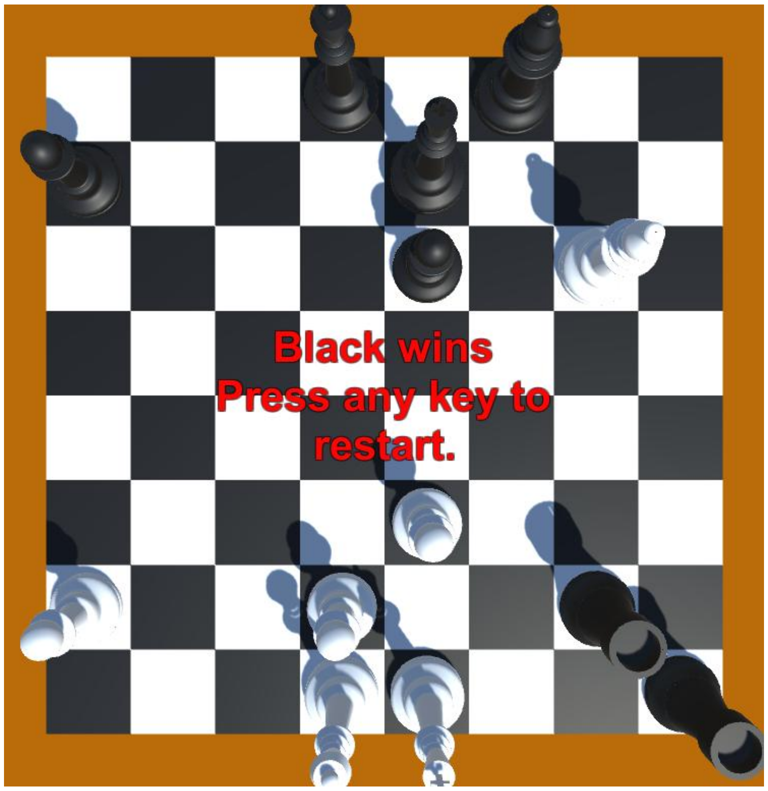


Figure 5.4 – Screenshot 4

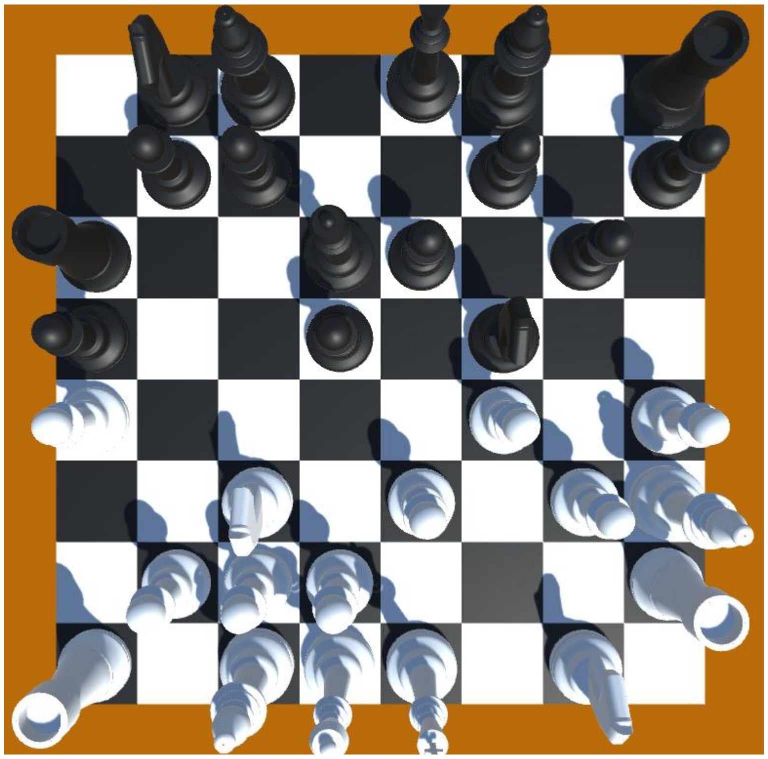


Figure 5.5 – Screenshot 5

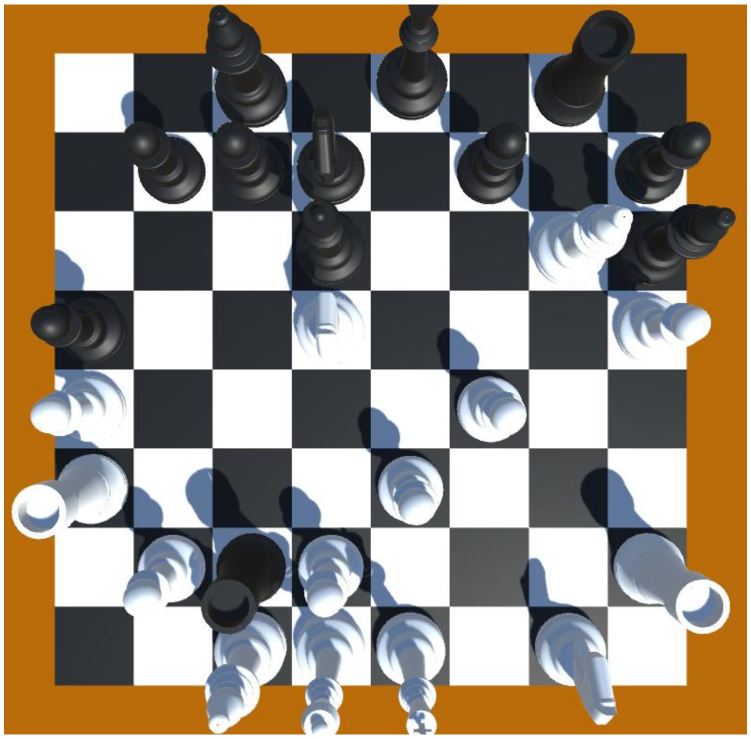


Figure 5.6 – Screenshot 6

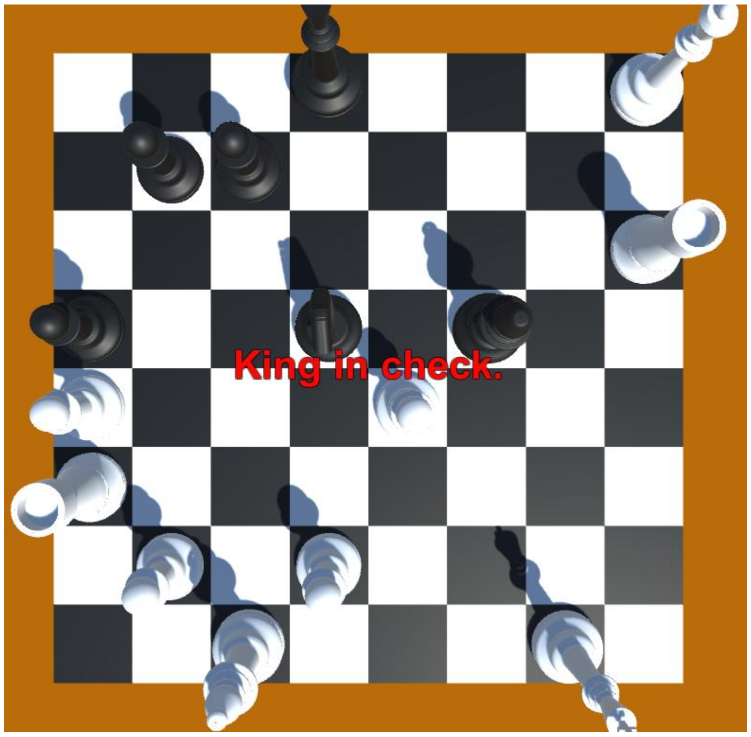
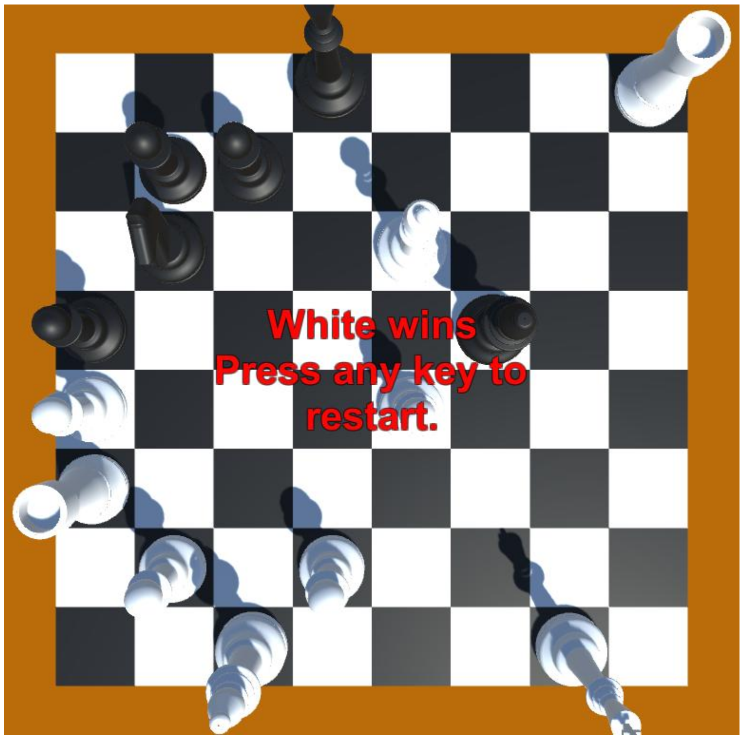


Figure 5.7 – Screenshot 7



Chapter 6

Conclusion and Future Work

6.1 Summary

e-Chess is a software game which is an implementation of the traditional chess game played by billions around the world. Chess is played between two players. Each player has 16 units at his disposal. The units for each player are 8 pawns, two knights, two bishops, two rooks, one queen and one king. Each type of unit has its own rules for movement. Because of these movement constraints, the game is a tactical and strategical arena for players to fight with their brains.

This game software is in 3D so the player feels like he is interacting with real-like chess pieces. The player views the chess board from the top and all the units cast shadows like real pieces on a real board.

6.2 Conclusion

Chess is globally rendered as strong competitive game with its own world ranking system. In its core it is built upon elements that force the players to play strategically using their mind. World famous chess players are widely known as geniuses. The ability to think many steps ahead while keeping in mind the resources at yours and opponent’s disposal is not an easy feat. e-Chess provides its player to a part of that experience and to hone their skills sitting at their own computers.

6.3 Future Work

e-Chess is currently played on one computer. For its future, an element of online play can be added to it. This will allow players to play with each other across the world sitting from their own homes at their own luxury. Unity3d has its own networking facilities which will have to be explored for this purpose.

7. ADDITIONAL MATERIAL

There are no additional requirements.

8. References

[1] (2016, Nov.) Wikipedia. [Online]. HYPERLINK "https://en.wikipedia.org/wiki/Chess" https://en.wikipedia.org/wiki/Chess

[2] (2016, Nov.) Wikipedia. [Online]. HYPERLINK "https://en.wikipedia.org/wiki/Chaturanga" https://en.wikipedia.org/wiki/Chaturanga

[3] (2016, Nov.) Wikipedia. [Online]. HYPERLINK "https://en.wikipedia.org/wiki/Xiangqi" https://en.wikipedia.org/wiki/Xiangqi

[4] (2016, Nov.) Wikipedia. [Online]. HYPERLINK "https://en.wikipedia.org/wiki/Janggi" https://en.wikipedia.org/wiki/Janggi

[5] (2016, Nov.) Wikipedia. [Online]. HYPERLINK "https://en.wikipedia.org/wiki/Shogi" https://en.wikipedia.org/wiki/Shogi

[6] (2017, Feb.) Wikipedia [Onilne]. HYPERLINK https://en.wikipedia.org/wiki/**Alpha–beta_pruning**

https://en.wikipedia.org/wiki/**Alpha–beta_pruning**

Book: C. Larman, APPLYING UML AND PATTERNS An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed., Massachusetts: Pearson Education, 2005

Book: Roger S. Pressman, Software Engineering - A Practitioner's Approach, McGraw Hill, 7th Edition, 2010