
CHECKERS 3D

**A Project Report**

*On*

# "CHECKERS 3D"

*by*


## SHIVAM PANDYA

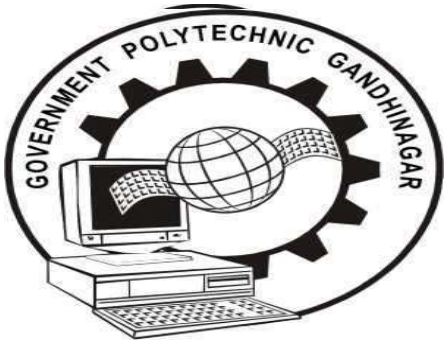
6th Sem I.T.


*In partial fulfillment for the award of the degree:*

## DIPLOMA ENGINEERING

*In*

## Information Technology





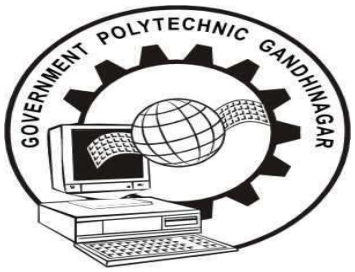
**Government Polytechnic, Gandhinagar**

*under*

**Gujarat Technological University, Ahmedabad**


June - 2023

# CERTIFICATE




**Government Polytechnic, Gandhinagar**
Information Technology
2023

This is to certify that Mr. **SHIVAM PANDYA** from Government Polytechnic, Gandhinagar having Enrollment No. **206230316150** has completed final project report having title "**CHECKERS 3D**" consulting individually ~~or in a group~~ under the guidance of the faculty named Mr. Keyurbhai Jani, during the term Jan-2023 to Jun-2023.

**Place:** GP, Gandhinagar

**Date:**

| **Internal Guide** | **Head of the Department** |
|---|---|
| MR. KEYURBHAI JANI | MS. HIRALBEN PATEL |

# Table of Contents

# List of Figures

# Acknowledgements

I would like to express my heartfelt gratitude to everyone who has contributed to the development and realization of "Checkers 3D." This project would not have been possible without the support, expertise, and encouragement of numerous individuals and resources.

First and foremost, I extend my deepest appreciation to my **project supervisor, Mr. Keyurbhai Jani**, for their invaluable guidance, mentorship, and continuous support throughout the development process. Their expertise and insightful feedback have greatly influenced the success of this project.

I also would like to extend my gratitude to our **Head of the Department, Ms. Hiralben Patel**, for their continuous support and encouragement throughout the development of "Checkers 3D." Their vision, guidance, and trust in our abilities have been instrumental in shaping this project and fostering an environment conducive to innovation and creativity.

Furthermore, would like to express my gratitude to the open-source community, which has provided a wealth of resources, libraries, and frameworks that have significantly expedited the development process. The contributions of countless developers and contributors have enriched the ecosystem of game development and have been indispensable in creating "Checkers 3D."

Lastly, would like to thank my family, friends, and loved ones for their unwavering support, understanding, and encouragement throughout this journey. Their belief in my abilities and their patience during long hours of development have been a constant source of motivation.

I am immensely grateful for the collective effort, dedication, and support from all those involved in making "Checkers 3D" a reality. It has been an incredible learning experience, and I look forward to the future iterations and enhancements that will further elevate the game.

Thanking you all for being a part of this journey.

# Abstract

CHECKERS 3D is an innovative and exciting remake of the classic board game American Checkers or Draughts, developed using Unity game engine and programmed with C#. This multiplayer game features a turn-based gameplay where players compete against each other to capture their opponent's pieces and progress diagonally across the board. The ultimate goal is to either capture all of the opponent's pieces or block them from making any further moves.

One of the key features of CHECKERS 3D is its engaging and intuitive user interface, which allows players to easily navigate the game board and make their moves. Additionally, the game offers players the ability to chat with each other in real-time during gameplay, thanks to the instant messaging feature.

The game also includes a local multiplayer mode, which enables players to play against each other on the same device, making it an ideal game for friends and family to play together.

This documentation provides a comprehensive overview of CHECKERS 3D, including its requirements, specifications, and mapping, as well as detailed explanations of the various components of the game. The testing and quality assurance procedures ensure that the game functions seamlessly and provides players with a fun and challenging gaming experience.

# CHAPTER 1: INTRODUCTION

## 1.1 Project Description

The Checkers 3D is a strategy board game which is turn based two player game. The game board has 64 squares which are alternate in color (32 green squares and 32 white squares). Each player having twelve disk shape pieces that are usually dark & light in color. Only the dark squares are used for play. The Player with white checkers moves first. Moves are allowed only on dark squares, diagonally. The player captures piece of the opponent by jumping. In one jump, only one capture is allowed. Multiple jumps are allowed in a single turn only in forward direction. The captured pieces are removed from the board.

The piece of the player which reaches the opponent side of the board is declared as King. The winner is decided when the player captures all the pieces of opponent or when the player blocks all the pieces of opponent and the opponent has no moves left.

The game will have several social features. The player will also be able to interact with one another through chat feature in the game.

## 1.2 Problem Summary

In the game of Checkers, play consists of advancing a piece diagonally forward to an adjoining vacant square. White moves first. If an opponent's piece is in such an adjoining vacant square, with a vacant space beyond, it must be captured and removed by jumping over it to the empty square. If this square presents the same situation, successive jumps forward in a straight or zigzag direction must be completed in the same play. When there is more than one way to jump, the player has a choice.

When a piece first enters the king row, the opponent's back row, it must be crowned by the opponent, who places another piece of the same color on it. In

our case when a piece reaches the back row of an opponent and becomes a king, we flip it to the side which has a crown drawn on top to be able to recognize it better. The piece, now called a king, has the added privilege of moving and jumping backward; if it moved to the last row with a capture, it must continue capturing backward if possible. A win is scored when an opponent's pieces are all captured or blocked so that they cannot move. When neither side can force a victory and the trend of play becomes repetitious, a draw game is declared.

**[Figure 1: Image of two players playing checkers board game]**

## 1.3 Project Scope

### 1.3.1 Justification

Games have been an integral part of everyone's life since childhood. However, the current trend in the game industry leans towards violent and aggressive content. It is refreshing to have a reboot of one of the most popular non-violent yet enjoyable games, such as Checkers. Checkers 3D is a digital adaptation of the classic board game, and every effort has been made to maintain the authenticity of the game while bringing it into the digital age. My aim is to keep the nostalgia of the traditional game alive while introducing new features and enhanced gameplay to provide an enjoyable experience to players.

### 1.3.2 Scope Description

The scope of "CHECKERS 3D" is to create a digital version of the popular game American Checkers or Draughts in Unity game engine. The game is turn-based where two players play against each other. The objective of the game is to capture all of the opponent's pieces or block the opponent so that there are no legal moves left.

The game includes features like instant messaging, local multiplayer mode, and multiplayer mode through raw TCP web sockets. The game also includes a 3D board, which enhances the user experience.

The game development has been divided into two phases: documentation phase and development phase. The documentation phase includes defining the requirements, specifications, and game rules. The development phase includes coding, testing, and debugging of the game.

The main objectives of this project are to create a stable and enjoyable game experience for players and provide a platform for players to connect and play with each other.

The deliverables of this project include a fully functional game with features like instant messaging, local multiplayer mode, and multiplayer mode through raw

TCP web sockets. The game should be stable and free of any bugs or errors.

The exclusions of this project include any modifications to the game rules, design changes that are not feasible or necessary, and any features that are not in line with the objectives of the project.

The constraints of this project include the limited time frame for development and the limited resources available for development.

### 1.3.3 Main Objectives

Checkers 3D aims to provide players with an authentic gaming experience that stays true to the rules and mechanics of the original board game. The primary objectives of this adaptation are three-fold:

- Firstly, the game will adhere to the same rules as the board game to maintain authenticity. By implementing the traditional gameplay mechanics of checkers, players will be able to relive the nostalgic experience of the classic game, with added visual enhancements that bring the game to life in a whole new dimension.
- Secondly, the game features a multiplayer mode that allows players to challenge friends in a competitive environment. The multiplayer mode is designed to be seamless and easy to use, allowing players to quickly connect with opponents and begin playing.
- Lastly, the game includes a chat feature that enables players to communicate with each other during gameplay. This feature provides an additional level of interaction between players, enhancing the overall gaming experience and creating a sense of community among players.

### 1.3.4 Deliverables

The deliverables listed aim to ensure that the game provides an authentic and enjoyable experience for the players, while also meeting their expectations for a modern digital adaptation of the classic game of Checkers.

- Fully functional Checkers game with an intuitive user interface.
- Authentic gameplay with the same rules and strategies as the physical board game.
- Multiplayer mode allowing players to play against each other from different devices.
- Chat feature to enable players to communicate with each other during gameplay.
- Stunning 3D graphics that enhance the overall gaming experience.
- Different game modes such as local multiplayer, and multiplayer.

### 1.3.5 Exclusions

These exclusions are important to consider as they help define the scope of the game and ensure that it remains focused on its primary objectives while avoiding unnecessary features that may increase the complexity or development time.

- Non-standard rules: The game will not include any non-standard rules that are not a part of the original game of checkers.
- In-game purchases: The game will not include any in-game purchases or microtransactions.
- AI opponents: The game will not include AI opponents, and players will only be able to play against each other, locally on a single device or through multiplayer connection on two separate devices.
- Tutorials: The game will not include any tutorials or training modes to teach players how to play the game.
- Customization: The game will not include any customization options for the board or pieces.
- Sound effects: The game will not include any sound effects or music.

- Additional game modes: The game will not include any additional game modes such as timed games or games with additional challenges.
- Platform-specific features: The game will not include any platform-specific features that may be available on certain devices or operating systems.

### 1.3.6 Constraints

Constraints of "CHECKERS 3D" include:

- Technology: The game will be developed using Unity game engine and will be limited to its capabilities. Therefore, any functionality or feature that cannot be implemented within the limitations of Unity may not be feasible for this game.
- Platform: The game will be developed for desktop platforms and may not be compatible with mobile devices or other platforms.
- Game Rules: The game will be developed according to the rules of traditional Checkers game with minor modifications to accommodate the 3D environment. Therefore, any significant changes to the game rules may not be feasible or acceptable for this game.
- Resources: The game development process will be constrained by the available resources, including time, budget, and personnel. Therefore, any additional features or functionalities may not be feasible if they require extensive resources.
- User Experience: The game will be developed with a focus on providing an enjoyable and engaging experience for the players. Therefore, any features or functionalities that negatively affect the user experience may not be included in the final product.
- Compatibility: The game will need to be compatible with different hardware and software configurations to ensure a broad user base. Therefore, any feature or functionality that is not compatible with a significant number of users may not be included.

### 1.3.7 Assumptions

Here are some assumptions for "CHECKERS 3D":

- Players will have a basic understanding of the rules and gameplay of traditional checkers.
- Players will have access to a device that can run the game, such as a computer.
- The game will be developed using the Unity game engine.
- The multiplayer mode will allow players to connect with each other over a local network.
- The chat feature will enable players to communicate with each other during gameplay.
- The game will support common operating systems Windows.
- The game will be developed and tested using industry-standard development and testing methodologies to ensure high-quality and bug-free gameplay.
- The game will be visually appealing and provide a satisfying user experience.

## 1.4 Project Objective

The purpose of creating "CHECKERS 3D" as a digital adaptation of the board game is to provide a nostalgic yet modern way of enjoying the game and to promote social interaction among players. In today's fast-paced digital world, people often forget the joy of spending quality time with their loved ones playing board games. The multiplayer mode allows two players to play from the comfort of their own spaces while still interacting with each other through the chat feature. This game strives to bridge the gap between traditional board games and modern digital gaming by offering the best of both worlds.

# CHAPTER 2: SYSTEM SPECIFICATIONS

## 2.1 Hardware Requirements

- A computer or device with a compatible operating system (Windows)
- A processor with at least 1.8 GHz speed (dual-core or higher recommended).
- At least 4GB of RAM (8GB or more recommended).
- A graphics card with DirectX 11 support (Optional).
- A display with a resolution of 1280x768 pixels or higher.
- Sufficient storage space to install and run the game.

(Note that these are just the recommended requirements and the game may still run on lower-end systems, although the performance may be affected.)

## 2.2 Software Requirements

Software requirements deals with defining software resource requirements and prerequisites that are necessary to run the game on a computer to provide optimal functioning of the dynamic functionalities of the game. These requirements or prerequisites generally consists an operating system with a running UI.

The software requirements for the development are as follows:

- Tools: Unity Game Engine, Visual Studio.
- Language: C#.
- Operating System: Windows 8 or higher.

## 2.3 Requirements Analysis

Requirements analysis is a critical step in software development. Here is a brief requirements analysis for "CHECKERS 3D":

### 2.3.1 Functional Requirements

The Checkers 3D Game should have the following functional requirements:

- The game board should have 64 alternating squares of a darker and white color.
- Each player should have 12 disk-shaped pieces of dark and light color.
- The pieces should move diagonally on the dark squares only.
- Moves should be turn-based and alternate between the players.
- Players should capture the opponent's piece by jumping over it and only one capture is allowed per turn.
- Multiple jumps should be allowed in a single turn only in forward direction.
- The captured pieces should be removed from the board.
- A piece that reaches the opponent's side of the board will be declared a King.
- A player wins when they capture all of the opponent's pieces or block all of their pieces, and the opponent has no moves left.
- Players should be able to interact with each other through the chat feature.

### 2.3.2 Non-Functional Requirements

The Checkers 3D Game should meet the following non-functional requirements:

- The game should be user-friendly and easy to navigate.
- The game should be visually appealing and engaging.
- The game should have a fast response time to ensure smooth gameplay.
- The game should be bug-free and reliable.
- The game should work on a variety of devices and operating systems.

# CHAPTER 3: SYSTEM DESIGN

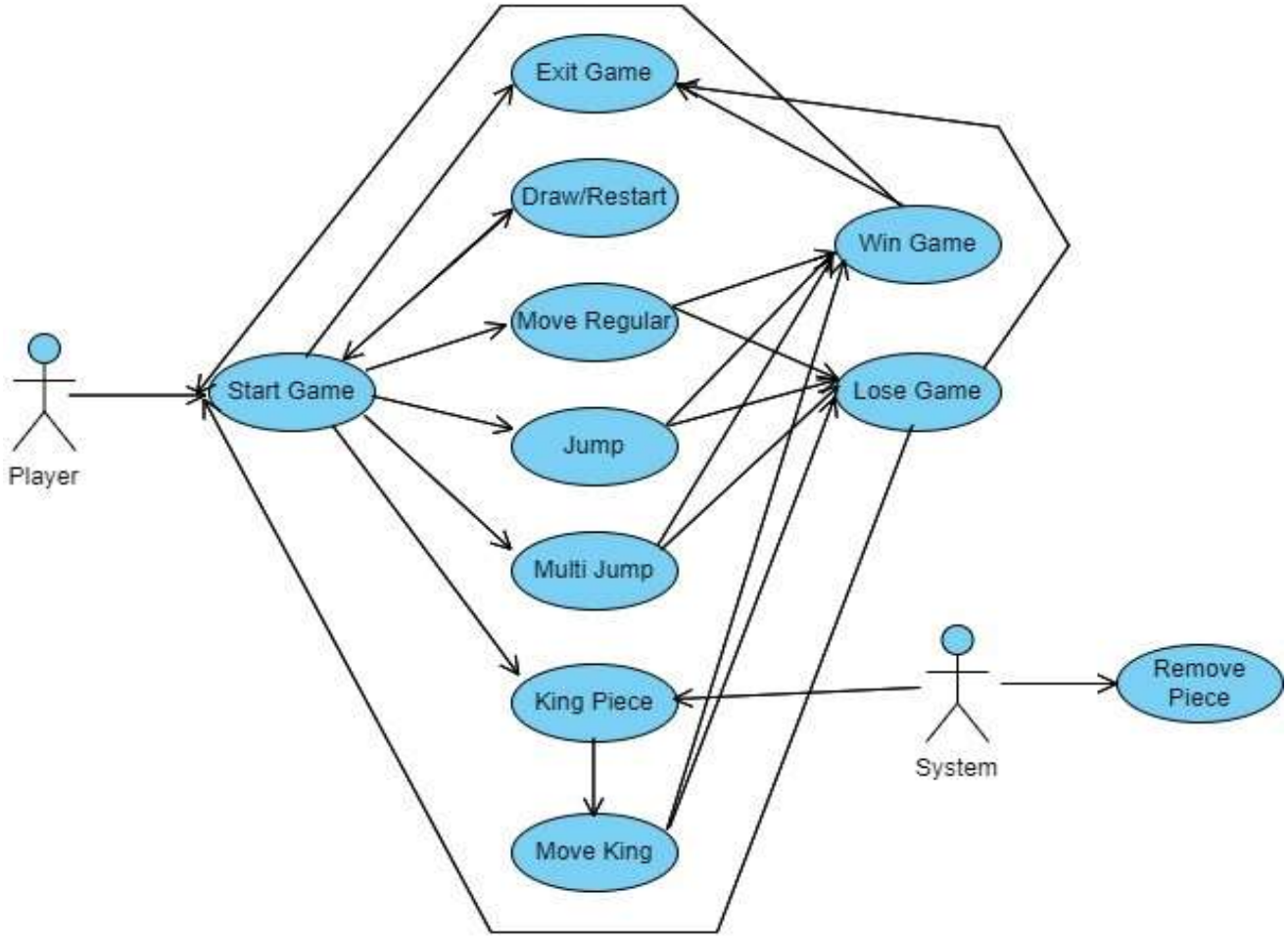
System Design is the process of defining the components, modules, interfaces, and data for a system to satisfy specified requirements using charts and diagrams

System Design includes diagrams for the software development that helps excessively to develop a proper planned software.

## 3.1 Use Case Diagram

A Use Case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well.

The figure below is a use case diagram representing the main activities inside our game that a player can perform.




**[Figure 2: Use Case of all main components in the game]**

## 3.2 User Interface Design

The user interface (UI) for the Checkers 3D Game will be designed to be simple, intuitive, and visually appealing. The UI will be designed with a minimalist style, using shades of blue and white color scheme to make it more soothing to the player's eye. The UI will be designed to work well on a variety of devices and screen sizes, with responsive design for different resolutions.

The game will consist of several scenes: a main menu scene, a scene for local multiplayer play, a scene for multiplayer play with UI for the chat feature. The main menu scene will allow the player to choose between the local and multiplayer modes, as well as let them enter a username.

The local play scene will display the checkers board, and the player will be able to move their pieces by dragging them with the mouse on a single shared screen. The multiplayer play scene will be similar to the local play scene, but the two players will be able to make moves simultaneously from different devices. The chat feature UI will allow the players to communicate with each other during gameplay.

The main menu scene will have buttons for local multiplayer play and Host-Connect buttons, Host creates a server instance hence hosts a game. Connect joins an established server instance by a host. The multiplayer play scene will have a simple interface with the checkers board displayed in the center of the screen, and a chat lobby right beside the board to make the game more interactive. The multiplayer play scene will be similar to the local multiplayer.

### 3.2.1 Chat Feature

The chat feature in the game allows players to communicate with each other in real-time while playing the game. This feature can enhance the overall gaming experience by enabling players to coordinate their moves, give each other feedback, and share their thoughts and ideas.

When a player wants to send a message, they can simply type it into the chat box located on the game interface. Once the message is entered, the player can hit the "send" button, and the message will be transmitted to the other player currently connected to the game.

Overall, the chat feature in the game is a valuable tool for promoting player engagement and communication. Whether players are working together to achieve a common goal or simply enjoying each other's company, the chat feature allows them to stay connected and engaged throughout the gaming experience.

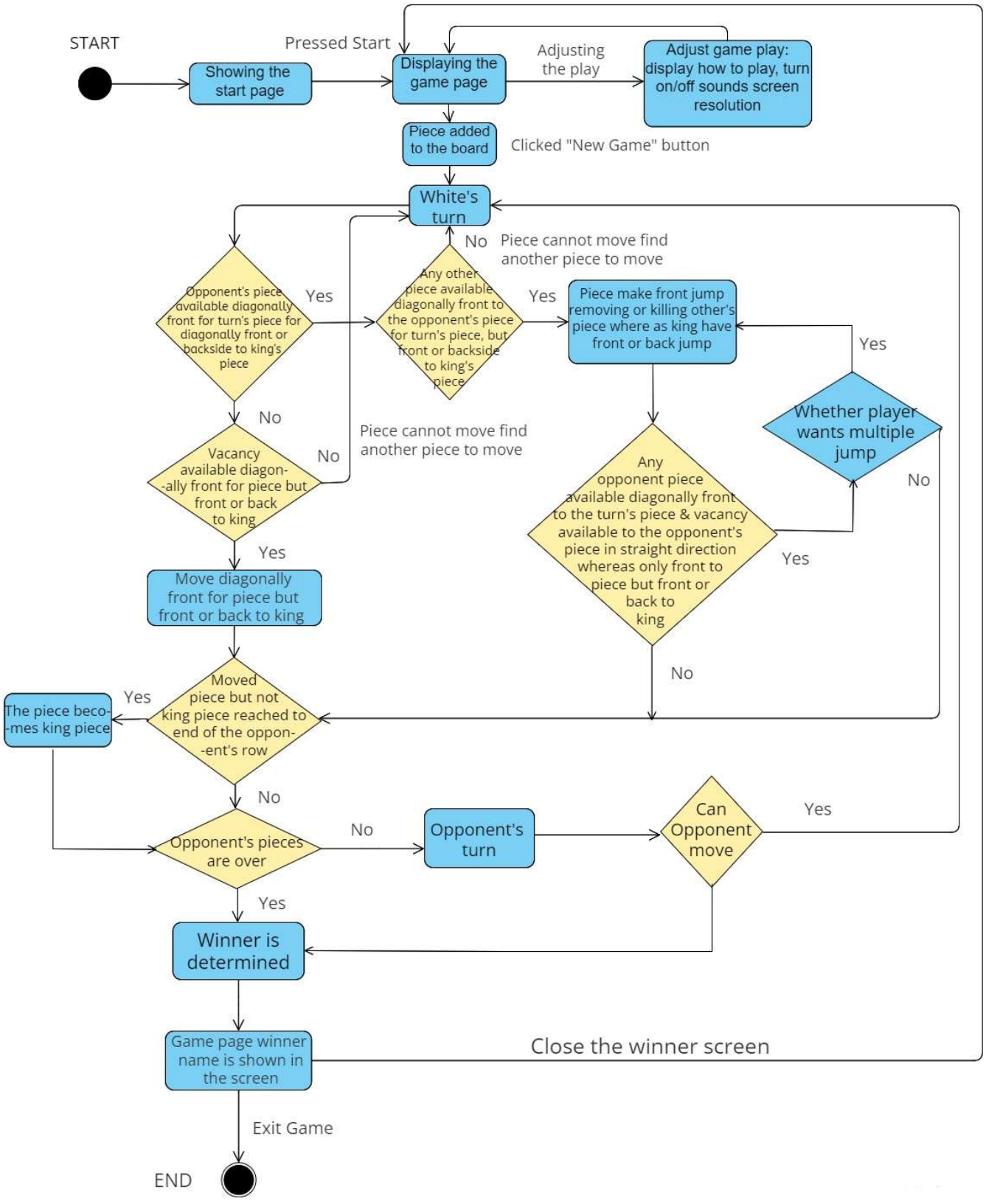
## 3.3 State Diagram

A State Diagram, also known as a state machine diagram or state-chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language UML.

In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity.

In our case, we have shown the functionalities this game consists along with most of the possibilities a player can run into while playing the game.

(Figure on next page)




**[Figure 3: State Diagram of the game]**

# CHAPTER 4: SYSTEM ARCHITECTURE

## 4.1 Overview:

System architecture for "Checkers 3D" involves the overall design and structure of the game, including its software and hardware components. It outlines how these components work together to create the game, and how they interact with each other.

## 4.2 Design Principles:

The design principles for the system architecture of "Checkers 3D" include both modularity and flexibility. The architecture is designed to be modular, so that different components can be easily added or removed without affecting the rest of the system. It is also designed to be flexible, so that it can be adapted to different hardware and software configurations.

## 4.3 Software Architecture:

The software architecture for "Checkers 3D" is based on a client-server model. The game client is responsible for rendering the game graphics and handling user input, while the game server manages game logic and state. The activation of portions; either of server or client scripts solely depends on which player hosted the game and which one connected a hosted game. The game client and server communicate with each other over the network using a protocol that is designed specifically for the game.

## 4.4 Client Architecture:

The client architecture is based on a three-tier architecture. The presentation layer handles the rendering of the game graphics and user input. The application layer is responsible for managing the game state and communicating with the server. The data layer stores the game data, such as player name and gameplay temporarily in order to reflect turns and determine victory/draw.

## 4.5 Server Architecture:

The server architecture is based on a three-tier architecture. The presentation layer handles communication with the client, and receives requests for game actions. The application layer manages game logic, such as player moves and game rules. The data layer stores the game state, including board positions and player information.

## 4.6 Hardware Architecture:

The hardware architecture for "Checkers 3D" is designed to be flexible. It is designed to configure the game logic in respect and accordance of the system it has been installed and running on, making flexible for all systems.

## 4.7 System Integration:

To ensure smooth system integration, it is important to establish clear interfaces and protocols between these modules. For example, the game engine must be able to communicate with the graphics and animation module to render the game board and pieces

Another important consideration for system integration is testing. Thorough testing must be carried out to ensure that each module is functioning correctly and that they are working together seamlessly. This involves both unit testing of individual modules and integration testing to ensure that the different modules are interacting correctly with one another.

To conclude, successful system integration is essential for the smooth functioning of Checkers 3D and for delivering a high-quality user experience. It requires careful planning, clear communication, and meticulous testing to ensure that all components are integrated effectively and working together seamlessly.

Overall, the system architecture for "Checkers 3D" is designed to be modular, scalable, and flexible, with a client-server architecture that provides seamless communication. And the hardware architecture being the key in developing a platform agnostic.

## 4.8 UML Diagrams

The Unified Modeling Language (UML) is a modeling language to provide a standard way to visualize the design of a system. There are two types of UML diagrams – Behavior and Structural Diagrams.

### 4.8.1 Behavior Diagrams

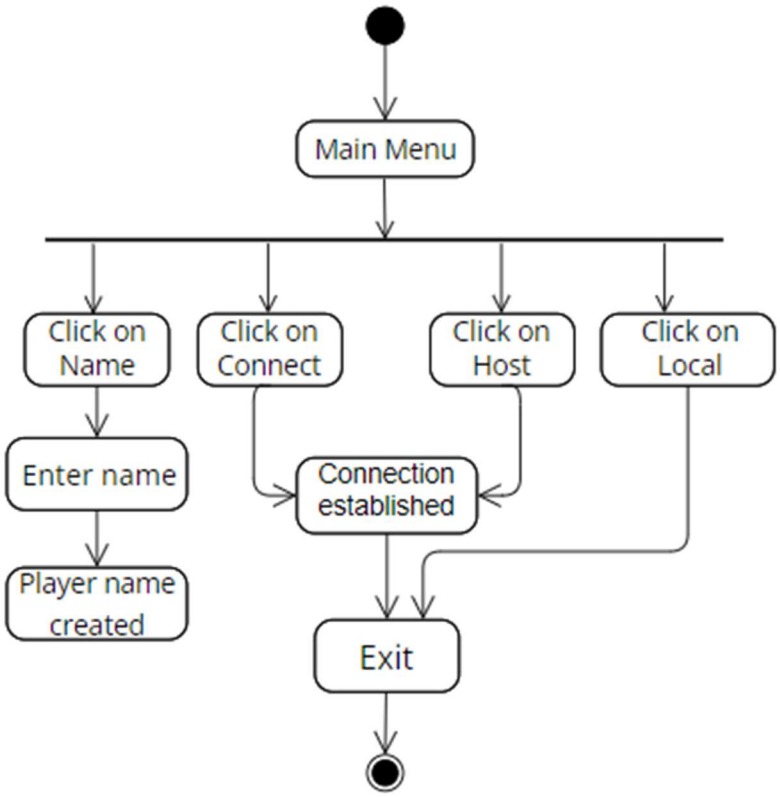
Behavior diagrams emphasize what should happen in the system or its behavior and are used to describe the functionality of software systems. Activity Diagrams describes the step-by-step activities in a system.

#### 4.8.1.1 Main Menu Activity

The Main Menu Activity diagram here shows when player enters the game and is directed to main menu, they can choose from multiple options, they are: 1. Enter their name, 2. Host a game, 3. Connect an already hosted game and 4. Select Local multiplayer mode.

All four options are displayed below in the form of a diagram:

(Figure on next page)




**[Figure 4: Main Menu activity diagram of the game]**

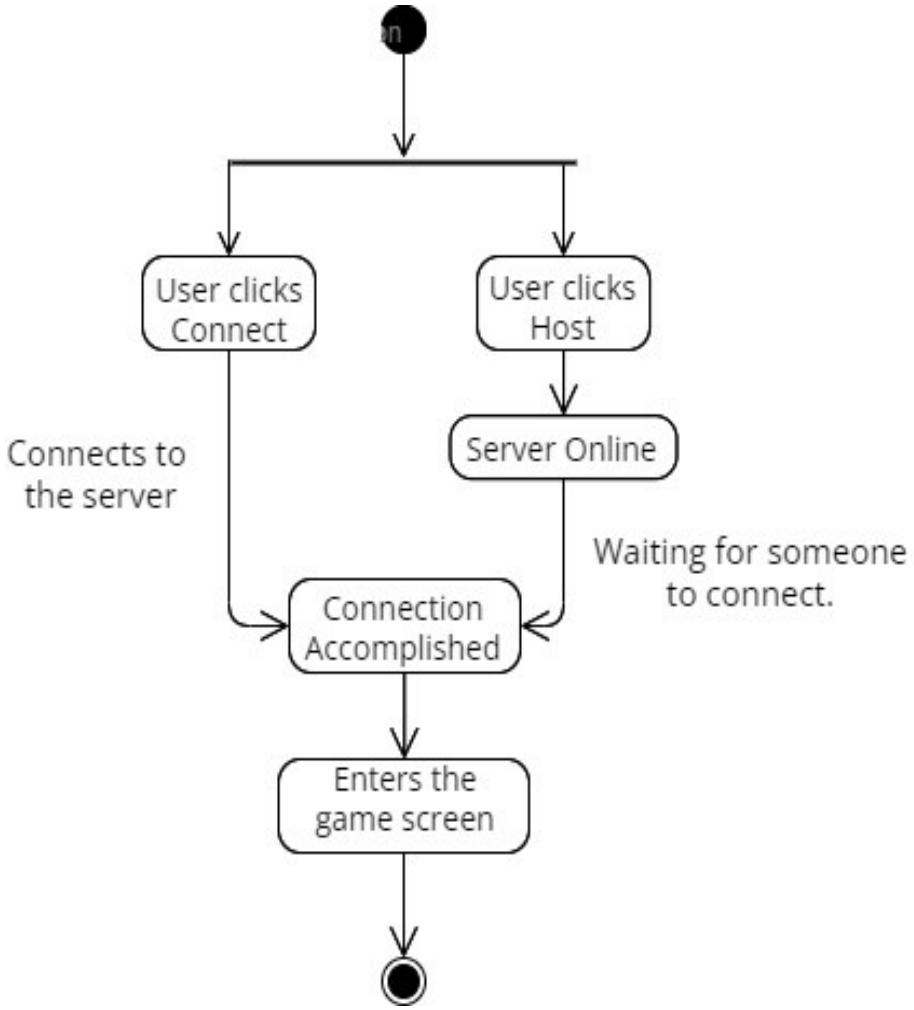
### 4.8.1.2 Multiplayer Connection Activity

Within the game, there are generally two types of nodes:

- **Server**: Hosts the game and respond to requests
- **Client**: Connects to the server and sends requests

Multiplayer connection process of our game is shown in the form of a diagram below:

(Figure on next page)




**[Figure 5: Multiplayer Connection activity diagram of the game]**

## 4.9 Structure Diagrams

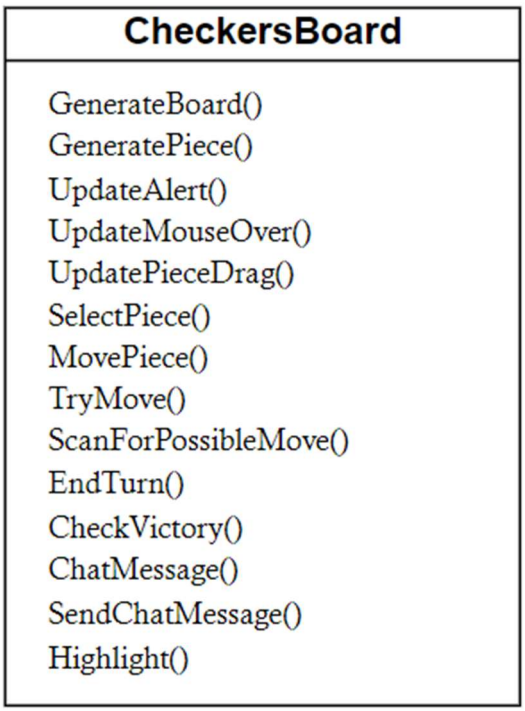
Structure diagrams shows all the functions that should be there in the system. They are used for defining the architecture of software systems.

### 4.9.1 Class Diagrams

Class Diagrams in Unified Modeling Language describes the structure of a system by showing the system's classes, properties or methods.

Below are the classes developed for this game:

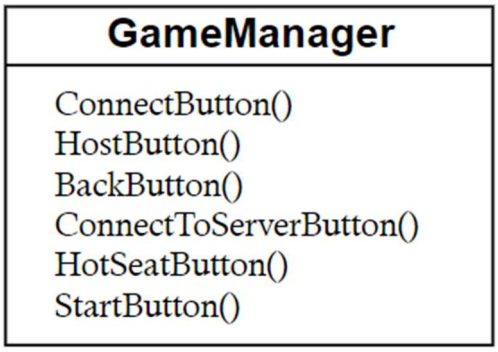
- **CheckersBoard:**




- **GenerateBoard()** – Generates the game board with all the necessary cells and initial piece placements.
- **GeneratePiece()** – Generates a new game piece and assigns it to a specific cell on the board.
- **UpdateAlert()** – Updates the alert or message displayed on the game interface.
- **UpdateMouseOver()** – Handles the updating of visuals and interactions when the mouse hovers over a specific cell or piece on the board.
- **UpdatePieceDrag()** – Handles the logic for dragging and moving a game piece on the board.
- **SelectPiece()** – Handles the selection of a game piece by the player for further actions.
- **MovePiece()** – Moves selected game piece to a specified target cell on the board.
- **TryMove()** – Checks if a move is valid and attempts to execute it.
- **ScanForPossibleMove()** – Scans the board to identify and highlight all possible moves for the current player.

- **EndTurn()** – Marks the end of the current player's turn and prepares for the next player's turn.
- **CheckVictory()** – Checks for victory conditions, such as capturing all opponent's pieces or blocking their moves.
- **ChatMessage()** – Handles the reception and display of chat messages during multiplayer gameplay.
- **SendChatMessage()** – Sends a chat message to the opponent player in a multiplayer game.
- **Highlight()** – Highlights a specific cell or piece on the board to draw attention or indicate certain conditions or events.

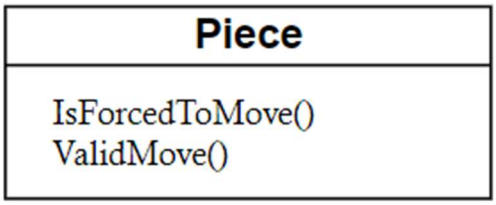
## • GameManager:




- **ConnectButton()** – Handles the logic for the connect button, allowing the player to establish a connection to a server.
- **HostButton()** – Handles the logic for the host button, enabling the player to create a server and wait for incoming connections.
- **BackButton()** – Manages the functionality of the back button, allowing the player to navigate to a previous screen or menu.
- **ConnectToServerButton()** – Handles the logic for the button that initiates the connection to a specific server.
- **HotseatButton()** – Manages the functionality of the button that enables the local multiplayer mode in the game, allowing multiple players to take turns playing on the same device or system.

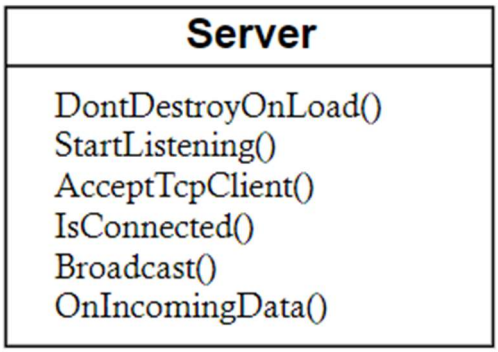
- **StartGame()** – Initiates the start of the game, transitioning from the lobby or setup phase to the actual gameplay phase.

- **Piece:**



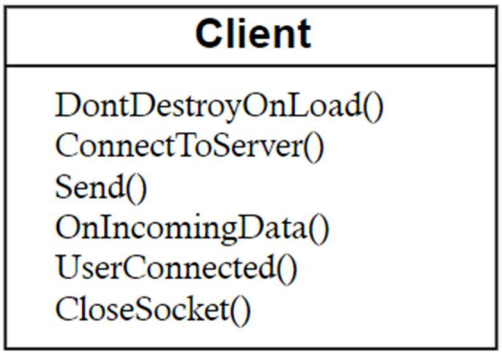

- **IsForceToMove()** – Checks if the piece is required to make a forced move, such as capturing an opponent's piece.
- **ValidMove()** – Determines if a move is valid for the piece based on the game rules and the current state of the game board.

- **Server:**




- **DontDestoyOnLoad()** – Ensures that the server object persists across scene changes in the game.
- **StartListening()** – Starts listening for incoming TCP connections from clients.
- **AcceptTcpClient()** – Accepts an incoming TCP client connection request.
- **IsConnented()** – Checks if the server is currently connected to any clients.
- **Broadcast()** – Sends a message or data to a connected client.
- **OnIncomingData()** – Handles the incoming data received from client.

- **Client:**




- **DontDestoryOnLoad()** – Ensures that the client object persists across scene changes in the game.
- **ConnectToServer()** – Initiates a connection to the server using TCP.
- **Send()** – Sends a message or data to the connected server.
- **OnIncomingData()** – Handles the incoming data received from the server.
- **UserConnected()** – Notifies the client when it connects to the server.
- **CloseSocket()** – Closes the client socket connection to the server.

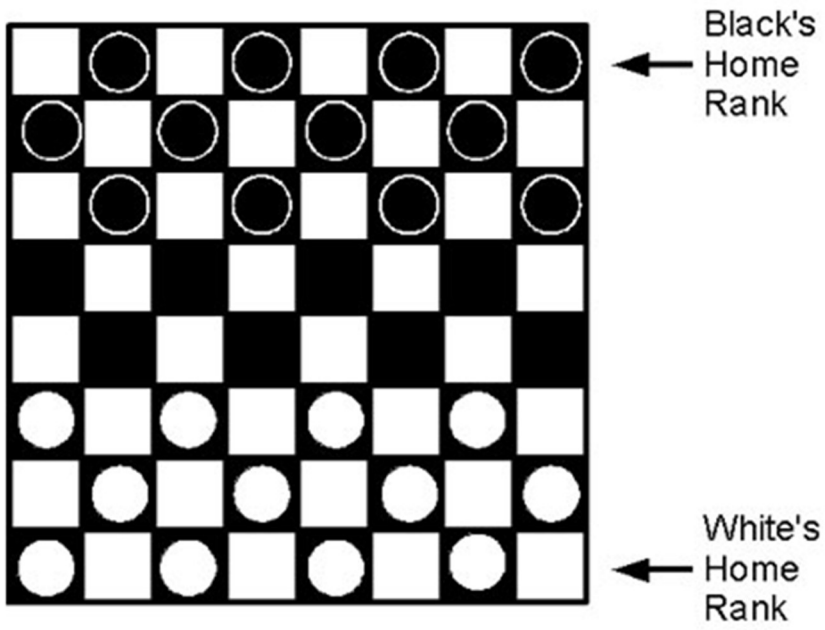
# CHAPTER 5: PLANNING AND MAPPING

Before diving into the implementation and working of this adaptation of Checkers, let's take a look into the rules and fundamentals of how the actual board game use to be.

## 5.1 GAME RULES

**Aim:** To capture all your opponent's pieces or block them so they cannot move.

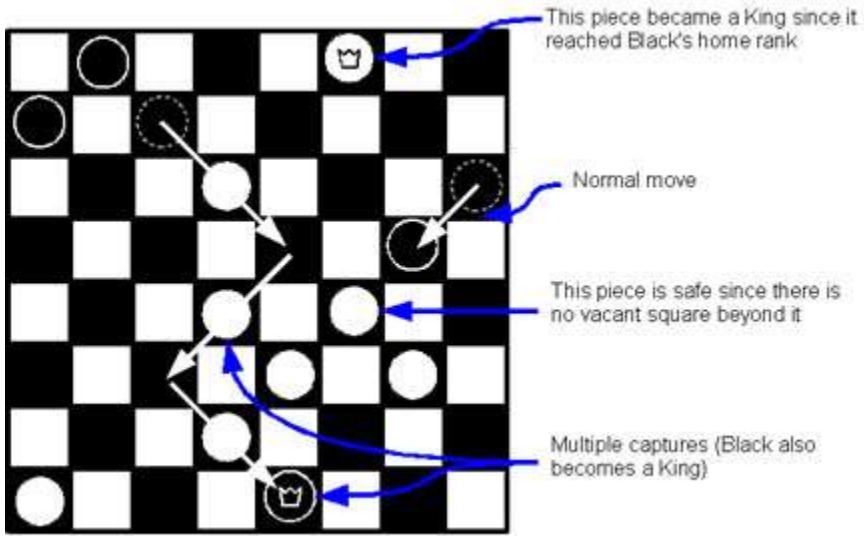
1. The checkers board is set up as shown. Note that the game is only played on the black squares, with the black square in the lower left corner. One player holds a black counter in one fist and a white counter in the other fist. The other player chooses a fist. The player with the black counter goes first.




**[Figure 6: Image showcasing the initial positions in the game]**

2. Players take alternate turns to move one piece diagonally to a vacant square, starting with Black. This means that only pieces in the third rank can move on the first turn. Pieces can only move forward (on a diagonal). Once a piece has moved and you take your finger off it, it cannot be moved again until your next turn.

3. Players can capture an opponent's piece by jumping over it in a straight line to a vacant square. Captured pieces are removed from the board. Capturing is not optional - you must capture an opponent's piece if there is an opportunity to do so. If you fail to do so, your piece can be removed from the board. If there are two or more pieces you could capture, you can choose which one you will take. You can also capture multiple pieces as long as there is a vacant square between each piece, even if they are not in a straight line (see diagram below). Note: Good players can use a forced capture in a strategic move to make their opponent move a piece for their own advantage. You may lose one piece but can position their pieces for a multiple capture.



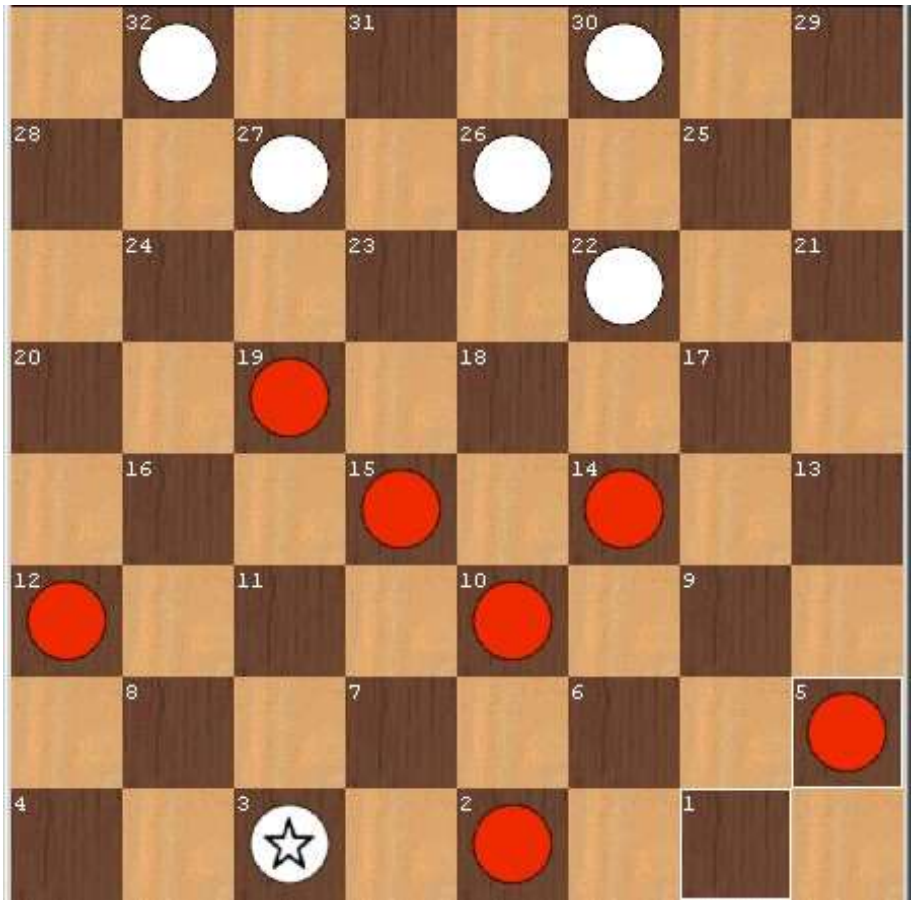

**[Figure 7: Image explaining the game dynamics]**

4. If a player moves one of their pieces to the home rank of the opposing player, that piece becomes a King. Some counters are designed with a crown on one side, so that the piece can be flipped over to show it is a King. Other counters are made to fit on top of each other so that a King is shown by two pieces joined together. A King can move one square forward or backwards in a diagonal line.

5. The game finishes when one player captures all their opponent's pieces or blocks their opponent so they cannot move any of their pieces. Unfortunately, games quite often end in a draw when neither player can block or capture all of their opponent's pieces.

## 5.2 GAME LOGIC ACCORDING TO RULES

Now let's see at how the original rules are going to look like in our adaptation "CHECKERS 3D".




**[Figure 8: Image shows numbering of the blocks inside a checkers board]**

To be able to insert a logic in the game, the board must be divided into certain blocks which will help to locate a player and make it move from one dark spot to another dark spot in the game.

In a game of Checkers, you can only move a piece in a certain direction and so setting a logic that only allows the player to move to a particular position is very essential. As mentioned in the rules above that:

"A move consists of moving a piece diagonally to an adjacent unoccupied square."

The game will also have a functionality where only the piece which is in the position from where it can remove the opponent's pieces, will be able to move, the player will have to figure out which piece is moveable by looking at all the defeatable pieces and move accordingly.

As soon as the player jumps over another opponent's piece, the crossed over piece will disappear (will be removed from the game logic running in the background).

**King Piece:**

According to the rules, when a piece of one player reaches the final/last row of another player's arena then that piece becomes a king. That piece is made king by placing another piece on top of it, In our case when a piece reaches the back row of an opponent and becomes a king, we flip it to the side which has a crown drawn on top hence making it more recognizable.

The king logic will be triggered for such piece as soon as it touches the respective last rows of their opposite sides.

## 5.3 Additional Rules

To make the game more challenging and interesting, we have added several additional rules to Checkers 3D. These include:

- King Jump: When a piece reaches the other side of the board, it is crowned as a king. Kings have the ability to move and capture in any diagonal direction.

- Double Jump: If a player captures an opponent's piece and lands in a position where they can capture another piece in the same move, they can continue to jump and capture additional pieces in a sequence.

- Forced Move: If a player has a piece that can capture an opponent's piece, they must make the move, even if it puts them in a disadvantageous position.

- By adding these additional rules and features, Checkers 3D provides a fresh and exciting take on the classic game of Checkers.

◀▶

# CHAPTER-6: CODING STANDARDS, TOOLS EXPLANATION, SYSTEM FLOW WITH SCREENSHOTS

## 6.1 Coding Standards

When developing the Checkers 3D game, a set of coding standards were followed to ensure that the code is consistent, readable, and maintainable. hese standards include:

- **Naming Conventions:**

In order to ensure consistency and readability, naming conventions were strictly followed throughout the codebase. All variables, functions, classes, and other objects were given meaningful and descriptive names. Variables were named using camelCase notation, while classes and objects used PascalCase notation. The names chosen for these elements were chosen to reflect their purpose and functionality within the code.

- **Code Formatting:**

Consistent code formatting was maintained throughout the codebase to ensure readability and maintainability. A standard indentation of 4 spaces was used throughout the codebase. In addition, each block of code was separated by a blank line to clearly demarcate different sections of the code.

- **Comments:**

Comments were used throughout the codebase to explain the purpose and functionality of different blocks of code. Comments were written in a clear and concise manner to ensure that they could be easily understood by others.

- **Error Handling:**

The codebase was designed with robust error handling mechanisms to ensure that the game remained stable and responsive even in the face of unexpected errors. Wherever possible, errors were caught and handled gracefully to minimize disruption to the user. Error messages were designed to be clear, indicating the nature of the error and any steps that the user could take to resolve the issue.

In summary, the coding standards followed in Checkers 3D were designed to ensure consistency, readability, and maintainability of the codebase. They were intended to make it easier for others to understand and work with the code and to ensure that the game remained stable and error-free.

## 6.2 Tools Explanation

- **Unity Game Engine:**

Unity is a powerful cross-platform game engine that allows developers to create games for various platforms like PC, mobile, consoles, and web. It has a wide range of features like 3D modeling, physics, animation, scripting, and audio that help developers to create immersive games. Unity supports a variety of programming languages including C#, JavaScript, and it also provides a wide range of plugins and assets that can be used to enhance the development process.

For the game Checkers 3D, Unity was utilized to create the game's 3D environment, user interface, gameplay mechanics, and other features. I was able to leverage Unity's built-in tools to create and manipulate 3D models, design and implement game logic, manage scene transitions, and implement visual effects.

- **C# programming language:**

C# is a powerful and modern programming language that is widely used in game development. It is an object-oriented language that allows developers to write clean, organized, and efficient code. C# is an integral part of the Unity game engine and is the primary scripting language used to develop games in Unity.

For the game Checkers 3D, I used C# to create scripts that controlled the behavior of various game objects like the game board, pieces, and user interface. I was able to utilize C#'s object-oriented features to create reusable code, manage game state, and implement game logic.

- **Visual Studio IDE:**

Visual Studio is a popular integrated development environment (IDE) that provides a wide range of tools for developing software applications. It supports various programming languages including C# and offers advanced features like

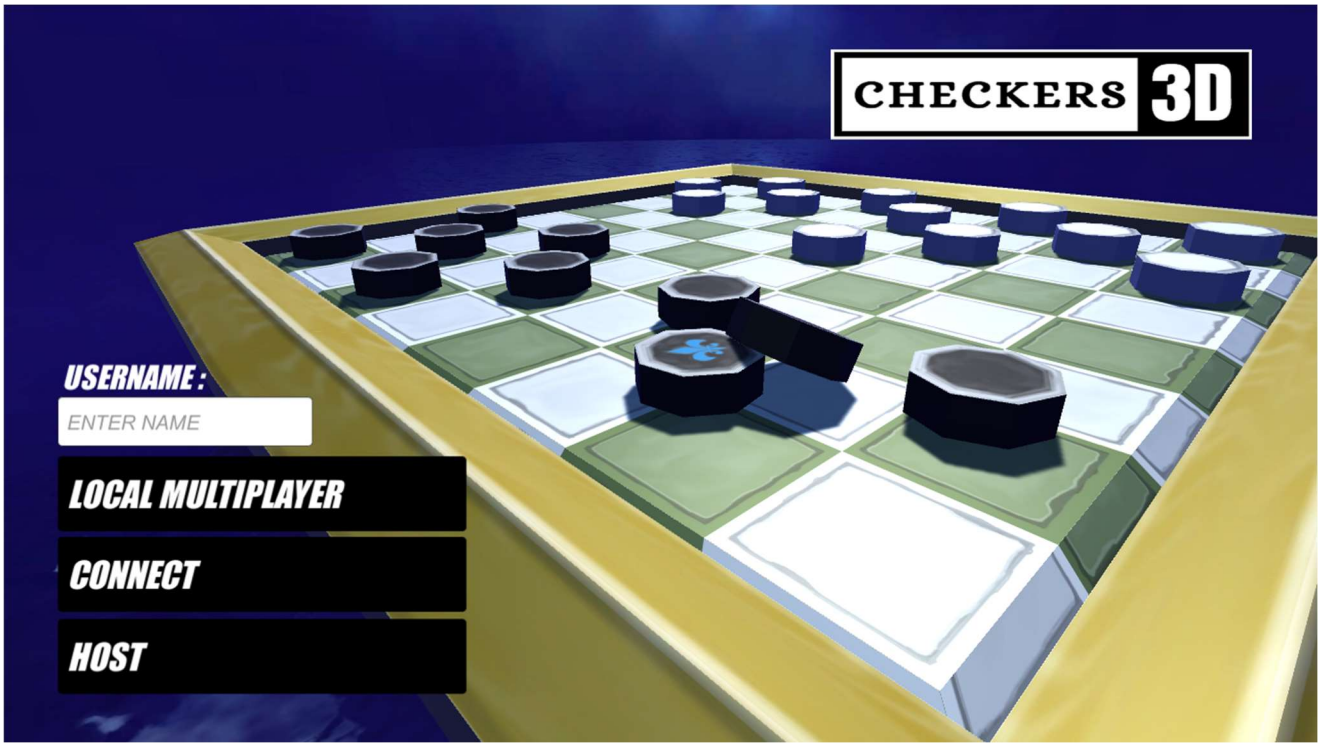
code highlighting, debugging, and code refactoring.

For the game Checkers 3D, Visual Studio was utilized as the primary IDE for writing and debugging C# code. I was able to take advantage of Visual Studio's powerful debugging tools to find and fix errors in the code, and was also able to use its code analysis and refactoring features to improve code's performance and readability.

## 6.3 System Flow

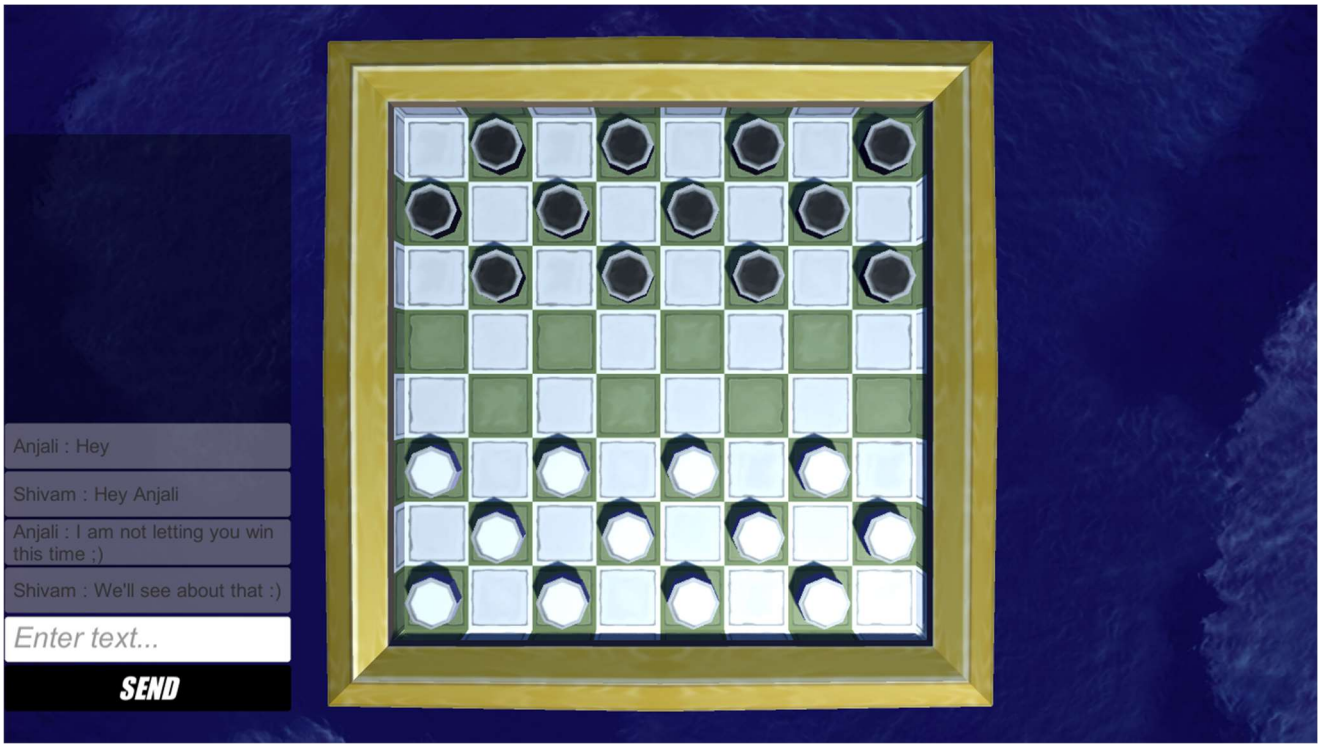
- **Game Flow:**

1. Launch: When the user launches the game, the main menu is displayed.
2. Main menu: The main menu allows the user to select from different options, including selecting a custom username, entering into local multiplayer mode, hosting a server or creating a game, connecting to a created server(game).
3. Board setup: Once the user selects the game mode, the board is set up and the pieces are placed in their starting positions. The player who goes first is also determined at this stage.
4. Game play: The game proceeds with players taking turns moving their pieces across the board, following the rules of the game. The game keeps track of the current player's turn and updates the board accordingly.
5. Move validation: Whenever a player makes a move, the game checks whether the move is valid according to the rules of the game. If the move is invalid, the piece doesn’t land and the player is prompted to make a valid move.
6. Capture validation: Whenever a player makes a capture move, the game checks whether the move is valid and captures the opponent's piece. If it cannot, means the move is invalid and the player is prompted to make a valid move.
7. End game: The game ends when one player successfully captures all of their opponent's pieces or when the players agree to a draw. The game displays a message indicating the winner and returns to the main menu.




**[Screenshot: Main Menu]**

The following screenshot shows some of the key aspects of the game, including the game environment, game board, and game pieces:




**[Screenshot: In-Game UI]**

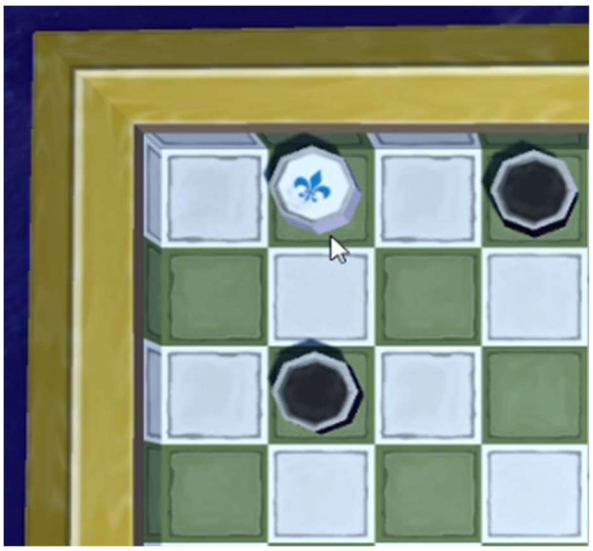
- **Image Description:**

In the in-game image of Checkers 3D, we can see the classic checkers board with 64 squares arranged in an 8x8 grid. The board is divided into two sections with a light and dark-colored square pattern. At the start of the game, all the pieces are in their initial positions. We can see that the pieces are black and white and each player has 12 pieces placed on their respective first three rows.

On the left side of the screen, we can see a chat box where two players are having a conversation before starting the game. The chat box allows players to communicate with each other during the game, share thoughts, and discuss strategies. It also adds an element of social interaction to the game, making it more engaging for players.

The chat box is designed with a minimalist interface, which blends well with the overall game design. It has a transparent, white background with black text, making it easy to read. The chat box also displays the usernames of the players along with their messages, making it easy to identify who is saying what.

Overall, this image sets the stage for an exciting game of Checkers 3D, where two players are about to engage in a strategic battle. The chat box adds a personal touch to the game and allows players to interact with each other, making the game more enjoyable and immersive.



- **Image description:**

The addition of the crown to the king piece in Checkers 3D not only enhances the visual appeal of the game, but also adds a functional element that makes it easier for players to identify and strategize with this vital game piece.

◀▶

# CHAPTER 7: TESTING AND QUALITY ASSURANCE

## 7.1 Overview

Testing and quality assurance play a crucial role in the development of Checkers 3D. A rigorous testing process ensures that the game functions as intended and meets the requirements specified in the project scope. In this section, we will discuss the testing and quality assurance processes that were employed during the development of Checkers 3D.

## 7.2 Testing Methods

Several testing methods were used to ensure that Checkers 3D functioned correctly. These methods included:

- **Unit Testing:**

Unit testing is a type of testing that is performed on individual units or components of the software. It is a testing technique that is used to validate that each unit of the software performs as expected and meets its design specifications. Unit testing is an essential part of the software development process and is used to identify defects and bugs early in the development cycle.

In the context of Checkers 3D, unit testing was performed on individual code components to ensure that they functioned as intended. For example, each class in the game, such as the CheckersBoard, GameManager, Piece, Client, Server was tested independently to ensure that it worked correctly.

- **Integration Testing:**

Integration testing is a critical aspect of software testing and is essential in ensuring that various code components work seamlessly together. In the case of Checkers 3D, integration testing was carried out to ensure that different game components such as the game board, game pieces, and game logic worked together without any issues.

The integration testing process started with identifying the different modules of the game and ensuring that each module could communicate with each other efficiently. This involved testing the communication channels between the game board, the game pieces, the user interface, and the game logic. The objective of this testing phase was to identify and resolve any issues that could arise due to different modules not communicating with each other properly.

This included testing the movement of game pieces on the game board, the capture of opposing game pieces, and the display of the game board on the user interface.

In conclusion, integration testing is a crucial part of software testing, and it played a critical role in ensuring that different modules of Checkers 3D worked seamlessly together. By testing the communication between different game components, I was able to identify and resolve any issues that could arise during gameplay. This testing phase helped to ensure that Checkers 3D was a high-quality and stable game that could provide an enjoyable gaming experience to users.

- **System Testing:**

System testing is a critical phase in software development, and it was an important part of the testing and quality assurance process for Checkers 3D. The purpose of system testing is to evaluate the game as a whole and ensure that it meets all functional - non-functional requirements, including any specified criteria.

For Checkers 3D, system testing was performed after the individual components were tested using integration testing. The system testing phase involved testing the entire game, including all of its components, features, and functionality, to verify that it met all of the project requirements and design specifications.

- **Regression Testing:**

Regression testing is a type of software testing that focuses on verifying that previously developed and tested functionalities of the software still work correctly after modifications or enhancements. It was performed to ensure that any changes made to the game did not cause any unintended side effects.

## 7.3 Quality Assurance

Quality assurance was a critical component of the development process for Checkers 3D. To ensure high-quality standards, the following techniques were employed:

- Code Reviews: Code reviews were conducted to ensure that code was well-organized, efficient, and free from bugs.
- Debugging: Debugging was performed to identify and eliminate any errors or bugs in the code.

## 7.4 Results

The testing and quality assurance processes employed during the development of Checkers 3D resulted in a game that meets the highest standards of quality. The game has been tested extensively and is free from any critical bugs or errors.

## 7.5 Future Improvements

While Checkers 3D meets the requirements specified in the project scope, there is always room for improvement. As part of ongoing quality assurance, will continue to identify areas for improvement. Future updates to the game may include additional features, bug fixes, and other improvements to enhance the overall gaming experience.

By employing a testing and quality assurance process, Checkers 3D has been developed to the highest standards of quality, providing a seamless and enjoyable gaming experience for players.

# CHAPTER 8: CONCLUSION AND FUTURE WORKS

## 8.1 Conclusion

In conclusion, the development of Checkers 3D has been a rewarding experience. The game successfully brings back the nostalgic gameplay of the traditional board game, Checkers or Draughts, in a digital format. By staying true to the original rules and providing an immersive 3D environment, Checkers 3D offers an enjoyable and engaging gaming experience for players of all ages.

Throughout this documentation, various aspects of the game are covered, including its project description, problem summary, scope, objectives, system specifications, design, planning, and implementation. We have discussed the testing and quality assurance procedures that were employed to ensure a robust and error-free game. The different ways of testing, such as unit testing, integration testing, and system testing, played a crucial role in validating the functionality and performance of Checkers 3D.

## 8.2 Future Works

While Checkers 3D has achieved its current objectives, there are several possibilities for future improvements and expansions to further enhance the game and its overall experience. Some potential areas for future work include:

- **AI Opponent:** Implementing an artificial intelligence (AI) opponent to provide a single-player mode. This would allow players to challenge the computer and test their skills against intelligent gameplay.
- **Additional Game Modes:** Introducing new game modes, such as a time-limited mode, tournament mode, or alternative board layouts, to offer players a variety of gameplay options and increase replay value.
- **Online Multiplayer:** Expanding the multiplayer functionality to support online gameplay, enabling players to compete against opponents from around the

world. This would require implementing network features, matchmaking systems, and secure communication protocols.

- **Enhanced Graphics and Visual Effects:** Improving the visual elements of the game, including enhanced graphics, animations, and special effects, to create a more immersive and visually appealing experience for players.
- **Customization Options**: Introducing customization features that allow players to personalize their game experience by selecting different board themes, piece designs, or even creating their own custom game boards.
- **Mobile Version:** Adapting Checkers 3D for mobile platforms, such as smartphones and tablets, to reach a wider audience and provide on-the-go gaming experiences.
- **Community Features:** Implementing social features, such as leaderboards, achievements, and player profiles, to foster a sense of competition and community among players.

By pursuing these future works, Checkers 3D can continue to evolve and attract players more, offering an even more enjoyable and engaging gaming experience.

In conclusion, Checkers 3D has successfully brought the classic game of Checkers to a digital platform while maintaining its authenticity. With future works and continuous improvements, the game can continue to evolve and provide players with new and exciting features, ensuring its longevity and appeal in the ever-growing gaming landscape.

# CHAPTER 9: REFERENCES

The development of "Checkers 3D" drew upon various resources and references to ensure the accuracy, quality, and adherence to good standards. The following references were consulted during the creation of this game:

**[1] Unity Scripting API Documentation:** The comprehensive documentation for the Unity Scripting API, which includes information about classes, methods, properties, and more.

Studied from: https://docs.unity3d.com/ScriptReference/

**[2] C# Programming Guide:** A comprehensive guide to C# programming provided by Microsoft, covering topics such as syntax, data types, control structures, and object-oriented programming.

Studied from: https://docs.microsoft.com/en-us/dotnet/csharp/

**[3] Checkers Game Rules:** The official rules and strategies for playing checkers.

Retrieved from: https://www.family-games-treasurehouse.com/checkers.html

**[4] Unity Multiplayer Documentation:** The official documentation on Unity's multiplayer functionality, including networking concepts, synchronization, and implementation.

Studied from: https://docs.unity3d.com/Manual/UNet.html

**[5] Socket Programming in C#:** A tutorial on socket programming in C#, explaining concepts such as TCP/IP communication, server-client model, and socket operations.

Studied from: https://www.c-sharpcorner.com/article/socket-programming-in-C-Sharp/

**[6] Game Development Patterns:** A collection of design patterns commonly used in game development.

Studied from: https://gameprogrammingpatterns.com/

**[7] Object-Oriented Design Principles:** An overview of object-oriented design principles, such as encapsulation, inheritance, polymorphism, and SOLID principles.

Studied from: https://www.geeksforgeeks.org/solid-principle-in-programming-understand-with-real-life-examples/

**[8] Regression Testing:** An explanation of regression testing.

Referred from: https://www.guru99.com/regression-testing.html

**[9] Integration Testing:** A comprehensive guide to integration testing.

Referred from: https://www.softwaretestinghelp.com/integration-testing-guide/

**[10] System Testing:** An overview of system testing in software development.

Referred from: https://www.geeksforgeeks.org/system-testing/

**[11] Unit Testing:** A comprehensive guide to unit testing.

Referred from:
https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm

**[12] UML Diagrams:** An overview of Unified Modeling Language (UML) diagrams, including class diagrams, activity diagrams, and use case diagrams.

Referred from: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

**[13] Web application used to create the diagrams**:

https://online.visual-paradigm.com

- **Diagrams/Figures:**

**[14] Figure 1:** "Image of two players playing checkers board game"

Retrieved from: https://www.istockphoto.com/photos/checkers-game

**[15] Figure 2:** "Use Case of all main components in the game"

Made using: https://online.visual-paradigm.com

**[16] Figure 3:** "State Diagram of the game"

Made using: https://online.visual-paradigm.com

**[17] Figure 4:** "Main Menu activity diagram of the game"

Made using: https://online.visual-paradigm.com

**[18] Figure 5:** "Multiplayer Connection activity diagram of the game"

Made using: https://online.visual-paradigm.com

**[19] Figure 4:** Class Diagrams:

Made using: https://online.visual-paradigm.com

**[20] Figure 6:** "Image showcasing the initial positions in the game"

Retrieved from: https://www.family-games-treasurehouse.com/checkers.html

**[21] Figure 7:** "Image explaining the game dynamics"

Retrieved from: https://www.family-games-treasurehouse.com/checkers.html

**[22] Figure 8:** "Image shows numbering of the blocks inside a checkers board"

Retrieved from: http://www.fierz.ch/checkerboard.php