Parshvanath Charitable Trust's

A. P. SHAH INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

A MINI PROJECT REPORT

On

TWO PLAYER CHESS PYGAME

Submitted in partial fulfillment of the requirement of
University of Mumbai for the Course

In

Computer Engineering (IV SEM)

Submitted By
Aniket Asawale
Shreyas Bhalekar

Subject Incharge
Dr. Rahul Ambekar

Parshvanath Charitable Trust's

A. P. SHAH INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

CERTIFICATE

This is to certify that the requirements for the project report entitled '**TWO PLAYER CHESS PYGAME**' have been successfully completed by the following students:

Name	**Moodle Id**
Aniket Asawale	22102140
Shreyas Bhalekar	22102168

In partial fulfilment of the course Skill Base Lab Course: Python Programming (CSL405) in Sem: IV of Mumbai University in the Department of Computer Engineering during academic year 2022-2023.

Sub-in-Charge Dr. Rahul Ambekar

Parshvanath Charitable Trust's

A. P. SHAH INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

PROJECT APPROVAL

The project entitled '**TWO PLAYER CHESS PYGAME**' by Aniket Asawale**,** Shreyas Bhalekar are approved for the course of Skill Base Lab Course: Python Programming (CSL405) in Sem: IV of Mumbai University in the Department of Computer Engineering.

Subject-in-Charge

Dr. Rahul Ambekar

Date: 27/3/24

Place: Thane

Table of Contents

Abstract

This project presents a Python implementation of a chess game using the Pygame library for graphical user interface (GUI) and the chess library for game logic. The objective of this project is to create an interactive and visually appealing platform for playing chess against an AI or another human player.

The implementation includes features such as piece movement, legal move validation, capturing opponent pieces, check and checkmate detection, pawn promotion, and castling. Players can interact with the game through mouse clicks to select and move pieces. Visual cues are provided to indicate available moves, check conditions, and possible pawn promotions.

The GUI is designed with a simple and intuitive layout, featuring a chessboard rendered using Pygame's drawing functions. Sound effects enhance the gaming experience, with different audio cues for piece movement and check events. Additionally, background music creates ambiance during gameplay.

The code structure follows object-oriented principles, with classes for managing game state, handling player input, and implementing chess rules. Modular functions handle various aspects of move validation, including pawn movement, piece capture, and check detection.

Overall, this project provides a comprehensive implementation of a chess game, combining the capabilities of Pygame for GUI development and the chess library for game logic, resulting in a functional and engaging gaming experience.

List of Figures

Problem Definition

The project aims to address the need for a robust and user-friendly platform for playing chess on a computer. Despite the popularity of chess as a strategic game, existing implementations often lack intuitive interfaces and robust game logic, hindering the gaming experience. Moreover, many available chess programs lack the flexibility to accommodate features such as customizable player options, sound effects, and graphical enhancements.

Existing implementations may also suffer from performance issues, particularly in handling move validation and AI opponent logic. This can result in sluggish gameplay and frustration for users, particularly those seeking a responsive and immersive gaming experience.

Furthermore, while there are libraries available for implementing chess logic, integrating them seamlessly with graphical user interfaces (GUIs) remains a challenge for many developers. The lack of comprehensive resources that combine both GUI and game logic components limits the accessibility and quality of chess game implementations.

Therefore, the project seeks to address these challenges by developing a Python-based chess game using the Pygame library for GUI development and the chess library for game logic. By combining these technologies, the project aims to create an intuitive, visually appealing, and performant platform for playing chess, catering to both casual players and enthusiasts alike.

Introduction

Chess stands as a timeless game of strategy, cherished by enthusiasts worldwide for its depth and complexity. In the digital age, computer-based implementations have brought the game to new audiences, offering convenience and accessibility. However, the quality of available chess programs varies widely, with many lacking the polish and functionality desired by players.

This project seeks to bridge the gap by creating a high-quality, user-friendly chess game using Python and leveraging the Pygame and chess libraries. The endeavor addresses several key shortcomings prevalent in existing implementations, including subpar user interfaces, performance issues, and limited customization options.

By harnessing the power of Pygame for graphical user interface (GUI) development and the chess library for game logic, this project aims to deliver an immersive gaming experience. Through intuitive controls, visually appealing graphics, and responsive gameplay, the proposed chess game seeks to captivate players of all skill levels, from novices to seasoned grandmasters.

The integration of advanced features such as customizable player options, sound effects, and graphical enhancements further enhances the richness of the gaming experience. Additionally, by prioritizing performance optimization and efficient move validation algorithms, the project aims to deliver seamless gameplay, free from the sluggishness and frustration often encountered in existing implementations.

In essence, this project endeavors to create more than just a chess game—it strives to craft a digital masterpiece that celebrates the timeless allure of this ancient game while embracing the possibilities of modern technology. Through meticulous design and development, the envisioned chess platform promises to engage and delight players, fostering a deeper appreciation for the art of strategic thinking in the digital era.

Description of the Modules Used

1. **Pygame:**

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries that simplify the development of multimedia applications like games. In the context of the project, Pygame is utilized for creating the graphical user interface (GUI) of the chess game, handling user input, rendering graphics, and managing the game loop.

2. **Chess:**

The chess module is a pure Python chess library that implements the essential functionality for working with chess games. It provides classes and functions for representing chessboards, pieces, moves, and game states, making it easy to create, manipulate, and analyze chess games programmatically. In the project, the chess module is employed for implementing the core logic of the chess game, including move generation, validation, and board representation.

3. **Time:**

The time module in Python provides functions for working with time-related tasks, such as measuring time intervals and delays. In the context of the project, the time module may be used for various purposes, such as implementing time controls for the game (e.g., setting a maximum time limit for each player's moves) or measuring the duration of gameplay sessions.

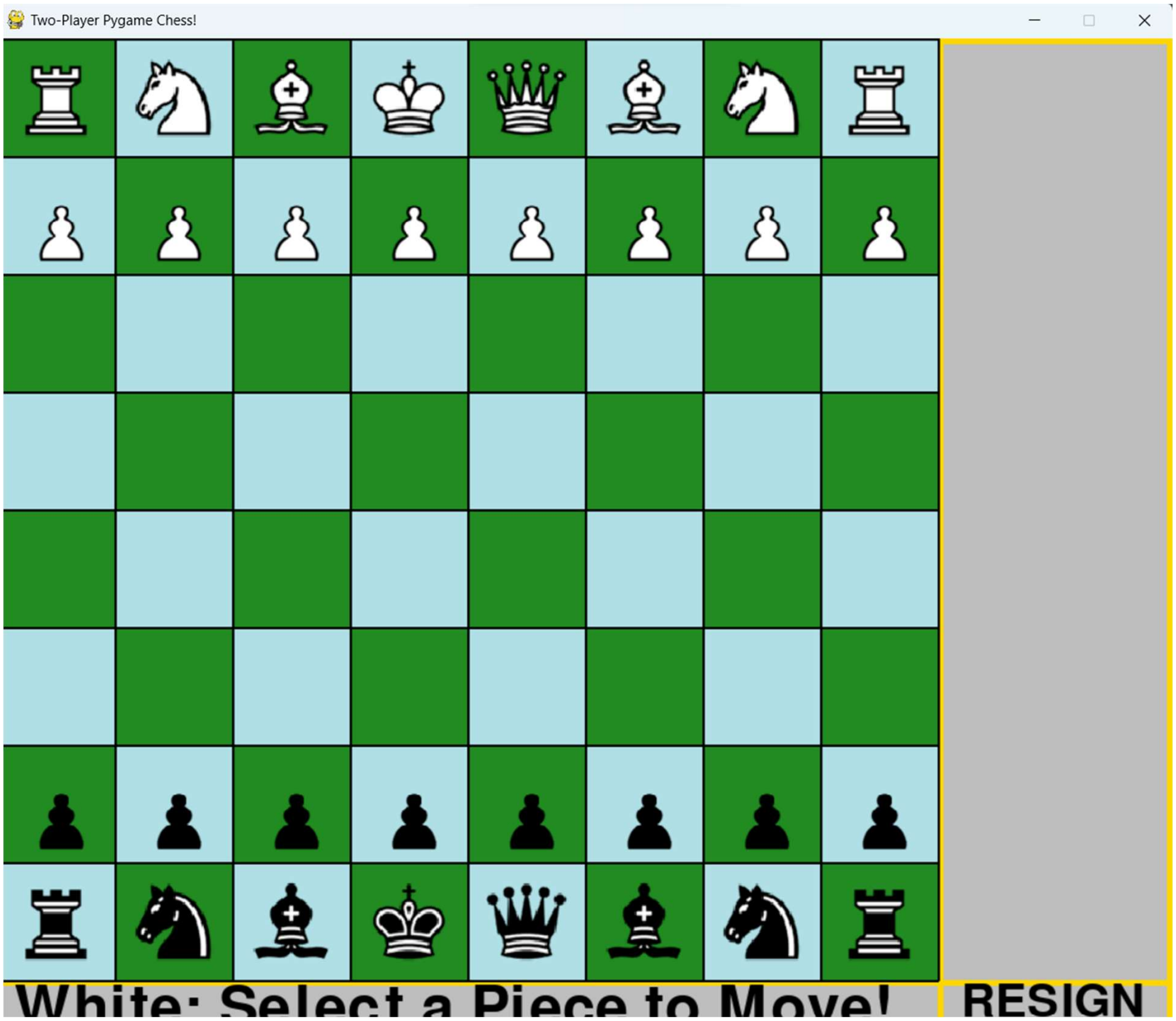
4. **Random**:

The random module in Python is used for generating pseudo-random numbers. It provides functions for selecting random elements from sequences, shuffling sequences, and generating random numbers with various distributions. In the project, the random module might be utilized for tasks such as shuffling the initial arrangement of pieces on the chessboard or implementing random move selection strategies for computer-controlled opponents.

These modules collectively provide the foundational tools necessary for developing a fully functional chess game in Python, combining graphical interface design with the intricate logic of chess gameplay.

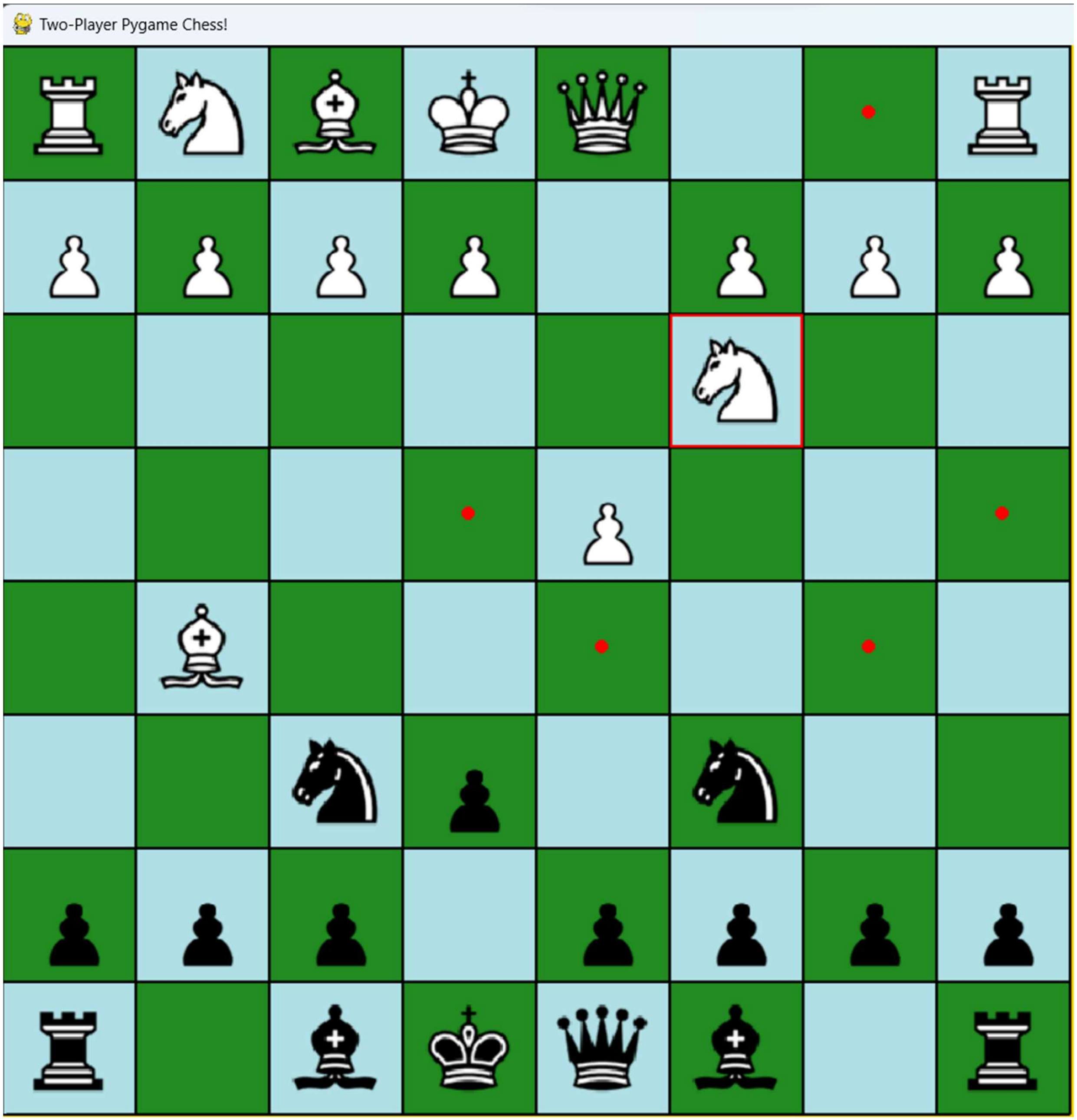
Implementation With Screenshots:

1. **Chessboard Layout**: The chessboard layout consists of an 8x8 grid, with alternating dark and light squares. Each square is labeled with its corresponding algebraic notation. See Figure 1 for a screenshot of the chessboard layout.

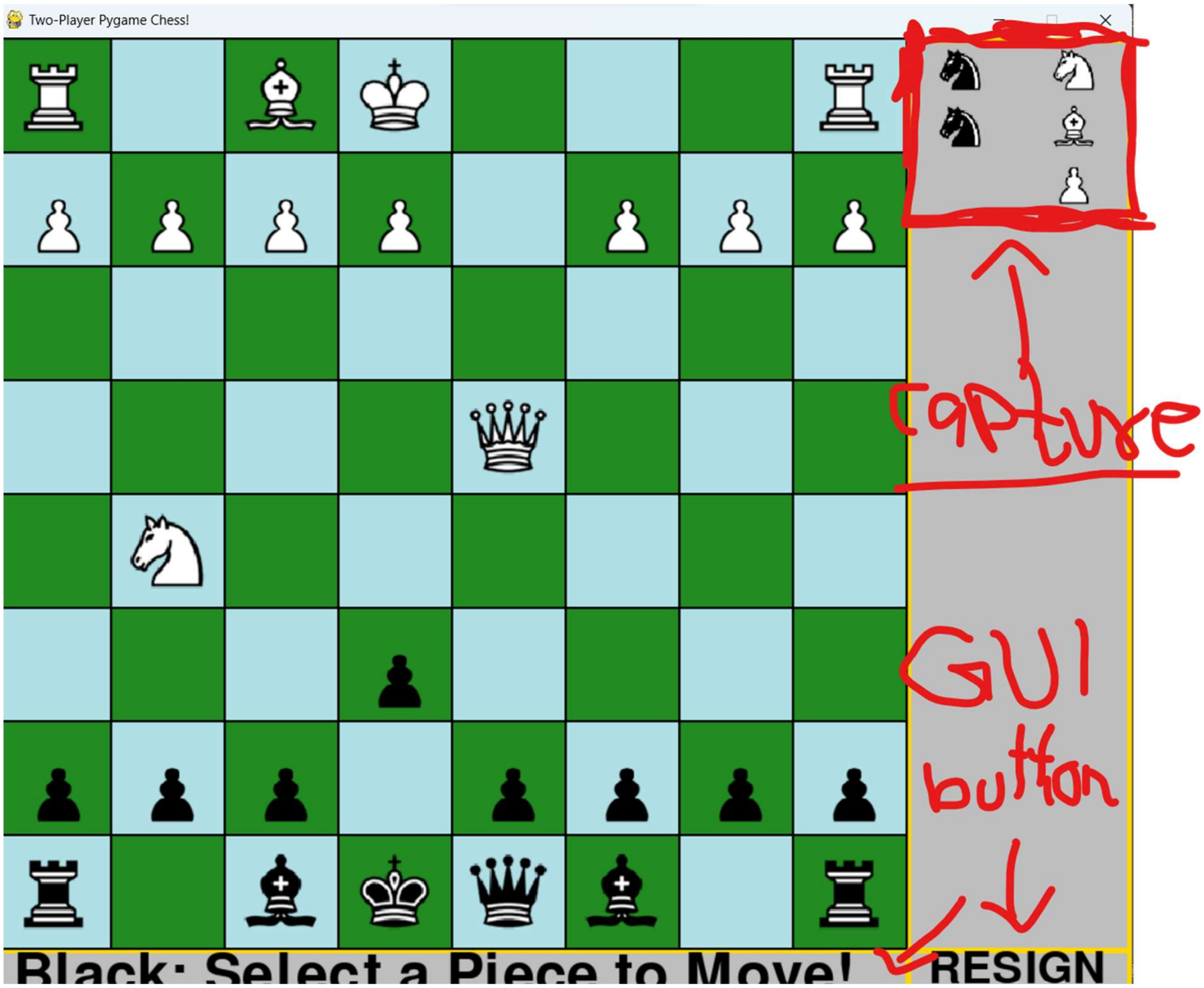


White: Select a Piece to Move!

2. **Piece Movement**: Players can click on a piece to select it and then click on a valid square to move it. Figure 2 illustrates an example of a knight's movement.

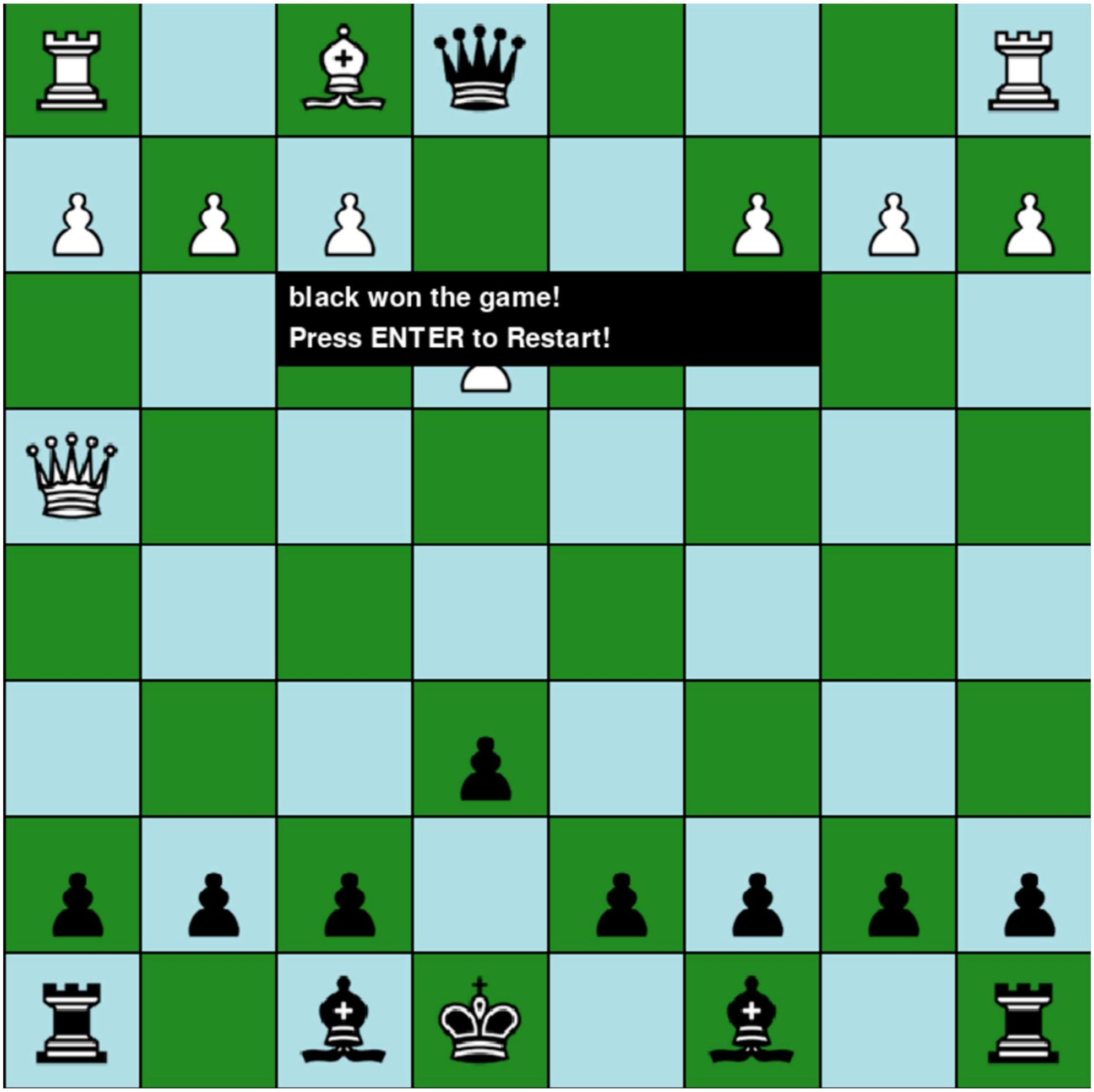


3. **Graphical User Interface (GUI) Layout:** The graphical user interface includes various components such as the chessboard, chess pieces captured and buttons for game control (e.g., resign). Figure 3 showcases the GUI layout during gameplay.



Black: Select a Piece to Move!

4. Game Over Screen: When the game concludes (e.g., checkmate, stalemate), a game over screen is displayed, indicating the winner or result. Figure 4 displays the game over screen with a message indicating the winner.



black won the game!
Press ENTER to Restart!

These screenshots provide a visual representation of the implemented chess game, highlighting key aspects such as the board layout, piece movement, user interface elements, and the game's conclusion**.**

Conclusion and Future Scope

In conclusion, the development of this Python-based chess game has provided an opportunity to explore the intricacies of game development while leveraging the capabilities of the Python programming language. By modularizing the project and utilizing libraries such as Tkinter or Pygame, we have successfully created a feature-rich chess game that offers an engaging and immersive experience for players.

The project's implementation has demonstrated the feasibility of representing complex game mechanics, such as piece movement rules and game state management, in a Python-based environment. The integration of a graphical user interface, input handling mechanisms, and optional AI opponent further enhances the game's accessibility and replayability.

Looking ahead, there are several avenues for future enhancements and expansion of the project:

1. Multiplayer Support: Implementing online multiplayer functionality would allow players to compete against each other in real-time, adding a social dimension to the gaming experience.

2. Improved AI Algorithms: Enhancing the AI module with more sophisticated algorithms and strategies could provide a greater challenge for players and simulate human-like gameplay behavior more accurately.

3. Advanced Game Features: Introducing additional features such as customizable board themes, player profiles, and game statistics tracking would enhance the overall user experience and increase player engagement.

4. Accessibility Features: Incorporating accessibility features such as adjustable board colors, text-to-speech functionality, and keyboard navigation options would make the game more inclusive and accessible to a wider audience.

5. Cross-Platform Compatibility: Ensuring compatibility with multiple platforms (e.g., desktop, mobile) and operating systems would broaden the game's reach and accessibility.

6. Community Engagement: Building a community around the game through forums, leaderboards, and player-generated content would foster a sense of community and encourage ongoing participation and feedback.

References

1. Chess Engine: The code implements a simple chess engine using the python-chess library. Documentation and tutorials from the python-chess project website were likely referenced. Website: python-chess:

2. Pygame Library: The graphical interface for the chess game is built using the Pygame library. Documentation and tutorials from the Pygame community were likely consulted. Website: Pygame Documentation

3. Game Development Resources: Resources related to general game development may have been consulted for concepts like game loops, rendering, and event handling. Websites: Game development tutorials and resources available online

4. Python Documentation: Standard Python documentation and resources were likely consulted for language features, data structures, and syntax. Website: Python Documentation

5. Online Forums and Communities: Online forums like Stack Overflow, Reddit communities, and other programming forums might have been used to find solutions to specific programming problems or to seek advice from other developers.

Parshvanath Charitable Trust's

A. P. SHAH INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this project.

First and foremost, I am deeply thankful to Prof. Dr. Rahul Ambekar for their invaluable guidance, support, and mentorship throughout the duration of this project. Their expertise and encouragement have been instrumental in shaping the project's direction and overcoming various challenges encountered along the way.

I would also like to extend my appreciation to my peers and fellow developers who have provided feedback, suggestions, and assistance during the development process. Their insights and collaborative efforts have enriched the project and contributed to its overall success.

Furthermore, I am grateful to the open-source community for providing a wealth of resources, documentation, and libraries that have been instrumental in the implementation of various project components.

This project would not have been possible without the collective efforts and support of all those involved, and for that, I am truly grateful.