**Project Report**

**Title:**

**Quantum Chess: A Pygame-Based Simulation Integrating Probabilistic Moves and Quantum Mechanics Concepts**

**Group Members:**

**Arham Mahmood 21K-3604**

Hayyan (21K-3634)

Bilal (21K-3551)

**1. Introduction**

This project introduces a novel approach to traditional chess by incorporating fundamental principles of quantum mechanics into a simplified chess simulation. Developed in Python using the Pygame library, the system enables chess pieces to exist in multiple probabilistic states simultaneously, emulating concepts such as superposition and probability collapse upon observation. The implementation involves two specialized quantum-inspired chess pieces — the **Quantum Knight (QN)** and the **Entangler (EN)** — and provides a graphical user interface (GUI) to allow players to interact with the game environment.

**2. Objectives**

The primary objectives of this project are as follows:

* To design and implement an interactive chessboard simulation using Python and Pygame.
* To incorporate quantum mechanics concepts such as probabilistic states and superposition into the behavior of chess pieces.
* To implement quantum moves, allowing pieces to split their position into probabilistic states.
* To develop an intuitive graphical interface for rendering the gameboard and managing user interactions.

**3. Tools and Technologies**

| **Technology/Library** | **Purpose** |
| --- | --- |
| Python 3.x | Core programming language |
| Pygame | Game development library for GUI rendering |
| NumPy | Numerical computations and array operations |
| Math, Random | Mathematical functions and probability handling |

**4. System Overview**

The project consists of three main modules:

* **Piece Class:** Manages the properties of individual chess pieces, their quantum states, and collapse mechanisms.
* **QuantumChessGame Class:** Handles the gameboard, piece placement, player turns, and move logic.
* **GUI Class:** Manages the graphical rendering of the board and pieces, processes player interactions, and updates the display in real-time.

**5. Game Mechanics**

**5.1 Board Configuration**

* The chessboard comprises a **10×10 grid**, where each square alternates between white and grey.

**5.2 Pieces**

The simulation features two specialized pieces:

* **Quantum Knight (QN):** Moves in predefined knight-like patterns.
* **Entangler (EN):** Moves horizontally or vertically, limited to three squares in any direction.

Each piece is characterized by:

* A specific type and color.
* A list of quantum\_states, indicating possible positions and associated probabilities.
* An optional entangled\_with attribute for future implementation of entanglement behavior.

**5.3 Quantum Moves**

* Moves can be classified as **classical** (certain moves) or **quantum** (probabilistic moves).
* A classical move collapses a piece’s probabilistic states to a single position.
* A quantum move splits the piece’s probability between its current state and a new position, simulating quantum superposition.

**6. Key Algorithms and Methods**

**6.1 collapse\_state()**

This method selects a position for the piece based on weighted probabilities, effectively collapsing its quantum state to a single, definite position.

**6.2 get\_possible\_moves(piece)**

Determines all possible legal moves for a given piece based on its type and current position.

**6.3 make\_move(piece, new\_position, is\_quantum)**

Executes a move by either adding a new probabilistic state (for quantum moves) or collapsing the existing state to a new position (for classical moves). It also manages turn transitions between players.

**7. Graphical User Interface (GUI)**

**7.1 Board Rendering**

The GUI draws a 10×10 grid with alternating square colors to visually represent the chessboard.

**7.2 Piece Rendering**

Pieces are displayed using Unicode chess symbols. The transparency of each piece reflects the probability of its respective state, allowing players to visually interpret quantum superpositions.

**7.3 Click Handling**

The system processes player clicks for piece selection and movement. It ensures that only pieces belonging to the current player can be selected and that moves adhere to the defined game mechanics.

**8. Features**

* Simulation of probabilistic quantum states and state splitting.
* Collapse of quantum states upon classical moves.
* Basic move validation for specialized pieces.
* Turn-based move control between players.
* Real-time graphical updates and intuitive player interaction.

**9. Limitations**

* The current implementation does not support check or checkmate detection.
* Only two piece types are included.
* No mechanism for capturing pieces or advanced chess rules is implemented.
* Move patterns are limited for demonstration purposes.
* Entanglement behavior is not yet active and reserved for future development.

**10. Future Work**

Several enhancements and extensions are proposed for future iterations of this project:

* Implementing entanglement mechanics to allow pieces to be interconnected.
* Expanding the game to include full chess rules and additional piece types.
* Enhancing move validation to prevent illegal moves and piece collisions.
* Introducing sound effects and animations for a more engaging user experience.
* Adding the ability to save and load game states.
* Developing an AI opponent capable of handling quantum moves.
* Visualizing entanglement links and probability distributions more explicitly.

**11. Conclusion**

This project successfully demonstrates the integration of quantum mechanics principles such as superposition and probabilistic outcomes into a traditional board game framework. Through the development of a functional prototype and an interactive GUI, the simulation provides an accessible and educational platform for exploring quantum concepts in a game-based setting. The foundation established through this implementation offers significant potential for further research and development in quantum-inspired game theory and educational tools.