

QUANTUM TIC TAC TOE

P.T.S.PRIYA , A.S.SANDHYA DEVI

2 MCA Final Semester,

Master of Computer Applications,

Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India

ABSTRACT:

Quantum Tic Tac Toe is an innovative twist on the classic game of Tic Tac Toe, inspired by principles of quantum mechanics such as superposition and entanglement. Unlike the traditional version, where players place definite marks (X or O), this quantum version allows players to place entangled moves that can exist in multiple states simultaneously until a collapse occurs. The game introduces unique gameplay mechanics such as quantum moves, entangled states, and wave function collapse, making it both a learning tool and an entertaining experience.

The project was developed using Python and integrated with a graphical user interface using Kivy, allowing for intuitive interaction and seamless gameplay. It supports features like reset, collapse, theme switching, and automatic cycle detection. These functionalities reflect core quantum concepts through visual and interactive methods, making the game suitable for both educational and recreational use.

The system architecture includes modules for input handling, entanglement logic, quantum state tracking, and user interface. By abstracting quantum behavior into game logic, this project demonstrates how complex scientific ideas can be represented through playful and accessible software design.

Index Terms :Quantum Computing, Superposition, Entanglement, Python, Kivy, Game Design, Cycle Detection, Quantum Collapse.

I. INTRODUCTION:

This project, **Quantum Tic Tac Toe**, aims to bridge the gap between **game-based learning** and **quantum theory** by redesigning the classical Tic Tac Toe game with quantum-inspired logic. The game introduces two fundamental quantum phenomena—**superposition** and **entanglement**—into its gameplay, allowing players to place their moves in a superposed state across multiple cells. These entangled moves are stored temporarily and only resolved into classical Xs and Os during a process called **collapse**, which imitates quantum. Unlike the deterministic nature of classical Tic Tac Toe, this game allows players to interact with uncertain and overlapping move states, significantly enhancing the complexity and educational value. The **collapse mechanism**, which transforms quantum-like states into final outcomes, adds a probabilistic twist and deeper strategic elements to an otherwise simple game.

Developed using the **Python programming language** and its standard **Tkinter library** for the graphical user interface, the application is equipped with a clean and responsive layout. Additional features such as **theme switching (Dark/Light mode)**, **reset and exit functionality**, and **win history tracking** improve the overall user

experience. The game logic includes checks for valid quantum moves, entanglement rules, and win conditions after the collapse phase.

1.1 Existing System

The classical Tic Tac Toe game is a well-known two-player strategy game played on a 3×3 grid, where players alternately place deterministic moves—either 'X' or 'O'—until one player achieves three in a row or the game ends in a draw. While this game is simple, it is entirely based on **predictable and deterministic logic**, leaving little room for innovation, complexity, or unpredictability once the optimal strategy is known. In terms of digital implementations, countless basic Tic Tac Toe games exist across web, mobile, and desktop platforms. These versions primarily utilize hardcoded logic or decision trees, often including features like AI-based opponents, score tracking, and sometimes multiplayer capabilities. However, these systems are **linear in design and lack any form of uncertainty, randomness, or dynamic state transition** inspired by more complex real-world systems such as those found in quantum mechanics. Existing versions do not incorporate any quantum behavior such as **superposition (a symbol being in two places at once)** or **entanglement (moves that are connected across multiple cells)**. Furthermore, traditional systems collapse into a known end-state immediately after each move, offering no probabilistic or non-deterministic outcome that would reflect quantum phenomena.

1.1.1 Challenges

- **Deterministic Gameplay:**
Predictable moves and fixed outcomes make the game repetitive once optimal strategy is known.
- **Lack of Complexity:**
Limited strategic depth with a 3×3 grid; most games end in a draw.
- **No Learning Potential:**
Does not teach or simulate advanced concepts; purely for recreation.
- **Fixed Moves:**
A cell holds only one player's move at a time; move decisions are binary.
- **UI Simplicity:**
Simple interfaces, but not suitable for representing complex states.
- **Limited Engagement:**
Quick to play but often repetitive; lacks variation in gameplay.

1.2 Proposed System

The proposed system redefines the traditional Tic Tac Toe game by integrating **quantum mechanics-inspired features** such as **superposition**, **entanglement**, and **state collapse** into the gameplay mechanics[1]. The system is developed as a Python-based desktop application using the tkinter GUI framework, designed to simulate the behavior of quantum systems in an intuitive and interactive manner.

In the classical game, each move results in a definite placement of a symbol ('X' or 'O') in a specific grid cell. However, in the **Quantum Tic Tac Toe** system, a player's move can exist in a **superposition** of two cells simultaneously. These **entangled moves** are tracked and visually represented in the game grid until a **collapse** is triggered, resolving the quantum state into a classical outcome.

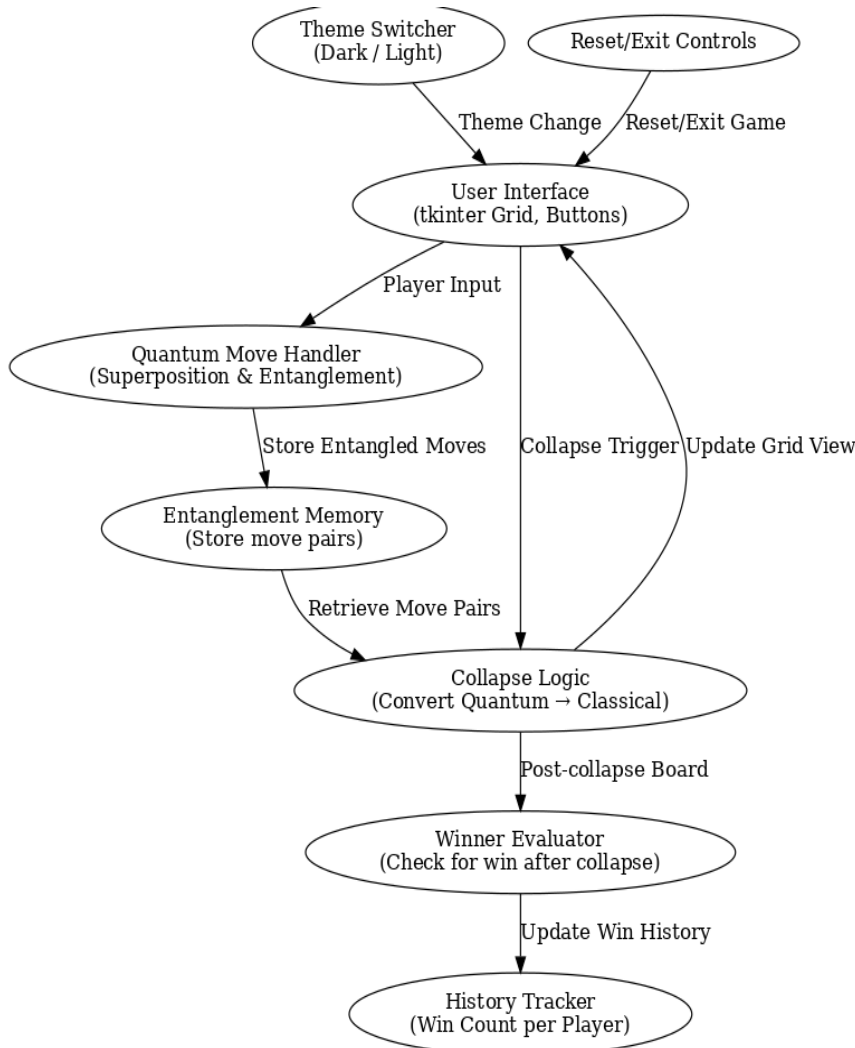


Fig: 1 Proposed Diagram

1.2.1 Advantages

- **Educational Value:**

The game introduces fundamental quantum mechanics concepts such as **superposition**, **entanglement**, and **state collapse** in a visual and interactive way.

- **Enhanced Strategic Complexity:**

Unlike traditional Tic Tac Toe, where the optimal strategy leads to frequent draws, this quantum version introduces **probabilistic decision-making**.

- **Encourages Interest in Quantum Computing:**

By gamifying key principles of quantum mechanics, the project helps **bridge the gap between entertainment and scientific curiosity**.

- **Dark And Light Theme Toggle:**

Provides users with **personalization and accessibility**, enhancing during extended gameplay or educational sessions

- **Win History Tracking:**

Tracks each player's win count, enabling friendly competition and **progress monitoring** across sessions.

- **Promotes Social Responsibility:**

Encourages people to donate rather than waste food. Builds a community driven by compassion and environmental awareness.

II. LITERATURE REVIEW

2.1 Architecture

The architecture of the **Quantum Tic Tac Toe** system is designed to simulate quantum mechanics-inspired logic through a modular, interactive Python application[4]. It integrates **entangled move generation, collapse resolution, winner detection**, and **user interaction** within a GUI-based environment using tkinter. The architecture is lightweight, standalone, and highly interpretable, making it ideal for both learning and demonstration purposes.

The system operates in a structured sequence of phases:

1. **User Interaction** through a GUI interface.
2. **Quantum Move Simulation** (entanglement of symbols)
3. **Move Collapse Logic** upon trigger
4. **Post-Collapse Evaluation** for game outcome
5. **Theme Switching & Session Controls**

The overall system is designed to demonstrate abstract quantum behavior using deterministic programming logic and real-time visual feedback.

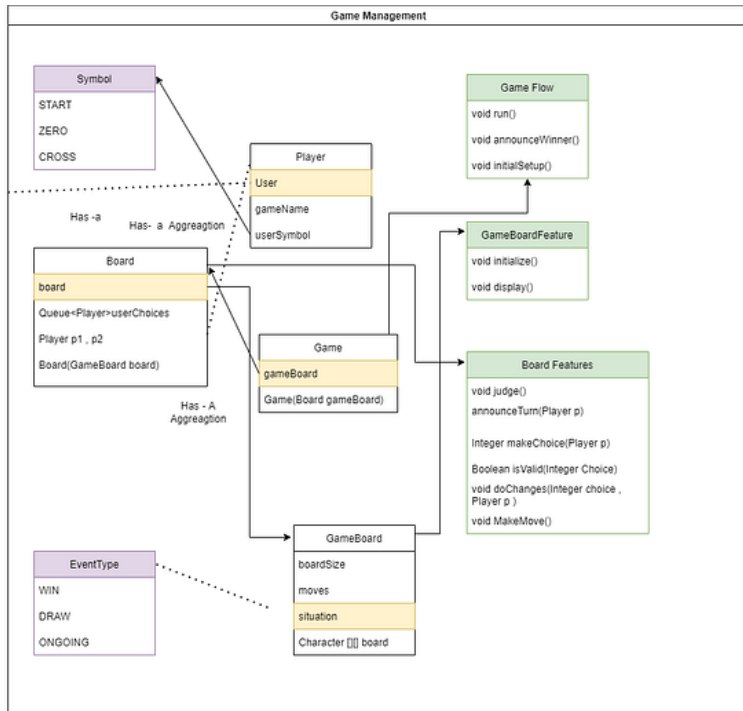


Fig:2 The architecture of the **Quantum Tic Tac Toe** system.

2.2 Algorithm:

The core logic of the **Quantum Tic Tac Toe** game is built upon a series of conditional and iterative processes that simulate quantum principles such as **entanglement**, **superposition**, and **state collapse**[8]. The following algorithm outlines the system's step-by-step workflow from move registration to collapse and winner determination.

□ Initialize Game:

Set up a 3×3 grid, set the current player to X, and prepare move counters and win history.

□ Make Quantum Move:

Each player selects two cells; their move (e.g., X1) is stored as an entangled pair in both cells.

□ Store & Display:

Entangled moves are saved in memory and shown on the board with player symbol and move number.

□ Collapse Moves:

When collapse is triggered, the earliest move in each entangled cell is selected as the final move.

□ Check Winner:

After collapse, check all rows, columns, and diagonals for three identical symbols to declare the winner.

2.3 Techniques:

1.Event-Driven Programming:Used to handle user interactions such as cell clicks, theme switching, and collapse triggers.

2.Simulation of Superposition & Entanglement:Each move is split into two positions, stored with player identity and move number to simulate quantum states.

3.State Management:In-memory data structures (like dictionaries) manage move history, entangled states, and player turns.

4.Custom Collapse Logic:Resolves quantum moves into classical outcomes based on move priority, mimicking quantum measurement.

5.Conditional Win Evaluation:After collapse, the board is scanned using logical conditions to determine the winning player.

6.Graphical Interface Design:Built using Python's tkinter to create a responsive, interactive 3×3 grid and control buttons.

7.Theme Toggle Implementation:Allows dynamic switching between light and dark modes using conditional styling logic.

8.Session Tracking:Win history is maintained within the session for both players, updated after each game.

These combined techniques ensure that the system remains fast, secure, and easy to maintain.

2.4 Tools:

- ☐ **Python 3:**Core programming language used to build the game logic and interface.
- ☐ **Tkinter:**Python's built-in GUI library used to design the game board, buttons, and layout.
- ☐ **IDLE / VS Code / PyCharm:**Integrated Development Environments (IDEs) used for writing, running, and debugging code.
- ☐ **Data Structures (List, Dictionary):**Used to store entangled moves, track player turns, and manage game state.
- ☐ **Graphviz :**Used to create system architecture and UML diagrams for documentation.
- ☐ **Windows OS:**Platform where the application was developed and tested.
- ☐ **Snipping Tool / Paint:**Used to capture game screenshots for reports or presentation

2.5 Methods:

- **Requirement Analysis:** Identified how quantum mechanics concepts like superposition, entanglement, and collapse could be translated into a game format using Python.
- **Modular Programming:** Divided the project into separate modules such as move handling, collapse logic, winner detection, theme control, and user interface.
- **Event-Driven Logic:** Used Python's tkinter library to handle user interactions through buttons and cell clicks in real time.
- **Iterative Development:** Developed and tested the game feature-by-feature, improving functionality in stages (e.g., theme switching, history tracking).
- **Algorithmic Design:** Applied logical conditions and data structures to simulate entangled states, perform collapse, and evaluate the board.
- **UI Design and Feedback:** Created a visually intuitive interface with move display, win announcements, and theme customization using tkinter.
- **Testing and Debugging:** Manually tested all possible game scenarios to verify correctness of move logic, collapse resolution, and winner detection.

III. METHODOLOGY

3.1 Input: The project is designed to simulate the abstract principles of quantum mechanics—such as superposition and entanglement—within a familiar and interactive gaming environment[6]. The system is built using **Python 3** with the tkinter GUI framework and is organized into distinct modules responsible for move management, entanglement tracking, collapse resolution, and user interaction.

- **Player Move Selection:** Players click on two different grid cells to place their quantum (entangled) move. Each move is labeled with the player symbol and move number (e.g., X1, O2).
- **Collapse Button:** Triggers the collapse of entangled moves into classical outcomes. Resolves superpositions and updates the board accordingly.
- **Switch Theme Button:** Allows players to toggle between dark mode and light mode and Enhances visual comfort and user experience.
- **Reset Button:** Clears the board for a new game while retaining win history. resets entanglement memory and move counter.
- **Exit Button:** Closes the application and ends the session.

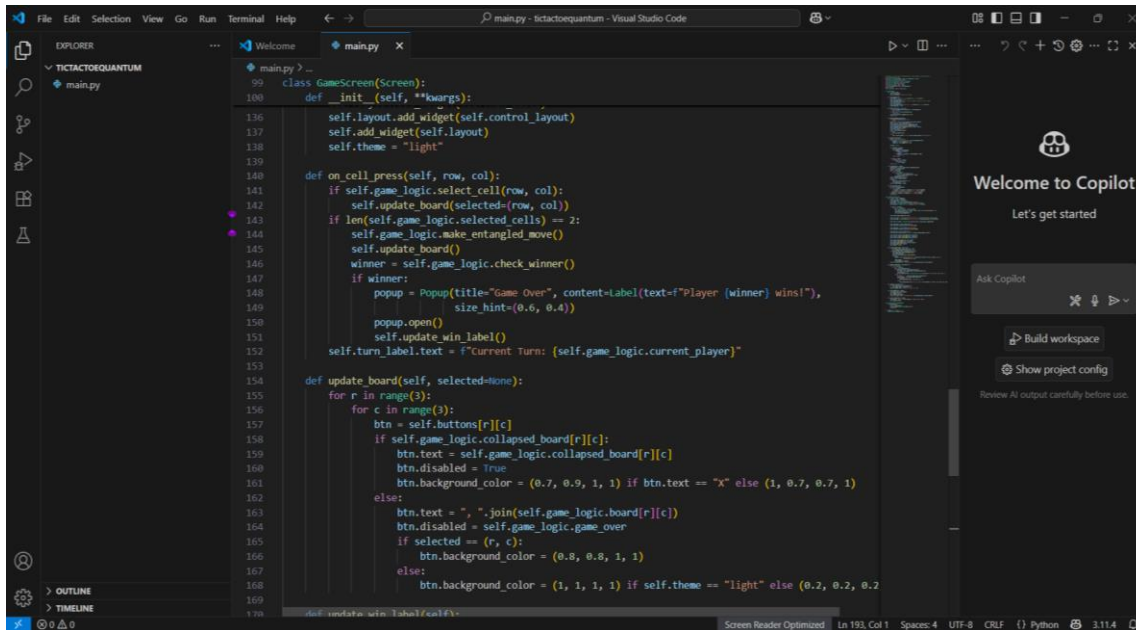


Fig:3 Input handling commands for creating input methods.

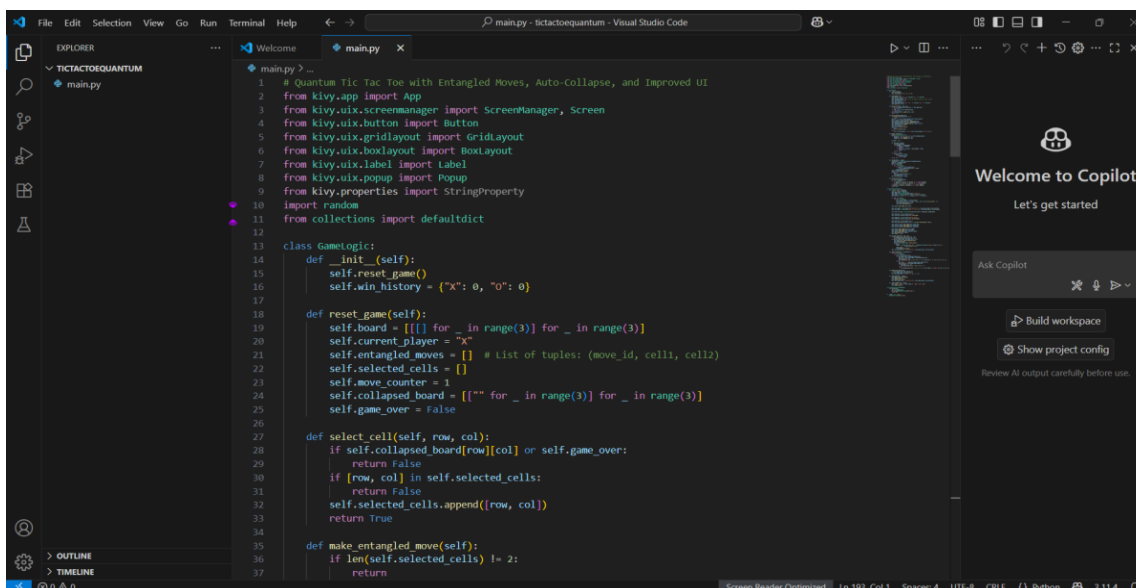


Fig:4 Game Logic.

1. Game Logic :

Tech Used: Python (OOP), Lists, Dictionaries, DFS

Purpose: Handles quantum rules, entanglement, cycle detection, collapsing moves, checking win.

Key Part: GameLogic class using Python data structures and custom algorithms.

2. Quantum Entangled Moves :

Tech Used: Python classes and list manipulation.

Purpose: Players select 2 cells to simulate entanglement; move is stored like X1, O2, etc.

Key Part: self.entangled_moves list with move ID and positions

3.Cycle Detection & Auto Collapse :Tech Used: Python DFS algorithm, defaultdict

Purpose: Automatically finds quantum cycles and collapses one move .

Key Part: detect_cycle() and collapse() methods

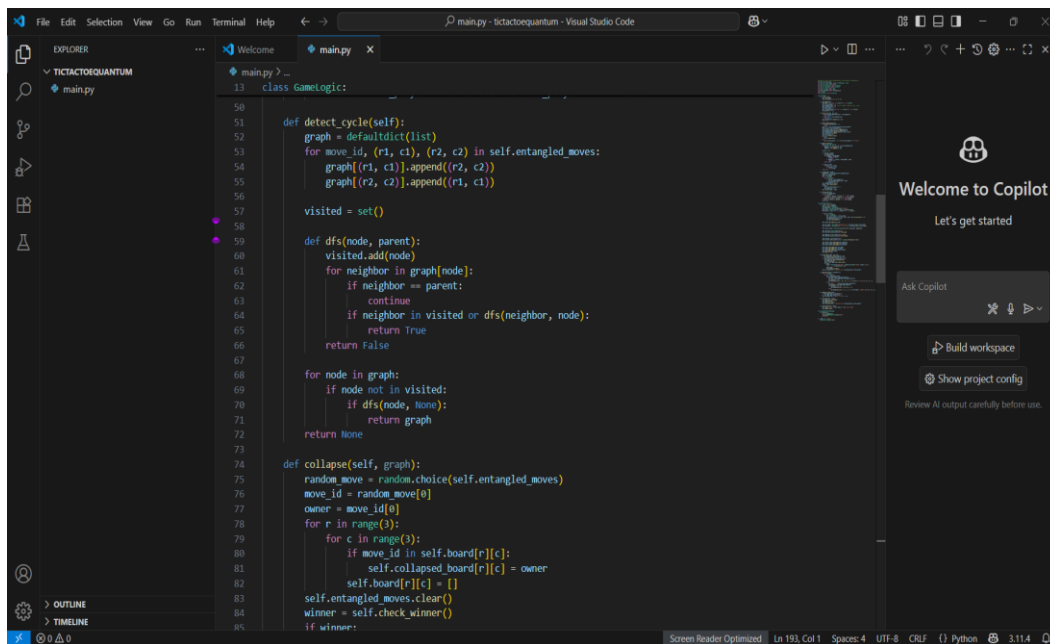


Fig 5: Logic on Cycle Detection

4.Graphical User Interface (UI) :

Tech Used: Kivy (Button, Label, BoxLayout, GridLayout).

Purpose: Builds the screen, places the grid and buttons, shows scores and turns.

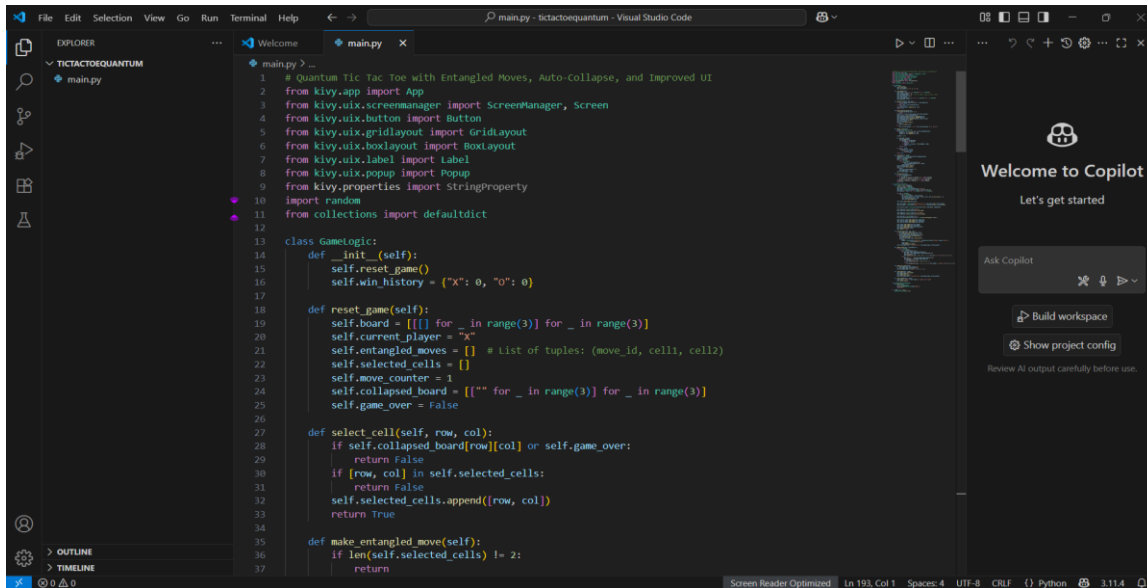
Key Part: GameScreen class

5. Dynamic Styling (Theme & Colors) :

Tech Used: Kivy background_color, conditionals.

Purpose: Switch between Light/Dark theme, color X/O buttons, highlight selected cells.

Key Part: update_board() method.



```

1 # Quantum Tic Tac Toe with Entangled Moves, Auto-collapse, and Improved UI
2 from kivy.app import App
3 from kivy.uix.screenmanager import ScreenManager, Screen
4 from kivy.uix.button import Button
5 from kivy.uix.gridlayout import GridLayout
6 from kivy.uix.boxlayout import BoxLayout
7 from kivy.uix.label import Label
8 from kivy.uix.popup import Popup
9 from kivy.properties import StringProperty
10 import random
11 from collections import defaultdict
12
13 class GameLogic:
14     def __init__(self):
15         self.reset_game()
16         self.win_history = {"X": 0, "O": 0}
17
18     def reset_game(self):
19         self.board = [[[] for _ in range(3)] for _ in range(3)]
20         self.current_player = "X"
21         self.entangled_moves = [] # list of tuples: (move_id, cell1, cell2)
22         self.selected_cells = []
23         self.move_counter = 1
24         self.collapsed_board = [[[] for _ in range(3)] for _ in range(3)]
25         self.game_over = False
26
27     def select_cell(self, row, col):
28         if self.collapsed_board[row][col] or self.game_over:
29             return False
30         if [row, col] in self.selected_cells:
31             return False
32         self.selected_cells.append([row, col])
33         return True
34
35     def make_entangled_move(self):
36         if len(self.selected_cells) != 2:
37             return

```

Fig:6 Game interface and logic.

6. Player Input Handling :

Tech Used: Kivy's on_press event binding.

Purpose: Detects button clicks (cell selection), executes game logic.

Key Part: Button binding like btn.bind(on_press=...)

7. Win Tracking and Display :

Tech Used: Python variables + Kivy Label.

Purpose: Keeps count of wins for X and O, updates label dynamically.

Key Part: update_win_label() updates Label text.

8. App Lifecycle :

Tech Used: Kivy App, ScreenManager.

Purpose: Controls app start, screen flow, and running the game.

Key Part: QuantumTicTacToeApp class, App.run()

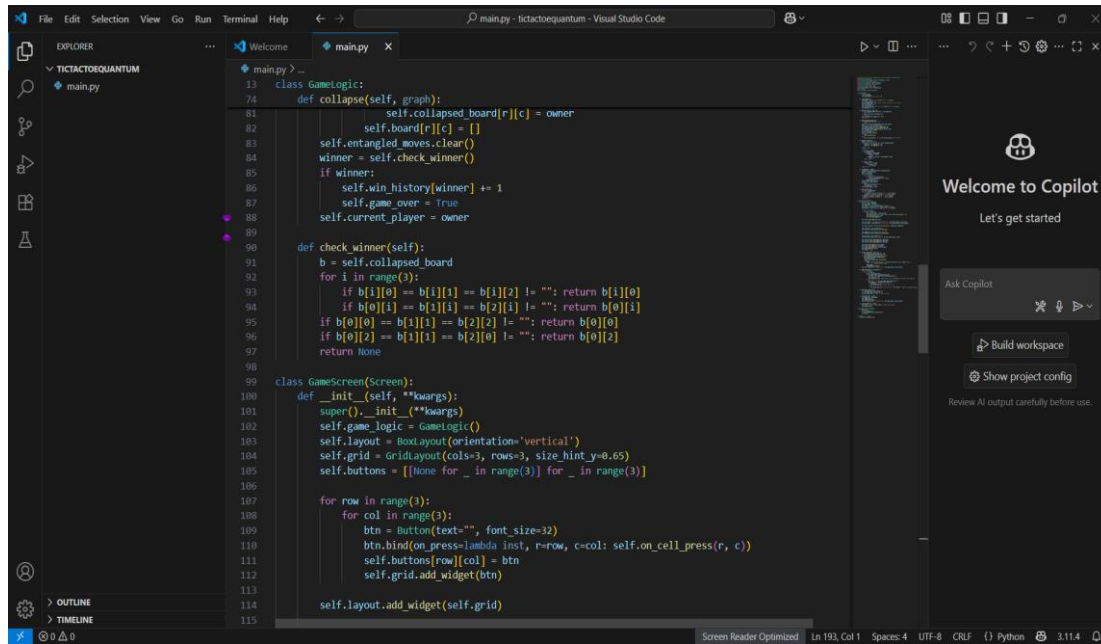


Figure 7: Tracking win count.

3.2 Method of Process

1. Understanding the Quantum Game Rules:

The first step involved researching the rules of Quantum Tic Tac Toe — particularly how moves are entangled, how quantum cycles form, and how collapses occur. These rules formed the core logic around which the game was designed.

2. Designing the Game Logic:

Using object-oriented programming in Python, core components were designed: A **GameLogic** class to manage the board, store entangled moves, detect cycles, and handle collapses. Entangled moves were represented as pairs of selected cells with unique IDs (e.g., X1, O2). A **graph-based cycle detection** algorithm (using Depth-First Search) was implemented to identify when a quantum collapse should occur.

3. Building the User Interface with Kivy:

A user interface was designed using the **Kivy framework** in Python[3]. A **GridLayout** was used to create a 3×3 game board. Buttons were added for each cell and for control options like Reset, Theme, and Exit. Also Labels were used to show the current player's turn and win history.

4. Integrating Entangled Input Handling:

Each player selects two different cells per turn to simulate entanglement. the app records these moves with unique IDs (e.g., X1, O2). A cycle detection algorithm checks if a loop of entangled moves is formed. When a cycle is found, the app automatically collapses the moves into classical X or O.

5. Visual Feedback and Theming:

Collapsed X and O cells are color-coded (blue for X, red for O). A theme switch button allows toggling between light and dark mode. Selected cells before collapse are highlighted for clarity. The interface updates in real time as the game progresses.

6. Testing and Debugging:

The game was tested with many move combinations for accuracy. Bugs like invalid inputs, multiple clicks, or early collapses were fixed. The cycle detection and collapse logic were debugged for edge cases. Win detection and UI updates were validated in both themes.

7. Finalization and Output:

The final app allows two entangled moves per turn with auto-collapse. The interface is fully responsive and shows turn info and win count. All control buttons function smoothly with themed visuals. The app runs as a standalone desktop game built with Python and Kivy.

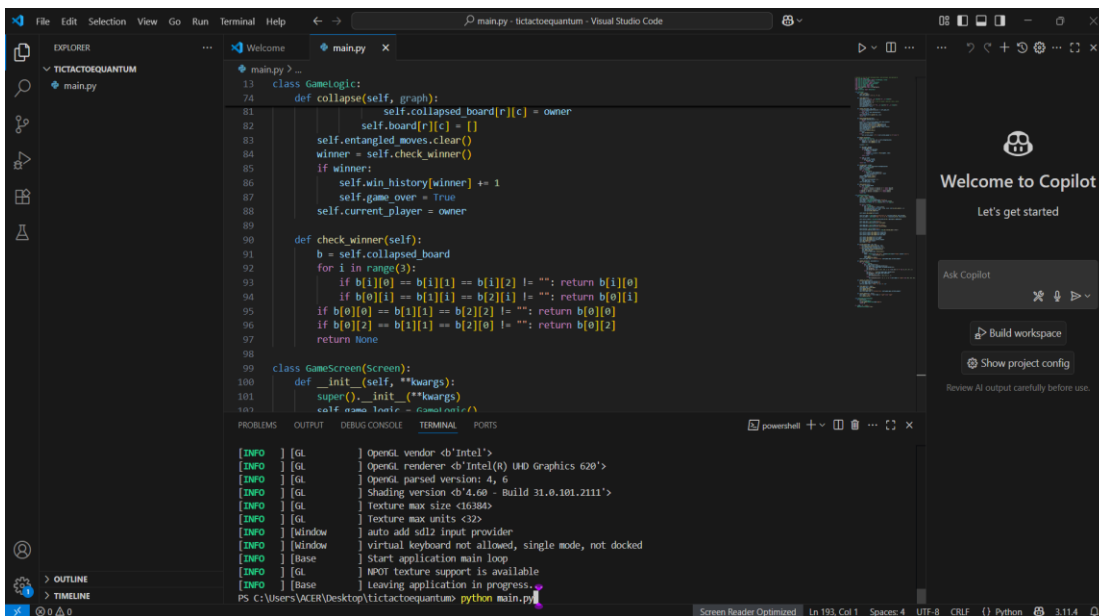


Fig :8 Running the file(main.py) by using it on terminal as (python main.py).

3.3 Output:

The output of the Quantum Tic Tac Toe game is a fully interactive Python application that simulates quantum principles such as **entangled moves**, **cycle detection**, and **automatic collapse** within a classical game environment.

1. Interactive Game Interface:

A 3×3 grid representing the game board, rendered using Kivy's GridLayout.

Each cell functions as a clickable button, dynamically updated to show:

Entangled moves (X1, O2, etc.), Collapsed values (X or O).

2. Turn and Win Tracking:

The Top label displays the **current player's turn** in real-time.

After a valid collapse and win, a **popup window displays the winner**.

A **win history label** keeps track of how many games each player has won.

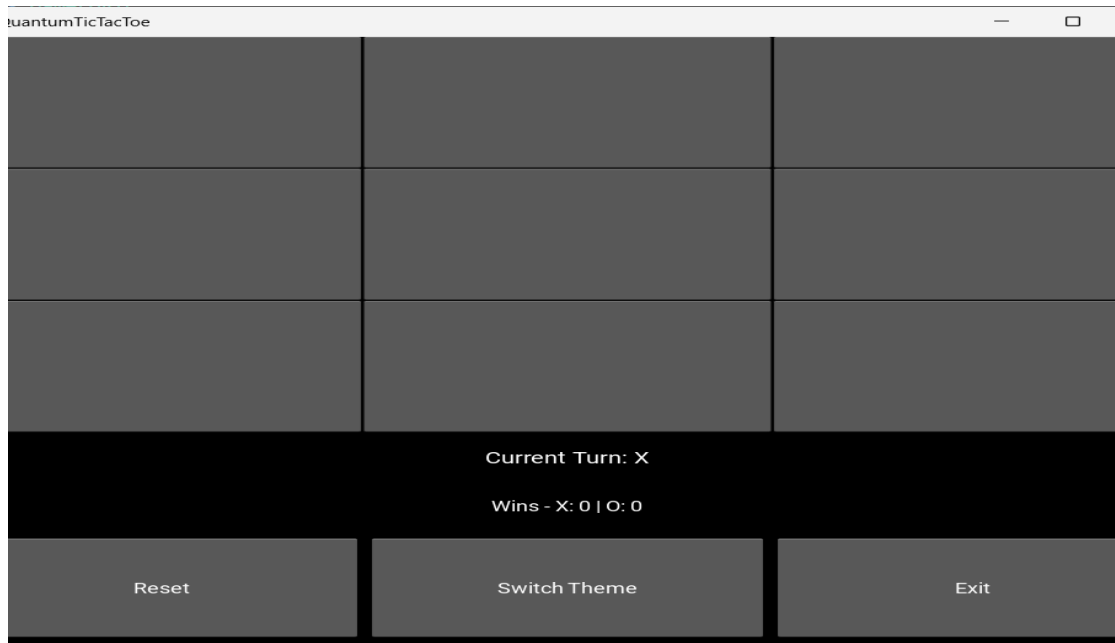


Fig:9 Game board of Quantum Tic Tac Toe game.

3. Automatic Collapse Output:

When a quantum entanglement cycle is detected, the application automatically chooses one move from the cycle at random.

Collapses entangled values to a single classical state (X or O).

4. Visual Output:

Collapsed cells are **highlighted**: Light blue background for X and for O. Active player's turn and selected entangled cells are visually marked and background color.

5. Functional Buttons:

Reset Button: Clears the board and resets game state without closing the app.

Switch Theme Button: Toggles between dark and light UI modes.

Exit Button: Closes the application gracefully.

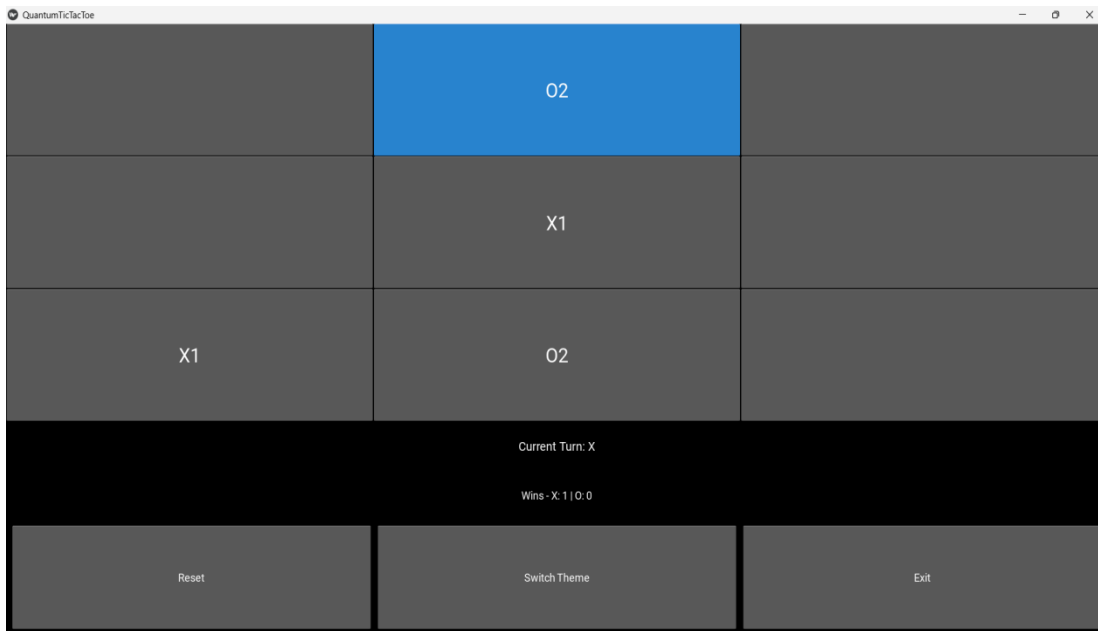


Fig:10 Highlighted background for x or o moves.

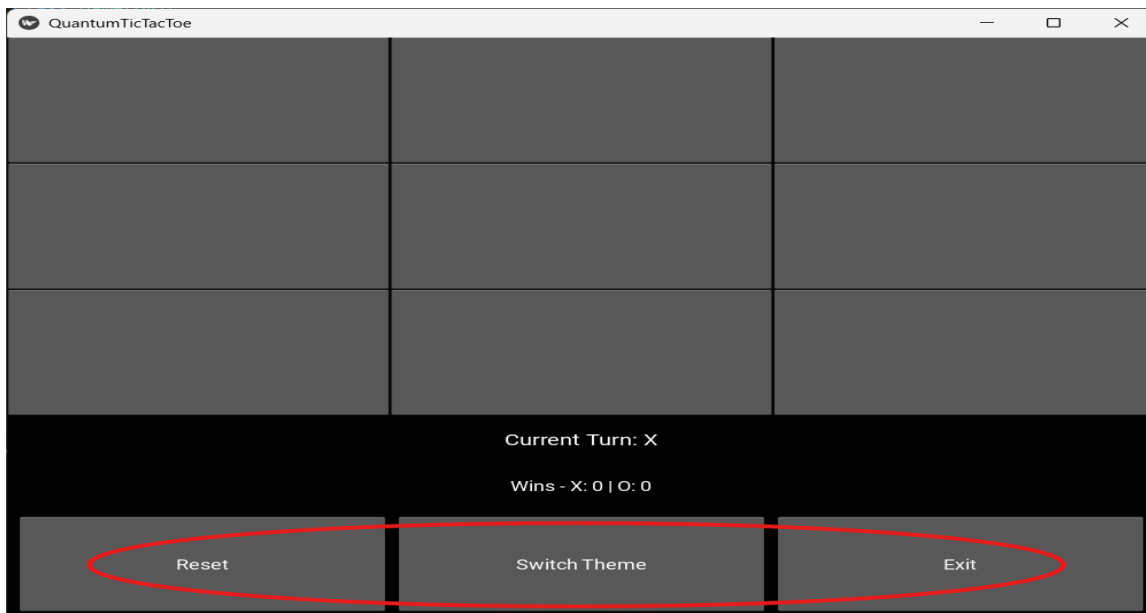


Fig: 11 Displayed Buttons.

IV. RESULTS:

The Quantum Tic Tac Toe game was successfully developed and tested using Python and the Kivy framework. The final application fulfilled all core objectives, demonstrating both functional gameplay and quantum behavior simulation through entangled moves and automatic collapses.

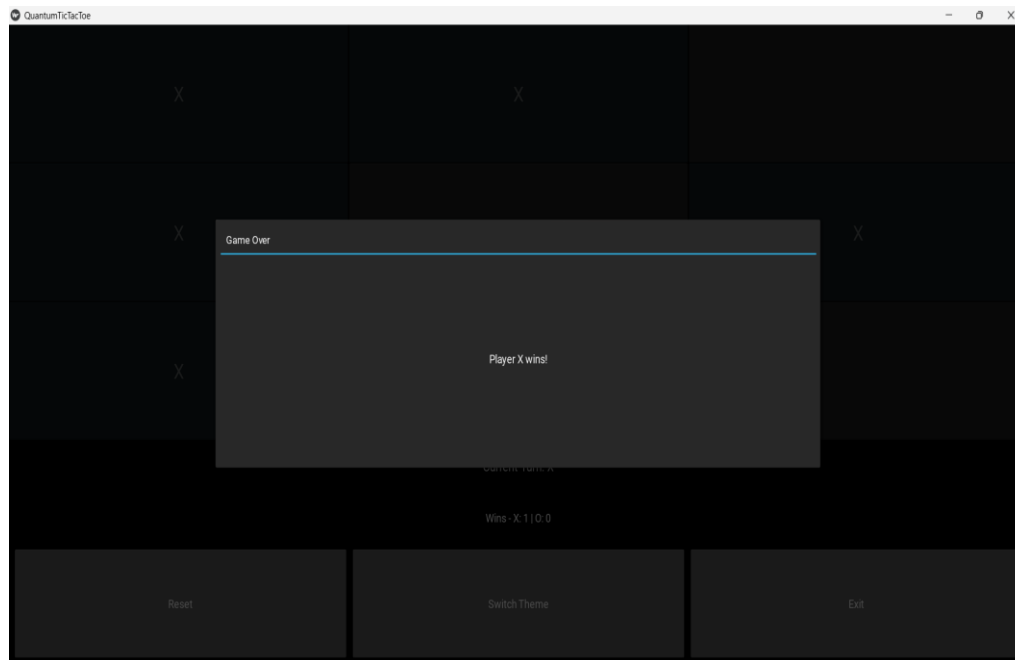


Fig: 12 A pop up window occurs when player wins.

1. Functional Accuracy:

The game allows each player to make two entangled moves per turn and Cycle detection is handled automatically using a graph-based DFS algorithm.

Upon detecting a cycle, the game correctly performs a collapse, resolving entangled moves into classical values.

The winner is announced accurately, and win history is updated accordingly.

2. Interface Responsiveness:

The UI updates in real-time after each interaction. Collapsed cells are color-coded, and the current turn is always displayed clearly.

Buttons disable automatically after a win or collapse, ensuring a smooth user experience.

3. Theme Switching:

Theme switching between light and dark mode works without affecting gameplay.

The game provides immediate visual feedback for selections, collapses, and game outcomes.

V. DISCUSSIONS

The development of the Quantum Tic Tac Toe game successfully blended fundamental quantum principles with classical game design[11]. By implementing features such as **entangled moves**, **cycle detection**, and **automatic collapse**, the application goes beyond a traditional game and acts as a simplified model to help users understand quantum behavior in an interactive format.

The project showcased how abstract quantum mechanics concepts can be simulated and visualized using a high-level programming language like Python. The automatic detection of cycles and random collapse of entangled states closely mimics the unpredictability and measurement behavior found in quantum systems.

During testing, the game consistently performed as expected, responding to valid user input, correctly collapsing cycles, and detecting win conditions. The theme toggle and visual feedback elements significantly enhanced the overall user experience, making the app not only educational but also enjoyable.

VI. CONCLUSION

The Quantum Tic Tac Toe project successfully demonstrates how core principles of quantum mechanics — such as superposition, entanglement, and measurement collapse — can be creatively integrated into a traditional game framework. Developed using Python and the Kivy framework, the application offers an interactive environment where users can experience two-move entanglement, automatic cycle detection, and visual collapse of quantum states.

The project not only meets its functional goals but also serves as an educational tool that simplifies abstract quantum behavior into a playable and intuitive format. It encourages critical thinking, enhances conceptual clarity, and provides a foundation for developing more complex quantum simulations or gamified models.

Overall, the project is a successful combination of theoretical innovation and practical software development. It opens the door for future improvements, such as online multiplayer support, animation-based collapses, or even real-time quantum randomness using QRNG APIs.

VII. FUTURE SCOPE

The Quantum Tic Tac Toe game provides a strong foundation for future enhancements that can elevate both its educational value and technical sophistication. One of the most promising directions is converting the desktop application into an Android mobile app using tools like Buildozer or Google Colab. This would allow wider accessibility and practical usage on smartphones. Visual improvements such as animations to show entanglement, collapse, and win detection can enhance user engagement and provide better clarity of quantum behaviors.

Additionally, the game could be expanded to support multiplayer modes, either online or via Bluetooth, enabling two players to interact in real time. Integrating an AI-based opponent would also allow single-player functionality, making the game more challenging and educational. Another significant improvement could involve the use of Quantum Random Number Generators (QRNGs) to introduce true quantum unpredictability during collapse events, simulating real quantum measurements. Lastly, embedding educational features like interactive tutorials, tooltips, and explanations of quantum principles would make the game a powerful tool for science outreach and classroom learning.

VIII. ACKNOWLEDGEMENT



Pinnamaraju T S Priya working as a HOD, Assistant Professor in Master of Computer Applications at Sanketika Vidya Parishad Engineering college, Visakhapatnam ,Andhra Pradesh with 13 years of experience in Master of Computer Applications (MCA), accredited by NAAC. Her area of intrest is in Computer Programming Using C, Computer Organisation, Software Engineering, Artificial Intelligence, Internet of Things(IoT) and Distributed Systems.



Adusumilli Sri Sandhya Devi is pursuing her final semester of MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Artificial intelligence K.Bhargavi has taken up her PG project on Quantum Tic Tac Toe and published the paper in connection to the project under the guidance Pinnamaraju T.S Priya Assistant Professor, HOD , SVPEC.

REFERENCES

- [1] Quantum Physics, retrieved from:url:[url: https://www.nature.com/subjects/quantum-physics](https://www.nature.com/subjects/quantum-physics)
- [2] Quantum Game Theory.

Retrieved from: [kurl: https://en.wikipedia.org/wiki/Quantum_game_theory](https://en.wikipedia.org/wiki/Quantum_game_theory)

- [3] V. Singh, B. K. Behera and P. K. Prasanta, Design of Quantum Circuits to Play Bingo Game in a Quantum Computer

- [4]Quantum-Tic-Tac-Toe

Retrieved from:https://en.wikipedia.org/wiki/Quantum_tic-tac-toe

- [5] A. Goff, Quantum Tic-Tac-Toe as Metaphor for Quantum Physics, AIP Conf. Proc. **699**, 1152-1159 (2004).

- [6] Roman Origin of Tic Tac Toe.

Retrieved from <https://www.sweettoothdesign.com/games-tic-tac-toe>

- [7] J. N. Leaw and S. A. Cheong, Strategic insights from play ing quantum tic-tac-toe, J. Phys. A: Math. Theor. **43**,

455304 (2010).

[8] Morris Games.

Retrieved from <http://www-cs.canisius.edu/~salley/SCA/Games/morris.html>

[9] M. Nagy and N. Nagy, Quantum Tic-Tac-Toe: A Genuine Probabilistic Approach, Appl. Math. **3**, 1779-1786 (2012).

[10] Noughts and Crosses.

Retrieved from <https://en.wikipedia.org/wiki/Tic-tac-toe>

[11] **To play our Quantum Tic-Tac-Toe game**, down load the Q3TGame.ipynb file of the game from here, and run the code in jupyter notebook, google colab or any other similar software:

Retrieved from <https://drive.google.com/file/d/1bYXSW6vsaH6rh9UbJnqUCIDIPjtaM5Ok/view>