Technological Institute of the Philippines

Quezon City Campus

Computer Engineering Department

Visualizing Quantum Superposition and Entanglement Using Qiskit

Final Project Report

In partial fulfillment of the requirements in
Emerging Technologies 1 (CPE 018)
1st semester, S.Y. 2024-2025

Engr. Dylan Josh Lopez

Members:

Cabos, Jhillian Millare

Catulay, Weslie Jee

Dela Cruz, Eugene

Garcia, John Carlos

Quibral, Juliann Vincent

Visualizing Quantum Superposition and Entanglement Using Qiskit

Cabos, Jhillian Millare, Catulay, Weslie Jee, Dela Cruz, Eugene, Garcia, John Carlos, Quibral, Juliann Vincent

College of Engineering and Architecture

Technological Institute of the Philippines

Abstract

This project presents a Quantum Circuit Simulator developed using Python and Qiskit. The simulator enables users to create and manipulate quantum circuits, apply quantum gates, and visualize the results through a graphical interface. It also includes a noise simulation feature to demonstrate the impact of imperfections on quantum operations. The tool provides an interactive platform for exploring quantum computing concepts and is designed to help users better understand quantum circuits and their behaviors.

I. Introduction

Quantum computing is a rapidly advancing field that holds the potential to solve complex problems that are difficult or impossible for classical computers. At the heart of quantum computing are quantum circuits, which manipulate qubits using quantum gates. These circuits use quantum properties like superposition and entanglement to perform calculations.

However, one of the biggest challenges in learning and experimenting with quantum circuits is the lack of accessible tools for beginners. Many people find it hard to visualize and understand how quantum circuits work, especially when learning the behavior of quantum gates and the effect of noise in quantum systems.

This project aims to address this problem by creating a Quantum Circuit Simulator using Qiskit. The simulator allows users to design quantum circuits, apply quantum gates, and visualize the outcomes of these operations. By offering an easy-to-use interface, it helps users better understand how quantum circuits function and allows them to experiment with basic quantum algorithms. This tool is designed to make quantum computing more accessible to learners and enthusiasts, providing a hands-on way to explore the fundamentals of quantum mechanics.

II. Review of Related Literature

One research direction focuses on improving quantum computing usability through innovative user interaction techniques. These include a circuit writer for generating code snippets from high-level circuit descriptions, a machine explorer for detailing hardware properties, a circuit viewer for comparing logical and compiled circuits, and a visualization tool for adjusting noisy measurement outcomes. Inspired by human-computer interaction (HCI) research, these tools address the complexities of quantum program design and hardware variability, enhancing existing notebook-based toolkits like Qiskit to support developers in navigating quantum computing tasks more intuitively.

This work introduces innovative user interaction techniques for quantum computing. It highlights four tools: a circuit writer for generating code from high-level circuit descriptions, a machine explorer for detailed hardware insights, a circuit viewer for comparing circuit configurations, and a visualization tool for adjusting measurement outcomes with noise. These tools aim to enhance notebook-based environments like Qiskit, addressing challenges in quantum program design and hardware variability. By improving usability and accessibility, this approach empowers developers to better navigate quantum computing workflows and leverage its potential across various applications.

III. Methodology

This outlines a structured approach to developing a Quantum Circuit Simulator, detailing each phase of the project from initiation to closure. It emphasizes the identification of objectives, gathering requirements, system design, development, testing, deployment, security measures, user documentation, and product orientation. The methodology ensures that the project focuses on quality, usability, and security throughout its lifecycle.

[2.1] Project Initiation:

a. Identify Objectives and Scope:

The goal of this project is to create a Quantum Circuit Simulator using Qiskit. The simulator will allow users to design quantum circuits, apply quantum gates, and visualize the results of these operations. The scope includes the development of an interactive user interface to visualize quantum gates and circuits.

b. Stakeholder Identification:

Stakeholders in this project include the development team, end users (Computer Engineering students), and external collaborators (such as quantum computing platforms like Qiskit). Third-party services like visualization libraries and simulation tools (Qiskit Aer, Matplotlib) are also considered.

c. Team Formation:

A cross-functional team is formed, with knowledge in quantum computing, Python programming, user interface design, and data visualization.

[2.2] Requirements Gathering:

a. User Needs Collection:

The primary user needs include the ability to create quantum circuits, apply quantum gates, visualize quantum states, and run simulations to view the effects of those operations. There will be an option to manipulate the circuit and update visualizations in real time.

b.Software Requirements:

The software tools required for this project include Qiskit for quantum computing, Tkinter for the user interface, Matplotlib for visualization, and Python as the primary programming language. No specialized hardware is needed as the simulator runs on standard computers.

[2.3] System Design:

a. Software Architecture:

The architecture will consist of a user interface built with Tkinter, which will communicate with Qiskit to build and simulate quantum circuits. Visualizations will be rendered using Matplotlib. A modular design will allow for easy integration of new features like different quantum gates.

b. User Interface Design:

The user interface will be designed to be simple and intuitive, allowing users to drag and drop gates into the quantum circuit, apply them, and instantly visualize the resulting quantum state on a Bloch sphere.

[2.4] Development:

a. Software Development:

The main development will focus on creating the quantum circuit simulator using Python. The development steps will include coding the backend logic for building quantum circuits, applying quantum gates, and running simulations. The front end will be developed using Tkinter to provide users with an interactive experience.

b. Visualization Integration:

Matplotlib will be used to render visualizations of quantum states and circuit diagrams, ensuring the results are easy to understand and accessible to users

[2.5] Testing:

a. Component Testing:

Individual components of the simulator, such as quantum gate application, state vector updates, and visualization rendering, will be tested for correctness.

b. Integration Testing:

Integration testing will ensure the seamless interaction between the user interface, Qiskit quantum circuits, and Matplotlib visualizations. The system will be tested with various quantum circuits to ensure accuracy in results and visualizations.

c. User Testing:

User testing will focus on the ease of use and accessibility of the simulator.

[2.6] Deployment:

a. Deployment of the Software:

The Quantum Circuit Simulator will be deployed as a standalone application that can be downloaded and run on user computers. Instructions for installation will be provided.

[2.7] User Documentation:

a. User Guide:

The user documentation will include a README.md file on GitHub that provides clear instructions on how to create quantum circuits, apply quantum gates, and visualize the results.

[2.11] Alternative Technology Integration:

a. Backup Strategy:

In case of issues with Qiskit or Matplotlib, alternative quantum computing libraries like Cirq or Quantum Development Kit (QDK) could be considered. Similarly, other visualization libraries such as Plotly can be explored as backups

[2.13] Project Closure:

a. Formal Closure:

Upon completion, The project will be closed, with documentation archived and future improvements outlined.

IV. RESULTS

The results show key points with screenshots and notes, making it easy to see the GUI's strengths and areas to improve.

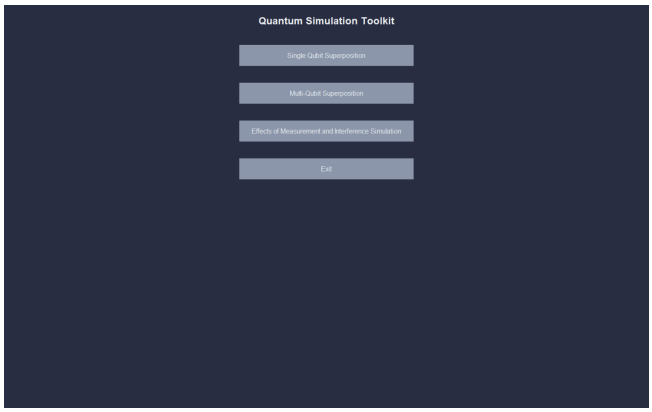

Figure 1: Main Page (Quantum Simulation Toolkit)

The user can explore options using the Quantum Simulation Toolkit interface.

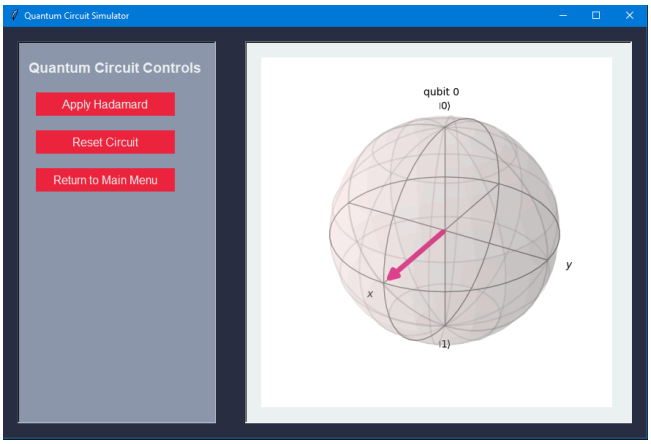

Figure 2.1 Single Qubit Superposition (Without Superposition)

This code visualizes a Bloch sphere representation of a single qubit's state.

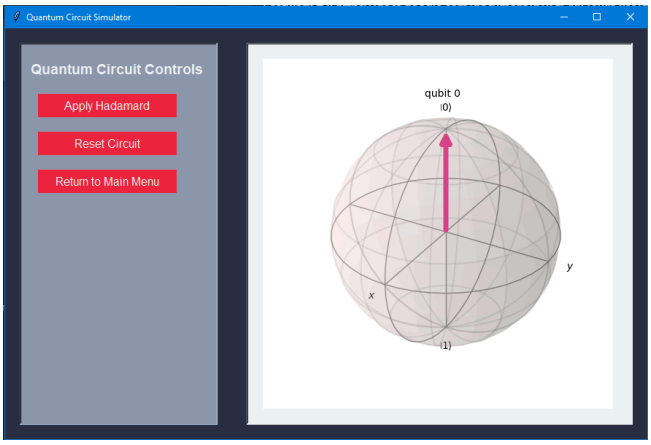

Figure 2.2 Single Qubit Superposition (With Superposition)

Upon clicking Apply Hadamard it will start transitioning from $|0\rangle$ to superposition using the Hadamard gate.

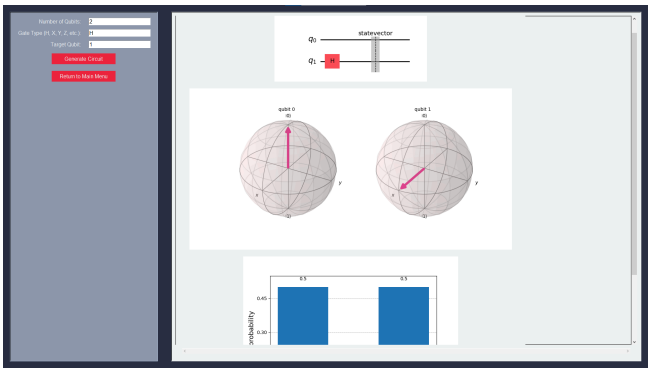

Figure 3.1: Multi-Qubit Output H

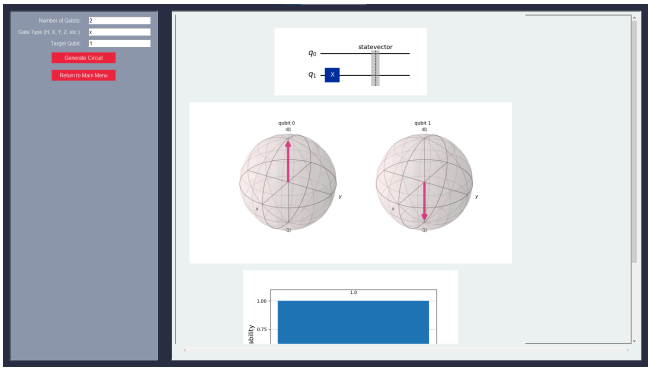

Figure 3.1: Multi-Qubit Output X

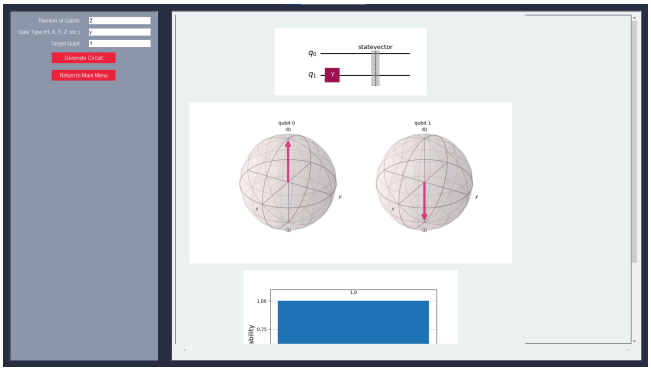

Figure 3.1: Multi-Qubit Output Y

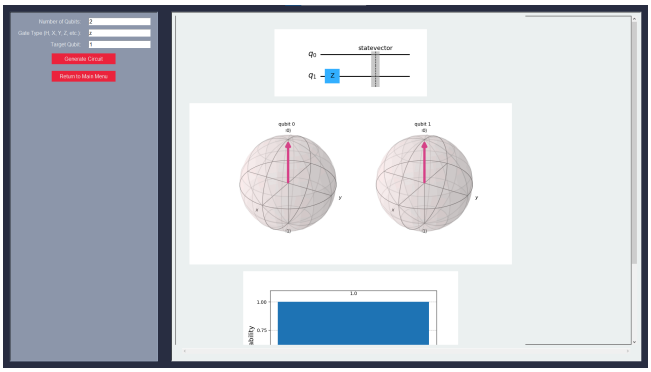

Figure 3.1: Multi-Qubit Output Z

This figure allows users to explore multi-qubit systems by selecting the number of qubits, the target qubit, and applying various gates such as H (Hadamard), X (Pauli-X),

Y (Pauli-Y), and Z (Pauli-Z). It visualizes the resulting quantum states, helping users understand the effects of different gate operations on multi-qubit systems.

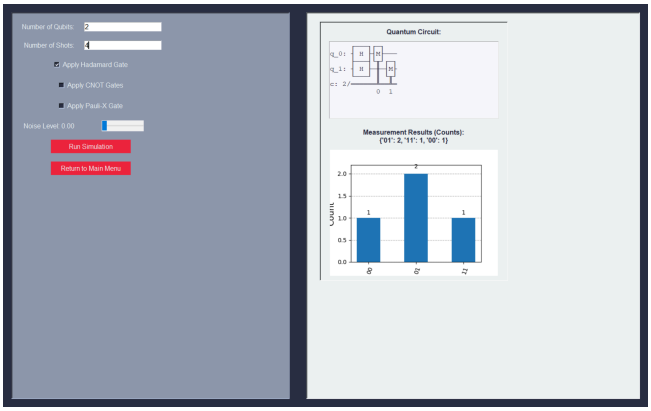

Figure 4.2 Multi-Qubit Superposition (Hadamard)

This figure demonstrates the application of the Hadamard gate on multiple qubits, creating an equal superposition state across all possible outcomes.

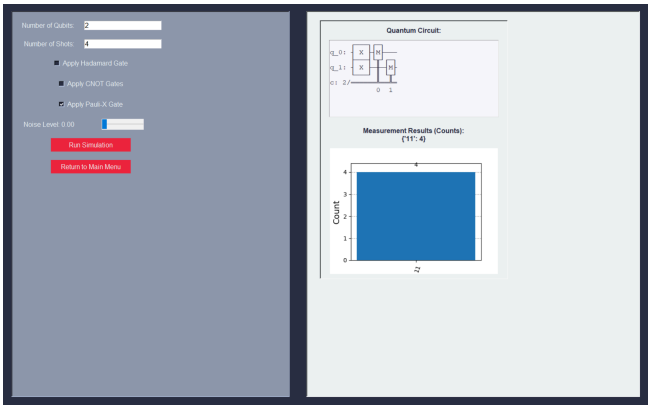

Figure 4.3 Multi-Qubit Superposition (Pauli-X)

This figure illustrates the use of the Pauli-X gate to flip the quantum state of selected qubits, showing its impact within a multi-qubit superposition.

Figure 4.4 Effect of measurement in Multi-Qubit Superposition (Hadamard)

This figure depicts the collapse of a multi-qubit superposition state created by the Hadamard gate into a classical state upon measurement, highlighting the probabilistic nature of quantum mechanics.

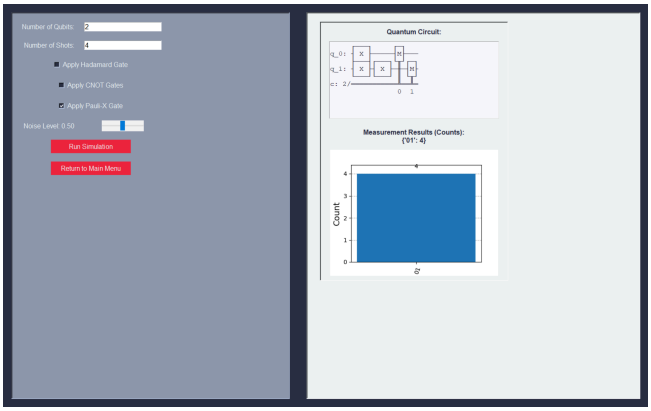

Figure 4.5 Effect of measurement in Multi-Qubit Superposition (Pauli-X)

This figure shows how the application of the Pauli-X gate followed by measurement affects the final state probabilities in a multi-qubit superposition.

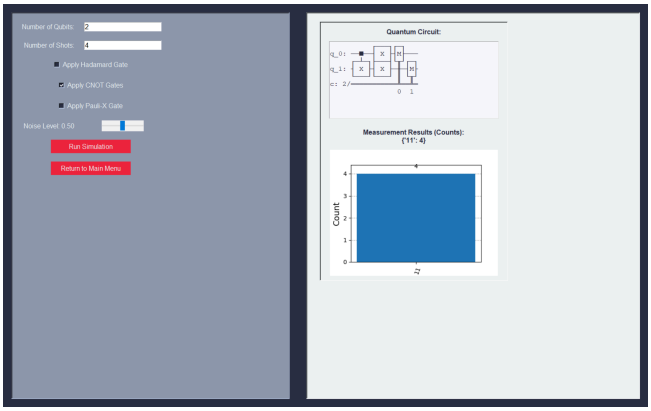

Figure 4.6 Effect of measurement in Multi-Qubit Superposition (CNOT)

This figure demonstrates the impact of the CNOT gate on a multi-qubit superposition state and the resulting measurement outcomes. It visualizes how the entanglement created by the CNOT gate influences the final probabilities, showcasing the interplay between control and target qubits in a quantum system.

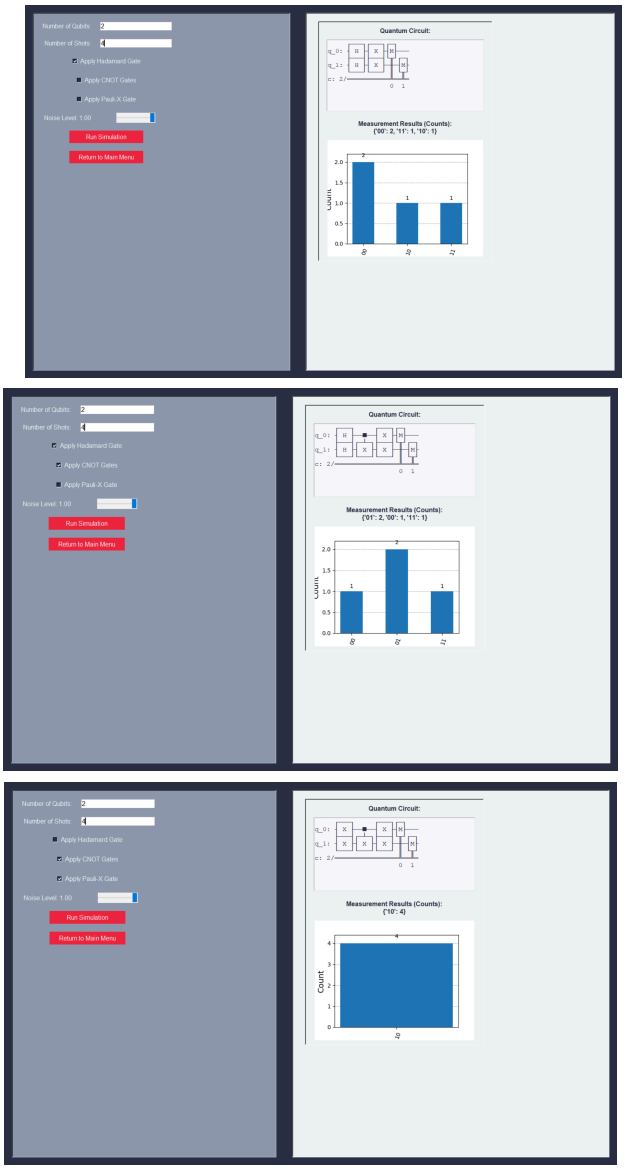

Figure 4.7 Effect of measurement in Multi-Qubit Superposition (Noise levels)

This figure illustrates how noise impacts the measurement outcomes of a multi-qubit superposition state. It highlights deviations from the ideal probabilities caused by hardware imperfections, such as gate errors, decoherence, and readout inaccuracies, showcasing the challenges in achieving precise quantum state control in noisy quantum computers.

V. Conclusion

This project successfully demonstrates the development of a Quantum Circuit Simulator using Python and Qiskit, providing an innovative, software-only solution for exploring quantum computing concepts. By enabling users to design quantum circuits, apply quantum gates, and visualize results through intuitive graphical tools, the simulator addresses key challenges in quantum computing education, such as the lack of accessible and beginner-friendly resources. Through the exclusive use of software tools such as Qiskit, the project achieves its goal of providing a robust and accessible platform for quantum computing enthusiasts and learners. The simulator effectively integrates features like a broad selection of quantum gates, real-time manipulation, noise simulation, and state visualization, offering a comprehensive and interactive environment to bridge the gap between theoretical learning and practical experimentation in quantum computing.

References

[1] Kim, H., & Smith, K. N. (2024, September 24). Interaction techniques for user-friendly interfaces for gate-based quantum computing. arXiv.org. https://arxiv.org/abs/2409.16475

[2] Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S., Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R., & Gambetta, J. M. (2024, May 14). Quantum computing with Qiskit. arXiv.org. https://arxiv.org/abs/2405.08810