Universiteit Leiden
The Netherlands

Opleiding Informatica

Quantum game development using Unitary

Ivor Brouwer

Supervisors:
Evert van Nieuwenburg & Mike Preuss

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

16/07/2024

Abstract

The accessibility of quantum game development for non-quantum experts is explored in this thesis, focusing on the Unitary library. We investigated by creating a puzzle game using Unitary that implements quantum principles such as superposition, entanglement, and measurement. In the game, players interact with quantum objects to solve levels, each representing a quantum circuit to be reverse-engineered. By progressively introducing more complex quantum concepts, we tested the difficulty of implementing these mechanics and documented insights on the process and potential improvements for accessibility. As Unitary aims to develop games that are compatible with quantum computers, we also tested this capability, finding that noise in current quantum hardware breaks game functionality. Our research reveals that non-quantum experts can indeed create quantum games using Unitary, as the library effectively abstracts underlying quantum mechanics. Basic concepts like superposition and quantum moves are made accessible, allowing users with minimal quantum knowledge to develop simple games. However, more complex concepts such as entanglement and phase require deeper understanding, challenging the effectiveness of simple abstractions. By sharing our results and providing insights on the current state, we aim to stimulate interest in quantum computing, quantum education, and contribute to improving the user-friendliness of future development tools.

Contents

1 Introduction

Quantum computing is a promising field that can solve certain computational problems faster than classical computers. Possible applications of quantum computing in gaming could be the introduction of quantum principles as game mechanics and to use a quantum computer to speed up certain algorithms in a game. But development of a quantum game that incorporates these ideas can be quite a difficult task if the developer has no expertise in quantum mechanics. These difficulties arise from the complexity of quantum concepts themselves and the logic for implementing them. As mentioned in a study conducted in 2020 [HBLM20] the knowledge on quantum for IT professionals is still far behind. This might pose a problem if quantum does become a bigger part of computing.

As quantum computing advanced, several quantum development frameworks have been created to simplify the process of programming quantum algorithms. These frameworks aim to provide accessible tools and resources for developers to make use of quantum computing. Some games, like Qrogue [AWW23], have been developed using such frameworks, educating players about quantum mechanics while being built entirely with a development kit, making it possible for certain computations to run on a quantum processor. However, working in such frameworks often requires extensive prior knowledge of quantum concepts since developers operate at a low level.

Quantum development will have to advance to a point where extensive low-level knowledge about quantum circuits/computers is not a necessity to make use of the possibilities of a quantum computer, especially if quantum processors become more standard. That is why this research will explore the current possibilities and what will need to change to achieve this goal by creating a quantum game.

1.1 The situation

Quantum computing is still developing and it can be a possible game changer for society [SWM+24]. A lot of attention is being drawn to the cracking of cryptography but there are many other applications for quantum. There are quantum algorithms that are proven to give a speed-up compared to their classic counterpart, many of these quantum algorithms can be found on quantum algorithm zoo [The24]. Right now a lot of people still do not understand the concepts of quantum, and a great tool to improve this could be games. Games excel at creating interest, awareness and understanding, all of which could significantly benefit the field of quantum computing.

To develop quantum programs, better workflows have already been developed. Quantum programs can currently be developed by using any quantum development kit, such as Qiskit [JATK+24] or Cirq [Dev24]. These kits already make it possible for people to make use of logic for qubit manipulation and execute the quantum circuits on a quantum processor. But creating a game using these development kits is still a complex task. One solution is to create a more user-friendly wrapper for quantum games, offering an even higher-level implementation. This is where Unitary comes in [AI24]. Unitary serves as a wrapper for the development kit Cirq implementing commonly used quantum gates and circuits to simplify the workflow. Its objective is to enable non-quantum experts to create quantum games that make use of the capabilities of quantum computers and create game mechanics based on quantum principles such as superposition, entanglement and

quantum gates/circuits.

We will explore whether it is possible for non-quantum experts to develop quantum games using Unitary. By creating a quantum game, we will assess Unitary's usability and effectiveness in simplifying quantum mechanics for game development. Our aim is to contribute insights into making quantum computing more accessible in the future, potentially increasing engagement with quantum computing and improving understanding of quantum concepts.

1.2 Thesis overview

This chapter contains the introduction; Section 2 defines the research question; Section 3 includes the definitions; Section 4 discusses related work; Section 5 describes the method of result extraction; Section 6; Section 7 describes the findings and Section 8 concludes.

For my bachelor thesis at LIACS, Evert van Nieuwenburg and the quantum AI group from LIACS(QPlai) led by Evert van Nieuwenburg helped supervise my thesis.

2 Research Question

We aim to add knowledge to the use of quantum computing in games and research the suitability of using a quantum development library by a non-quantum-expert for creating a quantum game. Which helps define the research question as:

"How well suited is Unitary in 2024 to create a quantum game by non-quantum-experts?"

2.1 Motivation

Games present an interesting way of exploring quantum computing applications and can help create interest. By investigating the use of the Unitary library, this research seeks to understand if current tools are accessible and practical for game developers who may not have any quantum computing knowledge.

2.2 Hypotheses

Below are the hypotheses for the research:

- Non-quantum-experts can use Unitary to create basic quantum games with minimal initial learning.
- As the complexity of the game increases, the need for a deeper understanding of quantum mechanics will become a significant challenge.

2.3 Potential impact

The research might create interest in quantum computing itself and its education, encourage the integration of quantum concepts in gaming, leading to new gameplay mechanics and add to the development of user-friendly quantum computing tools and libraries in the future.

3 Definitions

In this section important concepts that will be mentioned later on are explained in detail. These will be important to understand the goal and method of the thesis.

3.1 Qubits

Qubits [Qua24b], short for quantum bits, are the representation used for quantum information. Classical bits can only have one of two states (0 or 1), qubits can have both states simultaneously. This allows quantum computers to perform certain algorithms more efficiently than classical computers. For a qubit the state is written as $|0\rangle$ or $|1\rangle$.

Qubits always have 2 base states, but you can also have qudits. According to nLab[nLa23], qudits allow for d base states, where d is an integer greater than 2. Qudits can be used to have more information encoded in a single object, as they can represent more information, they potentially enable more efficient quantum algorithms.

3.2 Bloch sphere

The Bloch sphere [Gle05] is a geometric visualization of the state of a single qubit. Every point on the sphere represents a state a qubit can be in and some quantum gates (quantum gates will be explained in section 3.4) can be seen as a rotation on this sphere.

- The top represents the state $|0\rangle$.
- The bottom represents the state $|1\rangle$.
- Points on the surface of the sphere represent superposition states, with different points corresponding to different probabilities of measuring $|0\rangle$ or $|1\rangle$.
- The equator represents equal superposition states, such as those produced by the hadamard gate.

3.3 Quantum principles

When referring to the basic quantum principles, we often refer to superposition, entanglement and measurement. These principles are the basis for quantum computing and are important to defining what a quantum game is, which is further explained in section 3.6.

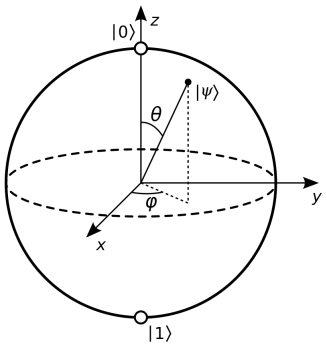

Figure 1: A geometric visualization of a Bloch sphere [Smi09].

Superposition
When a qubit is in a superposition it is simultaneously 0 and 1, which is written as $|0\rangle + |1\rangle$. This corresponds to any position on the Bloch sphere that is not one of the basis states.

Entanglement
Entanglement is a state qubits can be in where the state of one qubit depends on another qubit, for example when qubit 1 is equal to 1, qubit 2 will have to be 0. States of multiple qubits are written as $|10\rangle$, this is also how the earlier mentioned entangled state would be written.

Measurement
Measurement can be used to collapse these states into a single value. Measuring the state $|0\rangle + |1\rangle$ would collapse into either $|0\rangle$ or $|1\rangle$.

3.4 Quantum gates

Quantum gates [WW11] are used to manipulate qubits by changing the probability amplitudes of their quantum states. By doing this the quantum information a qubit has is changed. Below is a short description of quantum gates that will be used in the thesis.

3.4.1 Basic gates

These gates manipulate the state of one qubit. All of these gates also have a parameterized version where the amount of rotation can be adjusted, but often full rotations are used.

- **X Gate (NOT Gate):** Flips the state of a qubit, around the X-axis of the Bloch sphere, transforming $|0\rangle$ to $|1\rangle$ and vice versa.
- **Y Gate:** Rotates the state of a qubit around the Y-axis of the Bloch sphere, changing $|0\rangle$ to $i|1\rangle$ and vice versa.
- **Z Gate:** Rotates the state of a qubit around the Z-axis of the Bloch sphere, leaves the basis state $|0\rangle$ unchanged and changing $|1\rangle$ to $-|1\rangle$ and vice versa.
- **Hadamard Gate:** Places a qubit in a superposition state, resulting in a balanced mix of $|0\rangle$ and $|1\rangle$.

3.4.2 Multi-qubit/Control gates

Multi-qubit gates operate on multiple qubits. Control gates are multi-qubit gates that operate as a sort of conditional statement. If a qubit's state is equal to $|1\rangle$ a gate will be applied to another qubit. Below is a description of the controlled-NOT gate, but a control can be combined with any other gate.

- **CNOT Gate (Controlled-NOT):** Performs a NOT operation on the target qubit only if the control qubit is in the $|1\rangle$ state.
- **CHAD Gate (Controlled-Hadamard):** Performs a hadamard operation on the target qubit if the control qubit is in the $|1\rangle$ state.
- **SWAP Gate:** Exchanges the states of two qubits.

3.5 Unitary

Unitary [AI24] is open-source software which aims to make quantum game development accessible for non-quantum experts. Unitary has been in development since March 2020 and is still regularly updated. It is built on Cirq, which is a development framework created by Google for managing qubit states and doing quantum computations. These computations can be simulated or can be run on a quantum computer/processor. Cirq code is written in Python, this code is then transpiled into Open Quantum Assembly. Unitary is an abstraction layer on top of Cirq and implements common functionalities such as quantum objects using quantum movement.

3.6 Quantum games

According to Kiedos.ART, quantum games are "any type of games that connect to quantum physics by referencing quantum physical concepts, by numerically simulating quantum physical phenomena, or are running on quantum computers" [PH24]. These quantum games can be further divided into categories such as: those designed for research purposes or education and those created specifically for entertainment. A more detailed overview is provided in the paper "Defining Quantum Games" [PPW+22], which offers an elaborate description of quantum games and their history.

4 Related Work

In the related work section, relevant literature will be highlighted that has explored the fields of quantum game research, education and development. The overview will show where the thesis fits in.

4.1 Quantum game research

Quantum game research has been conducted for some time, but can be called a relatively new field. One of the first mentions of quantum games in research is "Quantum Games and Quantum Strategies" by [EWL99] in 1999. It is proven here that for the quantum prisoners' dilemma, there is a strategy

that is always beneficial to both players. This research was primarily based on quantum game theory.

Recent research, such as "Quantum games and interactive tools for quantum technologies outreach and education" by Seskir et al. (2022), has shifted focus towards practical applications [SMW+22]. This study aims to create guidelines for educational material to improve the accessibility of quantum concepts using quantum games and interactive tools. This shows the increasing focus of practical educational tools in quantum games. The paper highlights the use of games and platforms such as Hello Quantum [hel18], Particle in a Box [par], Quantum Moves [qua], Quantum Flytrap [Qua20a] and Quantum Odyssey [Qua20b] for educational purposes and the increasing funding and interest in the field.

Quantum education is often the focus of new papers in the field as it seeks to make complex quantum concepts more accessible and intuitive for new learners. For example, Chiofalo et al. (2022) conducted a study with high school students to evaluate the effectiveness of teaching quantum mechanics through quantum games [CFM+22]. The study found that students were able to grasp difficult concepts like superposition and entanglement more effectively through gameplay than through traditional teaching methods. This demonstrates the potential of quantum games as educational tools that create interest and enhance understanding of quantum principles.

4.2 Quantum game development

To improve accessibility to quantum computing, various quantum development kits have been created. The most notable development kits that are currently being updated include Qiskit, ProjectQ, Q#, and Cirq. These kits are commonly used to develop quantum games, offering essential tools and documentation to support developers.

In the paper "Comparing Quantum Software Development Kits for Introductory Level Education" [SY22], a detailed comparison of these quantum development kits is provided, highlighting their strengths and weaknesses. It was found that while all kits offer good documentation, the biggest hurdle is the visualization of quantum circuits, which is crucial for educational purposes. Effective visualization aids in understanding complex quantum concepts, making these tools more accessible to beginners.

The first quantum game created using one of these development kits was created in 2017 by Dr. James Wootton [Woo17]. It was a quantum version of rock-paper-scissors, developed using the ProjectQ kit. This game is playable on a quantum computer at IBM via their API, demonstrating the practical application of quantum computing in game development.

Since then, quantum game jams have become a yearly event in the quantum computing community [KPJ21]. These events aim to garner attention for the creation of quantum games among quantum physicists and enthusiasts. Quantum game jams provide a platform for developers to expand the landscape of quantum games, which helps to show the importance of engaging formats for teaching quantum computing.

The focus on education and generating interest in the field shows that the major challenges in quantum computing is the steep learning curve associated with quantum mechanics and the need for more people to be introduced to the field. This shows that user-friendly quantum development tools are important to advance the field further.

5 Method

To answer the research question, we investigated the possibilities of Unitary, assessed the necessary knowledge to create a quantum game and gathered information on other quantum games and their development resources. Below the objectives are explained in more detail.

O1. Quantum Games and Prototyping: First off, a literature review was done on existing quantum games and how they were made. How we can use Unitary to create a game that can be tested on a quantum processor and uses quantum principles. Multiple kinds of quantum game mechanics were brainstormed, but one idea was chosen as the prototype for a quantum game that would be created.

O2. Quantum Game Development: After creating a prototype, development started on the quantum game using Unitary. This gave insight into the possibilities and restrictions of creating a quantum game with Unitary. Creating the game also gave insight into the possibility to create a quantum game for by non-quantum-expert, as I myself am not a quantum expert.

O3. Quantum Mechanics Understanding: An assessment of the level of quantum mechanics knowledge required to effectively use Unitary was done while developing. This included understanding fundamental quantum principles such as superposition, entanglement, and quantum gates/circuits, and how they translate into game mechanics.

These objectives helped reach the primary goals of the research:

- To evaluate the accessibility of the Unitary library for non-quantum-experts.
- To measure the quantum understanding necessary for developing with Unitary.
- To assess the capabilities and limitations of Unitary for creating a quantum game.

6 The Quantum Game

After we researched quantum games, a decision was made to create a quantum puzzle game. The primary objective in the game is to teach players basic quantum concepts while evaluating the functionality and accessibility of Unitary for quantum game development.

6.1 Game concept

The game is designed as an educational tool to introduce players to the fundamental concepts of quantum mechanics. Players navigate through puzzles where they must manipulate quantum objects using quantum gates to progress from the start to the end of each level. These gates, provided at the beginning or found within the level, represent quantum operations that players must use strategically.

Each puzzle corresponds to a quantum circuit executable on a quantum computer. Players reverse-engineer these circuits to solve the puzzles, thereby learning about quantum principles through interactive gameplay. This approach allows for a logical progression from simple to more complex quantum concepts for the player, while providing insights into the quantum knowledge necessary for implementing these concepts using Unitary.

6.2 Game design

For the design a very simple 2D look was chosen, since the focus is on the integration of quantum based mechanics into the game and not on the visuals. The quantum objects which block the player from moving to the end are represented as pillars. And at the start tile the player spawns as a character.

Figure 2: From left to right the pillars represent $|1\rangle$, $|0\rangle + |1\rangle$, $|0\rangle$ - $|1\rangle$, pure $|0\rangle$ and high probability $|0\rangle$.

In Figure 2 the representations can be seen of the basis states and two superpositions of the pillars. These are the pillars that will have to be manipulated by the player to reach the end. The player can only move through pillars in the pure $|0\rangle$ state. Blue represents the probability of a pillar being $|1\rangle$, red represents the probability of a pillar being $|0\rangle$, this is also why a superposition with no phase is purple, this was done for clarity as slight changes in the blue color are hard to see. The phase is represented by the green color that is why $|0\rangle + |1\rangle$ is purple mixed with green. For clarity pillars that will always collapse to $|0\rangle$ are represented as transparent white.

Figure 3: Basic boxes that can be found in the game which provide the player with a gate.

Gates on the ground are visualized as boxes on the ground with the corresponding name of the gate on top of it. When a player moves to the tile that has the box on it the box disappears and the corresponding gate is added to the player's inventory. In Figure 3 some boxes can be seen. To use a gate on a pillar, the player has to be next to it.

Figure 4: Entanglement visualization using a line.

When pillars are entangled, which can be caused by a player using a multi-qubit control gate on the pillars, a white line between the pillars which can be seen in Figure 4 is used to represent the entanglement of these pillars. This is only shown when the player moves their cursor over the entangled pillars to reduce cluttering.

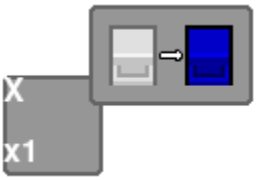

Figure 5: Every gate shows a hint of what the gate does to a pillar.

To help the player understand what each gate does to a pillar there is a hint when a player moves their cursor over the quantum gates in their inventory as can be seen in figure 5. This hint shows the player what the gate would do when applied to a gate in the $|0\rangle$ state, this way players still have to figure out that the gates are unitary and what they do to pillars in different states.

6.3 Development

The game is developed using Unitary for most of the quantum logic. Since Unitary is completely written in Python it was a simple choice to visualize the game using Pygame [Shi11]. Pygame is a game engine for prototyping and creating simple 2D games. The sprites that are used in the game are based on sprites made by Kenney Assets [Ken24]. The game is compatible with python version 3.8 and higher.

6.4 Puzzles

The first puzzles are based on simply teaching the player what every gate does to a pillar. The first level starts with introducing the player to a pillar in the $|1\rangle$ state, to complete the level the player will have to walk up to the pillar and use the x-gate in their inventory. The level can be seen in Figure 6.
The levels after have variations of this level with other single qubit gates such as the z-gate and the h-gate to teach the player how to overcome these obstacles. When these levels have been completed this knowledge will be put together to overcome a more complicated puzzle as seen in Figure 7.

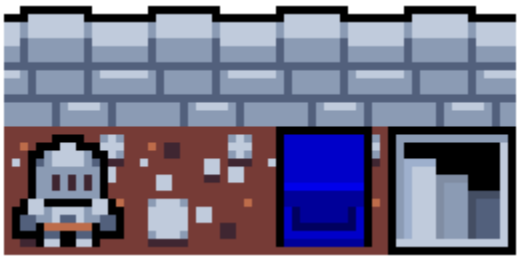

Figure 6: Level 1 of Qungeon, player on the left has to walk up to the pillar and use an x-gate to reach the stairs to the next level.

Figure 7: Level 6 introduces a fun and more complicated puzzle involving all the previously shown concepts.

Multi-qubit control gates are introduced last since these are visually complicated and can be quite confusing to get used to. First there is an introductory level which shows what a CNOT gate can do. The puzzles encountered can be seen below and are based on the interesting quantum states called the Greenberger–Horne–Zeilinger-state in Figure 8 and the W-state in Figure 9.

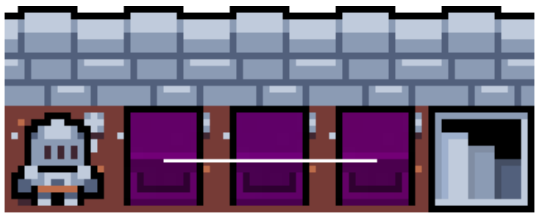

Figure 8: Puzzle corresponding to the Greenberger–Horne–Zeilinger-state.

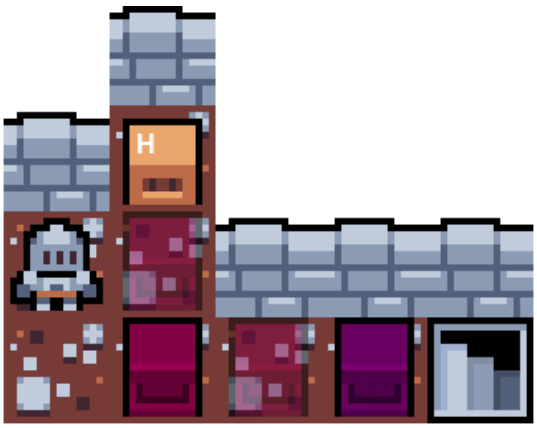

Figure 9: The puzzle corresponding to the W-state and an extra hadamard.

To complete the W-state level the qubits in the quantum circuit in Figure 10 have to be changed to the $|0\rangle$ state. As the player reverses the entangled qubits they can pick up the hadamard-gate and reach the end.

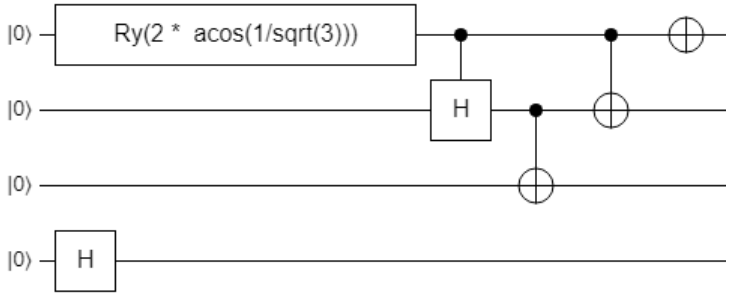

Figure 10: The quantum-circuit corresponding to the W-state, with extra hadamard.

7 Results

Below is an overview of the results found while reviewing Unitary. It starts with a full assessment of the functionality, then the documentation, assesses the amount of quantum knowledge necessary and suggestions for adding to the accessibility.

The functionality of Unitary can be separated into multiple parts. For each of these parts, there will be a separate section discussing the functionality, documentation, quantum understanding necessary, and what would help with the accessibility.

7.1 Quantum object

The Quantum Object class represents any object that will have a quantum state. These objects can be seen as a qubit or qudit (which can be chosen at initialization) in quantum computing.

7.1.1 Functionality

There is functionality to create a quantum object, which gets initialized with an effect automatically. A name will have to be given to later access the quantum object from the quantum world. An initial state can also be set for the object, which will be applied when it is added to the quantum world. There are also arithmetic functions such as `__iadd__` (cycles through the states) and `__neg__` (flips to the next state), but these functions can cause the code to become quite obscure and are better left alone to be used internally by Unitary.

7.1.2 Documentation

The quantum object is very straightforward and shows a clear way in which it needs to be used. There is no documentation for the `initial_effect`, `__iadd__` and `__neg__` functions, but these functions should not be explicitly used by users as they can make the code quite obscure. Functions used exclusively internally by Unitary might benefit from this being mentioned in the doc-string, this way users will know to not use these functions.

7.1.3 Quantum understanding

Making use of the quantum object is great, it abstracts away from how a qubit works and simply gives the user a straightforward way of creating a quantum object with which they can interact. No quantum knowledge is necessary for this step, by using enums a user can simply show which states are possible for a quantum object to have.

7.1.4 Suggestions

Printing Functionality
In addition to its current functionality, the quantum object class could benefit from a more informative description for when it is printed. Right now, it shows the object type and memory location but a user could benefit from seeing its name and potentially a probability measurement of its current state distribution.

Object exporting
Additionally, a standardized `__reduce__` method for pickling the class data might be helpful for game creation, this would enable people to very simply save and load objects in their game.

7.2 Quantum world

The quantum world class represents the entire state of a quantum game. It holds all the information on quantum objects and the effects applied to these objects. Compared to quantum computing on a lower level, the quantum world can be seen as a representation of a quantum circuit but with additional functionality. The additional functionalities include keeping a history so effects can be undone, performing measurements and choosing the type of simulation.

7.2.1 Functionality

To make use of effects on a quantum object it has to be added to the quantum world using the `add_object` function. After this the quantum world automatically updates the circuit based on effects applied to the quantum objects.

There is also functionality to load older snapshots of the quantum world, this is great if a user wants to reset one of their moves. There is also a possibility to reset the whole quantum world with the `clear` function. These functions are great since they implement common features that would otherwise have to be implemented by every programmer. Quantum worlds can also be combined by using the `combine_with` function.

There are multiple functions to gain information about the state of the quantum world. This can be done by using functions that return histograms for groups of quantum objects, correlated histograms for entanglement and probability distributions. All of these functions are based on the `peek` function that non-destructively (does not cause the quantum object to collapse) measures the quantum object's value. By doing this x amount of times a distribution can be created. This can be used to for example visualize quantum object states. It is also possible to destructively measure a quantum object by using force measurement. The pop function can be used to do force measurement on multiple quantum objects.

7.2.2 Documentation

Every function includes a description of what the function does except for `pop`. Some of these functions are used internally by Unitary, it might help users to understand which functions they should use while developing by mentioning this in the doc-string.

7.2.3 Quantum understanding

To make use of the basic functionality of the quantum world, such as adding in objects, measurement and changing the state of the world; only a low level of quantum understanding is necessary. It does get more complex when ancilla qubits are mentioned, This concept will require users to get more familiar with quantum theory and it might help to have a brief explanation on what they do and what they are used for.

7.2.4 Suggestions

Printing Functionality
Currently when printing the quantum world it only shows its class and memory location, it could benefit from showing its current circuit. This is already possible by printing a Cirq circuit, this function can be used to create this functionality. Other information could also be shown when outputting.

Phase measurement
Right now we can measure in the z-direction to get the end state of the qubits. But for visualization

of the quantum objects in the game it might be very valuable to have the possibility to also measure the y-direction, or in short the phase. This would make it possible to visualize the difference between $|0\rangle - |1\rangle$ and $|0\rangle + |1\rangle$. For Qungeon this is done by updating a boolean whenever a phase gate is applied.

Object removal
Removing a quantum object from a quantum world can be a very important feature for certain quantum games. This can be done by using the unhook function, but the function name is not very descriptive and the provided explanation is not easy to understand for a non-quantum expert. Changing the name to `remove_object` might make it easier to understand, but does not correctly represent what actually happens in quantum. Describing that it removes a quantum object from the quantum world might improve accessibility.

Entangled state measurement
When measuring entangled objects with the `correlated_histogram` function the full quantum world state is returned. To only get the state of the entangled objects the objects that are entangled will have to be managed by the programmer themselves. One way to do this is by giving every quantum object their own quantum world and then use the `combine_with` function, but this increases the overhead and is not the best way to make use of the quantum world class. To fix this in Qungeon, I created a grouping system that holds the information on these entangled groups. Adding in the feature to measure an entangled object with a new function which then returns the entangled group states might simplify the usage of Unitary.

World exporting
Having a standardized `__reduce__` method would also work well for a quantum world. This way it would become very simple to load and save full levels and their quantum states.

Ordered correlated histogram
A tiny improvement would be ordering the returned values of the correlated histogram. Right now they get returned in a random order. A user could order the data themselves, but having it in an ordered dictionary from the start improves readability and accessibility.

7.3 Quantum effect

With quantum effects various operations can be applied to the quantum objects. These effects manipulate the quantum state of the object. The names are chosen to slightly abstract away from their quantum gate equivalent, the names correlate with actions often performed in tile based games.

7.3.1 Functionality

Every quantum object in Unitary can be done in fractions, this allows anyone to implement other rotations than just full rotations. The amount of quantum objects necessary for the effect is different per effect.

Move
Move is a good descriptive name and is used to perform quantum moves, this will be used by a player when the player wants to move a quantum object to another grid. Its actual functionality works as a quantum SWAP-gate and will move a quantum object state into another object.

Superposition
As the name suggests, it is used to put a quantum object in a superposition, implemented using a H-gate.

Split
Split will split a quantum object exactly between two quantum objects given. This can be used by a player as a precise move to two quantum objects.

Flip
Flip can be used to flip a quantum object around the x-axis, corresponding to an X-gate(NOT-gate).

Phase
Corresponds to using a Z-gate. A game creator would be able to create new game mechanics using this gate. There is also a PhasedMove and PhasedSplit version, which use ISWAP to be implemented. They correspond with Move and Split but also add a phase effect.

7.3.2 Documentation

Documentation on these effects are great, it is simple enough to understand what each effect does. The creators did a great job at explaining what the effects do without going too deep into how they work exactly in quantum computing. Effects such as phase have a more elaborate description since this is not as easy to translate into a simple game mechanic.

7.3.3 Quantum understanding

For non-quantum experts using the effects is quite simple, this is great for accessibility. Move, superposition, split and flip are all very descriptive names which explain what they do and when they can be used. Phase is something more complicated and would require someone to get more familiar with phase.

Qungeon was designed with static quantum objects, which required me to get more familiar with quantum computing since most simple effects in Unitary are designed with quantum movement in mind. If you design the game with only superpositions and quantum movement you won't have to interact with more complex quantum gates and the quantum understanding necessary will be minimal.

7.3.4 Suggestions

FlipPhase implementation
For Qungeon the Y-gate was missing, this would be a small addition but adds the possibility to

make use of the FlipPhase effect in Unitary. It might cause more confusion for new users on what this effect achieves.

Game descriptions
Something that might help people understand what a certain gate does is to have a simple example of what happens to the states of the qubits after using an effect. A user could test them out but it might help them understand what can be done with each effect when reading the function. To explain the move we could add gamified descriptions as seen in 10. Additionally mentioning the corresponding gate in the documentation might help users research the effect. This could also be added to documentation in the repository.

```
Move Example:
Initial State:
- position A: Character 1
- position B: Character 2

Applying Move:
- position A: Character 2
- position B: Character 1

Phase Example:
Initial state:
- Character: Dead

Applying superposition:
- Character: Dead + Alive

Applying phase:
- Character: Dead - Alive

Applying superposition:
- Character: Alive
```

Figure 11: Examples of quantum effects and how they would work in a gamified context.

7.4 Quantum if

The quantum if class allows the implementation of control effects. These can be seen as conditional effects and are appropriately called quantum if. When a quantum object is in a superposition and a control gate is used an entangled state can be created. Using the quantum if any control gates can be created.

7.4.1 Functionality

By using `quantum_if(qubit).equals(state).apply(effect)(on_qubits)` a programmer can simply use the quantum if to create an entangled state. The other functionalities such as the QuantumThen which checks the condition and applies it if true won't explicitly be interacted with by the programmer.

7.4.2 Documentation

The documentation for the quantum if describes the class clearly and includes a great example on how to use it in any game.

7.4.3 Quantum understanding

The quantum if does not require any prior quantum understanding, even if a user has never heard of a control-gate or entanglement the quantum if clearly describes what can be done with it. However, creating gameplay mechanics around entanglement might prove complex without understanding how it works in quantum.

7.4.4 Suggestions

The quantum if is very well implemented, there is not much to suggest as it is made as accessible as possible. Only suggestion that could be done was to still include a gamified example to show what effect could be achieved by making use of the quantum if.

7.5 Sparse vector simulator

For the quantum simulation Unitary uses a noiseless sparse vector simulator by default. This simulator offers a significant advantage in memory usage and computational efficiency for simulating large and sparse quantum systems compared to the Cirq simulator. This is done by only storing and manipulating the non-zero or significant amplitudes. Most will make use of this default simulator but it is possible to make use of any simulator compatible with Cirq.

7.6 Wiki and Overall Accessibility

To improve accessibility for new users of Unitary, adding a GitHub wiki could serve as a centralized resource, making it easier for users to locate relevant information and find examples on how to use functions and incorporate quantum mechanics into their games. Currently, information is scattered across the docs folder and the Unitary alpha README, requiring users to look through the code to find details, including internal functions like `add_effect` in the quantum world class, which they should not explicitly use.

The docs folder of the Unitary repository contains files such as `game_design.md`, `overview.md`, `physics_terms.md`, and `getting_started.ipynb`. These documents provide valuable insights into designing games with Unitary and offer an overview of its functionalities. Incorporating these files into the GitHub wiki and mentioning the documentation folder in the README would guide users

to essential resources more effectively. Although there used to be a dedicated page for Unitary documentation on the Cirq website, having a dedicated webpage or wiki for Unitary again would simplify working with the library.

Unitary's implementation in Python ensures accessibility for individuals with basic programming experience, enabling them to run simulations and develop games, particularly using prototyping frameworks like Pygame. However, Python's capabilities may be limited for creating complex games. Integrating Unitary with popular game engines such as Unity or Godot could broaden its accessibility, allowing more developers to incorporate quantum simulations into their projects.

Currently, Unitary includes example games in its directory, aiding users in familiarizing themselves with quantum concepts. The Quantum Chess example is an excellent resource with well-documented features, introducing fundamental concepts such as moves and superposition and progressing to advanced topics like entanglement. This example, mentioned in the README, serves as a valuable starting point for new users.

While developing we encountered difficulties due to overlooking these documentations on game design with Unitary, resulting in a game not fully aligned with the library's intended use. This further complicated the implementation for a non-quantum expert as more understanding of quantum concepts was necessary to implement phase and entanglement in a game where you need to reverse a quantum circuit. This oversight also made the game reliant on classical computations to gather information on a qubit. Currently in Qungeon, classical computations are necessary to visualize the phase of quantum objects. This is also not in line with Unitary as it is primarily designed to run created games entirely on a quantum computer. This experience underscores the importance of easily accessible documentation and starting guides, such as a wiki or dedicated website.

7.7 Running on a quantum computer

Since the game uses Cirq the probability distribution for the pillars in the game can be calculated by a quantum computer. To run the quantum game on a quantum computer Quantum Inspire [qua24a] was used. Since Quantum Inspire circuits have to be written in cQASM instead of openQASM, which is used by Cirq, a small translation had to be done. The quantum circuit was tested on the Starmon-5 quantum computer, this computer has 5 qubits in total. Only the first level was tested.

In Figure 12 the start of level 1 can be seen where the probability distribution for the pillar is calculated by a quantum computer. In level 1 the pillar should be 100% $|1\rangle$ at the start which is represented by being completely blue. There does not seem to be anything off on first glance, but this is because the blue color of the pillar depends on probability that the state is $|1\rangle$ and the red color depends on the probability that the state is $|0\rangle$. The error is so small that the noise is not noticeable in the color. When using an X-gate on the pillar the state of the game changes to the state that can be seen in Figure 13. The red color of the pillar indicates that it has a high probability to be $|0\rangle$ but that it is not 100% $|0\rangle$, therefore the player will not be able to move through it.

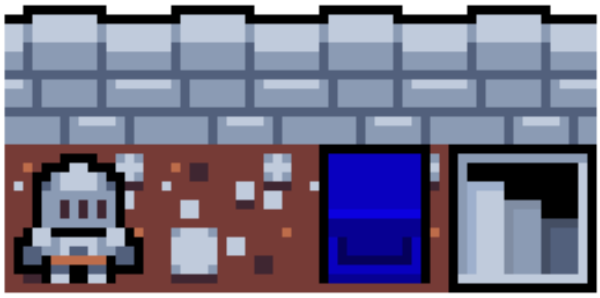

Figure 12: Level 1 ran on a quantum computer.

Figure 13: Level 1 after using an X-gate on the pillar.

Iteration	Probability Distribution Before		Probability Distribution After	
	1	0	1	0
1	0.962890625	0.037109375	0.0263671875	0.9736328125
2	0.96484375	0.03515625	0.021484375	0.978515625
3	0.9521484375	0.0478515625	0.0185546875	0.9814453125
4	0.9638671875	0.0361328125	0.021484375	0.978515625
5	0.9638671875	0.0361328125	0.0263671875	0.9736328125
Mean	0.0384765625		0.0228515625	
Std Dev	0.00472802		0.00306336	

Table 1: Probability distributions before and after applying the X-gate.

When checking the probability distribution of the pillar we get the distributions seen in Figure 1. In the distributions it can be seen that the error is always below 5%, this is why it is not noticeable in the color of the pillars. An expectation was that the error would increase even more after applying the X-gate as this would make the circuit larger, but the error and standard deviation of the error both became significantly smaller. This might have to do with the small sample size or there is better error mitigation when running a larger circuit. Below the t-test is shown calculating the significance of the difference in quantum error.

- **calculations:**

$$\text{Standard error of difference} = SE = \sqrt{\frac{s^2_{\text{before}}}{n} + \frac{s^2_{\text{after}}}{n}} = 0.003$$

$$\text{t-statistic} = t = \frac{\bar{X}_{\text{before}} - \bar{X}_{\text{after}}}{SE} = 6.2017$$

$$\text{degrees of freedom (df)} = 8$$

- **Confidence interval:**
  The mean of before minus after equals 0.015625.
  95% confidence interval of this difference: From 0.009815 to 0.021435.

- **Statistical significance:**
  The two-tailed P value equals 0.0003.
  This difference is considered to be statistically significant.

In the end, the game does become unplayable. One way to fix this might be to add error correction in my game itself. Instead of only letting a player move through a pillar when it is 100% in the $|0\rangle$ state, a small tolerance could be added, allowing movement through the pillar when it is in a state close to the $|0\rangle$ state, reducing the impact of quantum errors on gameplay. But this might only fix level 1 as other levels might have different quantum noise. On top of that, the issue of slow gameplay would remain since doing calculations on a quantum computer through an API currently takes quite some time.

8 Conclusions and Further Research

In conclusion, it is possible for a non-quantum expert to create a quantum game using Unitary, making Unitary highly suitable to create a quantum game in 2024. The platform makes basic concepts of quantum computing accessible by using descriptive names for quantum, which abstract away the underlying quantum mechanics. The quantum world/objects representation within Unitary effectively describes quantum circuits/qubits, offering a straightforward class interface for non-quantum experts. These implementations ensure that minimal quantum knowledge is necessary to create a simple quantum game that makes use of basic quantum concepts such as superposition and quantum moves.

As deeper quantum concepts such as entanglement and phase are introduced, the need for a more thorough understanding of quantum mechanics is necessary. Unitary does a great job at abstracting these concepts but introducing them as a game mechanic might prove complex without knowing the actual quantum concepts. Although Unitary offers documentation and information on quantum computing and game development, expanding the documentation and creating additional entry points for information, such as a wiki or a dedicated website, would significantly benefit users.

The possibility to run circuits on a quantum computer is also an excellent way to increase interest in the field and demonstrate the potential for integrating quantum calculations into gaming and other applications.

For further research several aspects can be improved or expanded upon, such as:

- **Implementation of Suggestions:** The suggestions outlined in the results section can be implemented through a fork in GitHub, allowing the creators of Unitary to selectively add these improvements. This could potentially enhance the accessibility of the platform.
- **Expanding Qungeon:** There is potential for further development of Qungeon. Future work could include adding more complex quantum concepts to the game, such as puzzles that incorporate quantum oracles. Additionally, integrating AI that utilizes quantum moves and superposition could make the game more dynamic and challenging instead of static as currently quantum movement is not used. This could require players to evade these AIs and improve their understanding of the probabilistic nature of quantum.

- **Exploring the complexity of Qungeon:** The game mechanic that a player has to be next to a quantum object could prove to influence the complexity of reverse engineering a quantum circuit, it could prove interesting to explore what the actual impact of this mechanic is.
- **Porting Unitary to game engines:** Creating an extension or version of Unitary for popular game engines such as Unity or Godot engine could make it possible to create quantum games without having to program. This could widen the reach even more as there are more game designers used to making games with these engines and might help with creating better visualization of complicated quantum states.
- **Testing Unitary and Cirq Limitations:** Other research could focus on the current limitations of Unitary and Cirq. For instance, determining the maximum number of quantum objects that can be added to a game world would be valuable. This limitation might be influenced by large entangled states and could limit the size of games that can be created on simulated quantum circuits. However, these limitations might be overcome when actual quantum processors become less noisy and have more qubits.
- **Educational Studies with Qungeon:** Conducting studies on Qungeon's impact on quantum understanding could provide interesting insights. These studies could involve various groups, including computer scientists, high-school students, non-quantum experts, and quantum experts, to evaluate what they learn from the quantum game. Such research could highlight the educational benefits and identify areas for improvement in teaching quantum concepts through gaming.

By addressing these points, future research can enhance the accessibility of quantum game development and contribute to overall better quantum understanding, improved quantum education and more engagement in quantum computing.

References

[AI24] Google Quantum AI. Unitary. https://github.com/quantumlib/unitary, 2024.

[AWW23] Michael Artner, Guenter Wallner, and Robert Wille. Introducing qrogue: Teaching quantum computing using a rogue-like game concept. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, FDG '23, New York, NY, USA, 2023. Association for Computing Machinery.

[CFM+22] Maria Luisa Chiofalo, Caterina Foti, Marisa Michelini, Lorenzo Santi, and Alberto Stefanel. Games for teaching/learning quantum mechanics: a pilot study with high-school students. *Education Sciences*, 12(7):446, 2022.

[Dev24] Cirq Developers. Cirq, May 2024.

[EWL99] Jens Eisert, Martin Wilkens, and Maciej Lewenstein. Quantum games and quantum strategies. *Phys. Rev. Lett.*, 83:3077–3080, Oct 1999.

[Gle05] Ian Glendinning. The bloch sphere. In *QIA Meeting*, pages 3–18, 2005.

[HBLM20] Isabell Heider, Harald Bendl, Jan-Rainer Lahmann, and Frauke Mörike. Approaching quantum entanglement developing a serious game in quantum computing for it professionals. In Iza Marfisi-Schottman, Francesco Bellotti, Ludovic Hamon, and Roland Klemke, editors, *Games and Learning Alliance*, pages 45–54, Cham, 2020. Springer International Publishing.

[hel18] Hello quantum. https://medium.com/qiskit/hello-quantum-2c1c00fe830c, 2018. Accessed: 2024-06-30.

[JATK+24] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.

[Ken24] Kenney. Kenney.nl assets. https://kenney.nl/assets, 2024.

[KPJ21] Annakaisa Kultima, Laura Piispanen, and Miikka Junnila. Quantum game jam–making games with quantum physicists. In *Proceedings of the 24th International Academic Mindtrek Conference*, pages 134–144, 2021.

[nLa23] nLab authors. Qudit. https://ncatlab.org/nlab/show/qudit, 2023.

[par] Particle in a box. https://qplaylearn.com/game-particle-in-a-box. Accessed: 2024-06-30.

[PH24] Laura Piispanen and Noora Heiskanen. kiedos.art: Entangling quantum with art. https://kiedos.art/, 2024.

[PPW+22] Laura Piispanen, Marcel Pfaffhauser, James Wootton, Julian Togelius, and Annakaisa Kultima. Defining quantum games. *arXiv preprint arXiv:2206.00089*, 2022.

[qua] Quantum moves. https://citizensciencegames.com/games/quantum-moves/. Accessed: 2024-06-30.

[Qua20a] Quantum Flytrap Team. Quantum Flytrap, 2020. Accessed: 2024-06-30.

[Qua20b] Quarks Interactive. Quantum odyssey. https://play.google.com/store/apps/details?id=com.QuarksInteractive.QuantumOdyssey&hl=nl, 2020. Accessed: 2024-06-30.

[qua24a] Quantum inspire. https://www.quantum-inspire.com/, 2024.

[Qua24b] Quantum Inspire. What is a qubit? https://www.quantum-inspire.com/kbase/what-is-a-qubit/, 2024.

[Shi11] Pete Shinners. Pygame. http://pygame.org/, 2011.

[Smi09] Smite-Meister. Bloch sphere. https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg, 2009.

[SMW+22] Zeki C. Seskir, Piotr Migdał, Carrie Weidner, Aditya Anupam, Nicky Case, Noah Davis, Chiara Decaroli, İlke Ercan, Caterina Foti, Paweł Gora, Klementyna Jankiewicz, Brian R. La Cour, Jorge Yago Malo, Sabrina Maniscalco, Azad Naeemi, Laurentiu Nita, Nassim Parvin, Fabio Scafirimuto, Jacob F. Sherson, Elif Surer, James Wootton, Lia Yeh, Olga Zabello, and Marilù Chiofalo. Quantum games and interactive tools for quantum technologies outreach and education. *Optical Engineering*, 61(08), July 2022.

[SWM+24] Travis L Scholten, Carl J Williams, Dustin Moody, Michele Mosca, William Hurley, William J Zeng, Matthias Troyer, Jay M Gambetta, et al. Assessing the benefits and risks of quantum computers. *arXiv preprint arXiv:2401.16317*, 2024.

[SY22] Marija Scekic and Abuzer Yakaryilmaz. Comparing quantum software development kits for introductory level education. *Baltic Journal of Modern Computing*, 10(1):87–104, 2022.

[The24] The National Institute of Standards and Technology (NIST). Quantum algorithm zoo. https://quantumalgorithmzoo.org/, 2024.

[Woo17] James Wootton. Introducing the world's first game for a quantum computer. https://decodoku.medium.com/introducing-the-worlds-first-game-for-a-quantum-computer-50640e3c22e4, 2017.

[WW11] Colin P Williams and Colin P Williams. Quantum gates. *Explorations in Quantum Computing*, pages 51–122, 2011.