Quantum NFT

WHITEPAPER

QuantumNFT

pQCee Pte Ltd

Dr Tan Teik Guan and Jonathan Liu

August 2023
V0.1

pQCee
Be . Quantum . Ready

A Note from the pQCee Product Team

To the best of our knowledge, the latest quantum algorithm developed was by Farhi, Goldstone, and Gutmann when they introduced the Quantum Approximate Optimization Algorithm (QAOA) in 2014.

In subsequent years, there has been a shortage of new algorithms being written. This scarcity of innovation is attributed, in part, to the shortage of quantum engineers and the steep learning curve associated with quantum computing. The Level of Math often seen in quantum whitepapers poses an intimidating barrier. To start writing quantum circuits, we believe that it does not need to be so difficult and painful.

In this whitepaper, we use quantum circuits, quantum programs, and quantum algorithms, somewhat interchangeably. Also, we explore the synergies between blockchain technology and quantum technology. It becomes evident that both fields offer reciprocal benefits and conflicting ideals to one another.

As we have seen with popular Non-Fungible Tokens (NFTs) projects such as Cryptopunks and Bored Ape Yacht Club (BAYC), NFTs have shown to be an effective tool in bringing together like-minded individuals toward shared objectives. Our vision is to utilize NFTs to cultivate a robust technical quantum community. This thought experiment is called QuantumNFT.

As far as we know, there isn't anyone focused on addressing this exact problem. We identify that for most developers their primary motivation lies in channeling their skills into meaningful pursuits. Quantum computing offers a promising avenue. Despite the current challenges in quantum hardware, we believe that it is only a matter of time before it becomes mainstream. And when that happens, every company and organization will need quantum engineers.

We propose a platform for building quantum circuits without requiring an advanced physics degree. We envision that as more developers build quantum circuits, these quantum programs will eventually bring about substantial impacts on humanity.

To attract greater developer engagement, we explore various potential business models and future developments.

The intention of this thought experiment is to democratize quantum development, foster a thriving technical quantum community, and ignite enthusiasm among both Web3 developers and computer science/engineering undergraduates regarding the potential that quantum holds.

1 Introduction

QuantumNFT.xyz (QuantumNFT) is yet another quantum-readiness project by pQCee.com. It seeks to bring together a whole new generation of quantum-capable programmers as the backbone to usher in the next computing paradigm --- quantum computers.

After more than 40 years of research and development by physicists around the world, commercially-useful quantum computers are expected to become mainstream within the next 5 years. Its many potential applications include the discovery of new chemical compounds, more accurate modelling, optimization of highly complex systems, faster processing, and analysis of huge amounts of data. Projected market valuations for quantum computers are anticipated to surpass $4 billion by 2028, demonstrating a compound annual growth rate (CAGR) of 38%. These studies point to the need for more than just upgrades to computing hardware, but also a wider expansion in system requirements. This includes but is not limited to redesigning existing workflows and standards, new expertise in programming quantum computers and maintaining quantum software, as well as the integration of quantum computers into our everyday lives.

The question we should be asking now is this: is the broader community prepared for this change? If all the Fortune 1000 companies were to build in-house quantum teams to jumpstart their next leap in productivity, are there enough people with the requisite quantum computing knowledge and expertise (ie. quantum developers) to hire? Where do we find them?

1.1 Problem

When we take a count of software developers in the world, estimates from various sources found on Google.com all exceed 20 million developers. If we dive deeper into a niche community such as the blockchain (or Web3), this number hovers at around 30,000 active developers. But a search on the number of physicists working on quantum computers is likely to yield less than 3,000 worldwide.

This startling shortfall of quantum-capable developers means that we are going to see a repeat of the 1970s where classical computers, despite heavy investments, did not yield the promised productivity growth till over a decade later.

But bridging this gap is not simply a matter of conducting more classes to teach about quantum computing. At present, despite the public availability of small-scale quantum computing resources on the cloud, the users have been mainly physicists who have used quantum computers for academic publications. The level of interest amongst undergraduates and enthusiasts who have attended Quantum 101 courses has quickly waned due to the lack of applicability of quantum computers in their day-to-day lives. Which leads us to the problem --- the lack of a viable use-case or “killer-app” for small-scale quantum computers.

To-date, there has been no lack of efforts by academic and industry researchers to conduct experiments to demonstrate the usefulness or advantages of quantum computers, but most of these have either been inconsistent (ie. results are not guaranteed), disproved (ie. regular computers may perform better), or impractical (ie. experiments are not general purpose enough). This is understandable, given the noisy and relatively nascent state of quantum computers today. Instead of trying to perform better than classical computers, our approach is to find a use-case that can complement existing applications to achieve the goal of building a community of at least 10,000 quantum developers.

1.2 Proposed Solution

To build the community, we have chosen to target the following two segments:

- **Computer Science / Engineering undergraduates**. We expect these undergraduates to possess a requisite level of programming logic and knowledge, coupled with the drive to distinguish themselves when they enter the labour market which has seen significant layoffs in the technical sectors throughout 2022-2023.
- **Web3 developers**. Many of these Web3 developers are self-starters and have persisted in adapting to the Web3 environment despite the myriad of ever-changing programming languages, frameworks, and tools. Since we expect quantum programming to face similar challenges, the resilience of Web3 developers will feature well here.

In Web3, NFTs or (Non-Fungible Tokens) have been used as a mechanism to bring Web3 communities together. Ownership of a NFT within a particular collection imbues some level of recognition that the owner is part of an exclusive community. NFTs can be traded, transferred, and sold for financial benefits. Furthermore, the value of each NFT appreciates in accordance with its unique and rare attributes. Since NFTs are familiar to most Web3 developers, we have chosen to use NFTs, named QuantumNFT, as the mechanism to build and grow the technical quantum community.

The general idea of QuantumNFT works as follows:

- A quantum developer writes different quantum programs which when executed produce a certain quantum state.
- Each quantum state can be minted into a NFT.
- These NFTs are deterministically rendered such that the same quantum state will always yield the same image.
- The only difference would be the serial number to differentiate when a particular state was minted.
- A quantum developer would be allowed to mint NFTs using the same circuit, only once.
- Circuit complexity is defined by size and depth. A quantum developer who has minted NFTs with more gates and layers will be able to demonstrate higher proficiency of quantum programming.
- A quantum developer who has minted specific NFTs which other developers are unable to mint will be able to demonstrate mastery in quantum programming.

The characteristics of the QuantumNFT collection are:

- NFT identification policy: NFTs are uniquely identified by TokenIDs which includes identification information about the quantum program and a unique serial number.
- Number of NFTs in circulation: Unlimited
- Minting: To be done by quantum developers.
- Selling/Transfer: No selling or transfer allowed.

2 Implementation

The walk-through of how a quantum developer can mint a QuantumNFT is shown in Figure 1 (below).

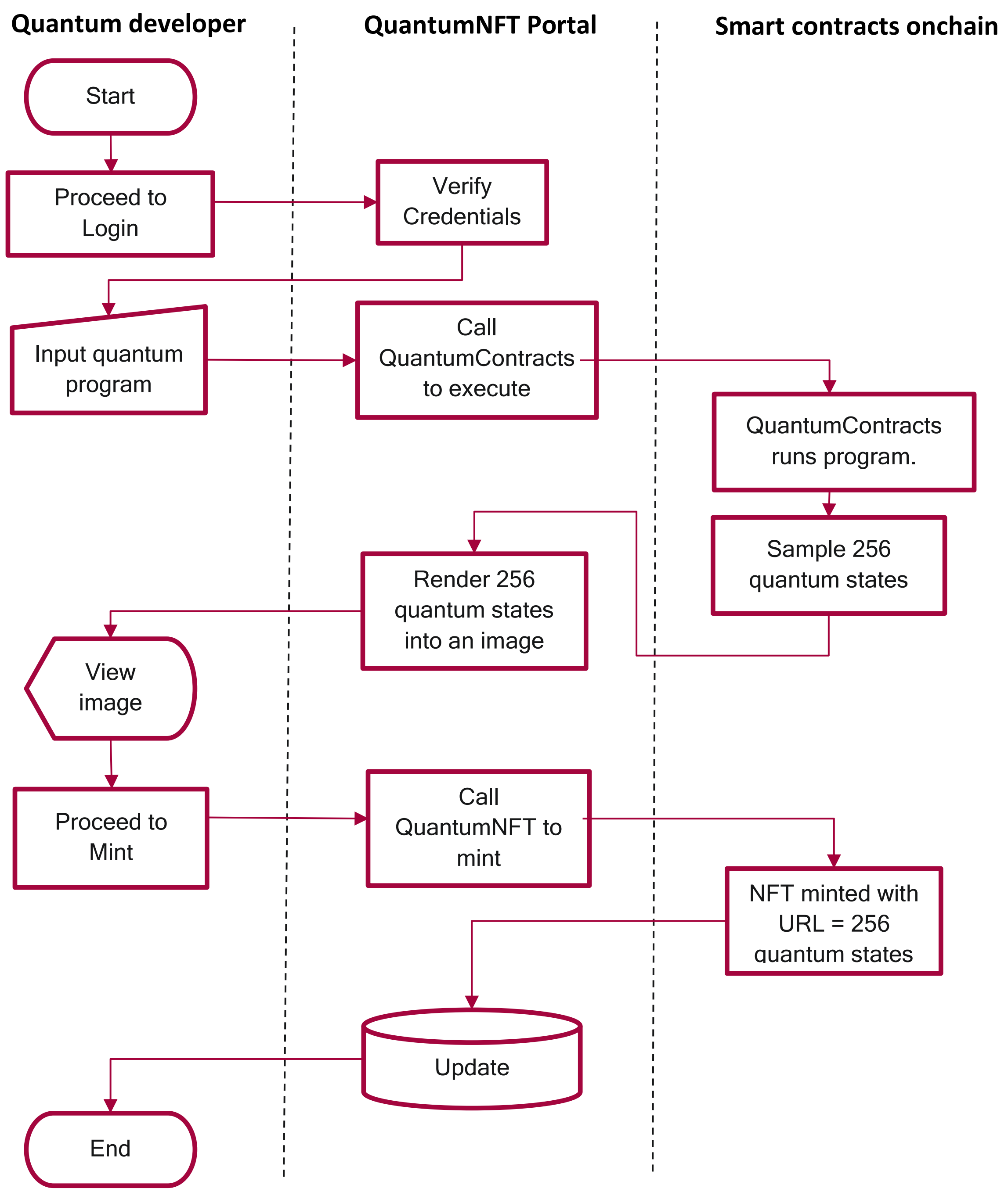


Figure 1.

1. The quantum developer starts by initiating the login process to establish an identity on the QuantumNFT portal.
2. Once authenticated, the quantum developer writes and submits the quantum program in the Quantum NFT portal.
3. The portal first calls QuantumContracts, a 8-qubit onchain quantum emulator smart contract to execute the quantum program.
4. QuantumContracts executes the quantum program, and the quantum state is returned. The state holds 256 possible results. For example, in the case of a 2-qubit bell state, QuantumContracts will return 128 results with “00” values, and 128 results with “11” values.
5. The portal will use the 256-value quantum state to deterministically render an image which is displayed to the quantum developer.
6. The quantum developer can then proceed to execute the mint operation which will store the quantum state on the blockchain. The execution of the quantum program is done separately from the minting of the NFT to reduce the gas fees required since QuantumContracts was implemented as a view function.
7. The portal will update the database to map the minted QuantumNFT to the quantum developer.

In the rest of the section, we cover the main implementation aspects of QuantumNFT.

2.1 Onchain Quantum Emulator

We built "QuantumContracts", an 8-qubit quantum emulator onchain in order to execute the quantum program submitted by the quantum developer. The language used to write the quantum program is "QuIC" (Quantum Interpreted Circuits) where the BNF notation of QuIC quantum program is as follows.

```
<Program>           ::= <Gate sequence> | <Program><Delim><Gate sequence>
<Gate sequence>     ::= <Gate> | <Gate><Gate sequence>
<Gate>              ::= H | C | N | X | Y | Z | P | p | T | t | I | m
<Delim>             ::= , | .
```

The use of QuIC allows for efficient description and execution of the quantum program which is important since we want to run the quantum program onchain. The gate operations supported by QuIC are Hadamard (H), Control-Not (CN), Pauli-X (X), Pauli-Y (Y), Pauli-Z (Z), п/2 and п/4 Phase-shift (P, T), Conjugate п/2 and п/4 Phase-shift (p, t), Identity (I) and measurement (m).

We first take the example of a 2-Qubit fully-entangled Bell state. Figure 2 (right), shows the circuit diagram of the quantum program (ie. Bell-state) which consists of passing the first qubit through a Hadamard gate, before a Control-Not gate which entangles the first qubit with the second qubit. This creates a fully-entangled state where the possible measurements of the 2 qubits are either "00" or "11". To represent this program in QuIC, we first segment the gates in an ordered sequence before enumerating the gates within the sequence. In Figure 2 (right), sequence 1 consists of a Hadamard gate on qubit 1 and Identity gate on qubit 2, and sequence 2 consists of a Control-Not function spanning from qubit 1 to qubit 2.

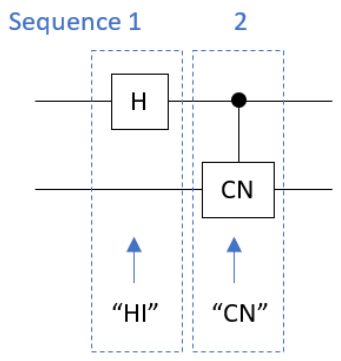


Figure 2 - 2-Qubit Bell-state

The eventual QuIC program is thus "`HI,CN.`" where the "," delimiter denotes the next sequence.

Some additional examples of QuIC programs of well-known quantum algorithms are:

- 3-Qubit GHZ :
  - `HII,CNI,ICN.`
- Grover's search for "11" :
  - `HHI,IIX,IIH,XXI,CCN,XXI,IIH,IIX,HHI,XXI,IHI,CNI,IHI,XXI,HHI.`

2.2 Data Stored Onchain

QuantumNFT stores the following data onchain:

- TokenID.
    - The TokenID uniquely identifies the NFT that is minted by each quantum developer.
    - It will be made up of <CircuitID> || <Serial Number>.
    - The <CircuitID> uniquely identifies the quantum program using CircuitID = Hash(quantum program). The same quantum program written by different developers will get the same <CircuitID>.
    - The <Serial Number> is an ever-increasing number on the total number of NFTs minted. Every NFT minted will cause the <Serial Number> to increase by 1.
- 256-value quantum state.
    - This is the result of 256 executions + measurement of the quantum program that is submitted by the quantum developer and is a representation of the potential usefulness of the quantum program.
    - The 256 execution results are sorted by qubit values starting with qubit values of all zeros and ending with qubit values of all ones.

Storing these two values does not reveal the actual quantum program submitted by the quantum developer but is able to allow a third party to verify the authenticity of the quantum program. A quantum developer can claim knowledge of a particular NFT by providing a quantum program such that:

- Hash(quantum program) = <CircuitID> which is part of the TokenID.
- Executing the quantum program 256 times results in the 256-value quantum state associated with the TokenID.

2.3 Image Representation of Quantum State

The visualization of the 256-value quantum state is an important feature within QuantumNFT. Most quantum programming platforms provide graphical representation of the quantum program or quantum circuit in the form of gates (see figure 2 above) and tend to depict the results in the form of histogram graphs. We believe there is room for innovation where better representation of the results can lead to better algorithms being written. For the initial version, we use a 2-dimension 256x256 palette of 24-bit RGB colour to represent the 256-value quantum state. Depending on the number of qubits used in the program, we segment the 24-bit colour for each pixel into equal parts, to be displayed if the qubit value is 1. The segmentation is shown in Figure 3 (below).

Qubit	Qubit colours when qubit = 1	Remarks
1	White	
2	🟨 🟦	12 bits per qubit
3	🟥 Red 🟩 Green 🟦 Blue	8 bits per qubit
4	🟥 🟩 🟦 ⬛	6 bits per qubit
6	🟥 ⬛ 🟩 ⬛ 🟦 ⬛	4 bits per qubit
8	🟥 ⬛ 🟩 🟩 ⬛ 🟦 🟦 ⬛	3 bits per qubit
	🟥 ⬛ 🟩 🟩 ⬛ 🟦 🟦 ⬛	

Figure 3 - 24-bit colour segmentation table

For example, for a 3-qubit program, the 24-bit pixel is equally divided to 8-bits to represent each qubit. If the result has the first qubit = 1 then the first 8 bits of the pixel is turned on, if the second qubit = 1 then the next 8 bits of the pixel is turned on, and similarly if the third qubit = 1 then the final 8 bits of the pixel is turned on. If all three qubits = 0, this means nothing is turned on resulting in black pixels. If all three qubits = 1, this means all 24 bits are turned on resulting in the presentation of a white pixel. We also divide the image with several rows representing the number of qubits in the program.

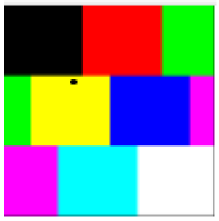
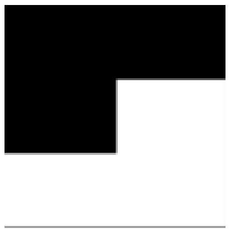


Figure 4 - Quantum state of "HHH"

Finally, we render the image in an animated gif for visual effects. Figure 4 (right) shows the representation a 3-qubit program "HHH." where all 3 qubits are put into a superposition.

The advantage of such a representation is its compactness in presenting the results. We give further examples of the image representation of the 3-qubit GHZ algorithm (on the left) and 8 qubits in superposition (on the right) in Figure 5 (below).


3 Qubit GHZ

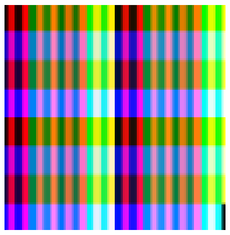

8 Qubit superposition

Figure 5 - More examples of quantum states

The future strategy will be to enhance the quality of the rendered NFTs. Because the quantum states are preserved on the blockchain, any enhancements made will not affect existing ones.

3 Business Model

There are several ways where monetization can happen for QuantumNFT:

- B2C Software-as-a-service.
    - Customer: Quantum developer.
    - In addition to gas fees, we can impose a per-transaction minting fee whenever a QuantumNFT is minted. This is similar to other Web3 projects.
- B2C Platform-as-a-service.
    - Customer: Quantum developer.
    - We can charge a subscription fee for the use of the QuantumContracts smart contract to perform quantum-computation on the submitted quantum program. This is similar to cloud providers that support quantum workloads.
- B2B Community Marketing and advertisement.
    - Customer: Companies in the quantum industry
    - We can charge for advertising banners or other marketing collateral which are displayed on the QuantumNFT portal. This is a similar business model to many marketplaces.
- B2B Data-as-a-service.
    - Customer: Companies wanting to hire quantum developers
    - We can charge for job searches by companies or publicizing job opportunities to the QuantumNFT community. This is a similar business model to many community portals.

At present, we are working on validating the appropriate model for QuantumNFT.

4 Future Developments

- Support other quantum programming languages. At present, QuantumNFT only supports QuIC. QuantumContracts can be extended to support other popular quantum programming languages including IBM Qiskit, Google CirQ, Microsoft Q#, QML, etc.
- Support more powerful quantum platforms including actual quantum computers. At present, only 8-qubit and below quantum programs are supported. This can be improved to support a higher number of qubits and possibly to run QuantumContracts on actual quantum hardware for performance improvements.
- Solution-bounty. At present, quantum developers can write different programs freely without any clear goals. This can further be improved by organizing contests for developers to compete on.