applied sciences

MDPI

Article

Q1Synth: A Quantum Computer Musical Instrument

Eduardo Reck Miranda [1,2,*], Peter Thomas [1] and Paulo Vitor Itaboraí [1]

[1] Interdisciplinary Centre for Computer Music Research (ICCMR), University of Plymouth, Plymouth PL4 8AA, UK
[2] Quantinuum, Partnership House, Carlisle Place, London SW1P 1BX, UK
* Correspondence: eduardo.miranda@plymouth.ac.uk

Abstract: This paper introduces Q1Synth, an unprecedented musical instrument that produces sounds from (a) quantum state vectors representing the properties of a qubit, and (b) its measurements. The instrument is presented on a computer screen (or mobile device, such as a tablet or smartphone) as a Bloch sphere, which is a visual representation of a qubit. The performer plays the instrument by rotating this sphere using a mouse. Alternatively, a gesture controller can be used, e.g., a VR glove. While the sphere is rotated, a continuously changing sound is produced. The instrument has a 'measure key'. When the performer activates this key, the instrument generates a program (also known as a *quantum circuit*) to create the current state vector. Then, it sends the program to a quantum computer over the cloud for processing, that is, *measuring*, in quantum computing terminology. The computer subsequently returns the measurement, which is also rendered into sound. Currently, Q1Synth uses three different techniques to make sounds: frequency modulation (FM), subtractive synthesis, and granular synthesis. The paper explains how Q1Synth works and details its implementation. A setup developed for a musical performance, *Spinnings*, with three networked Q1Synth instruments is also reported. Q1Synth and *Spinnings* are examples of how creative practices can open the doors to new application pathways for quantum computing technology. Additionally, they illustrate how such emerging technology is leading to new approaches to musical instrument design and musical creativity.

Keywords: quantum computing applications; quantum computer music; quantum musical instrument; quantum networks; music technology

Citation: Miranda, E.R.; Thomas, P.; Itaboraí, P.V. Q1Synth: A Quantum Computer Musical Instrument. *Appl. Sci.* **2023**, *13*, 2386. https://doi.org/10.3390/app13042386

Academic Editors: Rocco Zaccagnino and Lamberto Tronchin

Received: 26 December 2022
Revised: 30 January 2023
Accepted: 10 February 2023
Published: 13 February 2023

1. Introduction

Computers are omnipresent in almost every aspect of the music industry nowadays [1,2]. Emerging new quantum computing technologies are likely to continue this trend. They are bound to impact how we create, perform, listen, and commercialize music in the future.

Researchers and practitioners in the newborn field of *Quantum Computer Music* have begun to explore ways to leverage the quantum-mechanical nature of quantum computing for applications in music [3–5]. It is expected that this new technology will lead to new instruments and approaches to creating music. This paper introduces examples of this.

The majority of quantum computer music research to date has been focused on using quantum computers to generate music algorithmically. For instance, ref. [6] adapted a quantum natural language processing (QNLP) method to implement a system for creating music exploring the relationship between music and language. Additionally, ref. [7] introduced the Basak–Miranda generative music algorithm. This algorithm leverages a property of quantum mechanics known as constructive and destructive interference to compose tunes. Furthermore, ref. [8] introduced methods to generate music using a type of quantum computing known as adiabatic quantum computing.

We are interested, however, in developing new musical instruments. Research on this front has been less common. This is probably due to the fact that the quantum

representation of audio is not trivial. There is no agreed method to date for representing audio quantumly [9]. Moreover, at present, there is no hardware available for the fully-fledged implementation of quantum audio.

Nevertheless, there have been some initiatives to synthesise sounds using quantum computing. For instance, ref. [10] developed a system for controlling the parameters of a digital sound synthesiser with results from quantum computations. Another approach was proposed in [11]. In this case, data acquired from the workings of a quantum processor were converted into audio signals in real time during a performance.

This paper introduces Q1Synth and an example of a performance using it. Q1Synth is a novel software-based musical instrument that renders sounds from *quantum state vectors* representing the properties of a *qubit* and its *measurement*.

Q1Synth is presented on the computer screen as a Bloch sphere, which represents a qubit (Figure 1). The performer plays the instrument by rotating and measuring the qubit. It can be rotated using the computer's mouse or an external MIDI controller. While the qubit is rotated, a continuously changing sound is produced. Additionally, when the qubit is measured, the system also produces a dynamic sound.

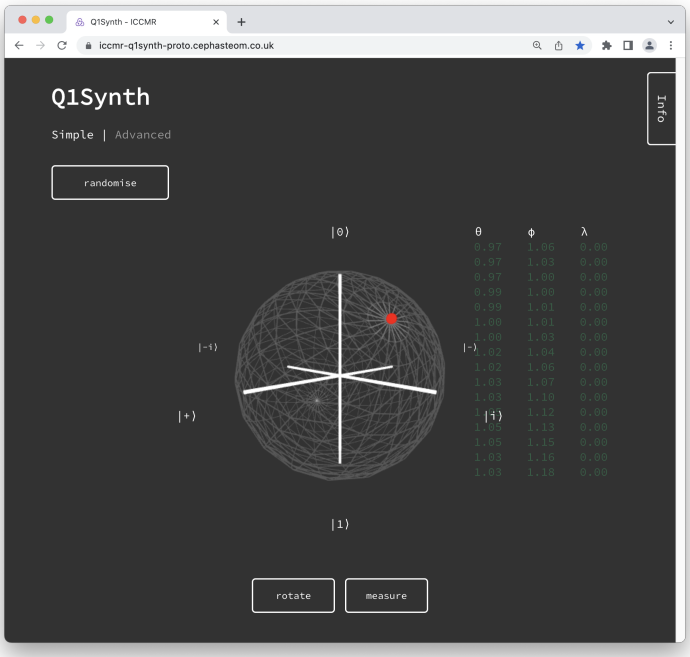

Figure 1. A screenshot of the Q1Synth in 'Simple' mode.

The paper begins with a brief introduction to quantum computing, focusing on the qubit and Bloch's sphere. This introduction is limited to what is deemed necessary to understand how Q1Synth works. For more detailed explanations of quantum computing, please refer to [12,13]. Then, it explains how the instrument synthesises sound. It focuses on just one of its synthesis techniques: frequency modulation. Next, it shows how we networked three Q1Synths to form a trio for a live performance. The paper ends with a discussion on ongoing developments.

2. The Qubit and Bloch's Sphere

A qubit is to a quantum computer what a bit is to a digital one: it is a basic unit for representing information. The state of a qubit is characterised within a two-dimensional vector space, known as 2D Hilbert space. The canonical basis vectors in this space are notated as $|0\rangle$ and $|1\rangle$, which is an abbreviated way to represent such vectors. In quantum mechanics, the bra–ket notation, or Dirac's notation, is used to represent quantum states.

The state $|\psi\rangle$ of a qubit is expressed mathematically as a linear combination of basis vectors as follows: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. This linear combination expresses a state of superposition.

Simply put, a quantum computer processes information with qubits in a state of superposition. But it returns binary numbers (0 s and 1 s) when we read the qubits. In quantum computing terminology, the act of reading qubits is referred to as *projective measurement* [12].

A single qubit is represented visually as a sphere with opposite poles, with $|0\rangle$ at the north pole and $|1\rangle$ at the south. This sphere is called *Bloch sphere*. From its centre, a unitary vector $|\psi\rangle$ can point to anywhere on the surface. This vector, which represents the state of the qubit, is referred to as a *state vector*.

In quantum mechanics, the variables α and β represent *amplitudes*. In practice, we can say that $|\alpha|^2$ and $|\beta|^2$ encode the probabilities for the possible results of measurements made on a quantum system.

Quantum computers are programmed by applying sequences of operations to qubits. Programming languages for quantum computing provide a number of operations, referred to as *gates*, which act on qubits. For instance, the **X** gate rotates the state vector of a qubit by 180 degrees around the x-axis of the Bloch sphere geometry: if the qubit vector is pointing to $|0\rangle$, then this gate flips it to $|1\rangle$, or vice versa (Figure 2).

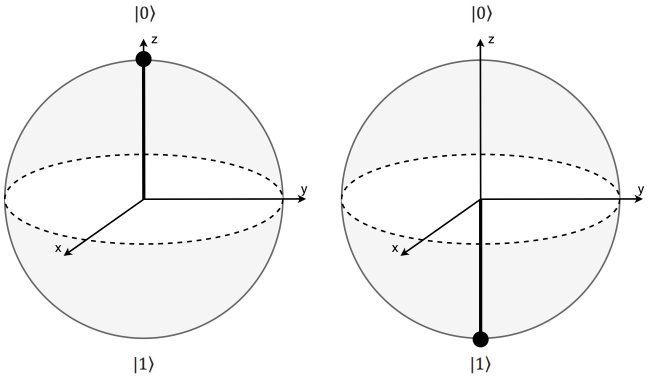

Figure 2. The **X** gate rotates the state vector by 180 degrees around the x-axis.

Essentially, quantum gates perform rotations. A generic rotation gate **U** is typically available for programmers, with three angles called *Euler angles*. Thus, any rotation gate can be specified in terms of **U**. For instance the gate **X** is equivalent to $\mathbf{U}(\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2})$.

A quantum program is often depicted as a circuit with sequences of quantum gates operating on qubits (Figure 3). Typically, the qubits start in the ground state $|0\rangle$.

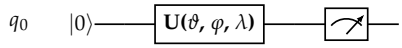

Figure 3. An example of a simple quantum circuit showing the application of a **U** gate to a single qubit q_0. The dial at the end represents the measurement operator.

3. The Q1Synth System

Q1Synth works with one qubit. Figure 1 shows a screenshot of the system in 'Simple' mode. It shows an artistic representation of a Bloch sphere and its respective axes: z, y, and x. Note the labels $|0\rangle$ on the north and $|1\rangle$ on the south of the vertical z axis. The labels $|+\rangle$ and $|-\rangle$ mark the opposing ends of the x axis, as $|i\rangle$ and $|-i\rangle$ denote y. The meaning of $|+\rangle$ and $|i\rangle$ come from quantum mechanics definitions [12] that are not so important to understand right now.

The coordinates of the Bloch sphere are given in the form of Euler angles $(\theta, \varphi, \lambda)$. They describe rotation angles necessary to position the state vector around the sphere, starting from the north pole. The position of the state vector is represented by a red dot.

The angle θ will determine the inclination (or latitude) of the vector, whereas φ is responsible for its azimuth (or longitude). The angle λ does not change the position of the red dot, but it will influence the orientation of the sphere. We refer to λ as the *phase* of the vector.

Once the button 'rotate' is pressed, the instrument begins to make a sound. Rotating the sphere (e.g., by clicking and dragging the mouse) moves the state vector and modifies the sound continuously. The red dot indicates where the state vector is pointing.

When the 'measure' button is activated, the system takes control from the user, and moves the vector to either north or south, depending on the result of the measurement. The vector moves in slow motion and is accompanied by a respective sound. Once the vector reaches the destination the sound ends. Activating the 'rotate' button again recommences the process, and so on. This is how the instrument is played.

When the measuring command is detected, the system builds a quantum circuit that implements a **U** gate and sends it to a quantum computer over the cloud (Figure 4). The coordinates of the sphere at the moment the 'measure' button is activated define the angles for the **U** gate. Then, the quantum computer returns the measurement result (c_0), which defines whether the vector moves north or south. Currently, Q1Synth connects to an IBM Quantum processor located in the USA.

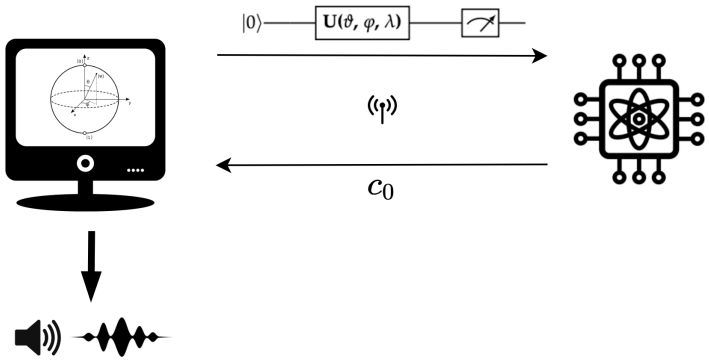

Figure 4. For measurement, Q1Synth builds a quantum circuit and sends it to a quantum computer over the cloud.

At the time of writing, Q1Synth uses three types of sound synthesis techniques [14]: frequency modulation (FM), additive synthesis, and granular synthesis. This paper focuses on the method for controlling FM. The methods for controlling the other two techniques are identical to the FM one.

FM synthesis is based upon the same principles used for FM radio transmission: the frequency of a waveform (referred to as the *carrier*) is altered with a modulating signal (referred to as the *modulator*) [15]. There are a number of variations in FM synthesiser design. The most basic comprises two sine wave oscillators: one acting as the modulator and the other as the carrier signal, respectively (Figure 5).

In FM sound synthesis, the carrier is the signal that we hear directly. By contrast, the modulator is heard only indirectly, because its output is added to the base frequency of the carrier (Figure 5). When the frequency of the modulator is in the audio range, numerous additional partials, or sidebands, are added to the spectrum of the carrier's wave.

Let us consider the vibrato effect as a starting point example to illustrate the FM technique. Vibrato is a tremulous effect imparted to vocal or instrumental notes. It is often used to add expressivity to musical notes. It is caused by variations in pitch. The fundamental difference, however, is that vibrato uses a sub-audio signal to modulate the pitch of a sound. A sub-audio signal is a low-frequency signal, well below the human hearing threshold, which is approximately 20 Hz. The resulting sound, in this case, has a perceptibly slow variation in its pitch. If the modulator's frequency is set to a value above the human

hearing threshold, then new components are added to the sound spectrum of the carrier. The frequency and amplitudes of these new components influence our perception of timbre.

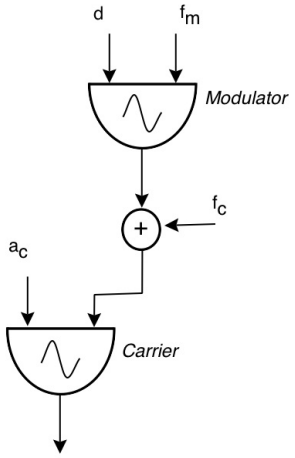

Figure 5. A simple FM synthesis scheme with two oscillators: a modulator and a carrier. Whereas d defines the amplitude and f_m the frequency of the modulator, respectively, a_c defines the amplitude of the carrier. The sinusoidal output from the modulator is added to the frequency of the carrier (f_c).

The Q1Synth implementation of FM requires values for seven parameters:

- Frequency (*freq*): defines the pitch of the sound.
- Amplitude (*amp*): defines the loudness of the sound.
- Reverberation (*reverb*): defines the amount of reverberation to add to the sound.
- Modulation index (*mod index*): defines the number of components in the sound spectrum. The higher this index, the higher the number of components.
- Harmonicity ratio (*harmonicity*): working alongside *mod index*, this parameter defines the richness of the sound. The higher this ratio, the richer the sound.
- LFO frequency (*lfo freq*): the low-frequency oscillator (LFO) creates a vibrato or tremolo effect. The higher the frequency, the faster the vibrato.
- LFO depth (*lfo depth*): defines the intensity of the vibrato.

In a nutshell, the system uses the angles (θ, φ, and λ) of the Bloch sphere to interpolate parameter values for the FM algorithm. First, the user defines which synthesis parameters each angle will control. For instance, by default the system associates the following parameters:

- θ inclination = {*freq*, *amp*, *reverb*},
- φ azimuth = {*mod index*, *harmonicity*},
- λ phase = {*lfo freq*, *lfo depth*}.

In 'Advanced' mode view, the system shows sliders on either side of the Bloch sphere. They are used to specify the range of values for each parameter (Figure 6). Each sphere axis is assigned a pair of slider banks, enabling the values at each extreme to be interpolated as the sphere is rotated. For example, when the state vector is pointing north, ($|0\rangle$ position), the frequency, amplitude, and reverb parameter values will correspond to the sliders on the left side of the screen. When the state vector is pointing south ($|1\rangle$ position), these parameters correspond to the sliders on the right. When the sphere is between these poles we hear an interpolation between each slider pair, corresponding to the angle of inclination. It is possible to save presets.

Similarly, pairs of slider banks are assigned to the azimuth (interpolating between the $|+\rangle$ and $|-\rangle$ directions), and the sphere's phase. By re-orienting the sphere around these planes, the user interpolates between the slider settings at each extreme. All interpolations are linear. For example, if a pair of controls is set to 0% on the left and 100% on the right, when the sphere is rotated to the middle, the parameter value will be equal to 50%.

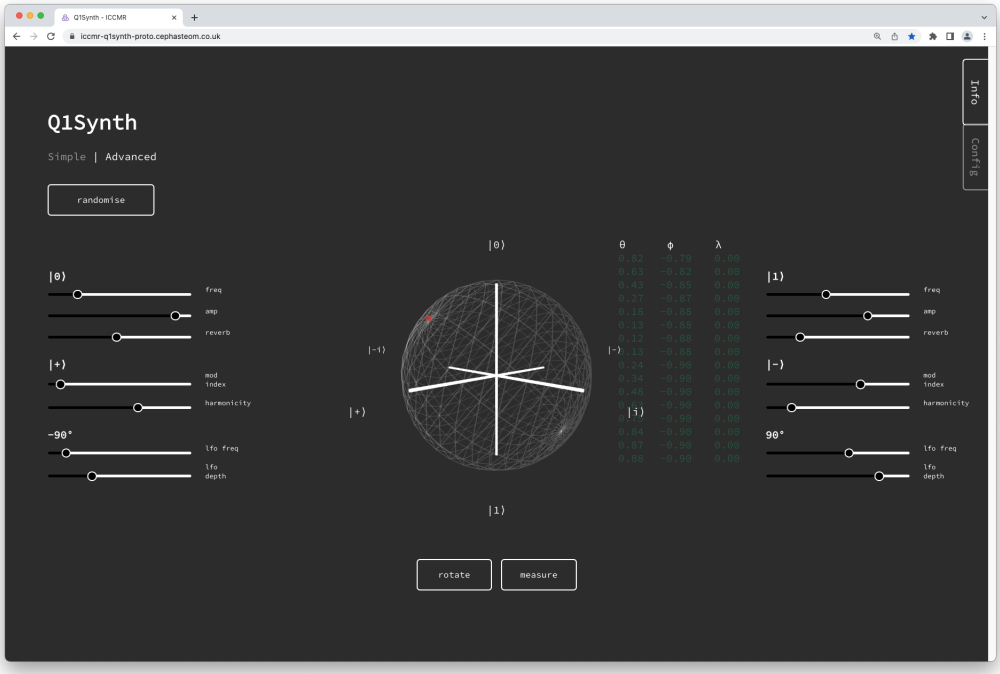

Figure 6. A screenshot of the Q1Synth in 'Advanced' mode. In contrast to the 'Simple' mode, in 'Advanced' mode the system provides options to adjust the range of the synthesis parameters.

We note that in quantum computing, a measurement causes the state vector to 'collapse' instantly to either $|0\rangle$ or $|1\rangle$ [16]. However, we took the liberty to convey this concept artistically: we added a virtual time delay to sonify the 'collapse'. Thus, after a measurement, Q1Synth makes an animated trajectory of the red dot towards the corresponding pole. In doing so, the synthesiser performs a complex, multidimensional modulation, interpolating between all parameters simultaneously until the red dot reaches its final destination, at which point the sound fades out.

4. Q1Synth Networks for Group Performance

This section introduces the setup developed for a performance entitled *Spinnings*, with three Q1Synth units. A MIDI gesture controller is used to rotate the qubit with hand movements.

Three Q1Synth instruments are networked in a star topology with an additional machine acting as a hub (Figure 7). As each player rotates the spheres, the instruments produce their respective sounds, which are mixed and relayed to the speakers (Figure 7c). The hub tracks the state vectors of each instrument and simulates a combined three-qubit quantum state dynamically (Figure 7b).

The hub is programmed to recognise the thumbs-up gesture as a command for measurement (Figure 7b). When a player makes a thumbs-up, the hub stops tracking the instruments and builds a quantum circuit expressing the current status of the simulated state vector. Then, it sends the circuit to the quantum computer for measuring (Figure 7d,e).

The measurement, which in this case will be three digits long ($[c_2, c_1, c_0]$), is relayed back to the hub. The hub informs each instrument of its respective outcome and synthesises the 'measurement sound'. This sound lasts for the period it takes for the vectors (i.e., red dots) to move to the north or sound poles of the respective instruments. It is programmed to take five seconds by default, but this is customisable. As soon as the sound ends, the players gain back control of the spheres and another cycle commences.

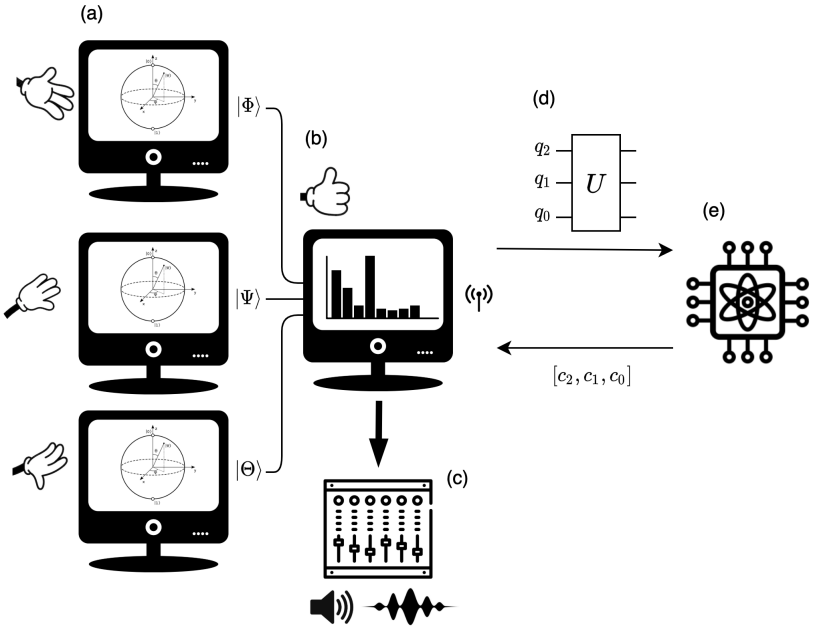

Figure 7. Three networked Q1Synth instruments for the performance *Spinnings*. Three Q1Synth units (**a**) are networked in a star topology with an additional machine (**b**) acting as a hub. The hub builds a quantum circuit expressing the current status of the simulated state vector (**c**) and sends the circuit to the quantum computer (**d**) for measuring. The hub informs each instrument of its respective outcome and synthesises the 'measurement sound' (**e**).

Even though each player has control over their own sound as they rotate the sphere, the measurement sound will be surprising: it depends on the measurement of the combined quantum state.

Spinnings was premiered on 8 November 2022. The concert took place at the Goethe-Institut in London, during a *Living in a Quantum State* event [17] to launch the book *Quantum Computer Music*, edited by E.R.M. [5]. The event was organised with Moth Quantum [18].

The instruments and the hub were projected on a cinema screen, which enabled the audience to follow the actions of the performers (Figure 8). A live recording of the performance is available [19].

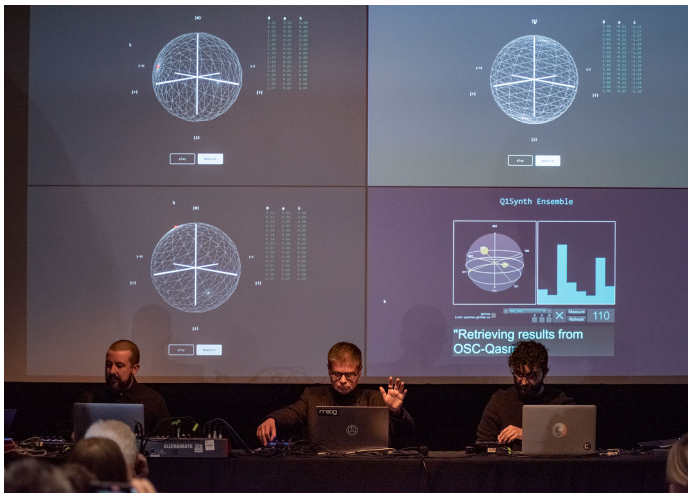

Figure 8. Premiere of *Spinnings* at the Goethe-Institut, starring P.T. (on the **left**), E.R.M (in the **middle**), and P.V.I. (on the **right**).

5. Implementation Technicalities

Q1Synth was developed as a Web application. First introduced in 2011, the Web Audio API provides a powerful system for handling audio on the Internet, turning any Web browser into a widely available, and potentially versatile, musical tool [20].

Q1Synth's sound synthesis engines were implemented with the Web Audio framework Tone.js [21]. The user interface is built with React.js [22] and written in TypeScript [23]. The Bloch sphere visualisation is rendered using P5.js [24].

The instruments were networked using OSC-Qasm, a Python package developed at ICCMR, University of Plymouth, as part of QuTune Project [25]. OSC-Qasm connects music programming environments with quantum hardware [26]. It is based on the OpenSound-Control (OSC) communication protocol, which is widely used by the music technology community [27].

OSC-Qasm includes a server and client applications. For *Spinnings*, the OSC-Qasm Client was developed with Max/MSP [28] and QuTune's QAC Toolkit package [29]. The client builds the quantum circuits from the Q1Synth coordinates in OpenQASM (Quantum Assembly Language) [30]. The OSC-Qasm Server is a cross-platform, Python-based application, that reads quantum circuits written in OpenQASM and runs them on quantum backends, either simulators or real hardware. It is powered by the Python-based quantum programming package Qiskit [31].

Finally, note that the Bloch sphere coordinates captured by Q1Synth's interface use the *ZYZ* convention [32] for defining the Euler angles passed to the quantum circuit. However, for visualisation, P5.js contains a different definition for its coordinate system. To achieve the desired visual result, it is necessary to use the coordinates in *YZY* convention for rotating the sphere.

6. Conclusions and Ongoing Work

This paper introduced a quantum computer-based musical instrument. To the best of our knowledge, this is an unprecedented concept.

Q1Synth and *Spinnings* are examples of how creative practices open the doors to new application pathways for quantum computing technology. Conversely, they illustrate how emerging quantum computing technology is leading to new approaches to musical instrument design and musical creativity.

Currently, quantum hardware is commonly available only through the cloud. They are geared towards operating in batch processing regimes. But musical interaction requires real-time processing. With Q1Synth, we started to tackle issues in real-time interaction with quantum computers.

Developing the instrument as a Web application paves the way for a Quantum Internet of Things (QuIT); Figure 9 shows Q1synth running on an iPhone. Additionally, *Spinnings* glimpsed at how QuIT might be harnessed for creative practices in the not-so-distant future.

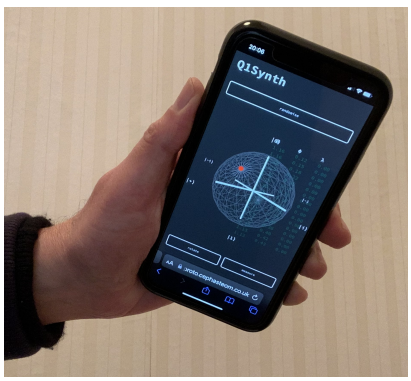

Figure 9. Towards a Quantum Internet of Things (QuIT): Q1Synth running on a smart phone.

Quantum computers need classical ones to function. Seamless integration and connectivity of different computing technologies will be of paramount importance to harness quantum computers for novel software solutions. Practical connectivity requirements for Q1Synth and *Spinnings* were resolved through OSC-Qasm [26]. Although we have been developing OSC-Qasm to connect music programming environments with quantum hardware, the OSC protocol is generic enough to embrace domains other than music.

Currently, we are working on *Spookings*, a musical performance with dozens of networked instruments geographically distributed, e.g., different cities.

The qubits in *Spinnings* do not interact with each other. For *Spookings*, we are advancing ways to entangle instruments, e.g., put instruments in a Bell state [12], whereby the measurement of one dictates the behaviour of another. This is a musical concept for group performance, which only quantum technology can truly afford. We intend to stage a performance of *Spookings* in the Metaverse [33].

A lite version of Q1Synth is freely available online [34]. As it works only in quantum simulation mode, it does not require a connection to a quantum hardware backend. This version makes sounds solely with the FM synthesis technique.

Author Contributions: Conceptualization, E.R.M.; methodology, E.R.M.; software, P.T. and P.V.I.; validation, E.R.M., P.T. and P.V.I.; investigation, E.R.M., P.T. and P.V.I.; resources, E.R.M.; writing—original draft preparation, E.R.M.; writing—review and editing, E.R.M., P.T. and P.V.I.; supervision, E.R.M.; project administration, E.R.M.; funding acquisition, E.R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by (a) the School of Culture and Society, University of Plymouth, UK and (b) QCS Hub, UK Quantum Technologies Programme, as part of the QuTune Project.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bevins, G. Computer Technology in Modern Music: A Study of Current Tools and How Musicians Use Them. Master's Theses, California State University, St. Chico, CA, USA, 2013.
2. Wikstrom, P. The Music Industry in an Age of Digital Distribution. In *Change: 19 Key Essays on How the Internet is Changing Our Lives*; Vazquez, J., Morozov, E., Castells, M., Gelemter, D., Eds.; Turner/BBVA Group: Madrid, Spain, 2013; pp. 1–24. ISBN 978-8415832454.
3. Souma, S. Exploring the Application of Gate-Type Quantum Computational Algorithm for Music Creation and Performance. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 83–103. ISBN 978-3-031-13909-3.
4. Mannone, M.; Rocchesso, D. Quanta in Sound, the Sound of Quanta: A Voice-Informed Quantum Theoretical Perspective on Sound. In *Quantum Computing in the Arts and Humanities*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 193–226, ISBN 978-3-030-95538-0.
5. Miranda, E.R., (Ed). *Quantum Computer Music: Foundations, Methods and Advanced Concepts* ; Springer Nature: Cham, Switzerland, 2022; ISBN 978-3-031-13909-3.
6. Miranda, E.R.; Yeung, R.; Pearson, A.; Meichanetzidis, K.; Coecke, B. A Quantum Natural Language Processing Approach to Musical Intelligence. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 313–356. ISBN 978-3-031-13909-3.
7. Miranda, E.R.; Basak, S. Quantum Computer Music: Foundations and Initial Experiments. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 43–67. ISBN 978-3-031-13909-3.
8. Arya, A.; Botelho, L.; Canete, F.; Kapadia, D.; Salehi, O. Applications of QuantumAnnealing to Music Theory. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 373–406. ISBN 978-3-031-13909-3.
9. Itaboraí, P.V.; Miranda, E.R. Quantum Representations of Sound: From Mechanical Waves to Quantum Circuits. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 223–274, ISBN 978-3-031-13909-3.

10. Hamido, O.C.; Cirilo, G.A.; Giusto, E. Quantum synth: A quantum computing-based synthesizer. In Proceedings of the 15th International Conference on Audio Mostly (AM'20), Graz, Austria, 15–17 September 2020; pp. 265–268. [CrossRef]
11. Topel, S.; Serniak, K.; Burkhart, L.; Carle, F. Superconducting Qubits as Musical Synthesizers for Live Performance. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 447–464, ISBN 978-3-031-13909-3.
12. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: New York, USA, 2011; ISBN 978-1-107-00217-3.
13. Sutor, R.S. *Dancing with Qubits: How Quantum Computing Works and How It Can Change the Word*; Packt Publishing Ltd.: Birmingham, UK, 2019; ISBN 978-1-83882-736-6.
14. Miranda, E.R. (Ed.) *Computer Sound Design: Synthesis Techniques and Programming*, 2nd ed.; Focal Press: Oxford, UK, 2022; ISBN 978-0-240-51693-6.
15. Chowning, J.; Bristow, D. *FM Theory and Applications: By Musicians for Musicians*; Yamaha Music Foundation: Hokkaido, Japan, 1986.
16. von Neumann, J. *Mathematical Foundations of Quantum Mechanics: New Edition*; Wheeler, N., Ed.; Princeton University Press: Princeton, NJ, USA, 2018; ISBN 978-0-691-17857-8.
17. Living in a Quantum State. Series of Events Organised by Goethe-Institut. Available online: https://www.goethe.de/prj/lqs/en/eve/sou.html (accessed on 19 December 2022).
18. Moth Quantum. Available online: https://www.mothquantum.com/ (accessed on 19 December 2022).
19. *Spinnings*; Goethe-Institut London: London, UK, 2022.
20. Web APIs. Available online: https://developer.mozilla.org/en-US/docs/Web/API (accessed on 14 December 2022).
21. Tone JavaScript Documentation Page. Available online: https://tonejs.github.io/ (accessed 16 December 2022).
22. React Documentation Page. Available online: https://reactjs.org/ (accessed on 14 December 2022).
23. TypeScript Documenation Page. Available online: https://www.typescriptlang.org/ (accessed on 14 December 2022).
24. P5 JavaScript Library Documentation Page. Available online: https://p5js.org/ (accessed on 14 December 2022).
25. QuTune Project Page. Available online: https://iccmr-quantum.github.io/ (accessed on 16 December 2022).
26. Hamido, O.C.; Itaborai, P.V. OSC-Qasm: Interfacing Music Software with Quantum Computing. *arXiv* **2022**, arXiv:2212.01615. Available online: https://arxiv.org/abs/2212.01615 (accessed on 16 December 2022).
27. OpenSoundControl Documentation Page. Available online: https://opensoundcontrol.stanford.edu/ (accessed on 17 December 2022).
28. Max 8 Documentation Page. Available online: https://docs.cycling74.com/max8 (accessed on 16 December 2022).
29. Hamido, O.C. QAC: Quantum-Computing Aided Composition. In *Quantum Computer Music: Foundations, Methods and Advanced Concepts*; Miranda, E.R., Ed.; Springer Nature: Cham, Switzerland, 2022; pp. 159–195. ISBN 978-3-031-13909-3.
30. Cross, A.W.; Bishop, L.S.; Smolin, J.A.; Gambetta, J.M. Open Quantum Assembly Language. *arXiv* **2017**, arXiv:1707.03429. Available online: Http://arxiv.org/abs/1707.03429 (accessed 17 December 2022).
31. Qiskit Website. Available online: https://qiskit.org/ (accessed on 14 December 2022).
32. Hanson, A.J. *Visualizing Quaternions*; Elsevier Science: Amsterdam, The Netherlands, 2006; pp. 52–54. ISBN 978-0-12-088400-1.
33. Ball, M. *The Metaverse: And How It Will Revolutionize Everything*; Liveright: New York, NY, USA, 2022.
34. Q1Synth Quantum Computer Synthesiser—Lite Demo Version. Available online: https://iccmr-q1synth-proto.cephasteom.co.uk/ (accessed on 16 December 2022).