Quantum Computing and Visualization: A Disruptive Technological Change Ahead

E. Wes Bethel*,† Mercy G. Amankwah‡, Jan Balewski†, Roel Van Beeumen†, Daan Camps†, Daniel Huang*,†, Talita Perciano†
*San Francisco State University, †Lawrence Berkeley National Laboratory ‡Case Western Reserve University,

Abstract—
*The focus of this Visualization Viewpoints article is to provide some background on Quantum Computing (QC), to explore ideas related to how visualization helps in understanding QC, and examine how QC might be useful for visualization with the growth and maturation of both technologies in the future. In a quickly evolving technology landscape, QC is emerging as a promising pathway to overcome the growth limits in classical computing. In some cases, QC platforms offer the potential to vastly outperform the familiar classical computer by solving problems more quickly or that may be intractable on any known classical platform. As further performance gains for classical computing platforms are limited by diminishing Moore's Law scaling, QC platforms might be viewed as a potential successor to the current field of exascale-class platforms. While present-day QC hardware platforms are still limited in scale, the field of quantum computing is robust and rapidly advancing in terms of hardware capabilities, software environments for developing quantum algorithms, and educational programs for training the next generation of scientists and engineers. After a brief introduction to QC concepts, the focus of this article is to explore the interplay between the fields of visualization and QC. First, visualization has played a role in QC by providing the means to show representations of the quantum state of single-*qubits *in superposition states and multiple-*qubits *in entangled states. Second, there are a number of ways in which the field of visual data exploration and analysis may potentially benefit from this disruptive new technology though there are challenges going forward.*

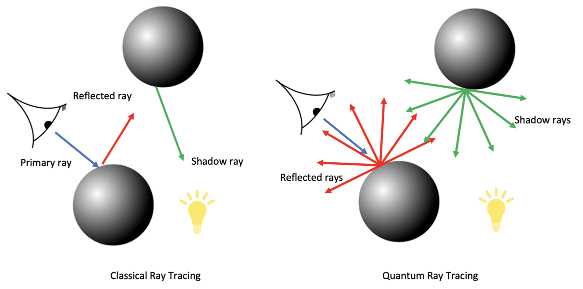
1. Introduction

Visualization as we commonly know it provides a means for "seeing the unseeable", for transforming abstract information into readily comprehensible images [2]. The field of visualization has evolved within the rubric of classical computing constructs, where information is represented as binary values, zeros and ones, and manipulated using boolean and arithmetic operations. Prior work in classical visualization is diverse and ranges from reformulating serial algorithms to perform better on parallel hardware, to incorporating the principles of optics and radiation transport modeling to create photorealistic renderings of models and data.

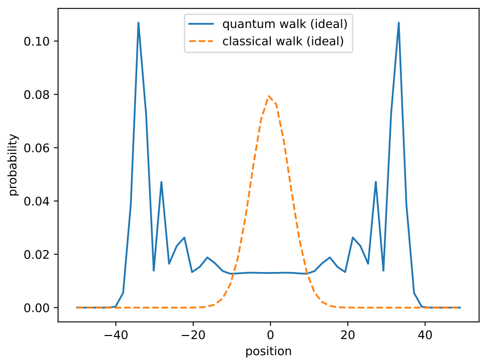
The emerging field of *quantum computing* (QC) is shaping up to be significantly more disruptive than previous changes we've been through with classical computing technology. These technologies include the physical means for storing information, the mathematical basis for conceptually manipulating that information, and the semantic and syntactic way of expressing algorithms for implementing those manipulation concepts. Because QC is so different than classical computing in so many respects, we expect the transition into QC to be even more disruptive than any of the technological transitions we have seen so far. This thought applies broadly to all of computing as well as specifically to the field of visualization. As the field of visualization continues to evolve, we may wonder how it will be impacted by QC and how it will impact QC.

While the transition to QC will likely be bumpy, there are several aspects of QC that offer potential gains for graphics and visualization in terms of scalability of methods and quality of results. Fig. 1a illustrates a key

Digital Object Identifier 10.1109/MCG.2023.3316932



(a) In classical raytracing, a single ray is tested against objects in the scene one after another. In the quantum world, a single ray is tested against all objects in the scene at once. Image adapted from [1].



(b) Classical random walks and quantum walks produce different probability distributions resulting in vastly different sampling quality characteristics.

FIGURE 1: Quantum algorithms exhibit fundamentally different behavior from classical ones.

difference between classical and quantum raytracing. In the classical implementation, a ray is tested against objects in the scene one after another. In the quantum implementation, a ray is tested against all objects in the scene with a single quantum operation applied at once to the 2^n quantum states created from n entangled *qubits*. Fig. 1b shows how a classical random walk converges to a probability distribution that looks like a Gaussian. On the other hand, the quantum walk produces a distribution that more thoroughly covers the mathematical space [3] resulting in a potentially higher quality result for algorithms that use randomized methods based on walks.

The objective for this article is to explore how visualization has helped understanding QC concepts (§3) as well as to consider ways that visualization may benefit from QC technology (§4). Because QC is so much different than classical computing, we first provide a brief introduction of key ideas and terminology (§2).

2. Quantum Computing Background

The language we use to describe concepts in QC, terms like *qubit*, *quantum state*, *superposition*, and *entanglement* may be unfamiliar to many. The information contained in classical bits and quantum bits (*qubits*) is completely different thereby necessitating completely different visual data presentation metaphors as illustrated in Fig. 2a and Fig. 2b. The methods for programming these systems bear little resemblance to general purpose languages like C++ and Python as shown in Fig. 2c and Fig. 2d. Unlike the practices we saw during the serial-vector-massively parallel (MPP) evolution, there is no straightforward path for taking a serial classical code and transforming it so that it will run on quantum platform. For additional background, Sidebar §7 provides a set of resources that are useful in learning more about QC.

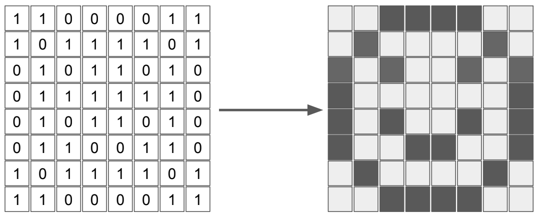
In QC, information is represented in the *quantum state* of *quantum bits*, or *qubits*. Unlike in the classical world where the state of a *bit* is either a value of 0 or 1, the state of a *qubit* is more complex owing to the nature of the physics of quantum mechanics upon which it is constructed. Intuitively, the *quantum state* of a *qubit* may be thought of as a combination of potential outcomes rather than a discrete binary value.

In the quantum world, we represent the *quantum state* of a *qubit* using *Dirac* notation, which is the standard way of describing states in quantum mechanics. Two possible states of a *qubit* are $|0\rangle$ and $|1\rangle$, which are known as *computational basis states* that correspond to the classical bit states of 0 and 1. Unlike classical bits, a *qubit* may be in a state other than $|0\rangle$ and $|1\rangle$. It is possible to form a linear combination of states known as *superpositions* [4]:

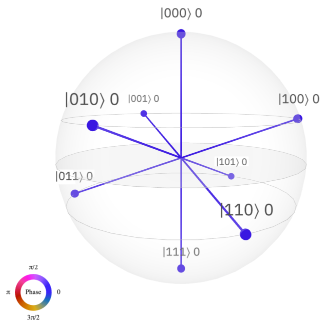
$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1}$$

where α and β are *amplitudes* and $\{\alpha, \beta\} \in \mathbb{C}^2$. Born's rule states that in a superposition of states, the modulus squared of the amplitude of a state is the probability of that state resulting after measurement. Furthermore, the sum of the squares of the amplitudes of all possible states in the superposition is equal to 1 [5]. Therefore, for the state $|q\rangle$ in Eq. 1, we have $|\alpha|^2 + |\beta|^2 = 1$ and the probability of measuring state $|0\rangle$ state is $|\alpha|^2$ and state $|1\rangle$ is $|\beta|^2$.

A useful analogy for a *quantum state* is a coin toss.



(a) Visualizing the state of 64 classical bits



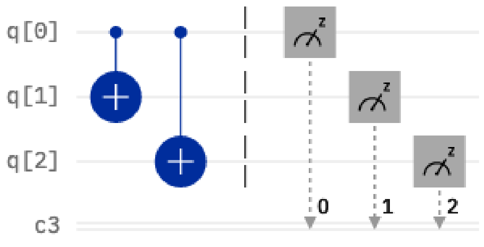
(b) Visualizing the quantum state of 3 quantum bits (*qubits*) in superposition.

```
in_data = [0b000, 0b001, 0b010, 0b011,
           0b100, 0b101, 0b110, 0b111]

conditional_flip = lambda x: ~(x&0b110) | x&0b1 \
    if x & 0b1 else x

out_data = [ conditional_flip(i)  for i in in_data]
```

(c) Classical (python) code to flip bits 1 and 2 if bit 0 is set.



(d) Quantum code (circuit) to flip the state of qubits 1 and 2 if the state of qubit 0 is $|1\rangle$

FIGURE 2: Comparison of classical quantum state visualization and programs.

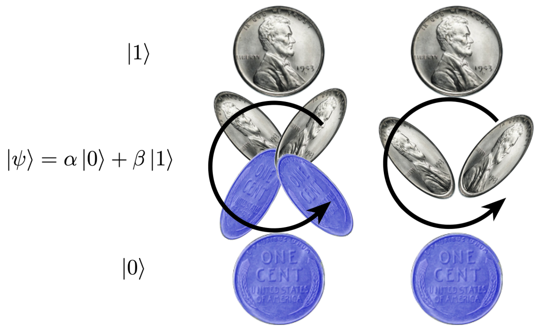
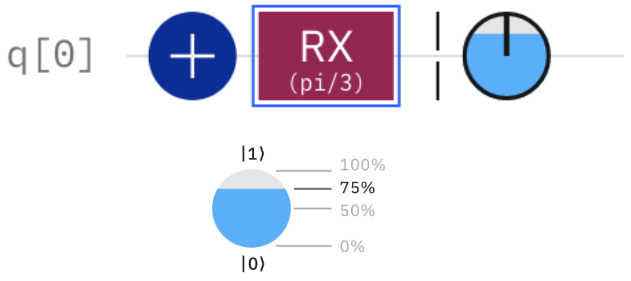


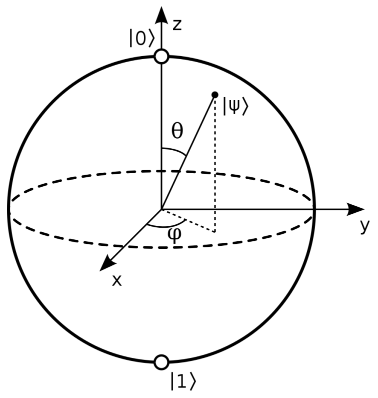
FIGURE 3: Using a coin toss analogy, when the coin is at rest, we have a 100% probability of a heads state $|1\rangle$ in the top row or a tails state $|0\rangle$ in the bottom row. When the coin is spinning (middle row), its state $|\psi\rangle$ is a combination of probabilities α and β. In the right column, while the probability of a heads is the same for both potential coin states in the middle, their *quantum phase* differs in that one head points more westerly and the other points more easterly.

While the coin is at rest and its state measured or observed, the coin is in either a heads or tails, $|1\rangle$ or $|0\rangle$ respectively as shown in the top and bottom rows of Fig. 3. On the other hand, while the coin is spinning in the air, the coin is in a state of *superposition* where its state is a combination of the heads and tails states as shown in the middle row of Fig. 3. In the quantum world, once the state of the *qubit* is observed or measured, the value we observe is either 0 or 1. Once we measure or observe a *qubit* its quantum state collapses and the α and β are no longer accessible.

In addition to the idea of a probability of a heads or tails state, there is also the notion of *quantum phase*. Consider the right column, middle row of Fig. 3 where there are two possible configurations of a coin in a being-tossed state. Both of these configurations appear to have the same probability of being a heads but they differ in *phase* since one head points in a more westerly direction and the other points in a more easterly direction. While there are two different types of quantum phase, a *global phase* and a *relative phase* [5], for the discussion here, the most important idea is that it is possible to have two (or more) quantum states that have identical probability amplitudes for a heads and tails state, but like the right side of Fig. 3, are completely different states. Quantum programs provide the means to manipulate the quantum state of



(a) A quantum circuit that transforms a *qubit* from an initial state of $|0\rangle$ first into $|1\rangle$ with a NOT gate (blue circle with a plus) then into a state of superposition by rotating the quantum state about the X axis with an RX gate by $\theta = \pi/3$. The phase disk on the right of the circuit shows the probability of a $|1\rangle$ state, and the legend below shows the probability in terms of percentage.



(b) The Bloch Sphere maps the 3-dimensional quantum state of a single *qubit* to a point on the unit sphere (Image source: Wikimedia Commons).

FIGURE 4: Two different visualization methods for seeing the quantum state of single qubit.

single or multiple entangled *qubits* in ways that change either the *probability amplitudes* of observing a $|0\rangle$ or $|1\rangle$ or the phase, or both. See Sidebar 8 for more information.

Contemporary quantum platforms are not quite yet as usable as classical platforms due to the fact the underlying hardware is still maturing. Unlike classical bits comprised of flip-flops that retain their state so long as voltage is present [6], *qubits* are able to reliably retain state for only a short amount of time, typically on the order of hundreds of microseconds [7]. Present day systems are referred to as *Noisy Intermediate Scale Quantum* (NISQ) platforms because the quantum state in *qubits* and gates between them is subject to interference from external sources like heat and energetic particles in the environment [8]. As a result, we typically will run a quantum program thousands of times to obtain a distribution of outcomes from a given computation.

Programming and the design of QC algorithms are fundamentally different than the classical world. In the classical world, we read in data from disk then perform operations on it. In the QC world, most programs create an initial set of quantum states through a process of *superposition* so that a *qubit* is in a state that looks like Eq. 1, and then the quantum program performs operations on the quantum states, and then finally measures the *qubits* to obtain the final answers. The operations on quantum states must conform to a specific set of rules and constraints that may be thought of as applying a series of rotation matrices to the quantum state as further explained in Sidebar §8. Transferring classical data into quantum systems and then back out again is an area of ongoing research and engineering. There are several useful surveys of contemporary environments and approaches for creating and running quantum programs (c.f. [5]).

3. How Visualization Helps QC

Visualization plays a central role in facilitating deeper understanding of QC. In the subsections that follow, our examples focus on visualization of the quantum state for single and multiple *qubit* systems using methods from contemporary QC programming environments. These examples are not an exhaustive set of such visualization techniques. Additionally, visualization has played a role in helping to understand performance data associated with quantum programs. Such data include measures of runtime performance, reliability as a function of program size, and more. A survey of these types of visualizations is beyond the scope of this article.

3.1. Visualizing the Quantum State of Individual *Qubits*

In the classical world, since the state of a *bit* is either 0 or 1, visualizing the state of a classical bit is straightforward using any number of scalar information visualization methods. Of historical importance is the role played by visual presentation of the state of groups of classical bits in many contexts, including presentation of type via bitmapped fonts, glyphs (see Fig. 2a), and animated characters in video games. The visualization of groups of classical bits is an integral part of the modern computational landscape on virtually all display devices.

Visualizing the quantum state of a *qubit* requires a different approach due to the nature of the information in the quantum state. One type of information is the probability of obtaining a $|0\rangle$ or $|1\rangle$ state upon measurement. As an example, consider the simple quantum circuit in Fig. 4a where *qubit* zero begins in state $|0\rangle$. Then, the state is inverted with a NOT gate (the "+" symbol) then transformed by the RX gate, which applies a rotation transformation about the X axis here in the amount of $\theta = \pi/3$. The result is that the quantum state has non-zero probabilities for states $|0\rangle$ and $|1\rangle$. At the rightmost part of the circuit is a small circle-shaped glyph that is partially filled with a shade of blue to indicate the probability of the *qubit* being in state $|1\rangle$. This particular glyph is known as a *phase disk* and is part of the IBM Quantum Composer suite of visualization tools [9].

The other type of information in the quantum state is *quantum phase*. To obtain the value of phase, we can rewrite Eq. 1 as

$$|\psi\rangle = cos\frac{\theta}{2}\,|0\rangle + e^{i\psi} sin\frac{\theta}{2}\,|1\rangle \qquad (2)$$

Due to the fact that $|\alpha|^2 + |\beta|^2 = 1$, the real numbers θ and ψ define a point on the surface of a unit-three dimensional sphere. This representation is often visualized using a glyph known as a Bloch Sphere [4], [10] and is shown in Fig. 4b. In the Bloch Sphere, the North pole corresponds to state $|0\rangle$ and the South pole to $|1\rangle$. The θ angle represents the relationship of the probability amplitudes of α, β, while the angle ψ is the phase of the quantum state. The phase disk glyph in Fig. 4a has a black line anchored at the center of the disk. Its orientation shows the phase of the quantum state like a compass.

3.2. Visualizing the Quantum State of Multiple *Qubits*

If we expand our earlier coin-toss scenario from a single coin to *n* simultaneous coins, the size of the state space describing all possible permutations of heads and tails combinations grows exponentially. This idea applies to *qubits* as well: if there is one coin, or *qubit*, there are 2^1 possible basis states in this system $\{|0\rangle, |1\rangle\}$; if there are two coins, or *qubits*, there are 2^2 possible basis states in this system $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$; if there are *N qubits*, then there are 2^N possible basis states.

The exponential growth in the size of the quantum state space is where QC shows promise in being able to tackle problems of a scale not possible on classical computers. For example, a 300 *qubit* system can theoretically represent a quantum state of size 2^{300}, which is more than the estimated number of particles in the visible universe and is far larger than any classical system.

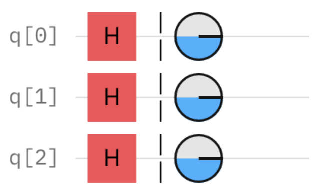
For visualizing the quantum state of multiple *qubits*, consider the example shown in Fig. 5. Three *qubits* in a quantum circuit are placed into states of superposition using a Hadamard gate, which has the effect of creating a state of superposition where there is an equal probabilities of states meaning there is an equal likelihood of measuring either a $|0\rangle$ and $|1\rangle$. This gate is one of the foundational building blocks of many quantum programs [5]. For this $n = 3$ *qubit* system, the size of the quantum state is $2^3 = 8$ and the bar chart Fig. 5b shows the probability of each of the 2^3 basis states. In this case, all the bars are the same height indicating that each of these basis states $\{|000\rangle, |001\rangle, \ldots, |111\rangle\}$ have equal probabilities.

Fig. 5c shows what is known as a *Q-sphere* presentation of quantum state [9]. Unlike the Bloch Sphere, which shows the state of a single *qubit*, the Q-sphere shows the global quantum state by associating each of the 2^n basis states from *n qubits* with a point, or node, on the surface of the sphere. The radius of each node on the sphere's surface is proportional to the probability of that basis state. In Fig. 5c, since each basis state has equal probability, all the nodes are the same size, though the perspective foreshortening results in the nodes appearing to be of varying sizes. The color of each node indicates quantum phase of that particular basis state. The circular color legend on the lower left provides the visual association between color and phase.

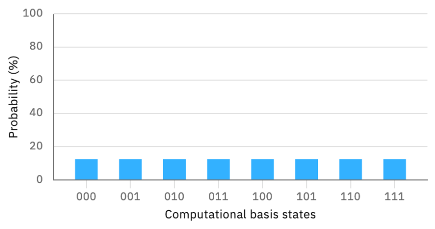
In terms of node positioning on the Q-sphere in Fig. 5c, nodes are laid out so that the $|000\rangle$ and $|111\rangle$ states are at the north and south pole, respectively. Going from the north pole and heading southward, each successive latitude has basis states with a greater number of $|1\rangle$ states and the latitude of a given basis is determined by its Hamming distance from the $|000\rangle$ state. In this example, there are three states with a Hamming distance of 1 from $|000\rangle$, namely $\{|001\rangle, |010\rangle, |100\rangle\}$, and three states with a Hamming distance of 2 from $|000\rangle$, namely $\{|011\rangle, |101\rangle, |110\rangle\}$.

4. How QC Helps Visualization

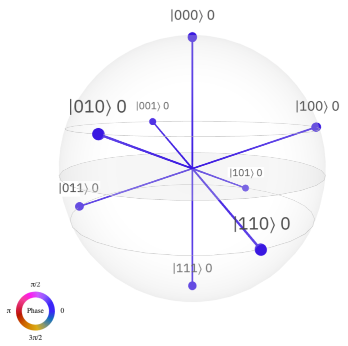
In 2001, A. Glassner presented a 3-part column in IEEE CGA [11] that provided an excellent introduction and overview to QC and also began exploration of how QC might be used in graphics and visualization. Langzagorta et al., [12] suggested two broad



(a) A quantum circuit where each *qubit* is put into a state of superposition. The phase disk shows equal probabilities of each state in each *qubit*.

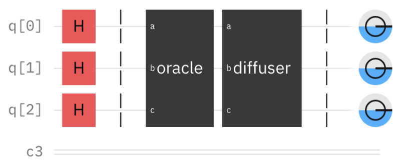


(b) Bar chart showing the (equal) probabilities of each of the quantum states.

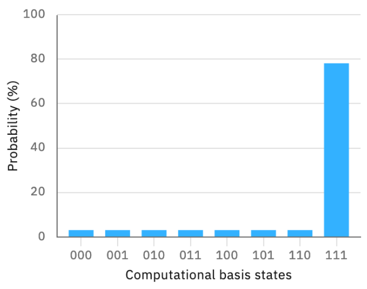


(c) Q-sphere visualization showing the probabilities and phase of each of quantum states.

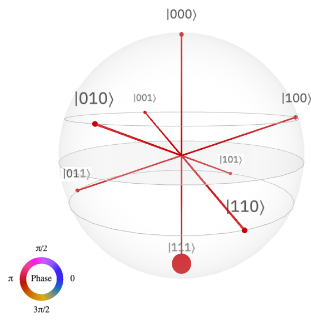
FIGURE 5: An $n = 3$ *qubit* circuit (left) where the quantum state of each *qubit* is placed in a state of superposition resulting in an overall quantum state of size 2^n. The probabilities of each of the 2^n states is shown via a bar chart (middle) and in a Q-sphere format (right).



(a) The two-stage circuit consists of unitary transformations that boost the probability of the desired solution state $|111\rangle$.



(b) The histogram shows the probability distribution of all potential quantum states with state $|111\rangle$ having the highest probability.



(c) The Q-sphere visualization shows that all quantum states have the same phase, but state $|111\rangle$ has a higher probability than all the others.

FIGURE 6: Quantum circuit and visualization of state for Grover's Search algorithm.

approaches when considering ways to leverage QC for graphics. The first is to identify portions of a classical algorithm that are similar to known methods in QC. The examples we present are works that pursue this approach. The second and much more difficult approach is to find structure in the classical algorithm that can fully exploit the quantum nature of quantum platforms, namely the ability to perform an operation in parallel across all dimensions of a quantum state.

We begin this section with a brief description of a commonly used approach in the design of QC methods where a quantum operation is applied in parallel to all basis states of the quantum state. Then, we present some examples of visualization and related methods that take advantage of QC.

4.1. QC Algorithms and the Quantum State

QCs power arises from the exponential growth in the size of the quantum state with increasing numbers of *qubits* combined with the ability to apply an operation simultaneously to all quantum states leading to an exponential speedup for certain types of computations. Many quantum programs begin by placing individual *qubits* into states of superposition with operations like the Hadamard gate [13] as shown in Fig. 5a. Next, the quantum program operates *in parallel* on all quantum basis states using a function U_f that is a *unitary* implementation of some classical function f.

Unitary operators have special properties; they may be thought of as rotation matrices that manipulate the quantum state (see Sidebar 8). The input to U_f is $|x\rangle$, and its output is $U_f\,|x\rangle$, which is U_f evaluated f on all possible values of the input x. The quantum circuit for such an operation is shown in Fig. 6a where two groups of U_f unitary operators, shown as *oracle* and *diffuser*, operate in parallel on all quantum states. The impact of this type of computational model is profound and captures the essence of the power of QC: we can evaluate some $f(x)$ over all 2^n quantum states of n entangled *qubits* in a single operation. Not shown in Fig. 6a is how the quantum program will create a state of entanglement between one or more *qubits* through the use of unitary operations like the Controlled NOT gate (CNOT) [13] which conditionally inverts the state of one *qubit* based upon the state of another.

One of the foundational quantum algorithms is Grover's Search (c.f. [13]) for finding a item or items in an unordered list of length N. A classical implementation of an unstructured search requires on average $O(N/2)$ steps since on average about half of the items in the set would need to be examined. The circuit for Grover's Search is shown in Fig. 6a for a 3-*qubit* system. The intention of the two unitary transforms *oracle* and *diffuser* is to boost the probability of solution states and reduce the probability of non-solution states. Those probabilities are shown as a histogram in Fig. 6b and in Q-sphere format in Fig. 6c.

The process of finding the set of unitary operations that manipulate the 2^n basis states of an $n-$*qubit* system is a non-trivial process. For this reason, nearly all work that has attempted make use of QC for graphics and visualization has focused on mapping a portion of a classical algorithm to make use of Grover's Search.

4.2. QC and Visualization

Glassner's 2001 column [11] provided an overview of how one might create a quantum radiosity rendering algorithm. The central idea is to encode subsets of the scene into *qubits* storing information like the amount of light being emitted, the amount of incident light, the amount of reflected light and so forth. Then, once the state of the scene is encoded into *qubits*, to use a variant of Grover's Search to find the configuration with minimum energy, which corresponds to a solution to the radiosity equation. This notional overview was impractical since it was estimated to require hundreds of millions of bits assuming a 5,000 surfaces and 32-bit resolution for encoding information.

Lanzagorta et al., 2003 [12] suggest using Grover's to accelerate searches of objects in a graphics database, a z-buffer algorithm to determine which object is closest to a viewer, and a ray-object intersection methodology. More recent work in this space [14] builds on this idea and includes an implementation that runs on a 20-*qubit* system. The key idea is to encode the scene and ray parameters into the quantum state then leverage Grover's to find ray-object intersections. This idea is illustrated in Fig. 1a. While Grover's offers computational advantages for unstructured search, encoding classical data on into quantum state is a non-trivial process and must take into account the limitations of the size of the circuit that may be reliably executed on a a given quantum platform.

A quantum version of the RANSAC algorithm [15] uses a Grover's variant to reduce computational complexity from $O(N)$ to $O(\sqrt{N})$. RANSAC fits models to data and uses many searches to find data that satisfy a set of criteria, e.g., maximums. This type of algorithm has uses in geometric modeling and vision problems, including camera pose estimation, structure from motion, and shape detection.

There are fewer examples of work in visualization that follow the more difficult approach where some or all of a classical algorithm is refactored for the quantum environment. One example that takes steps in that direction is in the area of pixel supersampling [16] using multiple quantum processing stages to produce what appear to be better visual results than an approach based on classical Monte Carlo sampling. These different results are a direct result of how a quantum sampling approach results in a significantly different probability distribution compared to a classical approach (Fig. 1b).

5. The Road Ahead

Despite the examples here that suggest QC may be able to tackle larger problems and produce better results than are possible on classical platforms, there are a number of significant challenges.

From a visualization-helps-quantum perspective, the examples we have shown in this article suggest that most visualization methods are useful for only a small number of *qubits*. The existing set of methods and approaches do not seem promising for use on configurations with dozens or hundreds of *qubits*, particularly considering that the size of the quantum state scales exponentially with the number of *qubits*. Additionally, there is much less work in the area of visualizing the behavior of large, entangled systems of *qubits*.

For the quantum approaches we have described, there is a common theme: initialize some *qubits*, place them into superposition, entangle them, then apply a unitary operator U_f to all quantum states at once. What is missing from this picture is somehow moving classical data into the quantum machine and then obtaining answers that are then returned to the classical world. A significant issue when designing and implementing quantum algorithms that process classical data is the *data encoding problem*, which refers to how data is encoded in the quantum state of a *qubit* register. Some recent work explores approaches for encoding gray-level classical pixel data on quantum platforms [17] through a process using gate-level operations to set up the quantum state to known values. Current quantum platforms have a limited number of *qubits* and have limited *quantum volume*, a term that refers to the number of program steps (gates) that can reliably run across *qubits*. Data encoding methods are not "for free": they increase the volume of the quantum circuit. Since current quantum platforms are of $O(10) - O(100)$ *qubits* (and quantum annealing systems like D-Wave's approaching $O(1000)$ of *qubits*), it is unclear how we will go about encoding "large data sets" for the purpose of analysis and visualization on quantum platforms. We will need to rethink our "large data visualization" approaches for quantum platforms.

In his 2021 Turing Award presentation, J. Dongarra suggested that QC platforms will likely be accelerators on future systems, at least in the near- to medium-term. These quantum-based accelerators will co-exist with classical computing infrastructure resulting in a hybrid system. It may be the case that the computational workload will be partitioned across classical and quantum machines much like we see a partitioning of workloads across CPU and GPU. Such *hybrid architectures* will likely figure prominently in successful approaches for quantum-leveraging visualization and graphics methods in the near- and medium-term.

6. Conclusion

The field of QC is "up and coming", and has potential to tackle problems that far exceed what is possible on classical platforms. It is a disruptive technological change that will require significant rethinking of how we formulate and implement algorithms. The focus of this article has been to highlight some synergies between visualization and QC both in terms of how visualization helps with understanding QC and how visualization might benefit from use of QC formulations of problems.

7. Sidebar: Getting Started with QC

If you are new to the world QC and would like to learn more, there are a number of useful resources providing additional breadth and depth beyond the scope of the article. A good start is Andrew Glassner's 3-part column in IEEE CGA from 2001, which provides a well rounded overview of QC along with some speculation of how it might be used in graphics [11].

Several excellent learning resources are developed and distributed online and at no cost by quantum platform companies in the interest of increasing awareness and expertise in QC topics. Some examples include IBM Quantum and their Qiskit platform [18], Xanadu.ai and their PennyLane platform [19], Google's Quantum AI and their Cirq platform [20]. This set is not exhaustive nor complete, there are numerous additional resources accessible via the web.

Online tutorials. Qiskit resources [21] include a "QC in a nutshell" as a good starting point along with a link to a YouTube channel with many useful videos that present quantum topics in an accessible way.

Coding environments. Some good resources for learning how to write quantum programs are online at the Qiskit [21], Xanadu [19], and Cirq [20] sites. These provide information about how to create quantum circuits, create states of superposition and entanglement, measure and visualize quantum state, and execute quantum programs on simulators and actual quantum hardware. The examples cited here are accessible as freely accessible Python modules. Code development consists of problem setup in Python using the specific capabilities of a particular QC coding platform.

Cloud-based platforms. With your first QC programs you will likely be running them via Python on a local *quantum simulator* that is included as part of the QC coding environment. When you are ready to try running on actual hardware, there are several different cloud-based providers of quantum platforms. Some of these systems are accessible at no cost while others use a fee-based access model. IBM Quantum [7] provides access to about two dozen systems ranging in size from 5 to 433 *qubits* and is well integrated with Qiskit's runtime system. Microsoft's Azure Quantum service [22] and Amazon's Braket [23] are both gateways to quantum platforms from IonQ, Quanntinuum, Rigetti, and others ranging in size from 10-80 *qubits* and supports running codes written in Qiskit, Cirq, PennyLane, and others.

8. Sidebar: Quantum Operations as Rotation Matrices

Quantum operations on n qubits $|\psi\rangle$ are described in the langauge of mathematics as $2^n \times 2^n$ *unitary matrices*. In particular, the quantum state resulting after applying a quantum operation U to a quantum state $|\psi\rangle$

is given by matrix multiplication

$$\left|\psi'\right\rangle = U\left|\psi\right\rangle \ . \tag{3}$$

Properties of quantum computation are consequences of unitary computation. For example, quantum states are closed under quantum operations since a unitary matrix U preserves the norm of an input vector so that $|\psi'\rangle$ also has unit norm. As another example, a unitary matrix has the property that $UU^\dagger = U^\dagger U = I$ where $U^\dagger$ is the conjugate transpose of U. This means that every quantum operation is *invertible* or *reversible* since the definition implies that $U^{-1} = U^\dagger$. Notably, classical computation does not have this property.

In practice, one cannot write a quantum program by specifying all the coefficients in the unitary matrix U since it is of size $2^n \times 2^n$, which is exponential in the number of qubits n. Instead, quantum hardware devices supply primitive *gates* from a finite set of gates which can be sequenced together to form a *quantum circuit*. Each quantum circuit corresponds to a unitary matrix. Thus a quantum circuit enables us to express a quantum computation in a compact form that is physically implementable without explicitly specifying all $2^n \times 2^n$ coefficients. As with classical computing, there are universality results in quantum computing that show that every unitary matrix can be approximated arbitrarily accurately as a quantum circuit provided that we have a universal set of quantum gates. To build some intuition for how a quantum operation (quantum circuit) works, let us highlight how it is possible to think of them as "generalized rotations".

Consider a 1-qubit system so that we can visualize the quantum state on a Bloch sphere. As a reminder, a 1-qubit system can be specified as

$$|\psi\rangle = cos\frac{\theta}{2}\,|0\rangle + e^{i\phi} sin\frac{\theta}{2}\,|1\rangle \tag{4}$$

on the Bloch sphere where θ is the polar angle and ϕ is the azimuthal angle. Since every 1-qubit quantum state lies on the Bloch sphere, we can visually reason that a quantum operation can be expressed as a 3-dimensional rotation in Euclidean space. Because we can use *Euler angles* to represent all such 3-dimensional rotations, we further reason that we need 3 degrees of freedom to specify a quantum operation. At this point, the astute reader may observe an issue with our description so far that serves to highlight the quirks of quantum computation. In particular, we seem to be missing 1 degree-of-freedom in our description since 2×2 unitary matrices have 4 degrees of freedom whereas 3 dimensional rotation matrices only have 3 degrees of freedom.

The missing degree of freedom is the *global phase* ϕ_g and can be seen by noting that $e^{i\phi_g}|\psi\rangle$ is indistinguishable from $|\psi\rangle$ by measurement. Graphically, we can see this since measurement in the computational basis gives either $|0\rangle$ or $|1\rangle$ in proportion to the distance of the vector $|\psi\rangle$ to $|0\rangle$ or $|1\rangle$ so that any rotation in the *xy*-plane of the Bloch sphere does not change the distance. The global phase is usually omitted since it is indistinguishable and the *relative phase* ϕ is what is kept. This leads to an idea that quantum algorithms can take advantage of which is to store information in the relative phase (e.g., see quantum phase estimation which is the building block of Shor's algorithm, c.f. [5]).

The analogy between generalized Euler angles and unitary matrices can be made formal. On 1 qubit systems, every unitary matrix can be written as

$$U = e^{i\phi_g} e^{iZ\alpha_1} e^{iY\alpha_2} e^{iZ\alpha_3} \tag{5}$$

where $\alpha_1, \alpha_2, \alpha_3$ are generalized Euler angles, ϕ_g is a global phase, and Y and Z are Pauli matrices that correspond to Pauli gates [4]. Moreover, explicit generalized Euler angle parameterizations for general unitary matrices of all dimensions are given by Tilma et al. [24]. Although the n-qubit case is difficult to visualize, it does shed some light on the intricacies of quantum operations, and in particular, the concept of *entanglement*.

Conceptually, entanglement enables local computations on certain qubits to affect other qubits. This is unlike classical computation again, where computations on bits is local to the bits that are operated on. The phrase "spooky action at a distance" commonly attributed to Einstein describes this counterintuitive (at least classically) consequence of entanglement. The generalized rotation view of a quantum operation helps make sense of this situation. More concretely, we might imagine that a sequence of rotations, each restricted to a single plane as with Euler angles so that at most two qubits are operated on at any given time, can still globally affect the entire quantum state, provided that the rotations are sequenced accordingly. Naturally, we are pushing the visual analogy a bit far in the context of this article to given intuition and not as a substitute for the underlying physics and mathematics.

Acknowledgment

This research was partially supported by the U.S. Department of Energy (DOE) under Contract No. DE-AC0205CH11231, through the Office of Advanced Scientific Computing Research (ASCR) Exploratory Research for Extreme-Scale Science Program. The images of quantum circuits and *qubit* state visualizations

were created using the Qiskit-Terra software [25] and IBM's Quantum Composer [9].

9. REFERENCES

1. X. Lu and H. Lin, "A framework for quantum ray tracing," 2022. [Online]. Available: https://arxiv.org/abs/2203.15451
2. B. H. McCormick, T. A. DeFanti, and M. D. Brown, "Visualization in Scientific Computing," *Computer Graphics*, vol. 21, no. 6, Nov. 1987, ACM SIGGRAPH: New York.
3. J. Kempe, "Quantum random walks: An introductory overview," *Contemporary Physics*, vol. 44, no. 4, pp. 307–327, 2003. [Online]. Available: https://doi.org/10.1080/00107151031000110776
4. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. USA: Cambridge University Press, 2011.
5. J. D. Hidary, *Quantum Computing: An Applied Approach*. Springer, 2019.
6. D. Patterson and J. Hennessy, *Computer organization and design : the hardware/software interface*, sixth edition. ed. Kidlington, Oxford ;: Morgan Kaufmann, 2021 - 2021.
7. I. Corporation, "IBM Quantum," online at https://quantum-computing.ibm.com/, last accessed Aug. 2023.
8. J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79
9. IBM, "IBM Quantum Composer User Guide," online at https://quantum-computing.ibm.com/composer/docs/iqx/, last accessed June 2023.
10. F. Bloch, "Nuclear induction," *Phys. Rev.*, vol. 70, pp. 460–474, Oct 1946. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.70.460
11. A. Glassner, "Quantum computing, part 1," *IEEE Comput. Graph. Appl.*, vol. 21, no. 4, p. 84–92, jul 2001. [Online]. Available: https://doi.org/10.1109/38.953464
12. M. O. Lanzagorta, R. B. Gomez, and J. K. Uhlmann, "Quantum rendering," in *Quantum Information and Computation*, E. Donkor, A. R. Pirich, and H. E. Brandt, Eds., vol. 5105, International Society for Optics and Photonics. SPIE, 2003, pp. 128 – 136. [Online]. Available: https://doi.org/10.1117/12.486410
13. R. LaPierre, *Introduction to Quantum Computing*. Springer, 2021.
14. L. P. Santos, T. Bashford-Rogers, J. Barbosa, and P. Navrátil, "Towards quantum ray tracing," 2022. [Online]. Available: https://arxiv.org/abs/2204.12797
15. S. Caraiman and V. I. Manta, "New applications of quantum algorithms to computer graphics: The quantum random sample consensus algorithm," in *Proceedings of the 6th ACM Conference on Computing Frontiers*, ser. CF '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 81–88. [Online]. Available: https://doi.org/10.1145/1531743.1531757
16. E. R. Johnston, "Quantum supersampling," in *ACM SIGGRAPH 2016 Talks*, ser. SIGGRAPH '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2897839.2927422
17. M. G. Amankwah, D. Camps, E. W. Bethel, R. Van Beeumen, and T. Perciano, "Quantum pixel representations and compression for *N*-dimensional images," *Scientific Reports*, vol. 12, no. 1, p. 7712, May 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-11024-y
18. Qiskit contributors, "IBM Quantum Learning," 2023, online at https://qiskit.org/learn, last accessed Aug. 2023.
19. C. Albornoz and et al., "Xanadu Quantum Cookbook," 2023, online at https://codebook.xanadu.ai/, last accessed Sep. 2023.
20. G. Q. Team, "Cirq: An open source framework for programming quantum computers," 2023, online at https://quantumai.google/cirq, last accessed Aug. 2023.
21. Qiskit contributors, "Qiskit Tutorials: An Open-source Framework for Quantum Computing," 2023, online at https://qiskit.org/documentation/tutorials.html, last accessed Aug. 2023.
22. Microsoft, "Quickstart: Submit a circuit with Cirq to Azure Quantum," 2023, online at https://learn.microsoft.com/en-us/azure/quantum/quickstart-microsoft-cirq.
23. Amazon, " Amazon Braket," 2023, online at https://aws.amazon.com/braket/, last accessed Aug. 2023.
24. T. Tilma and E. Sudarshan, "Generalized euler angle parameterization for u (n) with applications to su (n) coset volume measures," *Journal of Geometry and Physics*, vol. 52, no. 3, pp. 263–283, 2004.
25. Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023, online at https://qiskit.org/ and doi:10.5281/zenodo.2573505, last accessed Sep. 2023.

E. Wes Bethel is an Associate Professor of Computer Science at *San Francisco State University* and a Research Affiliate at *Lawrence Berkeley National Laboratory*. Contact him at ewbethel@sfsu.edu.

Mercy G. Amankwah is a Computer Science PhD student at *Case Western University*. Contact her at mercy.amankwah@case.edu.

Jan Belewski is a Computer Systems Engineer at the *National Energy Research Scientific Computing Center* at *Lawrence Berkeley National Laboratory*. Contact him at balewski@lbl.gov.

Roel Van Beeumen is a Research Scientist at *Lawrence Berkeley National Laboratory*. Contact him at rvanbeeumen@lbl.gov.

Daan Camps is a Computer Systems Engineer at the *National Energy Research Scientific Computing Center* at *Lawrence Berkeley National Laboratory*. Contact him at dcamps@lbl.gov.

Daniel Huang is an Assistant Professor of Computer Science at *San Francisco State University* and a Research Affiliate at *Lawrence Berkeley National Laboratory*. Contact him at danehuang@sfsu.edu.

Talita Perciano is a Research Scientist at *Lawrence Berkeley National Laboratory*. Contact her at tperciano@lbl.gov.