A Quantum Algorithm for Ray Casting using an Orthographic Camera

Carolina Alves
Universidade do Minho
Braga, Portugal
a75610@alunos.uminho.pt

Luís Paulo Santos
Universidade do Minho & INESC-TEC & Quantum Software Engineering, INL
Braga, Portugal
psantos@di.uminho.pt

Thomas Bashford-Rogers
University of West of England
Bristol, United Kingdom
Tom.Bashford-Rogers@uwe.ac.uk

***Abstract*—Quantum computing has the potential to provide solutions to many problems which are challenging or out of reach of classical computers. There are several problems in rendering which are amenable to being solved in quantum computers, but these have yet to be demonstrated in practice. This work takes a first step in applying quantum computing to one of the most fundamental operations in rendering: ray casting. This technique computes visibility between two points in a 3D model of the world which is described by a collection of geometric primitives. The algorithm returns, for a given ray, which primitive it intersects closest to its origin. Without a spatial acceleration structure, the classical complexity for this operation is $O(N)$. In this paper, we propose an implementation of Grover's Algorithm (a quantum search algorithm) for ray casting. This provides a quadratic speed up allowing for visibility evaluation for unstructured primitives in $O(\sqrt{N})$. However, due to technological limitations associated with current quantum computers, in this work the geometrical setup is limited to rectangles and parallel rays (orthographic projection).**

***Index Terms*—quantum computing, ray casting, Grover's algorithm, complexity**

I. Introduction

The field of quantum computing has registered huge developments over the last few years, raising the perspective for this alternative computing paradigm to become practical and advantageous in the medium term.

Richard Feynman is believed to have been the first to propose using quantum mechanics as a model for computation [1]. Theoretical results soon followed, with, for example, the proposal of the quantum Turing machine by Deutsch in 1985 [2] and algorithms that lie in the heart of most quantum programs, including Grover's for unstructured searching [3], Shor's for prime factorization [4] and the Quantum Fourier Transform [5].

More recently, technological advances made it possible for some companies to make quantum computers available to the community. These machines still impose quite demanding limitations on the size and execution time of the problems they can solve, due to limited coherence times, reduced number of qubits and significant error rates at the gates level. Nevertheless, the number of researchers, practitioners and companies experimenting with prototype applications to solve an ever growing variety of problems has been increasing exponentially. This rising interest on quantum computing is largely supported by an huge investment both from private companies and research funding institutions [6], [7].

An n-qubits quantum computation evolves on an exponential state space, representing at each instant a linear superposition of 2^n states. Each computing step leads the system to another superposition, acting on the 2^n states simultaneously without any resources replication. On the other hand, a classical n-bits system represents at each instant a single state out of the 2^n possible states and a computing step leads to a single subsequent state. It is this exponential quantum parallelism that holds the promise of the quantum advantage over classical computing. For some problems quantum algorithms have been found which exhibit better complexity than classical alternatives, allowing for the efficient computation of otherwise intractable problem sizes. Grover's algorithm [3] for searching on a N elements unstructured domain exhibits $\mathcal{O}(\sqrt{N})$ time complexity, as opposed to $\mathcal{O}(N)$ in the classical setting: in the worst case all N elements of the domain have to be verified.

Ray casting is a well known rendering technique, consisting on shooting a ray from a point along a given direction to determine whether i) that direction is occluded (i.e., the ray intersects at least one geometric primitive), or ii) what is the visible geometric primitive along the ray's direction (i.e., among all primitives intersected by the ray which one is nearer to the ray's origin). If there are N geometric primitives and if these are not ordered on any manner, then classical ray casting requires testing intersection against all primitives, with complexity $\mathcal{O}(N)$.

This paper demonstrates the application of Grover's algorithm to ray casting, addressing both the occlusion and the visibility problems, with worst-case complexity $\mathcal{O}(\sqrt{N})$. Even though using Grover's for this purpose has been suggested before [8]–[10], no implementations and no real results were, to the best of the authors' knowledge, ever presented. We

This work was partially financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia – within project: UID/EEA/50014/2019. This work was partially funded by SmartEGOV/NORTE-01-0145-FEDER-000037, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (EFDR).

propose concrete implementations and present results obtained both on a simulator and on a real quantum computer. Advantages and limitations of the proposed approach are discussed. All experiments, both simulations and real quantum computer executions, were performed using IBM's quantum computing framework, Qiskit[1], and Tokyo, the IBM Q Network 20 qubits machine.

Current circuit-based quantum computers provide no functional units to perform even the most fundamental arithmetic operations to mathematical data types, such as integers or floating point numbers. In fact, the program consists essentially on a list of basic transformations, referred to as gates, to apply to the qubits. Supporting complex operations, such as non-integer number representations or trigonometric functions, requires implementing them at the gate level. Besides the obvious programming overhead, these additional gates would dramatically increase the circuit depth and the number of required qubits, pushing the limits well beyond what is possible with today's noisy and limited scale quantum computers. In order to avoid such overheads this paper deals only with integers, the view plane is contained within the Z=0 plane, all geometric primitives are axis-aligned rectangles parallel to the view plane and the rays are perpendicular to the view plane and parallel to the Z-axis (thus the orthographic projection); all coordinates ((x, y) for pixels and rays, (x, y, z) for points in space) are integers. These restrictions facilitate the development of the quantum program and make it possible to use current simulators and real machines. There are, however, no theoretical reasons why floating point numbers and trigonometric functions cannot be used. They just wouldn't be feasible on current hardware given the limitations imposed by the technology state of the art.

II. Quantum Computing: Short Overview

A. Fundamentals

The basic quantum unit of information is the qubit $|b\rangle$, which allows a linear superposition of two orthogonal basis states $|0\rangle$ and $|1\rangle$:

$$|b\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle,\ \alpha_0, \alpha_1 \in \mathbb{C},\ |\alpha_0|^2 + |\alpha_1|^2 = 1$$

With n qubits a superposition $|\Psi\rangle$ over $N = 2^n$ basis states can be created:

$$|\Psi\rangle = \sum_{i=0}^{N-1} \alpha_i|i\rangle,\ \alpha_i \in \mathbb{C},\ \sum_{i=0}^{N-1} |\alpha_i|^2 = 1$$

For a uniform superposition all basis states' weights are equal, $\alpha_i = \alpha = \frac{1}{\sqrt{N}}$. Any operation over a superposition will act simultaneously on the 2^n states, in what is referred to as exponential quantum parallelism. This means that applying a function $f(\cdot)$ once to superposition $|\Psi\rangle$, effectively results on a new superposition $|\Psi'\rangle$ containing all $N = 2^n$ values of $f(|\Psi\rangle)$.

[1]https://qiskit.org/

Even though the quantum computation evolves on an exponentially large state space, computing all solutions simultaneously, this space is not accessible. Upon measurement the superposition $|\Psi\rangle$ collapses onto one basis state $|i\rangle$, among all basis states included on $|\Psi\rangle$, with probability $|\alpha_i|^2$. Any posterior measurements will return the same basis state and the probability with which it was selected ($|\alpha_i|^2$) is also not accessible - all the information on the superposition is lost with the measurement and the register behaves now as classical data. The role of quantum algorithms is to maximize the probability of measuring the desirable states within a superposition and to postpone measurements (i.e., reading quantum data) until the last step of the algorithm.

The laws of Quantum Mechanics require that all operations performed on qubits are unitary and reversible. The former means that the norm of the vector of basis states coefficients α_i is maintained and is equal to 1 after the transformation; this is required since the vector of α_i's is in fact a probability distribution. Reversibility is a consequence of the unitary requirement and means that given the outputs of the transformation its inputs can be known: the computation is reversible. Reversibility has several consequences at the circuit level: all quantum gates, which operate over qubits, must have the same number of inputs and outputs; this often results in circuits with a larger number of gates and more qubits than would be required for their classical counterpart. Information is never destroyed, since the computation can be reversed. According to the Landauer's principle [11] the erasure of a bit of information corresponds to the dissipation of energy in the form of heat to the environment; quantum computing does not destroy information (up to the final measurements) and has thus the potential to be more power efficient than classical computing.

The quantum circuit model represents quantum programs as a series of gates applied to qubits. Each qubit is represented as a horizontal line and each gate is connected to the qubits it interacts with. Whereas with classical circuits signals flow along the circuit, with quantum circuits time flows along the circuit, from left to right – qubits do not flow, rather gate operations are applied onto the qubits (for example as microwaves used in some technologies). Gates can be written in matrix form, which are applied to the qubits. For example, the Hadamard H and X (NOT) gates can be expressed as:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, X = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \quad (1)$$

Figure 1 illustrates one such circuit: the qubits initial state is $|0\rangle$, converted into an uniform superposition using Hadamard gates; then $|q_0\rangle$ is negated (the coefficients of basis states $|0\rangle$ and $|1\rangle$ are swapped) and finally measurement gates are used, reading one of the basis states present in $|q_0q_1\rangle$ to classical register c, according to the respective probability (in this case 0.25, since the 4 states are equally probable).

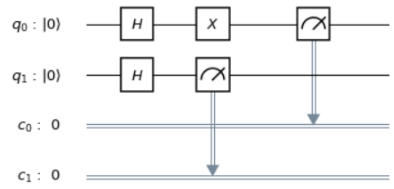


Fig. 1. An example quantum circuit.

B. Grover's Algorithm

For n qubits, and $N = 2^n$, let $f : \{0 \dots N-1\} \longrightarrow \{0, 1\}$, such that

$$f(x) = \begin{cases} 0 \leftarrow x \neq x^* \\ 1 \leftarrow x = x^* \end{cases}$$

The problem being addressed is finding the value(s) $x^* \in \{0 \dots N-1\} : f(x^*) = 1$. If nothing is known about $f(\cdot)$ then a classical approach must, in the worst case, evaluate the function $N - t + 1 \propto \mathcal{O}(N)$ times (where t is the number of different x^* satisfying $f(\cdot)$). Grover's algorithm will find x^* by executing the function only $\mathcal{O}(\sqrt{\frac{N}{t}})$ times, thus exhibiting a quadratic advantage.

Grover's algorithm starts by creating an uniform superposition over all basis states, $|\Psi\rangle$, by using Hadamard gates. $|\Psi\rangle$ can be expressed as a sum of two vectors: $|\Psi_1\rangle$ contains the basis states for which $f(\cdot) = 1$ (referred to as the "good" states), and $|\Psi_0\rangle$ contains the remaining. From Figure 2 left, it can be seen that $|\Psi\rangle = \sin\theta_a|\Psi_1\rangle + \cos\theta_a|\Psi_0\rangle$, thus the probability of measuring a state in $|\Psi_1\rangle$ is $\sin^2\theta_a$; this probability is also $\frac{t}{N}$, therefore $\sin\theta_a = \sqrt{\frac{t}{N}}$. The goal is to maximize the probability of measuring a state in $|\Psi_1\rangle$, which is achieved by applying multiple iterations of Grover's operator, Q. The first stage in this operator is the oracle O, which implements $f(\cdot)$ and negates the sign of the "good" states' coefficients according to $O|i\rangle = (-1)^{f(|i\rangle)} * |i\rangle$ (see Figure 2 center). Marking the "good" states does not change their probability; this is achieved by the diffusion operator, D, which performs a reflection over the average, depicted in Figure 2, right, as a reflection over the uniform superposition. After one application of $Q = DO$ the probability of measuring a "good" state has increased. It can be shown that after r iterations of Q the resulting state is $|\Psi^{(r)}\rangle = \sin((2r+1)\theta_a)|\Psi_1\rangle + \cos((2r+1)\theta_a)|\Psi_0\rangle$. Maximizing the probability of measuring a basis state in $|\Psi_1\rangle$ amounts to making $\sin((2r+1)\theta_a) \approx 1$, which for sufficiently large N results in $r \approx \frac{\pi}{4}\sqrt{\frac{N}{t}}$, i.e., $\mathcal{O}(\sqrt{\frac{N}{t}})$ Grover iterations are required, resulting on a quadratic speed up over the classical case. Performing more iterations than the ideal r actually reduces the probability of measuring a good state; successful application of Grover's depends on the ability to compute r and, consequently, on the previous knowledge of t.

III. Related Work

In 2001 Andrew Glassner discussed quantum computing on his notebook [8]–[10] and suggested applying Grover's algorithm to the Z-buffer and ray casting problems. For the former a superposition over all the polygons and respective depth could be created and Grover used to identify the minimum depth. For the latter a superposition over all spheres and respective parameters (a, b, c) could be created; equation $at^2 + bt + c = 0$ would then be solved simultaneously for all spheres and Grover used to locate the minimum positive t. For both problems a quadratic advantage could be obtained over an unordered classical search. Sadakane et al. [12] present a theoretical analysis of the complexity of Grover's, and a closely related minimum finding algorithm [13], applied to several geometric applications, including nearest neighbor, separation and function maximizing queries, geometric optimization problems (minimum enclosing ball, common intersection emptiness and convex hull computation) and intersection detection among large dimensional geometric elements. Simona Caraiman [14] also proposes applying Grover's algorithm to the Z-buffer and ray tracing problems. Additionally, she proposes addressing photon mapping, using Grover's for k-nearest neighbor queries, besides the photon shooting and the ray tracing stages. Lanzagorta and Uhlmann [15] propose using Grover's algorithm for a number of computational geometry problems, including nearest neighbour queries, object-object intersection, Z-buffering, ray tracing, radiosity and level of detail, among others. A detailed time and space complexity analysis is used to argue that quantum search is asymptotically advantageous over classical search for large domains (large N) and domain dimensionality $d >= 3$. Classical spatially ordering the search space in such conditions requires $\mathcal{O}(N \log^{d-1} N)$ space for a query time complexity of $\mathcal{O}(\log^d N)$, but since N and d are large, such spatial ordering in not feasible. Classical computing has thus to resort to unsorted search, which is $\mathcal{O}(N)$ in both space and time; quantum searching under such conditions is $\mathcal{O}(\sqrt{N})$ in time and $\mathcal{O}(N)$ in space, thus presenting a quadratic speedup.

All the above cited works maintain the discussion at the algorithmic and complexity analysis levels. No simulation, implementation or execution of the proposed algorithms is performed and no real experimental results are presented. This paper presents an actual implementation and real results for the ray casting problem.

A different problem is addressed by Johnston [16]. The author performs pixel level supersampling by resorting to the quantum amplitude estimation algorithm to numerically integrate sub pixel samples through a combination of a quantum algorithm and a classical lookup table. Experimental results show improvements, compared to those obtained with classical Monte Carlo integration for the same number of samples. These results are obtained using a simulator, a IBM five qubit machine, and a photonic quantum computer. This paper focuses on supersampling pixels based on a known underlying signal, whereas our work computes this signal from a scene

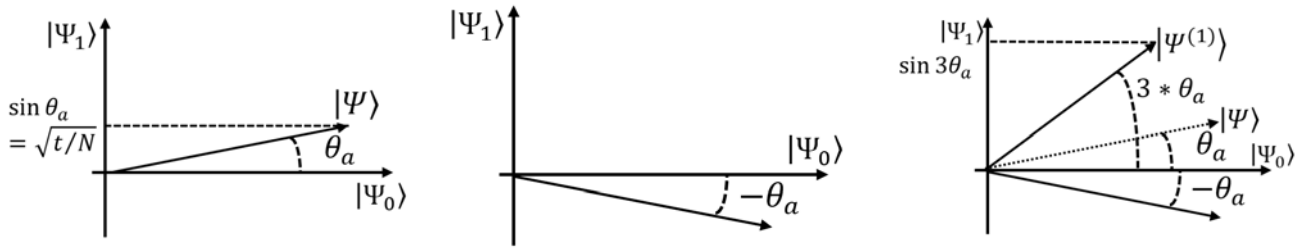


Fig. 2. Grover's algorithm. Left: Uniform superposition. Center: the oracle negates the sign of the "good" states' coefficients. Right: the diffusion operator reflects the quantum state over the uniform superposition; the probability of measuring a basis state in $|\Psi_1\rangle$ increased from $\sin^2\theta_a$ to $\sin^2(3 * \theta_a)$.

description.

IV. Ray Casting: single solution case

This section addresses the case where no more than one primitive projects to each pixel. For each primary ray, and thus for each quantum query, there is only zero or one solution. The important points are: i) the number of solutions is known ($t = 1$), which is fundamental for calculating the number of Grover's iterations ($t = 0$ will be handled as a failure to find a solution) and ii) the occlusion and visibility problems are the same, since the ray intersects only one primitive.

A. Geometric Setup

Current quantum computers do not include functional units to perform any kind of arithmetic operation over any data type (integers, floating point, etc.). To overcome such handicap an orthographic projection is used, with all coordinates being positive integers. The view plane is contained in plane $Z = 0$, primitives are rectangles parallel to the view plane (constant positive Z) and primary rays are parallel among themselves and the Z axis (X, Y constant); there is a one to one correspondence between pixels in the image plane and primary rays. Primitives are characterized by five parameters: $(min_X, max_X, min_Y, max_Y, Z)$; the latter is not relevant for the non-overlapping case addressed on this section, since at most one primitive projects onto a pixel (x, y). A ray intersects a primitive if $min_X \leq x \leq max_X \wedge min_Y \leq y \leq max_Y$, dispensing with any numerical calculations. Figure 3 presents

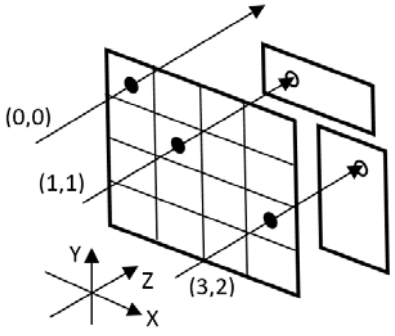


Fig. 3. Geometric setup for the non-overlapping case.

the non-overlapping setup for the 2D/3D case (Z is not relevant).

B. Quantum Algorithm

Grover's algorithm is used to identify intersections between a given ray (x, y) and the geometric primitives. A detailed description is given in Algorithms 1 and 2. These algorithms assume there are $N = 2^n$ primitives indexed from $0 \ldots N-1$ using a n qubit quantum register labelled as $|p\rangle$; quantum register $|c\rangle$ is prepared with the coordinates of the primitives, requiring nc qubits, $nc = \lceil \log_2(\max(max_X, max_Y)) \rceil$. Transforms $\{m|M\}\{X|Y\}$ generate the minimum (resp. maximum) X (resp. Y) coordinates of a primitive from its index; these are functions of the IDs of the primitives defined for each scene and the circuit is derived using basic Boolean simplification techniques.

Algorithm 1 Q_RayCast: quantum algorithm for ray (x, y), N primitives and r iterations

```
Superposition over the primitives IDs: |p⟩ = H^⊗n |0⟩^⊗n
for r iterations do
    Grover's oracle {Algorithm 2}
    Grover's diffusion operator
end for
ID ← Measure |p⟩
return ID
```

Algorithm 2 Oracle for ray (x, y) intersection with N primitives

```
|Hint⟩ ← |1⟩; |Vint⟩ ← |1⟩
min_X: |p⟩|c⟩ = mX|p⟩|0⟩^⊗nc
Negate Hint if x ≥ c; revert |c⟩
max_X: |p⟩|c⟩ = MX|p⟩|0⟩^⊗nc
Negate Hint if x ≤ c; revert |c⟩
min_Y: |p⟩|c⟩ = mY|p⟩|0⟩^⊗nc
Negate Vint if y ≥ c; revert |c⟩
max_Y: |p⟩|c⟩ = MY|p⟩|0⟩^⊗nc
Negate Vint if y ≤ c; revert |c⟩
|intersect⟩ = |Hint⟩ ∧ |Vint⟩
Flip the primitive's coefficient sign if intersect = 1: |p⟩ = (−1)^intersect * |p⟩
Revert |Hint⟩, |Vint⟩ and |intersect⟩
```

After execution of Algorithm 1 the probability of reading the ID of the primitive that intersects the ray, i.e., p_s, the suc-

TABLE I
r, p_s AND $p_{s,c=2}$ FOR GROVER'S WITH A SINGLE SOLUTION

	N					
	4	8	16	64	256	1024
$1/N$	0.250	0.125	0.063	0.016	0.04	0.01
r	1	2	3	6	12	25
p_s	1.000	0.945	0.961	0.997	1.000	0.999
$p_{s,c=2}$	1.000	0.997	0.999	1.000	1.000	1.000

cess probability, is $sin^2((2r+1)\theta_a)$ with $\theta_a = sin^{-1}(1/\sqrt{N})$. For $N > 4$ this probability is close to, but less than, 1. Table I presents p_s for different numbers of primitives. Upon measurement three different cases can occur:

1) the intersecting primitive is measured;
2) a non-intersecting primitive is measured, but in fact the ray intersects one primitive;
3) a non-intersecting primitive is measured and in fact there is no intersection (all primitives have a uniform probability $= 1/N$ of being measured).

Algorithm 3 Hybrid algorithm for the non overlapping case (single solution)

```
for all pixels (x, y) on the image plane do
  intersected = False
  iteration = 0
  r = ⌊(π/4)√N⌋
  while not intersected and (iteration < c) do
    ID = Q_RayCast (x,y,primitives,r) {Algorithm 1}
    intersected = intersect (x,y, ID)
    iteration ++
  end while
end for
```

Algorithm 3 calls quantum Algorithm 1 and then verifies whether the ray intersects the measured primitive – this is an hybrid algorithm, since the classical code calls a quantum program. If the measured primitive is not intersected the quantum algorithm is executed again. If after c such iterations no measured primitive intersects the ray, then that pixel is not occluded with probability

$$p_{s,c} = p_s + \sum_{j=1}^{c-1} p_s(1-p_s)^j$$

according to the geometric distribution. The last row of Table I presents $p_{s,c=2}$ for different numbers of primitives. Note that within this paper's context, verifying whether a given ray intersects a specific primitive is $\mathcal{O}(1)$: the goal of the quantum approach is to reduce the number of evaluations of the intersect function, not the cost of each such evaluation.

C. Circuit

Figure 4 presents the quantum circuit for a setup with 4 primitives and $\max(x, y) = 3$, which requires up to 2 qubits to represent max_X and max_Y ($|b\rangle$ on the circuit). The H subcircuit uses Hadamard gates to prepare $|p\rangle$ onto an uniform superposition. Only 1 Grover iteration is required ($r = 1$, see Table I) together with 5 ancillary qubits logically organized onto 2 registers: $|aux\rangle$ and $|aux2\rangle$. Finally, measurement gates measure the quantum state and store the result on classical register $|c\rangle$.

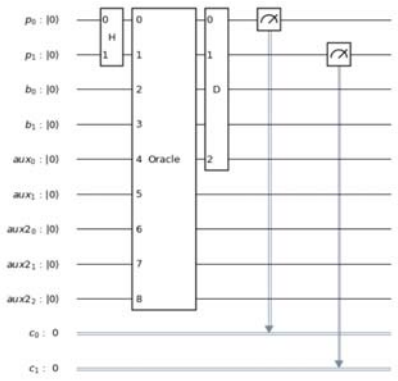


Fig. 4. Geometric setup for the non-overlapping case.

Figure 5 depicts the oracle. Each of the $\{m|M\}\{X|Y\}$ operators generates the minimum (resp. maximum) X (resp. Y) bounds of each primitive onto $|b\rangle$. The *ge* and *le* operators compare the bounds with the ray's coordinates (x, y), which are hardwired into the circuit; comparison results for y (resp. x) are stored onto $|aux2_1\rangle$ (resp. $|aux2_2\rangle$), referred above as $|$*Vint*$\rangle$ (resp. $|$*Hint*$\rangle$). These are then anded together onto $|aux2_0\rangle$ ($|$*intersect*$\rangle$), using a reversible Toffoli gate. The latter will thus be $|1\rangle$ for the primitives intersected by the ray and $|0\rangle$ for the remaining (the whole circuit is on a superposition over all possible states of $|p\rangle$: 4 basis states in this example). After each of the comparison operators the bounds generator operator appears again: this is required to reverse the state computed onto $|b\rangle$, resetting its value to $|00\rangle$ before it is used again for the next computation. Such is the nature of quantum reversible computing: all computations have to be undone before the qubits can be used again by some other operator. The detailed circuits for each of the bounds generator and the magnitude comparison operators are not shown since these are just basic Boolean operations implemented using reversible quantum gates. Finally, a rotation over the Z axis is applied to $|p\rangle$ conditioned to $|aux2_0\rangle$ being $|1\rangle$, thus flipping the sign of the coefficients of those primitives which intersect ray (x, y).

After the above described circuit the state of $|aux2\rangle$ hasn't been reversed to $|000\rangle$. However, all anciliary have to be reversed into the original state, which implies repeating the whole circuit in reversed order (or in fact in some order guaranteed to be the inverse of the state preparation circuit). The last Toffoli gate in Figure 5 reverses the state of $|aux2_0\rangle$; this is followed by a complete repetition of the remaining of the circuit, in order to reverse $|aux2_1\rangle$ and $|aux2_2\rangle$ – these are not depicted here due to space constraints.

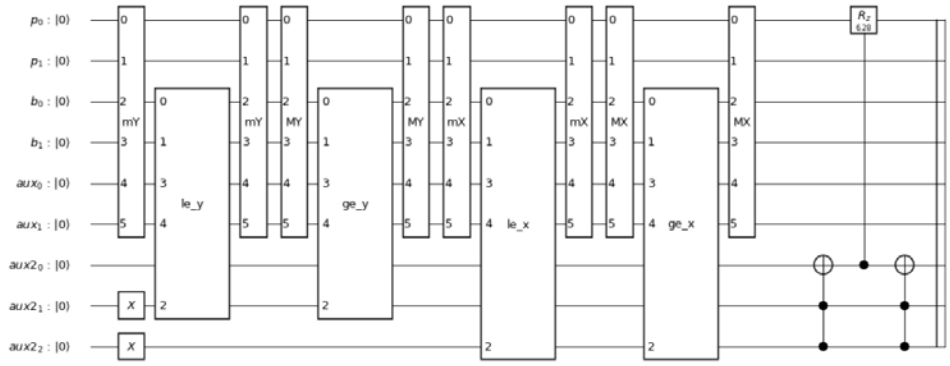
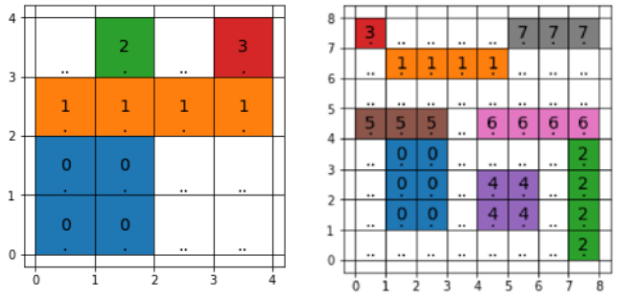


Fig. 5. Oracle for ray casting.

D. Results

Figure 6 presents the images obtained with the Qiskit simulator for two geometric configurations (scenes) with 4 and 8 primitives, respectively. The dots depicted in each pixel represent the number of iterations required to evaluate the pixel with $c = 2$. For the single solution case the probability of measuring a non intersecting primitive is not significant; the intersecting primitive was always measured on the first iteration of Algorithm 3. When there is no intersection the algorithm will run c iterations.



(a) 4x4 image and scene with 4 primitives

(b) 8x8 image and scene with 8 primitives

Fig. 6. Reference images obtained with the Qiskit simulator for the no overlapping case. Numbers identify the primitive; the number of dots indicates algorithm 3 iterations due to measuring a non intersecting primitive.

Table II presents the total number of gates, total number of qubits and circuit depth (length, in gates, of the longest path) for an exemplary pixel $(x = 1, y = 2)$ for both scenes (4 and 8 primitives). Even for the simpler scene the circuit depth is 33; this is well above what can be reliably executed on current quantum machines at the time of writing. In fact, noise is additive over the gates and, additionally, execution times must be kept short due to qubits dephasing and limited coherence times. The situation is even worse in practice since these circuits have to be mapped onto the real machine, which has limited connectivity among physical qubits and supports a limited set of gates; the executable circuit gate count and depth is therefore significantly larger. The results of such a large quantum circuit are therefore noisy to the point where

TABLE II
NUMBER OF GATES, TOTAL NUMBER OF QUBITS AND CIRCUIT DEPTH FOR BOTH SCENES

Scene	Gates	Depth	Qubits
4 primitives	83	33	9
8 primitives	195	68	15

no significant conclusion can be extracted from them. Figure 7 presents an histogram of the measured primitives for pixel $(x = 1, y = 2)$, with a single iteration of Algorithm 3 and 256 trials, for both the simulated and the real machine. While the former is 100% accurate, the latter is overwhelmed by noise, resulting in an almost uniform distribution of probability over all possible measured states.

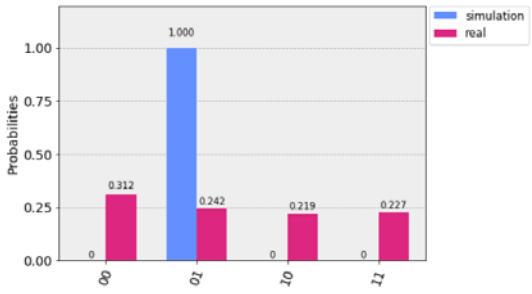
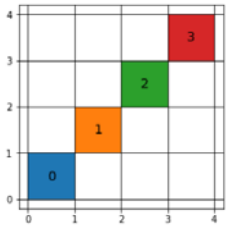


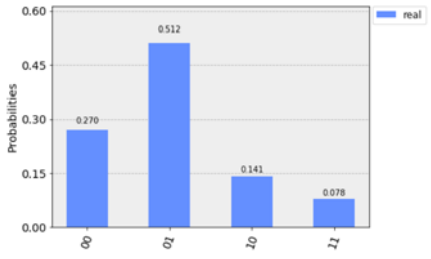
Fig. 7. Histogram: primitive measurements for simulator and real machine.

To allow execution on a real machine a simpler circuit was prepared for the scene depicted in Figure 8(a); each primitive has coordinates equal to its ID, therefore the operators $\{m|M\}\{X|Y\}$ that generate the primitive bounds are not required. The ray coordinates can be compared directly with the primitive ID and the bounds representation $|b\rangle$ is not required (compare with Figure 5). The circuit for pixel $(x = 2, y = 2)$ requires 7 qubits and 48 gates for a maximum depth of 19. Figure 8(b) presents the histogram for the execution of a single iteration of Algorithm 3, 256 trials with the simplified circuit. The correct primitive is measured with probability ≈ 0.5 while the second most probable measurement is $|00\rangle$ – this is due to qubit decoherence (the state relaxes towards $|00\rangle$) and can be

corrected (eventually not using state $|00\rangle$ as a primitive ID and treating it as an error). These results suggest that real quantum computers can be used in the future, as qubits coherence times increase and gate errors decrease.



(a) 4x4 image and scene with 4 primitives



(b) Histogram for 256 trials, pixel $(2, 2)$

Fig. 8. Simplified setup for execution on the real quantum machine.

V. Ray Casting: occlusion with multiple solutions

By adding a third dimension to the primitives it becomes possible that more than one primitive projects into the same pixel; the number of solutions to Grover's algorithm, t, can now range from 0 (no occlusion) to the total number of primitives, N. As long as t is known and less than $N/2$ the problem can still be solved using the algorithm proposed in the previous section by setting the number of Grover's iterations $r \leftarrow \lfloor \frac{\pi}{4}\sqrt{\frac{N}{t}} \rfloor$. But in real cases the number of primitives projecting onto a given pixel is unknown, therefore r cannot be set *a priori* to the ideal number of iterations.

A. Algorithm

We use an algorithm proposed by Boyer et al. [17] based on exponential search and demonstrated to converge in $\mathcal{O}(\sqrt{\frac{N}{t}})$ iterations; the case $t = 0$ is handled by terminating after a constant number of iterations, c. At each iteration, Algorithm 4 exponentially increases the maximum number of possible Grover iterations S and then randomly selects r from $1 \ldots S$. At each iteration the primitives are also randomly sampled, which will succeed with probability $\frac{t}{N}$.

B. Results

A scene with 8 primitives at 3 different depths (Figure 9) is used to empirically verify Algorithm 4. Simulation results show 100% convergence, with an occluding primitive being measured for all pixels where such a primitive exists

Algorithm 4 QSearch: Quantum exponential search algorithm for the overlapping case - pixel (x, y)

```
intersected = False; iteration = 0
l = 0 ; g = 1.3
while not intersected and (iteration < c) do
    ID = Q_SampleUniformDistribution (primitives)
    intersected = intersect (x,y, ID)
    if not intersected then
        l = l + 1
        S = max(g^l, √N)
        r = rand(1 ... S)
        ID = Q_RayCast (x,y,primitives,r)
        intersected = intersect (x,y, ID)
        iteration++
    end if
end while
```

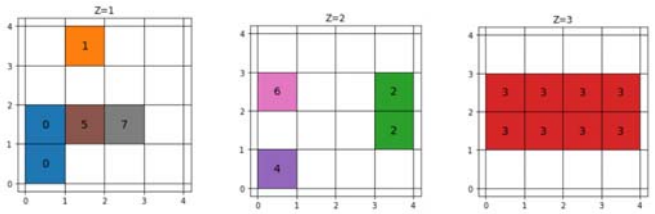


Fig. 9. 8 primitives scene with 3 different depths.

– see Figure 10. The probabilistic nature of the algorithm and the importance of the random sampling step are clearly demonstrated by the number of iterations required to find an occluding primitive. For example, pixels $(2, 1)$, $(1, 2)$ and $(3, 2)$ results are found with 0 iterations by random sampling and other pixels require arbitrarily 1 or 2 iterations (with an unreported number of Grover iterations).

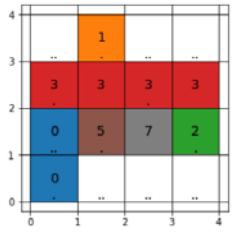


Fig. 10. 4x4 occlusion image with 8 primitives and multiple possible solutions per pixel.

VI. Ray Casting: visibility with multiple solutions

In order to compute which primitive, if any, is visible along each pixel the minimum depth (Z coordinate) has to be evaluated. We use an approach based on Durr et al. [13] and given as Algorithm 5. The minimum depth is initialized with some maximal value. The exponential search described in Algorithm 4, augmented with the current minimum depth, is then executed; if a primitive is found and if its depth is less than the current minimum then the latter and associated primitive are updated. Augmenting Algorithm 4 consists of

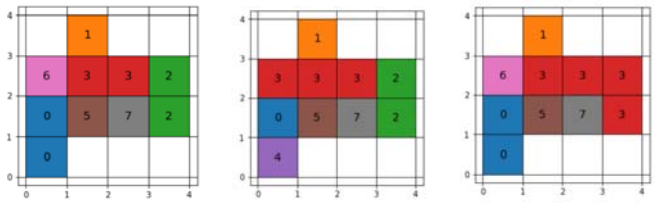
Algorithm 5 QMin: Quantum minimum algorithm

```
p = −1; pZ = maxZ; visible = False
while iteration < c do
    intersected, ID = Q_Search_Min (x, y, pZ, primitives)
    if intersected then
        visible = True
        if depth[ID] < pZ then
            pZ = depth[ID]; p = ID
        end if
    end if
    iteration ++
end while
```

augmenting the oracle given in Algorithm 2 (Figure 5) with depth data. Besides generating the primitives' bounds and comparing with the ray coordinates, now also the primitives depths are generated and compared with the current minimum depth; the oracle will only flip a primitive's coefficient sign if the 5 conditions (bounds and depth) are satisfied. The quantum circuit is now adaptive, being redesigned at each iteration of Algorithm 5 to account for the current minimum depth.

A. Results

Simulation results (see Figure 11) show that although occlusion is always correctly computed, there are a few pixels where the primitive with correct minimum depth was not found. The histogram in Figure 12 shows, for pixel (0,0), that the correct primitive is found $\approx 80\%$ of the trials, which is also confirmed by the presented images: 8 out of 10 pixels are correctly measured. A more thorough evaluation of the probabilities is required to increase the effectiveness of the proposed approach.



(a) Reference (b) Trial 1 (c) Trial 2

Fig. 11. Reference and visibility results for 2 simulation trials.

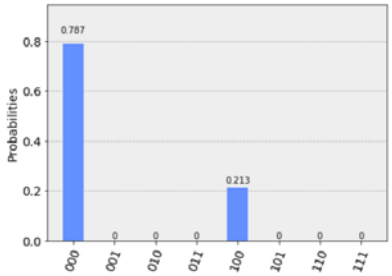


Fig. 12. Histogram: visibility for pixel (0,0) in 265 simulation trials.

VII. Conclusions

Practical algorithms and implementations of ray casting for visibility testing and occlusion from an orthographic camera based on quantum algorithms are proposed. These have been evaluated both via simulation of a quantum machine, indicative of future quantum computing capabilities, and in a simplified form on a real quantum computer. While these implementations are not yet fully fledged and further work is required to increase the probability of selecting intersecting primitives, the results show the potential of quantum computation for computer graphics through a reduction of time complexity for core operations in a rendering pipeline. Challenges, w.r.t. ray casting, include improvements on size and reliability at the hardware level and handling of more general geometric setups at the algorithmic level.

As future work, we intend to investigate quantum counting schemes to estimate the number of iterations required for Grover's algorithm. We aim to support generalised geometric setups, such as triangles and arbitrary ray directions. However, this will likely require more qubits and deeper circuits, and, consequently, noise tolerance in quantum algorithms for graphics, which might, eventually, be achieved by using quantum error correction schemes.

References

[1] R. P. Feynman, "Simulating Physics with Computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982.

[2] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London*, vol. 400, pp. 97–117, 1985.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. of the 28th Annual ACM Symposium on Theory of Computing*, STOC '96, pp. 212–219, ACM, 1996.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, pp. 1484–1509, 1997.

[5] D. Coppersmith, "An approximate fourier transform useful in quantum factoring," Tech. Rep. RC19642, IBM Research Division, 1994.

[6] A. de Touzalin, C. Marcus, F. Heijman, I. Cirac, R. Murray, and T. Calarco, "Quantum manifesto: A new era of technology," tech. rep., European Commission, 2016. http://qurope.eu/manifesto.

[7] E. Grumbling and M. Horowitz, "Quantum computing: Progress and prospects," tech. rep., Nat. Acad. of Sciences, Eng. and Medicine, 2019.

[8] A. Glassner, "Andrew Glassners Notebook: Quantum Computing - Part 1," *IEEE Computer Graphics and Applications*, no. 4, 2001.

[9] A. Glassner, "Andrew Glassners Notebook: Quantum Computing - Part 2," *IEEE Computer Graphics and Applications*, no. 5, 2001.

[10] A. Glassner, "Andrew Glassners Notebook: Quantum Computing - Part 3," *IEEE Computer Graphics and Applications*, no. 6, 2001.

[11] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, 1961.

[12] K. Sadakane, N. Sugawara, and T. Tokuyama, "Quantum Computation in Computational Geometry," *Interdisciplinary Information Sciences*, vol. 8, no. 2, pp. 129–136, 2002.

[13] C. Durr and P. Hoyer, "A quantum algorithm for finding the minimum," in *in LANL e-print quantph/9607014, http://xxx.lanl.gov*, 1996.

[14] S. Caraiman, "Quantum computer graphics algorithms," *Buletinul Institutului Politehnic din Iasi, Sectia Automatica si Calculatoare*, vol. 62, no. 4, pp. 21–38, 2012.

[15] M. Lanzagorta and J. Uhlmann, *Quantum Computer Science*. Synthesis Lectures on Quantum Computing, Morgan & Claypool, 2009.

[16] E. R. Johnston, "Quantum supersampling," in *ACM SIGGRAPH 2016 Talks*, SIGGRAPH '16, pp. 38:1–38:1, ACM, 2016.

[17] M. Boyer, G. Brassard, P. Hayer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte der Physik*, vol. 46, no. 5, 1998.