

Assignment 03: Quantum gates, circuits and registers

Objectives

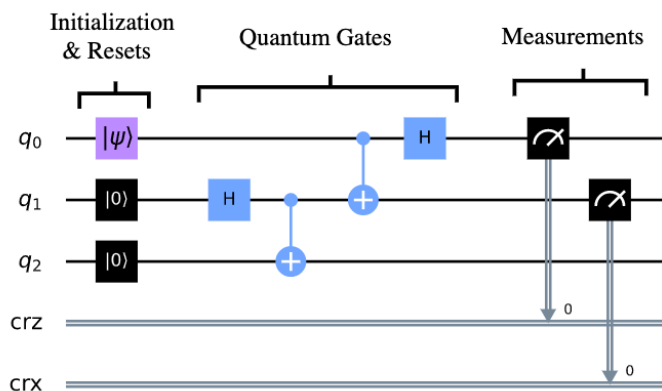
- You will understand the logic behind the basic types of quantum gates
- You will apply the basic operations of quantum gates and create several circuits

Requirements

- Notebook or Desktop Computer with access to the internet
- Access to IBM Quantum composer <https://quantum-computing.ibm.com/composer>

Solution Steps

The most basic operations performed on qubits are defined by quantum gates, similar to logical gates used in Classical Computers. Quantum gates can be used to build complex algorithms, which usually end in a measurement operation.



To implement those algorithms we need to build quantum circuits.

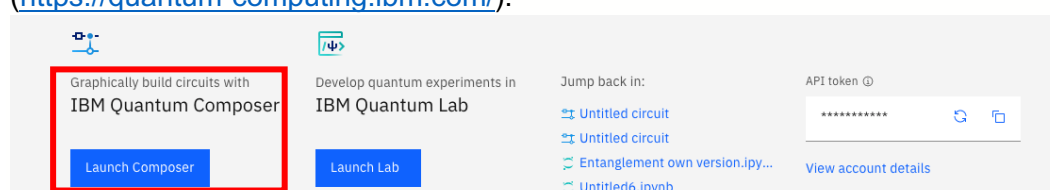
A quantum circuit is basically a collection of quantum gates that are interconnected with each other [1].

Quantum circuits enable quantum computations by creating a sequence of operations, usually starting with an initialization of qubits, followed by quantum gates and measurements [2].

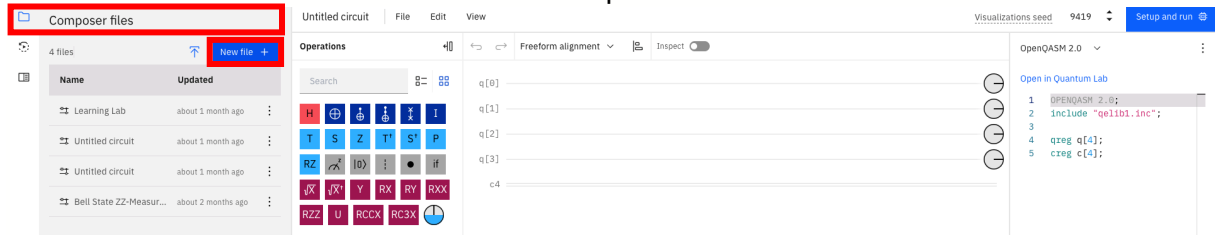
Qubits are initialized by creating a quantum register. In a Quantum Computer qubits are conceptually grouped together in so-called quantum registers. Each qubit in the register has an index, starting with 0 and increasing by 1 with each additional qubit. In the picture above, you can see a quantum register with three qubits, that are indexed by 0, 1 and 2. When performing operations on specific qubits within a quantum register, they can be addressed with using the respective index [3].

Quantum gates

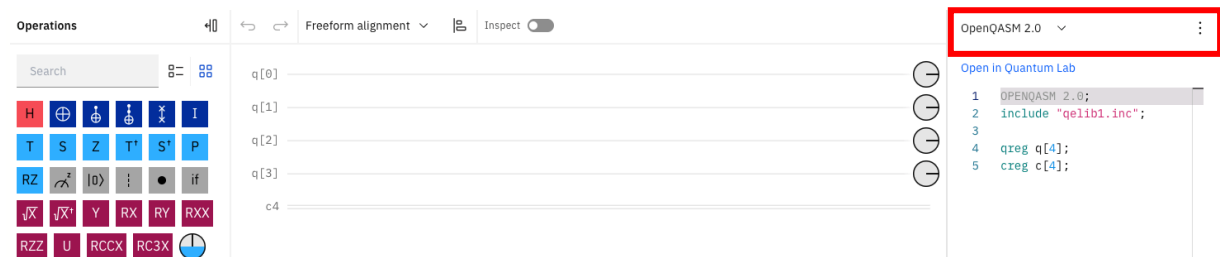
We are going to use IBM Quantum composer to get familiar with the quantum gates. You can access the Quantum Composer right from the dashboard of IBM Quantum Computing (<https://quantum-computing.ibm.com/>).



Please create a new file in IBM Quantum composer.

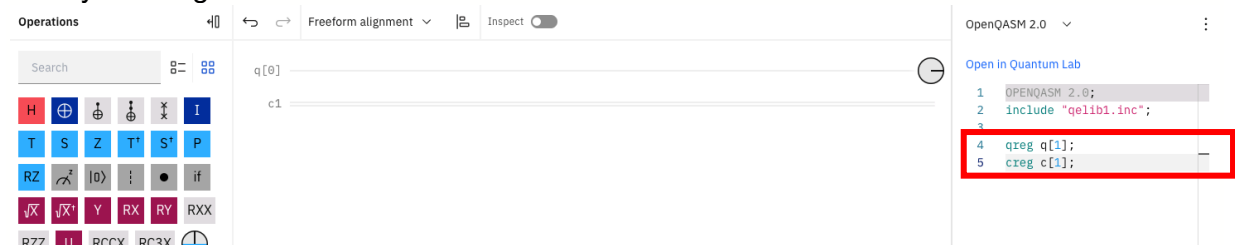


If “OpenQASM 2.0” is not chosen as the command-line interface by default, please use the drop-down menu to select it.



The command “qreg q[4]” on the sidebar indicates that by default our quantum register consists of four qubits. As we always need classical bits to measure qubits, the circuit needs to include the same number of classical bits. This can be achieved by creating a classical register with applying the command “creg c[4]”.

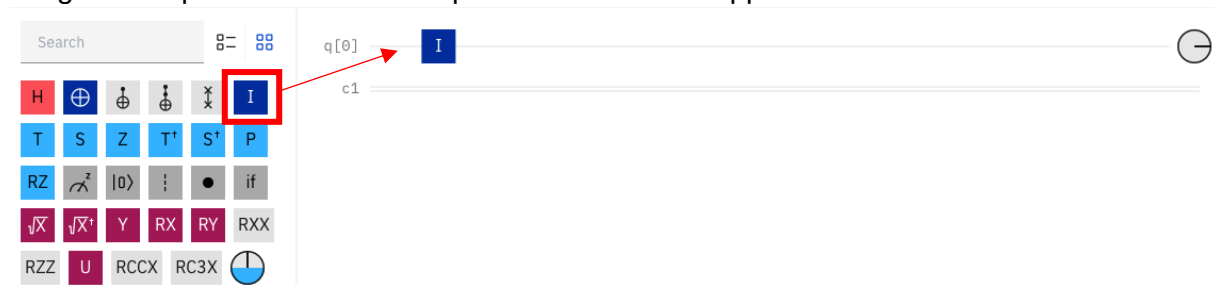
As we want to visualize the effects of gates on the state of a single qubit, please change the number of qubits and bits from four to one. You can simply change the number of qubits and bits by entering the desired number in the command cell.

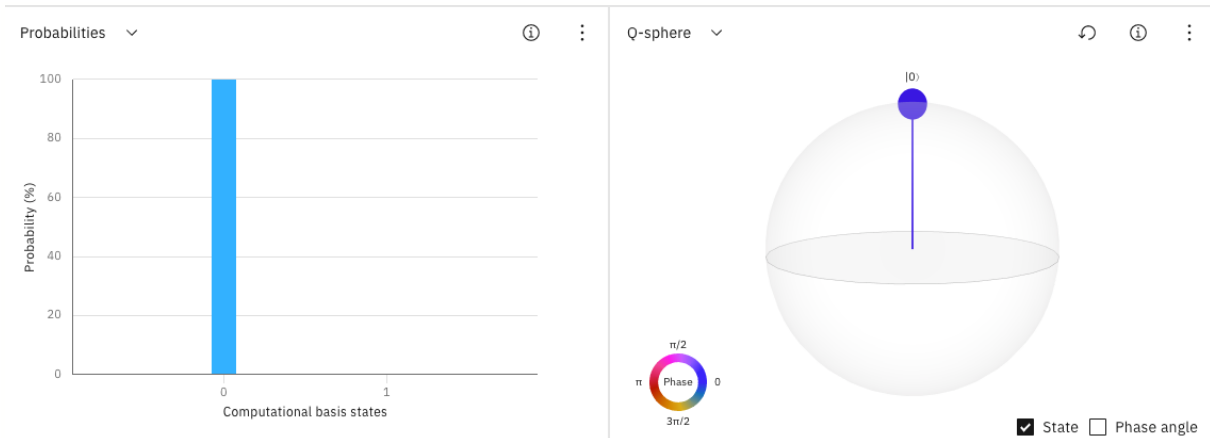


Please bear in mind, that the initial state of a qubit is by default always 0.

a) Identity gate (Id-gate)

Drag and drop the icon “I” on the qubit and see what happens.



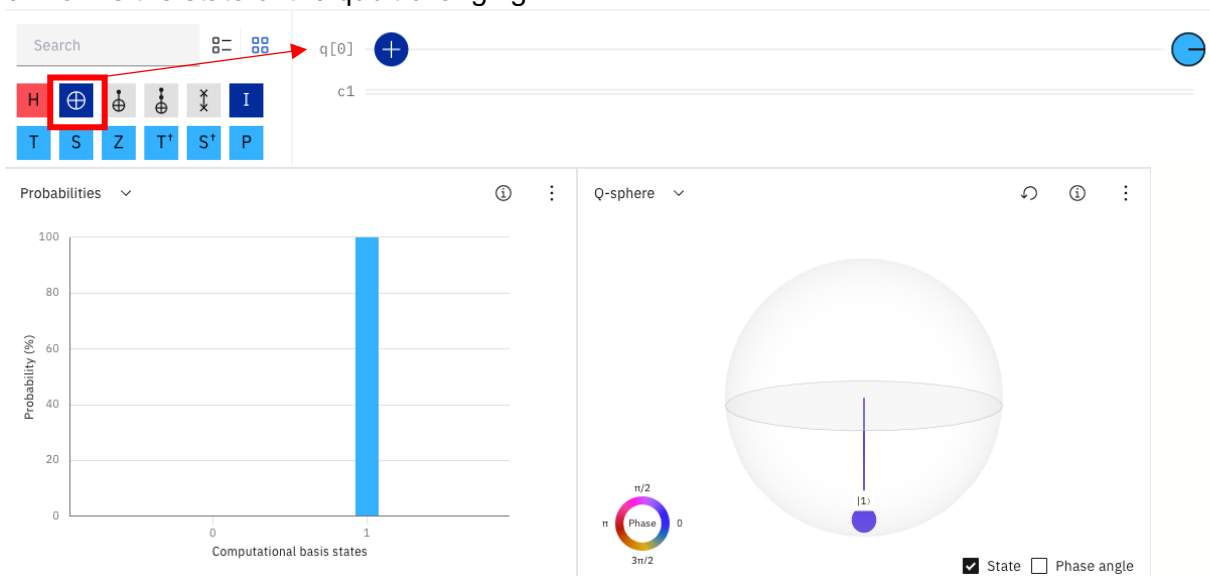


As you can see in the graph on the bottom left side, the qubit has state 0 with a probability of 100% and hence the Identity gate did not change the state of the qubit. By definition the identity gate does never modify the quantum state of a qubit, indicating that if the initial state was 0, the state is still 0 after performing the gate.

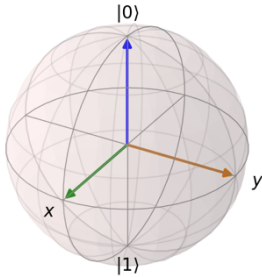
Although the states are not changed by this gate, its application is useful when it comes to simulations and error models, that aim to measure the error rate of a Quantum Computer [4]. In order to be able to test the next gate you need to delete the identity gate. You can do so by clicking on the gate and selecting the trash can.

b) X-gate

Again, use the drag and drop function to perform the X-gate, also called NOT-gate on qubit 0. How is the state of the qubit changing?



You can see in the diagram on the left side that the probability of the qubit having state 1 has changed from 0% to 100% by applying the X-gate, which is equal to the NOT-gate on a Classical Computer. As the X-gate can change the state from 0 to 1 or from 1 to 0 it is sometimes also called a bit-flip gate [5]. The graph on the right side of the picture above shows the Q-sphere, which is an extended version of the Bloch sphere as it is able to represent the states of several qubits while the Bloch sphere is a visual representation of the state of a single qubit. The Bloch sphere has three axes (z, x, y) [6].



To get a better understanding about the axes of the Bloch sphere please look at the figure on the left. The vertical axis (blue line) which indicates the states 0 and 1 is called z axis. The x and y axis are two horizontal axes that go through the middle of the Bloch sphere.

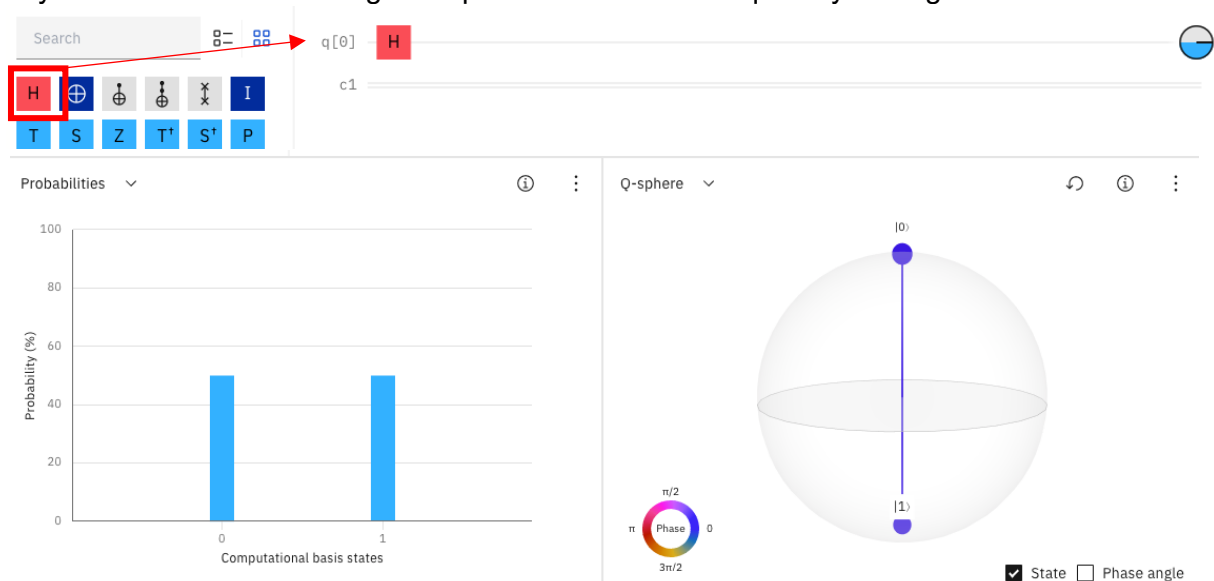
[7]

The Q-sphere in the Quantum Composer shows that applying the X-gate is equivalent to a 180-degree rotation around the x axis of the Bloch sphere [8].

Before continuing with the next gate please delete the NOT-gate from the circuit.

c) Hadamard gate (H-gate)

Try out how the Hadamard gate impacts the state of the qubit by adding “H” to the circuit.

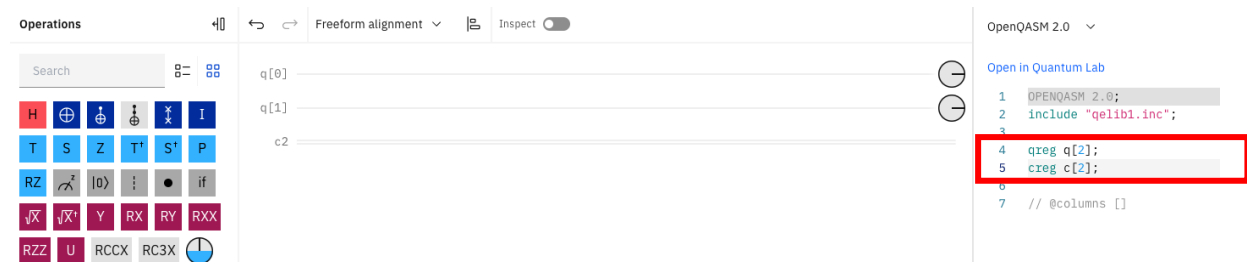


You can see that after applying the H-gate the qubit has a 50/50 probability to be in state 0 or 1. This phenomenon is called superposition [9]. Therefore, we can conclude that applying the H-gate enables qubits to be in superposition. Again, please delete the H-gate before we continue with applying the next gate.

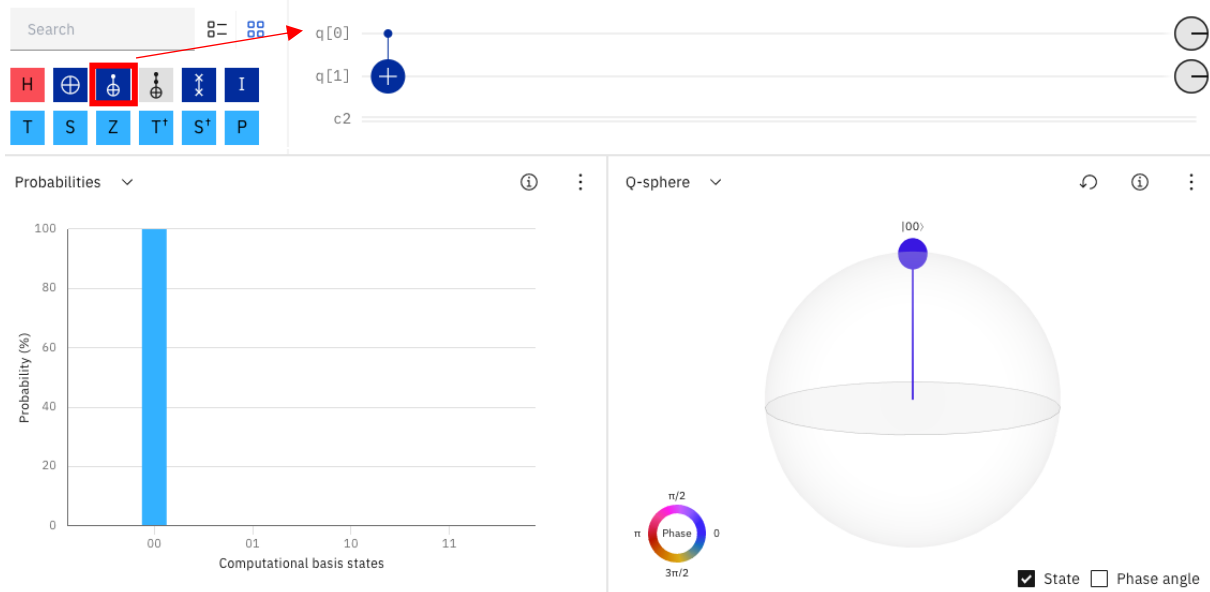
d) Controlled gates

With controlled gates we can transfer a gate that was applied on one qubit (control qubit) to another qubit (target qubit).

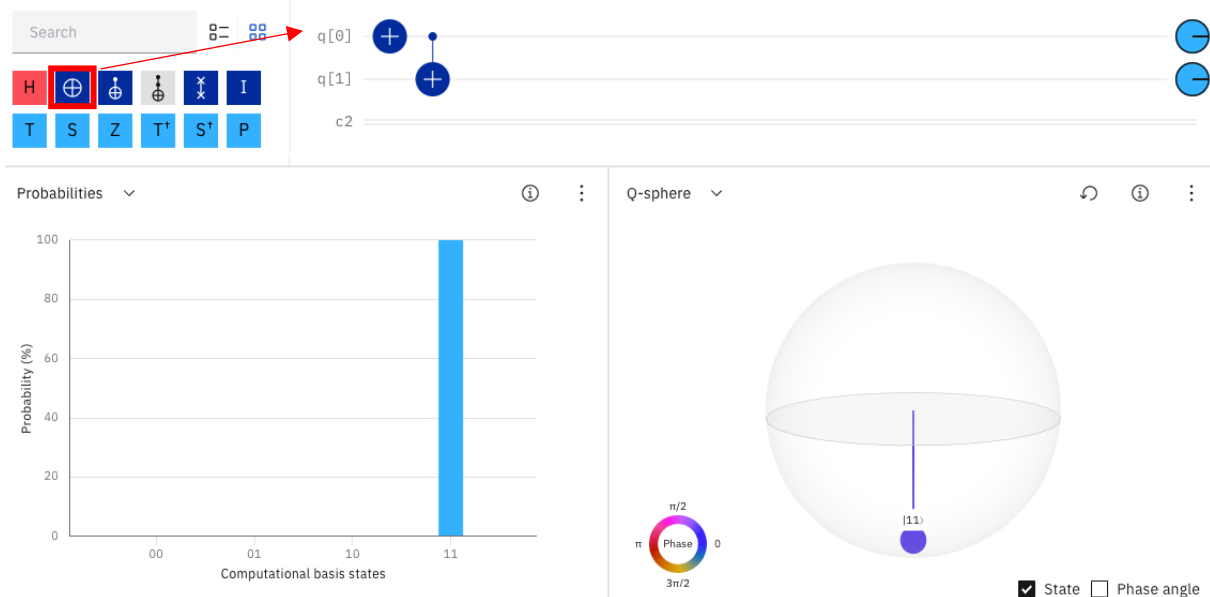
To be able to try out how controlled gates work, we need to add an additional qubit and one classical bit to our circuit.



Start with applying a CNOT-gate on qubit 0.



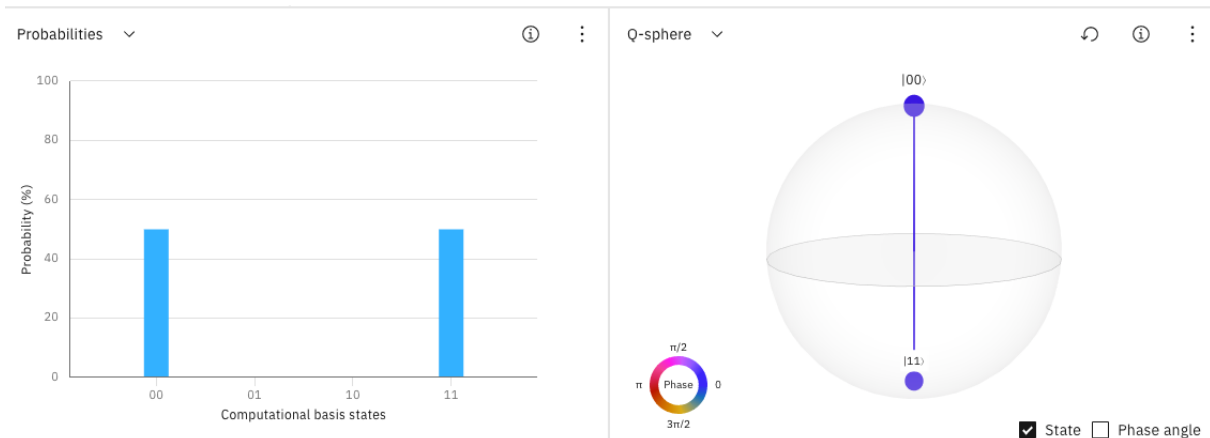
As there was no gate applied on control qubit (qubit 0) before the CNOT-gate was executed, it has no effect on the target qubit (qubit 1). As the CNOT-gate requires another gate to be performed before it is applied, let's try what happens when we add a X-gate on qubit 0 before the CNOT-gate. You can just drag and drop the X-gate at the desired position in the circuit.



You can see that the state of both qubits changed to 1, although we only applied the X-gate on the control qubit (q0). Due to the CNOT-gate the operation on the control qubit was transferred to the target qubit (q1).

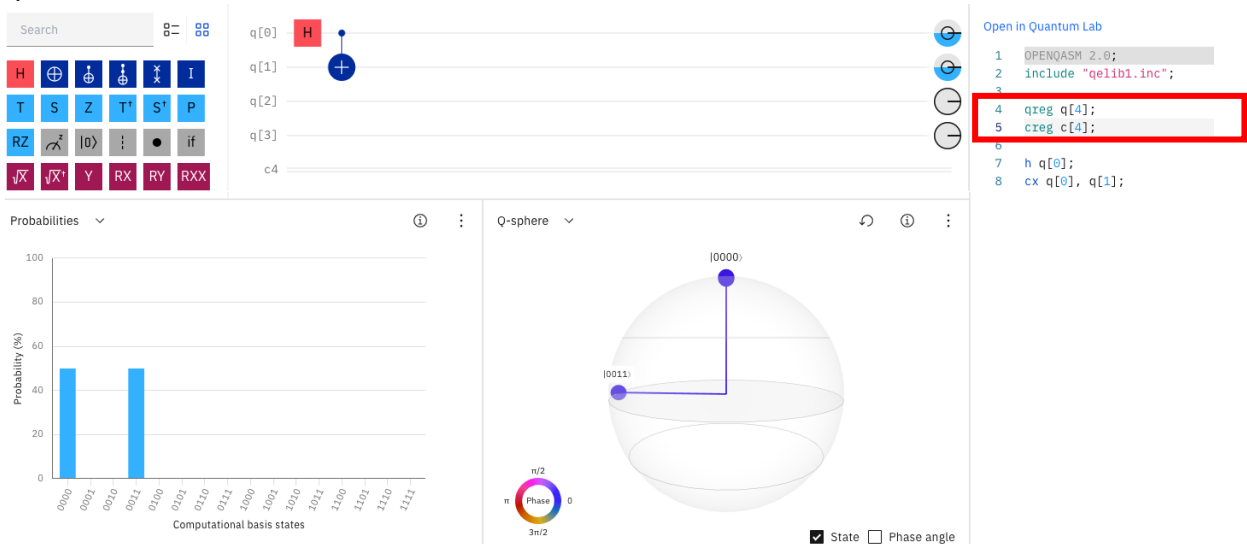
Let's try if this also works with a H-gate. Delete the X-gate and add a H-gate to qubit 0.





Congratulations, you created superposition for both qubits.

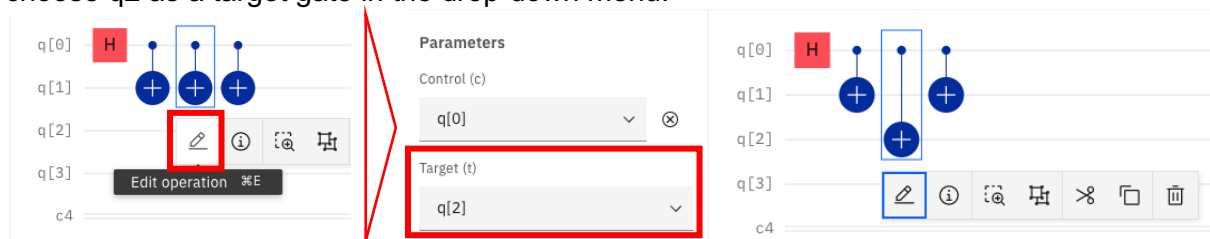
CNOT-gates can also be applied on several qubits. To try out how it works, add two additional qubits and classical bits to our circuit.



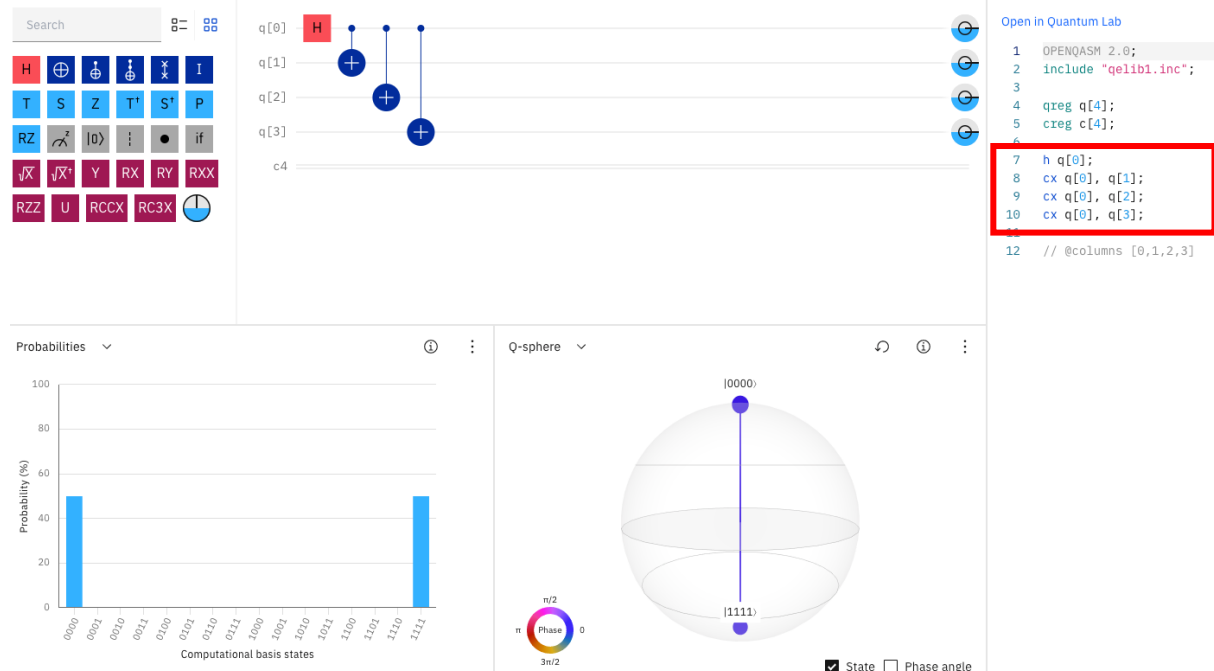
Little reminder from assignment 2: We read strings of qubits always from right to left. Therefore, the string 0011 for example tells us that the state of qubit 0 and qubit 1 are 1, while the state of qubit 2 and qubit 3 are 0.

To bring qubit 2 and qubit 3 also in superposition without applying additional Hadamard-gates you can add CNOT-gates from q0 to q2 and from q0 to q3.

This can be achieved by adding two CNOT-gates to the circuit. In the next step you have to change the target gate for the second CNOT-gate to q2 and for the third CNOT-gate to q3. This can be done by clicking on the CNOT-gate and selecting "edit operation". Now you can choose q2 as a target gate in the drop-down menu.



Please repeat the same steps for the third CNOT-gate to get the following result.



Task 1: Think about another way how the CNOT-gates could be used to achieve the same result (5 min.)

Hint: You do not need to add additional gates. CNOT-gates can also be applied on target qubits of other CNOT-gates.

Task 2: Try to create a circuit that leads to the following outcomes with a 50/50 probability: 01110, 11111 (15 min.)

Hint: You have to use **one X-gate/NOT-gate**, **one Hadamard-gate** and **several CNOT-gates**. Also, remember that the number of characters in the string indicates how many qubits and classical bits your circuit should have. Little reminder about the effects of the gates explained above:

- X-gate/NOT-gate: flips the state from 0 to 1 and vice versa
- H-gate: enables the qubit to be in state 0 and 1 with a 50/50 probability
- CNOT-gate: transfers the gate from a control qubit to a target qubit

If you can't come up with a solution within the suggested time frame you can skip this task as you will be provided with a solution after completing the workshop.

We can also create the circuit of task 2 with a code in IBM Quantum Lab. Although running the code is not part of this Learning Lab, you are provided with the code and the corresponding instructions in appendix 1. In case you want to further improve your coding skills as an extension to this learning lab feel free to try it out.

Useful Resources for Own Research

Overview quantum gates:

https://quantum-computing.ibm.com/composer/docs/iqx/operations_glossary


Single Qubit gates: <https://qiskit.org/textbook/ch-states/single-qubit-gates.html>

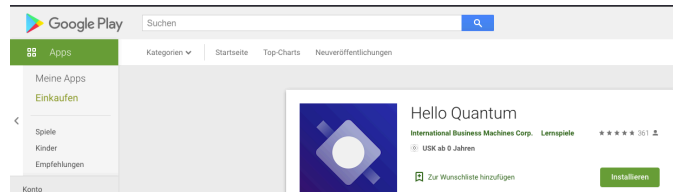
Hello Qiskit Game:

<https://qiskit.org/textbook/ch-ex/hello-qiskit.html#Level-2:-Basic-single-qubit-gates>

Hello Quantum Game:



Hello Quantum 
IBM
Designed for iPad
★★★★★ 4.7 • 37 Ratings
Free
[View in Mac App Store](#)



Retrospective

Please answer the following questions:

1. What is a quantum register?
2. Which components are usually needed to build a quantum circuit?
3. How is the gate called that does not change the state of a qubit and in which situations it might be useful to use this gate?
4. What is the effect of applying a X-gate and how is this effect visualized in the Bloch sphere?
5. You want to transfer the state of qubit 2 to qubit 6. Which gate do you have to apply and what specifications do you need to make?

Sources

- [1] <https://www.sciencedirect.com/topics/mathematics/quantum-circuit>
- [2] <https://qiskit.org/textbook/ch-algorithms/defining-quantum-circuits.html>
- [3] <https://www.quantum-inspire.com/kbase/qubit-register/>
- [4] <https://www.quantum-inspire.com/kbase/identity-gate/>
- [5] https://quantum-computing.ibm.com/composer/docs/iqx/operations_glossary
- [6] <https://quantum-computing.ibm.com/services/docs/services/terms-glossary#term-bloch-sphere>
- [7] <https://qutip.org/docs/4.1/guide/guide-bloch.html>
- [8] <https://qiskit.org/documentation/stubs/qiskit.circuit.library.XGate.html>
- [9] <https://www.quantum-inspire.com/kbase/hadamard/>

Appendix

Appendix 1: Creating a circuit in the Quantum Lab using Python

At first, we start with importing the basic functions.

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, Aer, execute
from qiskit.visualization import plot_bloch_multivector, plot_histogram
```

In the next step we create a quantum register called “qReg” with five qubits and a classical register called “cReg” with five classical bits.

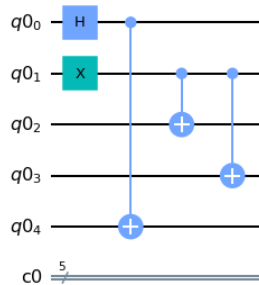
```
qReg = QuantumRegister(5)
cReg = ClassicalRegister(5)
```

We continue with building a circuit based on the quantum register and the classical register. After applying a H-gate on q0 we apply a controlled X-gate, where q0 is the control qubit and q4 is the target qubit. Furthermore, we add a X-gate on q1 and a controlled X-gate from q1 to q2 and q3. The last command enables us to draw the circuit and thus visualizes the operations we performed on the qubit.

```
circuit = QuantumCircuit(qReg, cReg);
circuit.h(0);
circuit.cx(0,4);
circuit.x(1);
circuit.cx(1,[2,3]);

circuit.draw()
```

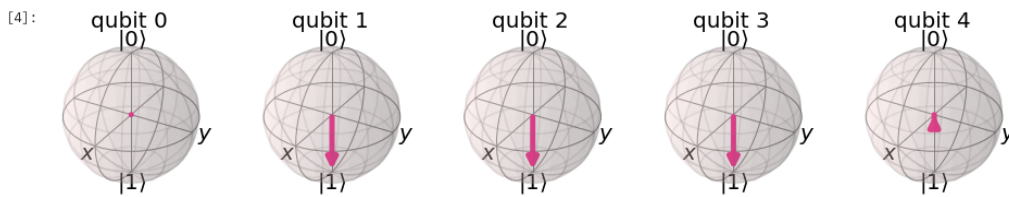
[3]:



Similar to the Quantum Composer we can also create a code for visualizing the states of the qubits by using the Bloch sphere. Execute the following code to look how the operations on the circuit changed the states of each qubit.

```
bloch_simulator = Aer.get_backend('statevector_simulator');
bloch_result = execute(circuit, backend = bloch_simulator).result();
statevector = bloch_result.get_statevector();
plot_bloch_multivector(statevector)
```

While we can see that q1, q2 and q3 have state 1, the pink dot in the middle of q0 and q4 indicates that those qubits are in superposition.



To get to know the states of the circuit as a whole, we need to measure the outcomes of the manipulations we performed on the qubits in step 3. Besides measuring the outcomes, the following code also counts how often which state occurred in the simulations and plots the result by using a histogram.

```

circuit.measure([0,1,2,3,4],[0,1,2,3,4]);
backend = Aer.get_backend('qasm_simulator');
result = execute(circuit, backend = backend, shots = 1024).result();
counts = result.get_counts();
plot_histogram(counts)

```

The histogram shows that after performing 1,024 simulations we almost have a 50/50 chance that all qubits in the circuit have the state 1 or that qubit 0 and qubit 4 have state 0 while the other qubits have the state 1. This result is in line with the visualization of the Bloch sphere of the previous code.

