Explore gates and circuits with the Quantum Composer

Category How-to

What is it?

IBM Quantum Composer is a graphical quantum programming tool that lets you drag and drop operations to build quantum circuits and run them on real quantum hardware or simulators.

What can it do?

Visualize qubit states

See how changes to your circuit affect the state of qubits, shown as an interactive q-sphere, or histograms showing measurement probabilities or statevector simulations.

Run on quantum hardware

Run your circuits on real quantum hardware to understand the effects of device noise.

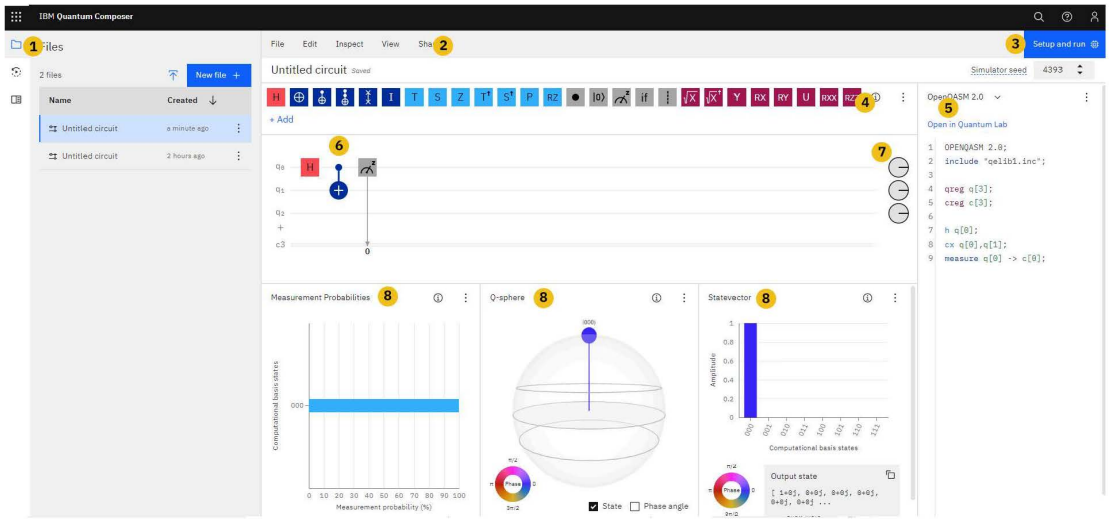
Automatically generate code

Instead of writing code by hand, automatically generate OpenQASM or Python code that behaves the same way as the circuit you created with Quantum Composer.

Tour of the interface

IBM Quantum Composer has a customizable set of tools that allow you to build, visualize, and run quantum circuits on quantum systems or simulators.

Use the More options menu on each window to access additional tools and actions.



1. **Tools panels** - Use the side panel to view your files or jobs, or the documentation. To close the side panel, click the icon for the open tab.
2. **Menu bar** - Use these menus to create a new circuit, manage saved circuits and registers, customize your workspace, and more.
3. **Run area** - Change the run settings and then run your circuit on a quantum system or simulator.
4. **Composer files** - The circuits you create save automatically and display in the Composer files panel.
5. **Circuit name** - Click here to name your circuit.
6. **Operations catalog** - These are the building blocks of quantum circuits. Drag and drop these gates and other operations onto the graphical circuit editor. Different types of gates are grouped together by color. For example, classical gates are dark blue, phase gates are light blue, and non-unitary operations are grey.

   To learn about the available gates and operations, right-click an operation and select Info to read its definition.
7. **Code editor** - Use the View menu to open or close the code editor, which allows you to view and edit the OpenQASM or Qiskit code for the circuit.
8. **Graphical circuit editor** - This is where you build a circuit. Drag gates and other operations onto the horizontal qubit “wires” that make up your quantum register.

To remove a gate from a wire, select the gate and click the trash can icon.

To edit the parameters and settings on gates that support editing, select the gate on the graphical editor and click Edit.

9. **Toolbar** - Access frequently used tools to undo and redo actions, change the gate alignment, and switch to inspect mode. Inspect mode lets you view a step-by-step view of the qubit states as you circuit computation evolves. To learn more, see Inspect your circuit, step-by-step.
10. **Phase disks** - The phase of the qubit state vector in the complex plane is given by the line that extends from the center of the diagram to the edge of the gray disk (which rotates counterclockwise around the center point).

    Use the View menu to show or hide the phase disks.
11. **Visualizations** - Visualizations characterize your circuit as you build it. They use a single-shot statevector simulator, which is different from the system specified in the Setup and run settings. Note that the visualizations ignore any measurement operations you add. Sign in and click **Setup and run** to get results from the specified backend instead.

    To learn about the visualizations, see Visualizations.

Build, edit, and inspect quantum circuits

1. Open IBM Quantum Composer

1. (Optional) If you are not currently signed in to IBM Quantum, select **Sign in** in the upper right corner. Then, you can either sign in or **Create an IBMid account**.

   If you don't sign in, the visualizations automatically show simulated results for up to four qubits. If you want to run your circuit on a quantum system or simulator, or if you want to visualize a circuit that has more than four qubits, you must sign in.
2. Open IBM Quantum Learning using the application switcher in the top right, and clicking **Learning**.
3. Open IBM Quantum Composer by clicking the **Composer** in IBM Quantum Learning navigation. The workspace displays an untitled empty circuit.
4. Name your circuit by clicking on the words **Untitled circuit** and typing in a name for your circuit. Click the checkmark to save the name.

5. (Optional) Customize your workspace:
   - Use the **View** menu to change from the default theme to a monochrome theme. You can also select which panels to include on your workspace, then use the menu in the right corner of any panel to access options for further customization. The options to show or hide phase disks, choose the alignment of the qubits on your circuit, and reset the workspace to the default are in the View menu as well.
   - Open your account settings to switch between dark and light workspace themes.
   - If you are signed in, the Files panel displays by default. You can close it by clicking the Files icon.

To build a circuit, you can either drag and drop operations, or you can enter OpenQASM code into the code editor. Your circuit is automatically saved every time you make a change.

2. Build your circuit with drag-and-drop

Operations catalog

Drag and drop operations from the operations catalog onto the quantum and classical registers. Start typing in the search window to quickly find an operation.

Collapse and expand the operations catalog by clicking the icon in the upper right corner of the operations panel. You can also switch between an expanded or condensed view of the operations by using the icons next to the search window.

Right-click an operation icon and select Info to view the definition of an operation, along with its QASM reference.

To undo or redo, use the curved arrows in the toolbar.

Alignment

Choose Freeform alignment to place operations anywhere on the circuit. For a more compact view of your circuit, choose Left alignment. To see the order in which operations will execute, choose Layers alignment, which will apply left alignment and add column delineators that indicate the execution order, from left to right and top to bottom.

Once operations are placed on your circuit, you can continue to drag and drop them to new positions.

Copy and paste

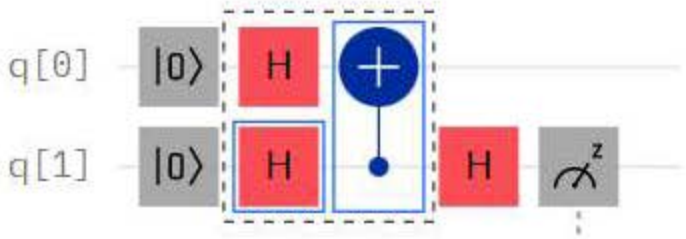
Click an operation and use the icons in the contextual menu to copy and paste it.

Select multiple operations

You can select several operations to copy and paste them, drag them to a new location, or group them into a custom unitary operation that displays in your operations catalog and functions as a single gate.

To select more than one operation, place your cursor just outside one of the operations, then click and drag across the area to select. Shift-click individual operations to select or deselect them. A dashed line outlines the set of operations you are selecting, and each operation that is actually part of the selection is outlined in blue.

For example, in the following image, the Hadamard gate on q1 and the CX gate are selected. The Hadamard gate on q0 is not selected.



Select Copy from the contextual menu to copy the group.

To paste the group of operations, right-click in the circuit and select Paste.

Build a custom operation using the group feature

To group several operations together and save them as a custom operation, first select the operations as described above, then select Group from the contextual menu. You are prompted to name the custom operation or you can accept the default name. Click OK, and the custom operation will be represented by a single box, both in your circuit and in the operations catalog.

You can now drag and drop the new operation throughout your circuit. Note that the operation is saved to this circuit but does not appear in the operations catalog for other circuits.

You can also build a custom operation directly in the OpenQASM code editor; see Create a custom operation in OpenQASM for more information.

Ungroup a custom or predefined operation

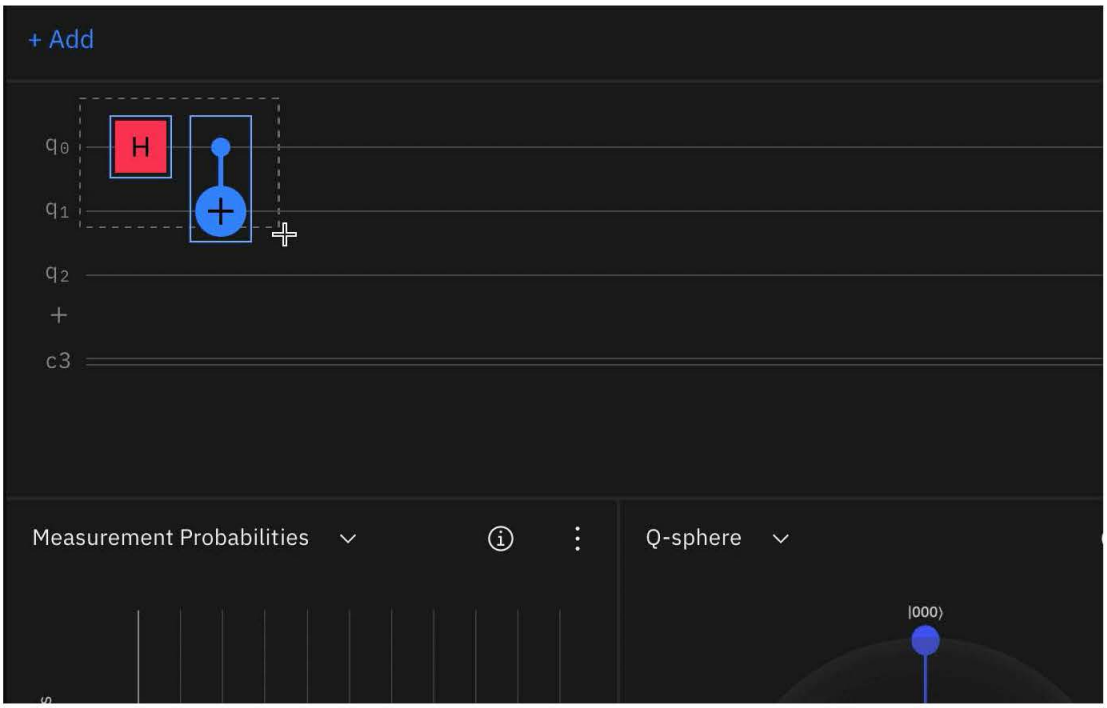
To ungroup the gates within a custom or predefined operation, click the operation on the Composer and select Ungroup from the contextual menu. You can now move the separate operations individually. When you ungroup an operation, each element in the former group executes independently, which may mean they execute in a different order from when they were grouped together.

Expand an operation's definition

To view the operations that constitute a custom or predefined operation without ungrouping them, click **Expand definition** from the contextual menu to see the defining gates. Click the icon again to collapse the definition.

Rename or delete a custom operation

To rename or delete a custom operation, right-click the operation in the operations catalog and select Rename or Delete. Deleting a custom operation from the operations catalog also deletes any instances of it on your circuit; however, deleting a custom operation from the circuit does not delete it from the operations catalog.



Add or remove registers

To add or remove quantum or classical registers, click Edit → Manage registers. You can increase or decrease the number of qubits or bits in your circuit and rename the registers. Click **Ok** to apply the changes. You can also simply click the register name (e.g., `q[0]`) and use the options in the contextual menu to quickly add or delete registers or qubits.

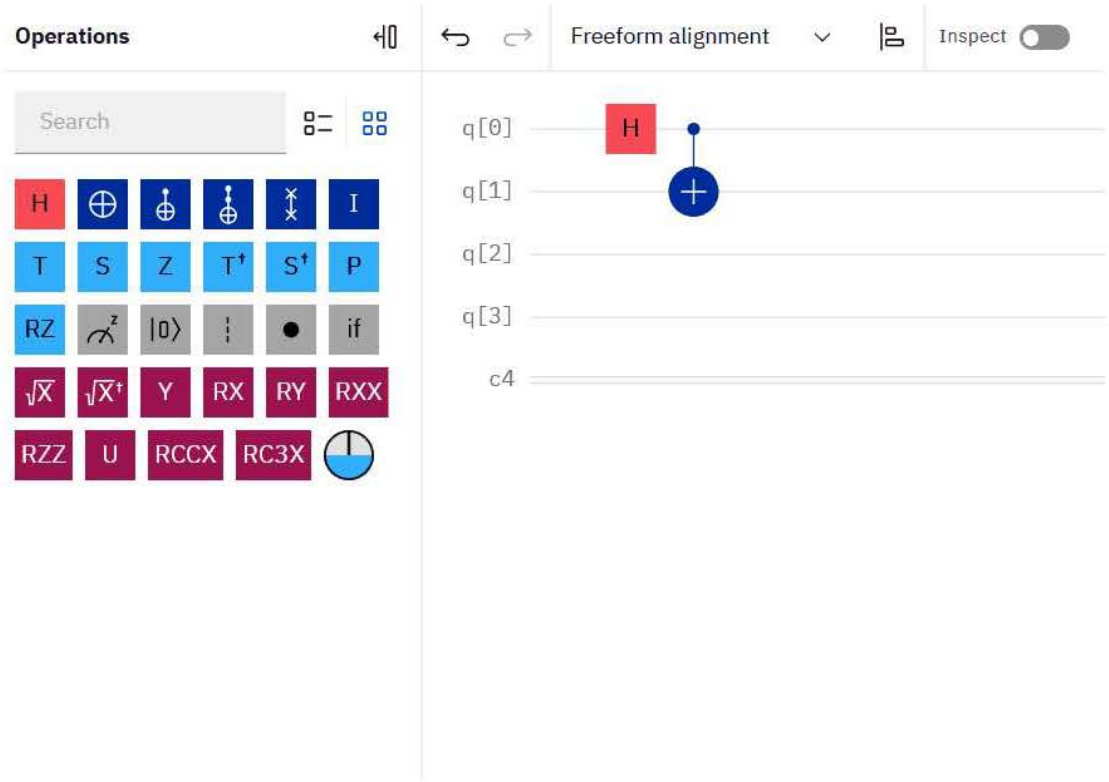
Add a conditional

To add a conditional to a gate, drag the `if` operation to the gate and set the parameters in the Edit operation panel that automatically opens. You can also double-click a gate to access the Edit operation panel, and set the conditional parameters that way.

Add a control modifier

A control modifier yields a gate whose original operation is now contingent on the state of the control qubit. For more details, right-click the control modifier symbol in the operations catalog, then click Info.

Drag the control modifier to a gate in order to add a control to it. A dot appears on the control qubit and a line connects it to the target qubit. To edit which qubit is the control or target, click the gate and select the Edit operation icon (or double-click the gate) to open the Edit operation panel, then specify your parameters. From the Edit operation panel, you can also remove a control from a qubit by clicking the x next to the qubit name.



Visualize with phase disks throughout your circuit

To visualize the state of all qubits at any point in your circuit, drag the phase disk icon from the operations catalog and place it anywhere in your circuit. A

column of barrier operations and a column of phase disks are added (one barrier operation and phase disk per qubit). Hover over each phase disk to read the state of the qubit at that point in the circuit. Note that adding the phase disks does not alter your circuit; they are merely a visualization tool.

Read more about the phase disk visualization here.

Export a circuit image

To export an image of your circuit, select File → Export. The Export options window opens, where you can choose a theme (light, dark, white on black, or black on white), a format (.svg or .png), and whether you want to apply a line wrap. After you've chosen your options, click Export.

Conditional reset

Several hardware systems can now perform conditional reset. If you have access to these systems, you can now initialize (or re-initialize) qubits in the $|0\rangle$ state, through measurement followed by a conditional X gate. Conditional reset will be rolled out to more systems, including open-access systems, over time.

3. Build your circuit with OpenQASM code

> **Note** IBM Quantum Composer currently supports OpenQASM 2.0.

- To open the code editor, click View → Panels → Code Editor.
- See the Composer operations glossary for OpenQASM references to gates and other operations.
- You can define your own custom operations; see Create a custom operation in OpenQASM.
- For more information on using the OpenQASM language, including sample lines of code, see the original research paper, Open Quantum Assembly Language. The table of OpenQASM language statements from the paper is reproduced below. The OpenQASM grammar can be found in Appendix A of the paper.

Statement	Description	Example
`OPENQASM 2.0;`	Denotes a file in OpenQASM format (see [a])	`OPENQASM 2.0;`

Statement	Description	Example
`qreg name[size];`	Declare a named register of qubits	`qreg q[5];`
`creg name[size];`	Declare a named register of bits	`creg c[5];`
`include “filename”;`	Open and parse another source file	`include “qelib1.inc”;`
`gate name(params) qargs`	Declare a unitary gate	(see text of paper)
`opaque name(params) qargs;`	Declare an opaque gate	(see text of paper)
`// comment text`	Comment a line of text	`// oops!`
`U(theta,phi,lambda) qubit|qreg;`	Apply built-in single-qubit gate(s) (see [b])	`U(pi/2,2\*pi/3,0) q[0];`
`CX qubit|qreg,qubit|qreg;`	Apply built-in CNOT gate(s)	`CX q[0],q[1];`
`measure qubit|qreg -> bit|creg;`	Make measurement(s) in Z basis	`measure q -> c;`
`reset qubit|qreg;`	Prepare qubit(s) in $\|0\rangle$	`reset q[0];`
`gatename(params) qargs;`	Apply a user-defined unitary gate	`crz(pi/2) q[1],q[0];`
`if(creg==int) qop;`	Conditionally apply quantum operation	`if(c==5) CX q[0],q[1];`
`barrier qargs;`	Prevent transformations across this source line	`barrier q[0],q[1];`

[a] This must appear as the first non-comment line of the file.

[b] The parameters theta, phi, and lambda are given by *parameter expressions*; for more information, see page 5 of the paper and Appendix A.

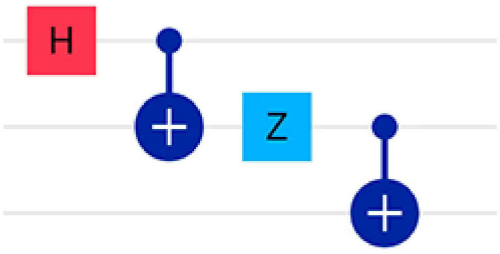
Create a custom operation in OpenQASM

You can define new unitary operations in the code editor (see the figure below for an example). The operations are applied using the statement `name(params) qargs;` just like the built-in operations. The parentheses are optional if there are no parameters.

To define a custom operation, enter it in the OpenQASM code editor using this format: `gatename(params) qargs;` . If you click **+Add** in the operations list, you will be prompted to enter a name for your custom operation, which you can then build in the code editor.

Once you have defined your custom operation, drag it onto the graphical editor and use the edit icon to fine-tune the parameters.

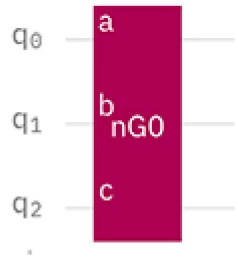
a) The gates to be included in the custom operation:



b) The code for the new operation:

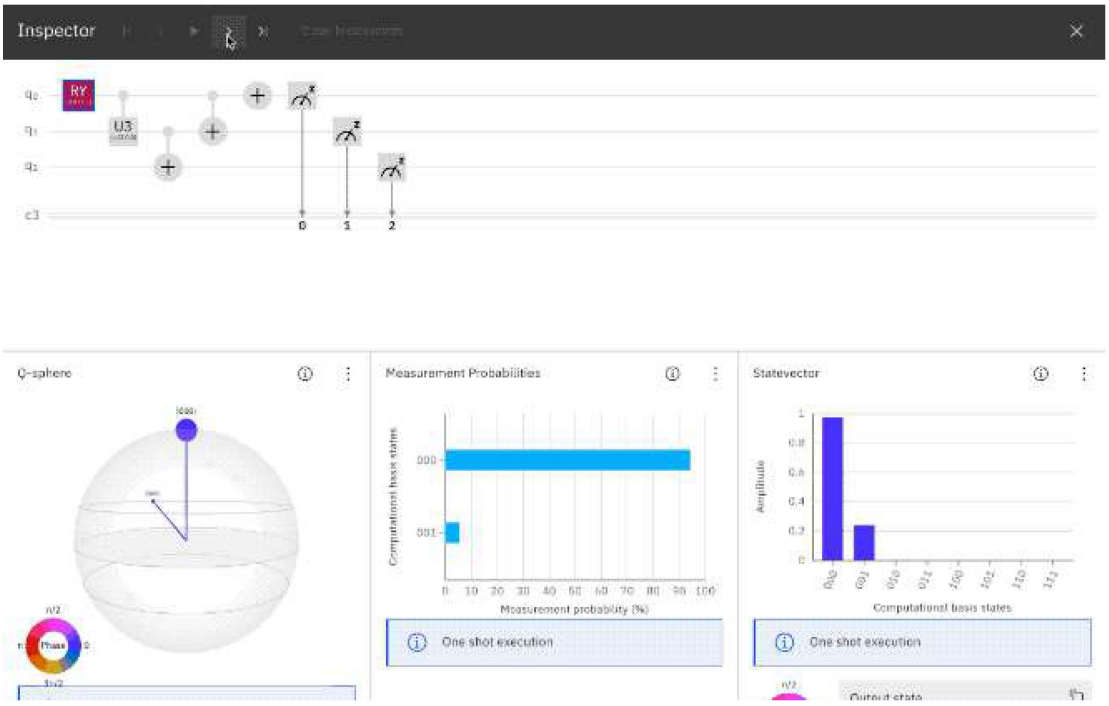
```
gate nG0 a, b, c  {
  h a;
  cx a, b;
  z b;
  cx b, c;
}
```

c) The new operation in the graphical editor:



4. Inspect your circuit, step-by-step

The Circuit Inspector de-mystifies the inner workings of the circuits you create. It steps through a simulation of your circuit, one layer at a time, so that you can see the state of the qubits as the computation evolves.



– In the **View** menu, select the panels for the visualizations that you want to use.

– Click the **Inspect** toggle in the toolbar. Note that once the Circuit Inspector is toggled on, you cannot add any more operations until it is turned off again.

– If you built your circuit with the Freeform alignment turned on, note that the Circuit Inspector automatically turns on the Left alignment.

– To move step-by-step through visualizations of your circuit's components, control the movement of the Circuit Inspector using the forward and rewind buttons.

Inspector controls, in order: jump to the beginning of a circuit, step backward, play from one break point to the next, step forward, jump to the end of a circuit.

– To inspect only some operations, click the operations you want to be inspected, and a colored overlay appears over each that indicates they will be included when you run the Circuit Inspector. To unselect an operation, click on it again, and the overlay disappears.
– To learn more about interpreting visualizations, see the Visualizations topic.
– To close the Inspector and return to editing your circuit, click the Inspect toggle in the toolbar.

> **Randomness in the simulator** The simulator creates randomness by generating results based on a *seed*. The seed is the initial value introduced into the algorithm that generates pseudorandom numbers. You can see the seed number in a box above Quantum Composer. You can also set the seed yourself by changing the value in the box.

Run circuits and view results

Follow the steps below to run quantum circuits on IBM Quantum systems or simulators, and view the results.

Choose your job settings

Click **Setup and run** in the upper right corner. In the window that opens, select an available system or simulator. Systems send the circuit to real quantum hardware, and simulators have the word 'simulator' in the name.

Next, you can choose your provider and set the number of shots (executions) of your circuit that the backend will perform.

You can optionally name the job and add tags in this panel as well. This will not change the name of the circuit.

If you have an open access account and choose a physical system, you will see the remaining number of jobs that you can have in the queue at any one time. This will not apply if you have a premium access account. Once a job has finished, the job limit will replenish. See more information in Fair-share queueing.

> **Note** When you run a circuit, it is automatically sent to the least busy system, unless you change the system in the Run settings. If you run the same circuit again, the system selection window will default to your previous choice.

Click "Run on xxxxx"

You can view the job's progress by clicking the Jobs icon.

View results

Once your job completes, the details are updated in the Jobs panel. You can also see more information by clicking **See more details**.

The Jobs results page displays run details, diagrams of the original and the transpiled circuit, a histogram of the results, and OpenQASM and Qiskit tabs to view both original and transpiled circuits in OpenQASM or Qiskit.

You can download the circuits and the histogram by clicking the menu in the upper right corner of each diagram, then selecting a format for the download (PNG, PDF, or SVG; additionally, you can export the histogram as a CSV file). You can open the OpenQASM circuits directly in Composer, and you can open the Qiskit circuits directly in Quantum Lab.

Visualizations

The live visualizations in IBM Quantum Composer show you different views of how quantum circuits affect the state of a collection of qubits. Each type of live visualization is explained in detail below.

> **Randomness in the simulator** The live visualizations come from a single-shot statevector simulator, which is different from the system specified in the Run settings, which can have multiple shots. The simulator creates randomness by generating results based on a *seed*. The seed is the initial value introduced into the algorithm that generates pseudorandom numbers and simulates quantum randomness. Because the seed is fixed during your circuit session, your live visualizations will be repeatable. However, when you close your circuit and re-open it, the seed will have a new value, so you may see a different visualization. You can set the seed yourself by changing the value in the box (up to four digits) after the words `Visualizations seed` above the Composer, and observe how the live visualizations for your circuit change.

View visualizations

The live visualizations are shown in windows at the bottom of the Quantum Composer workspace (except the phase disk, which appears at the end of each qubit wire). You can choose any combination of statevector, probabilities, and q-sphere visualizations to appear at the bottom of the workspace. Select or unselect visualizations in the **View** menu.

Download visualizations

Download one of the visualizations at the bottom of the Quantum Composer workspace by clicking the More options menu in the visualization window. You can download visualizations as an SVG, a PNG, or a CSV of the underlying data. You can also download the visualization images of the measurement probabilities and statevector histograms as a PDF.

Phase disk

A single-qubit state can be represented as

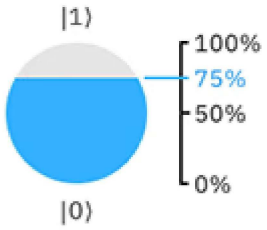
$$|\psi\rangle = \sqrt{1-p}|0\rangle + e^{j\varphi}\sqrt{p}|1\rangle,$$

where p is the probability that the qubit is in the $|1\rangle$ state, and φ is the quantum phase. p is strongly analogous to a classical probabilistic bit. For $p = 0$, the qubit is in the $|0\rangle$ state, for $p = 1$ the qubit is in the $|1\rangle$ state, and for $p = 1/2$ the qubit is a 50/50 mixture. We call this a superposition as, unlike classical bits, this mixture can have a quantum phase. The phase disk visualizes this state.

The phase disk at the terminus of each qubit in IBM Quantum Composer gives the local state of each qubit at the end of the computation. The components of the phase disk are described below.

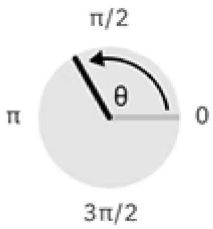
Probability the qubit is in the $|1\rangle$ state

The probability that the qubit is in the $|1\rangle$ state is represented by the blue disk fill.

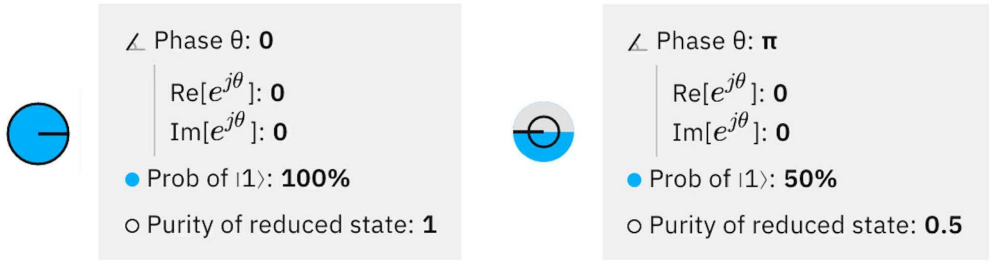


Quantum phase

The quantum phase of the qubit state is given by the line that extends from the center of the diagram to the edge of the gray disk (which rotates counterclockwise around the center point).



Example: phase disks for two different qubits



Two examples of the phase disk visualization. The first example is a $|1\rangle$ state and the second shows the $(|0\rangle - |1\rangle)/\sqrt{2}$ state with a nonzero relative phase.

Connection to the Bloch sphere

The phase disk, which contains all the information in the Bloch sphere, is a two-dimensional representation of a qubit. To convert to the Bloch sphere representation: $x = 2\sqrt{p(1-p)}\mathrm{Re}[e^{j\varphi}]$, $y = 2\sqrt{p(1-p)}\mathrm{Im}[e^{j\varphi}]$, and $z = 1 - 2p$.

N-qubit states: maximum 15 qubits

An N-qubit quantum state takes the form

$$|\psi\rangle = \sqrt{1-p}|0...0\rangle + \sum_{k=1}^{2^N-1} e^{j\varphi_k}\sqrt{p_k}|k\rangle,$$

where p_k is the probability that the qubits are in the state $|k\rangle$ with quantum phase φ_k with respect to the $|0...0\rangle$ state. $p = \sum_{k\neq 0} p_k$ is the probability that the qubits are not in the ground state $|0...0\rangle$. Here it is simple to see that for an N-qubit quantum state there are $2^N - 1$ probabilities and $2^N - 1$ phases. The phase disk fails to represent this state, as N-qubit phase disks would only contain N probabilities and N phases; this is because most states are entangled and are not separable into independent single-qubit quantum states. To represent that full information is not contained in this visualization, we introduce the reduced purity as a component in the phase disk.

Reduced purity of the qubit state

The radius of the black ring represents the reduced purity of the qubit state, which for qubit j in an N-qubit state $|\psi\rangle$ is given by $\mathrm{Tr}\left[\mathrm{Tr}_{i\neq j}[|\psi\rangle\langle\psi|]^2\right]$.

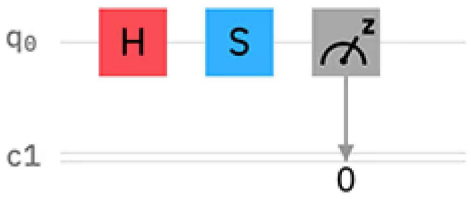
The reduced purity for a single qubit is in the range $[0.5, 1]$; a value of one indicates that the qubit is not entangled with any other party. In contrast, a reduced purity of 0.5 shows that the qubit is left in the completely mixed state, and has some level of entanglement over the remaining $N - 1$ qubits, and possibly even the environment.

Probabilities view

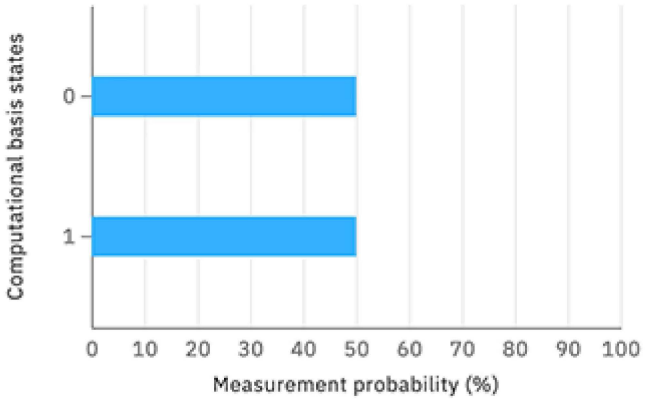
8-qubit limit

This view visualizes the probabilities of the quantum state as a bar graph. The horizontal axis labels the computational basis states. The vertical axis measures the probabilities in terms of percentages. In this view the quantum phases are not represented, and is therefore an incomplete representation. However, it is useful for predicting the outcomes if each qubit is measured and the value stored in its own classical bit.

Consider the following quantum circuit and its probabilities view:



A circuit consisting of a column of Hadamards that creates an equal superposition of the computational basis states, followed by a two-qubit controlled-Z (CZ) gate.



A quantum circuit and its probabilities view.

The circuit puts the two qubits into the state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$. The computational basis states are $|00\rangle$, $|10\rangle$, $|01\rangle$, and $|11\rangle$. The probabilities for each of the computational states is 1/4.

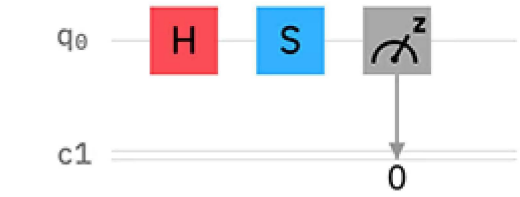
Q-sphere view

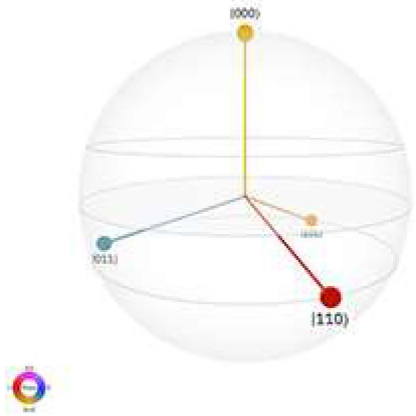
5-qubit limit

The q-sphere represents the state of a system of one or more qubits by associating each computational basis state with a point on the surface of a sphere. A node is visible at each point. Each node's radius is proportional to the probability (p_k) of its basis state, whereas the node color indicates the quantum phase (φ_k).

The nodes are laid out on the q-sphere so that the basis state with all zeros (e.g., $|000\rangle$) is at its north pole, and the basis state with all ones (e.g., $|111\rangle$) is at its south pole. Basis states with the same number of zeros (or ones) lie on a shared latitude of the q-sphere (e.g., $|001\rangle$, $|010\rangle$, $|100\rangle$). Beginning at the north pole of the q-sphere and progressing southward, each successive latitude has basis states with a greater number of ones; the latitude of a basis state is determined by its Hamming distance from the zero state. The q-sphere contains complete information about the quantum state in a compact representation.

Consider the following quantum circuit and its q-sphere:





A quantum circuit and the q-sphere representing the state the circuit creates.

You can select, hold, and drag to rotate the q-sphere. To return the q-sphere to its default orientation, select the **rewind-arrow button** to the top right of the q-sphere.

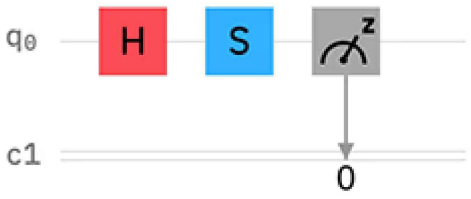
> **What is the difference between a Bloch sphere and a q-sphere?** It is important to highlight that the q-sphere is **not** the same as the Bloch sphere, even for a single qubit. Indeed, like the phase disk, the Bloch sphere gives a *local* view of the quantum state, where each qubit is viewed on its own. When trying to understand how registers of qubits (multi-qubit states) behave upon the application of quantum circuits, it is more insightful to take a *global* view and look at the quantum state in its entirety. The q-sphere provides a visual representation of the quantum state, and thus this *global* viewpoint. Therefore, when exploring quantum applications and algorithms on small numbers of qubits, the q-sphere should be the primary visualization method.

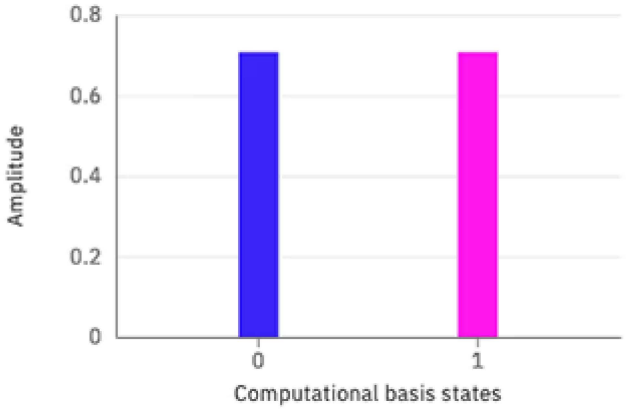
Statevector view

6-qubit limit

It is common to call $\sqrt{p_k}e^{i\varphi_k}$ the quantum amplitude. This view visualizes the quantum amplitudes as a bar graph. The horizontal axis labels the computational basis states. The vertical axis measures the magnitude of the amplitudes ($\sqrt{p_k}$) associated with each computational basis state. The color of each bar represents the quantum phase (φ_k).

Consider the following quantum circuit and its statevector view:





The statevector at the terminus of the above circuit. The color wheel maps phase angle to color. The output state is expressed as a list of complex numbers.

The circuit puts the two qubits into the state $|\psi\rangle = (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2$. The computational basis states are $|00\rangle$, $|10\rangle$, $|01\rangle$, and $|11\rangle$. The magnitudes of the amplitudes are $1/2$, and the quantum phases with respect to the ground state are 0 for $|01\rangle$ and $|10\rangle$, and π for $|11\rangle$.

Composer operations glossary

This page is a reference that defines the various classical and quantum operations you can use to manipulate qubits in a quantum circuit. Quantum operations include quantum gates, such as the Hadamard gate, as well as operations that are not quantum gates, such as the measurement operation.

Each entry below provides details and the OpenQASM reference for each operation. See the topic on Build your circuit with OpenQASM code for more information.

The q-sphere image in each gate entry below shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$, where n is the number of qubits needed to support the gate. See the q-sphere topic for more information on this visualization.

You can define a custom operation to use in IBM Quantum Composer. For instructions, see the Create a custom operation in OpenQASM topic.

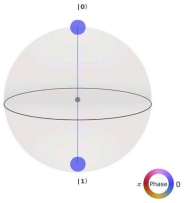
> **Note** The gate colors are slightly different in the light and dark themes. The colors from the light theme are shown here.

Click a quantum operation below to view its definition. Operations no longer used in Circuit Composer are listed in the Obsolete operations section as a historical reference.

Classical gates

NOT Gate

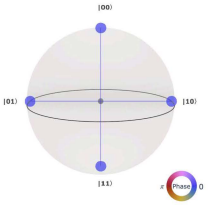
The NOT gate, also known as the Pauli X gate, flips the $|0\rangle$ state to $|1\rangle$, and vice versa. The NOT gate is equivalent to RX for the angle π or to 'HZH'.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`x q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \|i\rangle$, where n is the number of qubits needed to support the gate.

CNOT Gate

The controlled-NOT gate, also known as the controlled-x (CX) gate, acts on a pair of qubits, with one acting as 'control' and the other as 'target'. It performs a NOT on the target whenever the control is in state $|1\rangle$. If the control qubit is in a superposition, this gate creates entanglement.

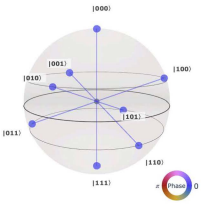
All unitary circuits can be decomposed into single qubit gates and CNOT gates. Because the two-qubit CNOT gate costs much more time to execute on real hardware than single qubit gates, circuit cost is sometimes measured in the number of CNOT gates.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`cx q[0], q[1];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \|i\rangle$, where n is the number of qubits needed to support the gate.

Toffoli gate

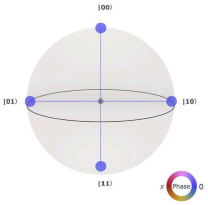
The Toffoli gate, also known as the double controlled-NOT gate (CCX), has two control qubits and one target. It applies a NOT to the target only when both controls are in state $|1\rangle$.

The Toffoli gate with the Hadamard gate is a universal gate set for quantum computing.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`ccx q[0], q[1], q[2];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \|i\rangle$, where n is the number of qubits needed to support the gate.

SWAP gate

The SWAP gate swaps the states of two qubits.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`swap q[0], q[1];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

Identity gate

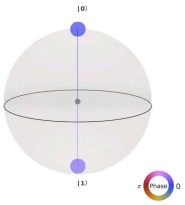
The identity gate (sometimes called the Id or the I gate) is actually the absence of a gate. It ensures that nothing is applied to a qubit for one unit of gate time.

Composer reference	Qasm reference
	`id q[0];`

Phase gates

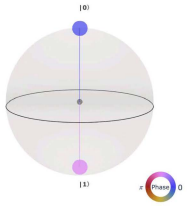
T gate

The T gate is equivalent to RZ for the angle $\pi/4$. Fault-tolerant quantum computers will compile all quantum programs down to just the T gate and its inverse, as well as the Clifford gates.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`t q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

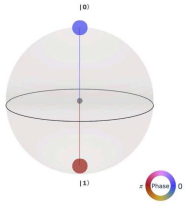
S gate

The S gate applies a phase of i to the $|1\rangle$ state. It is equivalent to RZ for the angle $\pi/2$. Note that S=P($\pi/2$).

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
S	`s q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

Z gate

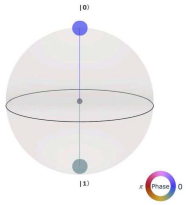
The Pauli Z gate acts as identity on the $|0\rangle$ state and multiplies the sign of the $|1\rangle$ state by -1. It therefore flips the $|+\rangle$ and $|-\rangle$ states. In the +/- basis, it plays the same role as the NOT gate in the $|0\rangle/|1\rangle$ basis.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
Z	`z q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

$T^\dagger$ gate

Also known as the Tdg or T-dagger gate.

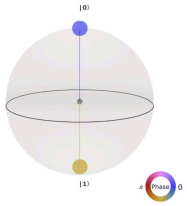
The inverse of the T gate.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
T†	`tdg q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

$S^\dagger$ gate

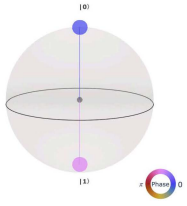
Also known as the Sdg or S-dagger gate.

The inverse of the S gate.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
S†	`sdg q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

Phase gate

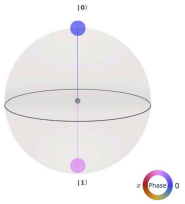
The Phase gate (previously called the U1 gate) applies a phase of $e^{i\theta}$ to the $|1\rangle$ state. For certain values of θ, it is equivalent to other gates. For example, P(π)=Z, P($\pi/2$)=S, and P($\pi/4$)=T. Up to a global phase of $e^{i\theta/2}$, it is equivalent to RZ(θ).

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
P	`p(theta) q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `theta` is $\pi/2$.

RZ gate

The RZ gate implements $exp(-i\frac{\theta}{2}Z)$. On the Bloch sphere, this gate corresponds to rotating the qubit state around the z axis by the given angle.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
RZ	`rz(angle) q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `angle` is $\pi/2$. Therefore, this is the angle used in the q-sphere visualization.

Non-unitary operators and modifiers

Reset operation

The reset operation returns a qubit to state $|0\rangle$, irrespective of its state before the operation was applied. It is not a reversible operation.

Composer reference	OpenQASM reference
|0⟩	`reset q[0];`

Measurement

Measurement in the standard basis, also known as the z basis or computational basis. Can be used to implement any kind of measurement when combined with gates. It is not a reversible operation.

Composer reference	OpenQASM reference
	`measure q[0];`

Control modifier

A control modifier yields a gate whose original operation is now contingent on the state of the control qubit. When the control is in the $|1\rangle$ state, the target qubit(s) undergo the specified unitary evolution. In contrast, no operation is performed if the control is in the $|0\rangle$ state. If the control is in a superposition state, then the resulting operation follows from linearity.

Drag the control modifier to a gate in order to add a control to it. Dots will appear above and below the gate, on the qubit wires that can be targets that control; click one or more dots to assign the target to one or more qubits. You can also assign a control by right-clicking a gate.

To remove a control, right-click the gate and select the option to remove control.

Composer reference	OpenQASM reference
	`c`

Barrier operation

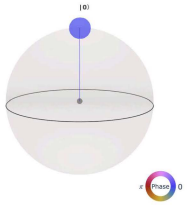
To make your quantum program more efficient, the compiler will try to combine gates. The barrier is an instruction to the compiler to prevent these combinations being made. Additionally, it is useful for visualizations.

Composer reference	OpenQASM reference
	`barrier q;`

Hadamard gate

H gate

The H, or Hadamard, gate rotates the states $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$, respectively. It is useful for making superpositions. If you have a universal gate set on a classical computer and add the Hadamard gate, it becomes a universal gate set on a quantum computer.

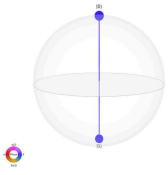
Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
H	`h q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

Quantum gates

$\sqrt{X}$ gate

Also known as the square-root NOT gate.

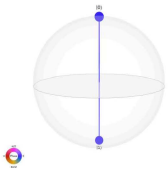
This gate implements the square-root of X, $\sqrt{X}$. Applying this gate twice in a row produces the standard Pauli-X gate (NOT gate). Like the Hadamard gate, $\sqrt{X}$ creates an equal superposition state if the qubit is in the state $|0\rangle$, but with a different relative phase. On some hardwares, it is a native gate that can be implemented with a $\pi/2$ or X90 pulse.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
√X	`sx q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

$\sqrt{X}^\dagger$ gate

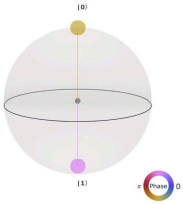
Also known as the SXdg or square-root NOT-dagger gate.

This is the inverse of the $\sqrt{X}$ gate. Applying it twice in a row produces the Pauli-X gate (NOT gate), since the NOT gate is its own inverse. Like the $\sqrt{X}$ gate, this gate can be used to create an equal superposition state, and it too is natively implemented on some hardwares using an X90 pulse.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`sxdg q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

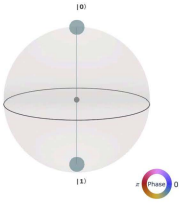
Y gate

The Pauli Y gate is equivalent to Ry for the angle π. It is equivalent to applying X and Z, up to a global phase factor.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`y q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

RX gate

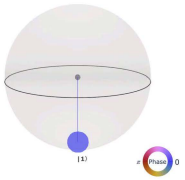
The RX gate implements $exp(-i\frac{\theta}{2}X)$. On the Bloch sphere, this gate corresponds to rotating the qubit state around the x axis by the given angle.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
	`rx(angle) q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `angle` is $\pi/2$. Therefore, this is the angle used in the q-sphere visualization.

RY gate

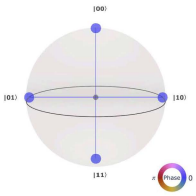
The RY gate implements $exp(-i\frac{\theta}{2}Y)$. On the Bloch sphere, this gate corresponds to rotating the qubit state around the y axis by the given angle and does not introduce complex amplitudes.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
RY	`ry(angle) q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `angle` is $\pi/2$. Therefore, this is the angle used in the q-sphere visualization below.

RXX gate

The RXX gate implements $\exp(-i\theta/2X\otimes X)$. The Mølmer–Sørensen gate, the native gate on ion-trap systems, can be expressed as a sum of RXX gates.

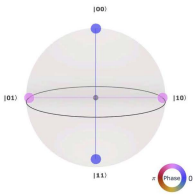
Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
RXX	`rxx(angle) q[0], q[1];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}\|i\rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `angle` is $\pi/2$.

RZZ gate

The RZZ gate requires a single parameter: an angle expressed in radians. This gate is symmetric; swapping the two qubits it acts on doesn't change

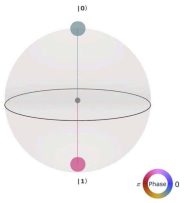
anything.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
RZZ	`rzz(angle) q[0], q[1];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \lvert i \rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `angle` is $\pi/2$.

U gate

(Previously called the U3 gate) The three parameters allow the construction of any single-qubit gate. Has a duration of one unit of gate time.

Composer reference	OpenQASM reference	Q-Sphere	Note about q-sphere representations
U	`u(theta, phi, lam) q[0];`		The q-sphere representation shows the state after the gate operates on the initial equal superposition state $\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \lvert i \rangle$, where n is the number of qubits needed to support the gate.

In IBM Quantum Composer, the default value for `angle` is $\pi/2$.