QuantumSphere

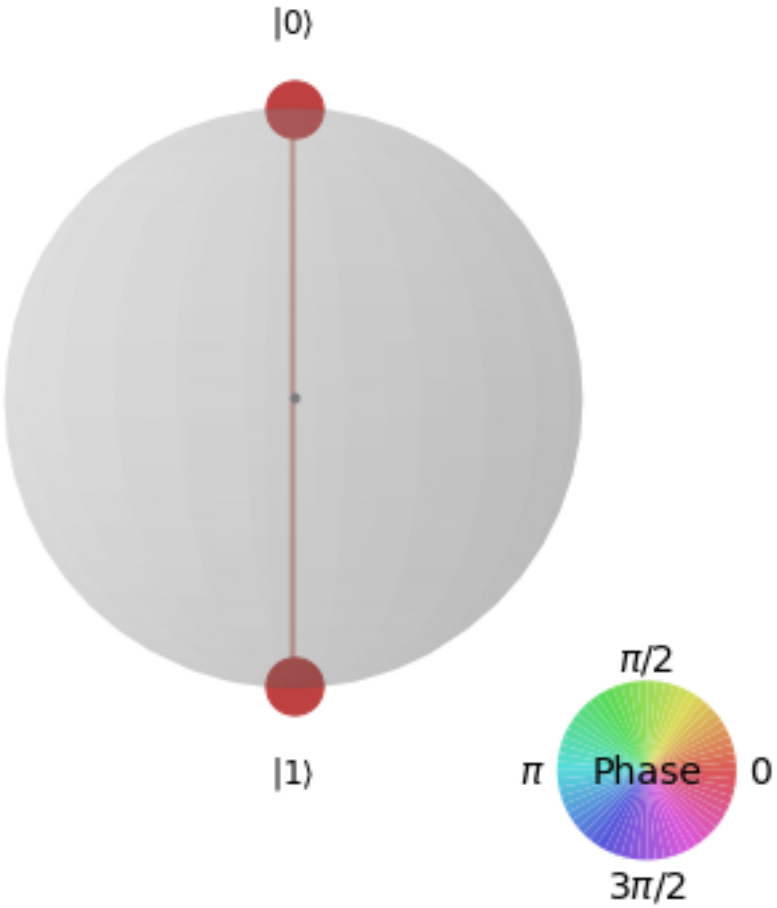
November 28, 2020

[2]:
```
import numpy as np
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector
from qiskit.visualization import plot_state_qsphere
```

[6]:
```
n = 1
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
statevec = Statevector.from_instruction(qc).data
print(statevec)
plot_state_qsphere(statevec)
```

```
[0.70710678+0.j 0.70710678+0.j]
```

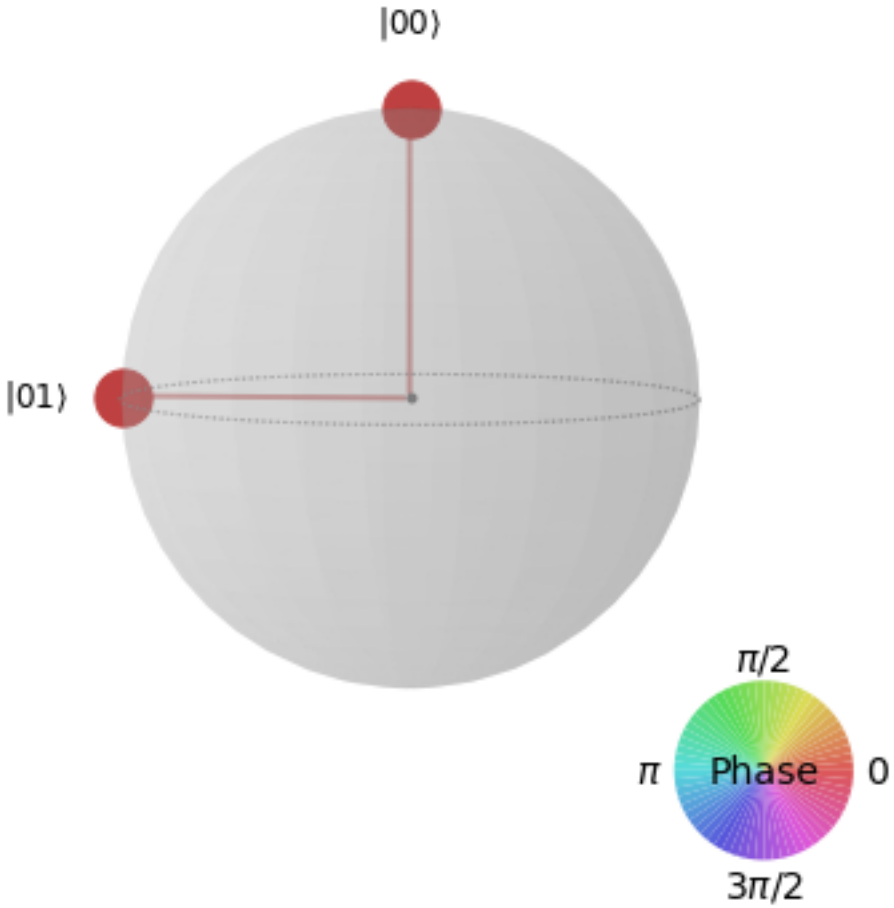
[6]:



[11]:
```
n = 2
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
# qc.cx(0,1)
statevec = Statevector.from_instruction(qc).data
print(statevec)
plot_state_qsphere(statevec)
```

```
[0.70710678+0.j 0.70710678+0.j 0.        +0.j 0.        +0.j]
```

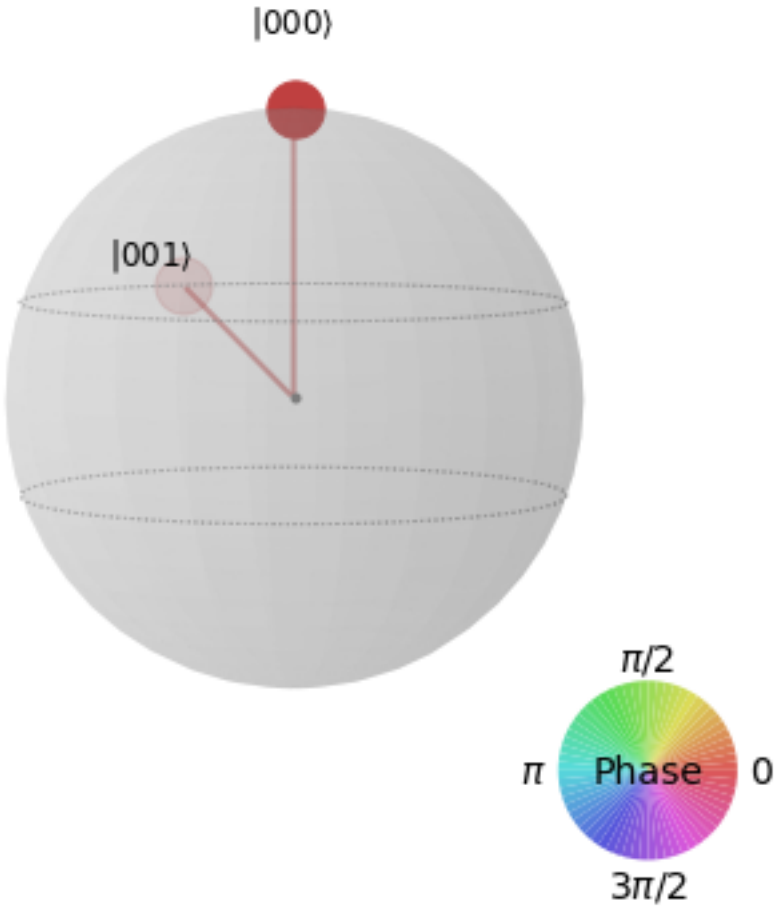
[11]:



[9]:
```
n = 3
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
# qc.cx(0,1)
statevec = Statevector.from_instruction(qc).data
print(statevec)
plot_state_qsphere(statevec)
```

```
[0.70710678+0.j 0.70710678+0.j 0.        +0.j 0.        +0.j
 0.        +0.j 0.        +0.j 0.        +0.j 0.        +0.j]
```

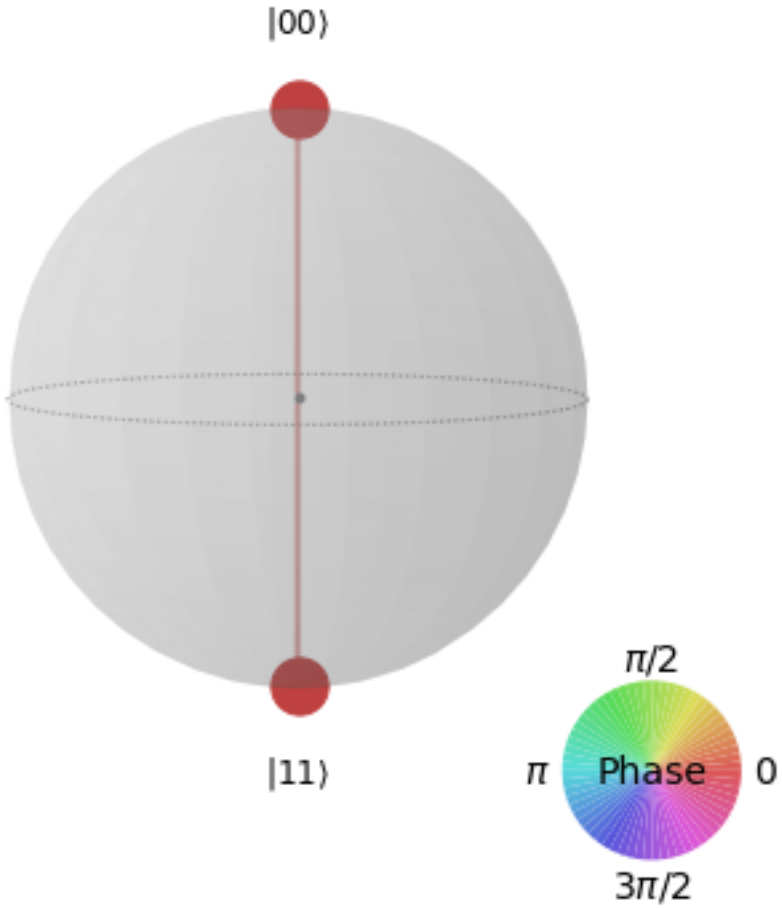
[9]:



[15]:
```
n = 2
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
qc.cx(0,1) # Applying CNOT GATE to get Bell state as output
statevec = Statevector.from_instruction(qc).data
print(statevec)
plot_state_qsphere(statevec)
```

```
[0.70710678+0.j 0.        +0.j 0.        +0.j 0.70710678+0.j]
```

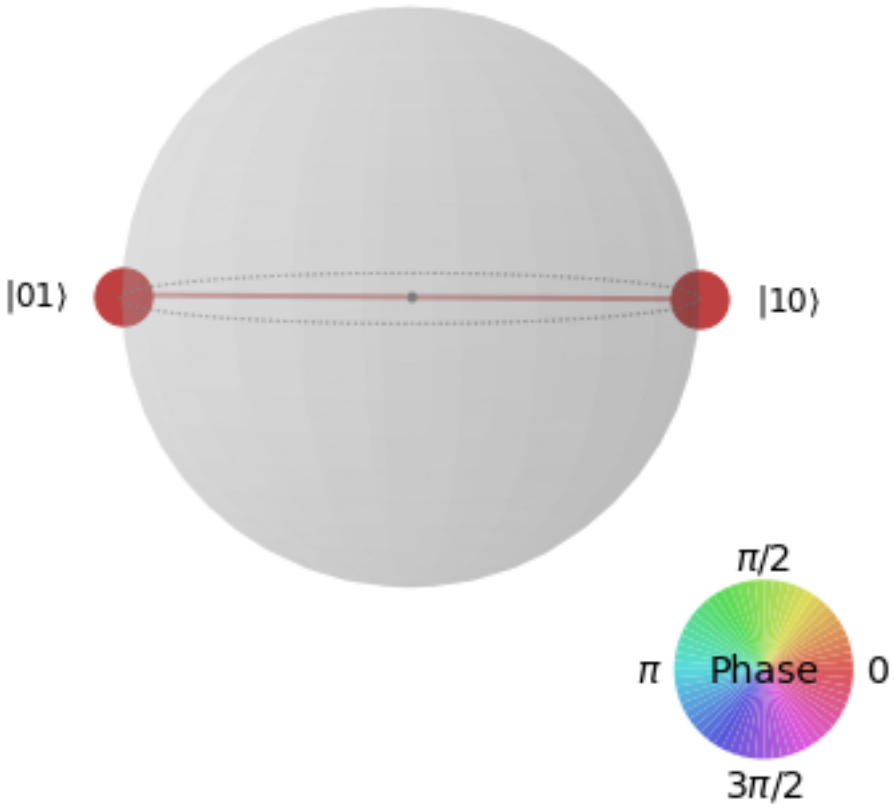
[15]:



[14]:
```
n = 2
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
qc.cx(0,1) # Applying CNOT GATE to get Bell state as output
qc.x(1) # Applying another not gate in the end to one of the qubits this will␣
 ↪result state |01> + |10>
statevec = Statevector.from_instruction(qc).data
print(statevec)
plot_state_qsphere(statevec)
```

```
[0.         +0.j 0.70710678+0.j 0.70710678+0.j 0.         +0.j]
```

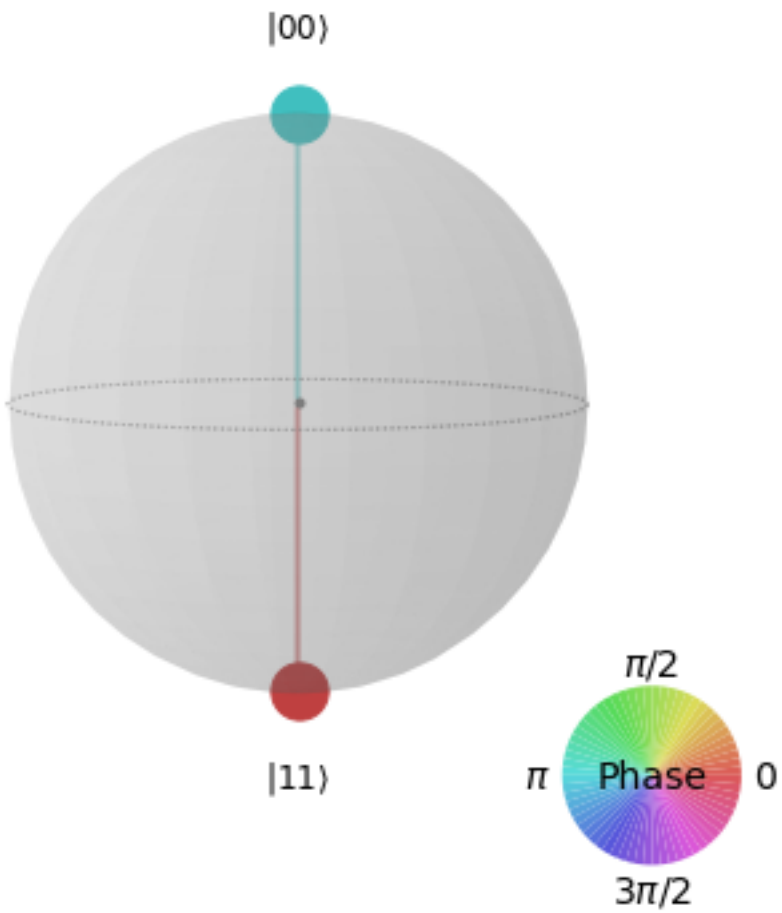
[14]:



[16]:
```
n = 2
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
qc.cx(0,1) # Applying CNOT GATE to get Bell state as output
qc.z(1) # Applying another not gate in the end to one of the qubits this will␣
 ↪result state |01> + |10> but in z phase
statevec = Statevector.from_instruction(qc).data
print(statevec)
plot_state_qsphere(statevec)
```

```
[ 0.70710678+0.j  0.        +0.j  0.        +0.j -0.70710678+0.j]
```

[16]:

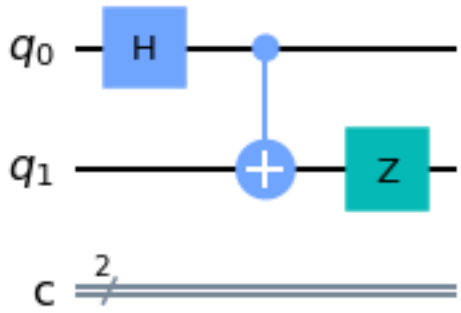


[18]:
```
# we can also plot the circuit as well.
# The CKT will represent our algorithm

n = 2
qc = QuantumCircuit(n,n)
qc.h(0) # Hadamard Gate
# we are in equal superposition with 0 and 1
qc.cx(0,1) # Applying CNOT GATE to get Bell state as output
qc.z(1) # Applying another not gate in the end to one of the qubits this will␣
 ↪result state |01> + |10> but in z phase
statevec = Statevector.from_instruction(qc).data
print(statevec)
qc.draw()
# plot_state_qsphere(statevec)
```

```
[ 0.70710678+0.j  0.        +0.j  0.        +0.j -0.70710678+0.j]
```

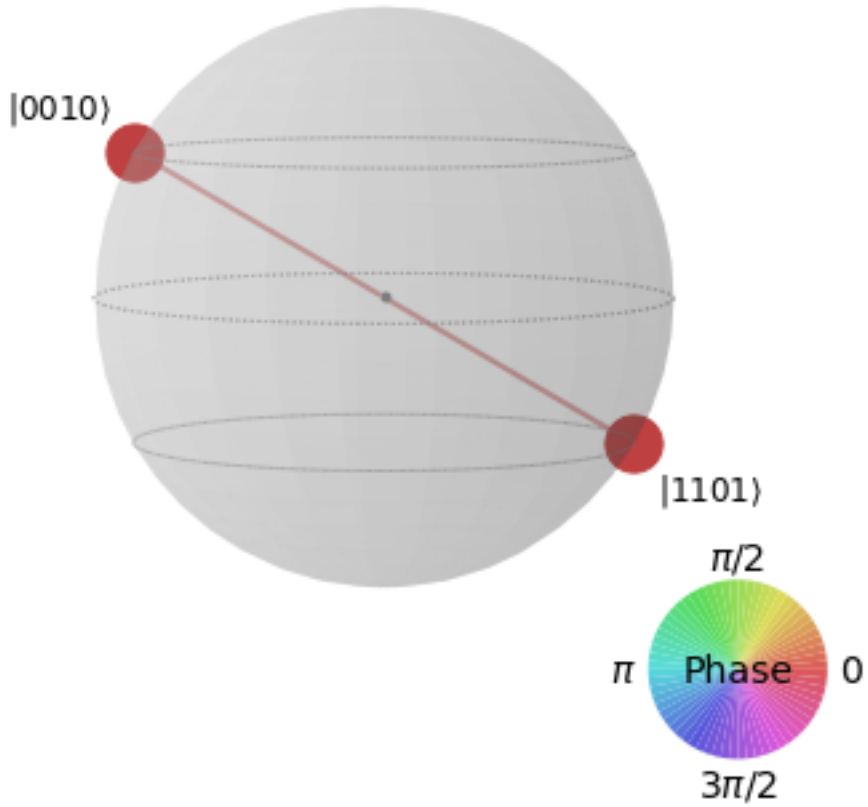
[18]:



[20]:

```
# qiskit logo
qc = QuantumCircuit(4)
qc.h(0)
for i in range(3):
    qc.cx(0,i+1)
qc.x(1)
statevec = Statevector.from_instruction(qc).data
plot_state_qsphere(statevec)
```

[20]:


|0010⟩
|1101⟩
π/2
π
Phase
0
3π/2

[ ]: