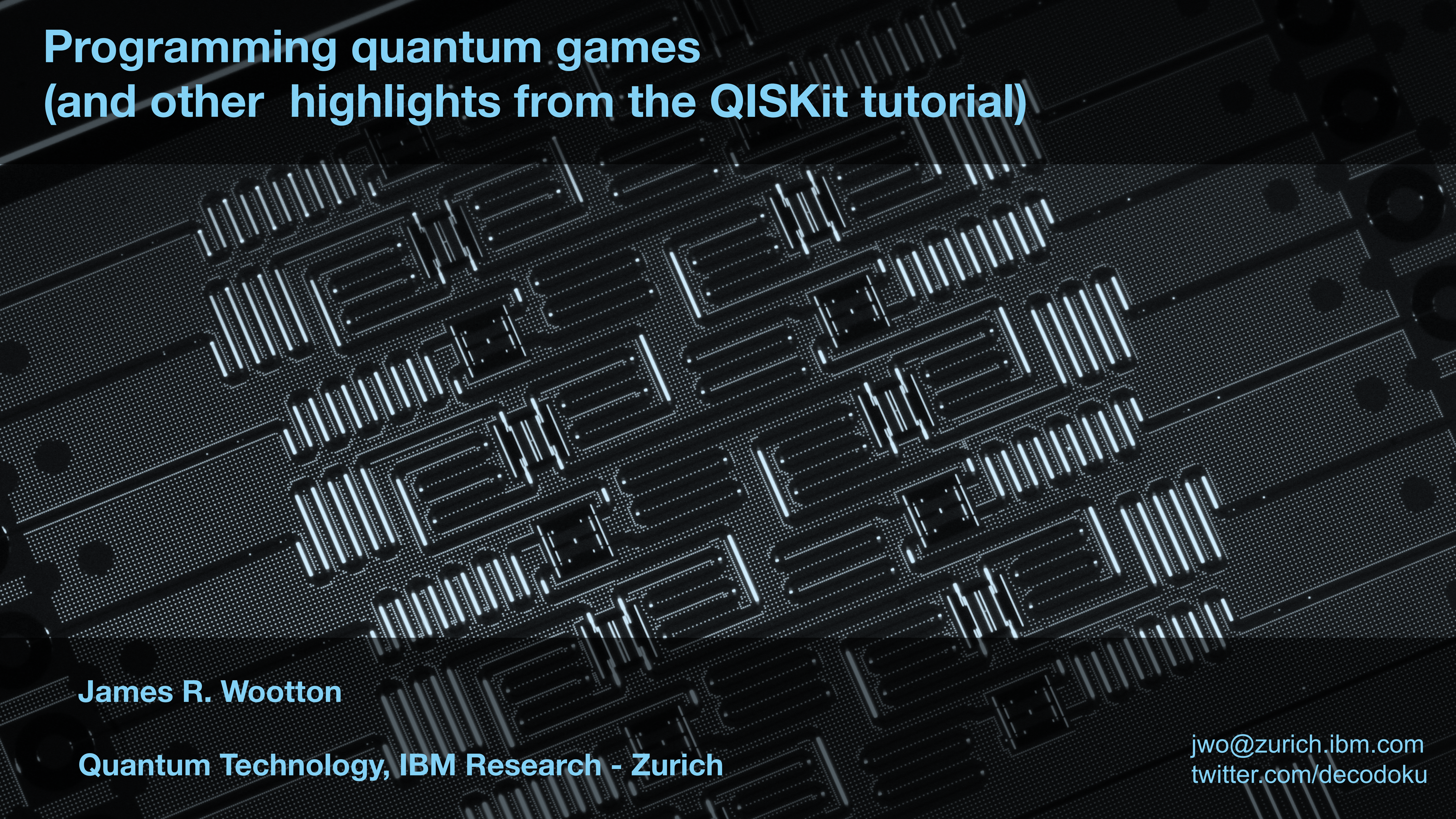
Programming quantum games
(and other highlights from the QISKit tutorial)



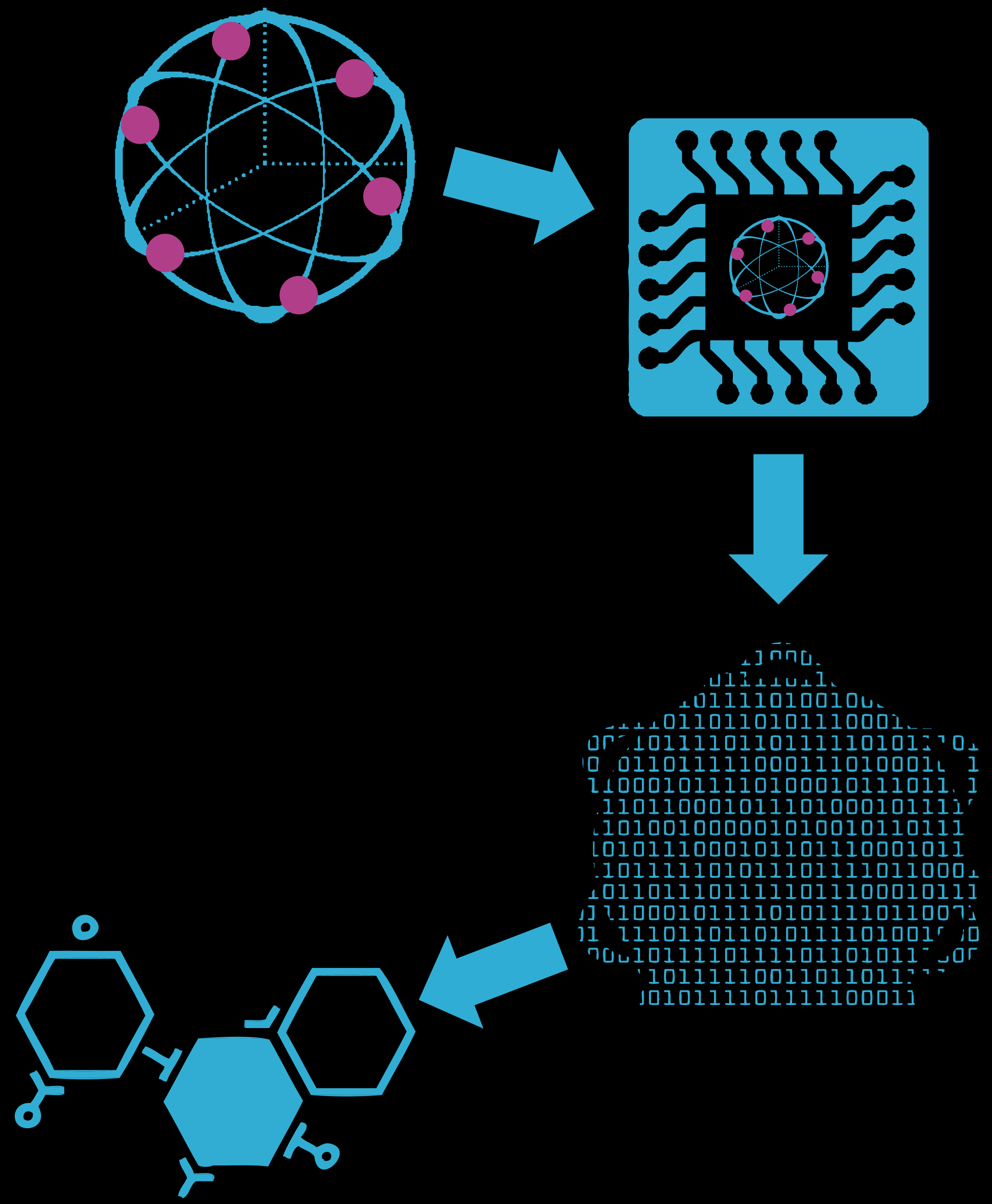
James R. Wootton

Quantum Technology, IBM Research - Zurich

jwo@zurich.ibm.com
twitter.com/decodoku

Your first quantum program

- When you start programming, you start small
  - Make something happen
  - Print to screen
  - Start interacting things



- You won’t save the world in your first program!

‘Hello World’

- ‘Hello World’ is the classic example
- Just make a program that prints some text to screen

```
print(‘Hello World’)
```

‘Hello World’

- ‘Hello World’ is the classic example
- Just make a program that prints some text to screen

```
print(‘Hello World’)
```

(or toast)

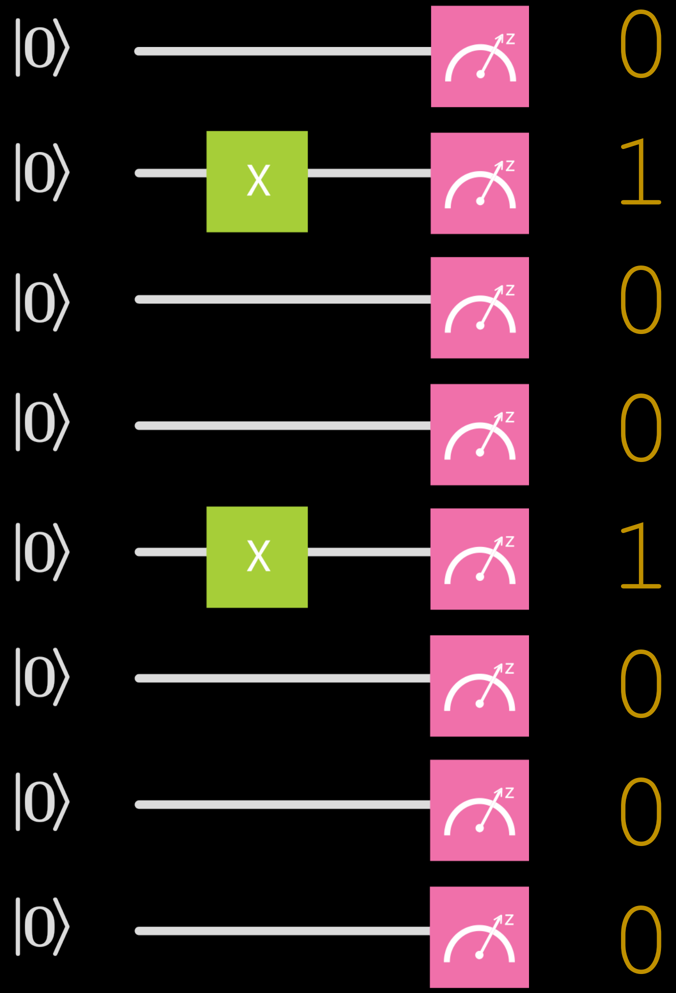
flickr.com/oskay

‘Hello World’

- Do this with a quantum computer
- Use the fact that computers encode in binary

1. Convert ‘Hello World’ to binary
2. **Encode the binary in qubits**
3. **Extract the string from qubits**
4. Convert binary to letters
5. Print to screen

- Requires 88 qubits
- Works just fine without the quantum part

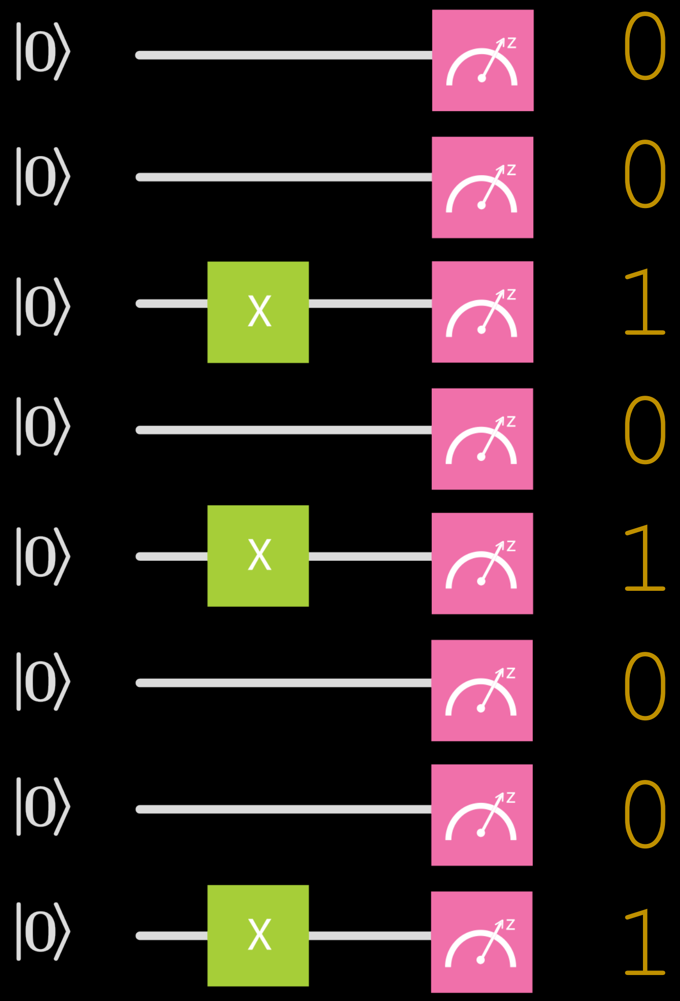


Quantum ‘Hello World’

- For :) we need only 16 qubits
- Can be done on the cloud with an IBM device
- To use the quantumness, we can superpose emoticons!

```
;)  =  0011101100101001
8)  =  0011100000101001
```

- Where the bit strings agree, this is done as before



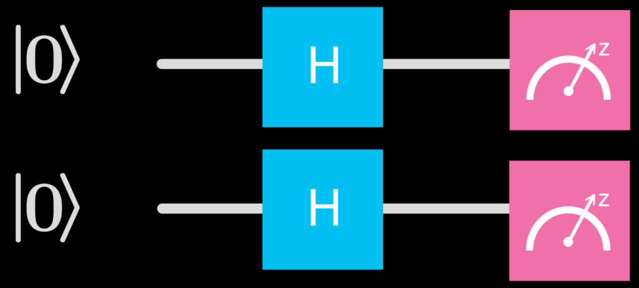
Quantum ‘Hello World’

- Where they differ, we need a superposition

;) = 001110**11**00101001

8) = 001110**00**00101001

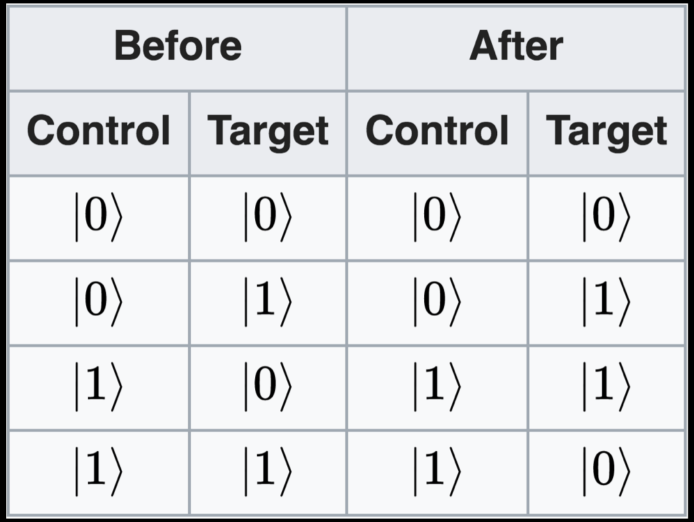
- H creates a superposition of 0 and 1



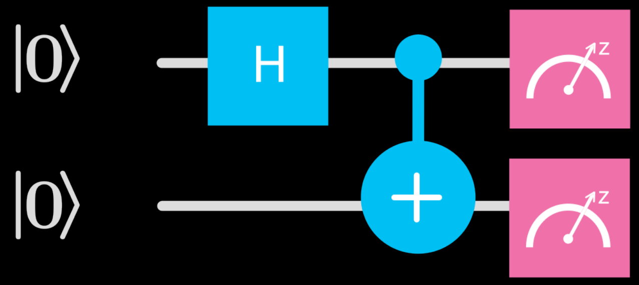
- Two create a superposition of 00, 01, 10 and 11
  - Not what we want: we need correlations

- Use one H for the superposition, and a CNOT to ‘spread’ it

00 → 00 + 10 → 00 + 11



Before		After	
Control	Target	Control	Target
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

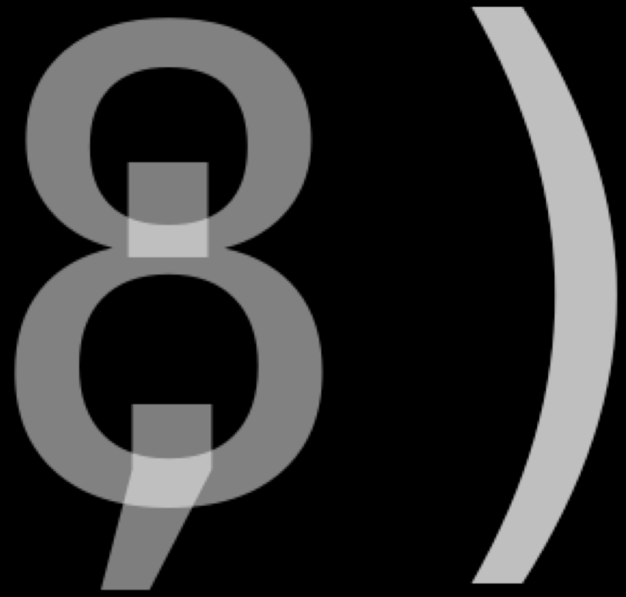


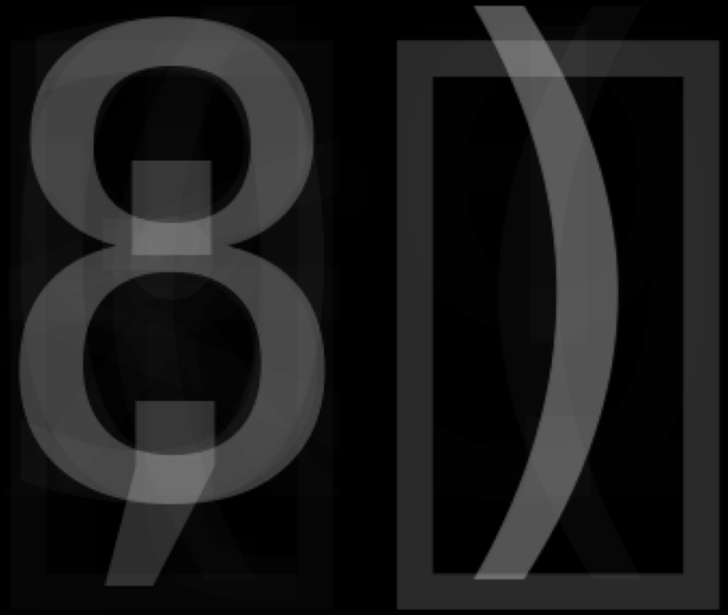
Quantum ‘Hello World’

- Measuring a superposition gives a random outcome

```
shots=1024, { ‘:)’:501, ‘8)’:523 }
```

- We can use the statistics to create an image





- Can show us the nature of a real device

Quantum ‘Hello World’

- Source code on QISKit tutorial
  `ibm.biz/qiskit-tutorial`

- ‘Making a quantum computer smile’ on QISKit blog
  `ibm.biz/quantum-emoticon`

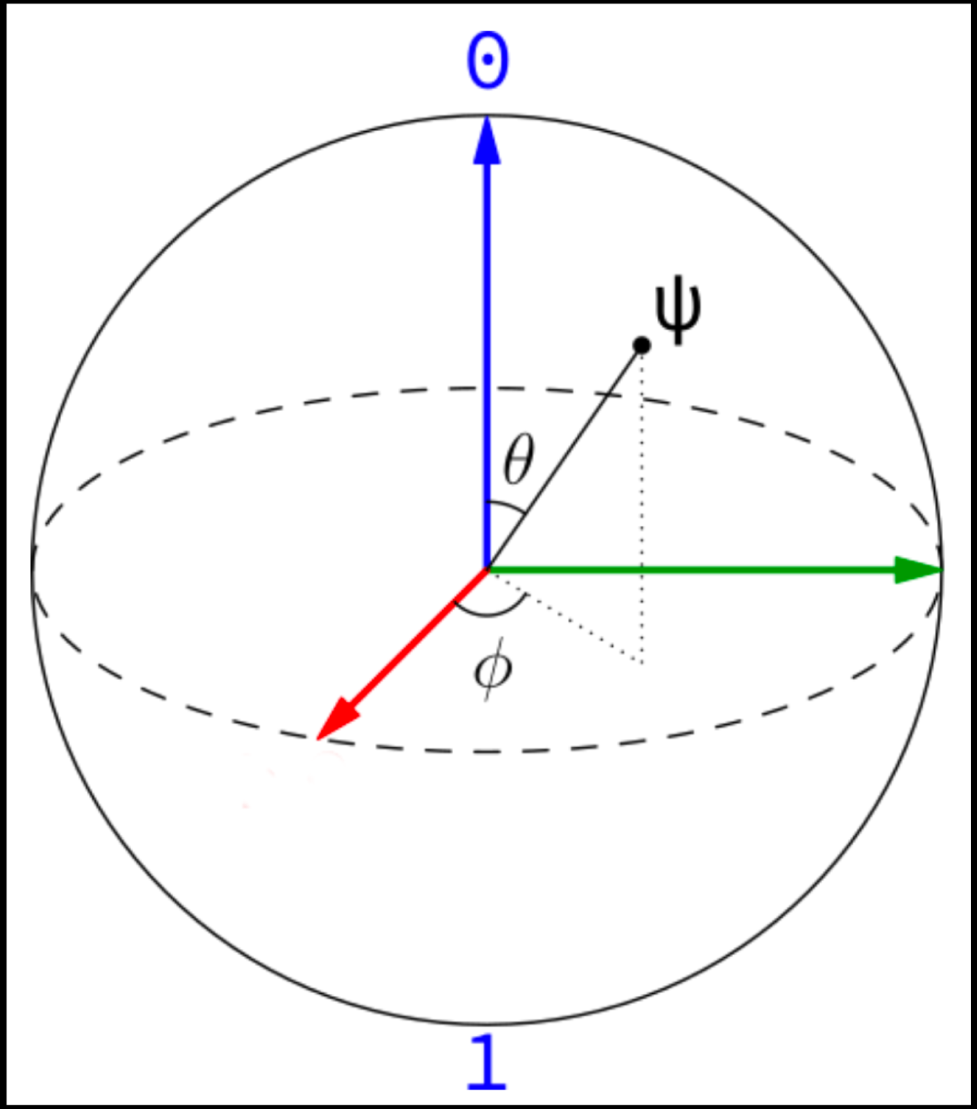
- Gamified guide to creating your own superposition with ‘Hello Quantum’
  `ibm.biz/helloquantum-cil`

- What’s your suggestion for a quantum ‘Hello World’?

Battleships with partial NOT gates

- Another simple application: games
  - Take a simple piece of quantum programming
  - Use it to implement a game mechanic
- For example: qubits allow partial NOT gates

```
qc.y( qr[0]                  # a NOT

qc.ry( np.pi, qr[0] )        # also a NOT

qc.ry( np.pi/2, qr[0] )      # half a NOT

qc.ry( np.pi/3, qr[0] )      # third of a NOT
```



Battleships with partial NOT gates

- Let’s make a variant of Battleships
  - All ships take up single position
  - Different ships need different number of hits to sink

- Classically, we could use a Bool and a NOT

  to implement a single hit ship

```
damage = False        # initially intact

damage = not damage    # attack implemented with
NOT

if damage:
print 'ship destroyed'
```

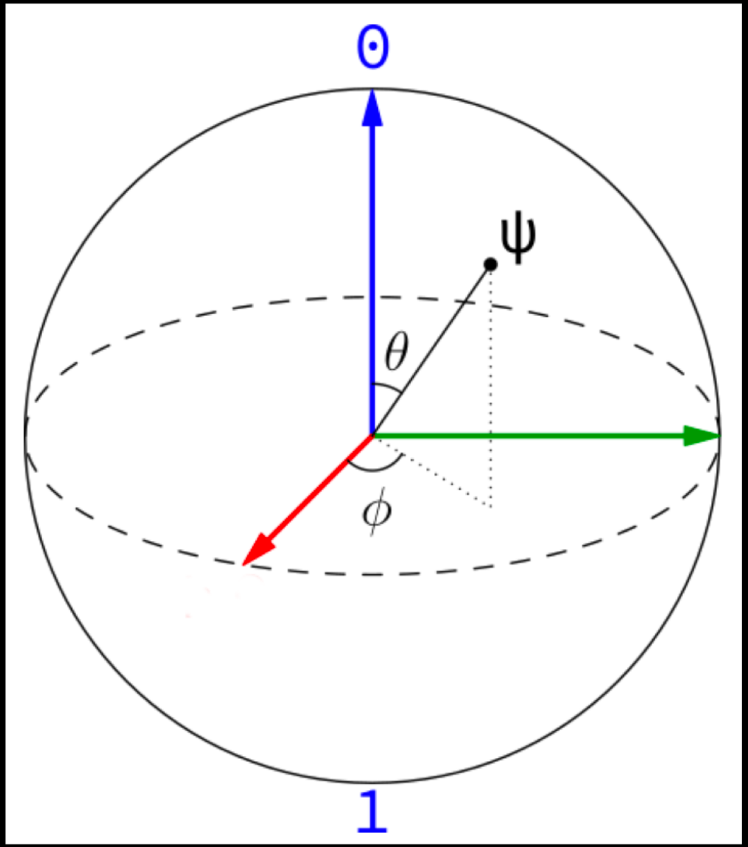
- Multi hit ships would need an Int

```
max_damage = 3
damage = 0     # initially intact

damage += 1    # attack implemented with
addition

if damage==max_damage:
    print 'ship destroyed'
```

- Quantumly, we can do both with a qubit

Battleships with partial NOT gates

- Qubits are the quantum version of a Bool
- The quantum gates X and Y serve as the NOT
- We can also do fractional versions
- These visit superposition states between 0 and 1



- Multi-hit ships can be implemented by dividing up the journey from 0 to 1

```
qr = QuantumRegister(1)            # initially intact

qc.ry( np.pi/max_damage, qr[0] )   # attack with partial NOT

qc.measure(qr,cr)
job = execute(qc,backend)

damage = job.result().get_counts()['1']/shots
if damage==1.0:
    print('ship destroyed')
```

Quantum Battleships

- Now another a bigger piece of quantum programming: measuring a Bell pair

```
qc.h( qr[0] )
qc.cx( qr[0], qr[1]
)
qc.ry( np.pi/4,
qr[1])
qc.h( qr[1] )




qc.measure(qr,cr)
```

85% agreement

```
qc.h( qr[0] )
qc.cx( qr[0], qr[1]
)
qc.ry( np.pi/4,
qr[1])
qc.h( qr[1] )

qc.h(qr[0])


qc.measure(qr,cr)
```

85% agreement

```
qc.h( qr[0] )
qc.cx( qr[0], qr[1]
)
qc.ry( np.pi/4,
qr[1])
qc.h( qr[1] )


qc.h(qr[1])

qc.measure(qr,cr)
```

85% agreement

```
qc.h( qr[0] )
qc.cx( qr[0], qr[1]
)
qc.ry( np.pi/4,
qr[1])
qc.h( qr[1] )

qc.h(qr[0])
qc.h(qr[1])

qc.measure(qr,cr)
```

12% agreement

- Could be used to make a size 2 ship in *Battleships*

Quantum Battleships / Battleships with partial NOT gates

- Source code on QISKit tutorial

  `ibm.biz/qiskit-tutorial`

- ‘How to program a quantum computer’ on QISKit blog

  `ibm.biz/quantum-battleships`

- Gamified guide to creating your own Bell states with ‘Hello Quantum’

  `ibm.biz/helloquantum-cil`

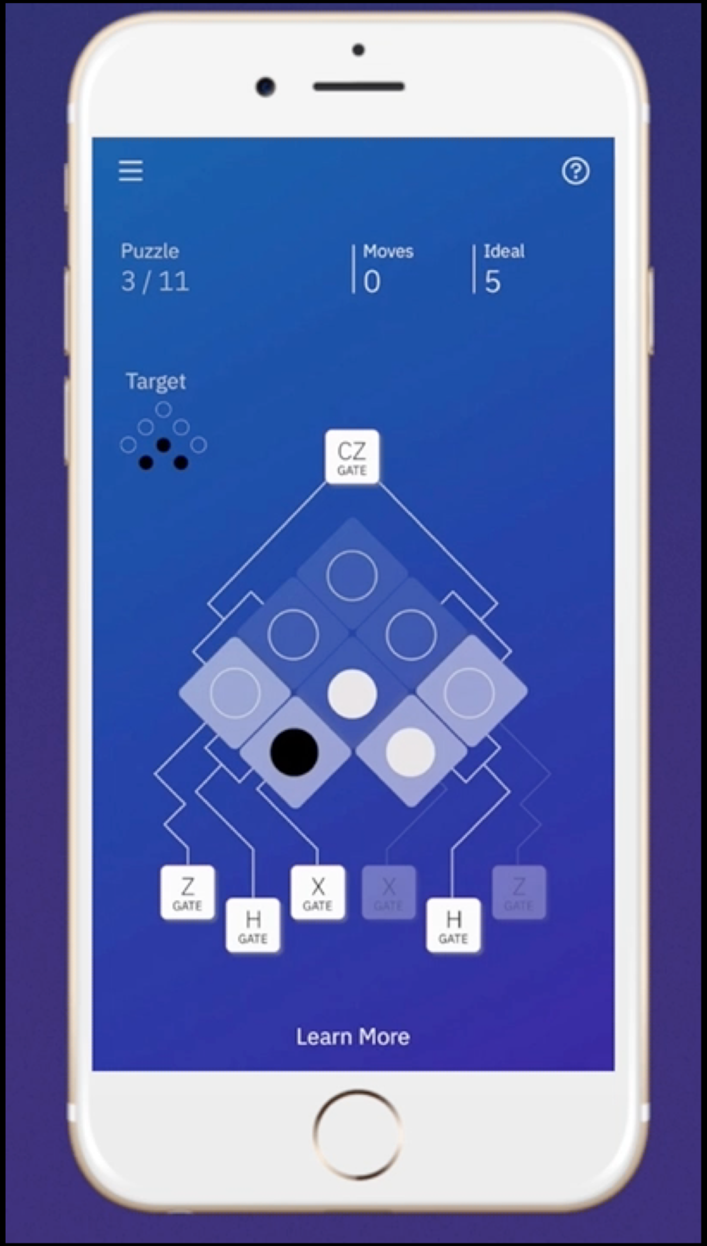
- List of games for quantum computers

  `ibm.biz/qc-games`

‘Hello Quantum’ and more

- IBMers are here to help you get started with quantum
  - ‘Hello Quantum’ for everyone `ibm.biz/helloquantum`
  - ‘Hello Quantum’ for programmers `ibm.biz/helloquantum-cil`
  - QISKit Slack `ibm.biz/join-qiskit-slack`
  - QC Stack Exchange `quantumcomputing.stackexchange.com`
- And to help you get started with QISKit

  `qiskit.org`
  `qiskit.slack.com`



Thanks for listening

Setup your IBM account for tomorrow!

`ibm.biz/qconfig-setup`