Games for Quantum Computers

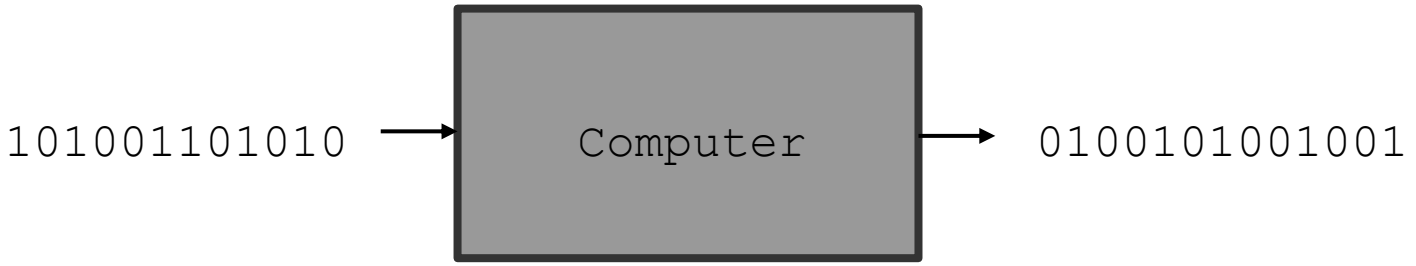
James R. Wootton
IBM Research

Email jwo@zurich.ibm.com
Twitter @decodoku
GitHub/Medium @quantumjim

Part 1

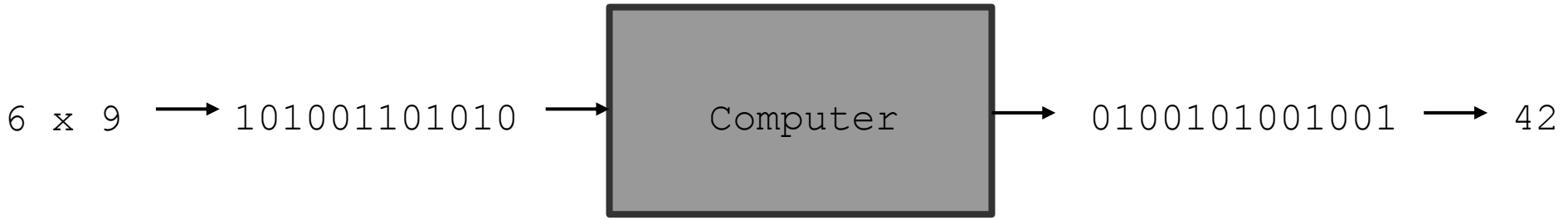
Games for tomorrow’s quantum computers

What is a computer?



- Turns input information into output information

What is a computer?



- Turns input information into output information
- Could be a mathematical problem

What is a computer?

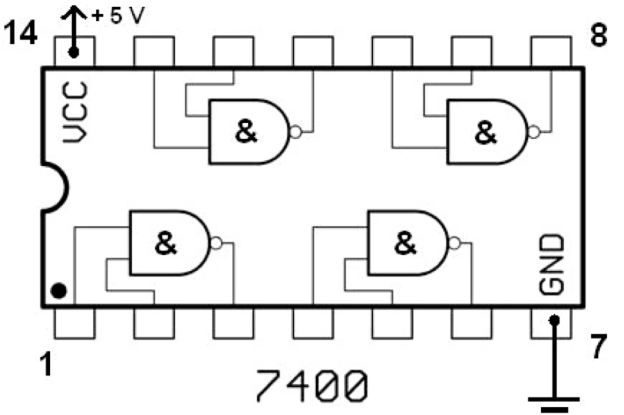
101001101010 → Computer → 0100101001001

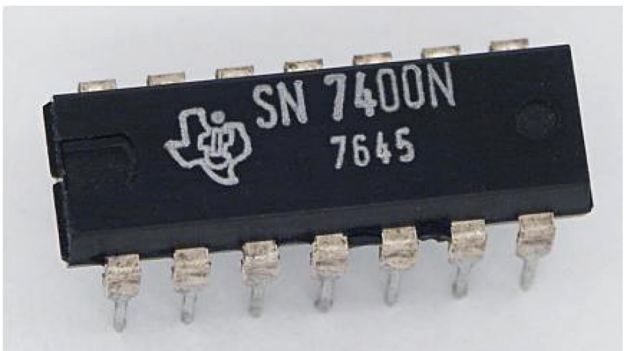
- Turns input information into output information
- Could be something fun!

Controller image: en.wikipedia.org/wiki/File:Nintendo-Super-NES-Controller.jpg
Mario image: flickr.com/photos/amarga/5022153663

The atoms of computation

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

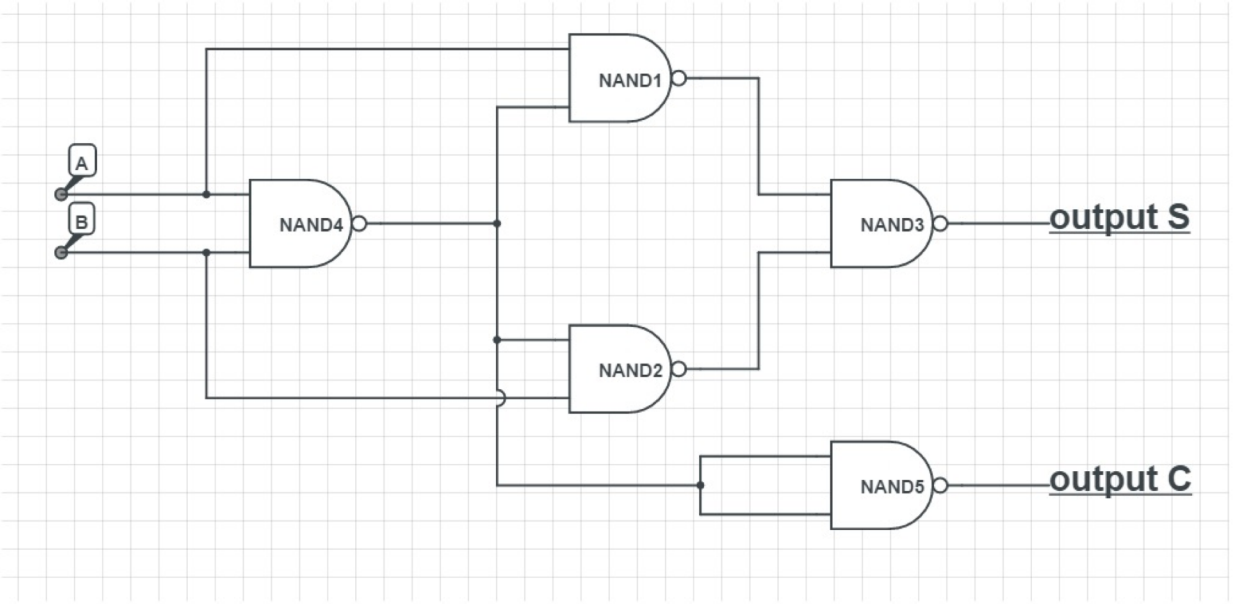




- What is the simplest element of computation?
- Given two bits, is at least one of them `0`?
- This is the NAND of two bits

Chip image: commons.wikimedia.org/wiki/File:7400.jpg
Table from en.wikipedia.org/wiki/NAND_gate

The atoms of computation



- Every other digital computation is a series of many NANDs
- It is the atom of digital computation

Image: commons.wikimedia.org/wiki/File:Half_adder_using_NAND_gates_only.jpg

Computational complexity

$$\begin{array}{r} 5 \\ 12 \\ \hline 17 \end{array}$$

- Different tasks require different amounts of computation
- Adding two *n* digit numbers requires *O(n)* operations

Image: commons.wikimedia.org/wiki/File:AdditionVertical.svg

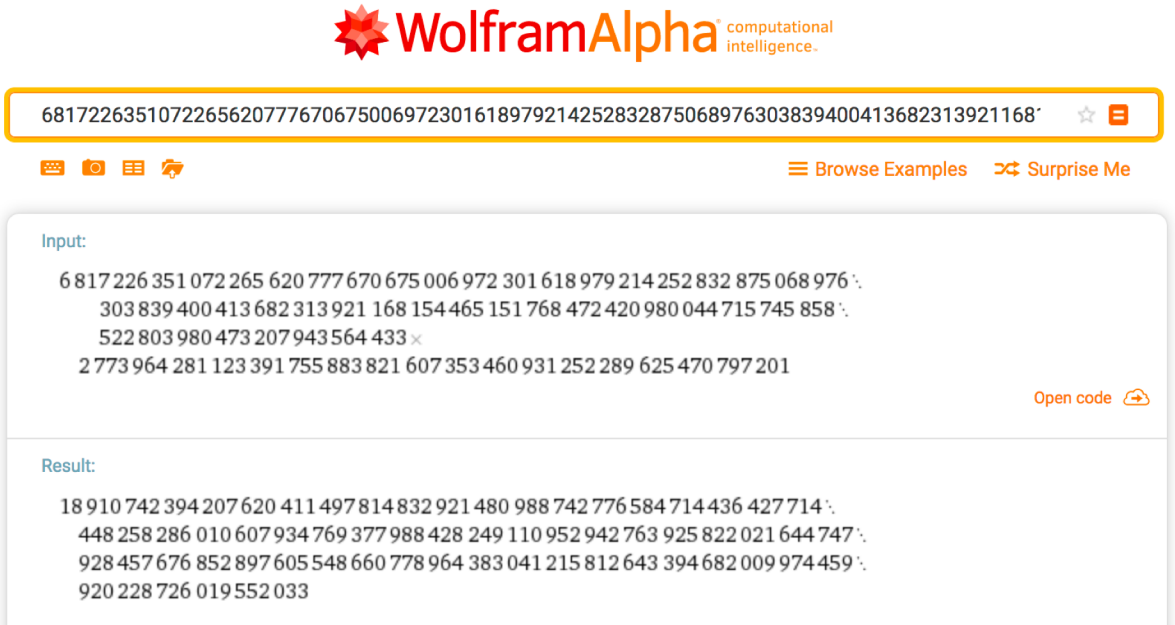
Computational complexity

$$2^{1061} - 1 = 6817226351072265620777670675$$
$$\times\ 27739642811233917558838216073$$

- For factorization, the complexity increases exponentially with *n*
- A 320 digit number took 300 CPU years to factorize!

Paper at eprint.iacr.org/2012/444

Computational complexity



- Multiplying the resulting ~150 digit primes takes less than 1s
- Example of a ‘one-way function’
  - Easy to compute in one direction, almost impossible in the other

wolframalpha.com

Computational complexity in games

- Games do many tasks, and need to do them fast
- Even ‘efficient’ algorithms aren’t efficient enough
  - E.g. Inverse square root in Quake III Area

Image: commons.wikimedia.org/wiki/File:OpenArena-Rocket.jpg

Computational complexity in games

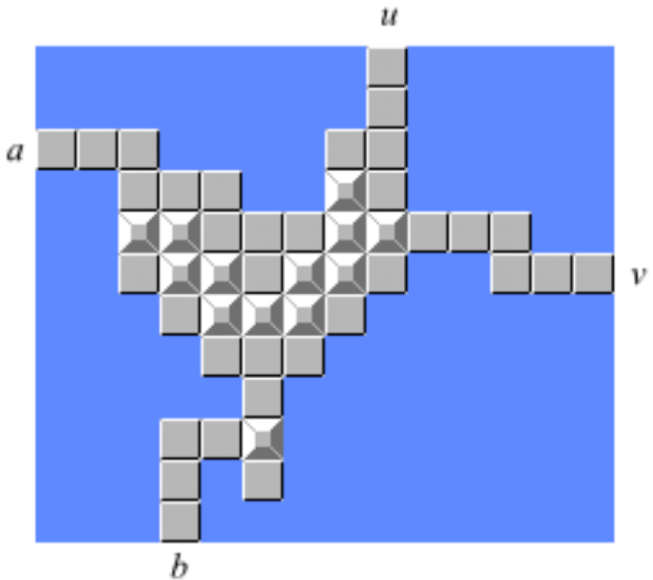


Figure 24: Modified Lock gadget for block pushing in Legend of Zelda

- Complexity of puzzles, levels, etc can also be measured
- Similarly for generating a solvable puzzle, level, etc

Classic Nintendo Games are (Computationally) Hard, Aloupis et al, arxiv.org/abs/1203.1895

Computational complexity in games

Computational complexity [edit]

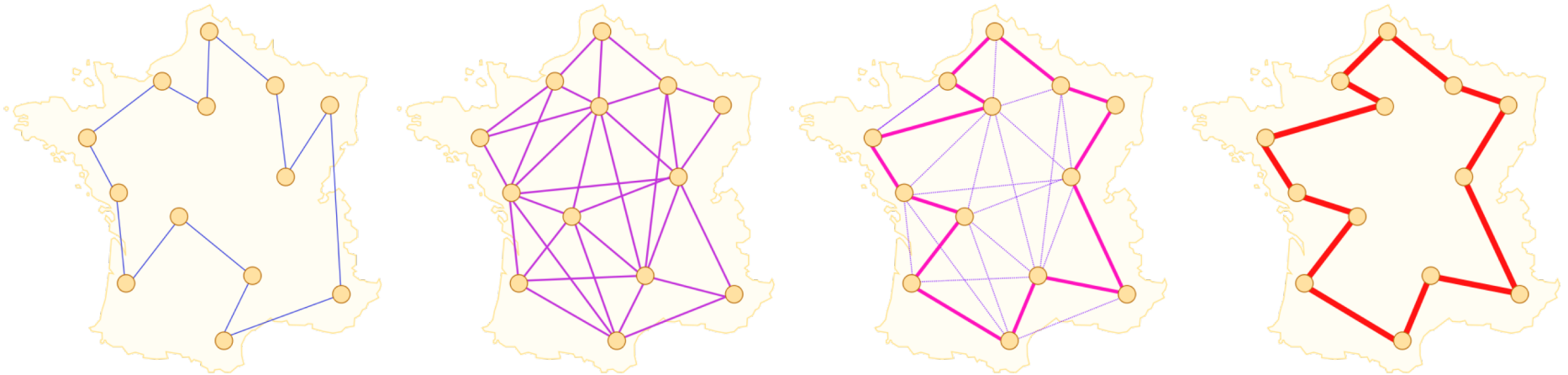
Various complexity results have been proven for certain formulations of the ray tracing problem. In particular, if the decision version of the ray tracing problem is defined as follows[24] – given a light ray's initial position and direction and some fixed point, does the ray eventually reach that point, then the referenced paper proves the following results:

- Ray tracing in 3D optical systems with a finite set of reflective or refractive objects represented by a system of rational quadratic inequalities is undecidable.
- Ray tracing in 3D optical systems with a finite set of refractive objects represented by a system of rational linear inequalities is undecidable.
- Ray tracing in 3D optical systems with a finite set of rectangular reflective or refractive objects is undecidable.
- Ray tracing in 3D optical systems with a finite set of reflective or partially reflective objects represented by a system of linear inequalities, some of which can be irrational is undecidable.
- Ray tracing in 3D optical systems with a finite set of reflective or partially reflective objects represented by a system of rational linear inequalities is PSPACE-hard.
- For any dimension equal to or greater than 2, ray tracing with a finite set of parallel and perpendicular reflective surfaces represented by rational linear inequalities is in PSPACE.

- Graphics is a problem where we care about runtime
- Rendering time depends on surfaces, dimension, etc

en.wikipedia.org/wiki/Ray_tracing_(graphics)#Computational_complexity

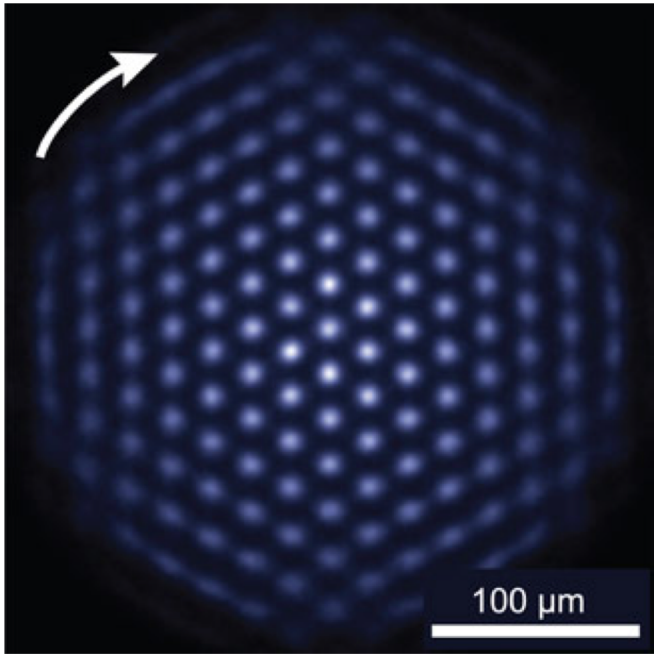
Computational complexity in games



- Computational complexity affects how we can make games
- For example: A ‘traveling salesman’ game
  - How do we score/offer feedback on solutions for large puzzles?

Image: commons.wikimedia.org/wiki/File:Aco_TSP.svg

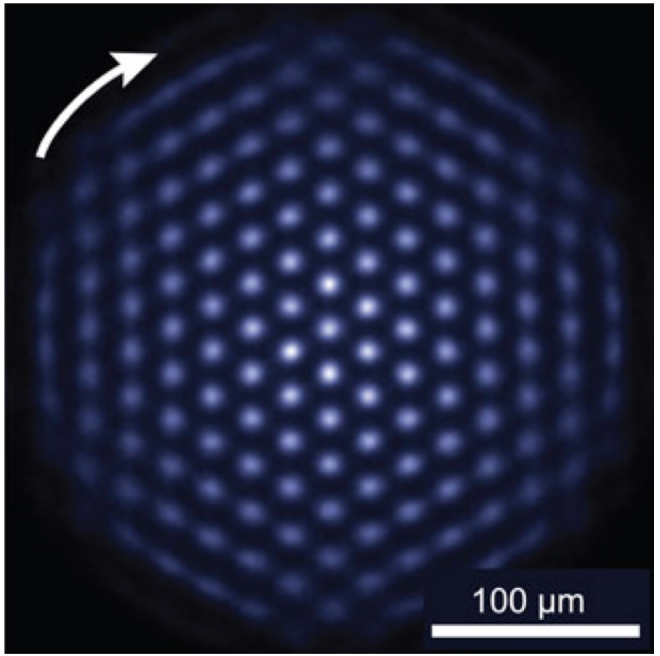
Quantum simulation



- Another hard problem is simulating quantum physics
- For *n* interacting quantum objects, complexity is exponential with *n*

Image: commons.wikimedia.org/wiki/File:Quantum_Simulator_Crystal.jpg

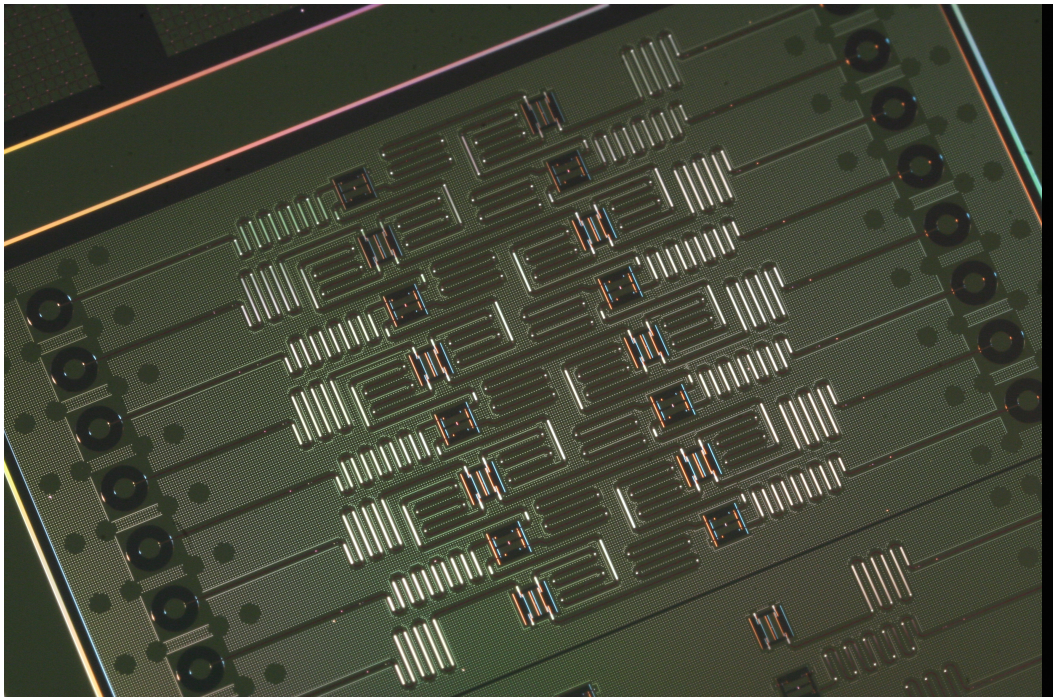
Quantum simulation

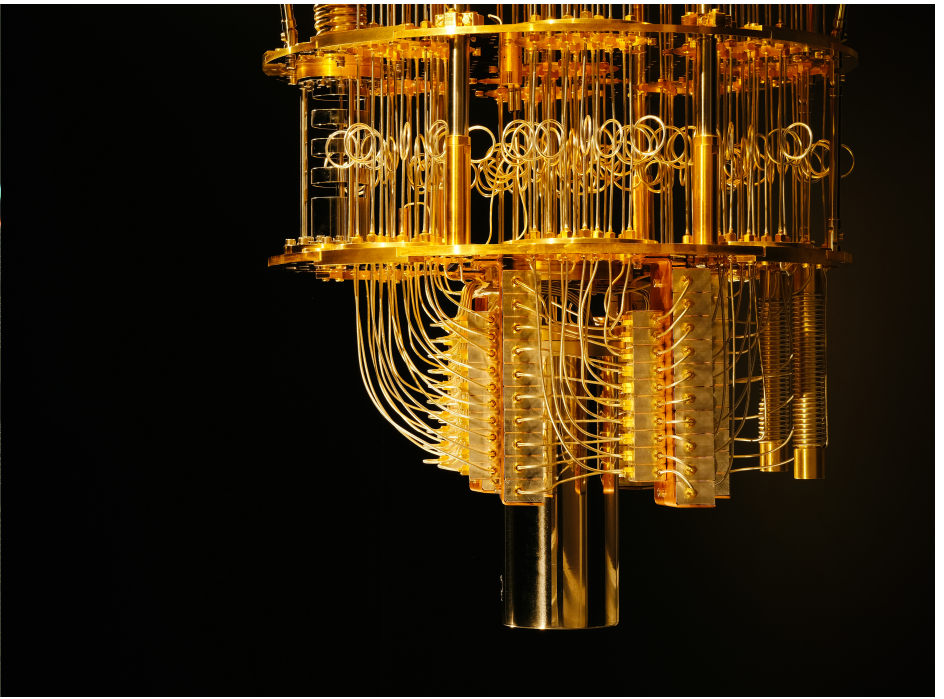


- But these quantum objects ‘simulate themselves’ without problem
- With many well controlled, reprogrammable quantum objects, we can we simulate efficiently

Image: commons.wikimedia.org/wiki/File:Quantum_Simulator_Crystal.jpg

Quantum simulation

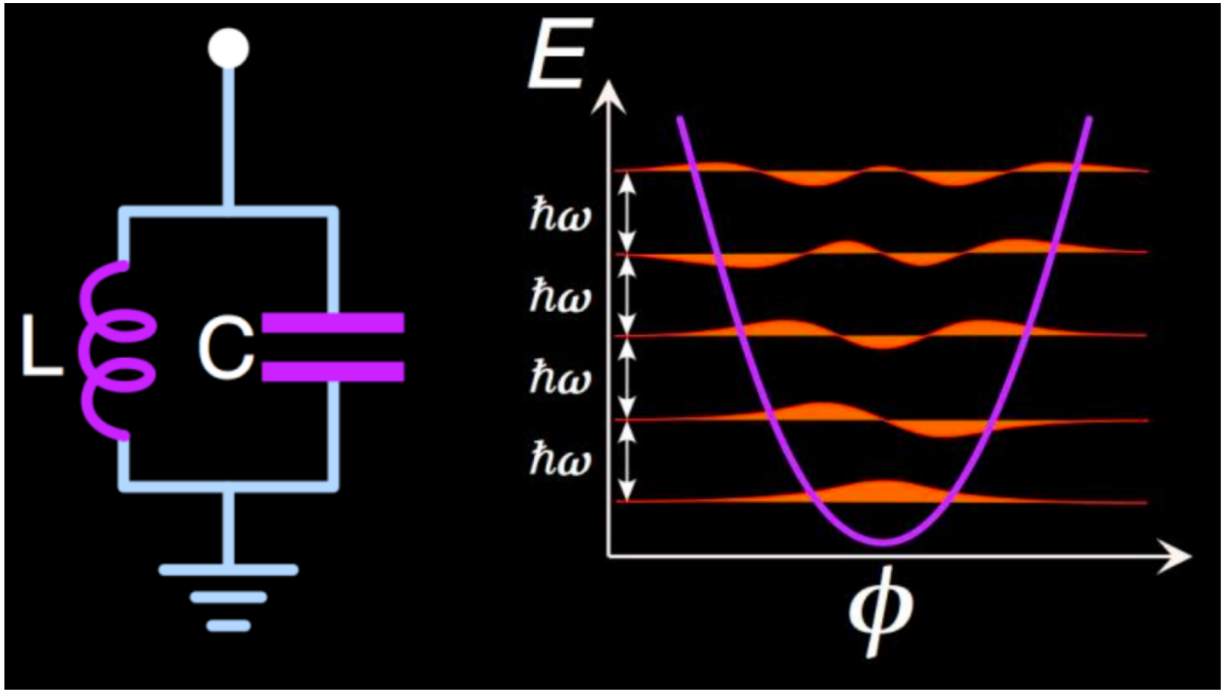




- So that’s what we’ve built

Image: IBM Research

“How does quantum computer store information?”



- Lowest two energy levels of anharmonic oscillator use to store bit states

Image: IBM Research

A new computing paradigm

```
24  def input_state(circ, q, n):
25      """n-qubit input state for QFT that produces output 1."""
26      for j in range(n):
27          circ.h(q[j])
28          circ.u1(math.pi/float(2**(j)), q[j]).inverse()
29
30
31  def qft(circ, q, n):
32      """n-qubit QFT on q in circ."""
33      for j in range(n):
34          for k in range(j):
35              circ.cu1(math.pi/float(2**(j-k)), q[j], q[k])
36          circ.h(q[j])
```

- But quantum computers can do more than just simulate
- Many problems can be solved more efficiently
  - factoring/search/optimization/random number generation

From github.com/Qiskit

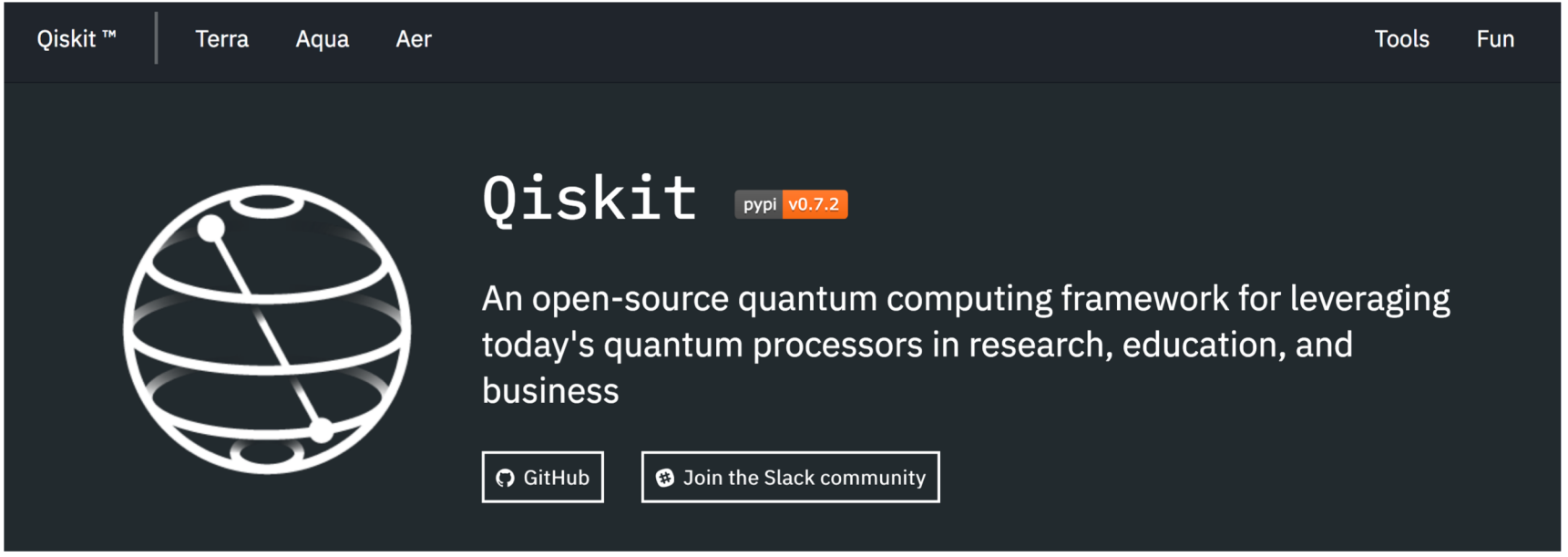
“…will we be able to instruct a quantum computer by writing code as if we were doing that with regular computer, and make a game for example?”

```
q = QuantumRegister(1) # initialize a register with a single qubit
c = ClassicalRegister(1) # initialize a register with a normal bit
qc = QuantumCircuit(q, c) # create and empty quantum program

qc.u3(math.pi,0,0, q[0]) # apply a NOT to the qubit

qc.measure( q[0], c[0] ) # measure the qubit
```

- Yes!
- See “How to program a quantum computer” on Medium
- ‘Battleships’ made using Qiskit

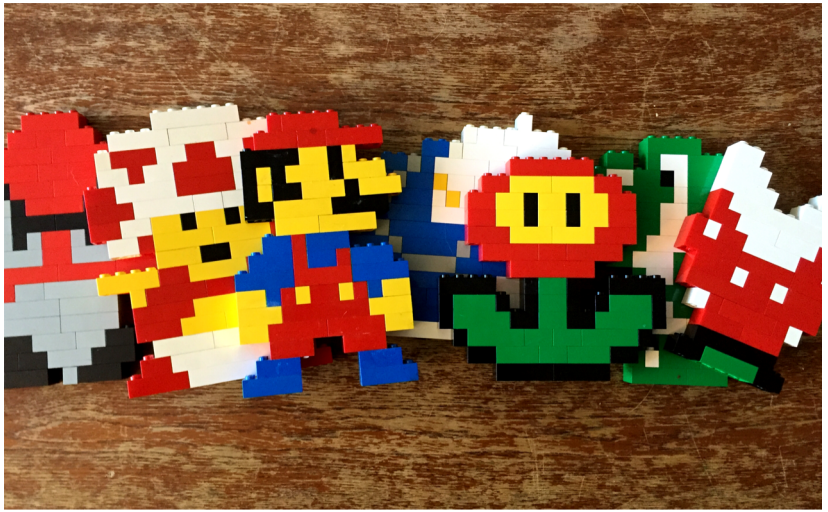
From medium.com/@quantumjim

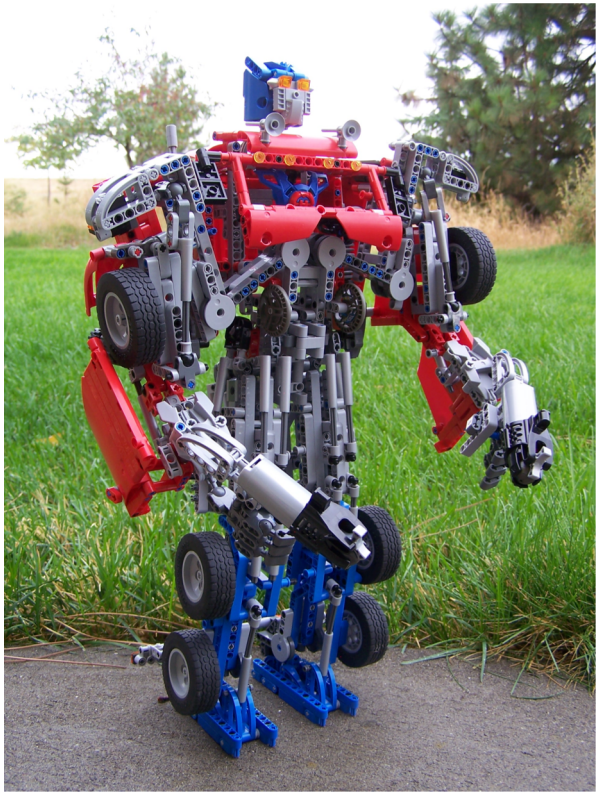
“This new type of computing should probably require some new type of interfaces and developing methods for software and applications, how do you see those developing in the coming years?”



- We are building a software framework at IBM
- Python based
- Low level control and abstract level tools

Splitting the atom

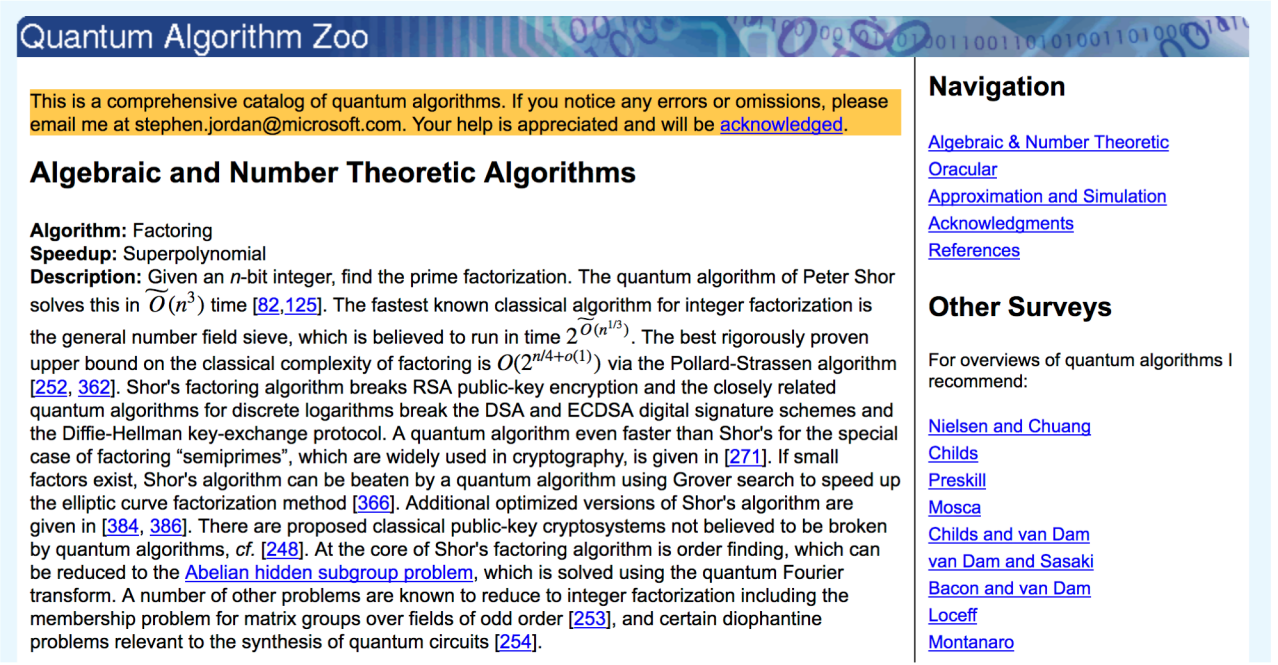




- You can do NANDs in a quantum computer too
- You can also do 1/2 a NAND, 1/3 of a NAND, or 3.14% of a NAND
- With more adaptable building blocks, we get more effective programs

Mario: flickr.com/photos/tomhenrich/21693149844
Optimus: flickr.com/photos/42988571@N08/6248041140

Quantum Algorithms



Quantum Algorithm Zoo

This is a comprehensive catalog of quantum algorithms. If you notice any errors or omissions, please email me at stephen.jordan@microsoft.com. Your help is appreciated and will be acknowledged.

Algebraic and Number Theoretic Algorithms

Algorithm: Factoring
Speedup: Superpolynomial
Description: Given an n-bit integer, find the prime factorization. The quantum algorithm of Peter Shor solves this in $\widetilde{O}(n^3)$ time [82,125]. The fastest known classical algorithm for integer factorization is the general number field sieve, which is believed to run in time $2^{\widetilde{O}(n^{1/3})}$. The best rigorously proven upper bound on the classical complexity of factoring is $O(2^{n/4+o(1)})$ via the Pollard-Strassen algorithm [252, 362]. Shor's factoring algorithm breaks RSA public-key encryption and the closely related quantum algorithms for discrete logarithms break the DSA and ECDSA digital signature schemes and the Diffie-Hellman key-exchange protocol. A quantum algorithm even faster than Shor's for the special case of factoring "semiprimes", which are widely used in cryptography, is given in [271]. If small factors exist, Shor's algorithm can be beaten by a quantum algorithm using Grover search to speed up the elliptic curve factorization method [366]. Additional optimized versions of Shor's algorithm are given in [384, 386]. There are proposed classical public-key cryptosystems not believed to be broken by quantum algorithms, *cf.* [248]. At the core of Shor's factoring algorithm is order finding, which can be reduced to the Abelian hidden subgroup problem, which is solved using the quantum Fourier transform. A number of other problems are known to reduce to integer factorization including the membership problem for matrix groups over fields of odd order [253], and certain diophantine problems relevant to the synthesis of quantum circuits [254].

Navigation

Algebraic & Number Theoretic
Oracular
Approximation and Simulation
Acknowledgments
References

Other Surveys

For overviews of quantum algorithms I recommend:

Nielsen and Chuang
Childs
Preskill
Mosca
Childs and van Dam
van Dam and Sasaki
Bacon and van Dam
Loceff
Montanaro

- We have many examples of algorithms with complexity reduction
- Previous focus has been on algorithms with good speedup proofs
- Focus is currently switching to practicality

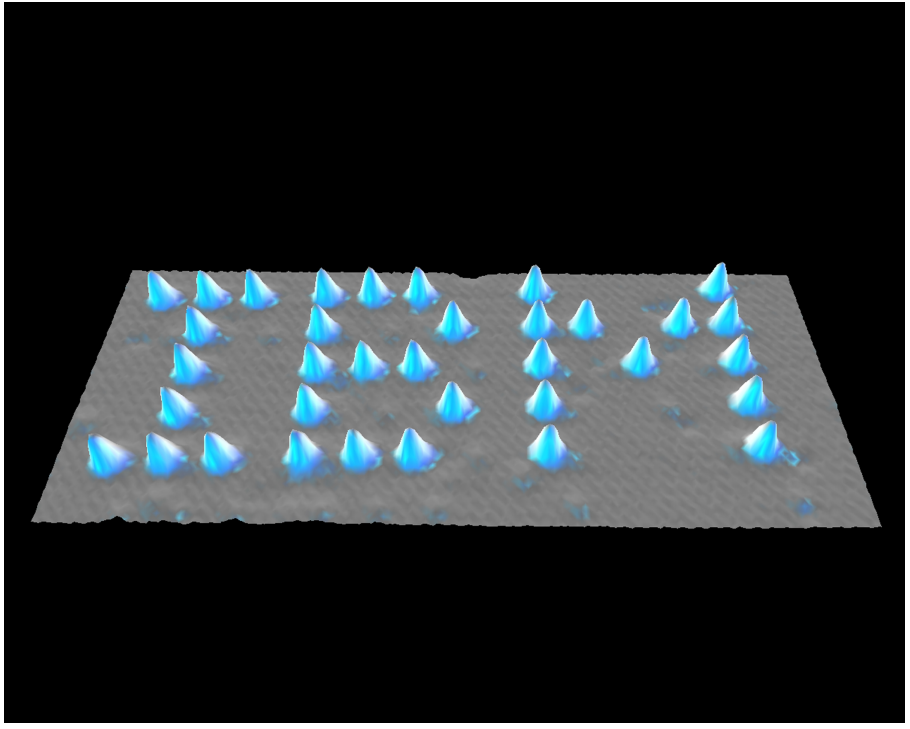
quantumalgorithmzoo.org

Quantum physics engines

- Physics lies at the heart of many games, from Portal to Angry Birds
- Quantum computers could allow for quantum physics games

Screenshot from ‘Quantum Cats’, see quantumcats.ca

Quantum graphics



- Searching databases is a significant part of rendering graphics
- Quantum search algorithms reduce the computational complexity

Image: IBM Research
See boxc.at and papers by Simona Caraiman

How to use a quantum computer

- Simulation and graphics require QC in the game loop
- Would be practical if your PC could have a ‘quantum card’
- Will quantum computer hardware will be available like this?

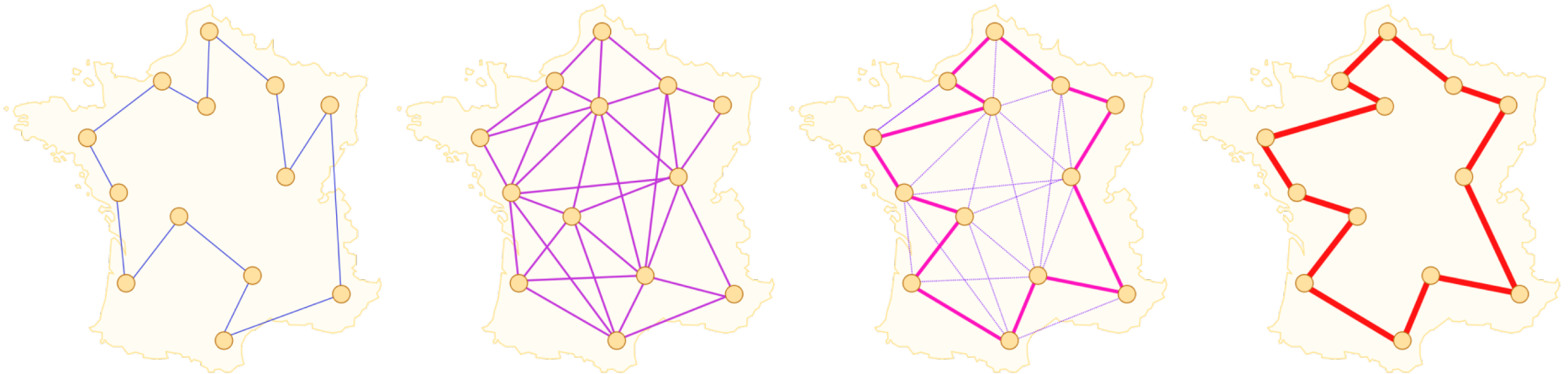
Image: IBM Research

How to use a quantum computer

- Noise reduction techniques are too big and expensive to take home
- Current model is of a cloud based service
- While demand > supply, there will also be a queue to contend with
- Better to take QC out of the game loop for now

Image: IBM Research

QC for optimization



- Puzzles based on computationally hard problems can be solved
- Full feedback could be given for player solutions
- QC used during the design stage

Image: commons.wikimedia.org/wiki/File:Aco_TSP.svg

QC for procedural generation

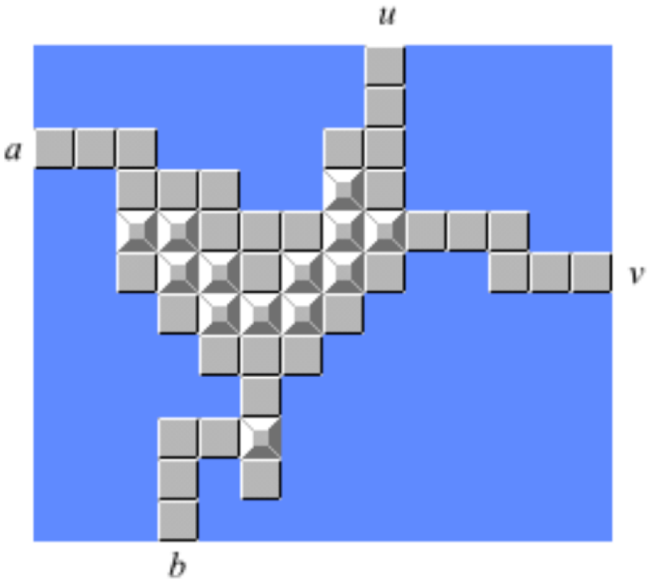


Figure 24: Modified Lock gadget for block pushing in Legend of Zelda

- Easier to determine whether a generated level is beatable
- Allows more freedom in how they are generated
- QC used during ‘loading screen’

Classic Nintendo Games are (Computationally) Hard, Aloupis et al, arxiv.org/abs/1203.1895

QC for procedural generation

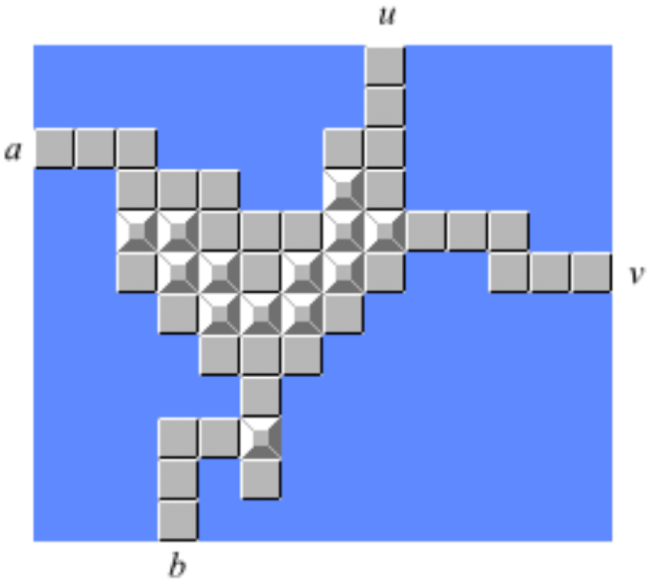


Figure 24: Modified Lock gadget for block pushing in Legend of Zelda

- QC can produce truly random numbers easily
- Removes need to rely on pseudorandom number generators
- QC used during ‘loading screen’

Classic Nintendo Games are (Computationally) Hard, Aloupis et al, arxiv.org/abs/1203.1895

QC for AI

- QC can be used to enhance AI
- QC used during wait for turn

Image: IBM Research

“If a quantum computer could be placed inside a robots or an androids brain, would it be possible to make the results of those kind of calculations resemble true emotions or feelings?”

- For a bowl of soup, or piece of cheese: In theory, it’s possible
- A brain should be no more complex physically, but it’s more complex philosophically

Image: commons.wikimedia.org/wiki/File:Blue_Stilton_02.jpg

Overview

- **Long-term uses for QC (inside game loop)**

- Quantum physics simulation
- Graphics rendering
- ...

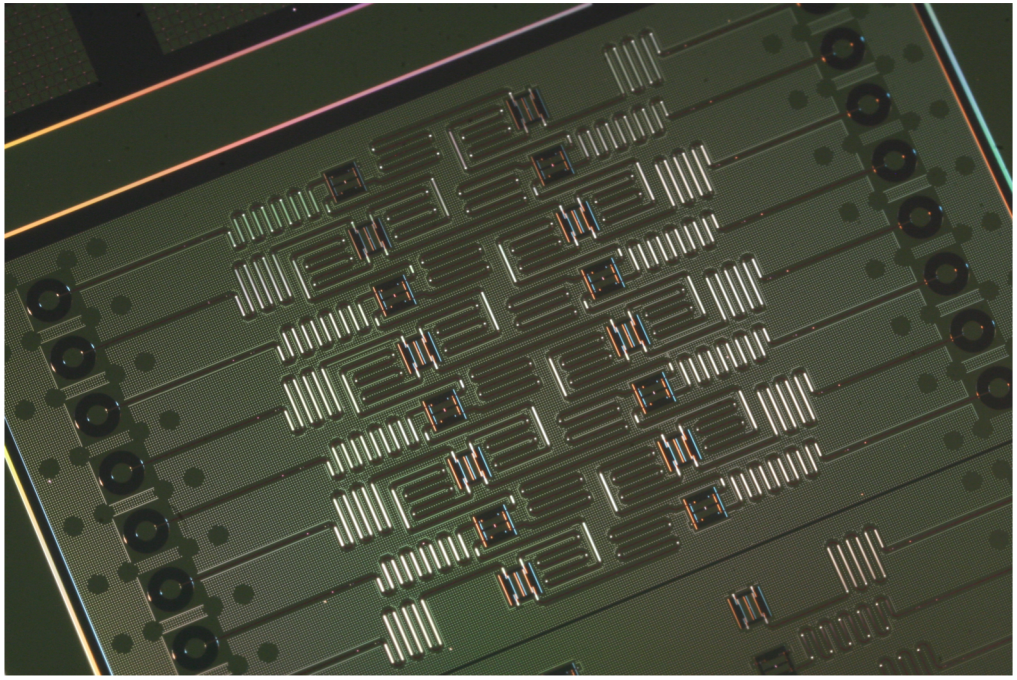
- **Medium- and long-term uses for QC (outside game loop)**

- Optimization based tasks
- Procedural generation
- AI
- Developing game mechanics
- ...

Part 2

Games for today’s quantum computers

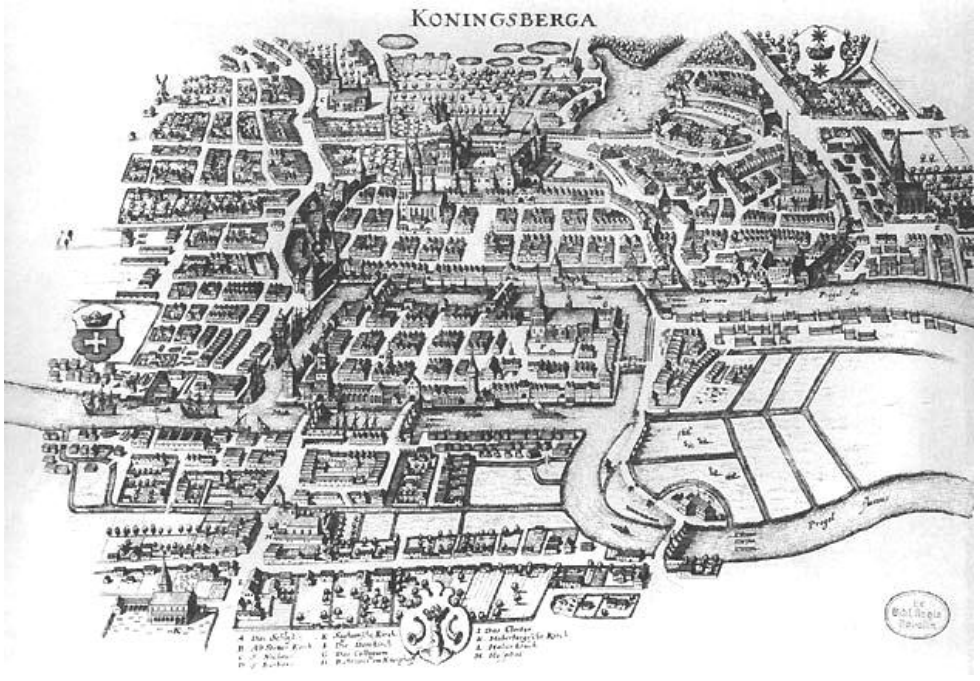
Current devices are small



- The basic quantum object is the *qubit*
- 50-100 needed to do something that digital computers cannot
- No more than 14 on devices currently usable by the public

Image: IBM Research

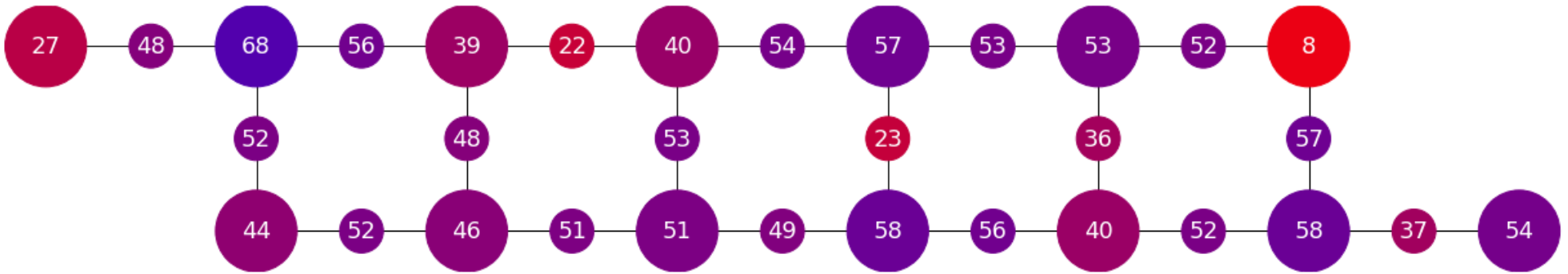
“Aside from speeding up processes, are there any bridges we can now cross using Quantum computers for games?”



- There are many bridges, but most aren’t on the map
- The fun is in finding them!

commons.wikimedia.org/wiki/File:Image-Koenigsberg,_Map_by_Merian-Erben_1652.jpg

Current devices are noisy



- Full error correction won’t be viable in the short term
- Quantum programs will contain spurious effects
- Program design and processing of readout need to mitigate this

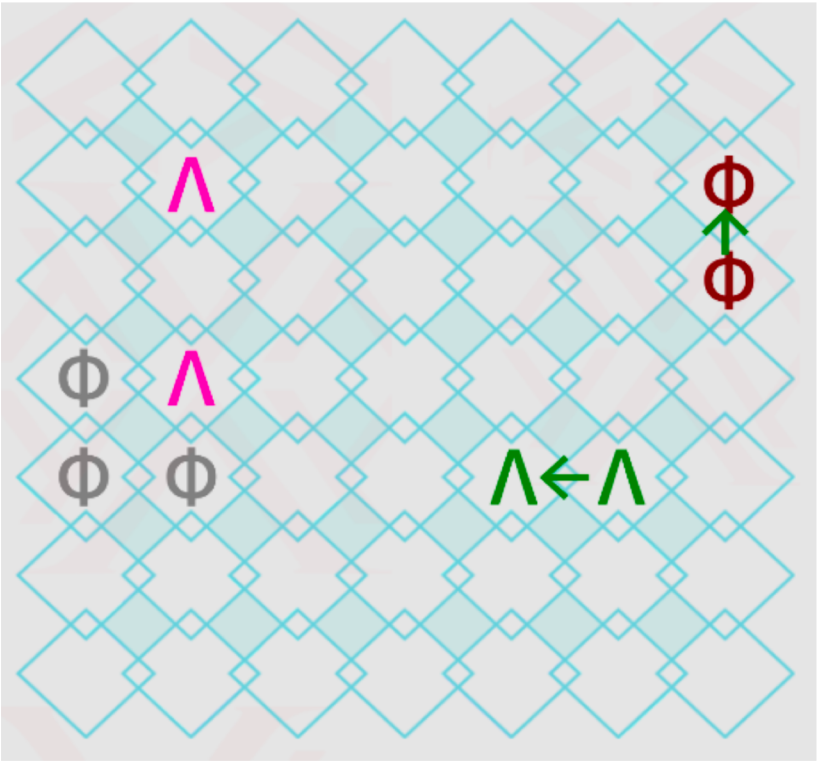
Image: medium.com/qiskit

Quantum games are serious

- Near-term quantum games will primarily be serious games
- This parallels the serious games era of early gaming history

Image: www.ibm.com/ibm/history/ibm100/us/en/icons/ibm700series/impacts

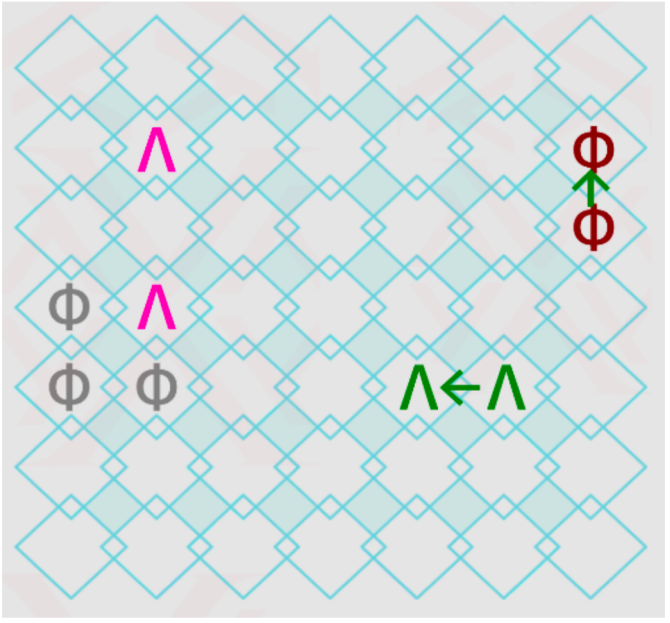
Quantum games for science



- Part of citizen science projects
- Educational tool

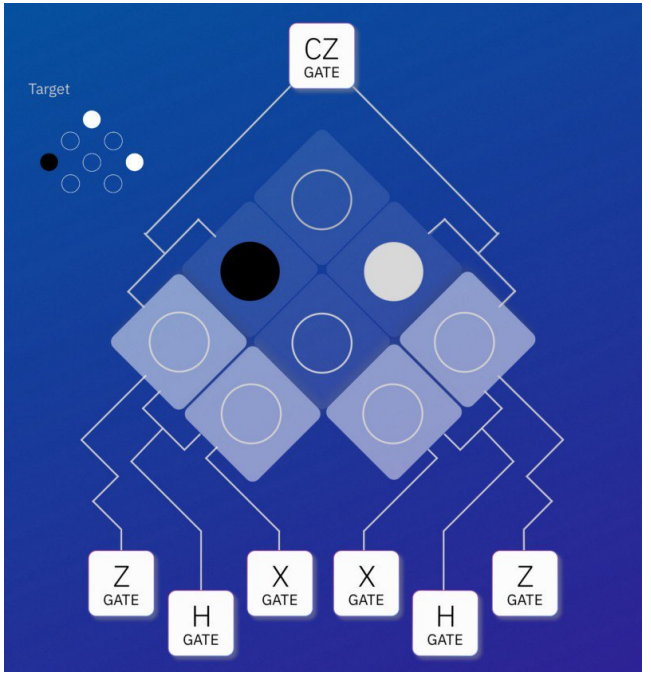
decodoku.itch.io/decodoku

“Could simulation games be improved so much that they would reflect reality so closely that you could actually make scientific discoveries just by playing?”



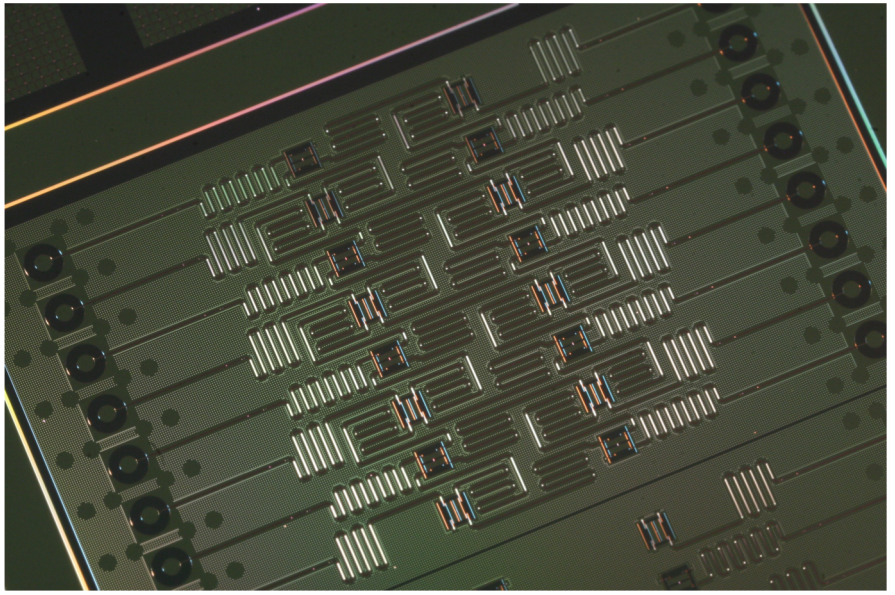
- We can already do this with easy to simulate things (Decodoku, Quantum Moves,…)
- We should have even more opportunities with QC

“What kind of challenges can there be for artists in game development with quantum computers?



- There’s a lot going on inside a quantum computer
- A challenge for gamified training, or games based on the physics, is visualization of quantum states

“How far away are we from the point where the quantum computing provides us the means of tactile interaction with a photon, or a beam of light?”



- You can do it now!
- Direct access to quantum oscillators via IBMQ
- Access to photons via University of Bristol’s ‘Quantum in the Cloud’

QC for inspiration

- Lots of sci-fi is based on (often uninformed) ideas from quantum
- QC could prototype and test new ideas for game mechanics
- QC used during design

commons.wikimedia.org/wiki/File:Ant-Man.jpg

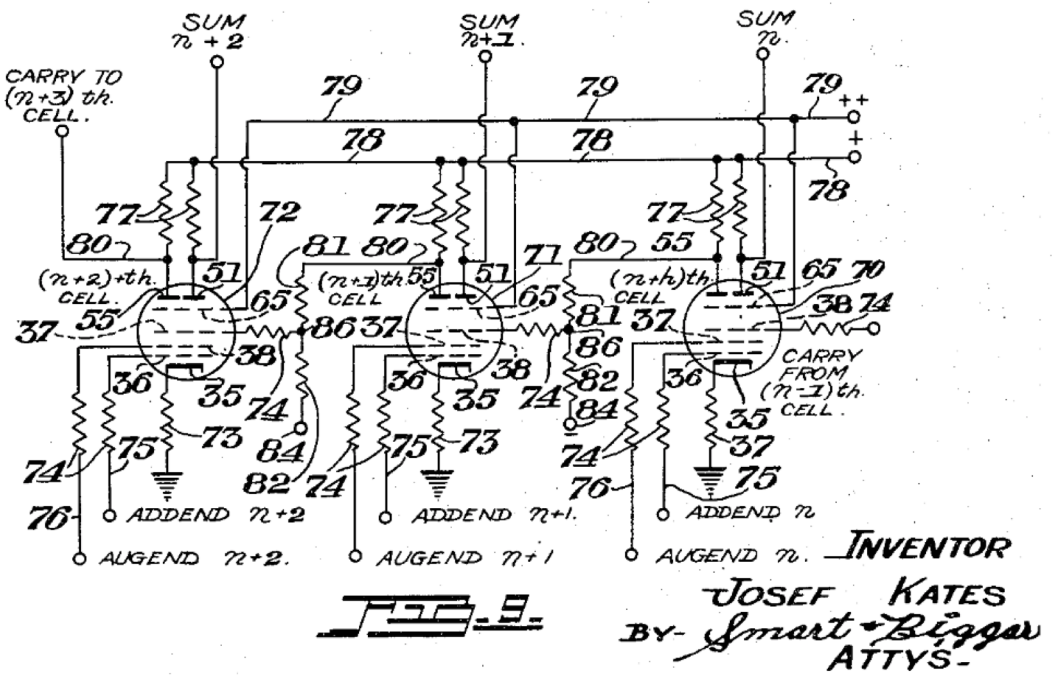
QC for genuine sci-fi experience

- Example: universe splitter
- Uses notion of parallel universes from ‘many worlds interpretation’
- Makes a coin-flip app into an interesting experience

- Games for player experience
  - Genuine interaction with popular sci-fi topic

Image: cheapuniverses.com/universesplitter

Bertie the Brain - 1950



- Tic-Tac-Toe with vacuum tubes and light bulbs
- Built to showcase a new vacuum tube design

Image: en.wikipedia.org/wiki/File:Additron_Tube_schematic.png

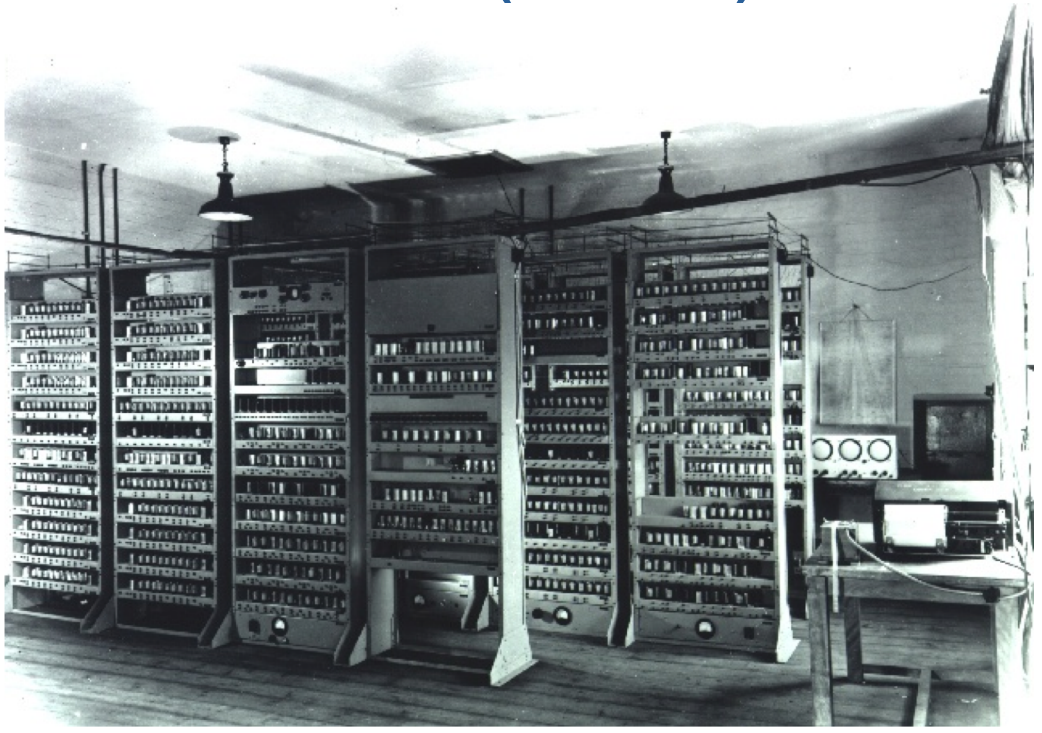
Nimrod - 1951

> “ It may appear that, in trying to make machines play games, we are wasting our time. This is not true as the theory of games is extremely complex and a machine that can play a complex game can also be programmed to carry out very complex practical problems. ”
>
> Pamphlet accompanying the Nimrod sold to Festival of Britain attendees.[5]

- Nim with vacuum tubes and light bulbs
- Built to “illustrate the algorithm and programming principles involved” (designer John Bennett)

- For source of first quote, see en.wikipedia.org/wiki/Nimrod_(computer)
- For source of second quote, see www.goodeveca.net/nimrod/bennett.html

Noughts and Crosses (OXO) - 1952



- Tic-Tac-Toe with vacuum tubes and cathode rays
- Built for research into human-computer interaction

- Image: commons.wikimedia.org/wiki/File:EDSAC_(19).jpg

IBM Checkers – 1950s

- Checkers with AI on IBM 700 series
- Built to demonstrate the power of computers (and raise IBM's stock price)

Image: www.ibm.com/ibm/history/ibm100/us/en/icons/ibm700series/impacts

System Training Project – 1950's

corporation, System Development Corporation (SDC). While the STP appears to be the first military simulation run primarily by a computer, it cannot really be classified as a complete computer game, as the 701 merely plays the film and traces flight paths over a two-hour training session. A human team was apparently still required to actually administer the exercise and interpret the results.

- Military simulation (running on IBM hardware)
- Built for training
- Implemented game mechanic, but not game loop

Quote: videogamehistorian.wordpress.com/2014/01/22/the-priesthood-at-play-computer-games-in-the-1950s

Spacewar! - 1962

- First novel computer game, running on PDP-1
- Built to
  - Test out a new device (and later, new installations)
  - Showcase its capabilities
  - Be fun!

Image: flickr.com/photos/35034362831@N01/494431001

Deep Blue/Watson/AlphaGo - 1996-2016

- AI playing games to demonstrate its power
- Won against human champions of chess/Jeopardy!/go

commons.wikimedia.org/wiki/File:Watson_Jeopardy_demo.jpg

Quantum Battleships - 2017

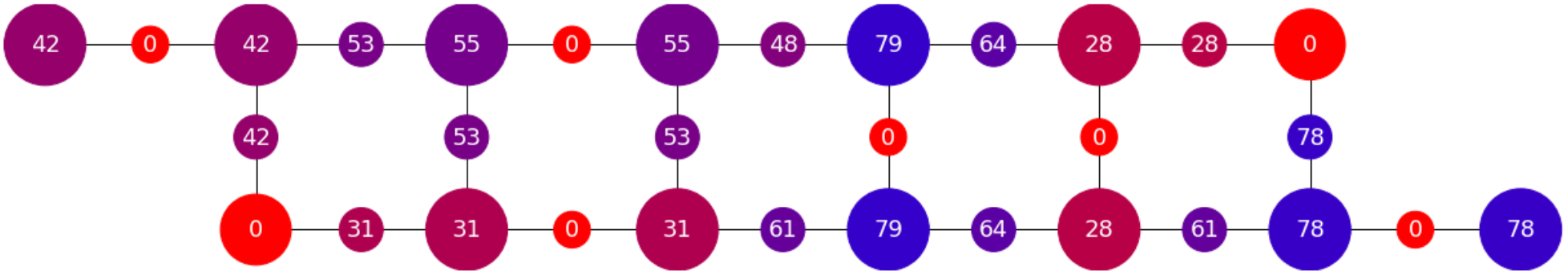
```
q = QuantumRegister(1) # initialize a register with a single qubit
c = ClassicalRegister(1) # initialize a register with a normal bit
qc = QuantumCircuit(q, c) # create and empty quantum program

qc.u3(math.pi,0,0, q[0]) # apply a NOT to the qubit

qc.measure( q[0], c[0] ) # measure the qubit
```

- Japanese-style Battleships built with partial NOT gates
- Forms the heart of a simple quantum programming tutorial

- Games to teach programmers:
  - Implement game mechanic with quantum operations

From medium.com/qiskit

Quantum Awesomeness - 2017



- Simple puzzle game of matching numbers
- Game grid determined by instruction set
- Difficulty determined by noise

- Games to teach players:
  - Have gameplay influenced by device specs

Image: medium.com/qiskit

Conclusions

- Quantum computers will accelerate certain processes
- Games use many kinds of process, and value speed, so quantum computers will be very useful
- In the near-term, best to keep QC out of the game loop
- In the short-term, games are more useful for QC than QC is for games
- Things to try:
    - Explore new game mechanics inspired by QC
    - Make games that explore the nature of quantum noise
    - Try out new methods for proc. gen.

Thanks for your attention

- History of games for quantum computers

  ibm.biz/qc-games

- How to program quantum Battleships

  ibm.biz/quantum-battleships

- A gamified intro to basic QC concepts

  ibm.biz/hello-qiskit

- Intro to QC for developers

  learnqiskit.gitbook.io

Email jwo@zurich.ibm.com
Twitter @decodoku
GitHub/Medium @quantumjim