



DEVOXX FRANCE 2025

Code Quantum : Le développeur post-quantique



VOTRE_SPEAKER



@jerome-leonard-dev.bsky.social



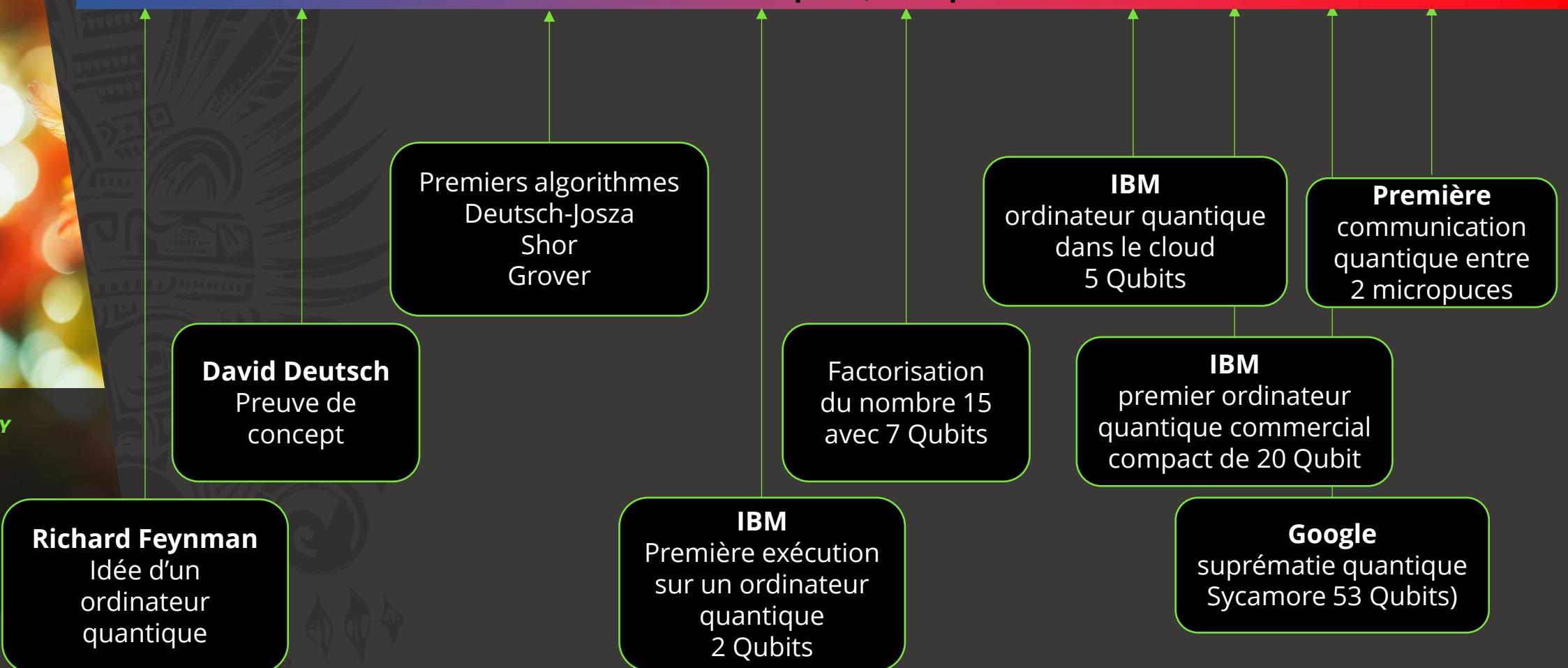
@jerome-leonard



D'OÙ ÇA SORT ET ON
EN EST OÙ ?

1970 1985 1992-1996 1998 2001 2016 2019 2023 2025

Informatique Quantique



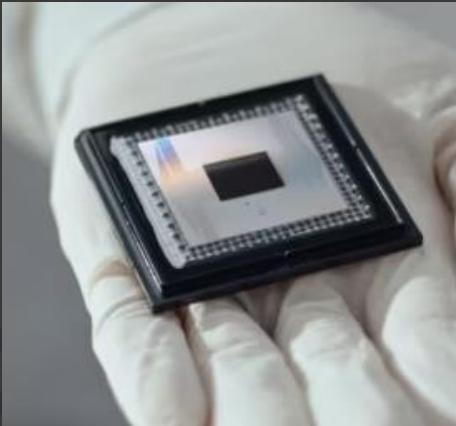
/HISTORY



QUAND EN AURAIS-
JE UN ?



/WHEN



Willow

- Système de refroidissement spécifiques
- Coûts de maintenance très élevés
- Très sensible à l'environnement
- A partir de 10 000 000\$
- De 100 à 200 Qubits



Bell-1

- Mars 2025
- Système de refroidissement intégré
- Consomme 1600 W
- Pèse 200Kg
- De l'ordre de 1 000 000\$
- 6 Qubit



Gemini mini



Gemini



Triangulum

- Sortie en 2021
- Fonctionne à température ambiante
- Consomme 60 W
- Pèse 15Kg
- De 10 000\$ à 65 000\$
- 2 à 3 Qubit



À QUOI ÇA SERT ?



Bases de données (Recherche)



Physiques (Simulation)



IA (Machine learning)



Logistique (Optimisation)



Finance (Modélisation)



Cryptographie

QUANTUM FOR REAL-WORLD IMPACT

PHASE | Active



XPRIZE
QUANTUM
APPLICATIONS

Presenting Partner: gesda

Google
Quantum AI

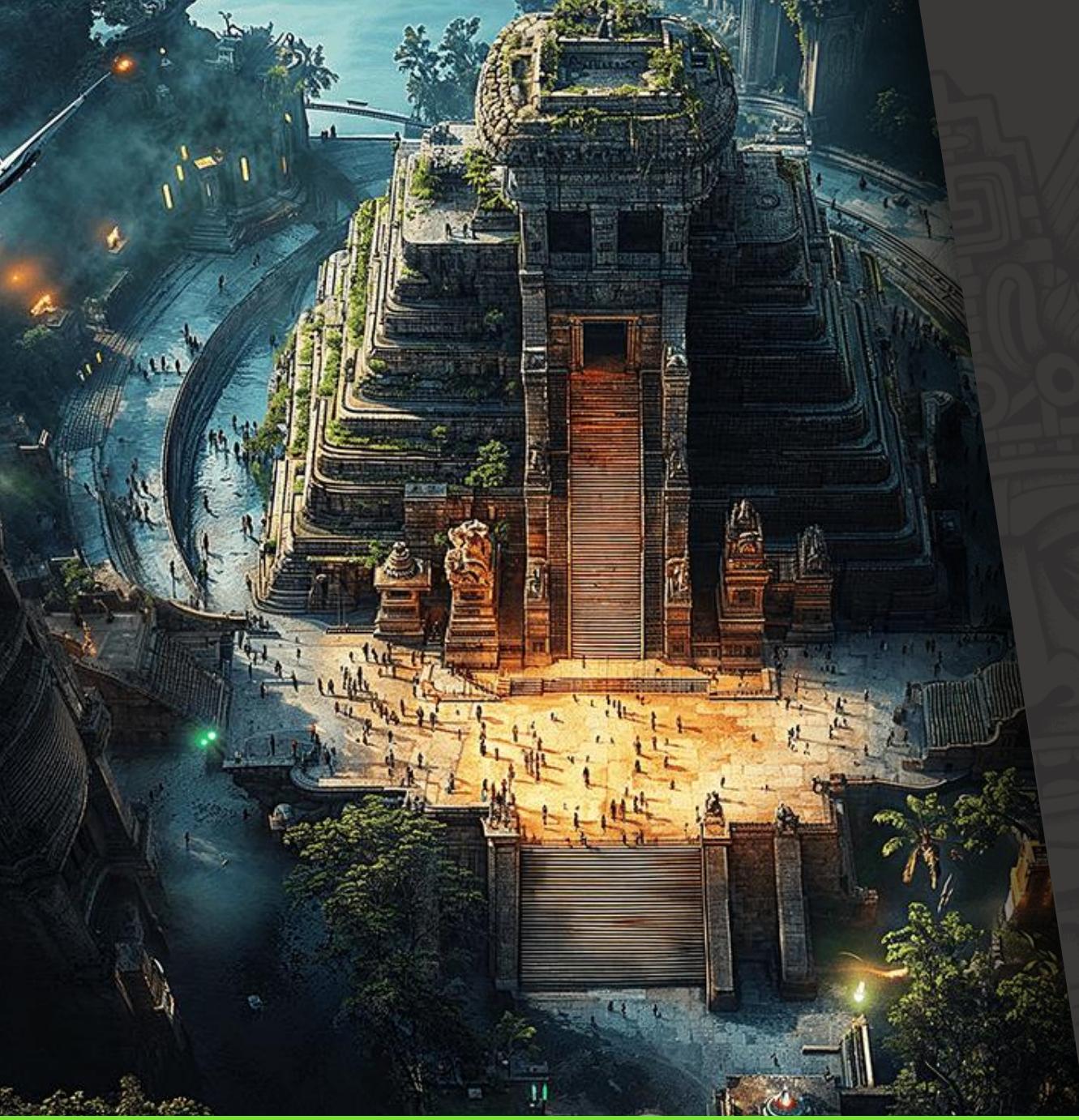
[OVERVIEW](#) [ACTIVITY](#) [SPONSORS](#) [PARTNERS](#) [GUIDELINES](#) [FAQ](#)

QUANTUM FOR
REAL-WORLD
IMPACT

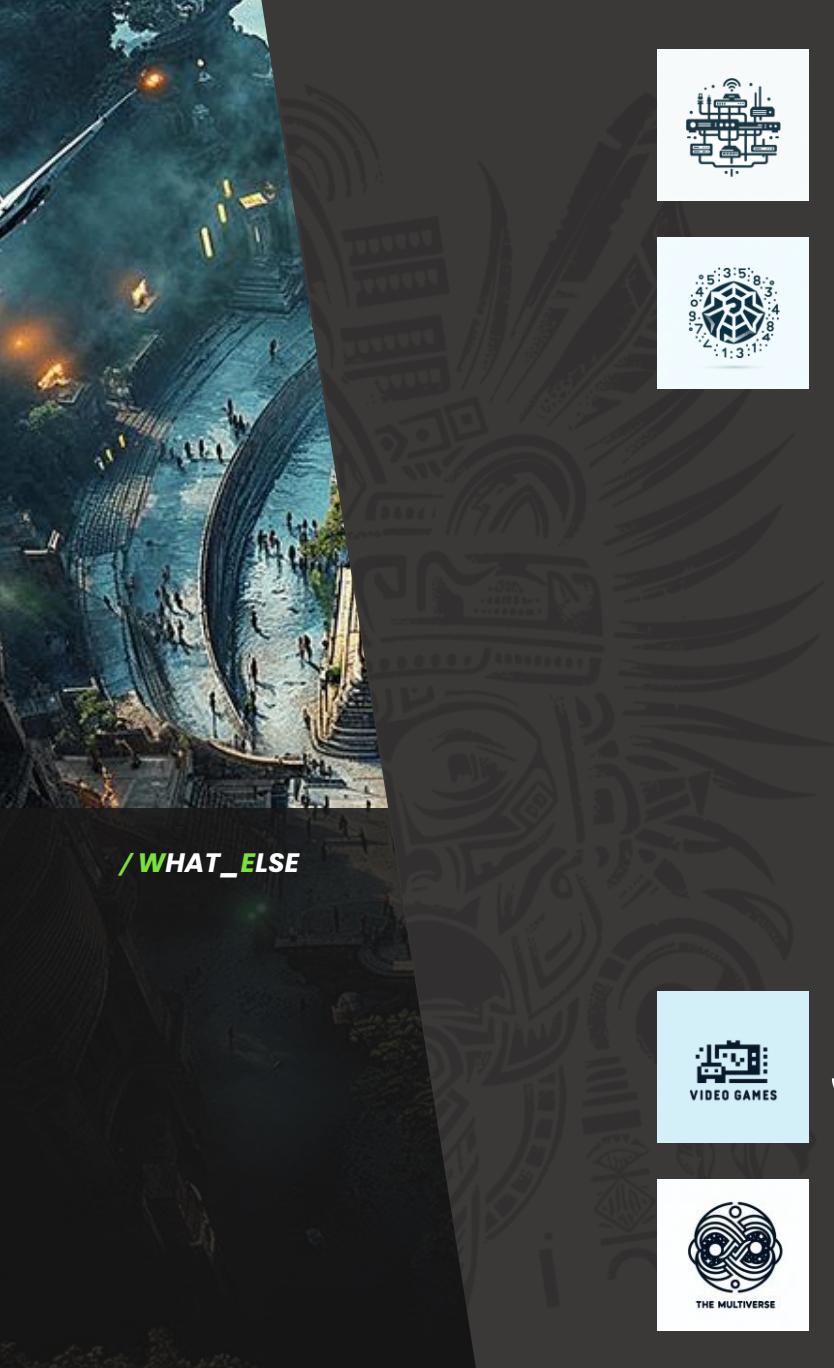
\$5 MILLION

Prize Purse

XPRIZE Quantum Applications is a 3-year, \$5M global competition designed to generate quantum computing (QC) algorithms that can be put into practice to help solve real-world challenges.



MAIS ENCORE ?



Réseaux



Nombres aléatoires

Jeux vidéo



Multivers



Google's Quantum Chip Sparks Debate On Multiverse Theory

Research Matt Swayne • December 16, 2024



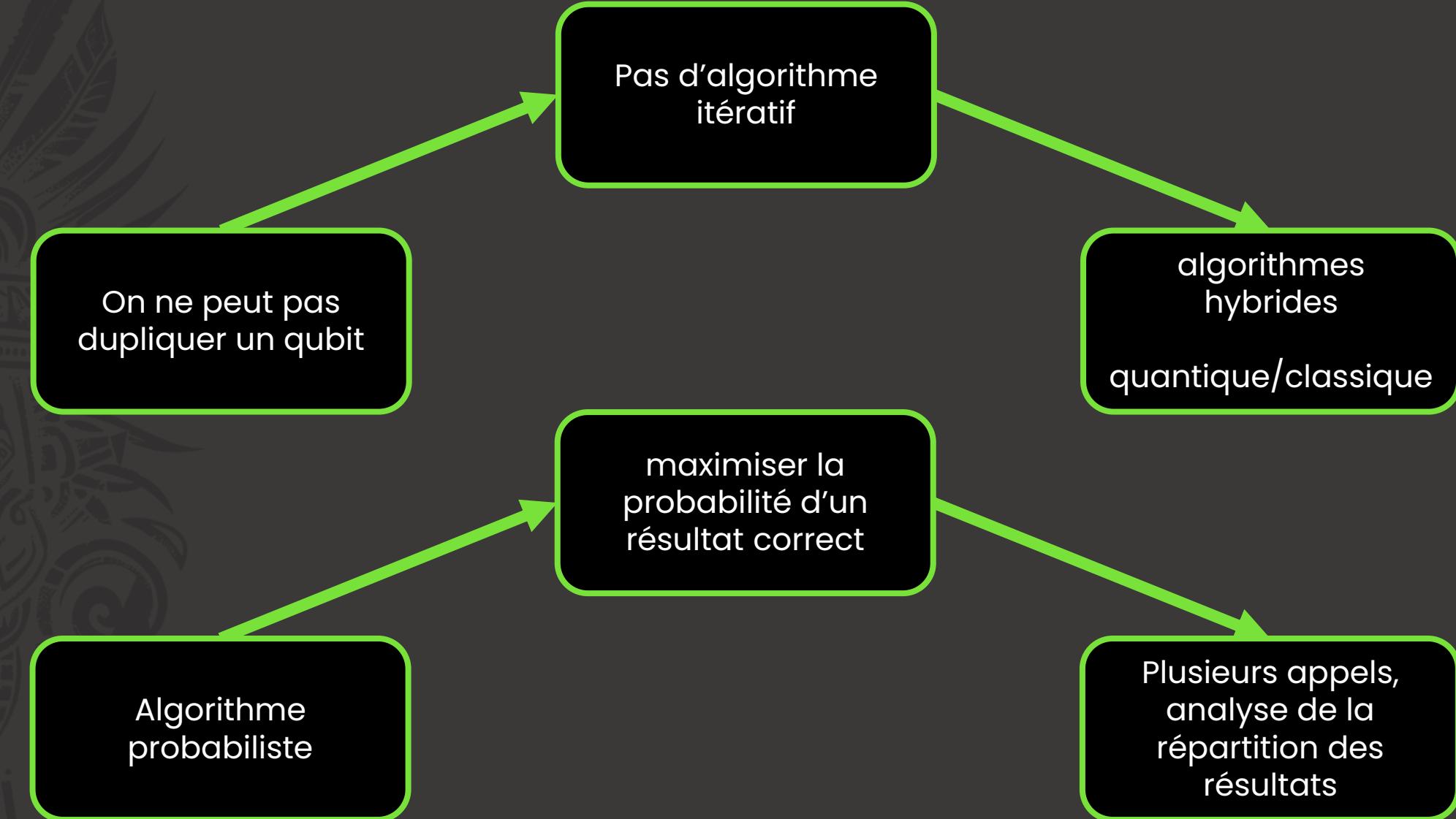
COMMENT CRÉER UN ALGORITHME QUANTIQUE ?

La dualité onde/corpuscule et La superposition



- Les assiettes sont à la fois entières et cassées
- Quand on ouvrira la porte, leur état sera fixé

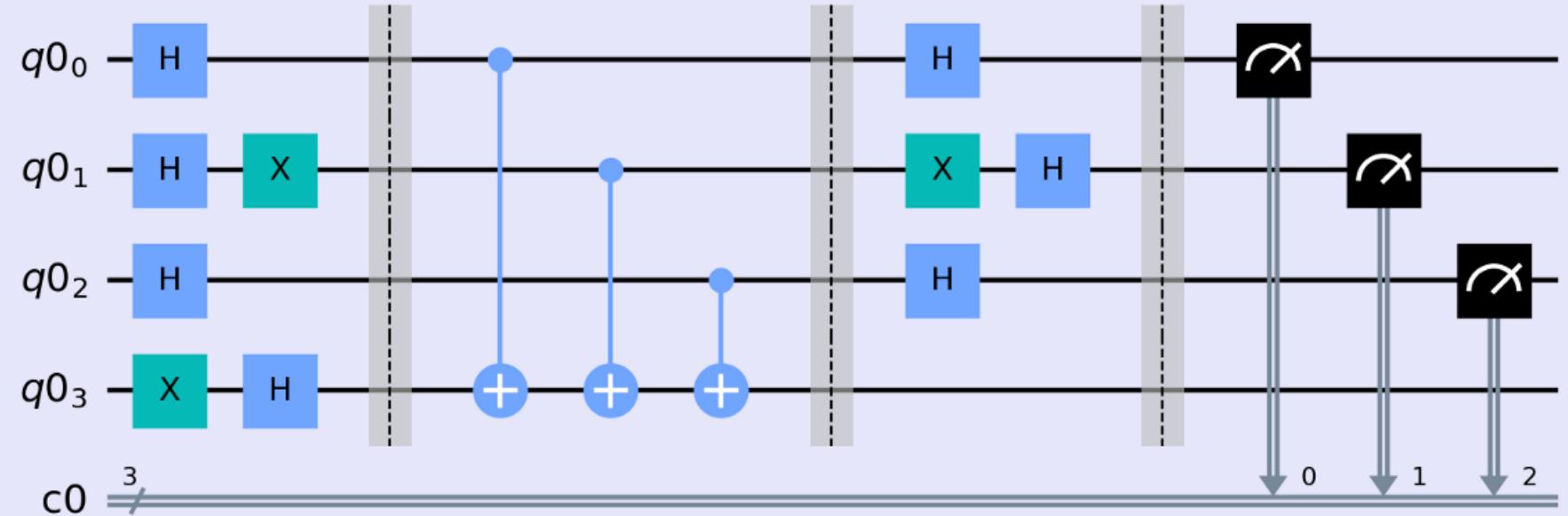
Un changement de paradigme





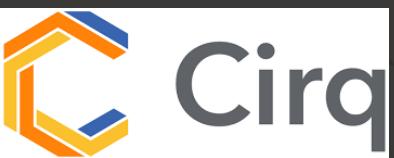
COMMENT ON
PROGRAMME ÇA?

- Globalement avec des portes quantiques
 - Portes de Pauli
 - Portes Hadamard
 - ...



- OpenQASM
 - Langage de description standard
 - Syntaxe proche du C

```
OPENQASM 3.0;
defcalgrammar "openpulse";
cal {
    extern constant(duration, float[64]) -> waveform;
    extern gaussian(duration, duration, float[64]) -> waveform;
    port dac1;
    port adc0;
    port dac0;
    frame tx_frame = newframe(dac1, 5752000000.0, 0);
    frame rx_frame = newframe(adc0, 5752000000.0, 0);
    frame xy_frame = newframe(dac0, 6431000000.0, 0);
}
duration delay_time = 0.0ns;
defcal reset $1 {
    delay[1000000.0ns];
}
defcal measure $1 {
    play(tx_frame, constant(2400.0ns, 0.2));
    capture(rx_frame, constant(2400.0ns, 1));
}
defcal x90 $1 {
    play(xy_frame, gaussian(32.0ns, 8.0ns, 0.2063));
}
for int shot_index in [0:99] {
    delay_time = 0.0ns;
    for int delay_index in [0:100] {
        reset $1;
        x90 $1;
        delay[delay_time] $1;
        x90 $1;
        measure $1;
        delay_time += 100.0ns;
    }
}
```



- Framework édité par Google AI Quantum Team
- Syntaxe Python
- Exécution en simulateur ou sur des machines IonQ, Pasqal, Rigetti

A screenshot of a Jupyter Notebook interface titled "Untitled1.ipynb". The code cell contains Python code for creating a quantum circuit:

```
[1] !pip install --quiet cirq
[2] import cirq
import cirq

# Pick a qubit.
qubit = cirq.GridQubit(0, 0)

# Create a circuit
circuit = cirq.Circuit(
    cirq.X(qubit)**0.5, # Square root of NOT.
    cirq.measure(qubit, key='m') # Measurement.
)
print("Circuit:")
print(circuit)

# Simulate the circuit several times.
simulator = cirq.Simulator()
result = simulator.run(circuit, repetitions=20)
print("Results:")
print(result)
```



- Framework créé par Microsoft
- Intégré à VS Code
- Exécution en simulateur ou sur des machines IonQ, Quantinuum, Rigetti, Microsoft

The screenshot shows the Microsoft Q# extension for VS Code. The main editor window displays the following Q# code:

```
C: > Users > Q# RandomNum.qs
1  /// # Sample
2  /// Random Bit
3  ///
4  /// # Description
5  /// This Q# program generates a random bit by setting a qubit in a superposition
6  /// of the computational basis states |0> and |1>, and returning the result.
7  ///
8  namespace Sample {
9
10    @EntryPoint()
11    Run | Histogram | Estimate | Debug
12    operation RandomBit() : Result {
13      // Qubits are only accessible for the duration of the scope where they
14      // are allocated and are automatically released at the end of the scope.
15      use qubit = Qubit();
16
17      // Set the qubit in superposition by applying a Hadamard transformation.
18      H(qubit);
19
20      // Measure the qubit. There is a 50% probability of measuring either
21      // `Zero` or `One`.
22    }
23 }
```

The Explorer sidebar on the left shows the following structure:

- EXPLORER
- > NO FOLDER OPENED
- > OUTLINE
- > TIMELINE
- > METADATA
- QUANTUM WORKSPACES
 - DocWorkspace
 - Providers
 - Microsoft
 - ionq
 - quantinuum
 - rigetti
 - rigetti.sim.qvm
 - rigetti.qpu.ankaa-9q-3
 - rigetti.qpu.ankaa-3
 - microsoft-qc
 - Jobs
 - MyQuantumJob
 - MyPythonJob
 - MyQuantumJob



Qiskit

- Framework édité par IBM
- Utilise la langage Python ou une interface graphique
- Permet d'exécuter les algorithmes sur simulateur ou sur les machines réelles d'IBM

The screenshot shows the IBM Quantum Learning interface. At the top, there's a navigation bar with 'IBM Quantum Learning', 'Home', 'Catalog', and 'Composer'. The 'Composer' tab is selected. On the left, there's a sidebar with 'Operations' and a search bar. The main area displays a quantum circuit with two qubits (q[0] and q[1]) and one classical register (c[0]). The circuit consists of two Hadamard (H) gates followed by a CNOT gate between q[0] and q[1]. Below the circuit, there's a 'Probabilities' section showing a bar chart for 1024 shots. The x-axis has four bins labeled '00', '01', '10', and '11', with probabilities around 25% each. To the right of the circuit, there's a code editor window titled 'HelloWorld' with the following Python code:

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8 circuit.h(qreg_q[0])
9 circuit.h(qreg_q[1])
10 circuit.measure(qreg_q[0], creg_c[0])
11 circuit.measure(qreg_q[1], creg_c[1])
```



UNE PETITE DÉMO?



EN RÉSUMÉ





L'informatique quantique

- En général ...
 - Encore très **expérimental**
 - Beaucoup d'**effets d'annonces**
 - Des coûts encore **prohibitifs**
- Pour moi développeur ...
 - **Un changement de paradigme pour la conception d'algorithmes**
 - **Une utilisation probablement plutôt sous forme de services**
- Et mes bitcoins ...
 - Il faudrait **8h et 20 000 000 Qubits pour casser une clé RSA 2048**
 - Bitcoins actifs vulnérables le **temps d'une transaction (10 min)**

/TAKEAWAY

Slides

MERCI

Feedback

MERCI
POUR
VOTRE
ATTENTION



DEVOXX
FRANCE 2025



@jerome-leonard-dev.bsky.social



@jerome-leonard