# Who Shall Win :Randomness or Strategy?

# About the problem

- The classical game of *Rock-Paper-Scissors* becomes biased when two people play against each other. One may easily catch on to their opponent's strategy if the opponent plays in a regular pattern. One might extend the classical game by introducing new moves as in the game *Rock-Paper-Scissors-Lizard-Spock* to mitigate this. But even then, the internal psychological biases of the human brain might not let the game be truly random, creating a stand-off .
- We propose a solution to this predicament by generating truly random moves using Quantum Random Number Generator. The computer will use the random generated moves to play against a human (who will play by using hand gestures).

# Aim of the project

- Develop a basic understanding of the quantum computing
- Explore different ways to create a quantum random number generator
- Take video input from user and identify different hand gesture .
- Develop a interface to play Rock-Paper-Scissor-Lizard-Spock against quantum computer.
- Analysing human psychology based strategy against complete randomness .

# What we achieved in this project

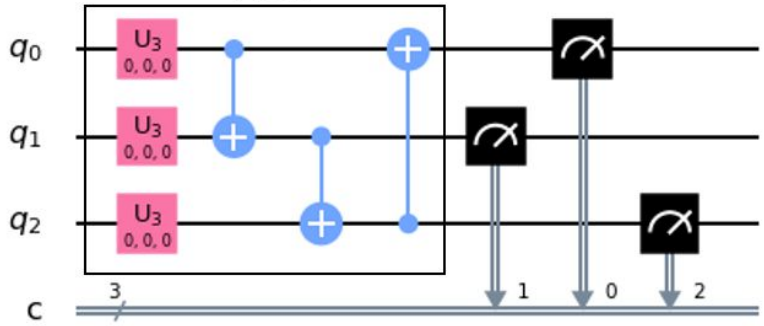
- Created a Quantum random generator using noise only over QuasmSimulator in Qiskit.
- Linked Qrng with web game .
- Train a classical machine learning model using python to identify different hand gestures for rock, paper, scissors, lizard and spock.
- Developed web interface for game using html, css ,javascript.

# Noise quantum number generator

- Noise is always present in the hardware of quantum computers, we can use this noise to our advantage to create a Random Number Generator.
- We first build a basic bit flip error noise model and map it to the Qasm Simulator.
- The different probabilities for an error to occur are taken from qiskit documentation.
- When resetting a qubit reset to 1 instead of 0 with probability pb_reset.
- When measuring a qubit, flip the state of the qubit with probability pb_meas.
- When applying a single qubit gate, flip the state of the qubit with probability pb_gate1.
- When applying a 2-qubit gate apply single-qubit errors to each qubit.

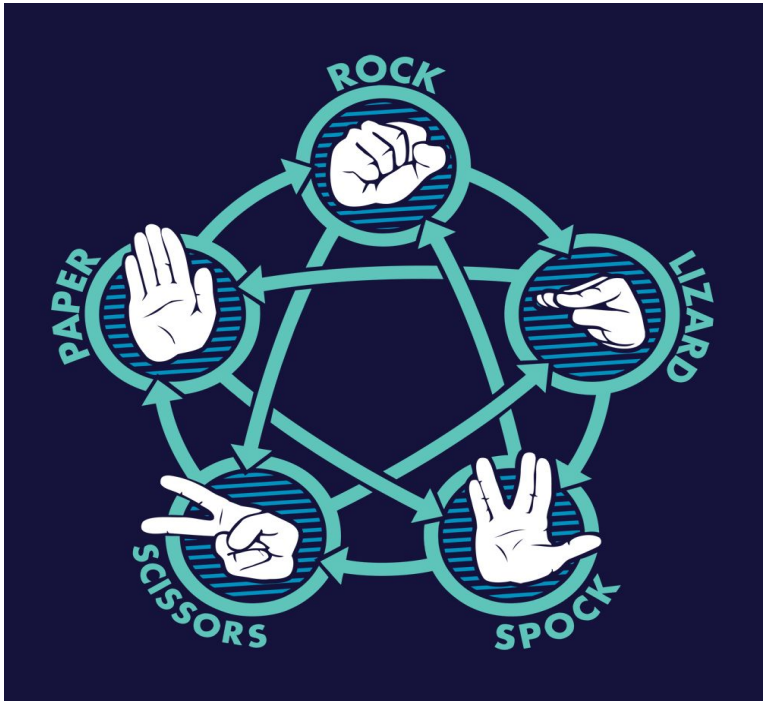
```
# Error probabilities
pb_reset = 0.03
pb_meas = 0.1
pb_gate1 = 0.05

# QuantumError objects
er_reset = pauli_error([('X', pb_reset), ('I', 1 - pb_reset)])
er_meas = pauli_error([('X',pb_meas), ('I', 1 - pb_meas)])
er_gate1 = pauli_error([('X',pb_gate1), ('I', 1 - pb_gate1)])
er_gate2 = er_gate1.tensor(er_gate1)

# Add errors to noise model
noise = NoiseModel()
noise.add_all_qubit_quantum_error(er_reset, "reset")
noise.add_all_qubit_quantum_error(er_meas, "measure")
noise.add_all_qubit_quantum_error(er_gate1, ["u1", "u2", "u3"])
noise.add_all_qubit_quantum_error(er_gate2, ["cx"])

return noise
```

- We create a quantum circuit full of I and CX gates. I and CX essentially have no effect on the circuit. They are there to add ample of noise.
- This circuit without noise should have no effect, i.e., output ‘000’ but with ample of noise results in random numbers.




```
def random_number():

    circ = QuantumCircuit(3)
    simulator = QasmSimulator()

    #NQRNS Circuit
    for i in range(200):
        circ.u3(0,0,0,0)
        circ.u3(0,0,0,1)
        circ.u3(0,0,0,2)
        circ.cx(0,1)
        circ.cx(1,2)
        circ.cx(0,2)
        circ.barrier()

    circ.measure_all()

    noise = get_noise()
    #get output
    job = execute(circ, simulator,
                  basis_gates=noise.basis_gates,
                  noise_model=noise, shots= 1)
    result = job.result()
    counts = result.get_counts(0)
```

# How to play the game




## Rules of the game

- Scissors cuts Paper
- Paper covers Rock
- Rock crushes Lizard
- Lizard poisons Spock
- Spock smashes Scissors
- Scissors decapitates Lizard
- Lizard eats Paper
- Paper disproves Spock
- Spock vaporizes Rock
- Rock crushes Scissors

# How to use GUI?

- Click on configure to configure the webcam
- Now make the hand gesture of whichever symbol you want to choose (i.e rock, paper, scissor, lizard, spock)
- Click on “Take Snapshot”
- Continue playing until either of you or computer scores 5 points
- Whoever scores 5 points first , wins the game
- Start new game for reloading the game .