# What is MUSIQu?

- A qubit exists in a superposition of states
- Similarly, a musical chord is a superposition of different notes

MUSIQu aims to represent quantum computing in the context of music

Purpose: to present quantum computing in a unique, engaging, and potentially artistic way - allows users to physically *hear* the effects of qubit operations

# How it Works: Musical Mappings

| Qubit State | Musical Note |
|---|---|
| \|000> | C4 - (261.63 Hz) |
| \|001> | E4 - (329.63 Hz) |
| \|010> | G4 - (392.00 Hz) |
| \|011> | B4 - (493.88 Hz) |
| \|100> | D4 - (293.66 Hz) |
| \|101> | F4 - (349.23 Hz) |
| \|110> | A4 - (440.00 Hz) |
| \|111> | C5 - (523.25 Hz) |

Three Steps:

1.) Prepare qubit sequence

2.) Apply transformations

3.) Map qubits to audio output

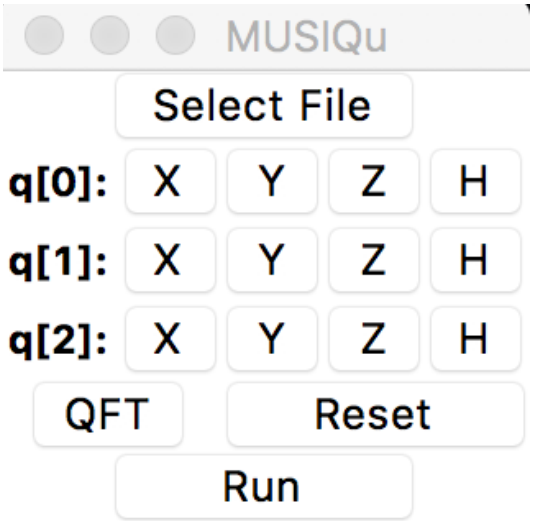
* Where the volume of a note corresponds to the probability of measuring the qubit in that quantum state

Consider the qubit:

q = 0.5 |000> + 0.5 |001> + 0.5 |010> + 0.5 |011> This maps to C4, E4, G4, and B4 (Cmaj7 chord) - all played with equal volume

# Using the Program

First, run MUSIQu as a Python program. The following interface should pop up:




Instructions:

1.) Select input .txt file for qubit states

2.) Apply qubit transformations (Note: multiple operations can be active at once)

3.) Run at any time to hear the audio output

4.) Reset qubits to their original states, or load a new input .txt file altogether

Text File Input:

- Each line represents a single qubit - supports any number of lines
- Format: three numbers (either 0 or 1) encoding the qubit, followed by a space, and then an optional number denoting the note duration (default value is 1.0, or quarter note). Ex: 001 0.5 ( |001>, eighth note)
- For reference, example .txt files are provided

# A Few Quick Tips

- To play chords, qubits must exist in a superposition of states. This can be done through applying various Hadamard or quantum Fourier transforms
- Rhythms are for musical purposes only and do not correlate to the qubits