import matplotlib.pyplot as plt  
from matplotlib.patches import Rectangle, FancyBboxPatch, Circle  
  
import qutip  
from qutip\_qip.circuit import QubitCircuit  
from qutip\_qip.operations import Gate

WIRE\_SEP = 0.7  
LAYER\_SEP = 0.6  
PAD\_LABEL = 0.3  
GATE\_HEIGHT = 0.2  
GATE\_WIDTH = 0.2  
GATE\_PAD = (LAYER\_SEP - GATE\_WIDTH)/2  
FONT\_SIZE = 10

def draw\_wire(ax, n\_wires):  
  
 n\_layers=2  
 wires = [[(0, n\_layers\*LAYER\_SEP) , (i\*WIRE\_SEP, i\*WIRE\_SEP) ] for i in range(n\_wires)]  
  
 for wire in wires:  
 line = plt.Line2D(xdata=wire[0], ydata=wire[1], color='black', zorder=1)  
 ax.add\_line(line)

def extend\_wire(ax, n\_wires, old\_layers, add\_layers):  
   
 wires = [[(old\_layers \* LAYER\_SEP, (old\_layers + add\_layers) \* LAYER\_SEP), (i \* WIRE\_SEP, i \* WIRE\_SEP)] for i in range(n\_wires)]  
  
 for wire in wires:  
 line = plt.Line2D(xdata=wire[0], ydata=wire[1], color='black', zorder=1)  
 ax.add\_line(line)

def add\_labels(ax, n\_wires, wire\_labels=[]):  
  
 if wire\_labels == []:  
 wire\_labels = [f"$q\_{{{i}}}$" for i in range(n\_wires)]  
   
 for i, label in enumerate(wire\_labels):  
 ax.text(-PAD\_LABEL, i \* WIRE\_SEP, label, fontsize=FONT\_SIZE, verticalalignment='center')

def draw\_control\_node(ax, layer, target):  
  
 # make a circle patch  
 node = Circle((layer \* LAYER\_SEP + LAYER\_SEP / 2, target \* WIRE\_SEP), 0.05, facecolor='blue', zorder=2)  
 ax.add\_patch(node)

def draw\_target\_node(ax, layer, target):  
 TARGET\_RADIUS = 0.12  
   
 # Draw the target node as a circle  
 node = Circle((layer \* LAYER\_SEP + LAYER\_SEP / 2, target \* WIRE\_SEP), TARGET\_RADIUS, facecolor='blue', zorder=2)  
 ax.add\_patch(node)  
   
 # Draw plus sign  
 vertical = plt.Line2D((layer \* LAYER\_SEP + LAYER\_SEP / 2, layer \* LAYER\_SEP + LAYER\_SEP / 2),  
 (target \* WIRE\_SEP - TARGET\_RADIUS/2, target \* WIRE\_SEP + TARGET\_RADIUS/2),  
 linewidth=1.5, color='white', zorder=3)  
   
 horizontal = plt.Line2D((layer \* LAYER\_SEP + LAYER\_SEP / 2 - TARGET\_RADIUS/2, layer \* LAYER\_SEP + LAYER\_SEP / 2 + TARGET\_RADIUS/2),  
 (target \* WIRE\_SEP, target \* WIRE\_SEP),  
 linewidth=1.5, color='white', zorder=3)  
   
 ax.add\_line(vertical)  
 ax.add\_line(horizontal)

# add SWAP GATE mark "X"" on the target qubit  
def draw\_swap\_mark(ax, layer, wire):  
  
 # Draw the diagonal lines to form an "X"   
 dia\_left = plt.Line2D(  
 [layer \* LAYER\_SEP + LAYER\_SEP / 2 - GATE\_WIDTH / 3, layer \* LAYER\_SEP + LAYER\_SEP / 2 + GATE\_WIDTH / 3],  
 [wire \* WIRE\_SEP - GATE\_HEIGHT / 2, wire \* WIRE\_SEP + GATE\_HEIGHT / 2],  
 color="blue", linewidth=1.5, zorder=3  
 )  
 dia\_right = plt.Line2D(  
 [layer \* LAYER\_SEP + LAYER\_SEP / 2 + GATE\_WIDTH / 3, layer \* LAYER\_SEP + LAYER\_SEP / 2 - GATE\_WIDTH / 3],  
 [wire \* WIRE\_SEP - GATE\_HEIGHT / 2, wire \* WIRE\_SEP + GATE\_HEIGHT / 2],  
 color="blue", linewidth=1.5, zorder=3  
 )  
   
 ax.add\_line(dia\_left)  
 ax.add\_line(dia\_right)

def draw\_bridge(ax, layer, wire1, wire2):  
  
 bridge = plt.Line2D(  
 [layer\*LAYER\_SEP + LAYER\_SEP/2, layer\*LAYER\_SEP + LAYER\_SEP/2],  
 [wire1\*WIRE\_SEP, wire2\*WIRE\_SEP],  
 color='blue', linewidth=1, zorder=2  
 )  
  
 ax.add\_line(bridge)

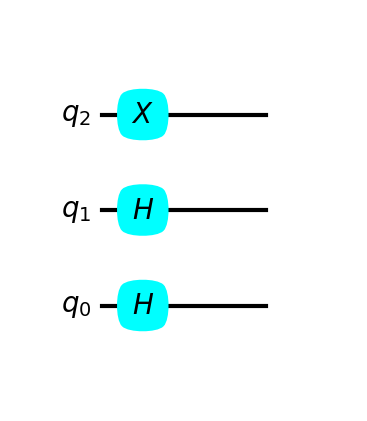
def multiq\_add\_gate(ax, layer, gate):  
  
 # check if gate is a qutip Gate  
 if not isinstance(gate, Gate):  
 print("Gate is not a qutip Gate")  
 return  
   
 # check if gate has multiple targets  
 if gate.name == "CNOT":  
 draw\_control\_node(ax, layer, gate.controls[0],)  
 draw\_target\_node(ax, layer, gate.targets[0])  
 draw\_bridge(ax, layer, gate.targets[0], gate.controls[0])  
  
 if gate.name == "SWAP":  
 draw\_swap\_mark(ax, layer, gate.targets[0])  
 draw\_swap\_mark(ax, layer, gate.targets[1])  
 draw\_bridge(ax, layer, gate.targets[0], gate.targets[1])

# adding gate, currently only works for qutip Gates  
def add\_gate(ax, layer, gate):  
   
 if isinstance(gate, Gate):  
 pass  
 # print("Gate is a qutip Gate")  
 else:  
 pass  
 # print("Gate is not a qutip Gate")  
  
   
 gate\_label = f"${gate.name}$"  
 gate\_wire = gate.targets[0]  
  
 gate\_patch = FancyBboxPatch(  
 (layer \* LAYER\_SEP + GATE\_PAD, gate\_wire \* WIRE\_SEP - GATE\_HEIGHT / 2),   
 GATE\_WIDTH,   
 GATE\_HEIGHT,   
 boxstyle="round4",   
 mutation\_scale=0.3,   
 facecolor="aqua",   
 edgecolor="aqua",  
 zorder=2  
 )   
   
 # gate\_patch = Rectangle((layer\*LAYER\_SEP + GATE\_PAD, gate\_wire\*WIRE\_SEP - GATE\_HEIGHT/2), GATE\_WIDTH, GATE\_HEIGHT, facecolor='aqua', zorder=2)  
 ax.add\_patch(gate\_patch)  
  
 # add gate label  
 ax.text(layer\*LAYER\_SEP + GATE\_PAD + GATE\_WIDTH/2, gate\_wire\*WIRE\_SEP, gate\_label, fontsize=FONT\_SIZE, verticalalignment='center', horizontalalignment='center')

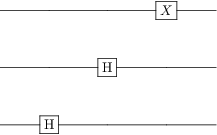
def canvas\_plot(qc, height, width):  
   
 fig, ax = plt.subplots(figsize=(width, height))  
  
 nwire = qc.N  
 display\_layers = 2  
 draw\_wire(ax, nwire) # generalize TODO   
 add\_labels(ax, nwire)  
  
 # make a dict with wire\_labels as keys and zero values  
 gate\_maintain = {f"q{i}": 0 for i in range(nwire)}  
  
 for gate in qc.gates:  
  
 if len(gate.targets) == 1 and gate.controls == None:  
 add\_gate(ax, gate\_maintain[f"q{gate.targets[0]}"], gate)  
 gate\_maintain[f"q{gate.targets[0]}"] += 1  
 else:  
 multiq\_add\_gate(ax, gate\_maintain[f"q{gate.targets[0]}"], gate)  
 # update all values to multiqubot target  
 temp = gate\_maintain[f"q{gate.targets[0]}"] + 1  
 gate\_maintain = {key:temp for key in gate\_maintain.keys()}  
   
 if max(gate\_maintain.values()) >= display\_layers:  
 extend\_wire(ax, nwire, display\_layers, add\_layers=1)  
  
 display\_layers += 1  
   
   
 ax.set\_ylim(-WIRE\_SEP, nwire\*WIRE\_SEP)  
 ax.set\_xlim(-LAYER\_SEP, (display\_layers + 1)\*LAYER\_SEP)  
 ax.set\_aspect('equal')  
 ax.axis('off')  
 fig.set\_dpi(200)

def calsize(qc):  
  
 n\_wires = qc.N   
 height = n\_wires \* 0.393701 \* WIRE\_SEP \* 3  
 width = 10  
  
 return height, width

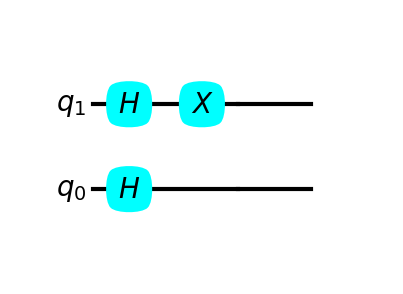
qc = QubitCircuit(3)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("H", targets=[1])  
qc.add\_gate("X", targets=[2])  
qc.gates[0].targets, qc.png  
  
h, w = calsize(qc)  
canvas\_plot(qc, h, w)



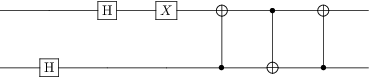
qc



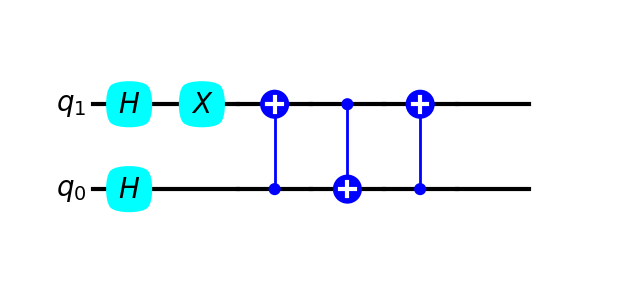
qc = QubitCircuit(2)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("H", targets=[1])  
qc.add\_gate("X", targets=[1])  
qc.gates[0].targets, qc.png  
  
h, w = calsize(qc)  
canvas\_plot(qc, h, w)



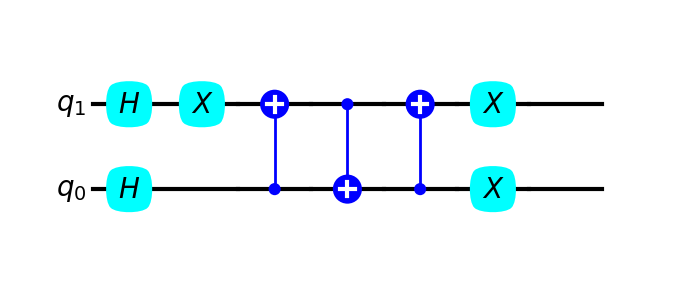
qc = QubitCircuit(2)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("H", targets=[1])  
qc.add\_gate("X", targets=[1])  
qc.add\_gate("CNOT", controls=[0], targets=[1])  
qc.add\_gate("CNOT", controls=[1], targets=[0])  
qc.add\_gate("CNOT", controls=[0], targets=[1])  
qc



h, w = calsize(qc)  
canvas\_plot(qc, h, w)

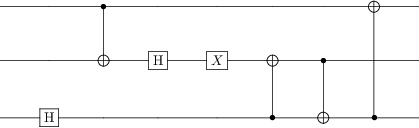


qc.add\_1q\_gate("X")  
canvas\_plot(qc, h, w)

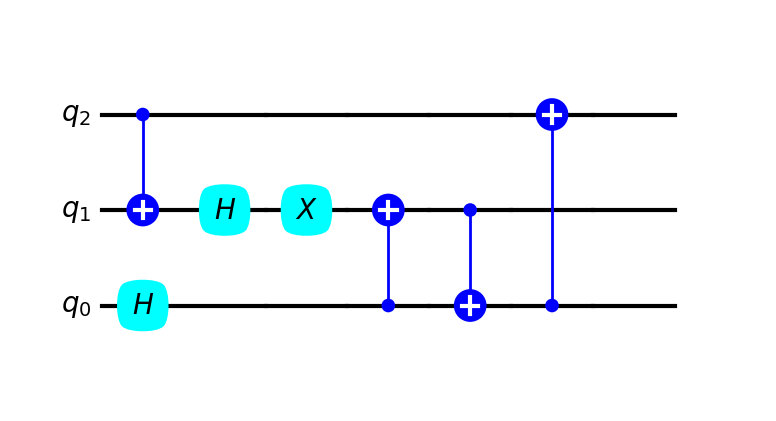


qc = QubitCircuit(3)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("CNOT", targets=[1], controls=[2])  
qc.add\_gate("H", targets=[1])  
qc.add\_gate("X", targets=[1])  
qc.add\_gate("CNOT", controls=[0], targets=[1])  
qc.add\_gate("CNOT", controls=[1], targets=[0])  
qc.add\_gate("CNOT", controls=[0], targets=[2])

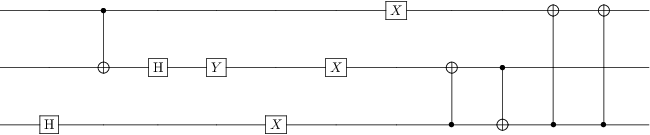
qc



h, w = calsize(qc)  
canvas\_plot(qc, h, w)



qc = QubitCircuit(3)  
qc.add\_gate("H", targets=[0])  
qc.add\_gate("CNOT", targets=[1], controls=[2])  
qc.add\_gate("H", targets=[1])  
qc.add\_gate("Y", targets=[1])  
qc.add\_1q\_gate("X")  
qc.add\_gate("CNOT", controls=[0], targets=[1])  
qc.add\_gate("CNOT", controls=[1], targets=[0])  
qc.add\_gate("CNOT", controls=[0], targets=[2])  
qc.add\_gate("CNOT", controls=[0], targets=[2])  
qc



h, w = calsize(qc)  
canvas\_plot(qc, h, w)

