Let us try to create here, through different cells, all the necessary material to run a very first simulation. We need: user inputs for initial positions; user-given (only for now!) values of 'reward' probability amplitudes; a bunch of operations, a Qiskit extension, and a loop to run the code multiple times. If we are even able to create a visual representation which is updated at each time step, that would be so cool!

(We have to use ESC + M to write a text rather than a coding line in Jupyter). The following cell shouldn't change across time.

In [1]:

```
# Some resources on Python robot simulation can be found here: https://jyro.read
```

In [2]:

```
from ibm_quantum_widgets import CircuitComposer
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, Aer, exec
import numpy as np
from numpy import pi
from ibm_quantum_widgets import draw_circuit
from qiskit.providers.aer import QasmSimulator
from qiskit.utils import QuantumInstance
from qiskit.visualization import plot_histogram, plot_state_qsphere
from qiskit import *
import random
import matplotlib.pyplot as plt
import pylab
import pandas as pd
from sklearn import preprocessing
import collections
from collections import Counter
```

In [3]:

```
# from: https://stackoverflow.com/questions/39298928/play-multiple-sounds-at-the

from pydub import AudioSegment
from pydub.playback import play
```

In [4]:

```
# 3d graph: Adapted from https://stackoverflow.com/questions/12423601/simplest-w

import sys
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
import numpy
from numpy import array
from scipy import newaxis
```

In [5]:

```
# mlab.points3d(x, y, z, value)
```

In [6]:

```
from mayavi import mlab
mlab.options.offscreen = True
mlab.test_contour3d()
mlab.savefig('example.png')
mlab.draw()
mlab.show()

# now it works :)
```

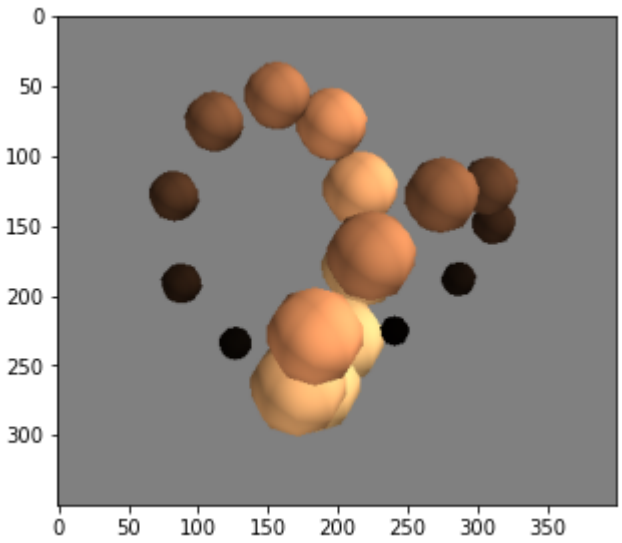
In [7]:

```
# example from https://stackoverflow.com/questions/67045944/embed-mayavi-into-a-

mlab.clf()

from mayavi import mlab
mlab.init_notebook()
import numpy as np

class Test:
    def __init__(self):
        self.fig = mlab.figure()
        self._add_data()

    def _add_data(self):
        pi = np.pi
        cos = np.cos
        sin = np.sin
        dphi, dtheta = pi / 250.0, pi / 250.0
        [phi, theta] = np.mgrid[0:pi + dphi * 1.5:dphi,
                                0:2 * pi + dtheta * 1.5:dtheta]
        m0 = 4; m1 = 3; m2 = 2; m3 = 3
        m4 = 6; m5 = 2; m6 = 6; m7 = 4
        r = sin(m0 * phi) ** m1 + cos(m2 * phi) ** m3 + \
            sin(m4 * theta) ** m5 + cos(m6 * theta) ** m7
        x = r * sin(phi) * cos(theta)
        y = r * cos(phi)
        z = r * sin(phi) * sin(theta)
        self.surf = mlab.mesh(x, y, z, figure=self.fig)

    def show(self):
        return self.surf

t=Test()
t.show()
```

```
Notebook initialized with ipy backend.
```

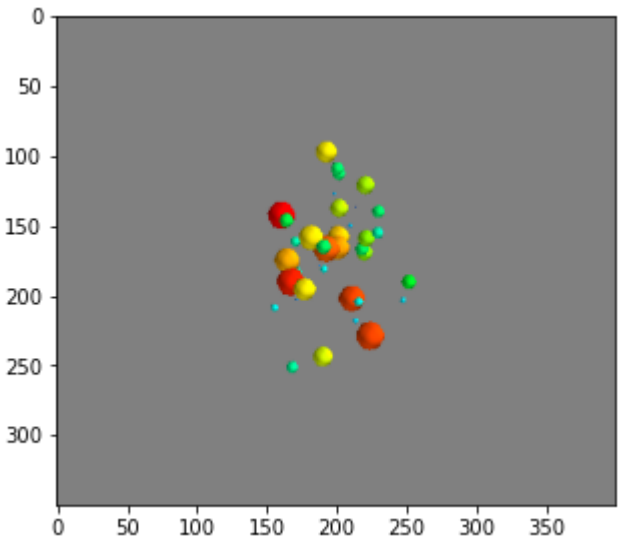
In [8]:

```
mlab.clf()

import numpy, pylab, mayavi, mayavi.mlab
import matplotlib.pyplot as plt

t = numpy.linspace(0, 4 * numpy.pi, 20)
cos,sin = numpy.cos, numpy.sin

x = sin(2 * t)
y = cos(t)
z = cos(2 * t)
s = 2 + sin(t)
mayavi.mlab.points3d(x, y, z, s, colormap="copper", scale_factor=.25)

arr = mayavi.mlab.screenshot()
fig = plt.figure(figsize=(5, 5))
pylab.imshow(arr)

plt.show()

```



In [9]:

```
 mlab.clf()

import numpy, pylab, mayavi, mayavi.mlab
import matplotlib.pyplot as plt

x, y, z, value = np.random.random((4, 40))
mlab.points3d(x, y, z, value)

arr = mayavi.mlab.screenshot()
fig = plt.figure(figsize=(5, 5))
pylab.imshow(arr)

plt.show()
```



Circuit components initialization. The specific qubits are on $|0\rangle$ by default. They will get a gate later on, according on attributes of classes. The following cell shouldn't change across time.

In [10]:

```
q = QuantumRegister(7, 'q'); # qubits # changed to 9, formerly 15
m3 = ClassicalRegister(1, 'c1'); # classical bits (separated is better)
m4 = ClassicalRegister(1, 'c2');
m5 = ClassicalRegister(1, 'c3');
m6 = ClassicalRegister(1, 'c4');

qc3 = QuantumCircuit(q, m3, m4, m5, m6); # to reach the target
qc4 = QuantumCircuit(q, m3, m4, m5, m6); # to get back to the nest
```

In [11]:

```
class Target:
    def __init__(self,name,x,y,z): # no indetermination in the target's positior
        self.name = name
        self.x = x
        self.y = y
        self.z = z
```

In [12]:

```
T = Target("T", 0.8, 0.8, 0.2) # deep in the ocean

# for getting back to the beginning
T2 = Target("T2", 0.2, 0.2, 1.0) # back to the ship
```

In [13]:

```
def reward(T, betax, betay, betaz):
    r = round(1 - ((T.x - betax)**2 + (T.y - betay)**2 + (T.z - betaz)**2)**0.5,
    # the closer the target, the less the distance, the higher the reward
    return r
```

Robot R_1: x-position $|q_0(t)\rangle = \alpha_1^x(t)|0\rangle + \beta_1^x(t)|1\rangle$; y-position $|q_1(t)\rangle = \alpha_1^y(t)|0\rangle + \beta_1^y(t)|1\rangle$; reward $|q_2(t)\rangle = \gamma_1(t)|0\rangle + \delta_1(t)|1\rangle$.

The class-initialization cells shouldn't change across time. However, cells with numerical values of class attributes should be updated.

In [14]:

```
class Robot:
  def __init__(self,name, alphax, betax, alphay, betay, alphaz, betaz, gamma, de
    self.name = name
    self.alphax = alphax
    self.betax = betax
    self.alphay = alphay
    self.betay = betay
    self.alphaz = alphaz
    self.betaz = betaz
    delta = reward(T, betax, betay, betaz)
    gamma = round(1 - delta, 2)
    self.gamma = gamma
    self.delta = delta
```

In [15]:

```
# manual intervention needed here to avoid circularity
reward(T, 0.2, 0.2, 0.7) # value of delta 0.2,0.2, 1.0
```

Out[15]:

```
0.02
```

In [16]:

```
# manual intervention needed here to avoid circularity
round(1 - reward(T, 0.2, 0.2, 0.7), 2) # value of gamma
```

Out[16]:

```
0.98
```

The following cell, and the other corresponding cells, should be updated by hand at each time:

In [17]:

```
# (name, alphax, betax, alphay, betay, gamma, delta)
R1 = Robot("R1",0.8, 0.2, 0.8, 0.2, 0.3, 0.7, 0.98, 0.02)
```

In [18]:

```
R1.gamma, R1.delta
```

Out[18]:

```
(0.98, 0.02)
```

Robot R_2: x-position $|q_3(t)\rangle = \alpha_2^x(t)|0\rangle + \beta_2^x(t)|1\rangle$; y-position $|q_4(t)\rangle = \alpha_2^y(t)|0\rangle + \beta_2^y(t)|1\rangle$; reward $|q_5(t)\rangle = \gamma_2(t)|0\rangle + \delta_2(t)|1\rangle$

In [19]:

```
reward(T, 0.2, 0.2, 0.6) # manual intervention needed here to avoid circularity
```

Out[19]:

```
0.06
```

In [20]:

```
round(1 - reward(T, 0.2, 0.2, 0.6), 2) # manual intervention needed here to avoi
```

Out[20]:

```
0.94
```

In [21]:

```
R2 = Robot("R2",0.8, 0.2, 0.8, 0.2, 0.4, 0.6, 0.94, 0.06) # update by hand this
```

In [22]:

```
R2.delta, R2.gamma, R2.alphax, R2.betax, R2.alphay, R2.betay, R2.alphaz, R2.beta
```

Out[22]:

```
(0.06, 0.94, 0.8, 0.2, 0.8, 0.2, 0.4, 0.6, 0.94, 0.06)
```

Robot R_3: x-position $|q_6(t)\rangle = \alpha_3^x(t)|0\rangle + \beta_3^x(t)|1\rangle$; y-position $|q_7(t)\rangle = \alpha_3^y(t)|0\rangle + \beta_3^y(t)|1\rangle$; reward $|q_8(t)\rangle = \gamma_3(t)|0\rangle + \delta_3(t)|1\rangle$

In [23]:

```
reward(T, 0.3, 0.1, 0.7) # manual intervention needed here to avoid circularity
```

Out[23]:

```
0.01
```

In [24]:

```
round(1 - reward(T, 0.3, 0.1, 0.7), 2) # manual intervention needed here to avoi
```

Out[24]:

```
0.99
```

In [25]:

```
R3 = Robot("R3", 0.7, 0.3, 0.9, 0.1, 0.3, 0.7, 0.99, 0.01) # to be updated by ha
R3.alphax, R3.betax, R3.alphay, R3.betay, R3.alphaz, R3.betaz, R3.gamma, R3.delt
```

Out[25]:

```
(0.7, 0.3, 0.9, 0.1, 0.3, 0.7, 0.99, 0.01)
```

In [26]:

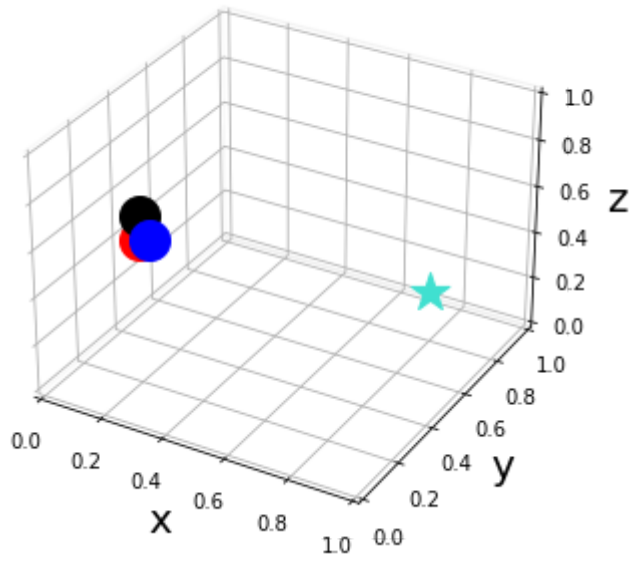
```
R3.gamma, R3.delta
```

Out[26]:

```
(0.99, 0.01)
```

In [27]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [28]:

```
R1.delta, R2.delta, R3.delta
```

Out[28]:

```
(0.02, 0.06, 0.01)
```

In [29]:

```
1 R3.alphay, R3.betay
```

Out[29]:

```
(0.9, 0.1)
```

In [30]:

```
1 # Audio section :)
```

In [31]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time1_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time1  = mixed_time1_.overlay(audio3)          # further combine , superi

mixed_time1.export("notes_/mixed_time1.mp3", format='mp3') # export mixed  audi
play(mixed_time1)                               # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tA#
fA#
cA#
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpl1a97skn.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.20 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
   7.27 M-A:  0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

In [32]:

```
# NEW! ---> January 13, 2022
```

NEW LINES of code: if the initial reward is high for all the three robots, but not 0.99 yet: --> randomly shuffle one of the positions.

In [33]:

```
if (R1.delta and R2.delta and R3.delta) >= 0.8 and (R1.delta and R2.delta and R3
    print("ciao ciao")
    R1.alphax = round(np.random.uniform(0,0.2), 3) # slightly shuffle position c
    R1.betax = round(1 - R1.alphax, 3)
    #R1.alphay = round(np.random.uniform(0,0.2), 3) # slightly shuffle position
    #R1.betay = round(1 - R1.alphay, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
```

In [34]:

```
R1.alphax, R1.betax, R1.alphay, R1.betay, R1.alphaz, R1.betaz
```

Out[34]:

```
(0.8, 0.2, 0.8, 0.2, 0.3, 0.7)
```

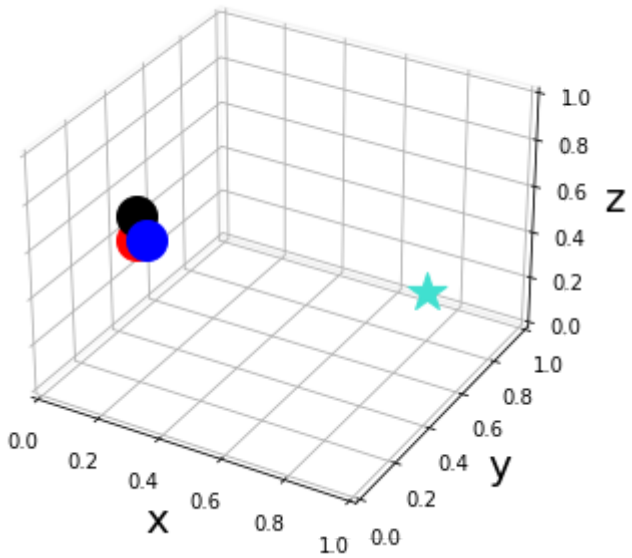
In [35]:

```
R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
print(R3.delta)
```

```
0.02
0.06
0.01
```

In [36]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



Rewards: here, they are an attribute of each class. This information should be provided by robots themselves according to their observations.

First check: if robots' positions are too far from the target, that is, initial positions guarantee a reward lower than a given threshold for all robots, then we have to change position. We can accomplish this by randomly moving robots (as in an exploration task), and evaluating again their rewards.

In [37]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time2_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time2  = mixed_time2_.overlay(audio3)          # further combine , superi

mixed_time2.export("notes_/mixed_time2.mp3", format='mp3') # export mixed  audi
play(mixed_time2)                                # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tA#
fA#
cA#
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpqj9fcqh7.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.23 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
   7.30 M-A:  0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

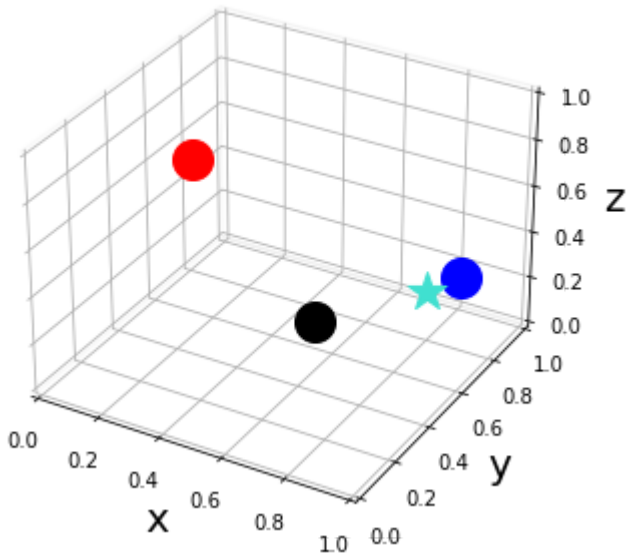
In [38]:

```
# threshold for initial reward
# random fluctuations

if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
    print("SOS")
    # R1
    R1.alphax = round(np.random.uniform(0,0.9), 3)
    R1.betax = round(1 - R1.alphax, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
    R1.alphay = round(np.random.uniform(0,0.9), 3)
    R1.betay = round(1 - R1.alphay, 3)
    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
    R1.alphaz = round(np.random.uniform(0,0.9), 3)
    R1.betaz = round(1 - R1.alphaz, 3)
    print("the new z-positions for R1 are: ", R1.alphaz, R1.betaz)
    # R2
    R2.alphax = round(np.random.uniform(0,0.9), 3)
    R2.betax = round(1 - R2.alphax, 3)
    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
    R2.alphay = round(np.random.uniform(0,0.9), 3)
    R2.betay = round(1 - R2.alphay, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    R2.alphaz = round(np.random.uniform(0,0.9), 3)
    R2.betaz = round(1 - R2.alphaz, 3)
    print("the new z-positions for R2 are: ", R2.alphaz, R1.betaz)
    # R3
    R3.alphax = round(np.random.uniform(0,0.9), 3)
    R3.betax = round(1 - R3.alphax, 3)
    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
    R3.alphay = round(np.random.uniform(0,0.9), 3)
    R3.betay = round(1 - R3.alphay, 3)
    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)
    R3.alphaz = round(np.random.uniform(0,0.9), 3)
    R3.betaz = round(1 - R3.alphaz, 3)
    print("the new y-positions for R3 are: ", R3.alphaz, R1.betaz)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
R1.gamma = 1 - R1.delta
R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
R2.gamma = 1 - R2.delta
R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
R3.gamma = 1 - R3.delta
print(R1.delta, R2.delta, R3.delta)
```

```
SOS
the new x-positions for R1 are:  0.377 0.623
the new y-positions for R1 are:  0.548 0.452
the new z-positions for R1 are:  0.765 0.235
the new x-positions for R2 are:  0.705 0.623
the new y-positions for R2 are:  0.658 0.452
the new z-positions for R2 are:  0.125 0.235
the new x-positions for R3 are:  0.134 0.623
the new y-positions for R3 are:  0.123 0.452
the new y-positions for R3 are:  0.767 0.235
0.61 0.04 0.89
```

In [39]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [40]:

```

# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
```

```
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
```

```
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
```

```
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
```

```
            print("cC2")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time3_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time3  = mixed_time3_.overlay(audio3)          # further combine , superi

mixed_time3.export("notes_/mixed_time3.mp3", format='mp3') # export mixed  audi
play(mixed_time3)                                  # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tE2
fA#
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpl516pxrk.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.26 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
   7.30 M-A:  0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

In [41]:

```
1 # January 22, 2022
```

I'm adding a check here as well.

NEW LINES of code: IF the initial reward is very high (greater than 0.8) for at least one of the three robots ("or"), THEN the other robots have to just reach it (with a pretty small fluctuation), without entering the circuit.

In [42]:

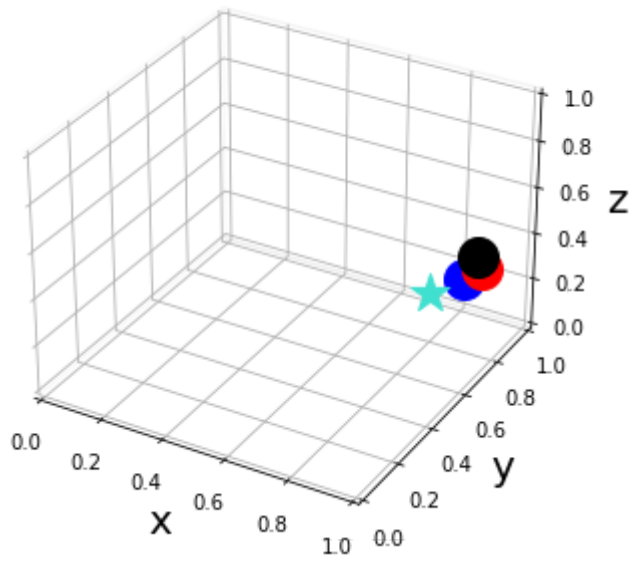
```
if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
    print('yuk')
    if (R1.delta > R2.delta and R1.delta > R3.delta):
        print('quokka')
        R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and late
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)
        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R2.betax, 3)
        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R2.betay, 3)
        R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R2.betaz, 3)
    if (R2.delta > R1.delta and R2.delta > R3.delta):
        print('quagga')
        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R3.betax, 3)
        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R3.betay, 3)
        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R3.betaz, 3)
    if (R3.delta > R1.delta and R3.delta > R2.delta):
        print('quark')
        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
print(R2.delta)
```

```
yuk
quark
0.81
```

```
0.83
0.83
```

In [43]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [44]:

```

# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
```

```
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
```

```
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
```

```
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
```

```
            print("cC2")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time4_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time4  = mixed_time4_.overlay(audio3)          # further combine , superi

mixed_time4.export("notes_/mixed_time4.mp3", format='mp3') # export mixed  audi
play(mixed_time4)                                  # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tF#2
fF#2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpn0zxt137.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.27 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

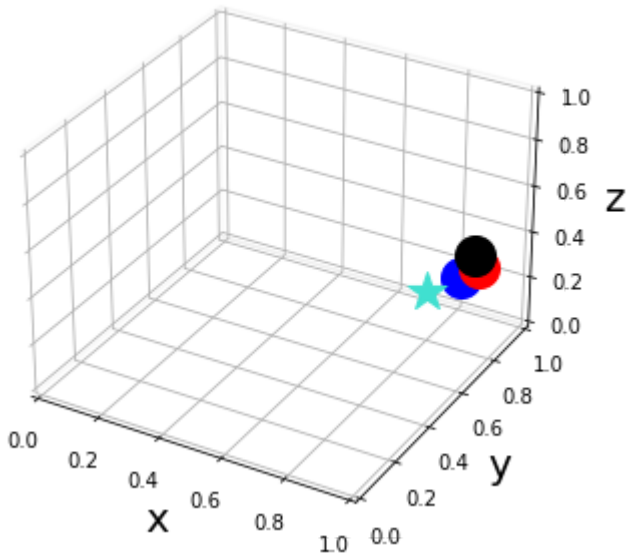
In [45]:

```
# Another round of SOS re-shuffle

# threshold for initial reward
# random fluctuations

if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
    print("SOS")
    # R1
    R1.alphax = round(np.random.uniform(0,0.9), 3)
    R1.betax = round(1 - R1.alphax, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
    R1.alphay = round(np.random.uniform(0,0.9), 3)
    R1.betay = round(1 - R1.alphay, 3)
    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
    R1.alphaz = round(np.random.uniform(0,0.9), 3)
    R1.betaz = round(1 - R1.alphaz, 3)
    print("the new z-positions for R1 are: ", R1.alphaz, R1.betaz)
    # R2
    R2.alphax = round(np.random.uniform(0,0.9), 3)
    R2.betax = round(1 - R2.alphax, 3)
    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
    R2.alphay = round(np.random.uniform(0,0.9), 3)
    R2.betay = round(1 - R2.alphay, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    R2.alphaz = round(np.random.uniform(0,0.9), 3)
    R2.betaz = round(1 - R2.alphaz, 3)
    print("the new y-positions for R2 are: ", R2.alphaz, R1.betaz)
    # R3
    R3.alphax = round(np.random.uniform(0,0.9), 3)
    R3.betax = round(1 - R3.alphax, 3)
    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
    R3.alphay = round(np.random.uniform(0,0.9), 3)
    R3.betay = round(1 - R3.alphay, 3)
    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)
    R3.alphaz = round(np.random.uniform(0,0.9), 3)
    R3.betaz = round(1 - R3.alphaz, 3)
    print("the new z-positions for R3 are: ", R3.alphaz, R1.betaz)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
R1.gamma = 1 - R1.delta
R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
R2.gamma = 1 - R2.delta
R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
R3.gamma = 1 - R3.delta
print(R1.delta, R2.delta, R3.delta)
```

```
0.81 0.83 0.89
```

In [46]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [47]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time5_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time5  = mixed_time5_.overlay(audio3)          # further combine , superi

mixed_time5.export("notes_/mixed_time5.mp3", format='mp3') # export mixed  audi
play(mixed_time5)                               # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tF#2
fF#2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmp7ppu4ofz.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.29 M-A: -0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

We can now attempt to relate class attributes with quantum states. This passage should be automatically changed when class attributes change, in the loop! (while).

Let us suppose that R_1 received a signal from R_2, R_3 with the message: ``Where I am, what I found.'' That is: xy-position and reward information. Then, R_1 chooses to follow the more successful robot that has the more precise position localization.

Before all of that, we use an if: if R_2 already has a high reward, it remains where it is. If we had the same minimization function for all robots, thus, already at the second step all robots would converge toward the same point.

Now: initialization of qubits.

If the robot with the highest reward is R_3, then $R_1 \rightarrow R_3$ and $R_2 \rightarrow R_3$ while entering the gate.
$q[0]$, $q[1]$, $q[2]$ takes positions (x and y) and reward of R_3 in this case. The output with $q[3]$, $q[4]$ ($q[2]$ remains the same) goes to new x, y of R_1 and of R_2.

GATE HERE!! GATE 1

In [48]:

```
1 R1.alphaz
```

Out[48]:

```
0.679
```

In [49]:

```
if (R1.delta > R2.delta) and (R1.delta > R3.delta):
    print("glu glu")
    if (R1.alphax < 0.3): # I have to customize state vectors according to prec
        print("bri")
        qc3.x(q[0])         # just using the NOT gate as a test
    if (R1.alphax == 0.5): # I have to customize state vectors according to pre
        qc3.h(q[0])
    if (R1.alphax >= 0.3) and (R2.alphax < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[0])
    if (R1.alphax >= 0.6) and (R2.alphax < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[0])
    if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
        print("cra cra")
        qc3.x(q[1])
    if (R1.alphay == 0.5): # I have to customize state vectors according to pre
        qc3.h(q[1])
    if (R1.alphay >= 0.3) and (R2.alphay < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[1])
    if (R1.alphay >= 0.6) and (R2.alphay < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[1])
    if (R1.alphaz <= 0.2): # else: the qubit sticks with the default value '0'
        qc3.x(q[2])
        print("augh")
    if (R1.alphaz == 0.5): # I have to customize state vectors according to pre
        print("ouch")
        qc3.h(q[2])
    if (R1.alphaz >= 0.3) and (R2.alphaz < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[2])
    if (R1.alphaz >= 0.6) and (R2.alphaz < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[2])
    if (R1.alphaz >= 0.2 and R1.alphax < 0.3):
        print("wolf")
        qc3.x(q[2])
    #if (R1.alphaz >= 0.7):
    #    print("wolf2") # leave the state as 0, default value
    if (R1.delta == 0.5):
        qc3.h(q[3])
    if (R1.delta == 0.6):
        qc3.h(q[3])
    if (R1.delta >= 0.7):
        qc3.x(q[3])
    if (R1.gamma >= 0.3) and (R2.gamma < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[3])
    if (R1.gamma >= 0.6) and (R2.gamma < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[3])
# February 13: IT WAS SUPPOSED TO BE (R2.delta > R1.delta) and (R2.delta > R3.d
# and R2 rather than R1; correct also in the other file!!! (for x-y)
#elif (R1.delta > R2.delta) and (R1.delta > R3.delta): # February 13: THAT MUST
elif (R2.delta > R1.delta) and (R2.delta > R3.delta): # February 13: THAT MUST
    print('dog')
    if (R2.alphax < 0.3): # I have to customize state vectors according to prec
```

```
        qc3.x(q[0])         # just using the NOT gate as a test
    if (R2.alphax == 0.5): # I have to customize state vectors according to pre
        qc3.h(q[0])
    if (R2.alphax >= 0.3) and (R1.alphax < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[0])
    if (R2.alphax >= 0.6) and (R1.alphax < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[0])
    if (R2.alphay <= 0.2): # else: the qubit sticks with the default value '0'
        qc3.x(q[1])
    if (R2.alphay == 0.5): # I have to customize state vectors according to pre
        qc3.h(q[1])
    if (R2.alphay >= 0.3) and (R1.alphay < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[1])
    if (R2.alphay >= 0.6) and (R1.alphay < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[1])
    if (R2.alphaz <= 0.2): # else: the qubit sticks with the default value '0'
        qc3.x(q[2])
    if (R2.alphay == 0.5): # I have to customize state vectors according to pre
        qc3.h(q[2])
    if (R2.alphaz >= 0.3) and (R1.alphaz < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[2])
    if (R2.alphaz >= 0.6) and (R1.alphaz < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[2])
    if (R1.alphaz >= 0.2 and R1.alphax < 0.3):
        print("wolf")
        qc3.x(q[2])
    if (R2.delta == 0.5):
        qc3.h(q[3])
    if (R2.delta == 0.6):
        qc3.h(q[3])
    if (R2.delta >= 0.7):
        qc3.x(q[3])
    if (R2.gamma >= 0.3) and (R1.gamma < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[3])
    if (R2.gamma >= 0.6) and (R1.gamma < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[3])
else: # February 13: REVISE ALL THIS SECTION!!!!
    print('cat') # I made some tests to check the IF conditions
    if (R3.alphax < 0.3):
        qc3.x(q[0])
    if (R3.alphax == 0.5):
        qc3.h(q[0])
    if (R3.alphax >= 0.3) and (R3.alphax < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[0])
    if (R3.alphax >= 0.6) and (R3.alphax < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[0])
    if (R3.alphay < 0.3):
        qc3.x(q[1])
    if (R3.alphay == 0.5):
        qc3.h(q[1])
    if (R3.alphay >= 0.3) and (R3.alphay < 0.5):
```

```
        print('jungle!')
        qc3.ry(1.9106332, q[1])
    if (R3.alphay >= 0.6) and (R3.alphay < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[1])
    if (R3.alphaz < 0.3):
        qc3.x(q[2])
    if (R3.alphaz == 0.5):
        qc3.h(q[2])
    if (R3.alphaz >= 0.3) and (R3.alphaz < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[2])
    if (R3.alphaz >= 0.6) and (R3.alphaz < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[2])
    if (R3.delta == 0.5):
        qc3.h(q[3])
    if (R3.delta == 0.6):
        qc3.h(q[3])
    if (R3.delta >= 0.7):
        qc3.x(q[3])
    if (R3.gamma >= 0.3) and (R3.gamma < 0.5):
        print('jungle!')
        qc3.ry(1.9106332, q[3])
    if (R3.gamma >= 0.6) and (R3.gamma < 0.7):
        print('ocean!')
        qc3.ry(1.2309594, q[3])
```

cat

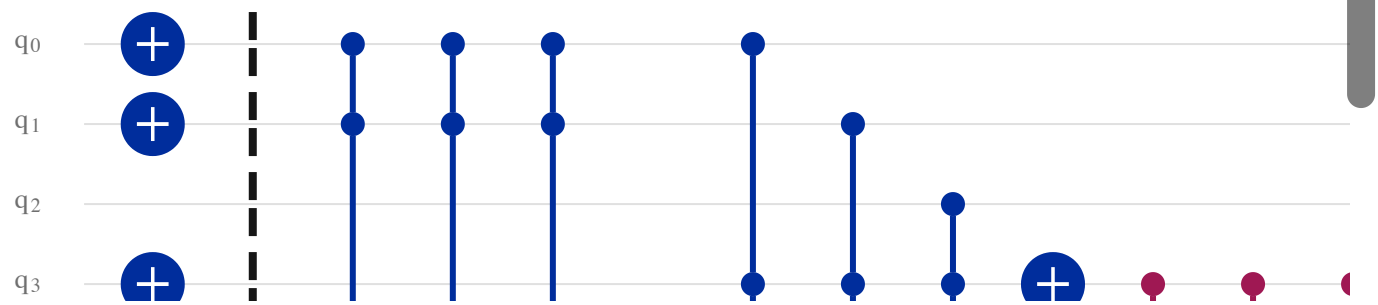
In [50]:

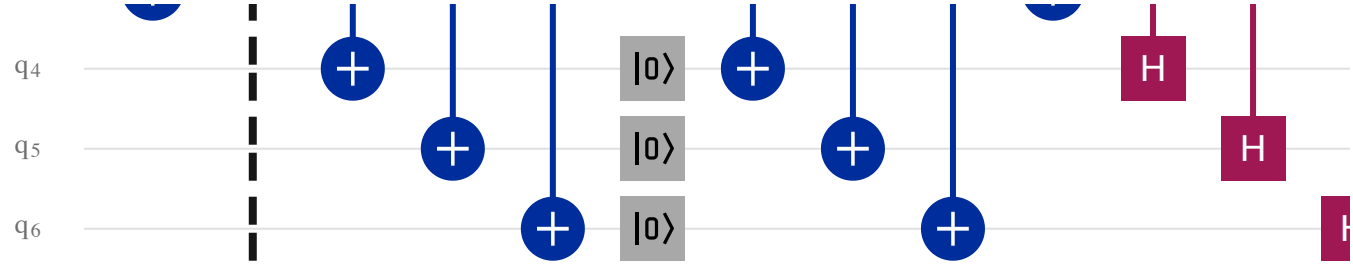
```
# qc3.x(q[1]), qc3.x(q[2]) # just to check
# qc3.h(q[2])
```

Numeration of qubits within IF instructions is slightly different than the initial one. In fact, some distinction across qubits was needed to clearly build the whole circuit later on. Thus, I decided to keep them apart.

In [51]:

```
# this is the core code, and it is unchanged across time

qc3.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6])
qc3.ccx(q[0], q[1], q[4])
qc3.ccx(q[0], q[1], q[5])
qc3.ccx(q[0], q[1], q[6])
qc3.reset(q[4])
qc3.reset(q[5])
qc3.reset(q[6])
qc3.ccx(q[0], q[3], q[4])
qc3.ccx(q[1], q[3], q[5])
qc3.ccx(q[2], q[3], q[6])
qc3.x(q[3])
qc3.ch(q[3], q[4])
qc3.ch(q[3], q[5])
qc3.ch(q[3], q[6])
qc3.x(q[3])
qc3.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6])
qc3.measure(q[3], m3[0])
qc3.measure(q[4], m4[0])
qc3.measure(q[5], m5[0])
qc3.measure(q[6], m6[0])

# visualization of the ciruit

draw_circuit(qc3)

# definition of quantum simulator

simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_simu
qc3 = transpile(qc3, simulator)
cc = collections.Counter()

# Run and get counts
result = simulator.run(qc3, shots=1024).result()
counts = result.get_counts(qc3)
counts2 = counts.most_frequent() # does not work if multiple states have the sam
# decide something if multiple states have the same count --> e.g., ``choose the
counts3 = cc.most_common(2)
print(counts)
print(counts2)
print(counts3)
result = simulator.run(qc3, shots=10, memory=True).result()
memory = result.get_memory(qc3)
print(memory)
plot_histogram(counts, title='outcomes')
# TAKE the TWO more present outcomes
```





In [45]:

```
1 # keep the two more present outcomes.
```

In [52]:

```
print(counts2) # order: R3, R2, R1. Add some uncertainty?
# export as an array
str = counts2
arr1 = str.split(' ') # to split the string and avoid empty spaces as array elem
print(arr1)
weight1 = 1024  # AT HAND ONLY FOR NOW

arr2 = ['0','1','1', '1'] # 111 # 011
print(arr2)
weight2 = 1024
# BY HAND ONLY FOR NOW


# an attempt, not so good, to automatize this passage:

print(memory)

data = Counter(memory)
data.most_common()   # Returns all unique items and their counts
data.most_common(3)

print(data.most_common())
print(data.most_common(1))
arrx1 = data.most_common(2)[0]
print(arrx1)
arrx2 = data.most_common(2)[1]
print(arrx2)


```

```
0 1 1 1
['0', '1', '1', '1']
['0', '1', '1', '1']
['0 1 1 1', '0 1 1 1', '0 1 1 1', '0 1 1 1', '0 1 1 1', '0 1 1 1', '0 1 1 1', '0 1
1 1', '0 1 1 1', '0 1 1 1']
[('0 1 1 1', 10)]
[('0 1 1 1', 10)]
('0 1 1 1', 10)

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1974/4039962.py in <mod
ule>
     24 arrx1 = data.most_common(2)[0]
     25 print(arrx1)
---> 26 arrx2 = data.most_common(2)[1]
     27 print(arrx2)
     28

IndexError: list index out of range
```

In [53]:

```
# array 1
arr1
```

Out[53]:

```
['0', '1', '1', '1']
```

In [54]:

```
# array 2
arr2
```

Out[54]:

```
['0', '1', '1', '1']
```

Let us create a sort of weighted sum.

It not convenient to set up q[3], q[4], because we need coordinates attribution....

Now, we re-calculate the positions of the robots that entered the gate. To this aim, use their reward (which is unchanged yet).

In [65]:

```
# February 13: ADD THE z-COMPONENT IN WHAT FOLLOWS
```

Position for R_1:

In [56]:

```
if (R1.delta > R2.delta) and (R1.delta > R3.delta):
    # if R1 didn't enter the gate, keep its position
    R1.alphax = R1.alphax
    R1.betax = R1.betax
    R1.alphay = R1.alphay
    R1.betay = R1.betay
    R1.alphaz = R1.alphaz
    R1.betaz = R1.betaz
else:
change of January 28: I'm substituting [0] with [1] and vice versa, because the outp
February 15: in the case with xyz, we have [2] first
    if (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 50 and (we
        print("bla")
        R1.alphax = 0.3
        R1.betax = 0.7
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 50 and (
        print("gulp")
        R1.alphax = 0.7
        R1.betax = 0.3
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and (weight1 == weight2 or np.absol
        print("stra-gulp")
        R1.alphax = 0.5 # change temporarily made on January 24: random generator rath
        R1.betax = 0.5 # same as above
    elif (arr1[2] == arr2[0]) and (arr1[2] == '1') and ((weight2 - weight1) > 800): #
        print("thunderstorm!")
        R1.alphax = 0
        R1.betax = 1
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 200 and
        print("avalanche!")
        R1.alphax = 0.1
        R1.betax = 0.9
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 200 and
        print("earthquake!")
        R1.alphax = 0.9
        R1.betax = 0.1
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 200 and
        print("avalanche bis!")
        R1.alphax = 0.1 # the same also in this case
        R1.betax = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight1 - weight2) > 50): # n
        print("bla 2")
        R1.alphax = 0.7  # the opposite??
        R1.betax = 0.3  # the opposite??
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight2 - weight1) > 50): # n
        print("gulp 2")
        R1.alphax = 0.3  # the opposite??
        R1.betax = 0.7  # the opposite??
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and (weight1 == weight2 or np.absol
        print("stra-gulp 2")
        R1.alphax = 0.5 # change temporarily made on January 24: random generator rath
        R1.betax = 0.5 # change temporarily made on January 24: random generator rathe
    # different outcomes
    elif arr1[2] != arr2[2]:  # January 23
        print("blue")
        if (arr1[2] != arr2[2]) and (weight1 == weight2 or np.absolute(weight1 - weigh
            print("google 1")
            R1.alphax = 0.5 # change temporarily made on January 24: random generator
            R1.betax = 0.5 # change temporarily made on January 24: random generator r
```

```
 60  if (arr1[2] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.absolute(
 61      # include the case of a very small difference!
 62      print("uffdah")
 63      R1.alphax = 0.5
 64      R1.betax = 0.5
 65  if (arr1[2] == '1' and arr2[1] == '0'):
 66      if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
 67          print("abc")
 68          R1.alphax = 0.3
 69          R1.betax = 0.7
 70      if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
 71          print("bca")
 72          R1.alphax = 0.7
 73          R1.betax = 0.3
 74      if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
 75          print("news")
 76          R1.alphax = 0.2 #
 77          R1.betax = 0.8 #
 78      if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
 79          print("idea")
 80          R1.alphax = 0.8 #
 81          R1.betax = 0.2 #
 82      if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no abs #
 83          print("news")
 84          R1.alphax = 0.1 #
 85          R1.betax = 0.9 #
 86      if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no abs #
 87          print("idea")
 88          R1.alphax = 0.9 #
 89          R1.betax = 0.1 #
 90  if (arr1[2] == '0') and (arr2[2] == '1'):
 91      if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
 92          print("bac")
 93          R1.alphax = 0.7
 94          R1.betax = 0.3
 95      if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
 96          print("cba")
 97          R1.alphax = 0.3
 98          R1.betax = 0.7
 99      if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
100          print("brain")
101          R1.alphax = 0.7 # 0.9
102          R1.betax = 0.3 # 0.1
103      if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
104          print("hand")
105          R1.alphax = 0.3 # 0.1
106          R1.betax = 0.7 # 0.9
107      if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no abs #
108          print("brain2")
109          R1.alphax = 0.9 # 0.9
110          R1.betax = 0.1 # 0.1
111      if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no abs #
112          print("hand2")
113          R1.alphax = 0.1 # 0.1
114          R1.betax = 0.9 # 0.9
115  # y part
116
117  # change of January 26
118
119
120  if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50 and (we
```

```
    print("bla")
    R1.alphay = 0.3
    R1.betay = 0.7
elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 50 and (
    print("gulp")
    R1.alphay = 0.7
    R1.betay = 0.3
elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or np.absol
    print("stra-gulp")
    R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
    R1.betay = 0.5 # same as above
elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 800): #
    print("thunderstorm!")
    R1.alphay = 0
    R1.betay = 1 # note of February 15: attention there: [1] rather than [0]
elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 200 and
    print("avalanche!")
    R1.alphay = 0.1
    R1.betay = 0.9
elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 200 and
    print("earthquake!")
    R1.alphay = 0.9
    R1.betay = 0.1
elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 200 and
    print("avalanche bis!")
    R1.alphay = 0.1 # the same also in this case
    R1.betay = 0.9 # the same also in this case
# same = outcome 0  # January 23
elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 50): # n
    print("bla 2")
    R1.alphay = 0.7  # the opposite??
    R1.betay = 0.3  # the opposite??
elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 50): # n
    print("gulp 2")
    R1.alphay = 0.3  # the opposite??
    R1.betay = 0.7  # the opposite??
elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or np.absol
    print("stra-gulp 2")
    R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
    R1.betay = 0.5 # change temporarily made on January 24: random generator rathe
# different outcomes
elif arr1[1] != arr2[1]:  # January 23
    print("blue")
    if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1 - weigh
        print("google 1")
        R1.alphay = 0.5 # change temporarily made on January 24: random generator
        R1.betay = 0.5 # change temporarily made on January 24: random generator r
    if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.absolute(
        # include the case of a very small difference!
        print("uffdah")
        R1.alphay = 0.5
        R1.betay = 0.5
    if (arr1[1] == '1' and arr2[1] == '0'):
        if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
            print("abc")
            R1.alphay = 0.3
            R1.betay = 0.7
        if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
            print("bca")
            R1.alphay = 0.7
            R1.betay = 0.3
```

```
        if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
            print("news")
            R1.alphay = 0.2 #
            R1.betay = 0.8 #
        if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
            print("idea")
            R1.alphay = 0.8 #
            R1.betay = 0.2 #
    if (arr1[1] == '0') and (arr2[1] == '1'):
        if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
            print("bac")
            R1.alphay = 0.7
            R1.betay = 0.3
        if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
            print("cba")
            R1.alphay = 0.3
            R1.betay = 0.7
        if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
            print("brain")
            R1.alphay = 0.7 # 0.9
            R1.betay = 0.3 # 0.1
        if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
            print("hand")
            R1.alphay = 0.3 # 0.1
            R1.betay = 0.7 # 0.9
        if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no abs #
            print("brain2")
            R1.alphay = 0.9 # 0.9
            R1.betay = 0.1 # 0.1
        if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no abs #
            print("hand2")
            R1.alphay = 0.1 # 0.1
            R1.betay = 0.9 # 0.9


# part

# change of February 15

if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50 and (we
    print("bla")
    R1.alphay = 0.3
    R1.betay = 0.7
elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 50 and (
    print("gulp")
    R1.alphay = 0.7
    R1.betay = 0.3
elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or np.absol
    print("stra-gulp")
    R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
    R1.betay = 0.5 # same as above
elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 800): #
    print("thunderstorm!")
    R1.alphay = 0
    R1.betay = 1 # note of February 15: attention there: [1] rather than [0]
elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 200 and
    print("avalanche!")
    R1.alphay = 0.1
    R1.betay = 0.9
elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 200 and
    print("earthquake!")
```

```
 R1.alphay = 0.9
 R1.betay = 0.1
elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 200 and
 print("avalanche bis!")
 R1.alphay = 0.1 # the same also in this case
 R1.betay = 0.9 # the same also in this case
# same = outcome 0  # January 23
elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 50): # n
 print("bla 2")
 R1.alphay = 0.7  # the opposite??
 R1.betay = 0.3  # the opposite??
elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 50): # n
 print("gulp 2")
 R1.alphay = 0.3  # the opposite??
 R1.betay = 0.7  # the opposite??
elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.absol
 print("stra-gulp 2")
 R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
 R1.betay = 0.5 # change temporarily made on January 24: random generator rathe
# different outcomes
elif arr1[0] != arr2[0]:  # January 23
 print("blue")
 if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1 - weigh
     print("google 1")
     R1.alphay = 0.5 # change temporarily made on January 24: random generator
     R1.betay = 0.5 # change temporarily made on January 24: random generator r
 if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.absolute(
     # include the case of a very small difference!
     print("uffdah")
     R1.alphay = 0.5
     R1.betay = 0.5
 if (arr1[0] == '1' and arr2[0] == '0'):
     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
         print("abc")
         R1.alphay = 0.3
         R1.betay = 0.7
     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
         print("bca")
         R1.alphay = 0.7
         R1.betay = 0.3
     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
         print("news")
         R1.alphay = 0.2 #
         R1.betay = 0.8 #
     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
         print("idea")
         R1.alphay = 0.8 #
         R1.betay = 0.2 #
 if (arr1[0] == '0') and (arr2[0] == '1'):
     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
         print("bac")
         R1.alphay = 0.7
         R1.betay = 0.3
     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
         print("cba")
         R1.alphay = 0.3
         R1.betay = 0.7
     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
         print("brain")
         R1.alphay = 0.7 # 0.9
         R1.betay = 0.3 # 0.1
```

```
    if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
        print("hand")
        R1.alphay = 0.3 # 0.1
        R1.betay = 0.7 # 0.9
    if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no abs #
        print("brain2")
        R1.alphay = 0.9 # 0.9
        R1.betay = 0.1 # 0.1
    if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no abs #
        print("hand2")
        R1.alphay = 0.1 # 0.1
        R1.betay = 0.9 # 0.9
```

```
stra-gulp
stra-gulp
stra-gulp 2
```

Position for R_2

In [57]:

```
if (R2.delta > R1.delta) and (R2.delta > R3.delta):
    # if R2 didn't entered the gate, keep its position
    R2.alphax = R2.alphax
    R2.betax = R2.betax
    R2.alphay = R2.alphay
    R2.betay = R2.betay
    R2.alphaz = R2.alphaz
    R2.betaz = R2.betaz
else:
# change of January 28: I'm substituting [0] with [1] and vice versa, because t
# February 15: in the case with xyz, we have [2] first
    if (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R2.alphax = 0.3
        R2.betax = 0.7
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R2.alphax = 0.7
        R2.betax = 0.3
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R2.alphax = 0.5 # change temporarily made on January 24: random generat
        R2.betax = 0.5 # same as above
    elif (arr1[2] == arr2[0]) and (arr1[2] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R2.alphax = 0
        R2.betax = 1
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R2.alphax = 0.1
        R2.betax = 0.9
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R2.alphax = 0.9
        R2.betax = 0.1
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphax = 0.1 # the same also in this case
        R2.betax = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphax = 0.7  # the opposite??
        R2.betax = 0.3  # the opposite??
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphax = 0.3  # the opposite??
        R2.betax = 0.7  # the opposite??
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphax = 0.5 # change temporarily made on January 24: random generat
        R2.betax = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[2] != arr2[2]:  # January 23
        print("blue")
        if (arr1[2] != arr2[2]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphax = 0.5 # change temporarily made on January 24: random gen
            R2.betax = 0.5 # change temporarily made on January 24: random gene
```

```
        if (arr1[2] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R2.alphax = 0.5
            R2.betax = 0.5
        if (arr1[2] == '1' and arr2[1] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphax = 0.3
                R2.betax = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphax = 0.7
                R2.betax = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphax = 0.2 #
                R2.betax = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphax = 0.8 #
                R2.betax = 0.2 #
            if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
                print("news")
                R2.alphax = 0.1 #
                R2.betax = 0.9 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("idea")
                R2.alphax = 0.9 #
                R2.betax = 0.1 #
        if (arr1[2] == '0') and (arr2[2] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphax = 0.7
                R2.betax = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphax = 0.3
                R2.betax = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R2.alphax = 0.7 # 0.9
                R2.betax = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R2.alphax = 0.3 # 0.1
                R2.betax = 0.7 # 0.9
            if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
                print("brain2")
                R2.alphax = 0.9 # 0.9
                R2.betax = 0.1 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("hand2")
                R2.alphax = 0.1 # 0.1
                R2.betax = 0.9 # 0.9
    # y part

    # change of January 26


    if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
```

```
        print("bla")
        R2.alphay = 0.3
        R2.betay = 0.7
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R2.alphay = 0.7
        R2.betay = 0.3
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R2.alphay = 0.5 # change temporarily made on January 24: random generat
        R2.betay = 0.5 # same as above
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R2.alphay = 0
        R2.betay = 1 # note of February 15: attention there: [1] rather than [0
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R2.alphay = 0.1
        R2.betay = 0.9
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R2.alphay = 0.9
        R2.betay = 0.1
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphay = 0.1 # the same also in this case
        R2.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphay = 0.7  # the opposite??
        R2.betay = 0.3  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphay = 0.3  # the opposite??
        R2.betay = 0.7  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphay = 0.5 # change temporarily made on January 24: random generat
        R2.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[1] != arr2[1]:  # January 23
        print("blue")
        if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphay = 0.5 # change temporarily made on January 24: random gen
            R2.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R2.alphay = 0.5
            R2.betay = 0.5
        if (arr1[1] == '1' and arr2[1] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphay = 0.3
                R2.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphay = 0.7
                R2.betay = 0.3
```

```
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphay = 0.2 #
                R2.betay = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphay = 0.8 #
                R2.betay = 0.2 #
        if (arr1[1] == '0') and (arr2[1] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphay = 0.7
                R2.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphay = 0.3
                R2.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R2.alphay = 0.7 # 0.9
                R2.betay = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R2.alphay = 0.3 # 0.1
                R2.betay = 0.7 # 0.9
            if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
                print("brain2")
                R2.alphay = 0.9 # 0.9
                R2.betay = 0.1 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("hand2")
                R2.alphay = 0.1 # 0.1
                R2.betay = 0.9 # 0.9


# z part

    # change of February 15

    if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R2.alphay = 0.3
        R2.betay = 0.7
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R2.alphay = 0.7
        R2.betay = 0.3
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R2.alphay = 0.5 # change temporarily made on January 24: random generat
        R2.betay = 0.5 # same as above
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R2.alphay = 0
        R2.betay = 1 # note of February 15: attention there: [1] rather than [0
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R2.alphay = 0.1
        R2.betay = 0.9
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
```

```
        R2.alphay = 0.9
        R2.betay = 0.1
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphay = 0.1 # the same also in this case
        R2.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphay = 0.7  # the opposite??
        R2.betay = 0.3  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphay = 0.3  # the opposite??
        R2.betay = 0.7  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphay = 0.5 # change temporarily made on January 24: random generat
        R2.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[0] != arr2[0]:  # January 23
        print("blue")
        if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphay = 0.5 # change temporarily made on January 24: random gen
            R2.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R2.alphay = 0.5
            R2.betay = 0.5
        if (arr1[0] == '1' and arr2[0] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphay = 0.3
                R2.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphay = 0.7
                R2.betay = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphay = 0.2 #
                R2.betay = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphay = 0.8 #
                R2.betay = 0.2 #
        if (arr1[0] == '0') and (arr2[0] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphay = 0.7
                R2.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphay = 0.3
                R2.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R2.alphay = 0.7 # 0.9
                R2.betay = 0.3 # 0.1
```

```
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R2.alphay = 0.3 # 0.1
                R2.betay = 0.7 # 0.9
            if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
                print("brain2")
                R2.alphay = 0.9 # 0.9
                R2.betay = 0.1 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("hand2")
                R2.alphay = 0.1 # 0.1
                R2.betay = 0.9 # 0.9
```

```
stra-gulp
stra-gulp
stra-gulp 2
```

Position for R_3

In [60]:

```
if (R3.delta > R1.delta) and (R3.delta > R2.delta):
    # if R3 didn't entered the gate, keep its position
    print("achtung!")
    R3.alphax = R3.alphax
    R3.betax = R3.betax
    R3.alphay = R3.alphay
    R3.betay = R3.betay
    R3.alphaz = R3.alphaz
    R3.betaz = R3.betaz
else:
# change of January 28: I'm substituting [0] with [1] and vice versa, because t
# February 15: in the case with xyz, we have [2] first
    if (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R3.alphax = 0.3
        R3.betax = 0.7
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R3.alphax = 0.7
        R3.betax = 0.3
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R3.alphax = 0.5 # change temporarily made on January 24: random generat
        R3.betax = 0.5 # same as above
    elif (arr1[2] == arr2[0]) and (arr1[2] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R3.alphax = 0
        R3.betax = 1
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R3.alphax = 0.1
        R3.betax = 0.9
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R3.alphax = 0.9
        R3.betax = 0.1
    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R3.alphax = 0.1 # the same also in this case
        R3.betax = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R3.alphax = 0.7  # the opposite??
        R3.betax = 0.3  # the opposite??
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R3.alphax = 0.3  # the opposite??
        R3.betax = 0.7  # the opposite??
    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R3.alphax = 0.5 # change temporarily made on January 24: random generat
        R3.betax = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[2] != arr2[2]:  # January 23
        print("blue")
        if (arr1[2] != arr2[2]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R3.alphax = 0.5 # change temporarily made on January 24: random gen
```

```
            R3.betax = 0.5 # change temporarily made on January 24: random gene
        if (arr1[2] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R3.alphax = 0.5
            R3.betax = 0.5
        if (arr1[2] == '1' and arr2[1] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R3.alphax = 0.3
                R3.betax = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R3.alphax = 0.7
                R3.betax = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R3.alphax = 0.2 #
                R3.betax = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R3.alphax = 0.8 #
                R3.betax = 0.2 #
            if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
                print("news")
                R3.alphax = 0.1 #
                R3.betax = 0.9 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("idea")
                R3.alphax = 0.9 #
                R3.betax = 0.1 #
        if (arr1[2] == '0') and (arr2[2] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R3.alphax = 0.7
                R3.betax = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R3.alphax = 0.3
                R3.betax = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R3.alphax = 0.7 # 0.9
                R3.betax = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R3.alphax = 0.3 # 0.1
                R3.betax = 0.7 # 0.9
            if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
                print("brain2")
                R3.alphax = 0.9 # 0.9
                R3.betax = 0.1 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("hand2")
                R3.alphax = 0.1 # 0.1
                R3.betax = 0.9 # 0.9
    # y part

    # change of January 26


```

```
    if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R3.alphay = 0.3
        R3.betay = 0.7
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R3.alphay = 0.7
        R3.betay = 0.3
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R3.alphay = 0.5 # change temporarily made on January 24: random generat
        R3.betay = 0.5 # same as above
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R3.alphay = 0
        R3.betay = 1 # note of February 15: attention there: [1] rather than [0
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R3.alphay = 0.1
        R3.betay = 0.9
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R3.alphay = 0.9
        R3.betay = 0.1
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R3.alphay = 0.1 # the same also in this case
        R3.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R3.alphay = 0.7  # the opposite??
        R3.betay = 0.3  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R3.alphay = 0.3  # the opposite??
        R3.betay = 0.7  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R3.alphay = 0.5 # change temporarily made on January 24: random generat
        R3.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[1] != arr2[1]:  # January 23
        print("blue")
        if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R3.alphay = 0.5 # change temporarily made on January 24: random gen
            R3.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R3.alphay = 0.5
            R3.betay = 0.5
        if (arr1[1] == '1' and arr2[1] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R3.alphay = 0.3
                R3.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R3.alphay = 0.7
```

```
                R3.betay = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R3.alphay = 0.2 #
                R3.betay = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R3.alphay = 0.8 #
                R3.betay = 0.2 #
        if (arr1[1] == '0') and (arr2[1] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R3.alphay = 0.7
                R3.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R3.alphay = 0.3
                R3.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R3.alphay = 0.7 # 0.9
                R3.betay = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R3.alphay = 0.3 # 0.1
                R3.betay = 0.7 # 0.9
            if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
                print("brain2")
                R3.alphay = 0.9 # 0.9
                R3.betay = 0.1 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("hand2")
                R3.alphay = 0.1 # 0.1
                R3.betay = 0.9 # 0.9


# z part

    # change of February 15

    if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R3.alphay = 0.3
        R3.betay = 0.7
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R3.alphay = 0.7
        R3.betay = 0.3
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R3.alphay = 0.5 # change temporarily made on January 24: random generat
        R3.betay = 0.5 # same as above
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R3.alphay = 0
        R3.betay = 1 # note of February 15: attention there: [1] rather than [0
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R3.alphay = 0.1
        R3.betay = 0.9
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
```

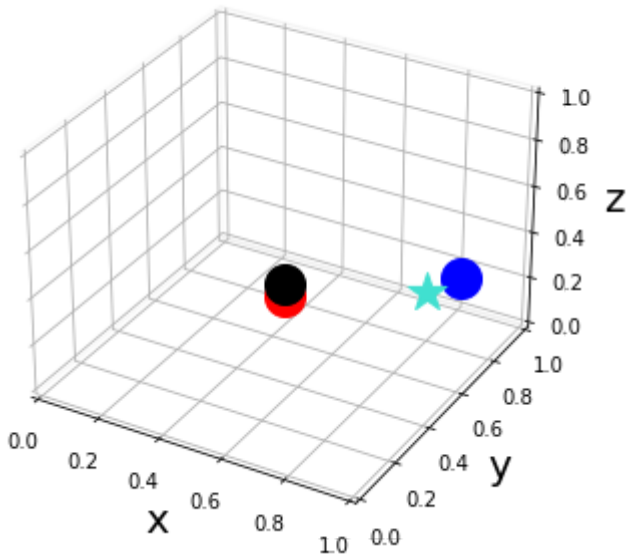
```
        print("earthquake!")
        R3.alphay = 0.9
        R3.betay = 0.1
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R3.alphay = 0.1 # the same also in this case
        R3.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R3.alphay = 0.7  # the opposite??
        R3.betay = 0.3  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R3.alphay = 0.3  # the opposite??
        R3.betay = 0.7  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R3.alphay = 0.5 # change temporarily made on January 24: random generat
        R3.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[0] != arr2[0]:  # January 23
        print("blue")
        if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R3.alphay = 0.5 # change temporarily made on January 24: random gen
            R3.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R3.alphay = 0.5
            R3.betay = 0.5
        if (arr1[0] == '1' and arr2[0] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R3.alphay = 0.3
                R3.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R3.alphay = 0.7
                R3.betay = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R3.alphay = 0.2 #
                R3.betay = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R3.alphay = 0.8 #
                R3.betay = 0.2 #
        if (arr1[0] == '0') and (arr2[0] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R3.alphay = 0.7
                R3.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R3.alphay = 0.3
                R3.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R3.alphay = 0.7 # 0.9
```

```
                R3.betay = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R3.alphay = 0.3 # 0.1
                R3.betay = 0.7 # 0.9
            if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
                print("brain2")
                R3.alphay = 0.9 # 0.9
                R3.betay = 0.1 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
                print("hand2")
                R3.alphay = 0.1 # 0.1
                R3.betay = 0.9 # 0.9
```

```
achtung!
```

In [61]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [62]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time6_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time6  = mixed_time6_.overlay(audio3)          # further combine , superi

mixed_time6.export("notes_/mixed_time6.mp3", format='mp3') # export mixed  audi
play(mixed_time6)                                # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tA2
fA2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmph_0vg3jx.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.23 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
   7.31 M-A:  0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

And only NOW, change their rewards according to the new positions! If a robot didn't change the position, the reward will remain the same.

In [63]:

```
# the former ones

R1.delta, R2.delta, R3.delta
```

Out[63]:

```
(0.81, 0.83, 0.89)
```

In [64]:

```
# the new ones

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
print(R3.delta)
```

```
0.56
0.57
0.89
```

In [ ]:

```

```

In [65]:

```
# January 28: SOS with a higher threshold
```

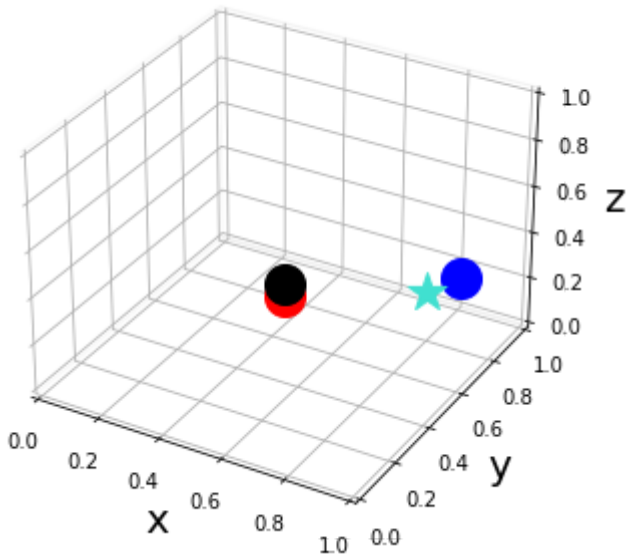
In [66]:

```
# Another round of SOS with a higher threshold. Added on January 28

# threshold for initial reward
# random fluctuations

if (R1.delta <= 0.6) and (R2.delta <= 0.6) and (R3.delta <= 0.6):
    print("SOS")
    # R1
    R1.alphax = round(np.random.uniform(0,0.9), 3)
    R1.betax = round(1 - R1.alphax, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
    R1.alphay = round(np.random.uniform(0,0.9), 3)
    R1.betay = round(1 - R1.alphay, 3)
    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
    R1.alphaz = round(np.random.uniform(0,0.9), 3)
    R1.betaz = round(1 - R1.alphaz, 3)
    print("the new y-positions for R1 are: ", R1.alphaz, R1.betaz)
    # R2
    R2.alphax = round(np.random.uniform(0,0.9), 3)
    R2.betax = round(1 - R2.alphax, 3)
    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
    R2.alphay = round(np.random.uniform(0,0.9), 3)
    R2.betay = round(1 - R2.alphay, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    R2.alphaz = round(np.random.uniform(0,0.9), 3)
    R2.betaz = round(1 - R2.alphaz, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    # R3
    R3.alphax = round(np.random.uniform(0,0.9), 3)
    R3.betax = round(1 - R3.alphax, 3)
    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
    R3.alphay = round(np.random.uniform(0,0.9), 3)
    R3.betay = round(1 - R3.alphay, 3)
    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)
    R3.alphaz = round(np.random.uniform(0,0.9), 3)
    R3.betaz = round(1 - R3.alphaz, 3)
    print("the new y-positions for R3 are: ", R3.alphaz, R1.betaz)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
R1.gamma = 1 - R1.delta
R2.delta = reward(T, R2.betax, R2.betay, R1.betaz)
R2.gamma = 1 - R2.delta
R3.delta = reward(T, R3.betax, R3.betay, R1.betaz)
R3.gamma = 1 - R3.delta
print(R1.delta, R2.delta, R3.delta, R1.betaz)
```

```
0.56 0.56 0.84 0.321
```

In [67]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [68]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time7_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time7  = mixed_time7_.overlay(audio3)          # further combine , superi

mixed_time7.export("notes_/mixed_time7.mp3", format='mp3') # export mixed  audi
play(mixed_time7)                               # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tA2
fA2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpwyjsd00p.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.28 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

In [77]:

```
1 # January 22, 2022
```

In [ ]:

```
1
```

NEW LINES of code: IF the initial reward is very high (greater than 0.8) for at least one of the three robots ("or"), THEN the other robots have to just reach it (with a pretty small fluctuation), without entering the circuit.

In [69]:

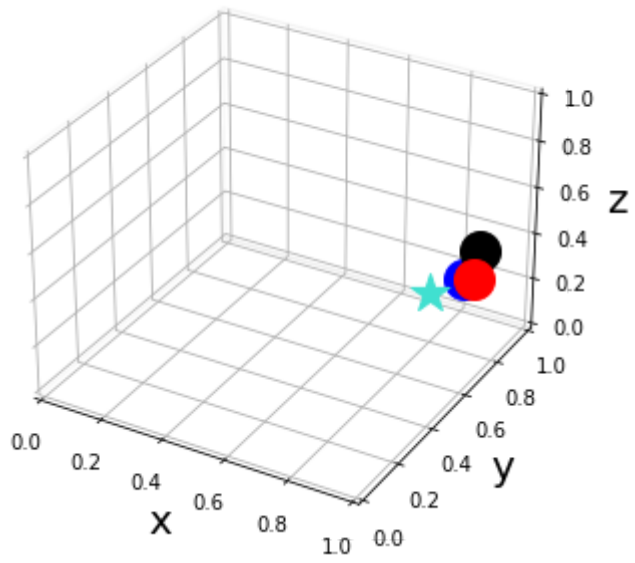
```
if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
    print('yuk')
    if (R1.delta > R2.delta and R1.delta > R3.delta):
        print('quokka')
        R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and late
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)
        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R2.betax, 3)
        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R2.betay, 3)
        R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R2.betaz, 3)
    if (R2.delta > R1.delta and R2.delta > R3.delta):
        print('quagga')
        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R3.betax, 3)
        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R3.betay, 3)
        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R3.betaz, 3)
    if (R3.delta > R1.delta and R3.delta > R2.delta):
        print('quark')
        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
print(R2.delta)
```

```
yuk
quark
0.79
```

```
0.87
0.87
```

In [70]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



Now, all robots reach the robot with the highest reward, with fluctuations:

In [71]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

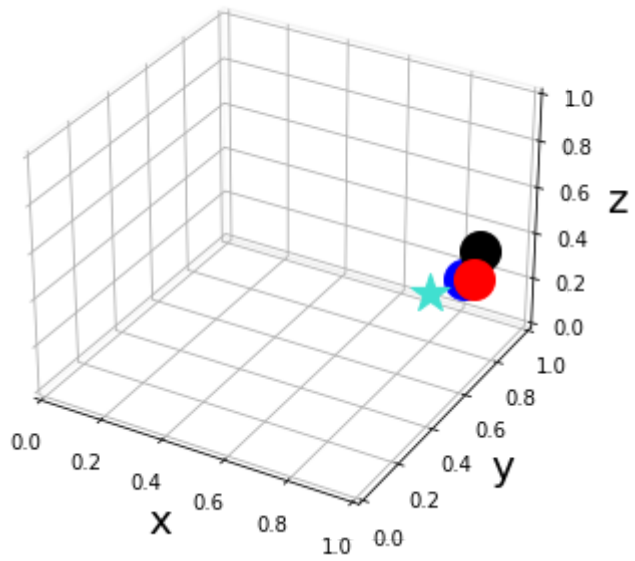
```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time8_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time8  = mixed_time8_.overlay(audio3)          # further combine , superi

mixed_time8.export("notes_/mixed_time8.mp3", format='mp3') # export mixed  audi
play(mixed_time8)                               # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tF#2
fF#2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpsmr6x88x.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.19 M-A: -0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
7.30 M-A: -0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

In [72]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [73]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time9_ = audio1.overlay(audio2)          # combine , superimpose audio fi
mixed_time9  = mixed_time9_.overlay(audio3)          # further combine , superi

mixed_time9.export("notes_/mixed_time9.mp3", format='mp3') # export mixed  audi
play(mixed_time9)                                # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tF#2
fF#2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmp0i8tfbi9.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.30 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

Let us now update $\gamma,\ \delta_i,\ i = 1, 2, 3$ according to the target (fixed) positions and the new positions.

New reward amplitude probabilities for R_1:

In [77]:

```
R1.delta = reward(T,R1.betax,R1.betay, R1.betaz)
R1.gamma = round((1 - R1.delta),3)
print(R1.delta)
```

```
0.79
```

New reward amplitude probabilities for R_2:

In [78]:

```
R2.delta = reward(T,R2.betax,R2.betay, R2.betaz)
R2.gamma = round((1 - R2.delta),3)
print(R2.delta)
```

```
0.87
```

New reward amplitude probabilities for R_3:

In [79]:

```
R3.delta = reward(T,R3.betax,R3.betay,R3.betaz)
R3.gamma = round((1 - R3.delta),3)
print(R3.delta)
```

```
0.89
```

In [86]:

```
# January 22, 2022
```

NEW LINES of code: IF the initial reward is very high (greater than 0.8) for at least one of the three robots ("or"), THEN the other robots have to just reach it (with a pretty small fluctuation), without entering the circuit.

In [80]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

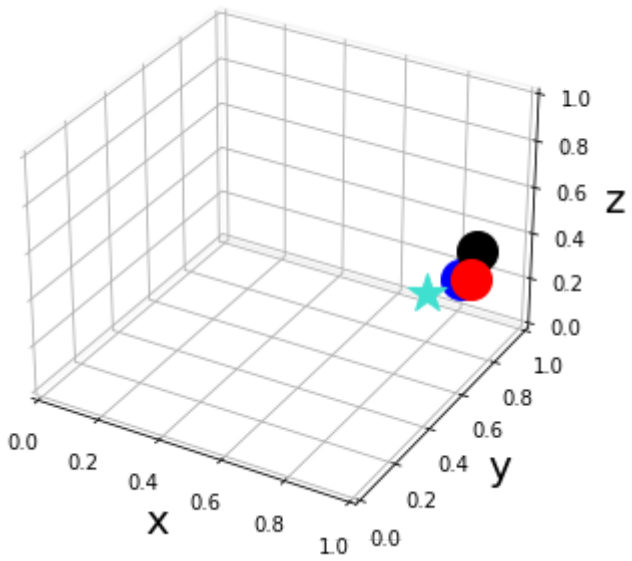
```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time10_ = audio1.overlay(audio2)          # combine , superimpose audio f
mixed_time10  = mixed_time10_.overlay(audio3)          # further combine , supe

mixed_time10.export("notes_/mixed_time10.mp3", format='mp3') # export mixed  au
play(mixed_time10)                                  # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tF#2
fF#2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpnitmw_az.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.20 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
7.27 M-A:  0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

In [81]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [82]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time11_ = audio1.overlay(audio2)          # combine , superimpose audio f
mixed_time11  = mixed_time11_.overlay(audio3)          # further combine , supe

mixed_time11.export("notes_/mixed_time11.mp3", format='mp3') # export mixed  au
play(mixed_time11)                                  # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tF#2
fF#2
cF#2
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpvadh4zld.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.20 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
7.31 M-A: -0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

Let us now try to use entanglement, teleportation, or what is needed, to somehow `glue' together two or more robots which are pretty close to the target.

In [91]:

```
# to be improved: probabilistic representation of positions for more position-ur
# Python probability plot???
```

When we measure the position of R_1 and we get 1, 1, also R_2 are R_3 in 1, 1. If we measure and we get 0, 0, also R_2, R_3 are in 0, 0. In the following code lines, I separated x, y for clarity, but the idea is the same. In this way, we create an entangled GHZ state $\frac{1}{\sqrt{2}}(|11111\rangle + |00000\rangle)$, where the qubits indicate x- and y-positions. Reward is not included in this discussion, because this section is activated only if all robots present almost the same reward (here, pairwise difference ≤ 0.1).

In [92]:

```
# a new circuit
q = QuantumRegister(9, 'q'); # qubits # now 12?
#c0 = ClassicalRegister(6, 'c0');
c0 = ClassicalRegister(1, 'c0');
c1 = ClassicalRegister(1, 'c1');
c3 = ClassicalRegister(1, 'c3');
c4 = ClassicalRegister(1, 'c4');
c6 = ClassicalRegister(1, 'c6');
c7 = ClassicalRegister(1, 'c7');
qc_small = QuantumCircuit(q, c0, c1, c3, c4, c6, c7);
```

In [93]:

```
if ((R3.delta - R2.delta) <= 0.3) and ((R3.delta - R1.delta) <= 0.3) and ((R2.de
    # 0.3 rather than 0.1
    print("cometa")# GHZ for all
    qc_small.h(q[0]) # Hadamard
    qc_small.cx(q[0], q[1]) # CNOT
    qc_small.cx(q[0], q[2]) # CNOT
    qc_small.cx(q[0], q[3]) # CNOT
    qc_small.cx(q[0], q[4]) # CNOT
    qc_small.cx(q[0], q[5]) # CNOT
    qc_small.cx(q[0], q[6]) # CNOT
    qc_small.cx(q[0], q[7]) # CNOT
    # barrier
    qc_small.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6], q[7])
    # measures
    qc_small.measure(q[0],c0[0])
    qc_small.measure(q[3],c3[0])
    qc_small.measure(q[6],c6[0])
    qc_small.measure(q[1],c1[0])
    qc_small.measure(q[4],c4[0])
    qc_small.measure(q[7],c7[0])
    # draw circuit
    draw_circuit(qc_small)
    # definition of quantum simulator
    simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_
    qc_small = transpile(qc_small, simulator)
    # Run and get counts
    result = simulator.run(qc_small, shots=1024).result()
    counts_GHZ_all = result.get_counts(qc_small)
    counts_GHZ = counts_GHZ_all.most_frequent() # does not work if multiple stat
    # decide something if multiple states have the same count --> e.g., ``choose
    print(counts_GHZ_all)
    print(counts_GHZ)
    #plot_histogram(counts_GHZ_all, title='outcomes')
    #plot_histogram(counts_GHZ, title='outcomes')
```

In [94]:

```
print(counts_GHZ) # order: R3, R2, R1. Add some uncertainty?
# export as an array
str_ = counts_GHZ
arr_GHZ = str_.split(' ') # to split the string and avoid empty spaces as array
print(arr_GHZ)
# We do not need to update rewards; they should be done externally... excluded s
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1965/3014944113.py in <
module>
----> 1 print(counts_GHZ) # order: R3, R2, R1. Add some uncertainty?
      2 # export as an array
      3 str_ = counts_GHZ
      4 arr_GHZ = str_.split(' ') # to split the string and avoid empty spaces as
 array elements
      5 print(arr_GHZ)

NameError: name 'counts_GHZ' is not defined
```

Define the 'new 0':

In [95]:

```
if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
    print('gosh')
    new_zero_betax = R1.betax + np.random.uniform(0,0.1)
    new_zero_alphax = 1 - R1.betax
    new_zero_betay = R1.betay + np.random.uniform(0,0.1)
    new_zero_alphay = 1 - R2.betay + np.random.uniform(0,0.1)

if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
    print('kinda')
    new_zero_betax = R2.betax + np.random.uniform(0,0.1)
    new_zero_alphax = 1 - R2.betax
    new_zero_betay = R2.betay + np.random.uniform(0,0.1)
    new_zero_alphay = 1 - R2.betay

if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
    print('uffdah')
    new_zero_betax = R3.betax + np.random.uniform(0,0.1)
    new_zero_alphax = 1 - R3.betax
    new_zero_betay = R3.betay + np.random.uniform(0,0.1)
    new_zero_alphay = 1 - R3.betay
```

```
kinda
```

Define the 'new 1':

In [96]:

```
# flip thanks to the 'minus' sign?
# I had tried with if(R1... < R2...) etc., but it is not ok,
# because we need to initialize all elements.

if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
    print('gosh')
    new_one_betax = R1.betax - np.random.uniform(0,0.1)
    new_one_alphax = 1 - R1.betax
    new_one_betay = R1.betay - np.random.uniform(0,0.1)
    new_one_alphay = 1 - R2.betay

if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
    print('kinda')
    new_one_betax = R2.betax - np.random.uniform(0,0.1)
    new_one_alphax = 1 - R2.betax
    new_one_betay = R2.betay - np.random.uniform(0,0.1)
    new_one_alphay = 1 - R2.betay

if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
    print('uffdah')
    new_one_betax = R3.betax - np.random.uniform(0,0.1)
    new_one_alphax = 1 - R3.betax
    new_one_betay = R3.betay - np.random.uniform(0,0.1)
    new_one_alphay = 1 - R3.betay
```

```
kinda
```

Choose the 'new 0' or the 'new 1' according to the outcome of GHZ circuit:

In [97]:

```
if (arr_GHZ[0] =='0'):
    R1.alphax = new_zero_alphax
    R1.betax = new_zero_betax
if (arr_GHZ[1] =='0'):
    R1.alphay = new_zero_alphay
    R1.betay = new_zero_betay
if (arr_GHZ[2] =='0'):
    R2.alphax = new_zero_alphax
    R2.betax = new_zero_betax
if (arr_GHZ[3] =='0'):
    R2.alphay = new_zero_alphay
    R2.betay = new_zero_betay
if (arr_GHZ[3] =='0'):
    R2.alphax = new_zero_alphax
    R2.betax = new_zero_betax
if (arr_GHZ[4] =='0'):
    R3.alphax = new_zero_alphax
    R3.betax = new_zero_betax
if (arr_GHZ[5] =='0'):
    R3.alphay = new_zero_alphay
    R3.betay = new_zero_betay


if (arr_GHZ[0] =='1'):
    R1.alphax = new_one_alphax
    R1.betax = new_one_betax
if (arr_GHZ[1] =='1'):
    R1.alphay = new_one_alphay
    R1.betay = new_one_betay
if (arr_GHZ[2] =='1'):
    R2.alphax = new_one_alphax
    R2.betax = new_one_betax
if (arr_GHZ[3] =='1'):
    R2.alphay = new_one_alphay
    R2.betay = new_one_betay
if (arr_GHZ[3] =='1'):
    R2.alphax = new_one_alphax
    R2.betax = new_one_betax
if (arr_GHZ[4] =='1'):
    R3.alphax = new_one_alphax
    R3.betax = new_one_betax
if (arr_GHZ[5] =='1'):
    R3.alphay = new_one_alphay
    R3.betay = new_one_betay

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1965/2933439320.py in <
module>
----> 1 if (arr_GHZ[0] =='0'):
      2     R1.alphax = new_zero_alphax
      3     R1.betax = new_zero_betax
      4 if (arr_GHZ[1] =='0'):
      5     R1.alphay = new_zero_alphay

NameError: name 'arr_GHZ' is not defined
```

if (arr_GHZ[0] =='0'): # all the other bits are supposed to be equal in GHZ..... R3.alphax = new_zero_alphax R3.betax = new_zero_betax R2.alphax = new_zero_alphax R2.betax = new_zero_betax R1.alphax = new_zero_alphax R1.betax = new_zero_betax if (arr_GHZ[0] =='1'): R3.alphax = new_zero_alphax R3.betax = new_zero_betax R2.alphax = new_zero_alphax R2.betax = new_zero_betax R1.alphax = new_zero_alphax R1.betax = new_zero_betax

New reward for R_1:

In [98]:

```
R1.delta = reward(T,R1.betax,R1.betay)
R1.gamma = round((1 - R1.delta),2)
print(R1.delta)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1965/2123306597.py in <
module>
----> 1 R1.delta = reward(T,R1.betax,R1.betay)
      2 R1.gamma = round((1 - R1.delta),2)
      3 print(R1.delta)

TypeError: reward() missing 1 required positional argument: 'betaz'
```

New reward for R_2:

In [99]:

```
R2.delta = reward(T,R2.betax,R2.betay)
R2.gamma = round((1 - R2.delta),2)
print(R2.delta)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1965/1048257776.py in <
module>
----> 1 R2.delta = reward(T,R2.betax,R2.betay)
      2 R2.gamma = round((1 - R2.delta),2)
      3 print(R2.delta)

TypeError: reward() missing 1 required positional argument: 'betaz'
```

New reward for R_3:

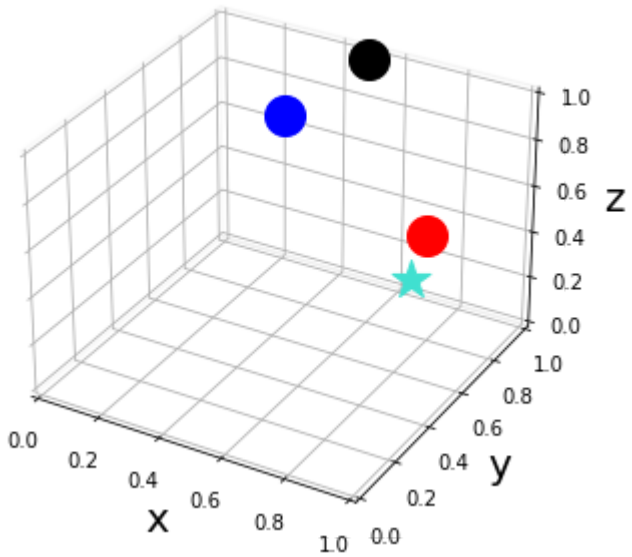
In [100]:

```
R3.delta = reward(T,R3.betax,R3.betay)
R3.gamma = round((1 - R3.delta),2)
print(R3.delta)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1965/1291155766.py in <
module>
----> 1 R3.delta = reward(T,R3.betax,R3.betay)
      2 R3.gamma = round((1 - R3.delta),2)
      3 print(R3.delta)

TypeError: reward() missing 1 required positional argument: 'betaz'
```

In [101]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [102]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time12_ = audio1.overlay(audio2)          # combine , superimpose audio f
mixed_time12  = mixed_time12_.overlay(audio3)          # further combine , supe

mixed_time12.export("notes_/mixed_time12.mp3", format='mp3') # export mixed  au
play(mixed_time12)                                  # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tA
fF#
cA
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpm6e4oea8.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.26 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
   7.29 M-A: -0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

In [103]:

```
1 R1.delta, R2.delta, R3.delta
```

Out[103]:

```
(0.21, 0.51, 0.33)
```

In [104]:

```
1 # January 22, 2022
```

NEW LINES of code: IF the initial reward is very high (greater than 0.8) for at least one of the three robots ("or"), THEN the other robots have to just reach it (with a pretty small fluctuation), without entering the circuit.

In [105]:

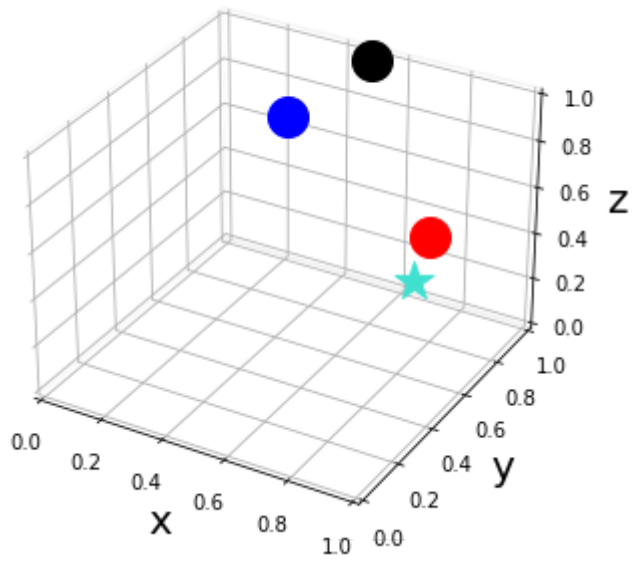
```
if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
    print('yuk')
    if (R1.delta > R2.delta and R1.delta > R3.delta):
        print('quokka')
        R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and late
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)
        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R2.betax, 3)
        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R2.betay, 3)
        R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R2.betaz, 3)
    if (R2.delta > R1.delta and R2.delta > R3.delta):
        print('quagga')
        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R3.betax, 3)
        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R3.betay, 3)
        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R3.betaz, 3)
    if (R3.delta > R1.delta and R3.delta > R2.delta):
        print('quark')
        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
print(R2.delta)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_1965/1730784750.py in <
module>
```

```
     32         R2.alphay = round(1 - R2.betay, 3)
     33
---> 34 R1.delta = reward(T, R1.betax, R1.betay)
     35 print(R2.delta)
     36

TypeError: reward() missing 1 required positional argument: 'betaz'
```

In [106]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [107]:

```
# audio 1, R_1

if(R1.betaz >= 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC.mp3")
            print("tC")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB.mp3")
            print("tB")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
            print("tC#")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
            print("tA#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD.mp3")
            print("tD")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
            print("tD#")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA.mp3")
            print("tA")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE.mp3")
            print("tE")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
            print("tG#")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF.mp3")
            print("tF")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG.mp3")
            print("tG")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
        print("tF#")
if(R1.betaz < 0.5):
    if (R1.betax == 0):
        if (R1.betay == 0.5):
            audio1 = AudioSegment.from_file("notes_/tC2.mp3")
            print("tC2")
    if (R1.betax > 0 and R1.betax <= 0.17):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tB2.mp3")
            print("tB2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
            print("tC#2")
    if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
            audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
            print("tA#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tD2.mp3")
            print("tD2")
    if (R1.betax > 0.3 and R1.betax <= 0.5):
        if (R1.betay < 0.5): # (R1.betay == 1):
            audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
            print("tD#2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tA2.mp3")
            print("tA2")
    if (R1.betax > 0.5 and R1.betax <= 0.64):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tE2.mp3")
            print("tE2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
            print("tG#2")
    if (R1.betax > 0.64 and R1.betax <= 0.84):
        if (R1.betay < 0.5):
            audio1 = AudioSegment.from_file("notes_/tF2.mp3")
            print("tF2")
        if (R1.betay >= 0.5):
            audio1 = AudioSegment.from_file("notes_/tG2.mp3")
            print("tG2")
    if (R1.betax > 0.84 and R1.betax <= 1):
        #if (R1.betay == 0.5):
        audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
        print("tF#2")



    # CHANGE from this point


# audio 2, R_2

if(R2.betaz < 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC2.mp3")
            print("fC2")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB2.mp3")
            print("fB2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
            print("fC#2")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
            print("fA#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD2.mp3")
            print("fD2")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
            print("fD#2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA2.mp3")
            print("fA2")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE2.mp3")
            print("fE2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
            print("fG#2")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF2.mp3")
            print("fF2")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG2.mp3")
            print("fG2")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
        print("fF#2")
if(R2.betaz >= 0.5):
    if (R2.betax == 0):
        if (R2.betay == 0.5):
            audio2 = AudioSegment.from_file("notes_/fC.mp3")
            print("fC")
    if (R2.betax > 0 and R2.betax <= 0.17):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fB.mp3")
            print("fB")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fC#.mp3")
            print("fC#")
    if (R2.betax > 0.17 and R2.betax <= 0.3):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fA#.mp3")
            print("fA#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fD.mp3")
            print("fD")
    if (R2.betax > 0.3 and R2.betax <= 0.5):
        if (R2.betay < 0.5): # (R1.betay == 1):
            audio2 = AudioSegment.from_file("notes_/fD#.mp3")
            print("fD#")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fA.mp3")
            print("fA")
    if (R2.betax > 0.5 and R2.betax <= 0.64):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fE.mp3")
            print("fE")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG#.mp3")
            print("fG#")
    if (R2.betax > 0.64 and R2.betax <= 0.84):
        if (R2.betay < 0.5):
            audio2 = AudioSegment.from_file("notes_/fF.mp3")
            print("fF")
        if (R2.betay >= 0.5):
            audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
            print("fG")
    if (R2.betax > 0.84 and R2.betax <= 1):
        #if (R2.betay == 0.5):
        audio2 = AudioSegment.from_file("notes_/fF#.mp3")
        print("fF#")




# audio 3, R_3

if (R3.betaz >= 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC.mp3")
            print("cC")
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB.mp3")
            print("cB")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#.mp3")
            print("cC#")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#.mp3")
            print("cA#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD.mp3")
            print("cD")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#.mp3")
            print("cD#")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA.mp3")
            print("cA")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE.mp3")
            print("cE")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#.mp3")
            print("cG#")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF.mp3")
            print("cF")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG.mp3")
            print("cG")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#.mp3")
        print("cF#")
if (R3.betaz < 0.5):
    if (R3.betax == 0):
        if (R3.betay == 0.5):
            audio3 = AudioSegment.from_file("notes_/cC2.mp3")
            print("cC2")
```

```
    if (R3.betax > 0 and R3.betax <= 0.17):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cB2.mp3")
            print("cB2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
            print("cC#2")
    if (R3.betax > 0.17 and R3.betax <= 0.3):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
            print("cA#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cD2.mp3")
            print("cD2")
    if (R3.betax > 0.3 and R3.betax <= 0.5):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
            print("cD#2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cA2.mp3")
            print("cA2")
    if (R3.betax > 0.5 and R3.betax <= 0.64):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cE2.mp3")
            print("cE2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
            print("cG#2")
    if (R3.betax > 0.64 and R3.betax <= 0.84):
        if (R3.betay < 0.5):
            audio3 = AudioSegment.from_file("notes_/cF2.mp3")
            print("cF2")
        if (R3.betay >= 0.5):
            audio3 = AudioSegment.from_file("notes_/cG2.mp3")
            print("cG2")
    if (R3.betax > 0.84 and R3.betax <= 1):
        #if (R3.betay == 0.5):
        audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
        print("cF#2")

mixed_time13_ = audio1.overlay(audio2)          # combine , superimpose audio f
mixed_time13  = mixed_time13_.overlay(audio3)          # further combine , supe

mixed_time13.export("notes_/mixed_time13.mp3", format='mp3') # export mixed  au
play(mixed_time13)                                  # play mixed audio file
# change this line at each time point, so in the end we can get a little piece

```

```
tA
fF#
cA
Could not import the PyAudio C module '_portaudio'.

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmp5s4l54d6.
wav':
  Duration: 00:00:07.34, bitrate: 1411 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
   7.23 M-A:  0.000 fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

```
   7.31 M-A: -0.000 fd=    0 aq=     0KB vq=     0KB sq=     0B f=0/0
```

In [2]:

```
1 R2.alphax, R2.betax, R2.alphay, R2.betay
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/ipykernel_2096/805032336.py in <m
odule>
----> 1 R2.alphax, R2.betax, R2.alphay, R2.betay

NameError: name 'R2' is not defined
```

In [93]:

```
1 R3.alphax, R3.betax, R3.alphay, R3.betay
```

Out[93]:

```
(0.08099999999999996,
 0.9434134380350225,
 0.42300000000000004,
 0.6478347955734719)
```

In [94]:

```
1 R1.delta, R2.delta, R3.delta
```

Out[94]:

```
(0.85, 0.85, 0.85)
```

In [95]:

```
1 # New # January 26
```

Section added on January 26: getting back to the initial position. We have two ways to do that: either creating arrays to keep individual initial positions, or creating a second target if initial positions were all close to each other. For the sake of simplicity, I'm choosing this second option.

REWARDS IN THE SECOND PART

In [96]:

```
# Recalculate delta according to the second target (e.g., getting back to the ne

R1.delta = reward(T2, R1.betax, R1.betay)
print(R1.delta)

R2.delta = reward(T2, R2.betax, R2.betay)
print(R2.delta)

R3.delta = reward(T2, R3.betax, R3.betay)
print(R3.delta)
```

```
0.24
0.24
0.24
```

Shuffle only one

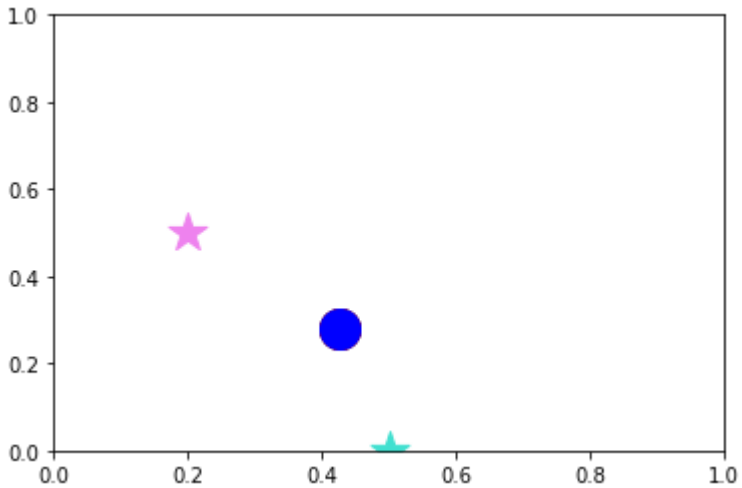
In [82]:

```
if (R1.delta and R2.delta and R3.delta) >= 0.8 and (R1.delta and R2.delta and R3
    print("ciao ciao")
    R1.alphax = round(np.random.uniform(0,0.2), 3) # slightly shuffle position c
    R1.betax = round(1 - R1.alphax, 3)
    #R1.alphay = round(np.random.uniform(0,0.2), 3) # slightly shuffle position
    #R1.betay = round(1 - R1.alphay, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
```

graph

In [83]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is violet

SOS shuffle

In [84]:

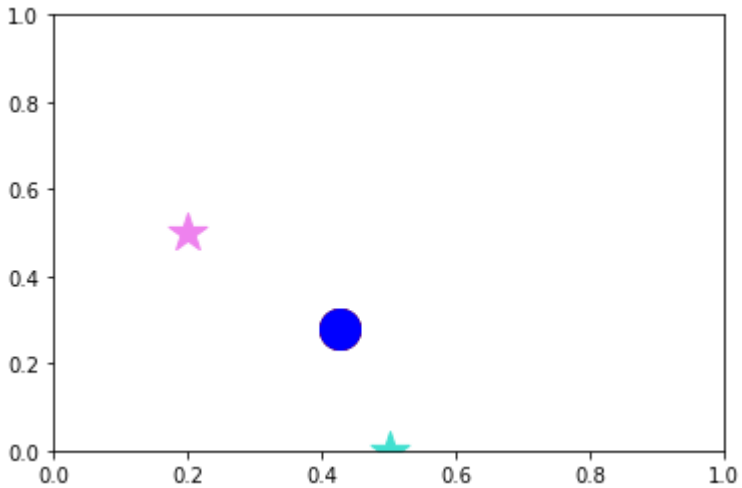
```
# threshold for initial reward
# random fluctuations

if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
    print("SOS")
    # R1
    R1.alphax = round(np.random.uniform(0,0.9), 3)
    R1.betax = round(1 - R1.alphax, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
    R1.alphay = round(np.random.uniform(0,0.9), 3)
    R1.betay = round(1 - R1.alphay, 3)
    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
    # R2
    R2.alphax = round(np.random.uniform(0,0.9), 3)
    R2.betax = round(1 - R2.alphax, 3)
    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
    R2.alphay = round(np.random.uniform(0,0.9), 3)
    R2.betay = round(1 - R2.alphay, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    # R3
    R3.alphax = round(np.random.uniform(0,0.9), 3)
    R3.betax = round(1 - R3.alphax, 3)
    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
    R3.alphay = round(np.random.uniform(0,0.9), 3)
    R3.betay = round(1 - R3.alphay, 3)
    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)

R1.delta = reward(T2, R1.betax, R1.betay)
R1.gamma = 1 - R1.delta
R2.delta = reward(T2, R2.betax, R2.betay)
R2.gamma = 1 - R2.delta
R3.delta = reward(T2, R3.betax, R3.betay)
R3.gamma = 1 - R3.delta
print(R1.delta, R2.delta, R3.delta)
```

```
0.69 0.69 0.69
```

graph

In [85]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```




```
R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is viol
et
```

reach the most successful robot

In [86]:

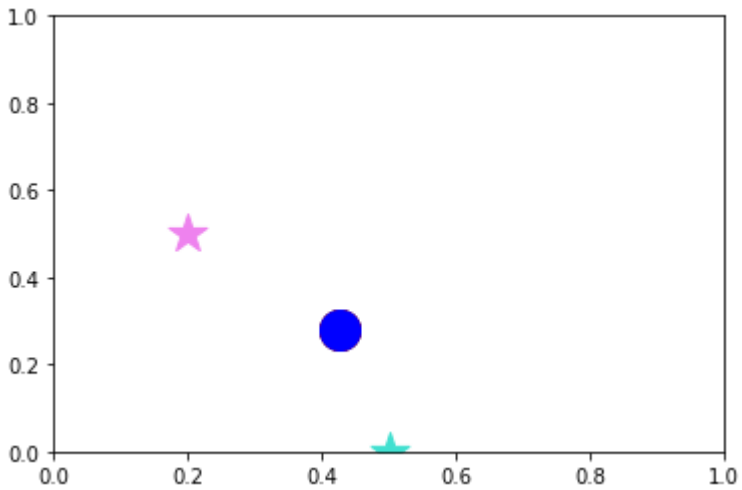
```
if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
    print('yuk')
    if (R1.delta > R2.delta and R1.delta > R3.delta):
        print('quokka')
        R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)
        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R2.betax, 3)
        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R2.betay, 3)
        R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R2.betaz, 3)
    if (R2.delta > R1.delta and R2.delta > R3.delta):
        print('quagga')
        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R3.betax, 3)
        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R3.betay, 3)
        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R3.betaz, 3)
    if (R3.delta > R1.delta and R3.delta > R2.delta):
        print('quark')
        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)

R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
print(R2.delta)
```

```
0.69
0.69
0.69
```

plot

In [87]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is violet

again SOS reshuffle

In [88]:

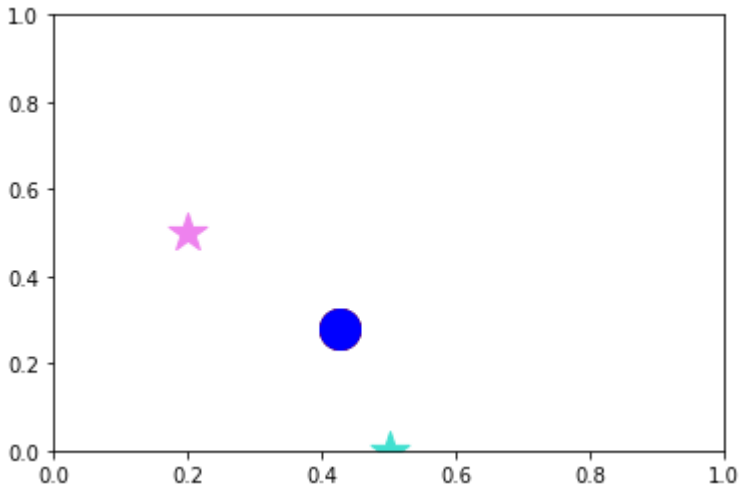
```
# Another round of SOS re-shuffle

# threshold for initial reward
# random fluctuations

if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
    print("SOS")
    # R1
    R1.alphax = round(np.random.uniform(0,0.9), 3)
    R1.betax = round(1 - R1.alphax, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
    R1.alphay = round(np.random.uniform(0,0.9), 3)
    R1.betay = round(1 - R1.alphay, 3)
    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
    # R2
    R2.alphax = round(np.random.uniform(0,0.9), 3)
    R2.betax = round(1 - R2.alphax, 3)
    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
    R2.alphay = round(np.random.uniform(0,0.9), 3)
    R2.betay = round(1 - R2.alphay, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    # R3
    R3.alphax = round(np.random.uniform(0,0.9), 3)
    R3.betax = round(1 - R3.alphax, 3)
    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
    R3.alphay = round(np.random.uniform(0,0.9), 3)
    R3.betay = round(1 - R3.alphay, 3)
    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)

R1.delta = reward(T2, R1.betax, R1.betay)
R1.gamma = 1 - R1.delta
R2.delta = reward(T2, R2.betax, R2.betay)
R2.gamma = 1 - R2.delta
R3.delta = reward(T2, R3.betax, R3.betay)
R3.gamma = 1 - R3.delta
print(R1.delta, R2.delta, R3.delta)
```

```
0.69 0.69 0.69
```

graph

In [89]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is violet

In [90]:

```
q = QuantumRegister(5, 'q'); # qubits # changed to 9, formerly 15
m2 = ClassicalRegister(1, 'c1'); # classical bits (separated is better)
m3 = ClassicalRegister(1, 'c2');
m4 = ClassicalRegister(1, 'c3');

qc4 = QuantumCircuit(q, m2, m3, m4); # to get back to the nest
```

Gate! SECOND GATE

In [91]:

```
if (R1.delta > R2.delta) and (R1.delta > R3.delta):
    if (R1.alphax < 0.3): # I have to customize state vectors according to prec
        qc4.x(q[0])        # just using the NOT gate as a test
    if (R1.alphax == 0.5): # I have to customize state vectors according to pre
        qc4.h(q[0])
    if (R1.alphax >= 0.3) and (R2.alphax < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[0])
    if (R1.alphax >= 0.6) and (R2.alphax < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[0])
    if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
        qc4.x(q[1])
    if (R1.alphay == 0.5): # I have to customize state vectors according to pre
        qc4.h(q[1])
    if (R1.alphay >= 0.3) and (R2.alphay < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[1])
    if (R1.alphay >= 0.6) and (R2.alphay < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[1])
    if (R1.delta == 0.5):
        qc4.h(q[2])
    if (R1.delta == 0.6):
        qc4.h(q[2])
    if (R1.delta >= 0.7):
        qc4.x(q[2])
    if (R1.gamma >= 0.3) and (R2.gamma < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[2])
    if (R1.gamma >= 0.6) and (R2.gamma < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[2])
elif (R1.delta > R2.delta) and (R1.delta > R3.delta):
    print('dog')
    if (R1.alphax < 0.3): # I have to customize state vectors according to prec
        qc4.x(q[0])        # just using the NOT gate as a test
    if (R1.alphax == 0.5): # I have to customize state vectors according to pre
        qc4.h(q[0])
    if (R1.alphax >= 0.3) and (R1.alphax < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[0])
    if (R1.alphax >= 0.6) and (R1.alphax < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[0])
    if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
        qc4.x(q[1])
    if (R1.alphay == 0.5): # I have to customize state vectors according to pre
        qc4.h(q[1])
    if (R1.alphay >= 0.3) and (R1.alphay < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[1])
    if (R1.alphay >= 0.6) and (R1.alphay < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[1])
    if (R1.delta == 0.5):
        qc4.h(q[2])
    if (R1.delta == 0.6):
        qc4.h(q[2])
```

```
    if (R1.delta >= 0.7):
        qc4.x(q[2])
    if (R1.gamma >= 0.3) and (R1.gamma < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[2])
    if (R1.gamma >= 0.6) and (R1.gamma < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[2])
else:
    print('cat') # I made some tests to check the IF conditions
    if (R3.alphax < 0.3):
        qc4.x(q[0])
    if (R3.alphax == 0.5):
        qc4.h(q[0])
    if (R3.alphax >= 0.3) and (R3.alphax < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[0])
    if (R3.alphax >= 0.6) and (R3.alphax < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[0])
    if (R3.alphay < 0.3):
        qc4.x(q[1])
    if (R3.alphay == 0.5):
        qc4.h(q[1])
    if (R3.alphay >= 0.3) and (R3.alphay < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[1])
    if (R3.alphay >= 0.6) and (R3.alphay < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[1])
    if (R3.delta == 0.5):
        qc4.h(q[2])
    if (R3.delta == 0.6):
        qc4.h(q[2])
    if (R3.delta >= 0.7):
        qc4.x(q[2])
    if (R3.gamma >= 0.3) and (R3.gamma < 0.5):
        print('jungle!')
        qc4.ry(1.9106332, q[2])
    if (R3.gamma >= 0.6) and (R3.gamma < 0.7):
        print('ocean!')
        qc4.ry(1.2309594, q[2])
```

```
cat
jungle!
```

the core code

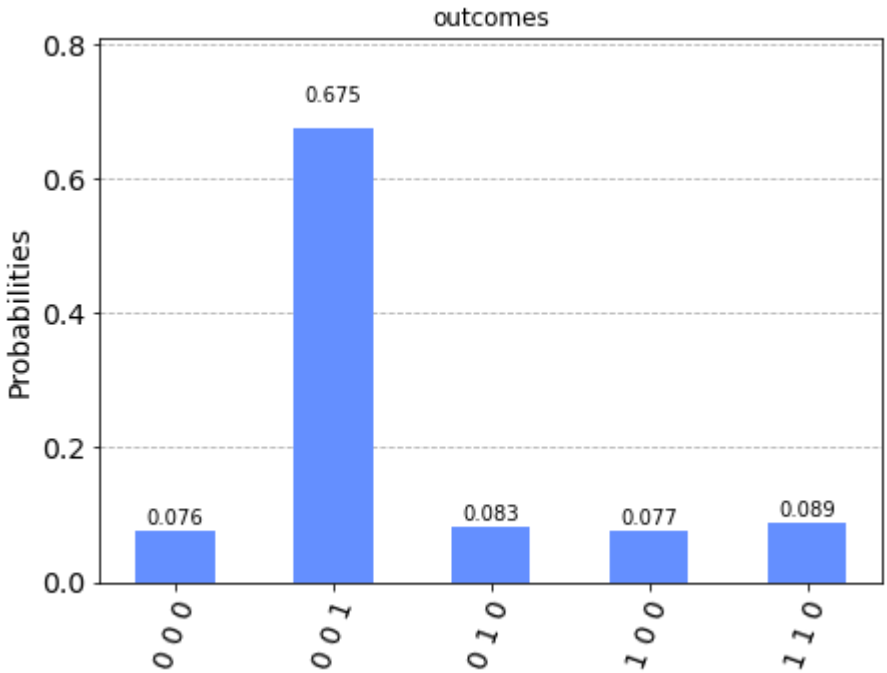
In [92]:

```
# this is the core code, and it is unchanged across time

qc4.barrier(q)
qc4.ccx(q[0],q[1],q[3])
qc4.ccx(q[0],q[1],q[4])

qc4.reset(q[3]);
qc4.reset(q[4]);

qc4.ccx(q[0],q[2],q[3])
qc4.ccx(q[1],q[2],q[4])

qc4.x(q[2])

qc4.ch(q[2],q[3])
qc4.ch(q[2],q[4])

qc4.x(q[2])

qc4.barrier(q)

# perform measurements and store them in classical bits

qc4.measure(q[2],m2[0])
qc4.measure(q[3],m3[0])
qc4.measure(q[4],m4[0])

# visualization of the ciruit

draw_circuit(qc4)

# definition of quantum simulator

simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_simu
qc4 = transpile(qc4, simulator)
cc = collections.Counter()

# Run and get counts
result = simulator.run(qc4, shots=1024).result()
counts = result.get_counts(qc4)
counts2 = counts.most_frequent() # does not work if multiple states have the sam
# decide something if multiple states have the same count --> e.g., ``choose the
counts3 = cc.most_common(2)
print(counts)
print(counts2)
print(counts3)
result = simulator.run(qc4, shots=10, memory=True).result()
memory = result.get_memory(qc4)
print(memory)
plot_histogram(counts, title='outcomes')
# TAKE the TWO more present outcomes
```

```
CircuitComposer(circuit=<qiskit.circuit.quantumcircuit.QuantumCircuit object at 0x
7fb66d793ee0>, editable=Fals…

{'1 0 0': 79, '0 1 0': 85, '1 1 0': 91, '0 0 0': 78, '0 0 1': 691}
0 0 1
[]
```

```
['0 0 1', '0 0 1', '0 0 1', '0 0 0', '0 1 0', '0 0 1', '0 0 1', '0 0 1', '0 0 1',
'1 0 0']
```

Out[92]:



In [93]:

```
1 # I should create new arrays 1, 2, but here I keep the same names for the sake o
2 # The arrays are automatically rewritten...
```

My intervention is needed here, where indicated:

In [93]:

```
print(counts2) # order: R3, R2, R1. Add some uncertainty?
# export as an array
str = counts2
arr1 = str.split(' ') # to split the string and avoid empty spaces as array elem
print(arr1)
weight1 = 691 # AT HAND ONLY FOR NOW

arr2 = ['1','1','0'] # 111 # 011
print(arr2)
weight2 = 91
# BY HAND ONLY FOR NOW


# an attempt, not so good, to automatize this passage:

print(memory)

data = Counter(memory)
data.most_common()   # Returns all unique items and their counts
data.most_common(3)

print(data.most_common())
print(data.most_common(1))
arrx1 = data.most_common(2)[0]
print(arrx1)
arrx2 = data.most_common(2)[1]
print(arrx2)
```

```
0 0 1
['0', '0', '1']
['1', '1', '0']
['0 0 1', '0 0 1', '0 0 1', '0 0 0', '0 1 0', '0 0 1', '0 0 1', '0 0 1', '0 0 1',
'1 0 0']
[('0 0 1', 7), ('0 0 0', 1), ('0 1 0', 1), ('1 0 0', 1)]
[('0 0 1', 7)]
('0 0 1', 7)
('0 0 0', 1)
```

check

In [94]:

```
arr1, arr2
```

Out[94]:

```
(['0', '0', '1'], ['1', '1', '0'])
```

reset position of R_1

In [95]:

```
if (R1.delta > R2.delta) and (R1.delta > R3.delta):
    # if R1 didn't enter the gate, keep its position
    R1.alphax = R1.alphax
    R1.betax = R1.betax
    R1.alphay = R1.alphay
    R1.betay = R1.betay
else:
    # same outcome = 1 # January 23
    # x part


    # change [0] --> [1] and vice versa, January 28
    if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R1.alphax = 0.3
        R1.betax = 0.7
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R1.alphax = 0.7
        R1.betax = 0.3
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R1.alphax = 0.5 # change temporarily made on January 24: random generat
        R1.betax = 0.5 # same as above
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R1.alphax = 0
        R1.betax = 1
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R1.alphax = 0.1
        R1.betax = 0.9
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R1.alphax = 0.9
        R1.betax = 0.1
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R1.alphax = 0.1 # the same also in this case
        R1.betax = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R1.alphax = 0.7  # the opposite??
        R1.betax = 0.3  # the opposite??
    elif (arr1[1] == arr2[0]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R1.alphax = 0.3  # the opposite??
        R1.betax = 0.7  # the opposite??
    elif (arr1[1] == arr2[0]) and (arr1[1] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R1.alphax = 0.5 # change temporarily made on January 24: random generat
        R1.betax = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[1] != arr2[1]:  # January 23
        print("blue")
        if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R1.alphax = 0.5 # change temporarily made on January 24: random gen
```

```
            R1.betax = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R1.alphax = 0.5
            R1.betax = 0.5
        if (arr1[1] == '1' and arr2[1] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R1.alphax = 0.3
                R1.betax = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R1.alphax = 0.7
                R1.betax = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R1.alphax = 0.2 #
                R1.betax = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R1.alphax = 0.8 #
                R1.betax = 0.2 #
        if (arr1[1] == '0') and (arr2[1] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R1.alphax = 0.7
                R1.betax = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R1.alphax = 0.3
                R1.betax = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R1.alphax = 0.7 # 0.9
                R1.betax = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R1.alphax = 0.3 # 0.1
                R1.betax = 0.7 # 0.9
    # y part

    # change of January 26

    if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R1.alphay = 0.3
        R1.betay = 0.7
    elif (arr1[0] == arr2[1]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R1.alphay = 0.7
        R1.betay = 0.3
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R1.alphay = 0.5 # change temporarily made on January 24: random generat
        R1.betay = 0.5 # same as above
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R1.alphay = 0
        R1.betay = 1
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
```

```
        print("avalanche!")
        R1.alphay = 0.1
        R1.betay = 0.9
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R1.alphay = 0.9
        R1.betay = 0.1
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R1.alphay = 0.1 # the same also in this case
        R1.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R1.alphay = 0.7  # the opposite??
        R1.betay = 0.3  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R1.alphay = 0.3  # the opposite??
        R1.betay = 0.7  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R1.alphay = 0.5 # change temporarily made on January 24: random generat
        R1.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[0] != arr2[0]:  # January 23
        print("blue")
        if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R1.alphay = 0.5 # change temporarily made on January 24: random gen
            R1.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R1.alphay = 0.5
            R1.betay = 0.5
        if (arr1[0] == '1' and arr2[0] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R1.alphay = 0.3
                R1.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R1.alphay = 0.7
                R1.betay = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R1.alphay = 0.2 #
                R1.betay = 0.8 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R1.alphay = 0.8 #
                R1.betay = 0.2 #
        if (arr1[0] == '0') and (arr2[0] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R1.alphay = 0.7
                R1.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R1.alphay = 0.3
```

```
                R1.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R1.alphay = 0.7 # 0.9
                R1.betay = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
                R1.alphay = 0.3 # 0.1
                R1.betay = 0.7 # 0.9
```

```
blue
brain
blue
brain
```

reset position of R_2

In [96]:

```
if (R2.delta > R1.delta) and (R2.delta > R3.delta):
    # if R2 didn't entered the gate, keep its position
    R2.alphax = R2.alphax
    R2.betax = R2.betax
    R2.alphay = R2.alphay
    R2.betay = R2.betay
else:
    # same outcome = 1 # January 23
    # x part
    if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R2.alphax = 0.3
        R2.betax = 0.7
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R2.alphax = 0.7
        R2.betax = 0.3
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R2.alphax = 0.5 # change temporarily made on January 24: random generat
        R2.betax = 0.5 # change temporarily made on January 24: random generato
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R2.alphax = 0
        R2.betax = 1
    elif (arr1[1] == arr2[0]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R2.alphax = 0.1
        R2.betax = 0.9
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R2.alphax = 0.9
        R2.betax = 0.1
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphax = 0.1 # the same also in this case
        R2.betax = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphax = 0.7  # the opposite??
        R2.betax = 0.3  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphax = 0.3  # the opposite??
        R2.betax = 0.7  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphax = 0.5 #1   # the opposite
        R2.betax = 0.5 #0  # the opposite
    # different outcomes
    elif arr1[1] != arr2[1]:  # January 23
        print("blue")
        if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphax = 0.5 # change temporarily made on January 24: random gen
            R2.betax = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
```

```
            print("uffdah")
            R2.alphax = 0.5 # change temporarily made on January 24: random gen
            R2.betax = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[1] == '0'):
            print("gasp")
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphax = 0.3
                R2.betax = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphax = 0.7
                R2.betax = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphax = 0.3 # 0. 126 January
                R2.betax = 0.7 # 0.9
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphax = 0.7 # 0.9
                R2.betax = 0.3 # 0.1
        if (arr1[1] == '0') and (arr2[1] == '1'):
            print("sigh")
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphax = 0.7
                R2.betax = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphax = 0.3
                R2.betax = 0.7

    # y part
     # change of January 26

     # to be modified: R1 to R2

     if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
         print("bla")
         R2.alphay = 0.3
         R2.betay = 0.7
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
         print("gulp")
         R2.alphay = 0.7
         R2.betay = 0.3
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
         print("stra-gulp")
         R2.alphay = 0.5 # change temporarily made on January 24: random generat
         R2.betay = 0.5 # same as above
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
         print("thunderstorm!")
         R2.alphay = 0
         R2.betay = 1
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
         print("avalanche!")
         R2.alphay = 0.1
         R2.betay = 0.9
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
         print("earthquake!")
         R2.alphay = 0.9
         R2.betay = 0.1
```

```
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphay = 0.1 # the same also in this case
        R2.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphay = 0.7  # the opposite??
        R2.betay = 0.3  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphay = 0.3  # the opposite??
        R2.betay = 0.7  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphay = 0.5 # change temporarily made on January 24: random generat
        R2.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[0] != arr2[0]:  # January 23
        print("blue")
        if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphay = 0.5 # change temporarily made on January 24: random gen
            R2.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R2.alphay = 0.5
            R2.betay = 0.5
        if (arr1[0] == '1' and arr2[0] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphay = 0.3
                R2.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphay = 0.7
                R2.betay = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphay = 0.3 #
                R2.betay = 0.7 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphay = 0.7 #
                R2.betay = 0.3 #
        if (arr1[0] == '0') and (arr2[0] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphay = 0.7
                R2.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphay = 0.3
                R2.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R2.alphay = 0.7 # 0.9
                R2.betay = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
```

```
182                 R2.alphay = 0.3 # 0.1
183                 R2.betay = 0.7 # 0.9
```

```
blue
sigh
blue
brain
```

reset position of R_3

In [97]:

```
if (R2.delta > R1.delta) and (R2.delta > R3.delta):
    # if R2 didn't entered the gate, keep its position
    R2.alphax = R2.alphax
    R2.betax = R2.betax
    R2.alphay = R2.alphay
    R2.betay = R2.betay
else:
    # same outcome = 1 # January 23
    # x part
    if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
        print("bla")
        R2.alphax = 0.3
        R2.betax = 0.7
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
        print("gulp")
        R2.alphax = 0.7
        R2.betax = 0.3
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
        print("stra-gulp")
        R2.alphax = 0.5 # change temporarily made on January 24: random generat
        R2.betax = 0.5 # change temporarily made on January 24: random generato
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
        print("thunderstorm!")
        R2.alphax = 0
        R2.betax = 1
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("avalanche!")
        R2.alphax = 0.1
        R2.betax = 0.9
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
        print("earthquake!")
        R2.alphax = 0.8
        R2.betax = 0.2
    elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphax = 0.2 # the same also in this case
        R2.betax = 0.8 # the same also in this case # January 26; this or 0.3,
    # same = outcome 0  # January 23
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphax = 0.7  # the opposite??
        R2.betax = 0.3  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphax = 0.3  # the opposite??
        R2.betax = 0.7  # the opposite??
    elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphax = 0.5 #1   # the opposite
        R2.betax = 0.5 #0  # the opposite
    # different outcomes
    elif arr1[1] != arr2[1]:  # January 23
        print("blue")
        if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphax = 0.5 # change temporarily made on January 24: random gen
            R2.betax = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
```

```
            print("uffdah")
            R2.alphax = 0.5 # change temporarily made on January 24: random gen
            R2.betax = 0.5 # change temporarily made on January 24: random gene
        if (arr1[1] == '1' and arr2[1] == '0'):
            print("gasp")
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphax = 0.3
                R2.betax = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphax = 0.7
                R2.betax = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphax = 0.3 #
                R2.betax = 0.7 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphax = 0.7 #
                R2.betax = 0.3 #
        if (arr1[1] == '0') and (arr2[1] == '1'):
            print("sigh")
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphax = 0.7
                R2.betax = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphax = 0.3
                R2.betax = 0.7

    # y part
     # change of January 26

     # to be modified: R1 to R2

     if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
         print("bla")
         R2.alphay = 0.3
         R2.betay = 0.7
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
         print("gulp")
         R2.alphay = 0.7
         R2.betay = 0.3
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
         print("stra-gulp")
         R2.alphay = 0.5 # change temporarily made on January 24: random generat
         R2.betay = 0.5 # same as above
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
         print("thunderstorm!")
         R2.alphay = 0
         R2.betay = 1
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
         print("avalanche!")
         R2.alphay = 0.1
         R2.betay = 0.9
     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
         print("earthquake!")
         R2.alphay = 0.9
         R2.betay = 0.1
```

```
    elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
        print("avalanche bis!")
        R2.alphay = 0.1 # the same also in this case
        R2.betay = 0.9 # the same also in this case
    # same = outcome 0  # January 23
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
        print("bla 2")
        R2.alphay = 0.7  # the opposite??
        R2.betay = 0.3  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
        print("gulp 2")
        R2.alphay = 0.3  # the opposite??
        R2.betay = 0.7  # the opposite??
    elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or n
        print("stra-gulp 2")
        R2.alphay = 0.5 # change temporarily made on January 24: random generat
        R2.betay = 0.5 # change temporarily made on January 24: random generato
    # different outcomes
    elif arr1[0] != arr2[0]:  # January 23
        print("blue")
        if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1
            print("google 1")
            R2.alphay = 0.5 # change temporarily made on January 24: random gen
            R2.betay = 0.5 # change temporarily made on January 24: random gene
        if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.ab
            # include the case of a very small difference!
            print("uffdah")
            R2.alphay = 0.5
            R2.betay = 0.5
        if (arr1[0] == '1' and arr2[0] == '0'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("abc")
                R2.alphay = 0.3
                R2.betay = 0.7
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("bca")
                R2.alphay = 0.7
                R2.betay = 0.3
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("news")
                R2.alphay = 0.3 #
                R2.betay = 0.7 #
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("idea")
                R2.alphay = 0.7 #
                R2.betay = 0.3 #
        if (arr1[0] == '0') and (arr2[0] == '1'):
            if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
                print("bac")
                R2.alphay = 0.7
                R2.betay = 0.3
            if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
                print("cba")
                R2.alphay = 0.3
                R2.betay = 0.7
            if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
                print("brain")
                R2.alphay = 0.7 # 0.9
                R2.betay = 0.3 # 0.1
            if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
                print("hand")
```

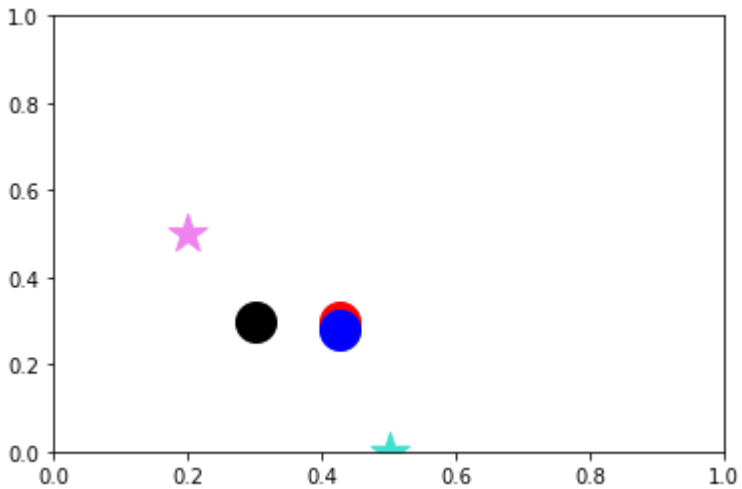
```
182                 R2.alphay = 0.3 # 0.1
183                 R2.betay = 0.7 # 0.9
```

```
blue
sigh
blue
brain
```

graph

In [98]:

```
x = R1.betax
y = R1.betay
#plt.plot(x,y, 'o', c = 'black');
plt.scatter(x,y, s = 400, c = 'black')

x = R2.betax
y = R2.betay
plt.scatter(x, y, s = 400, c = 'red')

x = R3.betax
y = R3.betay
plt.scatter(x, y, s = 400, c = 'blue')

x = T.x
y = T.y
plt.scatter(x, y, s = 400, marker = '*', c = 'turquoise')

x = T2.x
y = T2.y
plt.scatter(x, y, s = 400, marker = '*', c = 'violet')

plt.axis([0, 1, 0, 1])
plt.show()
print('R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target
```




```
R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is viol
et
```

In [99]:

```
# former rewards

R1.delta, R2.delta, R3.delta
```

Out[99]:

```
(0.69, 0.69, 0.69)
```

In [100]:

```
# new rewards

R1.delta = reward(T2, R1.betax, R1.betay)
print(R1.delta)

R2.delta = reward(T2, R2.betax, R2.betay)
print(R2.delta)

R3.delta = reward(T2, R3.betax, R3.betay)
print(R3.delta)
```

```
0.78
0.7
0.69
```

reach the most successful robot

In [101]:

```
if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
    print('yuk')
    if (R1.delta > R2.delta and R1.delta > R3.delta):
        print('quokka')
        R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and late
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)
        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R2.betax, 3)
        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R2.betay, 3)
        R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R2.betaz, 3)
    if (R2.delta > R1.delta and R2.delta > R3.delta):
        print('quagga')
        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R3.betax, 3)
        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R3.betay, 3)
        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R3.betaz, 3)
    if (R3.delta > R1.delta and R3.delta > R2.delta):
        print('quark')
        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)

R1.delta = reward(T2, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T2, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T2, R3.betax, R3.betay, R3.betaz)
print(R2.delta)
```

```
0.7
0.7
0.69
```

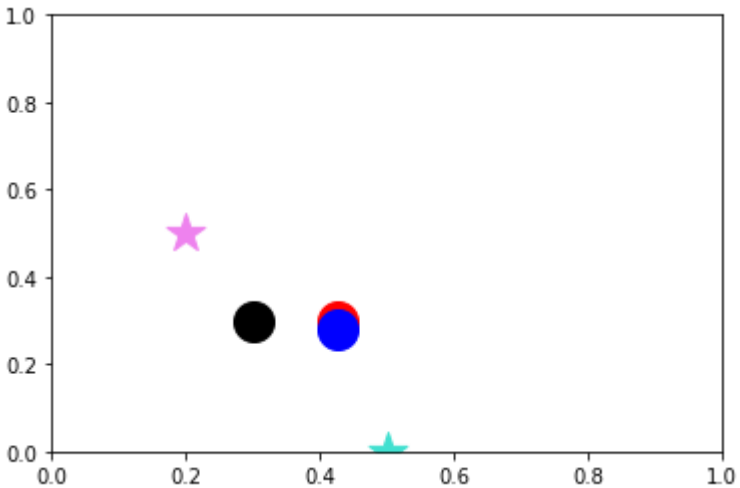
In [ ]:

```
1
```

plot

In [102]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is violet

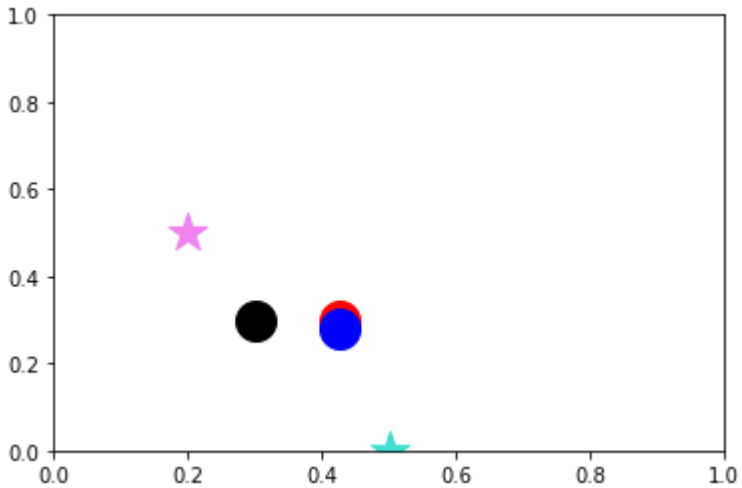
In [103]:

```
# Another round of SOS with a higher threshold. Added on January 28

# threshold for initial reward
# random fluctuations

if (R1.delta <= 0.6) and (R2.delta <= 0.6) and (R3.delta <= 0.6):
    print("SOS")
    # R1
    R1.alphax = round(np.random.uniform(0,0.9), 3)
    R1.betax = round(1 - R1.alphax, 3)
    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
    R1.alphay = round(np.random.uniform(0,0.9), 3)
    R1.betay = round(1 - R1.alphay, 3)
    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
    # R2
    R2.alphax = round(np.random.uniform(0,0.9), 3)
    R2.betax = round(1 - R2.alphax, 3)
    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
    R2.alphay = round(np.random.uniform(0,0.9), 3)
    R2.betay = round(1 - R2.alphay, 3)
    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
    # R3
    R3.alphax = round(np.random.uniform(0,0.9), 3)
    R3.betax = round(1 - R3.alphax, 3)
    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
    R3.alphay = round(np.random.uniform(0,0.9), 3)
    R3.betay = round(1 - R3.alphay, 3)
    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)

R1.delta = reward(T2, R1.betax, R1.betay)
R1.gamma = 1 - R1.delta
R2.delta = reward(T2, R2.betax, R2.betay)
R2.gamma = 1 - R2.delta
R3.delta = reward(T2, R3.betax, R3.betay)
R3.gamma = 1 - R3.delta
print(R1.delta, R2.delta, R3.delta)
```

```
0.78 0.7 0.69
```

In [104]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```




```
R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is viol
et
```

reach the most successful robot:

In [105]:

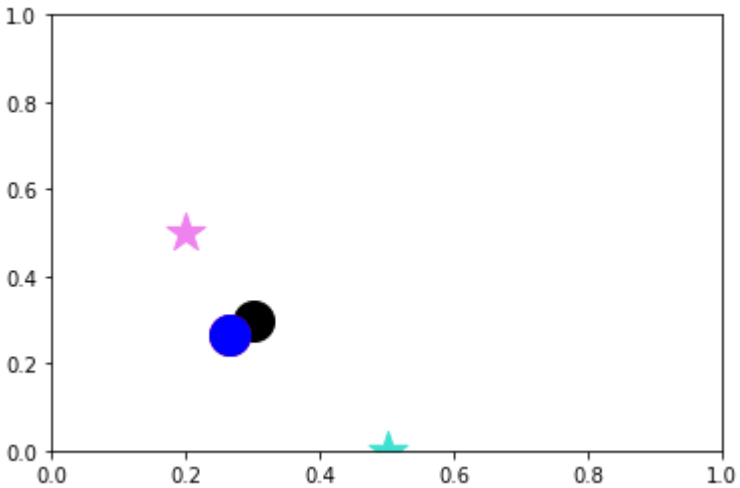
```
if (R1.delta > R2.delta) and (R1.delta > R2.delta): # now, brutally come close t
    print("ciao")
    # change R2, R3 but not R1
    # R2
    R2.alphax = round(R1.alphax + np.random.uniform(0,0.1), 3)
    R2.betax = round(1 - R2.alphax, 3)
    R2.alphay = round(R1.alphax + np.random.uniform(0,0.1), 3)
    R2.betay = round(1 - R2.alphax, 3)
    # R3
    R3.alphax = round(R1.alphax + np.random.uniform(0,0.1), 3)
    R3.betax = round(1 - R2.alphax, 3)
    R3.alphay = round(R1.alphax + np.random.uniform(0,0.1), 3)
    R3.betay = round(1 - R2.alphax, 3)
elif (R2.delta > R3.delta) and (R2.delta > R3.delta):
    print("glu glu")
    # change R1, R3 but not R2
    # R1
    R1.alphax = round(R2.alphax + np.random.uniform(0,0.1), 3)
    R1.betax = round(1 - R1.alphax, 3)
    R1.alphay = round(R2.alphay + np.random.uniform(0,0.1), 3)
    R1.betay = round(1 - R1.alphay, 3)
    # R3
    R3.alphax = round(R2.alphax + np.random.uniform(0,0.1), 3)
    R3.betax = round(1 - R2.alphax, 3)
    R3.alphay = round(R2.alphax + np.random.uniform(0,0.1), 3)
    R3.betay = round(1 - R2.alphax, 3)
elif (R3.delta > R1.delta) and (R3.delta > R2.delta):
    print("cri cri")
    # change R1, R2 but nor R3
    # R1
    R1.alphax = round(R3.alphax + np.random.uniform(0,0.1), 3)
    R1.betax = round(1 - R1.alphax, 3)
    R1.alphay = round(R3.alphay + np.random.uniform(0,0.1), 3)
    R1.betay = round(1 - R1.alphay, 3)
    # R2
    R2.alphax = round(R3.alphax + np.random.uniform(0,0.1), 3)
    R2.betax = round(1 - R2.alphax, 3)
    R2.alphay = round(R3.alphax + np.random.uniform(0,0.1), 3)
    R2.betay = round(1 - R2.alphax, 3)
```

```
ciao
```

plot and rewards

In [106]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```




```
R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is viol
et
0.78
0.76
0.76
```

check and move + rewards

In [107]:

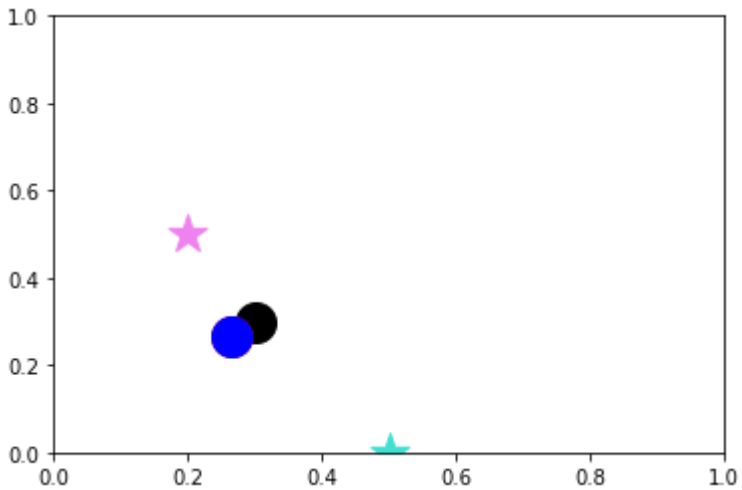
```
if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
    print('yuk')
    if (R1.delta > R2.delta and R1.delta > R3.delta):
        print('quokka')
        R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and late
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)
        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R2.betax, 3)
        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R2.betay, 3)
        R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R2.betaz, 3)
    if (R2.delta > R1.delta and R2.delta > R3.delta):
        print('quagga')
        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
        R3.alphax = round(1 - R3.betax, 3)
        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
        R3.alphay = round(1 - R3.betay, 3)
        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
        R3.alphaz = round(1 - R3.betaz, 3)
    if (R3.delta > R1.delta and R3.delta > R2.delta):
        print('quark')
        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R1.alphax = round(1 - R1.betax, 3)
        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R1.alphay = round(1 - R1.betay, 3)
        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R1.alphaz = round(1 - R1.betaz, 3)
        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
        R2.alphax = round(1 - R2.betax, 3)
        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
        R2.alphay = round(1 - R2.betay, 3)
        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
        R2.alphaz = round(1 - R2.betaz, 3)

R1.delta = reward(T2, R1.betax, R1.betay, R1.betaz)
print(R1.delta)

R2.delta = reward(T2, R2.betax, R2.betay, R2.betaz)
print(R2.delta)

R3.delta = reward(T2, R3.betax, R3.betay, R3.betaz)
print(R2.delta)
```

```
0.76
0.76
0.76
```

plot

In [108]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is violet

GHZ

In [117]:

```
# a new circuit
q = QuantumRegister(9, 'q'); # qubits
#c0 = ClassicalRegister(6, 'c0');
c0 = ClassicalRegister(1, 'c0');
c1 = ClassicalRegister(1, 'c1');
c3 = ClassicalRegister(1, 'c3');
c4 = ClassicalRegister(1, 'c4');
c6 = ClassicalRegister(1, 'c6');
c7 = ClassicalRegister(1, 'c7');
qc_small = QuantumCircuit(q, c0, c1, c3, c4, c6, c7);
```

In [118]:

```
if ((R3.delta - R2.delta) <= 0.3) and ((R3.delta - R1.delta) <= 0.3) and ((R2.de
    # 0.3 rather than 0.1
    print("cometa")# GHZ for all
    qc_small.h(q[0]) # Hadamard
    qc_small.cx(q[0], q[1]) # CNOT
    qc_small.cx(q[0], q[2]) # CNOT
    qc_small.cx(q[0], q[3]) # CNOT
    qc_small.cx(q[0], q[4]) # CNOT
    qc_small.cx(q[0], q[5]) # CNOT
    qc_small.cx(q[0], q[6]) # CNOT
    qc_small.cx(q[0], q[7]) # CNOT
    # barrier
    qc_small.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6], q[7])
    # measures
    qc_small.measure(q[0],c0[0])
    qc_small.measure(q[3],c3[0])
    qc_small.measure(q[6],c6[0])
    qc_small.measure(q[1],c1[0])
    qc_small.measure(q[4],c4[0])
    qc_small.measure(q[7],c7[0])
    # draw circuit
    draw_circuit(qc_small)
    # definition of quantum simulator
    simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_
    qc_small = transpile(qc_small, simulator)
    # Run and get counts
    result = simulator.run(qc_small, shots=1024).result()
    counts_GHZ_all = result.get_counts(qc_small)
    counts_GHZ = counts_GHZ_all.most_frequent() # does not work if multiple stat
    # decide something if multiple states have the same count --> e.g., ``choose
    print(counts_GHZ_all)
    print(counts_GHZ)
    #plot_histogram(counts_GHZ_all, title='outcomes')
    #plot_histogram(counts_GHZ, title='outcomes')

print(counts_GHZ) # order: R3, R2, R1. Add some uncertainty?
# export as an array
str_ = counts_GHZ
arr_GHZ = str_.split(' ') # to split the string and avoid empty spaces as array
print(arr_GHZ)
# We do not need to update rewards; they should be done externally... excluded s
```

```
cometa

CircuitComposer(circuit=<qiskit.circuit.quantumcircuit.QuantumCircuit object at 0x
7fb65849a8e0>, editable=Fals…

{'0 0 0 0 0 0': 533, '1 1 1 1 1 1': 491}
0 0 0 0 0 0
0 0 0 0 0 0
['0', '0', '0', '0', '0', '0']
```

In [119]:

```
# define the new 0:


if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
    print('gosh')
    new_zero_betax = R1.betax + np.random.uniform(0,0.1)
    new_zero_alphax = 1 - R1.betax
    new_zero_betay = R1.betay + np.random.uniform(0,0.1)
    new_zero_alphay = 1 - R2.betay + np.random.uniform(0,0.1)

if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
    print('kinda')
    new_zero_betax = R2.betax + np.random.uniform(0,0.1)
    new_zero_alphax = 1 - R2.betax
    new_zero_betay = R2.betay + np.random.uniform(0,0.1)
    new_zero_alphay = 1 - R2.betay

if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
    print('uffdah')
    new_zero_betax = R3.betax + np.random.uniform(0,0.1)
    new_zero_alphax = 1 - R3.betax
    new_zero_betay = R3.betay + np.random.uniform(0,0.1)
    new_zero_alphay = 1 - R3.betay
```

```
gosh
kinda
uffdah
```

In [120]:

```
# define the new 1:


# flip thanks to the 'minus' sign?
# I had tried with if(R1... < R2...) etc., but it is not ok,
# because we need to initialize all elements.

if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
    print('gosh')
    new_one_betax = R1.betax - np.random.uniform(0,0.1)
    new_one_alphax = 1 - R1.betax
    new_one_betay = R1.betay - np.random.uniform(0,0.1)
    new_one_alphay = 1 - R2.betay

if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
    print('kinda')
    new_one_betax = R2.betax - np.random.uniform(0,0.1)
    new_one_alphax = 1 - R2.betax
    new_one_betay = R2.betay - np.random.uniform(0,0.1)
    new_one_alphay = 1 - R2.betay

if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
    print('uffdah')
    new_one_betax = R3.betax - np.random.uniform(0,0.1)
    new_one_alphax = 1 - R3.betax
    new_one_betay = R3.betay - np.random.uniform(0,0.1)
    new_one_alphay = 1 - R3.betay
```

```
gosh
kinda
uffdah
```

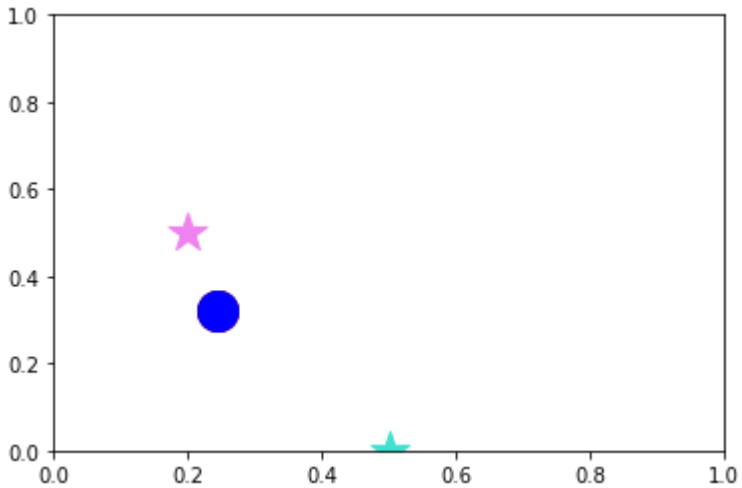
In [121]:

```
# choose the new 0 or new 1 according to the outcome of GHZ:

if (arr_GHZ[0] =='0'):
    R1.alphax = new_zero_alphax
    R1.betax = new_zero_betax
if (arr_GHZ[1] =='0'):
    R1.alphay = new_zero_alphay
    R1.betay = new_zero_betay
if (arr_GHZ[2] =='0'):
    R2.alphax = new_zero_alphax
    R2.betax = new_zero_betax
if (arr_GHZ[3] =='0'):
    R2.alphay = new_zero_alphay
    R2.betay = new_zero_betay
if (arr_GHZ[3] =='0'):
    R2.alphax = new_zero_alphax
    R2.betax = new_zero_betax
if (arr_GHZ[4] =='0'):
    R3.alphax = new_zero_alphax
    R3.betax = new_zero_betax
if (arr_GHZ[5] =='0'):
    R3.alphay = new_zero_alphay
    R3.betay = new_zero_betay
if (arr_GHZ[0] =='1'):
    R1.alphax = new_one_alphax
    R1.betax = new_one_betax
if (arr_GHZ[1] =='1'):
    R1.alphay = new_one_alphay
    R1.betay = new_one_betay
if (arr_GHZ[2] =='1'):
    R2.alphax = new_one_alphax
    R2.betax = new_one_betax
if (arr_GHZ[3] =='1'):
    R2.alphay = new_one_alphay
    R2.betay = new_one_betay
if (arr_GHZ[3] =='1'):
    R2.alphax = new_one_alphax
    R2.betax = new_one_betax
if (arr_GHZ[4] =='1'):
    R3.alphax = new_one_alphax
    R3.betax = new_one_betax
if (arr_GHZ[5] =='1'):
    R3.alphay = new_one_alphay
    R3.betay = new_one_betay


# new rewards:

R1.delta = reward(T2, R1.betax, R2.betay)
R2.delta = reward(T2, R2.betax, R3.betay)
R3.delta = reward(T2, R2.betax, R3.betay)
print(R1.delta, R2.delta, R3.delta)
```

```
0.81 0.81 0.81
```

In [122]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



R_1 is black, R_2 is red, and R_3 is blue, Target 1 is turquoise, Target 2 is violet

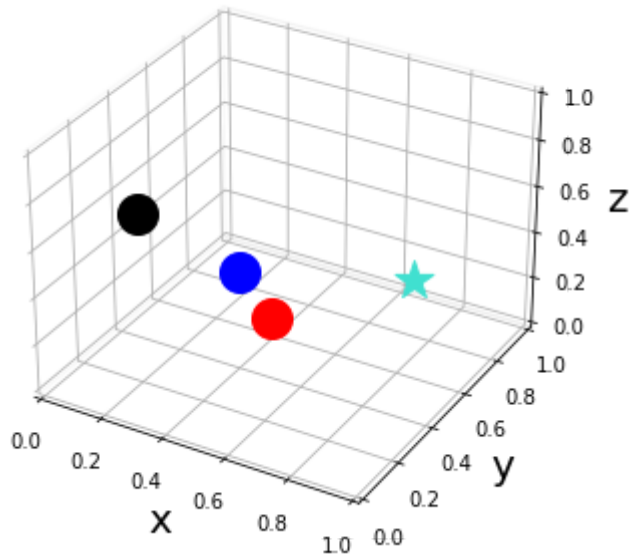
In [123]:

```
if (R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
print('yuk')
if (R1.delta > R2.delta and R1.delta > R3.delta):
    print('quokka')
    R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later: 0.
    R2.alphax = round(1 - R2.betax, 3)
    R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
    R2.alphay = round(1 - R2.betay, 3)
    R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
    R2.alphaz = round(1 - R2.betaz, 3)
    R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
    R3.alphax = round(1 - R2.betax, 3)
    R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
    R3.alphay = round(1 - R2.betay, 3)
    R3.betaz = round(R1.betay + np.random.uniform(0,0.1), 3)
    R3.alphaz = round(1 - R2.betaz, 3)
if (R2.delta > R1.delta and R2.delta > R3.delta):
    print('quagga')
    R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
    R1.alphax = round(1 - R1.betax, 3)
    R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
    R1.alphay = round(1 - R1.betay, 3)
    R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
    R1.alphaz = round(1 - R1.betaz, 3)
    R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
    R3.alphax = round(1 - R3.betax, 3)
    R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
    R3.alphay = round(1 - R3.betay, 3)
    R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
    R3.alphaz = round(1 - R3.betaz, 3)
if (R3.delta > R1.delta and R3.delta > R2.delta):
    print('quark')
    R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
    R1.alphax = round(1 - R1.betax, 3)
    R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
    R1.alphay = round(1 - R1.betay, 3)
    R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
    R1.alphaz = round(1 - R1.betaz, 3)
    R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
    R2.alphax = round(1 - R2.betax, 3)
    R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
    R2.alphay = round(1 - R2.betay, 3)
    R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
    R2.alphaz = round(1 - R2.betaz, 3)

delta = reward(T2, R1.betax, R1.betay, R1.betaz)
rint(R1.delta)

delta = reward(T2, R2.betax, R2.betay, R2.betaz)
rint(R2.delta)

delta = reward(T2, R3.betax, R3.betay, R3.betaz)
rint(R2.delta)
```

```
yuk
0.81
0.81
0.81
```

In [57]:

```
fig = plt.figure()

ax = Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)

ax.set_xlim3d(0, 1)
ax.set_ylim3d(0, 1)
ax.set_zlim3d(0, 1)

ax.xaxis.pane.fill = False
ax.yaxis.pane.fill = False
ax.zaxis.pane.fill = False

ax.set_xlabel('x', fontsize=20)
ax.set_ylabel('y', fontsize=20)
ax.set_zlabel('z', fontsize=20) # r'\alpha'

ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black
ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue'
# ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')

plt.show()

# find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
1
```

In [ ]:

```
1
```