

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]:

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

In [2]:

```
1 from ibm_quantum_widgets import CircuitComposer
2 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, Aer, exec
3 import numpy as np
4 from numpy import pi
5 from ibm_quantum_widgets import draw_circuit
6 from qiskit.providers.aer import QasmSimulator
7 from qiskit.utils import QuantumInstance
8 from qiskit.visualization import plot_histogram, plot_state_qsphere
9 from qiskit import *
10 import random
11 import matplotlib.pyplot as plt
12 import pylab
13 import pandas as pd
14 from sklearn import preprocessing
15 import collections
16 from collections import Counter
```

In [3]:

```
1 # from: https://stackoverflow.com/questions/39298928/play-multiple-sounds-at-the
2
3 from pydub import AudioSegment
4 from pydub.playback import play
```

In [4]:

```
1 # 3d graph: Adapted from https://stackoverflow.com/questions/12423601/simplest-v
2
3 import sys
4 import matplotlib
5 import matplotlib.pyplot as plt
6 from matplotlib.ticker import MaxNLocator
7 from matplotlib import cm
8 from mpl_toolkits.mplot3d import Axes3D
9 import numpy
10 from numpy import array
11 from scipy import newaxis
```

In [5]:

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

In [6]:

```
1 from mayavi import mlab
2 mlab.options.offscreen = True
3 mlab.test_contour3d()
4 mlab.savefig('example.png')
5 mlab.draw()
6 mlab.show()
7
8 # now it works :)
```

In [7]:

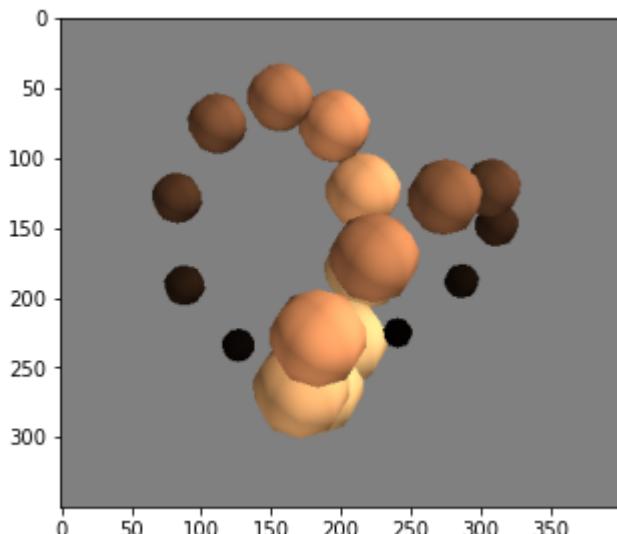
```
1 # example from https://stackoverflow.com/questions/67045944/embed-mayavi-into-a-
2
3 mlab.clf()
4
5 from mayavi import mlab
6 mlab.init_notebook()
7 import numpy as np
8
9 class Test:
10     def __init__(self):
11         self.fig = mlab.figure()
12         self._add_data()
13
14     def _add_data(self):
15         pi = np.pi
16         cos = np.cos
17         sin = np.sin
18         dphi, dtheta = pi / 250.0, pi / 250.0
19         [phi, theta] = np.mgrid[0:pi + dphi * 1.5:dphi,
20                               0:2 * pi + dtheta * 1.5:dtheta]
21         m0 = 4; m1 = 3; m2 = 2; m3 = 3
22         m4 = 6; m5 = 2; m6 = 6; m7 = 4
23         r = sin(m0 * phi) ** m1 + cos(m2 * phi) ** m3 + \
24             sin(m4 * theta) ** m5 + cos(m6 * theta) ** m7
25         x = r * sin(phi) * cos(theta)
26         y = r * cos(phi)
27         z = r * sin(phi) * sin(theta)
28         self.surf = mlab.mesh(x, y, z, figure=self.fig)
29
30     def show(self):
31         return self.surf
32
33 t=Test()
34 t.show()
```

Notebook initialized with ipy backend.



In [8]:

```
1 mlab.clf()
2
3 import numpy, pylab, mayavi, mayavi.mlab
4 import matplotlib.pyplot as plt
5
6 t = numpy.linspace(0, 4 * numpy.pi, 20)
7 cos,sin = numpy.cos, numpy.sin
8
9 x = sin(2 * t)
10 y = cos(t)
11 z = cos(2 * t)
12 s = 2 + sin(t)
13 mayavi.mlab.points3d(x, y, z, s, colormap="copper", scale_factor=.25)
14
15 arr = mayavi.mlab.screenshot()
16 fig = plt.figure(figsize=(5, 5))
17 pylab.imshow(arr)
18
19 plt.show()
20
```

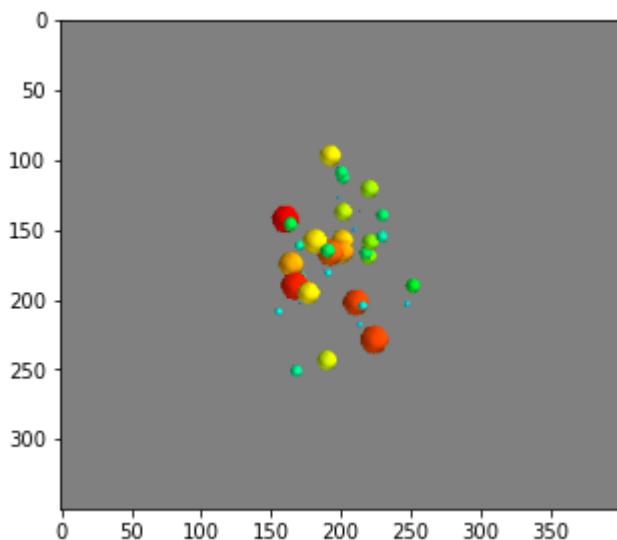


In [9]:

```

1 mlab.clf()
2
3 import numpy, pylab, mayavi, mayavi.mlab
4 import matplotlib.pyplot as plt
5
6 x, y, z, value = np.random.random((4, 40))
7 mlab.points3d(x, y, z, value)
8
9 arr = mayavi.mlab.screenshot()
10 fig = plt.figure(figsize=(5, 5))
11 pylab.imshow(arr)
12
13 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]:

```

1 q = QuantumRegister(7, 'q'); # qubits # changed to 9, formerly 15
2 m3 = ClassicalRegister(1, 'c1'); # classical bits (separated is better)
3 m4 = ClassicalRegister(1, 'c2');
4 m5 = ClassicalRegister(1, 'c3');
5 m6 = ClassicalRegister(1, 'c4');
6
7 qc3 = QuantumCircuit(q, m3, m4, m5, m6); # to reach the target
8 qc4 = QuantumCircuit(q, m3, m4, m5, m6); # to get back to the nest

```

In [11]:

```

1 class Target:
2     def __init__(self, name, x, y, z): # no indetermination in the target's position
3         self.name = name
4         self.x = x
5         self.y = y
6         self.z = z

```

In [12]:

```

1 T = Target("T", 0.8, 0.8, 0.2) # deep in the ocean
2
3 # for getting back to the beginning
4 T2 = Target("T2", 0.2, 0.2, 1.0) # back to the ship

```

In [13]:

```

1 def reward(T, betax, betay, betaz):
2     r = round(1 - ((T.x - betax)**2 + (T.y - betay)**2 + (T.z - betaz)**2)**0.5,
3               # the closer the target, the less the distance, the higher the reward
4     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]:

```

1 class Robot:
2     def __init__(self, name, alphax, betax, alphay, betay, alphaz, betaz, gamma, delta):
3         self.name = name
4         self.alphax = alphax
5         self.betax = betax
6         self.alphay = alphay
7         self.betay = betay
8         self.alphaz = alphaz
9         self.betaz = betaz
10        delta = reward(T, betax, betay, betaz)
11        gamma = round(1 - delta, 2)
12        self.gamma = gamma
13        self.delta = delta

```

In [15]:

```

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

```

Out[15]:

0.02

In [16]:

```

1 # manual intervention needed here to avoid circularity
2 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]:

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

In [18]:

```
1 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]:

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

Out[19]:

0.06

In [20]:

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

Out[20]:

0.94

In [21]:

```
1 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]:

```
1 R2.delta, R2.gamma, R2.alphax, R2.betax, R2.alphay, R2.betay, R2.alphaz, R2.bet
```

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]:

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

Out[23]:

0.01

In [24]:

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

Out[24]:

0.99

In [25]:

```
1 R3 = Robot("R3", 0.7, 0.3, 0.9, 0.1, 0.3, 0.7, 0.99, 0.01) # to be updated by ha
2 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]:

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

Out[26]:

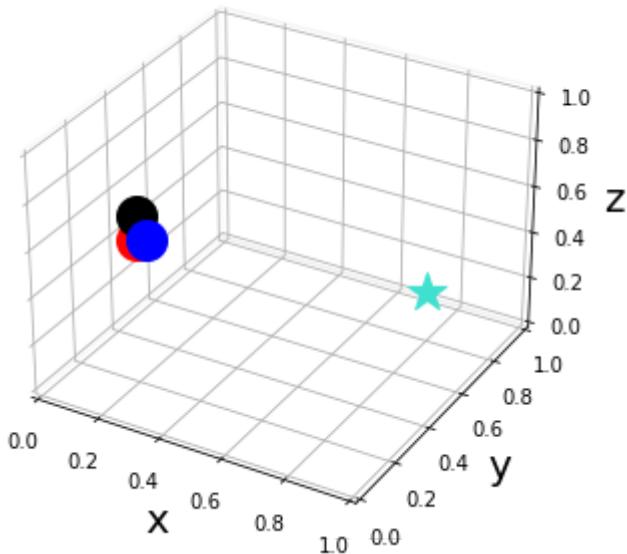
(0.99, 0.01)

In [27]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26
27 # find how to automatically create trajectories: maybe LinePlot between R1, R2,

```



In [28]:

```
1 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]:

```

1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):

```

```

60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66     if (R1.betax > 0.3 and R1.betax <= 0.5):
67         if (R1.betay < 0.5): # (R1.betay == 1):
68             audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69             print("tD#2")
70         if (R1.betay >= 0.5):
71             audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72             print("tA2")
73     if (R1.betax > 0.5 and R1.betax <= 0.64):
74         if (R1.betay < 0.5):
75             audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76             print("tE2")
77         if (R1.betay >= 0.5):
78             audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79             print("tG#2")
80     if (R1.betax > 0.64 and R1.betax <= 0.84):
81         if (R1.betay < 0.5):
82             audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83             print("tF2")
84         if (R1.betay >= 0.5):
85             audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86             print("tG2")
87     if (R1.betax > 0.84 and R1.betax <= 1):
88         #if (R1.betay == 0.5):
89         audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90         print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")

```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time1_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time1_ = mixed_time1_.overlay(audio3)      # further combine , superi
285
286 mixed_time1_.export("notes_/mixed_time1.mp3", format='mp3') # export mixed audi
287 play(mixed_time1)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

```

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

```

```

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmp11a97skn.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]:

```
1 # 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]:

```
1 if (R1.delta and R2.delta and R3.delta) >= 0.8 and (R1.delta and R2.delta and R3.delta) < 0.99:
2     print("ciao ciao")
3     R1.alphax = round(np.random.uniform(0,0.2), 3) # slightly shuffle position of R1
4     R1.betax = round(1 - R1.alphax, 3)
5     #R1.alphay = round(np.random.uniform(0,0.2), 3) # slightly shuffle position of R1
6     #R1.betay = round(1 - R1.alphay, 3)
7     print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
```

In [34]:

```
1 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]:

```
1 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
2 print(R1.delta)
3
4 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
5 print(R2.delta)
6
7 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
8 print(R3.delta)
```

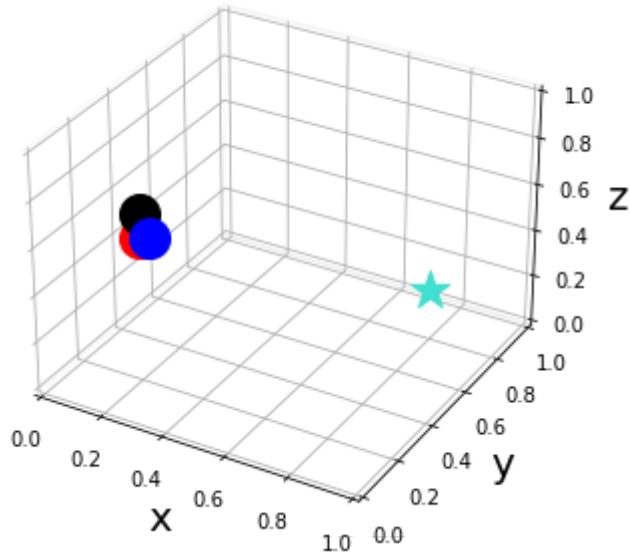
```
0.02
0.06
0.01
```

In [36]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```
1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time2_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time2_ = mixed_time2_.overlay(audio3)      # further combine , superi
285
286 mixed_time2_.export("notes_/mixed_time2.mp3", format='mp3') # export mixed audi
287 play(mixed_time2)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

```

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]:

```

1 # threshold for initial reward
2 # random fluctuations
3
4 if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
5     print("SOS")
6     # R1
7     R1.alphax = round(np.random.uniform(0,0.9), 3)
8     R1.betax = round(1 - R1.alphax, 3)
9     print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
10    R1.alphay = round(np.random.uniform(0,0.9), 3)
11    R1.betay = round(1 - R1.alphay, 3)
12    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
13    R1.alphaz = round(np.random.uniform(0,0.9), 3)
14    R1.betaz = round(1 - R1.alphaz, 3)
15    print("the new z-positions for R1 are: ", R1.alphaz, R1.betaz)
16    # R2
17    R2.alphax = round(np.random.uniform(0,0.9), 3)
18    R2.betax = round(1 - R2.alphax, 3)
19    print("the new x-positions for R2 are: ", R2.alphax, R2.betax)
20    R2.alphay = round(np.random.uniform(0,0.9), 3)
21    R2.betay = round(1 - R2.alphay, 3)
22    print("the new y-positions for R2 are: ", R2.alphay, R2.betay)
23    R2.alphaz = round(np.random.uniform(0,0.9), 3)
24    R2.betaz = round(1 - R2.alphaz, 3)
25    print("the new z-positions for R2 are: ", R2.alphaz, R2.betaz)
26    # R3
27    R3.alphax = round(np.random.uniform(0,0.9), 3)
28    R3.betax = round(1 - R3.alphax, 3)
29    print("the new x-positions for R3 are: ", R3.alphax, R3.betax)
30    R3.alphay = round(np.random.uniform(0,0.9), 3)
31    R3.betay = round(1 - R3.alphay, 3)
32    print("the new y-positions for R3 are: ", R3.alphay, R3.betay)
33    R3.alphaz = round(np.random.uniform(0,0.9), 3)
34    R3.betaz = round(1 - R3.alphaz, 3)
35    print("the new z-positions for R3 are: ", R3.alphaz, R3.betaz)
36
37 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
38 R1.gamma = 1 - R1.delta
39 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
40 R2.gamma = 1 - R2.delta
41 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
42 R3.gamma = 1 - R3.delta
43 print(R1.delta, R2.delta, R3.delta)

```

SOS

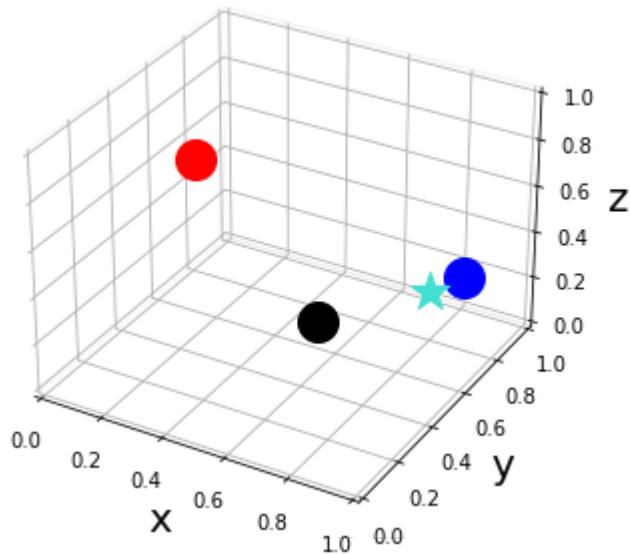
```

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 z-positions for R3 are: 0.767 0.235
0.61 0.04 0.89

```

In [39]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [40]:

```

1
2 # audio 1, R_1
3
4 if(R1.betaz >= 0.5):
5     if (R1.betax == 0):
6         if (R1.betay == 0.5):
7             audio1 = AudioSegment.from_file("notes_/tC.mp3")
8             print("tC")
9     if (R1.betax > 0 and R1.betax <= 0.17):
10        if (R1.betay < 0.5):
11            audio1 = AudioSegment.from_file("notes_/tB.mp3")
12            print("tB")
13        if (R1.betay >= 0.5):
14            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
15            print("tC#")
16    if (R1.betax > 0.17 and R1.betax <= 0.3):
17        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
18            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
19            print("tA#")
20        if (R1.betay >= 0.5):
21            audio1 = AudioSegment.from_file("notes_/tD.mp3")
22            print("tD")
23    if (R1.betax > 0.3 and R1.betax <= 0.5):
24        if (R1.betay < 0.5): # (R1.betay == 1):
25            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
26            print("tD#")
27        if (R1.betay >= 0.5):
28            audio1 = AudioSegment.from_file("notes_/tA.mp3")
29            print("tA")
30    if (R1.betax > 0.5 and R1.betax <= 0.64):
31        if (R1.betay < 0.5):
32            audio1 = AudioSegment.from_file("notes_/tE.mp3")
33            print("tE")
34        if (R1.betay >= 0.5):
35            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
36            print("tG#")
37    if (R1.betax > 0.64 and R1.betax <= 0.84):
38        if (R1.betay < 0.5):
39            audio1 = AudioSegment.from_file("notes_/tF.mp3")
40            print("tF")
41        if (R1.betay >= 0.5):
42            audio1 = AudioSegment.from_file("notes_/tG.mp3")
43            print("tG")
44    if (R1.betax > 0.84 and R1.betax <= 1):
45        #if (R1.betay == 0.5):
46        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
47        print("tF#")
48 if(R1.betaz < 0.5):
49     if (R1.betax == 0):
50         if (R1.betay == 0.5):
51             audio1 = AudioSegment.from_file("notes_/tc2.mp3")
52             print("tc2")
53     if (R1.betax > 0 and R1.betax <= 0.17):
54         if (R1.betay < 0.5):
55             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
56             print("tB2")
57         if (R1.betay >= 0.5):
58             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
59             print("tC#2")

```

```
60     if (R1.betax > 0.17 and R1.betax <= 0.3):
61         if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
62             audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
63             print("tA#2")
64         if (R1.betay >= 0.5):
65             audio1 = AudioSegment.from_file("notes_/tD2.mp3")
66             print("tD2")
67     if (R1.betax > 0.3 and R1.betax <= 0.5):
68         if (R1.betay < 0.5): # (R1.betay == 1):
69             audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
70             print("tD#2")
71         if (R1.betay >= 0.5):
72             audio1 = AudioSegment.from_file("notes_/tA2.mp3")
73             print("tA2")
74     if (R1.betax > 0.5 and R1.betax <= 0.64):
75         if (R1.betay < 0.5):
76             audio1 = AudioSegment.from_file("notes_/tE2.mp3")
77             print("tE2")
78         if (R1.betay >= 0.5):
79             audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
80             print("tG#2")
81     if (R1.betax > 0.64 and R1.betax <= 0.84):
82         if (R1.betay < 0.5):
83             audio1 = AudioSegment.from_file("notes_/tF2.mp3")
84             print("tF2")
85         if (R1.betay >= 0.5):
86             audio1 = AudioSegment.from_file("notes_/tG2.mp3")
87             print("tG2")
88     if (R1.betax > 0.84 and R1.betax <= 1):
89         #if (R1.betay == 0.5):
90         audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
91         print("tF#2")
92
93
94
95 # CHANGE from this point
96
97
98 # audio 2, R_2
99
100 if(R2.betaz < 0.5):
101     if (R2.betax == 0):
102         if (R2.betay == 0.5):
103             audio2 = AudioSegment.from_file("notes_/fc2.mp3")
104             print("fc2")
105     if (R2.betax > 0 and R2.betax <= 0.17):
106         if (R2.betay < 0.5):
107             audio2 = AudioSegment.from_file("notes_/fb2.mp3")
108             print("fb2")
109         if (R2.betay >= 0.5):
110             audio2 = AudioSegment.from_file("notes_/fc#2.mp3")
111             print("fc#2")
112     if (R2.betax > 0.17 and R2.betax <= 0.3):
113         if (R2.betay < 0.5):
114             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
115             print("fA#2")
116         if (R2.betay >= 0.5):
117             audio2 = AudioSegment.from_file("notes_/fd2.mp3")
118             print("fd2")
119     if (R2.betax > 0.3 and R2.betax <= 0.5):
120         if (R2.betay < 0.5): # (R1.betay == 1):
```

```
121     audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
122     print("fD#2")
123 if (R2.betay >= 0.5):
124     audio2 = AudioSegment.from_file("notes_/fA2.mp3")
125     print("fA2")
126 if (R2.betax > 0.5 and R2.betax <= 0.64):
127     if (R2.betay < 0.5):
128         audio2 = AudioSegment.from_file("notes_/fE2.mp3")
129         print("fE2")
130     if (R2.betay >= 0.5):
131         audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
132         print("fG#2")
133 if (R2.betax > 0.64 and R2.betax <= 0.84):
134     if (R2.betay < 0.5):
135         audio2 = AudioSegment.from_file("notes_/fF2.mp3")
136         print("fF2")
137     if (R2.betay >= 0.5):
138         audio2 = AudioSegment.from_file("notes_/fG2.mp3")
139         print("fG2")
140 if (R2.betax > 0.84 and R2.betax <= 1):
141     #if (R2.betay == 0.5):
142     audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
143     print("fF#2")
144 if(R2.betaz >= 0.5):
145     if (R2.betax == 0):
146         if (R2.betay == 0.5):
147             audio2 = AudioSegment.from_file("notes_/fC.mp3")
148             print("fC")
149     if (R2.betax > 0 and R2.betax <= 0.17):
150         if (R2.betay < 0.5):
151             audio2 = AudioSegment.from_file("notes_/fB.mp3")
152             print("fB")
153         if (R2.betay >= 0.5):
154             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
155             print("fC#")
156     if (R2.betax > 0.17 and R2.betax <= 0.3):
157         if (R2.betay < 0.5):
158             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
159             print("fA#")
160         if (R2.betay >= 0.5):
161             audio2 = AudioSegment.from_file("notes_/fD.mp3")
162             print("fD")
163     if (R2.betax > 0.3 and R2.betax <= 0.5):
164         if (R2.betay < 0.5): # (R1.betay == 1):
165             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
166             print("fD#")
167         if (R2.betay >= 0.5):
168             audio2 = AudioSegment.from_file("notes_/fA.mp3")
169             print("fA")
170     if (R2.betax > 0.5 and R2.betax <= 0.64):
171         if (R2.betay < 0.5):
172             audio2 = AudioSegment.from_file("notes_/fE.mp3")
173             print("fE")
174         if (R2.betay >= 0.5):
175             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
176             print("fG#")
177     if (R2.betax > 0.64 and R2.betax <= 0.84):
178         if (R2.betay < 0.5):
179             audio2 = AudioSegment.from_file("notes_/fF.mp3")
180             print("fF")
181         if (R2.betay >= 0.5):
```

```
182     audio2 = AudioSegment.from_file("notes_/fG.mp3")
183     print("fG")
184 if (R2.betax > 0.84 and R2.betax <= 1):
185     #if (R2.betay == 0.5):
186     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
187     print("fF#")
188
189
190
191
192
193 # audio 3, R_3
194
195 if (R3.betaz >= 0.5):
196     if (R3.betax == 0):
197         if (R3.betay == 0.5):
198             audio3 = AudioSegment.from_file("notes_/cC.mp3")
199             print("cC")
200     if (R3.betax > 0 and R3.betax <= 0.17):
201         if (R3.betay < 0.5):
202             audio3 = AudioSegment.from_file("notes_/cB.mp3")
203             print("cB")
204         if (R3.betay >= 0.5):
205             audio3 = AudioSegment.from_file("notes_/cC#.mp3")
206             print("cC#")
207     if (R3.betax > 0.17 and R3.betax <= 0.3):
208         if (R3.betay < 0.5):
209             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
210             print("cA#")
211         if (R3.betay >= 0.5):
212             audio3 = AudioSegment.from_file("notes_/cD.mp3")
213             print("cD")
214     if (R3.betax > 0.3 and R3.betax <= 0.5):
215         if (R3.betay < 0.5):
216             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
217             print("cD#")
218         if (R3.betay >= 0.5):
219             audio3 = AudioSegment.from_file("notes_/cA.mp3")
220             print("cA")
221     if (R3.betax > 0.5 and R3.betax <= 0.64):
222         if (R3.betay < 0.5):
223             audio3 = AudioSegment.from_file("notes_/cE.mp3")
224             print("cE")
225         if (R3.betay >= 0.5):
226             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
227             print("cG#")
228     if (R3.betax > 0.64 and R3.betax <= 0.84):
229         if (R3.betay < 0.5):
230             audio3 = AudioSegment.from_file("notes_/cF.mp3")
231             print("cF")
232         if (R3.betay >= 0.5):
233             audio3 = AudioSegment.from_file("notes_/cG.mp3")
234             print("cG")
235     if (R3.betax > 0.84 and R3.betax <= 1):
236         #if (R3.betay == 0.5):
237         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
238         print("cF#")
239     if (R3.betaz < 0.5):
240         if (R3.betax == 0):
241             if (R3.betay == 0.5):
242                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
```

```

243     print("cC2")
244 if (R3.betax > 0 and R3.betax <= 0.17):
245     if (R3.betay < 0.5):
246         audio3 = AudioSegment.from_file("notes_/cB2.mp3")
247         print("cB2")
248     if (R3.betay >= 0.5):
249         audio3 = AudioSegment.from_file("notes_/cc#2.mp3")
250         print("cC#2")
251 if (R3.betax > 0.17 and R3.betax <= 0.3):
252     if (R3.betay < 0.5):
253         audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
254         print("cA#2")
255     if (R3.betay >= 0.5):
256         audio3 = AudioSegment.from_file("notes_/cd2.mp3")
257         print("cd2")
258 if (R3.betax > 0.3 and R3.betax <= 0.5):
259     if (R3.betay < 0.5):
260         audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
261         print("cD#2")
262     if (R3.betay >= 0.5):
263         audio3 = AudioSegment.from_file("notes_/cA2.mp3")
264         print("cA2")
265 if (R3.betax > 0.5 and R3.betax <= 0.64):
266     if (R3.betay < 0.5):
267         audio3 = AudioSegment.from_file("notes_/cE2.mp3")
268         print("cE2")
269     if (R3.betay >= 0.5):
270         audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
271         print("cG#2")
272 if (R3.betax > 0.64 and R3.betax <= 0.84):
273     if (R3.betay < 0.5):
274         audio3 = AudioSegment.from_file("notes_/cF2.mp3")
275         print("cF2")
276     if (R3.betay >= 0.5):
277         audio3 = AudioSegment.from_file("notes_/cG2.mp3")
278         print("cG2")
279 if (R3.betax > 0.84 and R3.betax <= 1):
280     #if (R3.betay == 0.5):
281     audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
282     print("cF#2")
283
284 mixed_time3_ = audio1.overlay(audio2)           # combine , superimpose audio fi
285 mixed_time3 = mixed_time3_.overlay(audio3)       # further combine , superi
286
287 mixed_time3.export("notes_/mixed_time3.mp3", format='mp3') # export mixed audi
288 play(mixed_time3)                             # play mixed audio file
289 # change this line at each time point, so in the end we can get a little piece
290

```

tE2
fA#
cF#2

Could not import the PyAudio C module '_portaudio'.

```

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmp1516pxrk.
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]:

```

1 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
6         R2.alphax = round(1 - R2.betax, 3)
7         R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
8         R2.alphay = round(1 - R2.betay, 3)
9         R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
10        R2.alphaz = round(1 - R2.betaz, 3)
11        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
12        R3.alphax = round(1 - R2.betax, 3)
13        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
14        R3.alphay = round(1 - R2.betay, 3)
15        R3.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
16        R3.alphaz = round(1 - R2.betaz, 3)
17    if (R2.delta > R1.delta and R2.delta > R3.delta):
18        print('quagga')
19        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
20        R1.alphax = round(1 - R1.betax, 3)
21        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
22        R1.alphay = round(1 - R1.betay, 3)
23        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
24        R1.alphaz = round(1 - R1.betaz, 3)
25        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
26        R3.alphax = round(1 - R3.betax, 3)
27        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
28        R3.alphay = round(1 - R3.betay, 3)
29        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
30        R3.alphaz = round(1 - R3.betaz, 3)
31    if (R3.delta > R1.delta and R3.delta > R2.delta):
32        print('quark')
33        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
34        R1.alphax = round(1 - R1.betax, 3)
35        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
36        R1.alphay = round(1 - R1.betay, 3)
37        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
38        R1.alphaz = round(1 - R1.betaz, 3)
39        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
40        R2.alphax = round(1 - R2.betax, 3)
41        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
42        R2.alphay = round(1 - R2.betay, 3)
43        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
44        R2.alphaz = round(1 - R2.betaz, 3)
45
46 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
47 print(R1.delta)
48
49 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
50 print(R2.delta)
51
52 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
53 print(R2.delta)

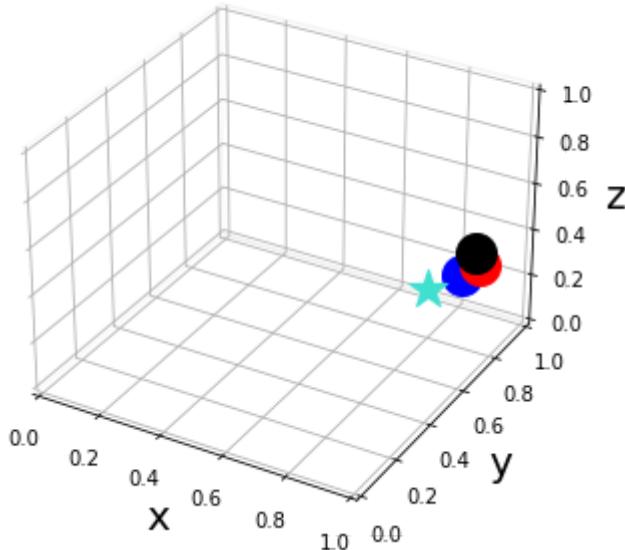
```

yuk
quark
0.81

0.83
0.83

In [43]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [44]:

```

1
2 # audio 1, R_1
3
4 if(R1.betaz >= 0.5):
5     if (R1.betax == 0):
6         if (R1.betay == 0.5):
7             audio1 = AudioSegment.from_file("notes_/tC.mp3")
8             print("tC")
9     if (R1.betax > 0 and R1.betax <= 0.17):
10        if (R1.betay < 0.5):
11            audio1 = AudioSegment.from_file("notes_/tB.mp3")
12            print("tB")
13        if (R1.betay >= 0.5):
14            audio1 = AudioSegment.from_file("notes_/tC#.mp3")
15            print("tC#")
16    if (R1.betax > 0.17 and R1.betax <= 0.3):
17        if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
18            audio1 = AudioSegment.from_file("notes_/tA#.mp3")
19            print("tA#")
20        if (R1.betay >= 0.5):
21            audio1 = AudioSegment.from_file("notes_/tD.mp3")
22            print("tD")
23    if (R1.betax > 0.3 and R1.betax <= 0.5):
24        if (R1.betay < 0.5): # (R1.betay == 1):
25            audio1 = AudioSegment.from_file("notes_/tD#.mp3")
26            print("tD#")
27        if (R1.betay >= 0.5):
28            audio1 = AudioSegment.from_file("notes_/tA.mp3")
29            print("tA")
30    if (R1.betax > 0.5 and R1.betax <= 0.64):
31        if (R1.betay < 0.5):
32            audio1 = AudioSegment.from_file("notes_/tE.mp3")
33            print("tE")
34        if (R1.betay >= 0.5):
35            audio1 = AudioSegment.from_file("notes_/tG#.mp3")
36            print("tG#")
37    if (R1.betax > 0.64 and R1.betax <= 0.84):
38        if (R1.betay < 0.5):
39            audio1 = AudioSegment.from_file("notes_/tF.mp3")
40            print("tF")
41        if (R1.betay >= 0.5):
42            audio1 = AudioSegment.from_file("notes_/tG.mp3")
43            print("tG")
44    if (R1.betax > 0.84 and R1.betax <= 1):
45        #if (R1.betay == 0.5):
46        audio1 = AudioSegment.from_file("notes_/tF#.mp3")
47        print("tF#")
48 if(R1.betaz < 0.5):
49     if (R1.betax == 0):
50         if (R1.betay == 0.5):
51             audio1 = AudioSegment.from_file("notes_/tc2.mp3")
52             print("tc2")
53     if (R1.betax > 0 and R1.betax <= 0.17):
54         if (R1.betay < 0.5):
55             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
56             print("tB2")
57         if (R1.betay >= 0.5):
58             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
59             print("tC#2")

```

```
60     if (R1.betax > 0.17 and R1.betax <= 0.3):
61         if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
62             audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
63             print("tA#2")
64         if (R1.betay >= 0.5):
65             audio1 = AudioSegment.from_file("notes_/tD2.mp3")
66             print("tD2")
67     if (R1.betax > 0.3 and R1.betax <= 0.5):
68         if (R1.betay < 0.5): # (R1.betay == 1):
69             audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
70             print("tD#2")
71         if (R1.betay >= 0.5):
72             audio1 = AudioSegment.from_file("notes_/tA2.mp3")
73             print("tA2")
74     if (R1.betax > 0.5 and R1.betax <= 0.64):
75         if (R1.betay < 0.5):
76             audio1 = AudioSegment.from_file("notes_/tE2.mp3")
77             print("tE2")
78         if (R1.betay >= 0.5):
79             audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
80             print("tG#2")
81     if (R1.betax > 0.64 and R1.betax <= 0.84):
82         if (R1.betay < 0.5):
83             audio1 = AudioSegment.from_file("notes_/tF2.mp3")
84             print("tF2")
85         if (R1.betay >= 0.5):
86             audio1 = AudioSegment.from_file("notes_/tG2.mp3")
87             print("tG2")
88     if (R1.betax > 0.84 and R1.betax <= 1):
89         #if (R1.betay == 0.5):
90         audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
91         print("tF#2")
92
93
94
95 # CHANGE from this point
96
97
98 # audio 2, R_2
99
100 if(R2.betaz < 0.5):
101     if (R2.betax == 0):
102         if (R2.betay == 0.5):
103             audio2 = AudioSegment.from_file("notes_/fc2.mp3")
104             print("fc2")
105     if (R2.betax > 0 and R2.betax <= 0.17):
106         if (R2.betay < 0.5):
107             audio2 = AudioSegment.from_file("notes_/fb2.mp3")
108             print("fb2")
109         if (R2.betay >= 0.5):
110             audio2 = AudioSegment.from_file("notes_/fc#2.mp3")
111             print("fc#2")
112     if (R2.betax > 0.17 and R2.betax <= 0.3):
113         if (R2.betay < 0.5):
114             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
115             print("fA#2")
116         if (R2.betay >= 0.5):
117             audio2 = AudioSegment.from_file("notes_/fd2.mp3")
118             print("fd2")
119     if (R2.betax > 0.3 and R2.betax <= 0.5):
120         if (R2.betay < 0.5): # (R1.betay == 1):
```

```
121     audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
122     print("fD#2")
123 if (R2.betay >= 0.5):
124     audio2 = AudioSegment.from_file("notes_/fA2.mp3")
125     print("fA2")
126 if (R2.betax > 0.5 and R2.betax <= 0.64):
127     if (R2.betay < 0.5):
128         audio2 = AudioSegment.from_file("notes_/fE2.mp3")
129         print("fE2")
130     if (R2.betay >= 0.5):
131         audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
132         print("fG#2")
133 if (R2.betax > 0.64 and R2.betax <= 0.84):
134     if (R2.betay < 0.5):
135         audio2 = AudioSegment.from_file("notes_/fF2.mp3")
136         print("fF2")
137     if (R2.betay >= 0.5):
138         audio2 = AudioSegment.from_file("notes_/fG2.mp3")
139         print("fG2")
140 if (R2.betax > 0.84 and R2.betax <= 1):
141     #if (R2.betay == 0.5):
142     audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
143     print("fF#2")
144 if(R2.betaz >= 0.5):
145     if (R2.betax == 0):
146         if (R2.betay == 0.5):
147             audio2 = AudioSegment.from_file("notes_/fC.mp3")
148             print("fC")
149     if (R2.betax > 0 and R2.betax <= 0.17):
150         if (R2.betay < 0.5):
151             audio2 = AudioSegment.from_file("notes_/fB.mp3")
152             print("fB")
153         if (R2.betay >= 0.5):
154             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
155             print("fC#")
156     if (R2.betax > 0.17 and R2.betax <= 0.3):
157         if (R2.betay < 0.5):
158             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
159             print("fA#")
160         if (R2.betay >= 0.5):
161             audio2 = AudioSegment.from_file("notes_/fD.mp3")
162             print("fD")
163     if (R2.betax > 0.3 and R2.betax <= 0.5):
164         if (R2.betay < 0.5): # (R1.betay == 1):
165             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
166             print("fD#")
167         if (R2.betay >= 0.5):
168             audio2 = AudioSegment.from_file("notes_/fA.mp3")
169             print("fA")
170     if (R2.betax > 0.5 and R2.betax <= 0.64):
171         if (R2.betay < 0.5):
172             audio2 = AudioSegment.from_file("notes_/fE.mp3")
173             print("fE")
174         if (R2.betay >= 0.5):
175             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
176             print("fG#")
177     if (R2.betax > 0.64 and R2.betax <= 0.84):
178         if (R2.betay < 0.5):
179             audio2 = AudioSegment.from_file("notes_/fF.mp3")
180             print("fF")
181         if (R2.betay >= 0.5):
```

```
182     audio2 = AudioSegment.from_file("notes_/fG.mp3")
183     print("fG")
184 if (R2.betax > 0.84 and R2.betax <= 1):
185     #if (R2.betay == 0.5):
186     audio2 = AudioSegment.from_file("notes_/fF#.mp3")
187     print("fF#")
188
189
190
191
192
193 # audio 3, R_3
194
195 if (R3.betaz >= 0.5):
196     if (R3.betax == 0):
197         if (R3.betay == 0.5):
198             audio3 = AudioSegment.from_file("notes_/cC.mp3")
199             print("cC")
200     if (R3.betax > 0 and R3.betax <= 0.17):
201         if (R3.betay < 0.5):
202             audio3 = AudioSegment.from_file("notes_/cB.mp3")
203             print("cB")
204         if (R3.betay >= 0.5):
205             audio3 = AudioSegment.from_file("notes_/cC#.mp3")
206             print("cC#")
207     if (R3.betax > 0.17 and R3.betax <= 0.3):
208         if (R3.betay < 0.5):
209             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
210             print("cA#")
211         if (R3.betay >= 0.5):
212             audio3 = AudioSegment.from_file("notes_/cD.mp3")
213             print("cD")
214     if (R3.betax > 0.3 and R3.betax <= 0.5):
215         if (R3.betay < 0.5):
216             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
217             print("cD#")
218         if (R3.betay >= 0.5):
219             audio3 = AudioSegment.from_file("notes_/cA.mp3")
220             print("cA")
221     if (R3.betax > 0.5 and R3.betax <= 0.64):
222         if (R3.betay < 0.5):
223             audio3 = AudioSegment.from_file("notes_/cE.mp3")
224             print("cE")
225         if (R3.betay >= 0.5):
226             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
227             print("cG#")
228     if (R3.betax > 0.64 and R3.betax <= 0.84):
229         if (R3.betay < 0.5):
230             audio3 = AudioSegment.from_file("notes_/cF.mp3")
231             print("cF")
232         if (R3.betay >= 0.5):
233             audio3 = AudioSegment.from_file("notes_/cG.mp3")
234             print("cG")
235     if (R3.betax > 0.84 and R3.betax <= 1):
236         #if (R3.betay == 0.5):
237         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
238         print("cF#")
239     if (R3.betaz < 0.5):
240         if (R3.betax == 0):
241             if (R3.betay == 0.5):
242                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
```

```

243     print("cC2")
244 if (R3.betax > 0 and R3.betax <= 0.17):
245     if (R3.betay < 0.5):
246         audio3 = AudioSegment.from_file("notes_/cB2.mp3")
247         print("cB2")
248     if (R3.betay >= 0.5):
249         audio3 = AudioSegment.from_file("notes_/cc#2.mp3")
250         print("cC#2")
251 if (R3.betax > 0.17 and R3.betax <= 0.3):
252     if (R3.betay < 0.5):
253         audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
254         print("cA#2")
255     if (R3.betay >= 0.5):
256         audio3 = AudioSegment.from_file("notes_/cd2.mp3")
257         print("cd2")
258 if (R3.betax > 0.3 and R3.betax <= 0.5):
259     if (R3.betay < 0.5):
260         audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
261         print("cD#2")
262     if (R3.betay >= 0.5):
263         audio3 = AudioSegment.from_file("notes_/cA2.mp3")
264         print("cA2")
265 if (R3.betax > 0.5 and R3.betax <= 0.64):
266     if (R3.betay < 0.5):
267         audio3 = AudioSegment.from_file("notes_/cE2.mp3")
268         print("cE2")
269     if (R3.betay >= 0.5):
270         audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
271         print("cG#2")
272 if (R3.betax > 0.64 and R3.betax <= 0.84):
273     if (R3.betay < 0.5):
274         audio3 = AudioSegment.from_file("notes_/cF2.mp3")
275         print("cF2")
276     if (R3.betay >= 0.5):
277         audio3 = AudioSegment.from_file("notes_/cG2.mp3")
278         print("cG2")
279 if (R3.betax > 0.84 and R3.betax <= 1):
280     #if (R3.betay == 0.5):
281     audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
282     print("cF#2")
283
284 mixed_time4_ = audio1.overlay(audio2)           # combine , superimpose audio fi
285 mixed_time4_ = mixed_time4_.overlay(audio3)      # further combine , superi
286
287 mixed_time4_.export("notes_/mixed_time4.mp3", format='mp3') # export mixed audi
288 play(mixed_time4)                            # play mixed audio file
289 # change this line at each time point, so in the end we can get a little piece
290

```

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]:

```

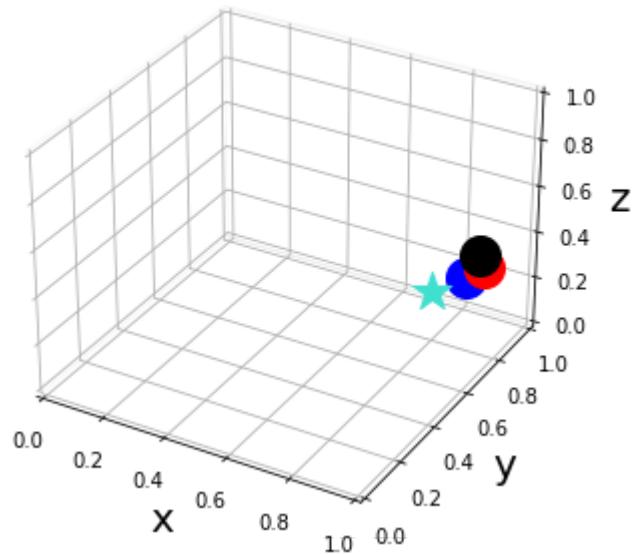
1 # Another round of SOS re-shuffle
2
3 # threshold for initial reward
4 # random fluctuations
5
6 if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
7     print("SOS")
8     # R1
9     R1.alphax = round(np.random.uniform(0,0.9), 3)
10    R1.betax = round(1 - R1.alphax, 3)
11    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
12    R1.alphay = round(np.random.uniform(0,0.9), 3)
13    R1.betay = round(1 - R1.alphay, 3)
14    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
15    R1.alphaz = round(np.random.uniform(0,0.9), 3)
16    R1.betaz = round(1 - R1.alphaz, 3)
17    print("the new z-positions for R1 are: ", R1.alphaz, R1.betaz)
18    # R2
19    R2.alphax = round(np.random.uniform(0,0.9), 3)
20    R2.betax = round(1 - R2.alphax, 3)
21    print("the new x-positions for R2 are: ", R2.alphax, R2.betax)
22    R2.alphay = round(np.random.uniform(0,0.9), 3)
23    R2.betay = round(1 - R2.alphay, 3)
24    print("the new y-positions for R2 are: ", R2.alphay, R2.betay)
25    R2.alphaz = round(np.random.uniform(0,0.9), 3)
26    R2.betaz = round(1 - R2.alphaz, 3)
27    print("the new z-positions for R2 are: ", R2.alphaz, R2.betaz)
28    # R3
29    R3.alphax = round(np.random.uniform(0,0.9), 3)
30    R3.betax = round(1 - R3.alphax, 3)
31    print("the new x-positions for R3 are: ", R3.alphax, R3.betax)
32    R3.alphay = round(np.random.uniform(0,0.9), 3)
33    R3.betay = round(1 - R3.alphay, 3)
34    print("the new y-positions for R3 are: ", R3.alphay, R3.betay)
35    R3.alphaz = round(np.random.uniform(0,0.9), 3)
36    R3.betaz = round(1 - R3.alphaz, 3)
37    print("the new z-positions for R3 are: ", R3.alphaz, R3.betaz)
38
39 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
40 R1.gamma = 1 - R1.delta
41 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
42 R2.gamma = 1 - R2.delta
43 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
44 R3.gamma = 1 - R3.delta
45 print(R1.delta, R2.delta, R3.delta)

```

0.81 0.83 0.89

In [46]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [47]:

```
1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time5_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time5_ = mixed_time5_.overlay(audio3)      # further combine , superi
285
286 mixed_time5_.export("notes_/mixed_time5.mp3", format='mp3') # export mixed audi
287 play(mixed_time5)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

```

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]:

```

1 if (R1.delta > R2.delta) and (R1.delta > R3.delta):
2     print("glu glu")
3     if (R1.alphax < 0.3): # I have to customize state vectors according to prec
4         print("bri")
5         qc3.x(q[0])           # just using the NOT gate as a test
6     if (R1.alphax == 0.5): # I have to customize state vectors according to pre
7         qc3.h(q[0])
8     if (R1.alphax >= 0.3) and (R2.alphax < 0.5):
9         print('jungle!')
10        qc3.ry(1.9106332, q[0])
11    if (R1.alphax >= 0.6) and (R2.alphax < 0.7):
12        print('ocean!')
13        qc3.ry(1.2309594, q[0])
14    if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
15        print("cra cra")
16        qc3.x(q[1])
17    if (R1.alphay == 0.5): # I have to customize state vectors according to pre
18        qc3.h(q[1])
19    if (R1.alphay >= 0.3) and (R2.alphay < 0.5):
20        print('jungle!')
21        qc3.ry(1.9106332, q[1])
22    if (R1.alphay >= 0.6) and (R2.alphay < 0.7):
23        print('ocean!')
24        qc3.ry(1.2309594, q[1])
25    if (R1.alphaz <= 0.2): # else: the qubit sticks with the default value '0'
26        qc3.x(q[2])
27        print("augh")
28    if (R1.alphaz == 0.5): # I have to customize state vectors according to pre
29        print("ouch")
30        qc3.h(q[2])
31    if (R1.alphaz >= 0.3) and (R2.alphaz < 0.5):
32        print('jungle!')
33        qc3.ry(1.9106332, q[2])
34    if (R1.alphaz >= 0.6) and (R2.alphaz < 0.7):
35        print('ocean!')
36        qc3.ry(1.2309594, q[2])
37    if (R1.alphaz >= 0.2 and R1.alphax < 0.3):
38        print("wolf")
39        qc3.x(q[2])
40    #if (R1.alphaz >= 0.7):
41    #    print("wolf2") # leave the state as 0, default value
42    if (R1.delta == 0.5):
43        qc3.h(q[3])
44    if (R1.delta == 0.6):
45        qc3.h(q[3])
46    if (R1.delta >= 0.7):
47        qc3.x(q[3])
48    if (R1.gamma >= 0.3) and (R2.gamma < 0.5):
49        print('jungle!')
50        qc3.ry(1.9106332, q[3])
51    if (R1.gamma >= 0.6) and (R2.gamma < 0.7):
52        print('ocean!')
53        qc3.ry(1.2309594, q[3])
54 # February 13: IT WAS SUPPOSED TO BE (R2.delta > R1.delta) and (R2.delta > R3.d
55 # and R2 rather than R1; correct also in the other file!!! (for x-y)
56 #elif (R1.delta > R2.delta) and (R1.delta > R3.delta): # February 13: THAT MUST
57 elif (R2.delta > R1.delta) and (R2.delta > R3.delta): # February 13: THAT MUST
58     print('dog')
59     if (R2.alphax < 0.3): # I have to customize state vectors according to prec

```

```

60      qc3.x(q[0])          # just using the NOT gate as a test
61  if (R2.alphax == 0.5): # I have to customize state vectors according to pre
62    qc3.h(q[0])
63  if (R2.alphax >= 0.3) and (R1.alphax < 0.5):
64    print('jungle!')
65    qc3.ry(1.9106332, q[0])
66  if (R2.alphax >= 0.6) and (R1.alphax < 0.7):
67    print('ocean!')
68    qc3.ry(1.2309594, q[0])
69  if (R2.alphay <= 0.2): # else: the qubit sticks with the default value '0'
70    qc3.x(q[1])
71  if (R2.alphay == 0.5): # I have to customize state vectors according to pre
72    qc3.h(q[1])
73  if (R2.alphay >= 0.3) and (R1.alphay < 0.5):
74    print('jungle!')
75    qc3.ry(1.9106332, q[1])
76  if (R2.alphay >= 0.6) and (R1.alphay < 0.7):
77    print('ocean!')
78    qc3.ry(1.2309594, q[1])
79  if (R2.alphaz <= 0.2): # else: the qubit sticks with the default value '0'
80    qc3.x(q[2])
81  if (R2.alphaz == 0.5): # I have to customize state vectors according to pre
82    qc3.h(q[2])
83  if (R2.alphaz >= 0.3) and (R1.alphaz < 0.5):
84    print('jungle!')
85    qc3.ry(1.9106332, q[2])
86  if (R2.alphaz >= 0.6) and (R1.alphaz < 0.7):
87    print('ocean!')
88    qc3.ry(1.2309594, q[2])
89  if (R1.alphaz >= 0.2 and R1.alphax < 0.3):
90    print("wolf")
91    qc3.x(q[2])
92  if (R2.delta == 0.5):
93    qc3.h(q[3])
94  if (R2.delta == 0.6):
95    qc3.h(q[3])
96  if (R2.delta >= 0.7):
97    qc3.x(q[3])
98  if (R2.gamma >= 0.3) and (R1.gamma < 0.5):
99    print('jungle!')
100   qc3.ry(1.9106332, q[3])
101  if (R2.gamma >= 0.6) and (R1.gamma < 0.7):
102    print('ocean!')
103    qc3.ry(1.2309594, q[3])
104 else: # February 13: REVISE ALL THIS SECTION!!!!
105  print('cat') # I made some tests to check the IF conditions
106  if (R3.alphax < 0.3):
107    qc3.x(q[0])
108  if (R3.alphax == 0.5):
109    qc3.h(q[0])
110  if (R3.alphax >= 0.3) and (R3.alphax < 0.5):
111    print('jungle!')
112    qc3.ry(1.9106332, q[0])
113  if (R3.alphax >= 0.6) and (R3.alphax < 0.7):
114    print('ocean!')
115    qc3.ry(1.2309594, q[0])
116  if (R3.alphay < 0.3):
117    qc3.x(q[1])
118  if (R3.alphay == 0.5):
119    qc3.h(q[1])
120  if (R3.alphay >= 0.3) and (R3.alphay < 0.5):

```

```
121     print('jungle!')
122     qc3.ry(1.9106332, q[1])
123 if (R3.alphay >= 0.6) and (R3.alphay < 0.7):
124     print('ocean!')
125     qc3.ry(1.2309594, q[1])
126 if (R3.alphaz < 0.3):
127     qc3.x(q[2])
128 if (R3.alphaz == 0.5):
129     qc3.h(q[2])
130 if (R3.alphaz >= 0.3) and (R3.alphaz < 0.5):
131     print('jungle!')
132     qc3.ry(1.9106332, q[2])
133 if (R3.alphaz >= 0.6) and (R3.alphaz < 0.7):
134     print('ocean!')
135     qc3.ry(1.2309594, q[2])
136 if (R3.delta == 0.5):
137     qc3.h(q[3])
138 if (R3.delta == 0.6):
139     qc3.h(q[3])
140 if (R3.delta >= 0.7):
141     qc3.x(q[3])
142 if (R3.gamma >= 0.3) and (R3.gamma < 0.5):
143     print('jungle!')
144     qc3.ry(1.9106332, q[3])
145 if (R3.gamma >= 0.6) and (R3.gamma < 0.7):
146     print('ocean!')
147     qc3.ry(1.2309594, q[3])
```

cat

In [50]:

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

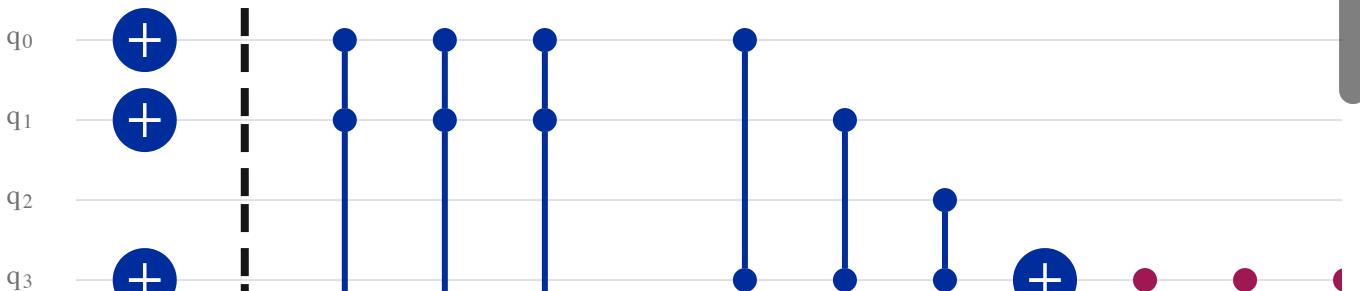
Numeration of qubits within IF instructions is slightly different than the initial one. In fact, some distinction across qubits was needed to clearly build the whole circuit later on. Thus, I decided to keep them apart.

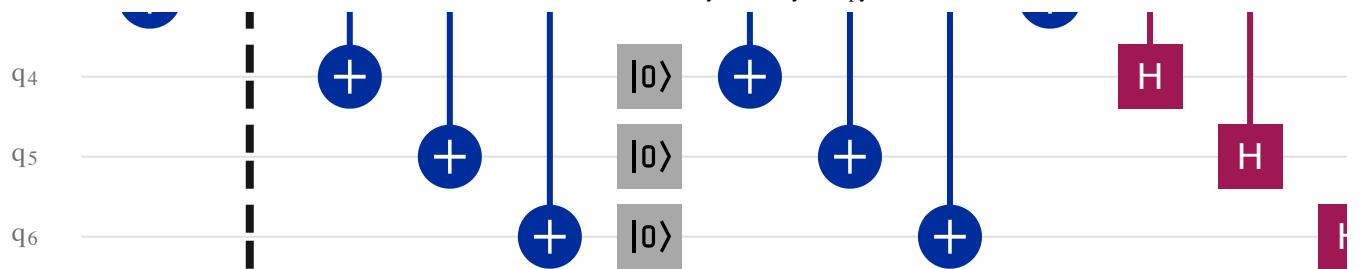
In [51]:

```

1 # this is the core code, and it is unchanged across time
2
3 qc3.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6])
4 qc3.ccx(q[0], q[1], q[4])
5 qc3.ccx(q[0], q[1], q[5])
6 qc3.ccx(q[0], q[1], q[6])
7 qc3.reset(q[4])
8 qc3.reset(q[5])
9 qc3.reset(q[6])
10 qc3.ccx(q[0], q[3], q[4])
11 qc3.ccx(q[1], q[3], q[5])
12 qc3.ccx(q[2], q[3], q[6])
13 qc3.x(q[3])
14 qc3.ch(q[3], q[4])
15 qc3.ch(q[3], q[5])
16 qc3.ch(q[3], q[6])
17 qc3.x(q[3])
18 qc3.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6])
19 qc3.measure(q[3], m3[0])
20 qc3.measure(q[4], m4[0])
21 qc3.measure(q[5], m5[0])
22 qc3.measure(q[6], m6[0])
23
24 # visualization of the circuit
25
26 draw_circuit(qc3)
27
28 # definition of quantum simulator
29
30 simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_simulator
31 qc3 = transpile(qc3, simulator)
32 cc = collections.Counter()
33
34 # Run and get counts
35 result = simulator.run(qc3, shots=1024).result()
36 counts = result.get_counts(qc3)
37 counts2 = counts.most_frequent() # does not work if multiple states have the same count
38 # decide something if multiple states have the same count --> e.g., choose the first one
39 counts3 = cc.most_common(2)
40 print(counts)
41 print(counts2)
42 print(counts3)
43 result = simulator.run(qc3, shots=10, memory=True).result()
44 memory = result.get_memory(qc3)
45 print(memory)
46 plot_histogram(counts, title='outcomes')
47 # TAKE the TWO more present outcomes

```





In [45]:

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

In [52]:

```
1 print(counts2) # order: R3, R2, R1. Add some uncertainty?
2 # export as an array
3 str = counts2
4 arr1 = str.split(' ') # to split the string and avoid empty spaces as array elem
5 print(arr1)
6 weight1 = 1024 # AT HAND ONLY FOR NOW
7
8 arr2 = ['0','1','1', '1'] # 111 # 011
9 print(arr2)
10 weight2 = 1024
11 # BY HAND ONLY FOR NOW
12
13
14 # an attempt, not so good, to automatize this passage:
15
16 print(memory)
17
18 data = Counter(memory)
19 data.most_common() # Returns all unique items and their counts
20 data.most_common(3)
21
22 print(data.most_common())
23 print(data.most_common(1))
24 arrx1 = data.most_common(2)[0]
25 print(arrx1)
26 arrx2 = data.most_common(2)[1]
27 print(arrx2)
28
29
```

```
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]:

```
1 # array 1
2 arr1
```

Out[53]:

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

In [54]:

```
1 # array 2
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]:

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

Position for R_1 :

In [56]:

```

(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
    se9
    change of January 28: I'm substituting [0] with [1] and vice versa, because the output
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
13     print("bla"))
14     R1.alphax = 0.3
15     R1.betax = 0.7
16     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 50 and (
17         print("gulp"))
18         R1.alphax = 0.7
19         R1.betax = 0.3
20         elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and (weight1 == weight2 or np.absol
21             print("stra-gulp"))
22             R1.alphax = 0.5 # change temporarily made on January 24: random generator rath
23             R1.betax = 0.5 # same as above
24             elif (arr1[2] == arr2[0]) and (arr1[2] == '1') and ((weight2 - weight1) > 800): #
25                 print("thunderstorm!")
26                 R1.alphax = 0
27                 R1.betax = 1
28                 elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 200 and
29                     print("avalanche!"))
30                     R1.alphax = 0.1
31                     R1.betax = 0.9
32                     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 200 and
33                         print("earthquake!"))
34                         R1.alphax = 0.9
35                         R1.betax = 0.1
36                         elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 200 and
37                             print("avalanche bis!"))
38                             R1.alphax = 0.1 # the same also in this case
39                             R1.betax = 0.9 # the same also in this case
40     same = outcome 0 # January 23
41     elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight1 - weight2) > 50): # n
42         print("bla 2")
43         R1.alphax = 0.7 # the opposite??
44         R1.betax = 0.3 # the opposite??
45         elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight2 - weight1) > 50): # n
46             print("gulp 2")
47             R1.alphax = 0.3 # the opposite??
48             R1.betax = 0.7 # the opposite??
49             elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and (weight1 == weight2 or np.absol
50                 print("stra-gulp 2"))
51                 R1.alphax = 0.5 # change temporarily made on January 24: random generator rath
52                 R1.betax = 0.5 # change temporarily made on January 24: random generator rathe
53     # different outcomes
54     elif arr1[2] != arr2[2]: # January 23
55         print("blue")
56         if (arr1[2] != arr2[2]) and (weight1 == weight2 or np.absolute(weight1 - weigh
57             print("google 1"))
58             R1.alphax = 0.5 # change temporarily made on January 24: random generator
59             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 # my 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

```

```
121 print("bla")
122 R1.alphay = 0.3
123 R1.betay = 0.7
124 if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 50 and (
125     print("gulp"))
126     R1.alphay = 0.7
127     R1.betay = 0.3
128 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or np.absol
129     print("stra-gulp"))
130     R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
131     R1.betay = 0.5 # same as above
132 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 800): #
133     print("thunderstorm!")
134     R1.alphay = 0
135     R1.betay = 1 # note of February 15: attention there: [1] rather than [0]
136 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 200 and
137     print("avalanche!"))
138     R1.alphay = 0.1
139     R1.betay = 0.9
140 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 200 and
141     print("earthquake!"))
142     R1.alphay = 0.9
143     R1.betay = 0.1
144 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 200 and
145     print("avalanche bis!"))
146     R1.alphay = 0.1 # the same also in this case
147     R1.betay = 0.9 # the same also in this case
148 same = outcome 0 # January 23
149 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 50): # n
150     print("bla 2")
151     R1.alphay = 0.7 # the opposite??
152     R1.betay = 0.3 # the opposite??
153 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 50): # n
154     print("gulp 2")
155     R1.alphay = 0.3 # the opposite??
156     R1.betay = 0.7 # the opposite??
157 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or np.absol
158     print("stra-gulp 2"))
159     R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
160     R1.betay = 0.5 # change temporarily made on January 24: random generator rath
161 different outcomes
162 elif arr1[1] != arr2[1]: # January 23
163     print("blue")
164     if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1 - weigh
165         print("google 1"))
166         R1.alphay = 0.5 # change temporarily made on January 24: random generator
167         R1.betay = 0.5 # change temporarily made on January 24: random generator r
168     if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.absolute(
169         # include the case of a very small difference!
170         print("uffdah"))
171         R1.alphay = 0.5
172         R1.betay = 0.5
173     if (arr1[1] == '1' and arr2[1] == '0'):
174         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
175             print("abc")
176             R1.alphay = 0.3
177             R1.betay = 0.7
178         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
179             print("bca")
180             R1.alphay = 0.7
181             R1.betay = 0.3
```

```

182     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
183         print("news")
184         R1.alphay = 0.2 #
185         R1.betay = 0.8 #
186     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
187         print("idea")
188         R1.alphay = 0.8 #
189         R1.betay = 0.2 #
190     if (arr1[1] == '0') and (arr2[1] == '1'):
191         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
192             print("bac")
193             R1.alphay = 0.7
194             R1.betay = 0.3
195         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
196             print("cba")
197             R1.alphay = 0.3
198             R1.betay = 0.7
199         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
200             print("brain")
201             R1.alphay = 0.7 # 0.9
202             R1.betay = 0.3 # 0.1
203         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
204             print("hand")
205             R1.alphay = 0.3 # 0.1
206             R1.betay = 0.7 # 0.9
207         if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no abs # c
208             print("brain2")
209             R1.alphay = 0.9 # 0.9
210             R1.betay = 0.1 # 0.1
211         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no abs # c
212             print("hand2")
213             R1.alphay = 0.1 # 0.1
214             R1.betay = 0.9 # 0.9
215
216
217 part
218
219 # change of February 15
220
221 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50 and (we
222     print("bla"))
223     R1.alphay = 0.3
224     R1.betay = 0.7
225 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 50 and (
226     print("gulp"))
227     R1.alphay = 0.7
228     R1.betay = 0.3
229 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or np.absol
230     print("stra-gulp"))
231     R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
232     R1.betay = 0.5 # same as above
233 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 800): #
234     print("thunderstorm!")
235     R1.alphay = 0
236     R1.betay = 1 # note of February 15: attention there: [1] rather than [0]
237 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 200 and
238     print("avalanche!"))
239     R1.alphay = 0.1
240     R1.betay = 0.9
241 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 200 and
242     print("earthquake!"))

```

```

243 R1.alphay = 0.9
244 R1.betay = 0.1
245 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 200 and
246 print("avalanche bis!")
247 R1.alphay = 0.1 # the same also in this case
248 R1.betay = 0.9 # the same also in this case
249 # same = outcome 0 # January 23
250 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 50): # n
251 print("bla 2")
252 R1.alphay = 0.7 # the opposite??
253 R1.betay = 0.3 # the opposite??
254 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 50): # n
255 print("gulp 2")
256 R1.alphay = 0.3 # the opposite??
257 R1.betay = 0.7 # the opposite??
258 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.absol
259 print("stra-gulp 2")
260 R1.alphay = 0.5 # change temporarily made on January 24: random generator rath
261 R1.betay = 0.5 # change temporarily made on January 24: random generator rath
262 # different outcomes
263 elif arr1[0] != arr2[0]: # January 23
264 print("blue")
265 if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.absolute(weight1 - weigh
266 print("google 1")
267 R1.alphay = 0.5 # change temporarily made on January 24: random generator
268 R1.betay = 0.5 # change temporarily made on January 24: random generator r
269 if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.absolute(
270 # include the case of a very small difference!
271 print("uffdah")
272 R1.alphay = 0.5
273 R1.betay = 0.5
274 if (arr1[0] == '1' and arr2[0] == '0'):
275     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
276         print("abc")
277         R1.alphay = 0.3
278         R1.betay = 0.7
279     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
280         print("bca")
281         R1.alphay = 0.7
282         R1.betay = 0.3
283     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
284         print("news")
285         R1.alphay = 0.2 #
286         R1.betay = 0.8 #
287     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
288         print("idea")
289         R1.alphay = 0.8 #
290         R1.betay = 0.2 #
291 if (arr1[0] == '0') and (arr2[0] == '1'):
292     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
293         print("bac")
294         R1.alphay = 0.7
295         R1.betay = 0.3
296     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
297         print("cba")
298         R1.alphay = 0.3
299         R1.betay = 0.7
300     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no abs # c
301         print("brain")
302         R1.alphay = 0.7 # 0.9
303         R1.betay = 0.3 # 0.1

```

```
304     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no abs # c
305         print("hand")
306         R1.alphay = 0.3 # 0.1
307         R1.betay = 0.7 # 0.9
308     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no abs #
309         print("brain2")
310         R1.alphay = 0.9 # 0.9
311         R1.betay = 0.1 # 0.1
312     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no abs #
313         print("hand2")
314         R1.alphay = 0.1 # 0.1
315         R1.betay = 0.9 # 0.9
```

stra-gulp
stra-gulp
stra-gulp 2

Position for R_2

In [57]:

```

1 if (R2.delta > R1.delta) and (R2.delta > R3.delta):
2     # if R2 didn't enter the gate, keep its position
3     R2.alphax = R2.alphax
4     R2.betax = R2.betax
5     R2.alphay = R2.alphay
6     R2.betay = R2.betay
7     R2.alphaz = R2.alphaz
8     R2.betaz = R2.betaz
9 else:
10    # change of January 28: I'm substituting [0] with [1] and vice versa, because t
11    # February 15: in the case with xyz, we have [2] first
12    if (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 50
13        print("bla")
14        R2.alphax = 0.3
15        R2.betax = 0.7
16    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 5
17        print("gulp")
18        R2.alphax = 0.7
19        R2.betax = 0.3
20    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and (weight1 == weight2 or n
21        print("stra-gulp")
22        R2.alphax = 0.5 # change temporarily made on January 24: random generat
23        R2.betax = 0.5 # same as above
24    elif (arr1[2] == arr2[0]) and (arr1[2] == '1') and ((weight2 - weight1) > 8
25        print("thunderstorm!")
26        R2.alphax = 0
27        R2.betax = 1
28    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
29        print("avalanche!")
30        R2.alphax = 0.1
31        R2.betax = 0.9
32    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
33        print("earthquake!")
34        R2.alphax = 0.9
35        R2.betax = 0.1
36    elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 2
37        print("avalanche bis!")
38        R2.alphax = 0.1 # the same also in this case
39        R2.betax = 0.9 # the same also in this case
40    # same = outcome 0 # January 23
41    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight1 - weight2) > 5
42        print("bla 2")
43        R2.alphax = 0.7 # the opposite??
44        R2.betax = 0.3 # the opposite??
45    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight2 - weight1) > 5
46        print("gulp 2")
47        R2.alphax = 0.3 # the opposite??
48        R2.betax = 0.7 # the opposite??
49    elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and (weight1 == weight2 or n
50        print("stra-gulp 2")
51        R2.alphax = 0.5 # change temporarily made on January 24: random generat
52        R2.betax = 0.5 # change temporarily made on January 24: random generato
53    # different outcomes
54    elif arr1[2] != arr2[2]: # January 23
55        print("blue")
56        if (arr1[2] != arr2[2]) and (weight1 == weight2 or np.absolute(weight1
57            print("google 1")
58            R2.alphax = 0.5 # change temporarily made on January 24: random gen
59            R2.betax = 0.5 # change temporarily made on January 24: random gene

```

```
60     if (arr1[2] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.abs(arr1[2] - arr2[1]) <= 50):
61         # include the case of a very small difference!
62         print("uffdah")
63         R2.alphax = 0.5
64         R2.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             R2.alphax = 0.3
69             R2.betax = 0.7
70         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
71             print("bca")
72             R2.alphax = 0.7
73             R2.betax = 0.3
74         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
75             print("news")
76             R2.alphax = 0.2 #
77             R2.betax = 0.8 #
78         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
79             print("idea")
80             R2.alphax = 0.8 #
81             R2.betax = 0.2 #
82         if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
83             print("news")
84             R2.alphax = 0.1 #
85             R2.betax = 0.9 #
86         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
87             print("idea")
88             R2.alphax = 0.9 #
89             R2.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             R2.alphax = 0.7
94             R2.betax = 0.3
95         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
96             print("cba")
97             R2.alphax = 0.3
98             R2.betax = 0.7
99         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
100            print("brain")
101            R2.alphax = 0.7 # 0.9
102            R2.betax = 0.3 # 0.1
103         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
104            print("hand")
105            R2.alphax = 0.3 # 0.1
106            R2.betax = 0.7 # 0.9
107         if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
108            print("brain2")
109            R2.alphax = 0.9 # 0.9
110            R2.betax = 0.1 # 0.1
111         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
112            print("hand2")
113            R2.alphax = 0.1 # 0.1
114            R2.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)
```

```

121     print("bla")
122     R2.alphay = 0.3
123     R2.betay = 0.7
124 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
125     print("gulp")
126     R2.alphay = 0.7
127     R2.betay = 0.3
128 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
129     print("stra-gulp")
130     R2.alphay = 0.5 # change temporarily made on January 24: random generat
131     R2.betay = 0.5 # same as above
132 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
133     print("thunderstorm!")
134     R2.alphay = 0
135     R2.betay = 1 # note of February 15: attention there: [1] rather than [0]
136 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
137     print("avalanche!")
138     R2.alphay = 0.1
139     R2.betay = 0.9
140 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
141     print("earthquake!")
142     R2.alphay = 0.9
143     R2.betay = 0.1
144 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
145     print("avalanche bis!")
146     R2.alphay = 0.1 # the same also in this case
147     R2.betay = 0.9 # the same also in this case
148 # same = outcome 0 # January 23
149 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
150     print("bla 2")
151     R2.alphay = 0.7 # the opposite??
152     R2.betay = 0.3 # the opposite??
153 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
154     print("gulp 2")
155     R2.alphay = 0.3 # the opposite??
156     R2.betay = 0.7 # the opposite??
157 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
158     print("stra-gulp 2")
159     R2.alphay = 0.5 # change temporarily made on January 24: random generat
160     R2.betay = 0.5 # change temporarily made on January 24: random generato
161 # different outcomes
162 elif arr1[1] != arr2[1]: # January 23
163     print("blue")
164     if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
165         print("google 1")
166         R2.alphay = 0.5 # change temporarily made on January 24: random gen
167         R2.betay = 0.5 # change temporarily made on January 24: random gene
168     if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
169         # include the case of a very small difference!
170         print("uffdah")
171         R2.alphay = 0.5
172         R2.betay = 0.5
173     if (arr1[1] == '1' and arr2[1] == '0'):
174         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
175             print("abc")
176             R2.alphay = 0.3
177             R2.betay = 0.7
178         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
179             print("bca")
180             R2.alphay = 0.7
181             R2.betay = 0.3

```

```

182     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
183         print("news")
184         R2.alphay = 0.2 #
185         R2.betay = 0.8 #
186     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
187         print("idea")
188         R2.alphay = 0.8 #
189         R2.betay = 0.2 #
190     if (arr1[1] == '0') and (arr2[1] == '1'):
191         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
192             print("bac")
193             R2.alphay = 0.7
194             R2.betay = 0.3
195         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
196             print("cba")
197             R2.alphay = 0.3
198             R2.betay = 0.7
199         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
200             print("brain")
201             R2.alphay = 0.7 # 0.9
202             R2.betay = 0.3 # 0.1
203         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
204             print("hand")
205             R2.alphay = 0.3 # 0.1
206             R2.betay = 0.7 # 0.9
207         if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
208             print("brain2")
209             R2.alphay = 0.9 # 0.9
210             R2.betay = 0.1 # 0.1
211         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
212             print("hand2")
213             R2.alphay = 0.1 # 0.1
214             R2.betay = 0.9 # 0.9
215
216
217 # z part
218
219 # change of February 15
220
221 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
222     print("bla")
223     R2.alphay = 0.3
224     R2.betay = 0.7
225 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
226     print("gulp")
227     R2.alphay = 0.7
228     R2.betay = 0.3
229 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
230     print("stra-gulp")
231     R2.alphay = 0.5 # change temporarily made on January 24: random generat
232     R2.betay = 0.5 # same as above
233 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
234     print("thunderstorm!")
235     R2.alphay = 0
236     R2.betay = 1 # note of February 15: attention there: [1] rather than [0]
237 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
238     print("avalanche!")
239     R2.alphay = 0.1
240     R2.betay = 0.9
241 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
242     print("earthquake!")

```

```

243     R2.alphay = 0.9
244     R2.betay = 0.1
245 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
246     print("avalanche bis!")
247     R2.alphay = 0.1 # the same also in this case
248     R2.betay = 0.9 # the same also in this case
249 # same = outcome 0 # January 23
250 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
251     print("bla 2")
252     R2.alphay = 0.7 # the opposite??
253     R2.betay = 0.3 # the opposite??
254 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
255     print("gulp 2")
256     R2.alphay = 0.3 # the opposite??
257     R2.betay = 0.7 # the opposite??
258 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.abs(
259     print("stra-gulp 2")
260     R2.alphay = 0.5 # change temporarily made on January 24: random generat
261     R2.betay = 0.5 # change temporarily made on January 24: random generato
262 # different outcomes
263 elif arr1[0] != arr2[0]: # January 23
264     print("blue")
265     if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.abs(weight1 - weight2) <= 1):
266         print("google 1")
267         R2.alphay = 0.5 # change temporarily made on January 24: random gen
268         R2.betay = 0.5 # change temporarily made on January 24: random gene
269     if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.abs(
270         # include the case of a very small difference!
271         print("uffdah")
272         R2.alphay = 0.5
273         R2.betay = 0.5
274     if (arr1[0] == '1' and arr2[0] == '0'):
275         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
276             print("abc")
277             R2.alphay = 0.3
278             R2.betay = 0.7
279         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
280             print("bca")
281             R2.alphay = 0.7
282             R2.betay = 0.3
283         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
284             print("news")
285             R2.alphay = 0.2 #
286             R2.betay = 0.8 #
287         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
288             print("idea")
289             R2.alphay = 0.8 #
290             R2.betay = 0.2 #
291     if (arr1[0] == '0') and (arr2[0] == '1'):
292         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
293             print("bac")
294             R2.alphay = 0.7
295             R2.betay = 0.3
296         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
297             print("cba")
298             R2.alphay = 0.3
299             R2.betay = 0.7
300         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
301             print("brain")
302             R2.alphay = 0.7 # 0.9
303             R2.betay = 0.3 # 0.1

```

```
304     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
305         print("hand")
306         R2.alphay = 0.3 # 0.1
307         R2.betay = 0.7 # 0.9
308     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
309         print("brain2")
310         R2.alphay = 0.9 # 0.9
311         R2.betay = 0.1 # 0.1
312     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
313         print("hand2")
314         R2.alphay = 0.1 # 0.1
315         R2.betay = 0.9 # 0.9
```

stra-gulp
stra-gulp
stra-gulp 2

Position for R_3

In [60]:

```

1 if (R3.delta > R1.delta) and (R3.delta > R2.delta):
2     # if R3 didn't enter the gate, keep its position
3     print("achtung!")
4     R3.alphax = R3.alphax
5     R3.betax = R3.betax
6     R3.alphay = R3.alphay
7     R3.betay = R3.betay
8     R3.alphaz = R3.alphaz
9     R3.betaz = R3.betaz
10 else:
11     # change of January 28: I'm substituting [0] with [1] and vice versa, because t
12     # February 15: in the case with xyz, we have [2] first
13     if (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 50
14         print("bla")
15         R3.alphax = 0.3
16         R3.betax = 0.7
17     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 5
18         print("gulp")
19         R3.alphax = 0.7
20         R3.betax = 0.3
21     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and (weight1 == weight2 or n
22         print("stra-gulp")
23         R3.alphax = 0.5 # change temporarily made on January 24: random generat
24         R3.betax = 0.5 # same as above
25     elif (arr1[2] == arr2[0]) and (arr1[2] == '1') and ((weight2 - weight1) > 8
26         print("thunderstorm!")
27         R3.alphax = 0
28         R3.betax = 1
29     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
30         print("avalanche!")
31         R3.alphax = 0.1
32         R3.betax = 0.9
33     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight2 - weight1) > 2
34         print("earthquake!")
35         R3.alphax = 0.9
36         R3.betax = 0.1
37     elif (arr1[2] == arr2[2]) and (arr1[2] == '1') and ((weight1 - weight2) > 2
38         print("avalanche bis!")
39         R3.alphax = 0.1 # the same also in this case
40         R3.betax = 0.9 # the same also in this case
41     # same = outcome 0 # January 23
42     elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight1 - weight2) > 5
43         print("bla 2")
44         R3.alphax = 0.7 # the opposite??
45         R3.betax = 0.3 # the opposite??
46     elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and ((weight2 - weight1) > 5
47         print("gulp 2")
48         R3.alphax = 0.3 # the opposite??
49         R3.betax = 0.7 # the opposite??
50     elif (arr1[2] == arr2[2]) and (arr1[2] == '0') and (weight1 == weight2 or n
51         print("stra-gulp 2")
52         R3.alphax = 0.5 # change temporarily made on January 24: random generat
53         R3.betax = 0.5 # change temporarily made on January 24: random generato
54     # different outcomes
55     elif arr1[2] != arr2[2]: # January 23
56         print("blue")
57         if (arr1[2] != arr2[2]) and (weight1 == weight2 or np.absolute(weight1
58             print("google 1")
59             R3.alphax = 0.5 # change temporarily made on January 24: random gen

```

```
60      R3.betax = 0.5 # change temporarily made on January 24: random gene
61  if (arr1[2] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.abs(
62      # include the case of a very small difference!
63      print("uffdah"))
64      R3.alphax = 0.5
65      R3.betax = 0.5
66  if (arr1[2] == '1' and arr2[1] == '0'):
67      if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
68          print("abc")
69          R3.alphax = 0.3
70          R3.betax = 0.7
71  if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
72      print("bca")
73      R3.alphax = 0.7
74      R3.betax = 0.3
75  if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
76      print("news")
77      R3.alphax = 0.2 #
78      R3.betax = 0.8 #
79  if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
80      print("idea")
81      R3.alphax = 0.8 #
82      R3.betax = 0.2 #
83  if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
84      print("news")
85      R3.alphax = 0.1 #
86      R3.betax = 0.9 #
87  if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
88      print("idea")
89      R3.alphax = 0.9 #
90      R3.betax = 0.1 #
91  if (arr1[2] == '0' and arr2[2] == '1'):
92      if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
93          print("bac")
94          R3.alphax = 0.7
95          R3.betax = 0.3
96  if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
97      print("cba")
98      R3.alphax = 0.3
99      R3.betax = 0.7
100 if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
101     print("brain")
102     R3.alphax = 0.7 # 0.9
103     R3.betax = 0.3 # 0.1
104 if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
105     print("hand")
106     R3.alphax = 0.3 # 0.1
107     R3.betax = 0.7 # 0.9
108 if ((weight1 - weight2) > 200 and (weight2 - weight1) >= 800): # no
109     print("brain2")
110     R3.alphax = 0.9 # 0.9
111     R3.betax = 0.1 # 0.1
112 if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
113     print("hand2")
114     R3.alphax = 0.1 # 0.1
115     R3.betax = 0.9 # 0.9
116 # y part
117
118 # change of January 26
119
120
```

```

121     if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
122         print("bla")
123         R3.alphay = 0.3
124         R3.betay = 0.7
125     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
126         print("gulp")
127         R3.alphay = 0.7
128         R3.betay = 0.3
129     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
130         print("stra-gulp")
131         R3.alphay = 0.5 # change temporarily made on January 24: random generat
132         R3.betay = 0.5 # same as above
133     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
134         print("thunderstorm!")
135         R3.alphay = 0
136         R3.betay = 1 # note of February 15: attention there: [1] rather than [0
137     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
138         print("avalanche!")
139         R3.alphay = 0.1
140         R3.betay = 0.9
141     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
142         print("earthquake!")
143         R3.alphay = 0.9
144         R3.betay = 0.1
145     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
146         print("avalanche bis!")
147         R3.alphay = 0.1 # the same also in this case
148         R3.betay = 0.9 # the same also in this case
149 # same = outcome 0 # January 23
150     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
151         print("bla 2")
152         R3.alphay = 0.7 # the opposite??
153         R3.betay = 0.3 # the opposite??
154     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
155         print("gulp 2")
156         R3.alphay = 0.3 # the opposite??
157         R3.betay = 0.7 # the opposite??
158     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
159         print("stra-gulp 2")
160         R3.alphay = 0.5 # change temporarily made on January 24: random generat
161         R3.betay = 0.5 # change temporarily made on January 24: random generato
162 # different outcomes
163     elif arr1[1] != arr2[1]: # January 23
164         print("blue")
165         if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
166             print("google 1")
167             R3.alphay = 0.5 # change temporarily made on January 24: random gen
168             R3.betay = 0.5 # change temporarily made on January 24: random gene
169         if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
170             # include the case of a very small difference!
171             print("uffdah")
172             R3.alphay = 0.5
173             R3.betay = 0.5
174         if (arr1[1] == '1' and arr2[1] == '0'):
175             if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
176                 print("abc")
177                 R3.alphay = 0.3
178                 R3.betay = 0.7
179             if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
180                 print("bca")
181                 R3.alphay = 0.7

```

```

182         R3.betay = 0.3
183     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
184         print("news")
185         R3.alphay = 0.2 #
186         R3.betay = 0.8 #
187     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
188         print("idea")
189         R3.alphay = 0.8 #
190         R3.betay = 0.2 #
191     if (arr1[1] == '0') and (arr2[1] == '1'):
192         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
193             print("bac")
194             R3.alphay = 0.7
195             R3.betay = 0.3
196         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
197             print("cba")
198             R3.alphay = 0.3
199             R3.betay = 0.7
200         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
201             print("brain")
202             R3.alphay = 0.7 # 0.9
203             R3.betay = 0.3 # 0.1
204         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
205             print("hand")
206             R3.alphay = 0.3 # 0.1
207             R3.betay = 0.7 # 0.9
208         if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
209             print("brain2")
210             R3.alphay = 0.9 # 0.9
211             R3.betay = 0.1 # 0.1
212         if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
213             print("hand2")
214             R3.alphay = 0.1 # 0.1
215             R3.betay = 0.9 # 0.9
216
217
218 # z part
219
220 # change of February 15
221
222 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
223     print("bla")
224     R3.alphay = 0.3
225     R3.betay = 0.7
226 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
227     print("gulp")
228     R3.alphay = 0.7
229     R3.betay = 0.3
230 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
231     print("stra-gulp")
232     R3.alphay = 0.5 # change temporarily made on January 24: random generat
233     R3.betay = 0.5 # same as above
234 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
235     print("thunderstorm!")
236     R3.alphay = 0
237     R3.betay = 1 # note of February 15: attention there: [1] rather than [0
238 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
239     print("avalanche!")
240     R3.alphay = 0.1
241     R3.betay = 0.9
242 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2

```

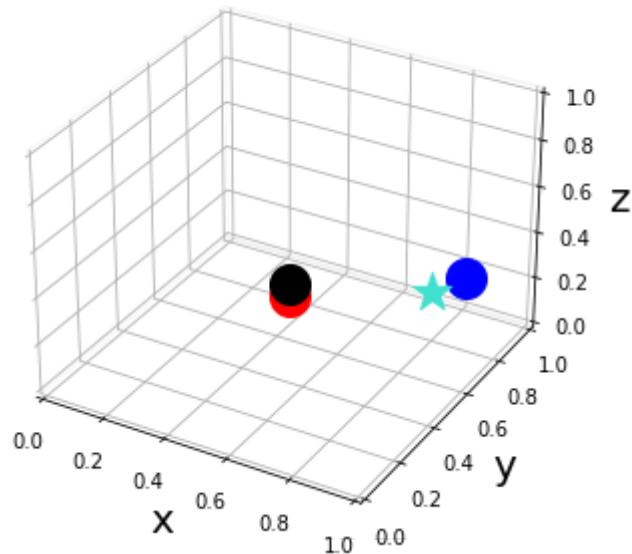
```
243     print("earthquake!")
244     R3.alphay = 0.9
245     R3.betay = 0.1
246     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
247         print("avalanche bis!")
248         R3.alphay = 0.1 # the same also in this case
249         R3.betay = 0.9 # the same also in this case
250     # same = outcome 0 # January 23
251     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
252         print("bla 2")
253         R3.alphay = 0.7 # the opposite??
254         R3.betay = 0.3 # the opposite??
255     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
256         print("gulp 2")
257         R3.alphay = 0.3 # the opposite??
258         R3.betay = 0.7 # the opposite??
259     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
260         print("stra-gulp 2")
261         R3.alphay = 0.5 # change temporarily made on January 24: random generation
262         R3.betay = 0.5 # change temporarily made on January 24: random generation
263     # different outcomes
264     elif arr1[0] != arr2[0]: # January 23
265         print("blue")
266         if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
267             print("google 1")
268             R3.alphay = 0.5 # change temporarily made on January 24: random generation
269             R3.betay = 0.5 # change temporarily made on January 24: random generation
270             if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
271                 # include the case of a very small difference!
272                 print("uffdah")
273                 R3.alphay = 0.5
274                 R3.betay = 0.5
275             if (arr1[0] == '1' and arr2[0] == '0'):
276                 if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
277                     print("abc")
278                     R3.alphay = 0.3
279                     R3.betay = 0.7
280                 if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
281                     print("bca")
282                     R3.alphay = 0.7
283                     R3.betay = 0.3
284                 if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
285                     print("news")
286                     R3.alphay = 0.2 #
287                     R3.betay = 0.8 #
288                 if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
289                     print("idea")
290                     R3.alphay = 0.8 #
291                     R3.betay = 0.2 #
292             if (arr1[0] == '0') and (arr2[0] == '1'):
293                 if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
294                     print("bac")
295                     R3.alphay = 0.7
296                     R3.betay = 0.3
297                 if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
298                     print("cba")
299                     R3.alphay = 0.3
300                     R3.betay = 0.7
301                 if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
302                     print("brain")
303                     R3.alphay = 0.7 # 0.9
```

```
304     R3.betay = 0.3 # 0.1
305     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
306         print("hand")
307         R3.alphay = 0.3 # 0.1
308         R3.betay = 0.7 # 0.9
309     if ((weight1 - weight2) > 200 and (weight1 - weight2) >= 800): # no
310         print("brain2")
311         R3.alphay = 0.9 # 0.9
312         R3.betay = 0.1 # 0.1
313     if ((weight2 - weight1) > 200 and (weight2 - weight1) >= 800): # no
314         print("hand2")
315         R3.alphay = 0.1 # 0.1
316         R3.betay = 0.9 # 0.9
```

achtung!

In [61]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [62]:

```
1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time6_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time6_ = mixed_time6_.overlay(audio3)      # further combine , superi
285
286 mixed_time6.export("notes_/mixed_time6.mp3", format='mp3') # export mixed audi
287 play(mixed_time6)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

tA2
fA2
cF#2

Could not import the PyAudio C module '_portaudio'.

```

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpph_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]:

```
1 # the former ones
2
3 R1.delta, R2.delta, R3.delta
```

Out[63]:

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

In [64]:

```
1 # the new ones
2
3 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
4 print(R1.delta)
5
6 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
7 print(R2.delta)
8
9 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
10 print(R3.delta)
```

```
0.56
0.57
0.89
```

In []:

```
1
```

In [65]:

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

In [66]:

```

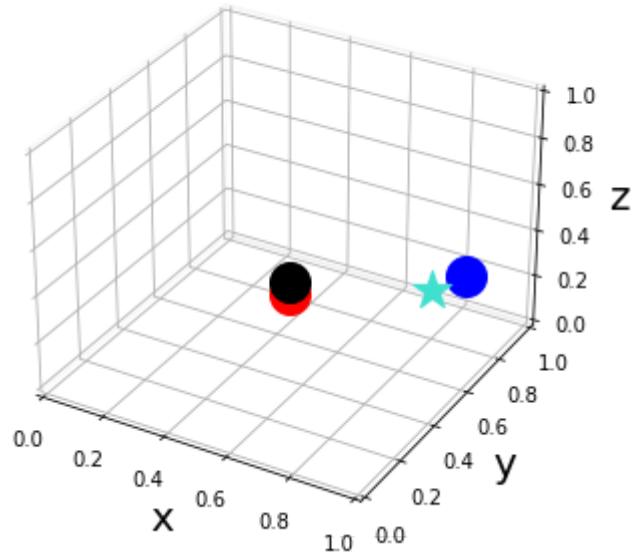
1 # Another round of SOS with a higher threshold. Added on January 28
2
3 # threshold for initial reward
4 # random fluctuations
5
6 if (R1.delta <= 0.6) and (R2.delta <= 0.6) and (R3.delta <= 0.6):
7     print("SOS")
8     # R1
9     R1.alphax = round(np.random.uniform(0,0.9), 3)
10    R1.betax = round(1 - R1.alphax, 3)
11    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
12    R1.alphay = round(np.random.uniform(0,0.9), 3)
13    R1.betay = round(1 - R1.alphay, 3)
14    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
15    R1.alphaz = round(np.random.uniform(0,0.9), 3)
16    R1.betaz = round(1 - R1.alphaz, 3)
17    print("the new z-positions for R1 are: ", R1.alphaz, R1.betaz)
18    # R2
19    R2.alphax = round(np.random.uniform(0,0.9), 3)
20    R2.betax = round(1 - R2.alphax, 3)
21    print("the new x-positions for R2 are: ", R2.alphax, R2.betax)
22    R2.alphay = round(np.random.uniform(0,0.9), 3)
23    R2.betay = round(1 - R2.alphay, 3)
24    print("the new y-positions for R2 are: ", R2.alphay, R2.betay)
25    R2.alphaz = round(np.random.uniform(0,0.9), 3)
26    R2.betaz = round(1 - R2.alphaz, 3)
27    print("the new z-positions for R2 are: ", R2.alphaz, R2.betaz)
28    # R3
29    R3.alphax = round(np.random.uniform(0,0.9), 3)
30    R3.betax = round(1 - R3.alphax, 3)
31    print("the new x-positions for R3 are: ", R3.alphax, R3.betax)
32    R3.alphay = round(np.random.uniform(0,0.9), 3)
33    R3.betay = round(1 - R3.alphay, 3)
34    print("the new y-positions for R3 are: ", R3.alphay, R3.betay)
35    R3.alphaz = round(np.random.uniform(0,0.9), 3)
36    R3.betaz = round(1 - R3.alphaz, 3)
37    print("the new z-positions for R3 are: ", R3.alphaz, R3.betaz)
38
39 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
40 R1.gamma = 1 - R1.delta
41 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
42 R2.gamma = 1 - R2.delta
43 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
44 R3.gamma = 1 - R3.delta
45 print(R1.delta, R2.delta, R3.delta, R1.betaz)

```

0.56 0.56 0.84 0.321

In [67]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [68]:

```
1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time7_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time7_ = mixed_time7_.overlay(audio3)      # further combine , superi
285
286 mixed_time7_.export("notes_/mixed_time7.mp3", format='mp3') # export mixed audi
287 play(mixed_time7)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

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]:

```

1 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
6         R2.alphax = round(1 - R2.betax, 3)
7         R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
8         R2.alphay = round(1 - R2.betay, 3)
9         R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
10        R2.alphaz = round(1 - R2.betaz, 3)
11        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
12        R3.alphax = round(1 - R2.betax, 3)
13        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
14        R3.alphay = round(1 - R2.betay, 3)
15        R3.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
16        R3.alphaz = round(1 - R2.betaz, 3)
17    if (R2.delta > R1.delta and R2.delta > R3.delta):
18        print('quagga')
19        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
20        R1.alphax = round(1 - R1.betax, 3)
21        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
22        R1.alphay = round(1 - R1.betay, 3)
23        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
24        R1.alphaz = round(1 - R1.betaz, 3)
25        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
26        R3.alphax = round(1 - R3.betax, 3)
27        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
28        R3.alphay = round(1 - R3.betay, 3)
29        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
30        R3.alphaz = round(1 - R3.betaz, 3)
31    if (R3.delta > R1.delta and R3.delta > R2.delta):
32        print('quark')
33        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
34        R1.alphax = round(1 - R1.betax, 3)
35        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
36        R1.alphay = round(1 - R1.betay, 3)
37        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
38        R1.alphaz = round(1 - R1.betaz, 3)
39        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
40        R2.alphax = round(1 - R2.betax, 3)
41        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
42        R2.alphay = round(1 - R2.betay, 3)
43        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
44        R2.alphaz = round(1 - R2.betaz, 3)
45
46 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
47 print(R1.delta)
48
49 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
50 print(R2.delta)
51
52 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
53 print(R2.delta)

```

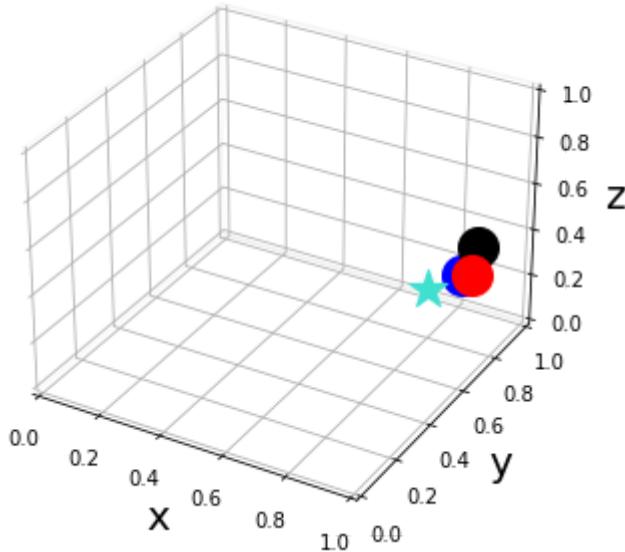
yuk
quark
0.79

```
0.87
0.87
```

In [70]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```
1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```

60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66     if (R1.betax > 0.3 and R1.betax <= 0.5):
67         if (R1.betay < 0.5): # (R1.betay == 1):
68             audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69             print("tD#2")
70         if (R1.betay >= 0.5):
71             audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72             print("tA2")
73     if (R1.betax > 0.5 and R1.betax <= 0.64):
74         if (R1.betay < 0.5):
75             audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76             print("tE2")
77         if (R1.betay >= 0.5):
78             audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79             print("tG#2")
80     if (R1.betax > 0.64 and R1.betax <= 0.84):
81         if (R1.betay < 0.5):
82             audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83             print("tF2")
84         if (R1.betay >= 0.5):
85             audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86             print("tG2")
87     if (R1.betax > 0.84 and R1.betax <= 1):
88         #if (R1.betay == 0.5):
89         audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90         print("tF#2")
91
92
93
94     # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")

```

```
121     print("fD#2")
122 if (R2.betay >= 0.5):
123     audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124     print("fA2")
125 if (R2.betax > 0.5 and R2.betax <= 0.64):
126     if (R2.betay < 0.5):
127         audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128         print("fE2")
129     if (R2.betay >= 0.5):
130         audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131         print("fG#2")
132 if (R2.betax > 0.64 and R2.betax <= 0.84):
133     if (R2.betay < 0.5):
134         audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135         print("fF2")
136     if (R2.betay >= 0.5):
137         audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138         print("fG2")
139 if (R2.betax > 0.84 and R2.betax <= 1):
140     #if (R2.betay == 0.5):
141     audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142     print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148 if (R2.betax > 0 and R2.betax <= 0.17):
149     if (R2.betay < 0.5):
150         audio2 = AudioSegment.from_file("notes_/fB.mp3")
151         print("fB")
152     if (R2.betay >= 0.5):
153         audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154         print("fC#")
155 if (R2.betax > 0.17 and R2.betax <= 0.3):
156     if (R2.betay < 0.5):
157         audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158         print("fA#")
159     if (R2.betay >= 0.5):
160         audio2 = AudioSegment.from_file("notes_/fD.mp3")
161         print("fD")
162 if (R2.betax > 0.3 and R2.betax <= 0.5):
163     if (R2.betay < 0.5): # (R1.betay == 1):
164         audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165         print("fD#")
166     if (R2.betay >= 0.5):
167         audio2 = AudioSegment.from_file("notes_/fA.mp3")
168         print("fA")
169 if (R2.betax > 0.5 and R2.betax <= 0.64):
170     if (R2.betay < 0.5):
171         audio2 = AudioSegment.from_file("notes_/fE.mp3")
172         print("fE")
173     if (R2.betay >= 0.5):
174         audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175         print("fG#")
176 if (R2.betax > 0.64 and R2.betax <= 0.84):
177     if (R2.betay < 0.5):
178         audio2 = AudioSegment.from_file("notes_/fF.mp3")
179         print("fF")
180     if (R2.betay >= 0.5):
181         audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time8_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time8_ = mixed_time8_.overlay(audio3)      # further combine , superi
285
286 mixed_time8.export("notes_/mixed_time8.mp3", format='mp3') # export mixed audi
287 play(mixed_time8)                             # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

```

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

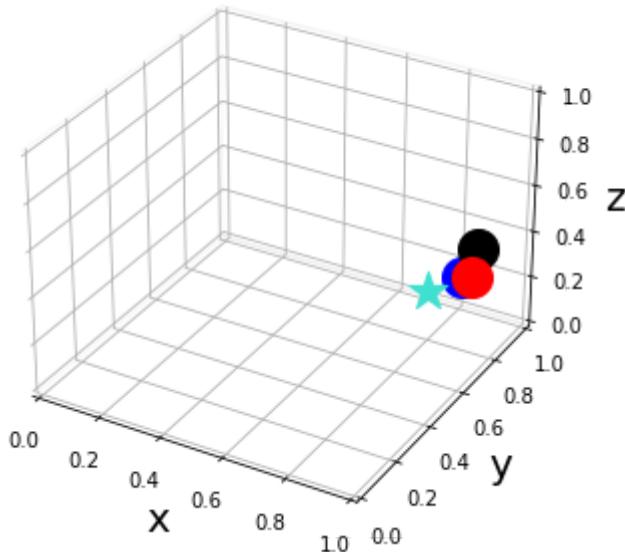
```

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

In [72]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [73]:

```

1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):

```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time9_ = audio1.overlay(audio2)           # combine , superimpose audio fi
284 mixed_time9_ = mixed_time9_.overlay(audio3)      # further combine , superi
285
286 mixed_time9_.export("notes_/mixed_time9.mp3", format='mp3') # export mixed audi
287 play(mixed_time9)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

```

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 γ , δ_i , $i = 1, 2, 3$ according to the target (fixed) positions and the new positions.

New reward amplitude probabilities for R_1 :

In [77]:

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

0.79

New reward amplitude probabilities for R_2 :

In [78]:

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

0.87

New reward amplitude probabilities for R_3 :

In [79]:

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

0.89

In [86]:

```
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 [80]:

```

1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):

```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time10_ = audio1.overlay(audio2)           # combine , superimpose audio f
284 mixed_time10  = mixed_time10_.overlay(audio3)      # further combine , supe
285
286 mixed_time10.export("notes_/mixed_time10.mp3", format='mp3') # export mixed au
287 play(mixed_time10)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

tF#2
fF#2
cF#2

Could not import the PyAudio C module '_portaudio'.

```

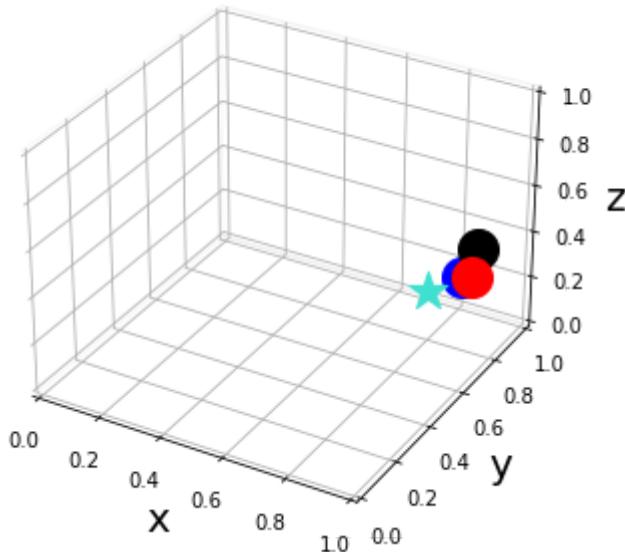
Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmpnitmw_az.
wav':
    Duration: 00:00:07.34, bitrate: 1411 kb/s
    Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 2 channels, s1
6, 1411 kb/s
    7.20 M-A: 0.000 fd= 0 aq= 0KB vq= 0KB sq= 0B f=0/0

```

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

In [81]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [82]:

```
1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):
```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time11_ = audio1.overlay(audio2)           # combine , superimpose audio f
284 mixed_time11_ = mixed_time11_.overlay(audio3)      # further combine , supe
285
286 mixed_time11_.export("notes_/mixed_time11.mp3", format='mp3') # export mixed au
287 play(mixed_time11)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

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]:

```
1 # to be improved: probabilistic representation of positions for more position-un
2 # 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]:

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

In [93]:

```

1 if ((R3.delta - R2.delta) <= 0.3) and ((R3.delta - R1.delta) <= 0.3) and ((R2.de
2     # 0.3 rather than 0.1
3     print("cometa")# GHZ for all
4     qc_small.h(q[0]) # Hadamard
5     qc_small.cx(q[0], q[1]) # CNOT
6     qc_small.cx(q[0], q[2]) # CNOT
7     qc_small.cx(q[0], q[3]) # CNOT
8     qc_small.cx(q[0], q[4]) # CNOT
9     qc_small.cx(q[0], q[5]) # CNOT
10    qc_small.cx(q[0], q[6]) # CNOT
11    qc_small.cx(q[0], q[7]) # CNOT
12    # barrier
13    qc_small.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6], q[7])
14    # measures
15    qc_small.measure(q[0],c0[0])
16    qc_small.measure(q[3],c3[0])
17    qc_small.measure(q[6],c6[0])
18    qc_small.measure(q[1],c1[0])
19    qc_small.measure(q[4],c4[0])
20    qc_small.measure(q[7],c7[0])
21    # draw circuit
22    draw_circuit(qc_small)
23    # definition of quantum simulator
24    simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_
25    qc_small = transpile(qc_small, simulator)
26    # Run and get counts
27    result = simulator.run(qc_small, shots=1024).result()
28    counts_GHZ_all = result.get_counts(qc_small)
29    counts_GHZ = counts_GHZ_all.most_frequent() # does not work if multiple stat
30    # decide something if multiple states have the same count --> e.g., ``choose
31    print(counts_GHZ_all)
32    print(counts_GHZ)
33    #plot_histogram(counts_GHZ_all, title='outcomes')
34    #plot_histogram(counts_GHZ, title='outcomes')

```

In [94]:

```

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
5 print(arr_GHZ)
6 # 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]:

```

1 if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
2     print('gosh')
3     new_zero_betax = R1.betax + np.random.uniform(0,0.1)
4     new_zero_alphax = 1 - R1.betax
5     new_zero_betay = R1.betay + np.random.uniform(0,0.1)
6     new_zero_alphay = 1 - R2.betay + np.random.uniform(0,0.1)
7
8 if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
9     print('kinda')
10    new_zero_betax = R2.betax + np.random.uniform(0,0.1)
11    new_zero_alphax = 1 - R2.betax
12    new_zero_betay = R2.betay + np.random.uniform(0,0.1)
13    new_zero_alphay = 1 - R2.betay
14
15 if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
16     print('uffdah')
17     new_zero_betax = R3.betax + np.random.uniform(0,0.1)
18     new_zero_alphax = 1 - R3.betax
19     new_zero_betay = R3.betay + np.random.uniform(0,0.1)
20     new_zero_alphay = 1 - R3.betay

```

kinda

Define the 'new 1':

In [96]:

```

1 # flip thanks to the 'minus' sign?
2 # I had tried with if(R1... < R2...) etc., but it is not ok,
3 # because we need to initialize all elements.
4
5 if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
6     print('gosh')
7     new_one_betax = R1.betax - np.random.uniform(0,0.1)
8     new_one_alphax = 1 - R1.betax
9     new_one_betay = R1.betay - np.random.uniform(0,0.1)
10    new_one_alphay = 1 - R2.betay
11
12 if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
13     print('kinda')
14     new_one_betax = R2.betax - np.random.uniform(0,0.1)
15     new_one_alphax = 1 - R2.betax
16     new_one_betay = R2.betay - np.random.uniform(0,0.1)
17     new_one_alphay = 1 - R2.betay
18
19 if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
20     print('uffdah')
21     new_one_betax = R3.betax - np.random.uniform(0,0.1)
22     new_one_alphax = 1 - R3.betax
23     new_one_betay = R3.betay - np.random.uniform(0,0.1)
24     new_one_alphay = 1 - R3.betay

```

kinda

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

In [97]:

```

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_alphaay
6     R1.betay = new_zero_betay
7 if (arr_GHZ[2] =='0'):
8     R2.alphax = new_zero_alphax
9     R2.betax = new_zero_betax
10 if (arr_GHZ[3] =='0'):
11     R2.alphay = new_zero_alphaay
12     R2.betay = new_zero_betay
13 if (arr_GHZ[4] =='0'):
14     R2.alphax = new_zero_alphax
15     R2.betax = new_zero_betax
16 if (arr_GHZ[5] =='0'):
17     R3.alphax = new_zero_alphax
18     R3.betax = new_zero_betax
19 if (arr_GHZ[6] =='0'):
20     R3.alphay = new_zero_alphaay
21     R3.betay = new_zero_betay
22
23
24 if (arr_GHZ[0] =='1'):
25     R1.alphax = new_one_alphax
26     R1.betax = new_one_betax
27 if (arr_GHZ[1] =='1'):
28     R1.alphay = new_one_alphaay
29     R1.betay = new_one_betay
30 if (arr_GHZ[2] =='1'):
31     R2.alphax = new_one_alphax
32     R2.betax = new_one_betax
33 if (arr_GHZ[3] =='1'):
34     R2.alphay = new_one_alphaay
35     R2.betay = new_one_betay
36 if (arr_GHZ[4] =='1'):
37     R2.alphax = new_one_alphax
38     R2.betax = new_one_betax
39 if (arr_GHZ[5] =='1'):
40     R3.alphax = new_one_alphax
41     R3.betax = new_one_betax
42 if (arr_GHZ[6] =='1'):
43     R3.alphay = new_one_alphaay
44     R3.betay = new_one_betay
45

```

```

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_alphaay

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]:

```
1 R1.delta = reward(T,R1.betax,R1.betay)
2 R1.gamma = round((1 - R1.delta),2)
3 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]:

```
1 R2.delta = reward(T,R2.betax,R2.betay)
2 R2.gamma = round((1 - R2.delta),2)
3 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 :

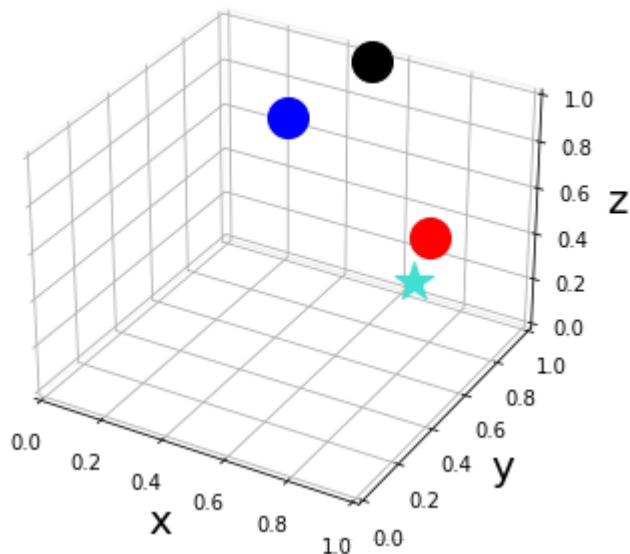
In [100]:

```
1 R3.delta = reward(T,R3.betax,R3.betay)
2 R3.gamma = round((1 - R3.delta),2)
3 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]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,
```



In [102]:

```

1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):

```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time12_ = audio1.overlay(audio2)           # combine , superimpose audio f
284 mixed_time12_ = mixed_time12_.overlay(audio3)      # further combine , supe
285
286 mixed_time12_.export("notes_/mixed_time12.mp3", format='mp3') # export mixed au
287 play(mixed_time12)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

```

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]:

```

1 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
6         R2.alphax = round(1 - R2.betax, 3)
7         R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
8         R2.alphay = round(1 - R2.betay, 3)
9         R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
10        R2.alphaz = round(1 - R2.betaz, 3)
11        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
12        R3.alphax = round(1 - R2.betax, 3)
13        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
14        R3.alphay = round(1 - R2.betay, 3)
15        R3.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
16        R3.alphaz = round(1 - R2.betaz, 3)
17    if (R2.delta > R1.delta and R2.delta > R3.delta):
18        print('quagga')
19        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
20        R1.alphax = round(1 - R1.betax, 3)
21        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
22        R1.alphay = round(1 - R1.betay, 3)
23        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
24        R1.alphaz = round(1 - R1.betaz, 3)
25        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
26        R3.alphax = round(1 - R3.betax, 3)
27        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
28        R3.alphay = round(1 - R3.betay, 3)
29        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
30        R3.alphaz = round(1 - R3.betaz, 3)
31    if (R3.delta > R1.delta and R3.delta > R2.delta):
32        print('quark')
33        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
34        R1.alphax = round(1 - R1.betax, 3)
35        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
36        R1.alphay = round(1 - R1.betay, 3)
37        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
38        R1.alphaz = round(1 - R1.betaz, 3)
39        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
40        R2.alphax = round(1 - R2.betax, 3)
41        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
42        R2.alphay = round(1 - R2.betay, 3)
43        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
44        R2.alphaz = round(1 - R2.betaz, 3)
45
46 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
47 print(R1.delta)
48
49 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
50 print(R2.delta)
51
52 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
53 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

```

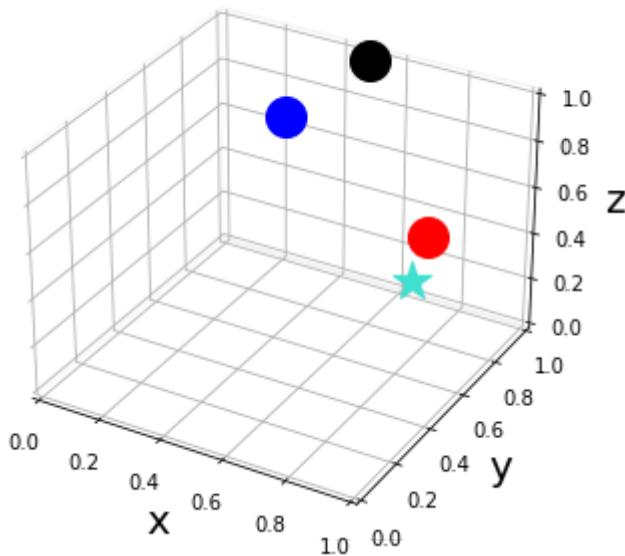
`TypeError: reward() missing 1 required positional argument: 'betaz'`

In [106]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4[0], R4[1], R4[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,

```



In [107]:

```

1 # audio 1, R_1
2
3 if(R1.betaz >= 0.5):
4     if (R1.betax == 0):
5         if (R1.betay == 0.5):
6             audio1 = AudioSegment.from_file("notes_/tC.mp3")
7             print("tC")
8         if (R1.betax > 0 and R1.betax <= 0.17):
9             if (R1.betay < 0.5):
10                 audio1 = AudioSegment.from_file("notes_/tB.mp3")
11                 print("tB")
12             if (R1.betay >= 0.5):
13                 audio1 = AudioSegment.from_file("notes_/tC#.mp3")
14                 print("tC#")
15         if (R1.betax > 0.17 and R1.betax <= 0.3):
16             if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
17                 audio1 = AudioSegment.from_file("notes_/tA#.mp3")
18                 print("tA#")
19             if (R1.betay >= 0.5):
20                 audio1 = AudioSegment.from_file("notes_/tD.mp3")
21                 print("tD")
22         if (R1.betax > 0.3 and R1.betax <= 0.5):
23             if (R1.betay < 0.5): # (R1.betay == 1):
24                 audio1 = AudioSegment.from_file("notes_/tD#.mp3")
25                 print("tD#")
26             if (R1.betay >= 0.5):
27                 audio1 = AudioSegment.from_file("notes_/tA.mp3")
28                 print("tA")
29         if (R1.betax > 0.5 and R1.betax <= 0.64):
30             if (R1.betay < 0.5):
31                 audio1 = AudioSegment.from_file("notes_/tE.mp3")
32                 print("tE")
33             if (R1.betay >= 0.5):
34                 audio1 = AudioSegment.from_file("notes_/tG#.mp3")
35                 print("tG#")
36         if (R1.betax > 0.64 and R1.betax <= 0.84):
37             if (R1.betay < 0.5):
38                 audio1 = AudioSegment.from_file("notes_/tF.mp3")
39                 print("tF")
40             if (R1.betay >= 0.5):
41                 audio1 = AudioSegment.from_file("notes_/tG.mp3")
42                 print("tG")
43         if (R1.betax > 0.84 and R1.betax <= 1):
44             #if (R1.betay == 0.5):
45             audio1 = AudioSegment.from_file("notes_/tF#.mp3")
46             print("tF#")
47 if(R1.betaz < 0.5):
48     if (R1.betax == 0):
49         if (R1.betay == 0.5):
50             audio1 = AudioSegment.from_file("notes_/tC2.mp3")
51             print("tC2")
52     if (R1.betax > 0 and R1.betax <= 0.17):
53         if (R1.betay < 0.5):
54             audio1 = AudioSegment.from_file("notes_/tB2.mp3")
55             print("tB2")
56         if (R1.betay >= 0.5):
57             audio1 = AudioSegment.from_file("notes_/tC#2.mp3")
58             print("tC#2")
59     if (R1.betax > 0.17 and R1.betax <= 0.3):

```

```
60     if (R1.betay < 0.5): # if (R1.betay >= 0.17 and R1.betay < 0.3):
61         audio1 = AudioSegment.from_file("notes_/tA#2.mp3")
62         print("tA#2")
63     if (R1.betay >= 0.5):
64         audio1 = AudioSegment.from_file("notes_/tD2.mp3")
65         print("tD2")
66 if (R1.betax > 0.3 and R1.betax <= 0.5):
67     if (R1.betay < 0.5): # (R1.betay == 1):
68         audio1 = AudioSegment.from_file("notes_/tD#2.mp3")
69         print("tD#2")
70     if (R1.betay >= 0.5):
71         audio1 = AudioSegment.from_file("notes_/tA2.mp3")
72         print("tA2")
73 if (R1.betax > 0.5 and R1.betax <= 0.64):
74     if (R1.betay < 0.5):
75         audio1 = AudioSegment.from_file("notes_/tE2.mp3")
76         print("tE2")
77     if (R1.betay >= 0.5):
78         audio1 = AudioSegment.from_file("notes_/tG#2.mp3")
79         print("tG#2")
80 if (R1.betax > 0.64 and R1.betax <= 0.84):
81     if (R1.betay < 0.5):
82         audio1 = AudioSegment.from_file("notes_/tF2.mp3")
83         print("tF2")
84     if (R1.betay >= 0.5):
85         audio1 = AudioSegment.from_file("notes_/tG2.mp3")
86         print("tG2")
87 if (R1.betax > 0.84 and R1.betax <= 1):
88     #if (R1.betay == 0.5):
89     audio1 = AudioSegment.from_file("notes_/tF#2.mp3")
90     print("tF#2")
91
92
93
94 # CHANGE from this point
95
96
97 # audio 2, R_2
98
99 if(R2.betaz < 0.5):
100     if (R2.betax == 0):
101         if (R2.betay == 0.5):
102             audio2 = AudioSegment.from_file("notes_/fC2.mp3")
103             print("fC2")
104     if (R2.betax > 0 and R2.betax <= 0.17):
105         if (R2.betay < 0.5):
106             audio2 = AudioSegment.from_file("notes_/fB2.mp3")
107             print("fB2")
108         if (R2.betay >= 0.5):
109             audio2 = AudioSegment.from_file("notes_/fC#2.mp3")
110             print("fC#2")
111     if (R2.betax > 0.17 and R2.betax <= 0.3):
112         if (R2.betay < 0.5):
113             audio2 = AudioSegment.from_file("notes_/fA#2.mp3")
114             print("fA#2")
115         if (R2.betay >= 0.5):
116             audio2 = AudioSegment.from_file("notes_/fD2.mp3")
117             print("fD2")
118     if (R2.betax > 0.3 and R2.betax <= 0.5):
119         if (R2.betay < 0.5): # (R1.betay == 1):
120             audio2 = AudioSegment.from_file("notes_/fD#2.mp3")
```

```
121         print("fD#2")
122     if (R2.betay >= 0.5):
123         audio2 = AudioSegment.from_file("notes_/fA2.mp3")
124         print("fA2")
125     if (R2.betax > 0.5 and R2.betax <= 0.64):
126         if (R2.betay < 0.5):
127             audio2 = AudioSegment.from_file("notes_/fE2.mp3")
128             print("fE2")
129         if (R2.betay >= 0.5):
130             audio2 = AudioSegment.from_file("notes_/fG#2.mp3")
131             print("fG#2")
132     if (R2.betax > 0.64 and R2.betax <= 0.84):
133         if (R2.betay < 0.5):
134             audio2 = AudioSegment.from_file("notes_/fF2.mp3")
135             print("fF2")
136         if (R2.betay >= 0.5):
137             audio2 = AudioSegment.from_file("notes_/fG2.mp3")
138             print("fG2")
139     if (R2.betax > 0.84 and R2.betax <= 1):
140         #if (R2.betay == 0.5):
141         audio2 = AudioSegment.from_file("notes_/fF#2.mp3")
142         print("fF#2")
143 if(R2.betaz >= 0.5):
144     if (R2.betax == 0):
145         if (R2.betay == 0.5):
146             audio2 = AudioSegment.from_file("notes_/fC.mp3")
147             print("fC")
148     if (R2.betax > 0 and R2.betax <= 0.17):
149         if (R2.betay < 0.5):
150             audio2 = AudioSegment.from_file("notes_/fB.mp3")
151             print("fB")
152         if (R2.betay >= 0.5):
153             audio2 = AudioSegment.from_file("notes_/fC#.mp3")
154             print("fC#")
155     if (R2.betax > 0.17 and R2.betax <= 0.3):
156         if (R2.betay < 0.5):
157             audio2 = AudioSegment.from_file("notes_/fA#.mp3")
158             print("fA#")
159         if (R2.betay >= 0.5):
160             audio2 = AudioSegment.from_file("notes_/fD.mp3")
161             print("fD")
162     if (R2.betax > 0.3 and R2.betax <= 0.5):
163         if (R2.betay < 0.5): # (R1.betay == 1):
164             audio2 = AudioSegment.from_file("notes_/fD#.mp3")
165             print("fD#")
166         if (R2.betay >= 0.5):
167             audio2 = AudioSegment.from_file("notes_/fA.mp3")
168             print("fA")
169     if (R2.betax > 0.5 and R2.betax <= 0.64):
170         if (R2.betay < 0.5):
171             audio2 = AudioSegment.from_file("notes_/fE.mp3")
172             print("fE")
173         if (R2.betay >= 0.5):
174             audio2 = AudioSegment.from_file("notes_/fG#.mp3")
175             print("fG#")
176     if (R2.betax > 0.64 and R2.betax <= 0.84):
177         if (R2.betay < 0.5):
178             audio2 = AudioSegment.from_file("notes_/fF.mp3")
179             print("fF")
180         if (R2.betay >= 0.5):
181             audio2 = AudioSegment.from_file("notes_/fG.mp3")
```

```
182         print("fG")
183     if (R2.betax > 0.84 and R2.betax <= 1):
184         #if (R2.betay == 0.5):
185         audio2 = AudioSegment.from_file("notes_/fF#.mp3")
186         print("fF#")
187
188
189
190
191
192 # audio 3, R_3
193
194 if (R3.betaz >= 0.5):
195     if (R3.betax == 0):
196         if (R3.betay == 0.5):
197             audio3 = AudioSegment.from_file("notes_/cc.mp3")
198             print("cC")
199     if (R3.betax > 0 and R3.betax <= 0.17):
200         if (R3.betay < 0.5):
201             audio3 = AudioSegment.from_file("notes_/cB.mp3")
202             print("cB")
203         if (R3.betay >= 0.5):
204             audio3 = AudioSegment.from_file("notes_/cc#.mp3")
205             print("cC#")
206     if (R3.betax > 0.17 and R3.betax <= 0.3):
207         if (R3.betay < 0.5):
208             audio3 = AudioSegment.from_file("notes_/cA#.mp3")
209             print("cA#")
210         if (R3.betay >= 0.5):
211             audio3 = AudioSegment.from_file("notes_/cD.mp3")
212             print("cD")
213     if (R3.betax > 0.3 and R3.betax <= 0.5):
214         if (R3.betay < 0.5):
215             audio3 = AudioSegment.from_file("notes_/cD#.mp3")
216             print("cD#")
217         if (R3.betay >= 0.5):
218             audio3 = AudioSegment.from_file("notes_/cA.mp3")
219             print("cA")
220     if (R3.betax > 0.5 and R3.betax <= 0.64):
221         if (R3.betay < 0.5):
222             audio3 = AudioSegment.from_file("notes_/cE.mp3")
223             print("cE")
224         if (R3.betay >= 0.5):
225             audio3 = AudioSegment.from_file("notes_/cG#.mp3")
226             print("cG#")
227     if (R3.betax > 0.64 and R3.betax <= 0.84):
228         if (R3.betay < 0.5):
229             audio3 = AudioSegment.from_file("notes_/cF.mp3")
230             print("cF")
231         if (R3.betay >= 0.5):
232             audio3 = AudioSegment.from_file("notes_/cG.mp3")
233             print("cG")
234     if (R3.betax > 0.84 and R3.betax <= 1):
235         #if (R3.betay == 0.5):
236         audio3 = AudioSegment.from_file("notes_/cF#.mp3")
237         print("cF#")
238     if (R3.betaz < 0.5):
239         if (R3.betax == 0):
240             if (R3.betay == 0.5):
241                 audio3 = AudioSegment.from_file("notes_/cc2.mp3")
242                 print("cC2")
```

```

243     if (R3.betax > 0 and R3.betax <= 0.17):
244         if (R3.betay < 0.5):
245             audio3 = AudioSegment.from_file("notes_/cB2.mp3")
246             print("cB2")
247         if (R3.betay >= 0.5):
248             audio3 = AudioSegment.from_file("notes_/cC#2.mp3")
249             print("cC#2")
250     if (R3.betax > 0.17 and R3.betax <= 0.3):
251         if (R3.betay < 0.5):
252             audio3 = AudioSegment.from_file("notes_/cA#2.mp3")
253             print("cA#2")
254         if (R3.betay >= 0.5):
255             audio3 = AudioSegment.from_file("notes_/cD2.mp3")
256             print("cD2")
257     if (R3.betax > 0.3 and R3.betax <= 0.5):
258         if (R3.betay < 0.5):
259             audio3 = AudioSegment.from_file("notes_/cD#2.mp3")
260             print("cD#2")
261         if (R3.betay >= 0.5):
262             audio3 = AudioSegment.from_file("notes_/cA2.mp3")
263             print("cA2")
264     if (R3.betax > 0.5 and R3.betax <= 0.64):
265         if (R3.betay < 0.5):
266             audio3 = AudioSegment.from_file("notes_/cE2.mp3")
267             print("cE2")
268         if (R3.betay >= 0.5):
269             audio3 = AudioSegment.from_file("notes_/cG#2.mp3")
270             print("cG#2")
271     if (R3.betax > 0.64 and R3.betax <= 0.84):
272         if (R3.betay < 0.5):
273             audio3 = AudioSegment.from_file("notes_/cF2.mp3")
274             print("cF2")
275         if (R3.betay >= 0.5):
276             audio3 = AudioSegment.from_file("notes_/cG2.mp3")
277             print("cG2")
278     if (R3.betax > 0.84 and R3.betax <= 1):
279         #if (R3.betay == 0.5):
280         audio3 = AudioSegment.from_file("notes_/cF#2.mp3")
281         print("cF#2")
282
283 mixed_time13_ = audio1.overlay(audio2)           # combine , superimpose audio f
284 mixed_time13_ = mixed_time13_.overlay(audio3)      # further combine , supe
285
286 mixed_time13_.export("notes_/mixed_time13.mp3", format='mp3') # export mixed au
287 play(mixed_time13)                                # play mixed audio file
288 # change this line at each time point, so in the end we can get a little piece
289

```

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

Input #0, wav, from '/var/folders/tc/5k6bdv0s421bnc52mnnj7p_w0000gn/T/tmp5s4154d6.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, s16, 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 <module>  
----> 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]:

```
(0.0809999999999996,  
 0.9434134380350225,  
 0.4230000000000004,  
 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]:

```
1 # Recalculate delta according to the second target (e.g., getting back to the ne
2
3 R1.delta = reward(T2, R1.betax, R1.betay)
4 print(R1.delta)
5
6 R2.delta = reward(T2, R2.betax, R2.betay)
7 print(R2.delta)
8
9 R3.delta = reward(T2, R3.betax, R3.betay)
10 print(R3.delta)
```

0.24
0.24
0.24

Shuffle only one

In [82]:

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

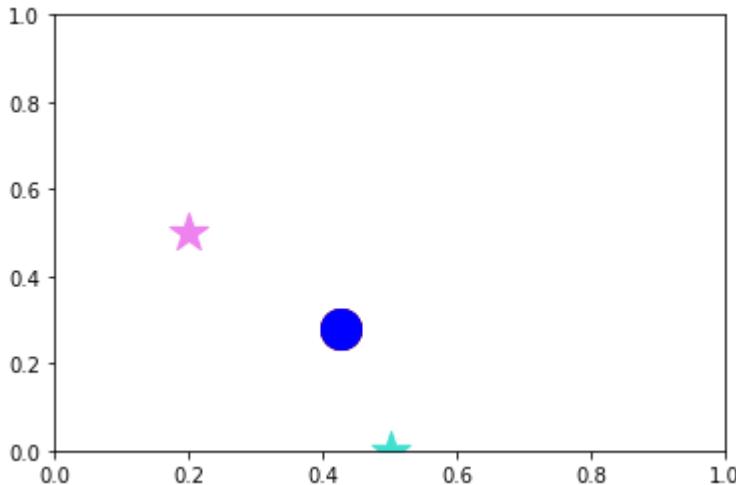
graph

In [83]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```
1 # threshold for initial reward
2 # random fluctuations
3
4 if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
5     print("SOS")
6     # R1
7     R1.alphax = round(np.random.uniform(0,0.9), 3)
8     R1.betax = round(1 - R1.alphax, 3)
9     print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
10    R1.alphay = round(np.random.uniform(0,0.9), 3)
11    R1.betay = round(1 - R1.alphay, 3)
12    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
13    # R2
14    R2.alphax = round(np.random.uniform(0,0.9), 3)
15    R2.betax = round(1 - R2.alphax, 3)
16    print("the new x-positions for R2 are: ", R2.alphax, R1.betax)
17    R2.alphay = round(np.random.uniform(0,0.9), 3)
18    R2.betay = round(1 - R2.alphay, 3)
19    print("the new y-positions for R2 are: ", R2.alphay, R1.betay)
20    # R3
21    R3.alphax = round(np.random.uniform(0,0.9), 3)
22    R3.betax = round(1 - R3.alphax, 3)
23    print("the new x-positions for R3 are: ", R3.alphax, R1.betax)
24    R3.alphay = round(np.random.uniform(0,0.9), 3)
25    R3.betay = round(1 - R3.alphay, 3)
26    print("the new y-positions for R3 are: ", R3.alphay, R1.betay)
27
28 R1.delta = reward(T2, R1.betax, R1.betay)
29 R1.gamma = 1 - R1.delta
30 R2.delta = reward(T2, R2.betax, R2.betay)
31 R2.gamma = 1 - R2.delta
32 R3.delta = reward(T2, R3.betax, R3.betay)
33 R3.gamma = 1 - R3.delta
34 print(R1.delta, R2.delta, R3.delta)
```

0.69 0.69 0.69

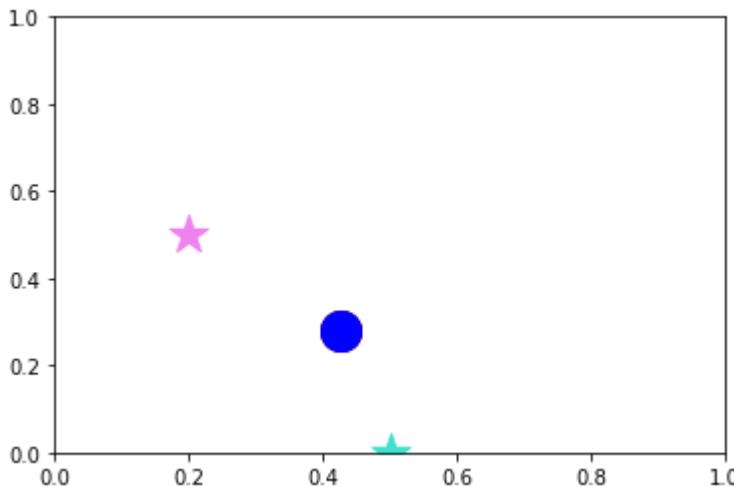
graph

In [85]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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

reach the most successful robot

In [86]:

```

1 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later.
6         R2.alphax = round(1 - R2.betax, 3)
7         R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
8         R2.alphay = round(1 - R2.betay, 3)
9         R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
10        R2.alphaz = round(1 - R2.betaz, 3)
11        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
12        R3.alphax = round(1 - R2.betax, 3)
13        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
14        R3.alphay = round(1 - R2.betay, 3)
15        R3.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
16        R3.alphaz = round(1 - R2.betaz, 3)
17     if (R2.delta > R1.delta and R2.delta > R3.delta):
18         print('quagga')
19         R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
20         R1.alphax = round(1 - R1.betax, 3)
21         R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
22         R1.alphay = round(1 - R1.betay, 3)
23         R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
24         R1.alphaz = round(1 - R1.betaz, 3)
25         R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
26         R3.alphax = round(1 - R3.betax, 3)
27         R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
28         R3.alphay = round(1 - R3.betay, 3)
29         R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
30         R3.alphaz = round(1 - R3.betaz, 3)
31     if (R3.delta > R1.delta and R3.delta > R2.delta):
32         print('quark')
33         R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
34         R1.alphax = round(1 - R1.betax, 3)
35         R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
36         R1.alphay = round(1 - R1.betay, 3)
37         R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
38         R1.alphaz = round(1 - R1.betaz, 3)
39         R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
40         R2.alphax = round(1 - R2.betax, 3)
41         R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
42         R2.alphay = round(1 - R2.betay, 3)
43         R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
44         R2.alphaz = round(1 - R2.betaz, 3)
45
46 R1.delta = reward(T, R1.betax, R1.betay, R1.betaz)
47 print(R1.delta)
48
49 R2.delta = reward(T, R2.betax, R2.betay, R2.betaz)
50 print(R2.delta)
51
52 R3.delta = reward(T, R3.betax, R3.betay, R3.betaz)
53 print(R2.delta)

```

0.69
0.69
0.69

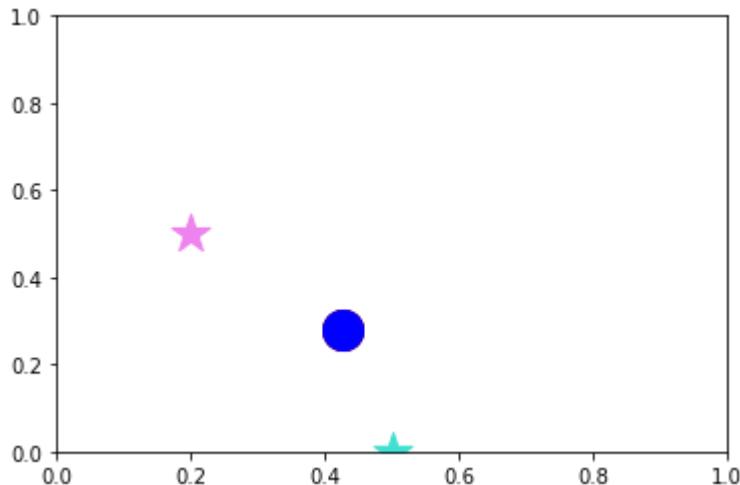
plot

In [87]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```
1 # Another round of SOS re-shuffle
2
3 # threshold for initial reward
4 # random fluctuations
5
6 if (R1.delta <= 0.4) and (R2.delta <= 0.4) and (R3.delta <= 0.4):
7     print("SOS")
8     # R1
9     R1.alphax = round(np.random.uniform(0,0.9), 3)
10    R1.betax = round(1 - R1.alphax, 3)
11    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
12    R1.alphay = round(np.random.uniform(0,0.9), 3)
13    R1.betay = round(1 - R1.alphay, 3)
14    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
15    # R2
16    R2.alphax = round(np.random.uniform(0,0.9), 3)
17    R2.betax = round(1 - R2.alphax, 3)
18    print("the new x-positions for R2 are: ", R2.alphax, R2.betax)
19    R2.alphay = round(np.random.uniform(0,0.9), 3)
20    R2.betay = round(1 - R2.alphay, 3)
21    print("the new y-positions for R2 are: ", R2.alphay, R2.betay)
22    # R3
23    R3.alphax = round(np.random.uniform(0,0.9), 3)
24    R3.betax = round(1 - R3.alphax, 3)
25    print("the new x-positions for R3 are: ", R3.alphax, R3.betax)
26    R3.alphay = round(np.random.uniform(0,0.9), 3)
27    R3.betay = round(1 - R3.alphay, 3)
28    print("the new y-positions for R3 are: ", R3.alphay, R3.betay)
29
30 R1.delta = reward(T2, R1.betax, R1.betay)
31 R1.gamma = 1 - R1.delta
32 R2.delta = reward(T2, R2.betax, R2.betay)
33 R2.gamma = 1 - R2.delta
34 R3.delta = reward(T2, R3.betax, R3.betay)
35 R3.gamma = 1 - R3.delta
36 print(R1.delta, R2.delta, R3.delta)
```

0.69 0.69 0.69

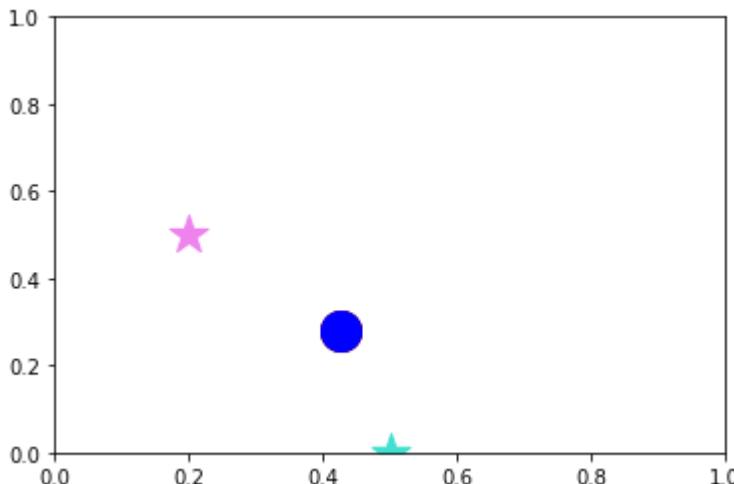
graph

In [89]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```

1 q = QuantumRegister(5, 'q'); # qubits # changed to 9, formerly 15
2 m2 = ClassicalRegister(1, 'c1'); # classical bits (separated is better)
3 m3 = ClassicalRegister(1, 'c2');
4 m4 = ClassicalRegister(1, 'c3');
5
6 qc4 = QuantumCircuit(q, m2, m3, m4); # to get back to the nest

```

Gate! SECOND GATE

In [91]:

```
1 if (R1.delta > R2.delta) and (R1.delta > R3.delta):
2     if (R1.alphax < 0.3): # I have to customize state vectors according to prec
3         qc4.x(q[0])           # just using the NOT gate as a test
4     if (R1.alphax == 0.5): # I have to customize state vectors according to pre
5         qc4.h(q[0])
6     if (R1.alphax >= 0.3) and (R2.alphax < 0.5):
7         print('jungle!')
8         qc4.ry(1.9106332, q[0])
9     if (R1.alphax >= 0.6) and (R2.alphax < 0.7):
10        print('ocean!')
11        qc4.ry(1.2309594, q[0])
12    if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
13        qc4.x(q[1])
14    if (R1.alphay == 0.5): # I have to customize state vectors according to pre
15        qc4.h(q[1])
16    if (R1.alphay >= 0.3) and (R2.alphay < 0.5):
17        print('jungle!')
18        qc4.ry(1.9106332, q[1])
19    if (R1.alphay >= 0.6) and (R2.alphay < 0.7):
20        print('ocean!')
21        qc4.ry(1.2309594, q[1])
22    if (R1.delta == 0.5):
23        qc4.h(q[2])
24    if (R1.delta == 0.6):
25        qc4.h(q[2])
26    if (R1.delta >= 0.7):
27        qc4.x(q[2])
28    if (R1.gamma >= 0.3) and (R2.gamma < 0.5):
29        print('jungle!')
30        qc4.ry(1.9106332, q[2])
31    if (R1.gamma >= 0.6) and (R2.gamma < 0.7):
32        print('ocean!')
33        qc4.ry(1.2309594, q[2])
34 elif (R1.delta > R2.delta) and (R1.delta > R3.delta):
35     print('dog')
36     if (R1.alphax < 0.3): # I have to customize state vectors according to prec
37         qc4.x(q[0])           # just using the NOT gate as a test
38     if (R1.alphax == 0.5): # I have to customize state vectors according to pre
39         qc4.h(q[0])
40     if (R1.alphax >= 0.3) and (R1.alphax < 0.5):
41         print('jungle!')
42         qc4.ry(1.9106332, q[0])
43     if (R1.alphax >= 0.6) and (R1.alphax < 0.7):
44         print('ocean!')
45         qc4.ry(1.2309594, q[0])
46     if (R1.alphay <= 0.2): # else: the qubit sticks with the default value '0'
47         qc4.x(q[1])
48     if (R1.alphay == 0.5): # I have to customize state vectors according to pre
49         qc4.h(q[1])
50     if (R1.alphay >= 0.3) and (R1.alphay < 0.5):
51         print('jungle!')
52         qc4.ry(1.9106332, q[1])
53     if (R1.alphay >= 0.6) and (R1.alphay < 0.7):
54         print('ocean!')
55         qc4.ry(1.2309594, q[1])
56     if (R1.delta == 0.5):
57         qc4.h(q[2])
58     if (R1.delta == 0.6):
59         qc4.h(q[2])
```

```
60     if (R1.delta >= 0.7):
61         qc4.x(q[2])
62     if (R1.gamma >= 0.3) and (R1.gamma < 0.5):
63         print('jungle!')
64         qc4.ry(1.9106332, q[2])
65     if (R1.gamma >= 0.6) and (R1.gamma < 0.7):
66         print('ocean!')
67         qc4.ry(1.2309594, q[2])
68 else:
69     print('cat') # I made some tests to check the IF conditions
70     if (R3.alphax < 0.3):
71         qc4.x(q[0])
72     if (R3.alphax == 0.5):
73         qc4.h(q[0])
74     if (R3.alphax >= 0.3) and (R3.alphax < 0.5):
75         print('jungle!')
76         qc4.ry(1.9106332, q[0])
77     if (R3.alphax >= 0.6) and (R3.alphax < 0.7):
78         print('ocean!')
79         qc4.ry(1.2309594, q[0])
80     if (R3.alphay < 0.3):
81         qc4.x(q[1])
82     if (R3.alphay == 0.5):
83         qc4.h(q[1])
84     if (R3.alphay >= 0.3) and (R3.alphay < 0.5):
85         print('jungle!')
86         qc4.ry(1.9106332, q[1])
87     if (R3.alphay >= 0.6) and (R3.alphay < 0.7):
88         print('ocean!')
89         qc4.ry(1.2309594, q[1])
90     if (R3.delta == 0.5):
91         qc4.h(q[2])
92     if (R3.delta == 0.6):
93         qc4.h(q[2])
94     if (R3.delta >= 0.7):
95         qc4.x(q[2])
96     if (R3.gamma >= 0.3) and (R3.gamma < 0.5):
97         print('jungle!')
98         qc4.ry(1.9106332, q[2])
99     if (R3.gamma >= 0.6) and (R3.gamma < 0.7):
100        print('ocean!')
101        qc4.ry(1.2309594, q[2])
```

cat
jungle!

the core code

In [92]:

```

1 # this is the core code, and it is unchanged across time
2
3 qc4.barrier(q)
4 qc4.ccx(q[0],q[1],q[3])
5 qc4.ccx(q[0],q[1],q[4])
6
7 qc4.reset(q[3]);
8 qc4.reset(q[4]);
9
10 qc4.ccx(q[0],q[2],q[3])
11 qc4.ccx(q[1],q[2],q[4])
12
13 qc4.x(q[2])
14
15 qc4.ch(q[2],q[3])
16 qc4.ch(q[2],q[4])
17
18 qc4.x(q[2])
19
20 qc4.barrier(q)
21
22 # perform measurements and store them in classical bits
23
24 qc4.measure(q[2],m2[0])
25 qc4.measure(q[3],m3[0])
26 qc4.measure(q[4],m4[0])
27
28 # visualization of the circuit
29
30 draw_circuit(qc4)
31
32 # definition of quantum simulator
33
34 simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_simulator
35 qc4 = transpile(qc4, simulator)
36 cc = collections.Counter()
37
38 # Run and get counts
39 result = simulator.run(qc4, shots=1024).result()
40 counts = result.get_counts(qc4)
41 counts2 = counts.most_frequent() # does not work if multiple states have the same count
42 # decide something if multiple states have the same count --> e.g., ``choose the
43 counts3 = cc.most_common(2)
44 print(counts)
45 print(counts2)
46 print(counts3)
47 result = simulator.run(qc4, shots=10, memory=True).result()
48 memory = result.get_memory(qc4)
49 print(memory)
50 plot_histogram(counts, title='outcomes')
51 # TAKE the TWO more present outcomes

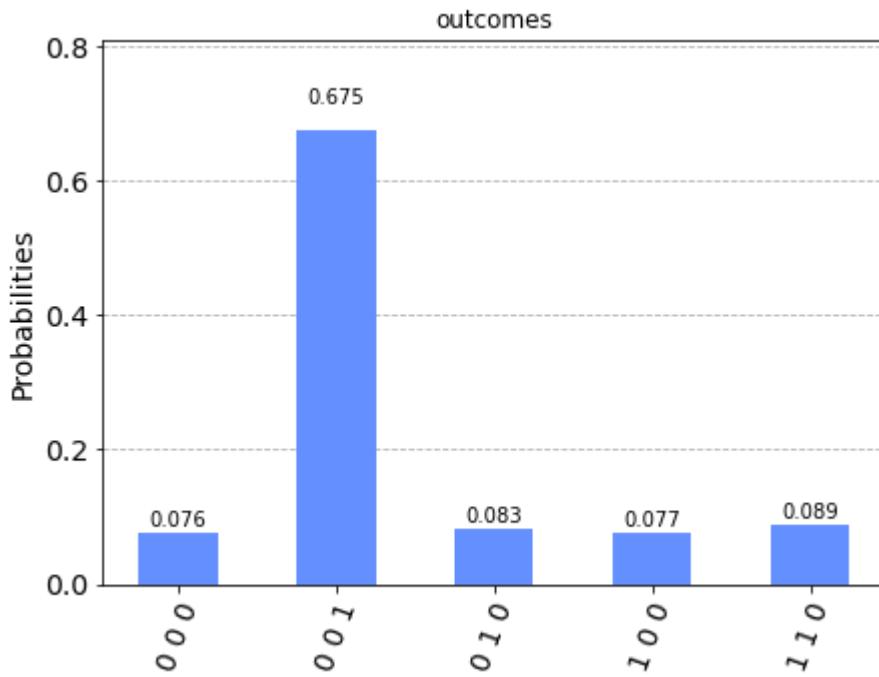
```

CircuitComposer(circuit=<qiskit.circuit.quantumcircuit.QuantumCircuit object at 0x7fb66d793ee0>, editable=False)

```
{'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 of consistency  
2 # The arrays are automatically rewritten...
```

My intervention is needed here, where indicated:

In [93]:

```

1 print(counts2) # order: R3, R2, R1. Add some uncertainty?
2 # export as an array
3 str = counts2
4 arr1 = str.split(' ') # to split the string and avoid empty spaces as array elements
5 print(arr1)
6 weight1 = 691 # AT HAND ONLY FOR NOW
7
8 arr2 = ['1','1','0'] # 111 # 011
9 print(arr2)
10 weight2 = 91
11 # BY HAND ONLY FOR NOW
12
13
14 # an attempt, not so good, to automatize this passage:
15
16 print(memory)
17
18 data = Counter(memory)
19 data.most_common() # Returns all unique items and their counts
20 data.most_common(3)
21
22 print(data.most_common())
23 print(data.most_common(1))
24 arrx1 = data.most_common(2)[0]
25 print(arrx1)
26 arrx2 = data.most_common(2)[1]
27 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]:

1	arr1, arr2
---	------------

Out[94]:

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

reset position of R_1

In [95]:

```

1 if (R1.delta > R2.delta) and (R1.delta > R3.delta):
2     # if R1 didn't enter the gate, keep its position
3     R1.alphax = R1.alphax
4     R1.betax = R1.betax
5     R1.alphay = R1.alphay
6     R1.betay = R1.betay
7 else:
8     # same outcome = 1 # January 23
9     # x part
10
11
12     # change [0] --> [1] and vice versa, January 28
13 if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
14     print("bla")
15     R1.alphax = 0.3
16     R1.betax = 0.7
17 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
18     print("gulp")
19     R1.alphax = 0.7
20     R1.betax = 0.3
21 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
22     print("stra-gulp")
23     R1.alphax = 0.5 # change temporarily made on January 24: random generat
24     R1.betax = 0.5 # same as above
25 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
26     print("thunderstorm!")
27     R1.alphax = 0
28     R1.betax = 1
29 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
30     print("avalanche!")
31     R1.alphax = 0.1
32     R1.betax = 0.9
33 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
34     print("earthquake!")
35     R1.alphax = 0.9
36     R1.betax = 0.1
37 elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
38     print("avalanche bis!")
39     R1.alphax = 0.1 # the same also in this case
40     R1.betax = 0.9 # the same also in this case
41 # same = outcome 0 # January 23
42 elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
43     print("bla 2")
44     R1.alphax = 0.7 # the opposite??
45     R1.betax = 0.3 # the opposite??
46 elif (arr1[1] == arr2[0]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
47     print("gulp 2")
48     R1.alphax = 0.3 # the opposite??
49     R1.betax = 0.7 # the opposite??
50 elif (arr1[1] == arr2[0]) and (arr1[1] == '0') and (weight1 == weight2 or n
51     print("stra-gulp 2")
52     R1.alphax = 0.5 # change temporarily made on January 24: random generat
53     R1.betax = 0.5 # change temporarily made on January 24: random generato
54 # different outcomes
55 elif arr1[1] != arr2[1]: # January 23
56     print("blue")
57     if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
58         print("google 1")
59         R1.alphax = 0.5 # change temporarily made on January 24: random gen

```

```

 60      R1.betax = 0.5 # change temporarily made on January 24: random gene
 61  if (arr1[1] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.abs(
 62      # include the case of a very small difference!
 63      print("uffdah"))
 64      R1.alphax = 0.5
 65      R1.betax = 0.5
 66  if (arr1[1] == '1' and arr2[1] == '0'):
 67      if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
 68          print("abc")
 69          R1.alphax = 0.3
 70          R1.betax = 0.7
 71  if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
 72      print("bca")
 73      R1.alphax = 0.7
 74      R1.betax = 0.3
 75  if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
 76      print("news")
 77      R1.alphax = 0.2 #
 78      R1.betax = 0.8 #
 79  if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
 80      print("idea")
 81      R1.alphax = 0.8 #
 82      R1.betax = 0.2 #
 83  if (arr1[1] == '0') and (arr2[1] == '1'):
 84      if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
 85          print("bac")
 86          R1.alphax = 0.7
 87          R1.betax = 0.3
 88  if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
 89      print("cba")
 90      R1.alphax = 0.3
 91      R1.betax = 0.7
 92  if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
 93      print("brain")
 94      R1.alphax = 0.7 # 0.9
 95      R1.betax = 0.3 # 0.1
 96  if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
 97      print("hand")
 98      R1.alphax = 0.3 # 0.1
 99      R1.betax = 0.7 # 0.9
100  # y part
101
102  # change of January 26
103
104  if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
105      print("bla")
106      R1.alphay = 0.3
107      R1.betay = 0.7
108  elif (arr1[0] == arr2[1]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
109      print("gulp")
110      R1.alphay = 0.7
111      R1.betay = 0.3
112  elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
113      print("stra-gulp")
114      R1.alphay = 0.5 # change temporarily made on January 24: random generat
115      R1.betay = 0.5 # same as above
116  elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
117      print("thunderstorm!")
118      R1.alphay = 0
119      R1.betay = 1
120  elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2

```

```
121     print("avalanche!")
122     R1.alphay = 0.1
123     R1.betay = 0.9
124 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2):
125     print("earthquake!")
126     R1.alphay = 0.9
127     R1.betay = 0.1
128 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2):
129     print("avalanche bis!")
130     R1.alphay = 0.1 # the same also in this case
131     R1.betay = 0.9 # the same also in this case
132 # same = outcome 0 # January 23
133 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5):
134     print("bla 2")
135     R1.alphay = 0.7 # the opposite??
136     R1.betay = 0.3 # the opposite??
137 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5):
138     print("gulp 2")
139     R1.alphay = 0.3 # the opposite??
140     R1.betay = 0.7 # the opposite??
141 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
142     print("stra-gulp 2")
143     R1.alphay = 0.5 # change temporarily made on January 24: random generated
144     R1.betay = 0.5 # change temporarily made on January 24: random generated
145 # different outcomes
146 elif arr1[0] != arr2[0]: # January 23
147     print("blue")
148     if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
149         print("google 1")
150         R1.alphay = 0.5 # change temporarily made on January 24: random generated
151         R1.betay = 0.5 # change temporarily made on January 24: random generated
152     if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
153         # include the case of a very small difference!
154         print("uffdah")
155         R1.alphay = 0.5
156         R1.betay = 0.5
157     if (arr1[0] == '1' and arr2[0] == '0'):
158         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
159             print("abc")
160             R1.alphay = 0.3
161             R1.betay = 0.7
162         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
163             print("bca")
164             R1.alphay = 0.7
165             R1.betay = 0.3
166         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
167             print("news")
168             R1.alphay = 0.2 #
169             R1.betay = 0.8 #
170         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
171             print("idea")
172             R1.alphay = 0.8 #
173             R1.betay = 0.2 #
174     if (arr1[0] == '0' and arr2[0] == '1'):
175         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
176             print("bac")
177             R1.alphay = 0.7
178             R1.betay = 0.3
179         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
180             print("cba")
181             R1.alphay = 0.3
```

```
182     R1.betay = 0.7
183     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
184         print("brain")
185         R1.alphay = 0.7 # 0.9
186         R1.betay = 0.3 # 0.1
187     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
188         print("hand")
189         R1.alphay = 0.3 # 0.1
190         R1.betay = 0.7 # 0.9
```

blue
brain
blue
brain

reset position of R_2

In [96]:

```

1  if (R2.delta > R1.delta) and (R2.delta > R3.delta):
2      # if R2 didn't enter the gate, keep its position
3      R2.alphax = R2.alphax
4      R2.betax = R2.betax
5      R2.alphay = R2.alphay
6      R2.betay = R2.betay
7  else:
8      # same outcome = 1 # January 23
9      # x part
10     if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
11         print("bla")
12         R2.alphax = 0.3
13         R2.betax = 0.7
14     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
15         print("gulp")
16         R2.alphax = 0.7
17         R2.betax = 0.3
18     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
19         print("stra-gulp")
20         R2.alphax = 0.5 # change temporarily made on January 24: random generat
21         R2.betax = 0.5 # change temporarily made on January 24: random generato
22     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
23         print("thunderstorm!")
24         R2.alphax = 0
25         R2.betax = 1
26     elif (arr1[1] == arr2[0]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
27         print("avalanche!")
28         R2.alphax = 0.1
29         R2.betax = 0.9
30     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
31         print("earthquake!")
32         R2.alphax = 0.9
33         R2.betax = 0.1
34     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
35         print("avalanche bis!")
36         R2.alphax = 0.1 # the same also in this case
37         R2.betax = 0.9 # the same also in this case
38     # same = outcome 0 # January 23
39     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
40         print("bla 2")
41         R2.alphax = 0.7 # the opposite??
42         R2.betax = 0.3 # the opposite??
43     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
44         print("gulp 2")
45         R2.alphax = 0.3 # the opposite??
46         R2.betax = 0.7 # the opposite??
47     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
48         print("stra-gulp 2")
49         R2.alphax = 0.5 #1 # the opposite
50         R2.betax = 0.5 #0 # the opposite
51     # different outcomes
52     elif arr1[1] != arr2[1]: # January 23
53         print("blue")
54         if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
55             print("google 1")
56             R2.alphax = 0.5 # change temporarily made on January 24: random gen
57             R2.betax = 0.5 # change temporarily made on January 24: random gene
58         if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
59             # include the case of a very small difference!

```

```
60     print("uffdah")
61     R2.alphax = 0.5 # change temporarily made on January 24: random gen
62     R2.betax = 0.5 # change temporarily made on January 24: random gene
63 if (arr1[1] == '1' and arr2[1] == '0'):
64     print("gasp")
65     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
66         print("abc")
67         R2.alphax = 0.3
68         R2.betax = 0.7
69     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
70         print("bca")
71         R2.alphax = 0.7
72         R2.betax = 0.3
73     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
74         print("news")
75         R2.alphax = 0.3 # 0. 126 January
76         R2.betax = 0.7 # 0.9
77     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
78         print("idea")
79         R2.alphax = 0.7 # 0.9
80         R2.betax = 0.3 # 0.1
81 if (arr1[1] == '0') and (arr2[1] == '1'):
82     print("sigh")
83     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
84         print("bac")
85         R2.alphax = 0.7
86         R2.betax = 0.3
87     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
88         print("cba")
89         R2.alphax = 0.3
90         R2.betax = 0.7
91
92 # y part
93 # change of January 26
94
95 # to be modified: R1 to R2
96
97 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
98     print("bla")
99     R2.alphay = 0.3
100    R2.betay = 0.7
101 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
102     print("gulp")
103     R2.alphay = 0.7
104     R2.betay = 0.3
105 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
106     print("stra-gulp")
107     R2.alphay = 0.5 # change temporarily made on January 24: random generat
108     R2.betay = 0.5 # same as above
109 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
110     print("thunderstorm!")
111     R2.alphay = 0
112     R2.betay = 1
113 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
114     print("avalanche!")
115     R2.alphay = 0.1
116     R2.betay = 0.9
117 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
118     print("earthquake!")
119     R2.alphay = 0.9
120     R2.betay = 0.1
```

```

121     elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
122         print("avalanche bis!")
123         R2.alphay = 0.1 # the same also in this case
124         R2.betay = 0.9 # the same also in this case
125 # same = outcome 0 # January 23
126     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
127         print("bla 2")
128         R2.alphay = 0.7 # the opposite??
129         R2.betay = 0.3 # the opposite??
130     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
131         print("gulp 2")
132         R2.alphay = 0.3 # the opposite??
133         R2.betay = 0.7 # the opposite??
134     elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5)
135         print("stra-gulp 2")
136         R2.alphay = 0.5 # change temporarily made on January 24: random generated
137         R2.betay = 0.5 # change temporarily made on January 24: random generated
138 # different outcomes
139     elif arr1[0] != arr2[0]: # January 23
140         print("blue")
141         if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
142             print("google 1")
143             R2.alphay = 0.5 # change temporarily made on January 24: random generated
144             R2.betay = 0.5 # change temporarily made on January 24: random generated
145         if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
146             # include the case of a very small difference!
147             print("uffdah")
148             R2.alphay = 0.5
149             R2.betay = 0.5
150         if (arr1[0] == '1' and arr2[0] == '0'):
151             if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
152                 print("abc")
153                 R2.alphay = 0.3
154                 R2.betay = 0.7
155             if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
156                 print("bca")
157                 R2.alphay = 0.7
158                 R2.betay = 0.3
159             if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
160                 print("news")
161                 R2.alphay = 0.3 #
162                 R2.betay = 0.7 #
163             if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
164                 print("idea")
165                 R2.alphay = 0.7 #
166                 R2.betay = 0.3 #
167         if (arr1[0] == '0') and (arr2[0] == '1'):
168             if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
169                 print("bac")
170                 R2.alphay = 0.7
171                 R2.betay = 0.3
172             if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
173                 print("cba")
174                 R2.alphay = 0.3
175                 R2.betay = 0.7
176             if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
177                 print("brain")
178                 R2.alphay = 0.7 # 0.9
179                 R2.betay = 0.3 # 0.1
180             if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
181                 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]:

```

1  if (R2.delta > R1.delta) and (R2.delta > R3.delta):
2      # if R2 didn't enter the gate, keep its position
3      R2.alphax = R2.alphax
4      R2.betax = R2.betax
5      R2.alphay = R2.alphay
6      R2.betay = R2.betay
7  else:
8      # same outcome = 1 # January 23
9      # x part
10     if (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 50
11         print("bla")
12         R2.alphax = 0.3
13         R2.betax = 0.7
14     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 5
15         print("gulp")
16         R2.alphax = 0.7
17         R2.betax = 0.3
18     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and (weight1 == weight2 or n
19         print("stra-gulp")
20         R2.alphax = 0.5 # change temporarily made on January 24: random generat
21         R2.betax = 0.5 # change temporarily made on January 24: random generato
22     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 8
23         print("thunderstorm!")
24         R2.alphax = 0
25         R2.betax = 1
26     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
27         print("avalanche!")
28         R2.alphax = 0.1
29         R2.betax = 0.9
30     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight2 - weight1) > 2
31         print("earthquake!")
32         R2.alphax = 0.8
33         R2.betax = 0.2
34     elif (arr1[1] == arr2[1]) and (arr1[1] == '1') and ((weight1 - weight2) > 2
35         print("avalanche bis!")
36         R2.alphax = 0.2 # the same also in this case
37         R2.betax = 0.8 # the same also in this case # January 26; this or 0.3,
38     # same = outcome 0 # January 23
39     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight1 - weight2) > 5
40         print("bla 2")
41         R2.alphax = 0.7 # the opposite??
42         R2.betax = 0.3 # the opposite??
43     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and ((weight2 - weight1) > 5
44         print("gulp 2")
45         R2.alphax = 0.3 # the opposite??
46         R2.betax = 0.7 # the opposite??
47     elif (arr1[1] == arr2[1]) and (arr1[1] == '0') and (weight1 == weight2 or n
48         print("stra-gulp 2")
49         R2.alphax = 0.5 #1 # the opposite
50         R2.betax = 0.5 #0 # the opposite
51     # different outcomes
52     elif arr1[1] != arr2[1]: # January 23
53         print("blue")
54         if (arr1[1] != arr2[1]) and (weight1 == weight2 or np.absolute(weight1
55             print("google 1")
56             R2.alphax = 0.5 # change temporarily made on January 24: random gen
57             R2.betax = 0.5 # change temporarily made on January 24: random gene
58         if (arr1[1] == '1' and arr2[1] == '0') and (weight1 == weight2 or np.ab
59             # include the case of a very small difference!

```

```
60     print("uffdah")
61     R2.alphax = 0.5 # change temporarily made on January 24: random gen
62     R2.betax = 0.5 # change temporarily made on January 24: random gene
63 if (arr1[1] == '1' and arr2[1] == '0'):
64     print("gasp")
65     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
66         print("abc")
67         R2.alphax = 0.3
68         R2.betax = 0.7
69     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
70         print("bca")
71         R2.alphax = 0.7
72         R2.betax = 0.3
73     if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
74         print("news")
75         R2.alphax = 0.3 #
76         R2.betax = 0.7 #
77     if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
78         print("idea")
79         R2.alphax = 0.7 #
80         R2.betax = 0.3 #
81 if (arr1[1] == '0') and (arr2[1] == '1'):
82     print("sigh")
83     if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
84         print("bac")
85         R2.alphax = 0.7
86         R2.betax = 0.3
87     if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
88         print("cba")
89         R2.alphax = 0.3
90         R2.betax = 0.7
91
92 # y part
93 # change of January 26
94
95 # to be modified: R1 to R2
96
97 if (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 50
98     print("bla")
99     R2.alphay = 0.3
100    R2.betay = 0.7
101 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 5
102     print("gulp")
103     R2.alphay = 0.7
104     R2.betay = 0.3
105 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and (weight1 == weight2 or n
106     print("stra-gulp")
107     R2.alphay = 0.5 # change temporarily made on January 24: random generat
108     R2.betay = 0.5 # same as above
109 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 8
110     print("thunderstorm!")
111     R2.alphay = 0
112     R2.betay = 1
113 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
114     print("avalanche!")
115     R2.alphay = 0.1
116     R2.betay = 0.9
117 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight2 - weight1) > 2
118     print("earthquake!")
119     R2.alphay = 0.9
120     R2.betay = 0.1
```

```
121 elif (arr1[0] == arr2[0]) and (arr1[0] == '1') and ((weight1 - weight2) > 2
122     print("avalanche bis!")
123     R2.alphay = 0.1 # the same also in this case
124     R2.betay = 0.9 # the same also in this case
125 # same = outcome 0 # January 23
126 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight1 - weight2) > 5
127     print("bla 2")
128     R2.alphay = 0.7 # the opposite??
129     R2.betay = 0.3 # the opposite??
130 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and ((weight2 - weight1) > 5
131     print("gulp 2")
132     R2.alphay = 0.3 # the opposite??
133     R2.betay = 0.7 # the opposite??
134 elif (arr1[0] == arr2[0]) and (arr1[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
135     print("stra-gulp 2")
136     R2.alphay = 0.5 # change temporarily made on January 24: random generated
137     R2.betay = 0.5 # change temporarily made on January 24: random generated
138 # different outcomes
139 elif arr1[0] != arr2[0]: # January 23
140     print("blue")
141     if (arr1[0] != arr2[0]) and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
142         print("google 1")
143         R2.alphay = 0.5 # change temporarily made on January 24: random generated
144         R2.betay = 0.5 # change temporarily made on January 24: random generated
145     if (arr1[0] == '1' and arr2[0] == '0') and (weight1 == weight2 or np.abs(weight1 - weight2) <= 5):
146         # include the case of a very small difference!
147         print("uffdah")
148         R2.alphay = 0.5
149         R2.betay = 0.5
150     if (arr1[0] == '1' and arr2[0] == '0'):
151         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
152             print("abc")
153             R2.alphay = 0.3
154             R2.betay = 0.7
155         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
156             print("bca")
157             R2.alphay = 0.7
158             R2.betay = 0.3
159         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
160             print("news")
161             R2.alphay = 0.3 #
162             R2.betay = 0.7 #
163         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
164             print("idea")
165             R2.alphay = 0.7 #
166             R2.betay = 0.3 #
167     if (arr1[0] == '0' and arr2[0] == '1'):
168         if (weight1 - weight2 >= 50 and weight1 - weight2 <= 200):
169             print("bac")
170             R2.alphay = 0.7
171             R2.betay = 0.3
172         if (weight2 - weight1 >= 50 and weight1 - weight2 <= 200):
173             print("cba")
174             R2.alphay = 0.3
175             R2.betay = 0.7
176         if ((weight1 - weight2) > 200 and (weight1 - weight2) < 800): # no
177             print("brain")
178             R2.alphay = 0.7 # 0.9
179             R2.betay = 0.3 # 0.1
180         if ((weight2 - weight1) > 200 and (weight2 - weight1) < 800): # no
181             print("hand")
```

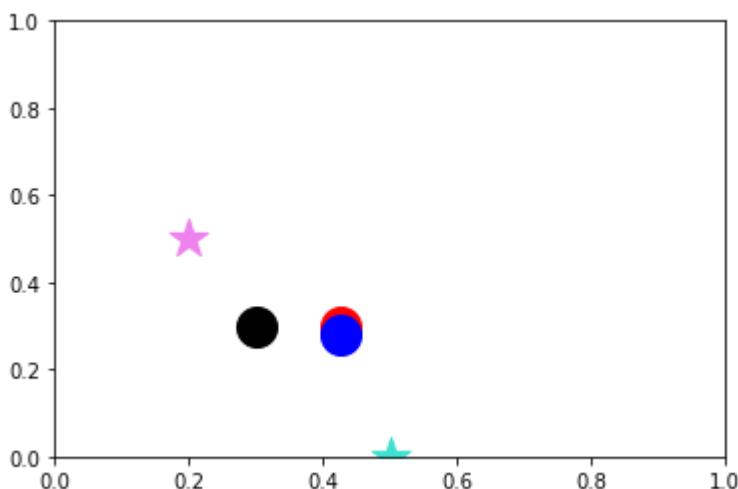
```
182 R2.alphay = 0.3 # 0.1  
183 R2.betay = 0.7 # 0.9
```

blue
sigh
blue
brain

graph

In [98]:

```
1 x = R1.betax
2 y = R1.betay
3 #plt.plot(x,y, 'o', c = 'black');
4 plt.scatter(x,y, s = 400, c = 'black')
5
6 x = R2.betax
7 y = R2.betay
8 plt.scatter(x, y, s = 400, c = 'red')
9
10 x = R3.betax
11 y = R3.betay
12 plt.scatter(x, y, s = 400, c = 'blue')
13
14 x = T.x
15 y = T.y
16 plt.scatter(x, y, s = 400, marker = '*', c = 'turquoise')
17
18 x = T2.x
19 y = T2.y
20 plt.scatter(x, y, s = 400, marker = '*', c = 'violet')
21
22 plt.axis([0, 1, 0, 1])
23 plt.show()
24 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 violet

In [99]:

```
1 # former rewards
2
3 R1.delta, R2.delta, R3.delta
```

Out[99]:

(0.69, 0.69, 0.69)

In [100]:

```
1 # new rewards
2
3 R1.delta = reward(T2, R1.betax, R1.betay)
4 print(R1.delta)
5
6 R2.delta = reward(T2, R2.betax, R2.betay)
7 print(R2.delta)
8
9 R3.delta = reward(T2, R3.betax, R3.betay)
10 print(R3.delta)
```

0.78
0.7
0.69

reach the most successful robot

In [101]:

```

1 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
6         R2.alphax = round(1 - R2.betax, 3)
7         R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
8         R2.alphay = round(1 - R2.betay, 3)
9         R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
10        R2.alphaz = round(1 - R2.betaz, 3)
11        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
12        R3.alphax = round(1 - R2.betax, 3)
13        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
14        R3.alphay = round(1 - R2.betay, 3)
15        R3.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
16        R3.alphaz = round(1 - R2.betaz, 3)
17    if (R2.delta > R1.delta and R2.delta > R3.delta):
18        print('quagga')
19        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
20        R1.alphax = round(1 - R1.betax, 3)
21        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
22        R1.alphay = round(1 - R1.betay, 3)
23        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
24        R1.alphaz = round(1 - R1.betaz, 3)
25        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
26        R3.alphax = round(1 - R3.betax, 3)
27        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
28        R3.alphay = round(1 - R3.betay, 3)
29        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
30        R3.alphaz = round(1 - R3.betaz, 3)
31    if (R3.delta > R1.delta and R3.delta > R2.delta):
32        print('quark')
33        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
34        R1.alphax = round(1 - R1.betax, 3)
35        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
36        R1.alphay = round(1 - R1.betay, 3)
37        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
38        R1.alphaz = round(1 - R1.betaz, 3)
39        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
40        R2.alphax = round(1 - R2.betax, 3)
41        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
42        R2.alphay = round(1 - R2.betay, 3)
43        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
44        R2.alphaz = round(1 - R2.betaz, 3)
45
46 R1.delta = reward(T2, R1.betax, R1.betay, R1.betaz)
47 print(R1.delta)
48
49 R2.delta = reward(T2, R2.betax, R2.betay, R2.betaz)
50 print(R2.delta)
51
52 R3.delta = reward(T2, R3.betax, R3.betay, R3.betaz)
53 print(R2.delta)

```

0.7
0.7
0.69

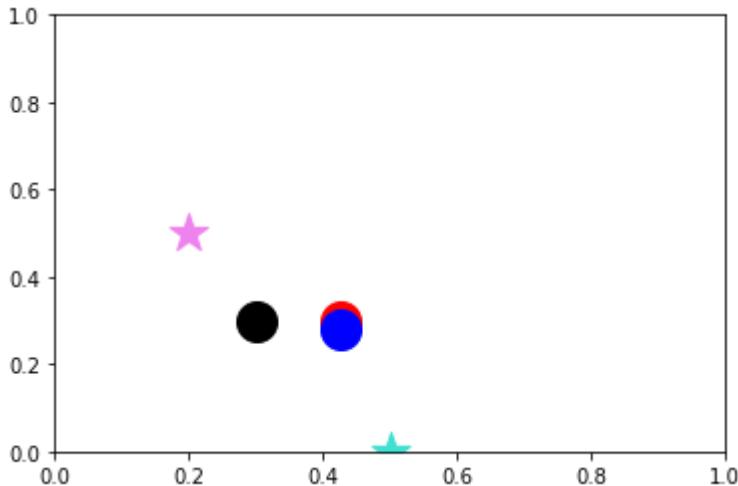
In []:

1

plot

In [102]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```
1 # Another round of SOS with a higher threshold. Added on January 28
2
3 # threshold for initial reward
4 # random fluctuations
5
6 if (R1.delta <= 0.6) and (R2.delta <= 0.6) and (R3.delta <= 0.6):
7     print("SOS")
8     # R1
9     R1.alphax = round(np.random.uniform(0,0.9), 3)
10    R1.betax = round(1 - R1.alphax, 3)
11    print("the new x-positions for R1 are: ", R1.alphax, R1.betax)
12    R1.alphay = round(np.random.uniform(0,0.9), 3)
13    R1.betay = round(1 - R1.alphay, 3)
14    print("the new y-positions for R1 are: ", R1.alphay, R1.betay)
15    # R2
16    R2.alphax = round(np.random.uniform(0,0.9), 3)
17    R2.betax = round(1 - R2.alphax, 3)
18    print("the new x-positions for R2 are: ", R2.alphax, R2.betax)
19    R2.alphay = round(np.random.uniform(0,0.9), 3)
20    R2.betay = round(1 - R2.alphay, 3)
21    print("the new y-positions for R2 are: ", R2.alphay, R2.betay)
22    # R3
23    R3.alphax = round(np.random.uniform(0,0.9), 3)
24    R3.betax = round(1 - R3.alphax, 3)
25    print("the new x-positions for R3 are: ", R3.alphax, R3.betax)
26    R3.alphay = round(np.random.uniform(0,0.9), 3)
27    R3.betay = round(1 - R3.alphay, 3)
28    print("the new y-positions for R3 are: ", R3.alphay, R3.betay)
29
30 R1.delta = reward(T2, R1.betax, R1.betay)
31 R1.gamma = 1 - R1.delta
32 R2.delta = reward(T2, R2.betax, R2.betay)
33 R2.gamma = 1 - R2.delta
34 R3.delta = reward(T2, R3.betax, R3.betay)
35 R3.gamma = 1 - R3.delta
36 print(R1.delta, R2.delta, R3.delta)
```

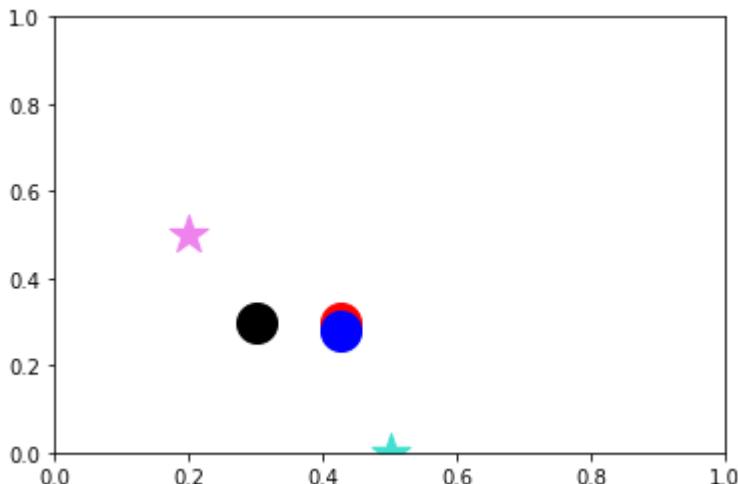
0.78 0.7 0.69

In [104]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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

reach the most successful robot:

In [105]:

```
1 if (R1.delta > R2.delta) and (R1.delta > R3.delta): # now, brutally come close to
2     print("ciao")
3     # change R2, R3 but not R1
4     # R2
5     R2.alphax = round(R1.alphax + np.random.uniform(0,0.1), 3)
6     R2.betax = round(1 - R2.alphax, 3)
7     R2.alphay = round(R1.alphay + np.random.uniform(0,0.1), 3)
8     R2.betay = round(1 - R2.alphay, 3)
9     # R3
10    R3.alphax = round(R1.alphax + np.random.uniform(0,0.1), 3)
11    R3.betax = round(1 - R3.alphax, 3)
12    R3.alphay = round(R1.alphay + np.random.uniform(0,0.1), 3)
13    R3.betay = round(1 - R3.alphay, 3)
14 elif (R2.delta > R3.delta) and (R2.delta > R3.delta):
15     print("glu glu")
16     # change R1, R3 but not R2
17     # R1
18     R1.alphax = round(R2.alphax + np.random.uniform(0,0.1), 3)
19     R1.betax = round(1 - R1.alphax, 3)
20     R1.alphay = round(R2.alphay + np.random.uniform(0,0.1), 3)
21     R1.betay = round(1 - R1.alphay, 3)
22     # R3
23     R3.alphax = round(R2.alphax + np.random.uniform(0,0.1), 3)
24     R3.betax = round(1 - R2.alphax, 3)
25     R3.alphay = round(R2.alphay + np.random.uniform(0,0.1), 3)
26     R3.betay = round(1 - R2.alphay, 3)
27 elif (R3.delta > R1.delta) and (R3.delta > R2.delta):
28     print("cri cri")
29     # change R1, R2 but nor R3
30     # R1
31     R1.alphax = round(R3.alphax + np.random.uniform(0,0.1), 3)
32     R1.betax = round(1 - R1.alphax, 3)
33     R1.alphay = round(R3.alphay + np.random.uniform(0,0.1), 3)
34     R1.betay = round(1 - R1.alphay, 3)
35     # R2
36     R2.alphax = round(R3.alphax + np.random.uniform(0,0.1), 3)
37     R2.betax = round(1 - R2.alphax, 3)
38     R2.alphay = round(R3.alphay + np.random.uniform(0,0.1), 3)
39     R2.betay = round(1 - R2.alphay, 3)
```

ciao

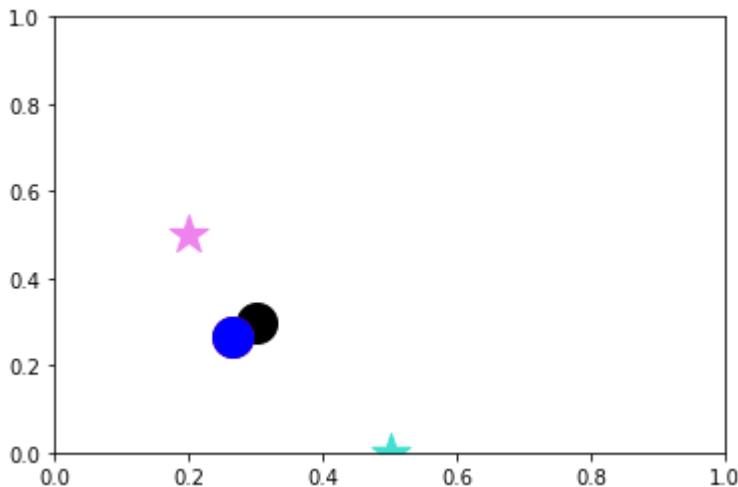
plot and rewards

In [106]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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
 et
 0.78
 0.76
 0.76

check and move + rewards

In [107]:

```

1 if((R1.delta >= 0.8) or (R2.delta >= 0.8) or (R3.delta >= 0.8)):
2     print('yuk')
3     if (R1.delta > R2.delta and R1.delta > R3.delta):
4         print('quokka')
5         R2.betax = round(R1.betax + np.random.uniform(0,0.1), 3) # Here and later
6         R2.alphax = round(1 - R2.betax, 3)
7         R2.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
8         R2.alphay = round(1 - R2.betay, 3)
9         R2.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
10        R2.alphaz = round(1 - R2.betaz, 3)
11        R3.betax = round(R1.betax + np.random.uniform(0,0.1), 3)
12        R3.alphax = round(1 - R2.betax, 3)
13        R3.betay = round(R1.betay + np.random.uniform(0,0.1), 3)
14        R3.alphay = round(1 - R2.betay, 3)
15        R3.betaz = round(R1.betaz + np.random.uniform(0,0.1), 3)
16        R3.alphaz = round(1 - R2.betaz, 3)
17    if (R2.delta > R1.delta and R2.delta > R3.delta):
18        print('quagga')
19        R1.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
20        R1.alphax = round(1 - R1.betax, 3)
21        R1.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
22        R1.alphay = round(1 - R1.betay, 3)
23        R1.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
24        R1.alphaz = round(1 - R1.betaz, 3)
25        R3.betax = round(R2.betax + np.random.uniform(0,0.1), 3)
26        R3.alphax = round(1 - R3.betax, 3)
27        R3.betay = round(R2.betay + np.random.uniform(0,0.1), 3)
28        R3.alphay = round(1 - R3.betay, 3)
29        R3.betaz = round(R2.betaz + np.random.uniform(0,0.1), 3)
30        R3.alphaz = round(1 - R3.betaz, 3)
31    if (R3.delta > R1.delta and R3.delta > R2.delta):
32        print('quark')
33        R1.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
34        R1.alphax = round(1 - R1.betax, 3)
35        R1.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
36        R1.alphay = round(1 - R1.betay, 3)
37        R1.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
38        R1.alphaz = round(1 - R1.betaz, 3)
39        R2.betax = round(R3.betax + np.random.uniform(0,0.1), 3)
40        R2.alphax = round(1 - R2.betax, 3)
41        R2.betay = round(R3.betay + np.random.uniform(0,0.1), 3)
42        R2.alphay = round(1 - R2.betay, 3)
43        R2.betaz = round(R3.betaz + np.random.uniform(0,0.1), 3)
44        R2.alphaz = round(1 - R2.betaz, 3)
45
46 R1.delta = reward(T2, R1.betax, R1.betay, R1.betaz)
47 print(R1.delta)
48
49 R2.delta = reward(T2, R2.betax, R2.betay, R2.betaz)
50 print(R2.delta)
51
52 R3.delta = reward(T2, R3.betax, R3.betay, R3.betaz)
53 print(R2.delta)

```

0.76
0.76
0.76

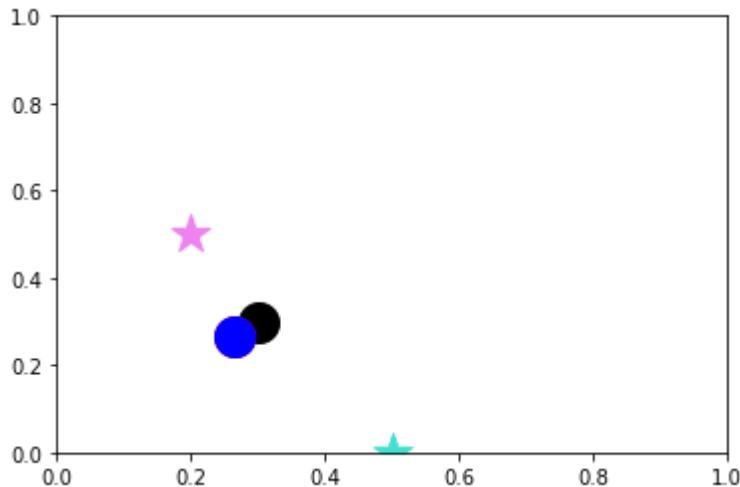
plot

In [108]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```
1 # a new circuit
2 q = QuantumRegister(9, 'q'); # qubits
3 #c0 = ClassicalRegister(6, 'c0');
4 c0 = ClassicalRegister(1, 'c0');
5 c1 = ClassicalRegister(1, 'c1');
6 c3 = ClassicalRegister(1, 'c3');
7 c4 = ClassicalRegister(1, 'c4');
8 c6 = ClassicalRegister(1, 'c6');
9 c7 = ClassicalRegister(1, 'c7');
10 qc_small = QuantumCircuit(q, c0, c1, c3, c4, c6, c7);
```

In [118]:

```

1
2
3 if ((R3.delta - R2.delta) <= 0.3) and ((R3.delta - R1.delta) <= 0.3) and ((R2.de
4   # 0.3 rather than 0.1
5   print("cometa")# GHZ for all
6   qc_small.h(q[0]) # Hadamard
7   qc_small.cx(q[0], q[1]) # CNOT
8   qc_small.cx(q[0], q[2]) # CNOT
9   qc_small.cx(q[0], q[3]) # CNOT
10  qc_small.cx(q[0], q[4]) # CNOT
11  qc_small.cx(q[0], q[5]) # CNOT
12  qc_small.cx(q[0], q[6]) # CNOT
13  qc_small.cx(q[0], q[7]) # CNOT
14  # barrier
15  qc_small.barrier(q[0], q[1], q[2], q[3], q[4], q[5], q[6], q[7])
16  # measures
17  qc_small.measure(q[0],c0[0])
18  qc_small.measure(q[3],c3[0])
19  qc_small.measure(q[6],c6[0])
20  qc_small.measure(q[1],c1[0])
21  qc_small.measure(q[4],c4[0])
22  qc_small.measure(q[7],c7[0])
23  # draw circuit
24  draw_circuit(qc_small)
25  # definition of quantum simulator
26  simulator = Aer.get_backend('qasm_simulator') # statevector_simulator # aer_
27  qc_small = transpile(qc_small, simulator)
28  # Run and get counts
29  result = simulator.run(qc_small, shots=1024).result()
30  counts_GHZ_all = result.get_counts(qc_small)
31  counts_GHZ = counts_GHZ_all.most_frequent() # does not work if multiple stat
32  # decide something if multiple states have the same count --> e.g., ``choose
33  print(counts_GHZ_all)
34  print(counts_GHZ)
35  #plot_histogram(counts_GHZ_all, title='outcomes')
36  #plot_histogram(counts_GHZ, title='outcomes')
37
38 print(counts_GHZ) # order: R3, R2, R1. Add some uncertainty?
39 # export as an array
40 str_ = counts_GHZ
41 arr_GHZ = str_.split(' ') # to split the string and avoid empty spaces as array
42 print(arr_GHZ)
43 # We do not need to update rewards; they should be done externally... excluded s

```

cometa

CircuitComposer(circuit=<qiskit.circuit.quantumcircuit.QuantumCircuit object at 0x7fb65849a8e0>, editable=False)

```

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

```

In [119]:

```
1 # define the new 0:
2
3
4 if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):
5     print('gosh')
6     new_zero_betax = R1.betax + np.random.uniform(0,0.1)
7     new_zero_alphax = 1 - R1.betax
8     new_zero_betay = R1.betay + np.random.uniform(0,0.1)
9     new_zero_alphaay = 1 - R2.betay + np.random.uniform(0,0.1)
10
11 if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):
12     print('kinda')
13     new_zero_betax = R2.betax + np.random.uniform(0,0.1)
14     new_zero_alphax = 1 - R2.betax
15     new_zero_betay = R2.betay + np.random.uniform(0,0.1)
16     new_zero_alphaay = 1 - R2.betay
17
18 if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):
19     print('uffdah')
20     new_zero_betax = R3.betax + np.random.uniform(0,0.1)
21     new_zero_alphax = 1 - R3.betax
22     new_zero_betay = R3.betay + np.random.uniform(0,0.1)
23     new_zero_alphaay = 1 - R3.betay
```

gosh
kinda
uffdah

In [120]:

```
1 # define the new 1:  
2  
3  
4 # flip thanks to the 'minus' sign?  
5 # I had tried with if(R1... < R2...) etc., but it is not ok,  
6 # because we need to initialize all elements.  
7  
8 if (R1.delta >= R2.delta) and (R1.delta >= R3.delta):  
9     print('gosh')  
10    new_one_betax = R1.betax - np.random.uniform(0,0.1)  
11    new_one_alphax = 1 - R1.betax  
12    new_one_betay = R1.betay - np.random.uniform(0,0.1)  
13    new_one_alphaay = 1 - R2.betay  
14  
15 if (R2.delta >= R1.delta) and (R2.delta >= R3.delta):  
16     print('kinda')  
17     new_one_betax = R2.betax - np.random.uniform(0,0.1)  
18     new_one_alphax = 1 - R2.betax  
19     new_one_betay = R2.betay - np.random.uniform(0,0.1)  
20     new_one_alphaay = 1 - R2.betay  
21  
22 if (R3.delta >= R2.delta) and (R3.delta >= R1.delta):  
23     print('uffdah')  
24     new_one_betax = R3.betax - np.random.uniform(0,0.1)  
25     new_one_alphax = 1 - R3.betax  
26     new_one_betay = R3.betay - np.random.uniform(0,0.1)  
27     new_one_alphaay = 1 - R3.betay
```

gosh
kinda
uffdah

In [121]:

```

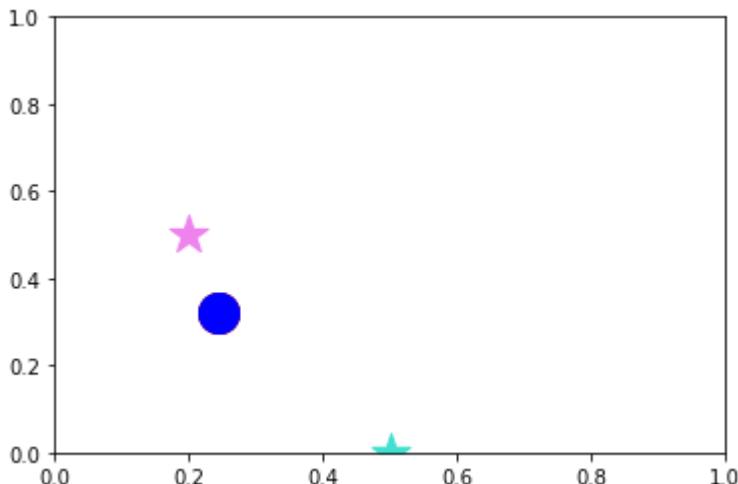
1 # choose the new 0 or new 1 according to the outcome of GHZ:
2
3 if (arr_GHZ[0] =='0'):
4     R1.alphax = new_zero_alphax
5     R1.betax = new_zero_betax
6 if (arr_GHZ[1] =='0'):
7     R1.alphay = new_zero_alphahay
8     R1.betay = new_zero_betay
9 if (arr_GHZ[2] =='0'):
10    R2.alphax = new_zero_alphax
11    R2.betax = new_zero_betax
12 if (arr_GHZ[3] =='0'):
13    R2.alphay = new_zero_alphahay
14    R2.betay = new_zero_betay
15 if (arr_GHZ[4] =='0'):
16    R2.alphax = new_zero_alphax
17    R2.betax = new_zero_betax
18 if (arr_GHZ[5] =='0'):
19    R3.alphax = new_zero_alphax
20    R3.betax = new_zero_betax
21 if (arr_GHZ[0] =='1'):
22    R1.alphax = new_one_alphax
23    R1.betax = new_one_betax
24 if (arr_GHZ[1] =='1'):
25    R1.alphay = new_one_alphahay
26    R1.betay = new_one_betay
27 if (arr_GHZ[2] =='1'):
28    R2.alphax = new_one_alphax
29    R2.betax = new_one_betax
30 if (arr_GHZ[3] =='1'):
31    R2.alphay = new_one_alphahay
32    R2.betay = new_one_betay
33 if (arr_GHZ[4] =='1'):
34    R2.alphax = new_one_alphax
35    R2.betax = new_one_betax
36 if (arr_GHZ[5] =='1'):
37    R3.alphax = new_one_alphax
38    R3.betax = new_one_betax
39 if (arr_GHZ[0] =='1'):
40    R3.alphay = new_one_alphahay
41    R3.betay = new_one_betay
42 if (arr_GHZ[1] =='1'):
43    R3.alphay = new_one_alphahay
44    R3.betay = new_one_betay
45
46
47 # new rewards:
48
49 R1.delta = reward(T2, R1.betax, R2.betay)
50 R2.delta = reward(T2, R2.betax, R3.betay)
51 R3.delta = reward(T2, R2.betax, R3.betay)
52 print(R1.delta, R2.delta, R3.delta)

```

0.81 0.81 0.81

In [122]:

```
1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # 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]:

```

f(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.betaz + 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)
print(R1.delta)
48
delta = reward(T2, R2.betax, R2.betay, R2.betaz)
print(R2.delta)
51
delta = reward(T2, R3.betax, R3.betay, R3.betaz)
print(R2.delta)

```

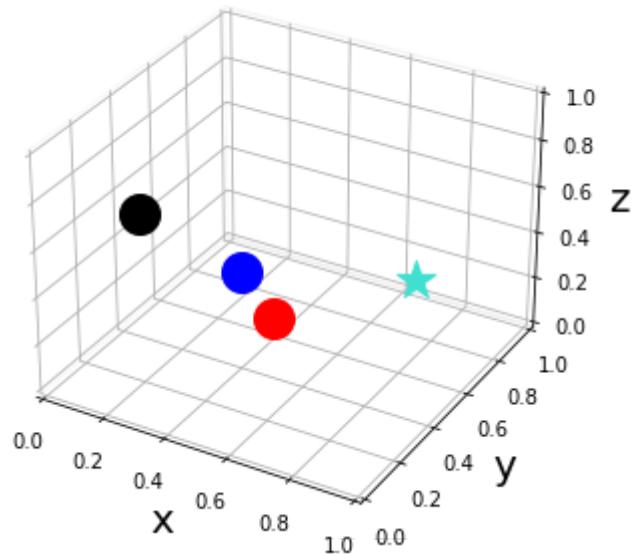
yuk
0.81
0.81
0.81

In [57]:

```

1 fig = plt.figure()
2
3 ax = Axes3D(fig, auto_add_to_figure=False)
4 fig.add_axes(ax)
5
6 ax.set_xlim3d(0, 1)
7 ax.set_ylim3d(0, 1)
8 ax.set_zlim3d(0, 1)
9
10 ax.xaxis.pane.fill = False
11 ax.yaxis.pane.fill = False
12 ax.zaxis.pane.fill = False
13
14 ax.set_xlabel('x', fontsize=20)
15 ax.set_ylabel('y', fontsize=20)
16 ax.set_zlabel('z', fontsize=20) # r'\alpha'
17
18 ax.scatter3D(R1.betax, R1.betay, R1.betaz, s = 400, marker = 'o', color = 'black')
19 ax.scatter3D(R2.betax, R2.betay, R2.betaz, s = 400, marker = 'o', color = 'red')
20 ax.scatter3D(R3.betax, R3.betay, R3.betaz, s = 400, marker = 'o', color = 'blue')
21 # ax.scatter3D(R4_[0], R4_[1], R4_[2], s = 400, marker = 'o', color = 'green')
22 ax.scatter3D(T.x, T.y, T.z, s = 400, marker = '*', color = 'turquoise')
23
24 plt.show()
25
26 # find how to automatically create trajectories: maybe LinePlot between R1, R2,

```



In []:

1

In []:

1

In []:

1	
---	--

In []:

1	
---	--